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

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάλυση Απαιτήσεων Στη Διοίκηση Έργων Ανάπτυξης Πληροφοριακών Συστημάτων

ΖΗΣΗΣ ΔΗΜΗΤΡΗΣ

ΠΕΙΡΑΙΑΣ 2013

ΚΑΘΗΓΗΤΡΙΑ: κα ΜΑΛΑΜΑΤΕΝΙΟΥ ΦΛΩΡΑ

Περιεχόμενα

1 Μηχανική Απαιτήσεων 11
1.1 Εισαγωγή 11
1.2 Τι είναι η Μηχανική Απαιτήσεων; 12
1.3 Η Σημασία των Απαιτήσεων 13
1.4 Προσεγγίσεις στη Μηχανική Απαιτήσεων σε Έργα Ανάπτυξης Πληροφοριακών Συστημάτων 14
1.4.1 Η Ορθόδοξη Προσέγγιση 14
1.4.2 Η Προσέγγιση Επίλυσης Προβλήματος όσον αφορά στη Μηχανική Απαιτήσεων 17
1.4.3 Η Προοπτική των Τριών Διαστάσεων 20
1.5 Σύγκριση των Προσεγγίσεων 25
1.6 Ο Αναλυτής 26
2 Μηχανική Απαιτήσεων και Τεχνολογία Λογισμικού 28
2.1 Εισαγωγή 28
2.2 Διασφάλιση Ποιότητας Λογισμικού (Software Quality Assurance) 30
2.3 Η Τεχνολογία Λογισμικού ως Διαδικασία 33
2.4 Η Μηχανική Απαιτήσεων στον Κύκλο Ζωής του Έργου 36
2.4.1 Το Μοντέλο του Καταρράκτη 36
2.4.2 Το Αυξητικό Μοντέλο 40
2.4.3 Το Σπειροειδές Μοντέλο 41
2.4.4 Το μοντέλο του Πίδακα 42
2.4.5 Το Εξελικτικό Μοντέλο RUP 44
2.4.6 Κωδικοποίηση και διόρθωση (code and fix) 45
2.4.7 Ταχεία ανάπτυξη λογισμικού (rapid application development) 46
2.5 Παράδειγμα Ανάλυσης Απαιτήσεων σε έργο πληροφορικής 47
3 Εξόρυξη Απαιτήσεων 51
3.1 Εισαγωγή 51
3.2 Εκκίνηση της διαδικασίας Εξόρυξης Απαιτήσεων 51
3.3 Προσδιορισμός των ενδιαφερόμενων μερών 53
3.4. Τεχνικές Εξόρυξης Απαιτήσεων 54
3.4.1 Συνεντεύξεις (Interviews) 54
3.4.2 Ερωτηματολόγια (Questionnaries) 55
3.4.3 Συναντήσεις Συγκέντρωσης (Focus Groups) 55
3.4.4 Άμεση Παρατήρηση (Direct Observation) 56
3.4.5 Άμεση Εκτέλεση (Apprenticing) 56
3.4.6 Διαλογισμός (Brainstorming) 56
3.4.7 Επιθεώρηση εγγράφων (Inspecting Documents) 57
3.4.8 Κοινή Ανάπτυξη Εφαρμογών (JAD) 57
3.4.9 Ενδοσκόπηση (Introspection) 58

3.4.10 Ταξινόμηση καρτών (card sorting) ... 58
3.4.11 Προτυποποίηση (prototyping) ... 59
4 Περιπτώσεις Χρήσης (Use Cases) ... 60
4.1 Εισαγωγή ... 60
4.1.1 Ονόματα (Names) ... 65
4.1.2 Γενίκευση μεταξύ ρόλων ... 66
4.1.3 Γενίκευση (Generalization) ... 66
4.1.4 Συμπερίληψη-Συνυπολογισμός (Inclusion) ... 67
4.1.5 Επέκταση (Extension) ... 68
4.2 Παραδείγματα Εκμαίευσης Απαιτήσεων με τη χρήση Περιπτώσεων Χρήσης ... 68
4.2.1 Παράδειγμα «ATM» ... 68
4.2.2 Παράδειγμα Συστήματος Σημείου Πώλησης ... 70
4.2.3 Παράδειγμα Συστήματος Κράτησης Θέσεων ... 72
4.3 Ολοκλήρωση Περιπτώσεων Χρήσης ... 73
4.3.1 Πρόσοψη ... 74
4.3.2 Η περίπτωση χρήσης Γέμισης ... 75
4.3.3 Η Επανάληψη Συγκέντρωσης ... 77
4.3.4 Η Επανάληψη Ολοκλήρωσης ... 79
5 Ανάλυση και Μοντελοποίηση με τη γλώσσα UML ... 81
5.1 Εισαγωγή ... 81
5.2 Μοντελοποίηση ... 81
5.3 UML (Unified Modelling Language): ... 84
5.3.1 Διαγράμματα Κλάσεων (Class Diagrams) ... 85
5.4 Το παράδειγμα με τα ATM ... 88
5.5 Το Παράδειγμα με τον Εξομοιωτή Χειρουργικής Επέμβασης ... 95
5.6 Διαγράμματα Αλληλεπίδρασης ... 104
5.6.1 Βασικές Χρήσεις των Διαγραμμάτων Αλληλεπίδρασης ... 107
5.7 Ακολουθιακά Διαγράμματα ... 107
5.8 Διαγράμματα Κατάστασης ... 109
5.9 Διαγράμματα Δραστηριότητας ... 110
6 Προδιαγραφή και Επικύρωση Απαιτήσεων ... 112
6.1 Εισαγωγή ... 112
6.2 Ο σκοπός της προδιαγραφής απαιτήσεων ... 112
6.2.1 Περίγραμμα του Εγγράφου Προδιαγραφής Απαιτήσεων (SRS) ... 113
6.2.2 Ιδιότητες ενός καλού Εγγράφου Προδιαγραφής Απαιτήσεων (SRS) ... 117
6.2.3 Οδηγίες Προδιαγραφής Απαιτήσεων ... 119
6.2.4 Τυπικές μέθοδοι Προδιαγραφής Απαιτήσεων ... 122
6.3 Επικύρωση Απαιτήσεων ... 122

6.3.1 Αναφορές και Επιθεωρήσεις 123
6.3.2 Μοντελοποίηση και Προτυποποίηση 125
6.3.3 Προοπτικές Προτυποποίησης 126
7 Διαχείριση Απαιτήσεων 130
7.1 Εισαγωγή 130
7.2 Απαιτήσεις και Συναισθήματα 132
7.3 Προγραμματισμός Διαχείρισης Απαιτήσεων 135
7.4 Διαχείριση Αλλαγής Απαιτήσεων 137
7.5 Οφέλη Εφαρμογής Τεχνικών Διαχείρισης Απαιτήσεων 138
7.6 Εργαλεία Διαχείρισης Απαιτήσεων 140
7.7 Προκλήσεις Στη Μηχανική Απαιτήσεων 145
7.8 Ευέλικτες Μεθοδολογίες (Agile) στη Διαχείριση Απαιτήσεων 148
7.9 Κρίσιμοι παράγοντες επιτυχίας (critical success factors) 151
8 Μελέτη Περίπτωσης 153
8.1 Περιγραφή μελέτης περίπτωσης 153
8.2 Χρήστες συστήματος 154
8.3 Προβλήματα λειτουργίας συστήματος ηλεκτρονικής ψηφοφορίας 154
8.4 Λειτουργικές απαιτήσεις 155
8.5 Μη Λειτουργικές Απαιτήσεις 156
8.6 Απαιτήσεις Ασφάλειας 157
8.7 Μοντέλο Οντοτήτων Σχέσεων 158
8.8 Διάγραμμα κλάσεων 160
8.9 Περιπτώσεις Χρήσης 164
9 Συμπεράσματα– Προοπτικές 168
9.1 Συμπεράσματα της έρευνας 168
9.2 Περιορισμοί στη μελέτη περίπτωσης 169
9.3 Μελλοντικές επεκτάσεις 171
9.4 Προτάσεις για επόμενες έρευνες 171
Βιβλιογραφία 173

Περιεχόμενα Σχημάτων

Σχήμα 1.1: Οι σχέσεις κόστους διόρθωσης λαθών (εντοπισμός και διόρθωση) για κάθε μία από τις φάσεις του έργου 14

Σχήμα 1.2: Καθορισμός του εύρους των απαιτήσεων-λύσεων 18

Σχήμα 1.3: Οι διαστάσεις της προοπτικής τριών διαστάσεων 21

Σχήμα 2.1: Σκοπός ελέγχου σε έργο ανάπτυξης λογισμικού σε κάθε στάδιο ανάπτυξης 29

Σχήμα 2.2: Σχηματική απεικόνιση μίας τυπικής φάσης, έργου ανάπτυξης πληροφοριακού συστήματος. 35

Σχήμα 2.3: Πιθανή έκδοση φάσης μοντέλου καταρράκτη 36

Σχήμα 2.4 Το Μοντέλο του Καταρράκτη με δυνατότητα ανατροφοδότησης πληροφοριών των φάσεων μεταξύ τους (19) 38

Σχήμα 2.5: Το Μοντέλο του Καταρράκτη 39

Σχήμα 2.6: Το Μοντέλο "V" για έργα ανάπτυξης λογισμικού 40

Σχήμα 2.7: Το Αυξητικό Μοντέλο, έργων ανάπτυξης λογισμικού (19) 41

Σχήμα 2.8: Το Σπειροειδές Μοντέλο (19) 42

Σχήμα 2.10: Το Εξελικτικό Μοντέλο RUP (22) 45

Σχήμα 2.11: Μοντέλο ταχείας ανάπτυξης λογισμικού 46

Σχήμα 4.1 Παράδειγμα Περίπτωσης Χρήσης «Δανειστής βιβλίων» 60

Σχήμα 4.2 Παράδειγμα Περίπτωσης Χρήσης «Δανειστής βιβλίων» χωρίς απαίτηση ειδικής άδειας δανεισμού 62

Σχήμα 4.3 Παράδειγμα Περίπτωσης Χρήσης «Δανειστής βιβλίων» με απαίτηση ειδικής άδειας δανεισμού 63

Σχήμα 4.4 Περίγραμμα προσδιορισμού περιπτώσεων χρήσης (flow of events diagram) 64

Σχήμα 4.5: Μία πιθανή περίπτωση χρήσης για ένα απλό σύστημα βιβλιοθήκης 65

Σχήμα 4.7: Δόμηση περιπτώσεων χρήσης 67

Σχήμα 4.8: Δόμηση Περιπτώσεων χρήσης με γενίκευση 67

Σχήμα 4.9 Διάγραμμα περίπτωσης χρήσης συστήματος τραπεζών 70

Σχήμα 4.10: Διάγραμμα περίπτωσης χρήσης απομακρυσμένου σημείου πώλησης 72

Σχήμα 4.11: Περίπτωση χρήσης online συστήματος κράτησης θέσεων 73

Σχήμα 4.10. Περίπτωση Χρήσης Πρόσοψης για την παρακολούθηση πωλήσεων ένδυσης 76

Σχήμα 4.11: Μία περίπτωση χρήσης γέμισης, σχετική με δάνειο 77

Σχήμα 4.12: Πώληση ενός ακινήτου. Περίπτωση χρήσης συγκέντρωσης. Συμφωνία των όρων 80

Σχήμα 5.1: Χρήση Μοντέλων Ανάλυσης για ανάδραση στη διαδικασία Εξόρυξης Απαιτήσεων 81

Σχήμα 5.2: Παράδειγμα Μεθόδου «Petri Nets» 83
Σχήμα 5.3: Γενίκευση 87
Σχήμα 5.4: Σχέση ανάμεσα τις κλάσεις Εργαζόμενος και Εργοδότης 87
Σχήμα 5.5: Σχέση Συσσωμάτωσης. Η εταιρία είναι το «σύνολο» και το τμήμα είναι το «μέρος» του συνόλου 88
Σχήμα 5.6: Το Δίκτυο Τραπεζών με τα ATM 89
Σχήμα 5.7 Περίπτωση Χρήσης Συναλλαγής ATM για το Πρόβλημα Δικτύου Τραπεζών 93
Σχήμα 5.8: Οι ενδεχόμενες κλάσεις του προβλήματος του δικτύου των ATM, σε κατηγορίες 94
Σχήμα 5.9: Οι ενδεχόμενες Κλάσεις του Δικτύου Τραπεζών 94
Σχήμα 5.10 Περιπτώσεις χρήσης εξομοιωτή DPL 96
Σχήμα 5.11: Η Περίπτωση Χρήσης «Διενέργεια DPL» 99
Σχήμα 5.12. Αρχικό μοντέλο για την DPL 100
Σχήμα 5.13: Το Μοντέλο των Αντικειμένων της περίπτωσης, πιο ολοκληρωμένο 102
Σχήμα 5.14: Σχετικό κόστος διορθώσεων σε ένα λογισμικό ανάλογα το στάδιο ανάπτυξης (45) 104
Σχήμα 5.15: Ένα Διάγραμμα Αλληλεπίδρασης για το παράδειγμα με τα ATM. 105
Σχήμα 5.16 Ένα Διάγραμμα Συνεργασίας για το Παράδειγμα με τα ATM 106
Σχήμα 5.17: Ένα Ακολουθιακό Διάγραμμα για την εκτέλεση τομής του Παραδείγματος «Εξομοιωτής Χειρουργικής Επέμβασης» 108
Σχήμα 5.18: Ένα Ακολουθιακό Διάγραμμα για την Εισαγωγή του Σωληνίσκου 109
Σχήμα 5.19: Ένα Διάγραμμα Δραστηριότητας περίπτωσης κατασκευής 110
Σχήμα 6.1: Περίγραμμα Εγγράφου Προδιαγραφής Απαιτήσεων (SRS). Από το Πρότυπο IEEE Std. 830-1998 114
Σχήμα 6.2: Η 3η ενότητα του Εγγράφου Προδιαγραφής Απαιτήσεων (SRS), δομημένο σύμφωνα με τις Κλάσεις Απαιτήσεων. Από το Πρότυπο IEEE Std. 830-1998 114
Σχήμα 6.3: Η 3η Ενότητα του Εγγράφου Προδιαγραφής Απαιτήσεων δομημένη σύμφωνα με τις Λειτουργίες του Συστήματος. Από το Πρότυπο IEEE Std. 830-1998 115
Σχήμα 6.4: Η 3η Ενότητα του Εγγράφου Προδιαγραφής Απαιτήσεων δομημένο σύμφωνα με τα Αντικείμενα. Από το Πρότυπο IEEE Std. 830-1998 116
Σχήμα 6.5: Η 3η Ενότητα του Εγγράφου Προδιαγραφής Απαιτήσεων (SRS) δομημένο σύμφωνα με τις Περιπτώσεις Χρήσης. Από το Πρότυπο IEEE Std. 830-1998 117
Σχήμα 6.6: Λίστα βασικών τυπικών μεθόδων. 122
Σχήμα 6.7: Η μέθοδος προτυποποίησης throw-away 127
Σχήμα 7.1: Ανατροφοδότηση απαιτήσεων στον κύκλο ζωής ενός λογισμικού 131

Σχήμα 7.2: Η εξέλιξη των απαιτήσεων κατά την Ανάπτυξη Λογισμικού 136
Σχήμα 7.3: Λόγοι Αποτυχίας Έργων Πληροφορικής 145
Σχήμα 7.4: Διαχείριση απαιτήσεων με τη χρήση ευέλικτων μεθοδολογιών στην ανάπτυξη λογισμικού 150
Σχήμα 8.1: Μοντέλο Οντοτήτων Συσχετίσεων - μελέτη περίπτωσης 159
Σχήμα 8.2: Διάγραμμα κλάσεων - μελέτη περίπτωσης 163
Σχήμα 8.3: Περίπτωση Χρήσης - Ψηφοφορία 166
Σχήμα 8.4: Περίπτωση Χρήσης – Ρύθμιση Ψηφοφορίας 167

Περιεχόμενα Πινάκων

Πίνακας 5.1 Πιθανά Αντικείμενα για το Σενάριο DPL 97
Πίνακας 5.2 Ανάθεση ευθυνών στα αντικείμενα ως μέρος της διαδικασίας της ανάλυσης 101

Περίληψη

Η Ανάλυση Απαιτήσεων είναι το πρώτο στάδιο στην ανάπτυξη ενός έργου πληροφορικής και αποτελεί κρίσιμο παράγοντα για την επιτυχία του . Σύμφωνα με τον Laplante (1) η Ανάλυση Απαιτήσεων «*είναι ένας κλάδος της μηχανικής λογισμικού που ασχολείται με τον καθορισμό των στόχων, των λειτουργιών, και των περιορισμών των συστημάτων υλικού και λογισμικού.*»

Βέβαια, η Ανάλυση Απαιτήσεων δεν περιορίζεται στην αρχή της ανάπτυξης ενός λογισμικού, μα συνεχίζει τη δράση της σε όλα τα στάδια της ανάπτυξης ενός λογισμικού. Σκοπός της είναι να διασφαλίσει πως «*η φωνή του πελάτη*» ακούγεται σε όλα τα σημεία της διαδικασίας ανάπτυξης, από την αρχική σύλληψη του συστήματος, τη διάρκεια του σχεδιασμού, της υλοποίησης, του ελέγχου, της συντήρησης και της μελλοντικής εξέλιξης του συστήματος.

Στα πλαίσια της παρούσας έρευνας γίνεται μια βιβλιογραφική ανασκόπηση της Ανάλυσης Απαιτήσεων, των προσεγγίσεων της Μηχανικής Απαιτήσεων, του ρόλου της Ανάλυσης Απαιτήσεων στον κύκλο ζωής ενός λογισμικού και περιγράφονται οι βασικές διαστάσεις – στάδια της Μηχανικής Απαιτήσεων.

Στη συνέχεια, μελετάται ο τρόπος εφαρμογής των μοντέλων – σταδίων της Ανάλυσης Απαιτήσεων που περιγράφηκαν σε μία περίπτωση ανάπτυξης συστήματος ηλεκτρονικής ψηφοφορίας και αναφέρονται τρόποι που ο κλάδος της Ανάλυσης Απαιτήσεων μπορεί να βοηθήσει στην αποτελεσματικότερη ανάπτυξη του έργου Πληροφορικής που μελετήθηκε.

Σκοπός της μελέτης περίπτωσης είναι ο εντοπισμός του κέρδους από την εφαρμογή τεχνικών - μεθόδων Ανάλυσης Απαιτήσεων στην ανάπτυξη ενός έργου Πληροφορικής.

Τέλος, παρουσιάζονται τα συμπεράσματα της παρούσας έρευνας, περιορισμοί στην έρευνα και προτάσεις για πιθανές μελλοντικές έρευνες – επεκτάσεις της παρούσας, στον τομέα της Ανάλυσης Απαιτήσεων.

Λέξεις κλειδιά: Ανάλυση Απαιτήσεων, Τεχνολογία Λογισμικού, Εξόρυξη Απαιτήσεων, Μοντελοποίηση, Επικύρωση Απαιτήσεων, Διαχείριση Απαιτήσεων

Abstract

The Requirements Analysis is the first stage in the development of an IT project and a critical factor for project success. According to Laplante (2007) Requirements Analysis is "*a sub discipline of systems engineering and software engineering that is concerned with determining the goals, functions, and constraints of hardware and software systems*".

Requirements Analysis is not limited in the first stages of software development, but it continues the action at all stages of the development cycle of a software. The aim is to ensure that "the voice of the customer" is heard in all parts of the development process, from the initial conception of the system during the design, implementation, testing, maintenance and future system updates.

As part of this research a literature review on Requirements Analysis, on approaches of requirements management and the role of requirements analysis in the lifecycle of software is done. Furthermore, the key dimensions - stages of requirements management are described.

Additionally, the application of models - phases of Requirements Analysis are described for an e-voting system and ways requirements analysis can help in the efficient development of e-voting system are mentioned.

The purpose of the research is to identify the benefits of applying Requirements Analysis techniques in the development of an IT project.

Finally, the results and the conclusions from the survey are presented, the limitations of the research are mentioned and suggestions are made for potential future research in requirements management.

Keywords: Requirements Analysis, Software Engineering, Requirements Elicitation, Modelling, Requirements Evaluation, Requirements Management

1 Μηχανική Απαιτήσεων

1.1 Εισαγωγή

Το επιστημονικό πεδίο της Μηχανικής Λογισμικού εξελίσσεται, διευρύνεται και αναπτύσσεται ραγδαία. Αυτό συμβαίνει διότι, είναι πλέον επιτακτική η ανάγκη, το εκάστοτε λογισμικό και κατ' επέκταση το εκάστοτε πληροφοριακό σύστημα, να αναπτύσσονται, αυστηρά, σύμφωνα με τον οικονομικό προϋπολογισμό του έργου, αξιόπιστα και έγκαιρα. Προκειμένου να εξασφαλιστεί η επιτυχία του έργου, πληρώντας τις παραπάνω προϋποθέσεις, η Μηχανική Λογισμικού υιοθετεί μία συστηματική και πειθαρχημένη προσέγγιση. Μερικές από τις αρχές που διέπουν αυτή την προσέγγιση είναι οι εξής:

1. Ουσιαστική κατανόηση της διαδικασίας ανάπτυξης, της διαχείρισης έργων και ικανότητα ελέγχου και παρακολούθησης της συνολικής διαδικασίας
2. Ουσιαστική κατανόηση του προβλήματος το οποίο πρέπει να επιλυθεί, των μεθόδων σχεδίασης που απαιτούνται για την επίλυσή του και των πλατφορμών που χρησιμοποιήθηκαν για την εφαρμογή της λύσης
3. Ουσιαστική κατανόηση του εύρους των εργαλείων και των τεχνικών που απαιτούνται για την υποστήριξη των διαδικασιών, καθώς επίσης και του τρόπου με τον οποίο θα χρησιμοποιηθούν αποτελεσματικά για την επίλυση του προβλήματος

Υιοθετώντας αυτές τις αρχές μηχανικής, η διαδικασία ανάπτυξης λογισμικού μπορεί να χωριστεί σε τρείς φάσεις:

1. **Ανάλυση**, κατά την οποία θα πρέπει να κατανοήσουμε ,**«τί»** θα πρέπει να αναπτύξουμε και **«πώς»** θα λειτουργεί, ώστε να εξυπηρετεί τον σκοπό της δημιουργίας του,
2. **Σχεδίαση**, κατά την οποία θα πρέπει να αντλήσουμε στοιχεία από την προηγούμενη φάση, για τον κατάλληλο σχεδιασμό,
3. **Πραγματοποίηση**, κατά την οποία , το προϊόν του έργου παραδίδεται συνολικά και ελέγχεται ,σχετικά με το «αν ικανοποιεί τις απαιτήσεις που τέθηκαν από τον πελάτη κατά τα πρώτα στάδια ανάπτυξης».

Η Μηχανική Απαιτήσεων είναι το πεδίο της Μηχανικής Λογισμικού, που ασχολείται με την πρώτη φάση, την Ανάλυση. Η αποτελεσματική Ανάλυση του εκάστοτε προβλήματος που αντιμετωπίζει ο πελάτης οδηγεί στον ακριβή ορισμό του λογισμικού που θα πρέπει να αναπτυχθεί, ώστε να επιλύει το

αρχικό πρόβλημα. Προκειμένου να τελεσφορήσει η Ανάλυση, είναι απαραίτητο να καθοριστούν και να οργανωθούν με σαφή τρόπο όλες οι λειτουργίες, οι περιορισμοί, οι δυνατότητες και άλλες, σχετικές με το συνολικό σύστημα, πληροφορίες. Η ανάλυση των απαιτήσεων μπορεί να πραγματοποιείται επαναληπτικά και σε άλλες φάσεις του έργου ανάπτυξης λογισμικού (2) (3) (4) (5) (6).

Σε αυτή την εργασία, η Μηχανική Απαιτήσεων, πολλές φορές, θα μπορούσε να θεωρηθεί και παρουσιάζεται, ως τρόπος κατανόησης και εξονυχιστικής ανάλυσης, επιχειρησιακών κυρίως, προβλημάτων.

1.2 Τι είναι η Μηχανική Απαιτήσεων;

Η πολυπλοκότητα και το ευρύ φάσμα της Μηχανικής Απαιτήσεων δημιουργούν δυσκολίες στην διατύπωση ενός σαφή και ολοκληρωμένου ορισμού του πεδίου. Παρόλα αυτά, στο παρελθόν έχουν γίνει προσπάθειες για την απόδοση της έννοιας του πεδίου σε μορφή ενός ολοκληρωμένου ορισμού. Αρχικά, όμως, θα πρέπει να οριστεί η έννοια της Απαίτησης:

> Ορισμός 1.1 *Η απαίτηση αποτελεί εντολή εκτέλεσης, μετατροπής, παραγωγής ή παροχής μίας συγκεκριμένης εργασίας, δραστηριότητας ή υπηρεσίας.*

Υιοθετώντας τον ορισμό της απαίτησης, μπορεί να διατυπωθεί *ο σκοπός* της Μηχανικής Απαιτήσεων, που είναι η ανάδειξη του, ***«τί»*** πρέπει να εκτελεί, να μετατρέπει, να παράγει ή να παρέχει το σύστημα, προκειμένου να ικανοποιούνται οι ανάγκες του πελάτη. Αυτές θα πρέπει, όπως αναφέρθηκε προηγουμένως, να αναλύονται σε βάθος, ώστε να επέρχεται σαφής κατανόηση του προβλήματος, συνολικά. Σύμφωνα με τα παραπάνω, ένας προφανής ορισμός της Μηχανικής Απαιτήσεων είναι ο εξής:

> Ορισμός 1.2 *Η Μηχανική Απαιτήσεων είναι η πειθαρχία που σχετίζεται με την κατανόηση και τεκμηρίωση των απαιτήσεων του λογισμικού.*

Το πρότυπο «IEEE Std 610.12-1990» υιοθετεί τους εξής ορισμούς για την Μηχανική Απαιτήσεων:

> Ορισμός 1.3 *(1) Ένας όρος ή μία ικανότητα που απαιτείται από έναν χρήστη για την επίλυση ενός προβλήματος ή την κατάκτηση ενός στόχου. (2) Ένας όρος ή μία ικανότητα που πρέπει να ικανοποιεί ένα σύστημα ή μέρος του*

συστήματος προκειμένου να ικανοποιείται ένα συμβόλαιο, ένα πρότυπο, ή μία προδιαγραφή.

Αυτό που μπορεί να παρατηρηθεί είναι ότι, οι παραπάνω ορισμοί έχουν κοινό πλαίσιο αναφοράς. Το πλαίσιο αυτό είναι η εκάστοτε απαίτηση. Αν θεωρήσουμε το λογισμικό που πρέπει να αναπτυχθεί, ως ένα μέρος ενός συστήματος ή το έργο μηχανικής, ως ένα μέρος ενός μεγαλύτερου έργου, τότε ο ορισμός της Μηχανικής Απαιτήσεων θα πρέπει να λαμβάνει υπόψη και τους όρους (προδιαγραφές, συμβόλαια, πρότυπα) που τίθενται από το εξωτερικό περιβάλλον.

Ορισμός 1.4 *Η Μηχανική Απαιτήσεων είναι η πειθαρχία που σχετίζεται με την κατανόηση των εξωτερικών συνθηκών που επιβάλλονται σε ένα σύστημα υπολογιστών και με την οποία καθορίζονται και τεκμηριώνονται οι ικανότητες του συστήματος, ως απαιτήσεις λογισμικού για το υπολογιστικό σύστημα.*

(2) (3) (4) (5) (6)

1.3 Η Σημασία των Απαιτήσεων

Σε πολλές περιπτώσεις διοίκησης έργων ανάπτυξης πληροφοριακών συστημάτων έχει παρατηρηθεί ότι, δεν δίνεται η απαραίτητη προσοχή στο ζήτημα των απαιτήσεων, με αποτέλεσμα να μην αποσαφηνίζονται οι απαιτήσεις του πελάτη και το κυρίαρχο πρόβλημα που, αυτός, επιθυμεί να επιλύσει. Επιπλέον, οι προγραμματιστές βιάζονται να πάρουν αποφάσεις σχεδιαστικού χαρακτήρα, με αποτέλεσμα να μην υπολογίζουν όλους τους περιορισμούς του συστήματος. Το αποτέλεσμα είναι η εισαγωγή μη έγκυρων υποθέσεων στο σύστημα, το οποίο τελικά αποτυγχάνει να εξυπηρετήσει τις ανάγκες του πελάτη.

Ένας τρόπος για να κατανοήσει κανείς την σημασία των απαιτήσεων είναι να παρατηρήσει τα κόστη διόρθωσης λαθών για κάθε μία από της φάσεις ανάπτυξης του συστήματος. Οι σχέσεις κόστους διόρθωσης λαθών για κάθε μία από τις φάσεις παρουσιάζονται στο Σχήμα 1.1(7):

Phase	Relative Cost
Requirements	0.1-0.2
Design	0.5
Coding	1
Unit Testing	2
Acceptance Testing	5
Maintenance	20

Σχήμα 1.1: Οι σχέσεις κόστους διόρθωσης λαθών (εντοπισμός και διόρθωση) για κάθε μία από τις φάσεις του έργου.

Μπορεί κανείς να παρατηρήσει ότι, ένα λάθος ή μία αστοχία του συστήματος ***που γίνεται αντιληπτή και διορθώνεται*** κατά την φάση των απαιτήσεων, κοστίζει 0,2 φορές σε σχέση με το κόστος της ίδιας διαδικασίας, κατά την φάση της ανάπτυξης του κώδικα του λογισμικού.

Σε έρευνά τους οι Sallis, Tate και McDonell (1996) επισημαίνουν ότι, το ποσοστό των λαθών που παρατηρούνται σε λογισμικά και συστήματα και προέρχονται από την φάση των απαιτήσεων ξεπερνά το 50% του συνόλου των αστοχιών όλων των φάσεων του έργου (8).

1.4 Προσεγγίσεις στη Μηχανική Απαιτήσεων σε Έργα Ανάπτυξης Πληροφοριακών Συστημάτων

Τα τελευταία τριάντα (30) χρόνια, έχουν καθιερωθεί αρκετές προσεγγίσεις σχετικά με τη Μηχανική Απαιτήσεων. Παρακάτω, περιγράφονται συνοπτικά, μερικές από αυτές:

1.4.1 Η Ορθόδοξη Προσέγγιση

Η συγκεκριμένη προσέγγιση επικεντρώνεται στις βασικές δραστηριότητες και αρχές της Μηχανικής Απαιτήσεων, υπογραμμίζοντας την συλλογή και συγκέντρωση των απαιτήσεων και την ακριβή κατανόηση του κυρίαρχου προβλήματος (9):

Εξόρυξη Απαιτήσεων: Η διαδικασία, μέσω της οποίας, πελάτες και ειδικοί συζητούν, διατυπώνουν και κατανοούν τις ανάγκες του πελάτη και τους περιορισμούς του λογισμικού.

Η εξόρυξη απαιτήσεων, ως διαδικασία, απαιτεί, τουλάχιστον, την ανάμειξη των πελατών, των ειδικών και των τελικών χρηστών, προκειμένου να αποδώσει τις

κατάλληλες πληροφορίες σχετικά με τις απαιτήσεις του πελάτη και τους περιορισμούς του συστήματος. Οι δημοφιλέστερες τεχνικές εξόρυξης απαιτήσεων είναι : (1) Συνεντεύξεις, (2) Ερωτηματολόγια, (3) Ομάδες Εστίασης και (4) Μοντελοποίηση.

Ανάλυση Απαιτήσεων: Η διαδικασία της ανάλυσης των απαιτήσεων του πελάτη, προκειμένου να οριστούν σαφώς. Αν η εξόρυξη απαιτήσεων αποσκοπεί στην συλλογή των πληροφοριών, τότε η ανάλυση απαιτήσεων αποσκοπεί στην κατανόηση των αναγκών του πελάτη και των περιορισμών του συστήματος. Συνήθως, η μοντελοποίηση των απαιτήσεων πραγματοποιείται παράλληλα με την εξόρυξη των απαιτήσεων.

Προσδιορισμός Απαιτήσεων: Η διαδικασία της δημιουργίας ενός εγγράφου, στο οποίο προσδιορίζονται σαφώς οι απαιτήσεις.

Αυτό που πρέπει να σημειωθεί, είναι ότι οι απαιτήσεις υπάρχουν, ανεξάρτητα από τον προσδιορισμό τους. Στο έγγραφο που συντάσσεται, οι απαιτήσεις εκφράζονται.

Το IEEE υιοθετεί 8 κριτήρια για την αξιολόγηση ενός τέτοιου εγγράφου: (1) Ορθότητα των απαιτήσεων, (2) Πληρότητα των απαιτήσεων, (3) Συνοχή των απαιτήσεων, (4) Ανιχνευσιμότητα των απαιτήσεων, (5) Προτεραιότητα των απαιτήσεων, (6) Ιεράρχηση των απαιτήσεων, (7) Επαληθευσιμότητα απαιτήσεων, (8) Δυνατότητα τροποποίησης απαιτήσεων

Επικύρωση Απαιτήσεων: Η διαδικασία κατά την οποία επιβεβαιώνεται η συμμόρφωση των απαιτήσεων και του προσδιορισμού των απαιτήσεων, με τις ανάγκες του πελάτη και του συστήματος.

Η επικύρωση των απαιτήσεων δεν είναι καθόλου εύκολη διαδικασία. Είναι πιθανό, οι πελάτες, οι χρήστες και οι ειδικοί, να μην γνωρίζουν τι πρέπει και αν θα είναι σωστό να συμβαίνει στο σύστημα. Ένα πολύ χαρακτηριστικό παράδειγμα αποτελεί η περίπτωση του έργου A3420 Airbus, κατά την διάρκεια του οποίου, προβλήματα κατανόησης των απαιτήσεων και του σχεδιασμού του συστήματος, παρατηρήθηκαν ταυτοχρόνως.

Οι δημοφιλέστερες τεχνικές σε αυτή τη φάση είναι: (1) Αναφορές και επιθεωρήσεις των απαιτήσεων και (2) Προτυποποίηση .

Διαχείριση Απαιτήσεων: Ο προγραμματισμός και η παρακολούθηση των φάσεων και διαδικασιών της Μηχανικής Απαιτήσεων, καθώς και της αλληλεξάρτησης των απαιτήσεων, όπως διαμορφώνεται στον χρόνο.

Σύμφωνα με την ορθόδοξη προσέγγιση, οι απαιτήσεις διαχωρίζονται σε 4 βασικές κατηγορίες (9):

1. ***Λειτουργικές Απαιτήσεις:*** Οι λειτουργικές απαιτήσεις καθορίζουν **«τί»** πρέπει να κάνει το σύστημα. Οι κύριες λειτουργίες του συστήματος ορίζουν «τί» πρέπει να μεταμορφώνει, ολοκληρώνει, παράγει, ή να παρέχει το σύστημα. Για παράδειγμα, το λογισμικό για ένα A320 Airbus πρέπει να παρέχει:

   - Έναν ασφαλή τρόπο διατήρησης του αεροσκάφος μεταξύ της μέγιστης γωνίας ανόδου και της μέγιστης γωνίας καθόδου.

   - Λειτουργίες σταθεροποίησης της πτήσης, για περιπτώσεις αναταραχών του αεροσκάφους.

   Σημείωση: Οι λειτουργικές απαιτήσεις δεν σχετίζονται με το «πώς» αυτές οι λειτουργίες θα αναπτυχθούν, παρά μόνο με το «τί» λειτουργίες θα αναπτυχθούν.

2. ***Το Μοντέλο Εστίασης Πληροφορίας:*** Το μοντέλο πληροφορίας για το πρόβλημα αποτελείται από την λογική αναπαράσταση των εξής παραμέτρων:
   - «Τι είδους πληροφορίες υπάρχουν στο σύστημα;»,
   - «Ποιός έχει την ανάγκη των πληροφοριών και από που προέρχεται;»,
   - «Ποιά είναι η ροή της πληροφορίας και ποιός την προωθεί;»,
   - «Τι μεταμορφώσεις της πληροφορίας συμβαίνουν;».
3. ***Η Προβλεπόμενη Συμπεριφορά του Συστήματος:*** Η συμπεριφορά του συστήματος προσδιορίζεται από τον τρόπο με τον οποίο το σύστημα ανταποκρίνεται σε γεγονότα που λαμβάνουν χώρα στο εξωτερικό περιβάλλον. Για παράδειγμα, «πως θα ανταποκριθεί το σύστημα, αν πιέσουμε το δεξί κουμπί ενός ποντικιού, όταν έχουμε ανοιχτή μία εφαρμογή;» Κυριότερα, αν ένα Αυτόματο Σύστημα Φρεναρίσματος (Automated Braking System - ABS) εντοπίσει ένα όχημα σε συγκεκριμένη απόσταση από το αυτοκίνητο, τότε «ποιά θα πρέπει να είναι η συμπεριφορά του αυτοκινήτου;», «Θα πρέπει να μειώσει ταχύτητα ή να

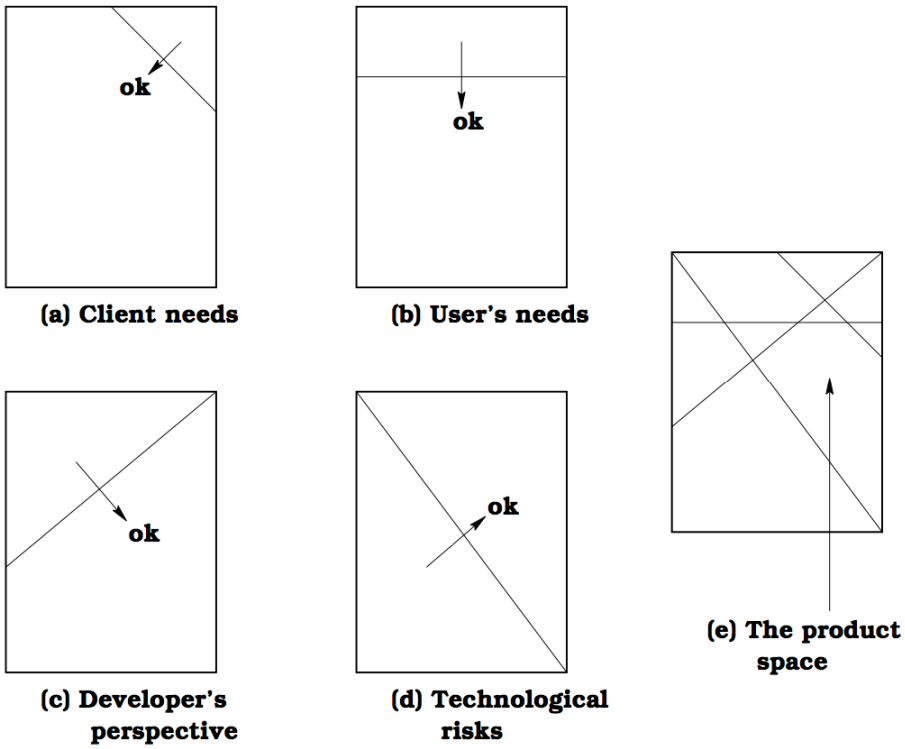
ενεργοποιήσει πλήρως τα φρένα;» Αυτές οι ερωτήσεις θα πρέπει να απαντηθούν απαραίτητα προκειμένου να οριστεί η συμπεριφορά του εκάστοτε συστήματος.

4. ***Μη λειτουργικές Απαιτήσεις:*** Αυτή η κατηγορία σχετίζεται με τα χαρακτηριστικά γνωρίσματα και τις ιδιότητες του λογισμικού και όχι με τις λειτουργίες. Συμπεριλαμβάνονται πτυχές όπως, η απόδοση του λογισμικού, η χρηστικότητα και η αξιοπιστία του, καθώς επίσης και κάθε πτυχή ασφάλειας.

Σημείωση: Το πλαίσιο του συστήματος, το οποίο συμπεριλαμβάνει τις ανάγκες και τις φιλοδοξίες του πελάτη, τους πιθανούς χρήστες του συστήματος, τους ρυθμιστικούς φορείς και άλλα ενδιαφερόμενα μέρη, θα πρέπει να λαμβάνεται, σοβαρά, υπόψη. Φυσικοί και περιβαλλοντικοί παράγοντες, οι κίνδυνοι, οι διαθέσιμες τεχνολογίες, η τεχνογνωσία και τα κόστη συμπεριλαμβάνονται επίσης σε αυτό.

1.4.2 Η Προσέγγιση Επίλυσης Προβλήματος όσον αφορά στη Μηχανική Απαιτήσεων

Ο Davis (10) επικεντρώνεται πιο πολύ στη διαδικασία με την συγκεκριμένη προσέγγιση. Ξεκινά θεωρώντας ότι, *η* ***ανάλυση*** *ενός προβλήματος σχετίζεται με την* ***κατανόηση*** *του κυρίαρχου προβλήματος*. Τελικά, όμως, θα πρέπει να παραχθεί ένα έγγραφο προδιαγραφών απαιτήσεων, που θα οριοθετεί το εύρος των πιθανών, ή των αποδεχτών λύσεων του προβλήματος. Αυτό το εύρος των λύσεων, θα πρέπει οπωσδήποτε να οριστεί. Αρχικά, δεν γνωρίζουμε αρκετές πληροφορίες για το πρόβλημα αλλά, όσο εξελίσσεται η διαδικασία εξόρυξης των απαιτήσεων, το εύρος των πιθανών λύσεων μικραίνει. Πολλοί παράγοντες μπορούν να επηρεάσουν το σύνολο των απαιτήσεων, όσο η διαδικασία κατανόησης του προβλήματος συνεχίζεται. Στο Σχήμα 1.2, φαίνεται ο τρόπος με τον οποίο ορίζεται το εύρος των απαιτήσεων (10).



Σχήμα 1.2: Καθορισμός του εύρους των απαιτήσεων-λύσεων

- Οι ανάγκες που θα εκφράσει ο πελάτης αποτελούν, αδιαμφισβήτητα, ένα μεγάλο μέρος των απαιτήσεων. Οι πελάτες εκφράζουν τις απαιτήσεις τους, όμως οι μηχανικοί θα πρέπει να ορίζουν τις πραγματικές απαιτήσεις. Ο πελάτης, όσον αφορά στη συγκεκριμένη διεργασία, είναι το άτομο ή η οργάνωση που αναθέτει το έργο ανάπτυξης του συστήματος. Οι τυπικές ανησυχίες του πελάτη συμπεριλαμβάνουν την λειτουργικότητα, το χρόνο ανάπτυξης, το κόστος, τη δυνατότητα παραμετροποίησης, τη συντήρηση, την αξιοπιστία και την απόδοση. Οι πελάτες θα πρέπει, επίσης, να ασχολούνται με το ηθικό και το νομικό περιβάλλον, εντός του οποίου θα λειτουργούν, το σύστημα και το λογισμικό.
- Οι πιθανοί χρήστες, επίσης, μπορεί να προτρέχουν και να εκφράζουν τις απαιτήσεις τους δραστικά και με έμφαση. Είναι, όμως, και πάλι ευθύνη του μηχανικού να ορίσει τις απαιτήσεις. Οι χρήστες ασχολούνται με την λειτουργικότητα, την αξιοπιστία και την επίδοση, αλλά και με την αλληλεπίδραση τους με το σύστημα. Η αλληλεπίδραση των χρηστών με το σύστημα συμπεριλαμβάνει ζητήματα χρηστικότητας και διεπαφών, καθώς επίσης και το ζήτημα της επίδρασης του συστήματος στην εργασία των χρηστών.

- Ο προγραμματιστής ή η επιχείρηση που αναλαμβάνει να αναπτύξει ένα πληροφοριακό σύστημα έχει επίσης, μία προσέγγιση του συστήματος. Τον ενδιαφέρουν: (1) η παράδοση ενός συστήματος που να ικανοποιεί τις ανάγκες του πελάτη, (2) νομικά ζητήματα, (3) το μερίδιο αγοράς, (4) το περιθώριο κέρδους, (5) οι σχέσεις με ήδη υπάρχοντα προϊόντα, (6) η τεχνογνωσία που απαιτείται για την εκτέλεση του έργου.
- Το νομικό και ηθικό πλαίσιο, εντός του οποίου θα λειτουργεί το λογισμικό, επηρεάζει, επίσης, τις απαιτήσεις. Σχετίζεται, κυρίως, με ζητήματα ασφαλείας και κινδύνους, όπου το κόστος ενός λάθους μπορεί να είναι πολύ μεγάλο.
- Η τεχνολογία επιδρά σημαντικά στο εύρος των λύσεων. Σχετίζεται, κυρίως, με ζητήματα χρόνου παράδοσης του έργου. Σε αυτά, η τεχνολογία διαδραματίζει σημαντικό ρόλο. Τη συγκεκριμένη προσέγγιση ενδιαφέρουν ζητήματα όπως: «Ποιοί είναι οι κίνδυνοι χρήσης της ήδη υπάρχουσας τεχνολογίας;» , «Ποιοί είναι οι κίνδυνοι χρήσης πιο εξελιγμένης τεχνολογίας;», «Ποιά από τις τεχνολογίες έχει τις περισσότερες πιθανότητες να επιδράσει θετικά στην επιτυχία του συστήματος;», «Θα είναι διαθέσιμη εντός των χρονικών περιορισμών;»

Κατά την διάρκεια της ανάλυσης του προβλήματος θα πρέπει να εξετάζονται όλες οι πτυχές του(10):

- Οι ανάγκες των πελατών,
- Οι ανάγκες των ενεργών χρηστών,
- Οι λειτουργίες στις οποίες εμπλέκονται και συμμετέχουν οι χρήστες,
- Το φυσικό, διανοητικό, νομικό, ηθικό περιβάλλον, εντός του οποίου, ενεργούν και απασχολούνται οι χρήστες.

Ο Davis (10) σημειώνει ότι, η ανάλυση ενός προβλήματος, συνήθως, συμπεριλαμβάνει ανθρώπους. Στα περισσότερα έργα ανάπτυξης πληροφοριακών συστημάτων, κατά τη διαδικασία της ανάλυσης του προβλήματος, συμμετέχουν οι πελάτες ή οι εξουσιοδοτημένοι εκπρόσωποι αυτών, οι πιθανοί χρήστες του συστήματος, οι τεχνικοί, οι ειδικοί, ενώ μερικές φορές, είναι απαραίτητο να συμμετέχουν και ρυθμιστικοί φορείς, μέσω αντιπροσώπων. Αυτοί οι άνθρωποι συγκεντρώνουν ένα μεγάλο μέρος γνώσης, σχετικά με το κυρίαρχο πρόβλημα. Σκοπός του αναλυτή του συστήματος είναι η εξόρυξη της σχετικής γνώσης.

Συνεντεύξεις, επίσημες συναντήσεις με τον πρόεδρο και τους δυνητικούς χρήστες, ερωτηματολόγια και άμεσες παρατηρήσεις, είναι μέθοδοι που χρησιμοποιούνται για τις σχετικές πληροφορίες. Πρέπει να σημειωθεί ότι, η διαδικασία της συλλογής - εκμαίευσης των πληροφοριών είναι επαναληπτική. Καταλήγοντας σε συγκεκριμένα πορίσματα, μετά την συγκέντρωση ενός μέρους των πληροφοριών που αφορούν στην ανάλυση του προβλήματος, είναι πιθανό να αναδύονται πιο συγκεκριμένα και ουσιαστικά ερωτήματα.

Σημείωση: Υπάρχει μία τάση, κατά την διάρκεια της ανάλυσης του προβλήματος, να ξεκινά και η επίλυση του προβλήματος ή ο σχεδιασμός του συστήματος. Αυτές οι δραστηριότητες διαφέρουν. Η ανάλυση επικεντρώνεται στις ανάγκες του πελάτη, ενώ την ενδιαφέρουν οι λειτουργίες («τί;») που πρέπει να εκτελεί το σύστημα, ώστε να ικανοποιηθούν οι ανάγκες αυτές. Ο σχεδιασμός επικεντρώνεται, στον τρόπο («πως;») με τον οποίο, οι ανάγκες αυτές, θα ικανοποιηθούν. Θα ήταν ιδανικά, αν μπορούσε να διαχωριστεί πλήρως η φάση των απαιτήσεων από αυτή του σχεδιασμού. Κάτι τέτοιο, όμως, είναι πολύ δύσκολο. Η τελική προδιαγραφή των απαιτήσεων θα πρέπει να λαμβάνει υπόψη το κόστος των απαιτήσεων, επομένως, η τεχνική αξιολόγηση του συστήματος θα πρέπει να έχει πραγματοποιηθεί. Πιο πρόσφατες απόψεις πάντως, εκφράζουν και προτείνουν ότι, η προδιαγραφή των απαιτήσεων και ο σχεδιασμός θα πρέπει να εκτελούνται παράλληλα (10).

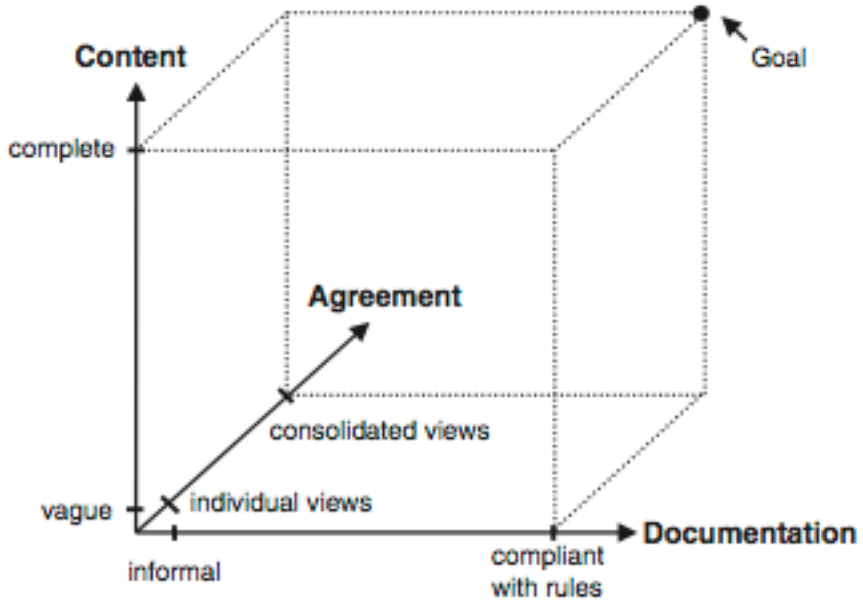
1.4.3 Η Προοπτική των Τριών Διαστάσεων

Υπό το πρίσμα της συγκεκριμένης σκοπιάς, η Μηχανική Απαιτήσεων αποτελεί επαναληπτική, συνεργατική και οριακή διαδικασία και αποσκοπεί στην επιβεβαίωση των εξής (11):

1. Όλες οι σχετικές απαιτήσεις είναι γνωστές και κατανοητές στον απαραίτητο βαθμό λεπτομέρειας.
2. Συμφωνία μεταξύ των εμπλεκόμενων μερών, όσον αφορά στις απαιτήσεις του συστήματος.
3. Όλες οι απαιτήσεις τεκμηριώνονται και καθορίζονται σύμφωνα με τους σχετικούς κανόνες και πρότυπα τεκμηρίωσης και καθορισμού.

Προκειμένου να επιβεβαιωθούν τα παραπάνω, η συγκεκριμένη προοπτική προσδιορίζει και υιοθετεί τρείς βασικές δραστηριότητες. Κάθε μία από τις βασικές δραστηριότητες συμβάλλει σημαντικά στην επίτευξη ενός εκ των τριών παραπάνω σημείων. Τις βασικές δραστηριότητες υποστηρίζουν άλλες δύο.

Παρακάτω παρουσιάζονται οι δραστηριότητες της προοπτικής των τριών διαστάσεων (11):



Σχήμα 1.3: Οι διαστάσεις της προοπτικής τριών διαστάσεων

Τεκμηρίωση

Η συγκεκριμένη δραστηριότητα επικεντρώνεται στην τεκμηρίωση και την προδιαγραφή των απαιτήσεων που συλλέγονται, με γνώμονα τους καθορισμένους κανόνες τεκμηρίωσης και προδιαγραφής. Επιπλέον, τεκμηριώνονται οι αποφάσεις και ο τρόπος λήψης αυτών. Η δραστηριότητα της Τεκμηρίωσης διαχωρίζει τους κανόνες ως εξής (11):

Γενικοί Κανόνες Τεκμηρίωσης: Οι κανόνες αυτοί εφαρμόζονται σε όλα τα είδη πληροφοριών, όπως για παράδειγμα στα πρωτόκολλα συναντήσεων, στις συνεντεύξεις, σε πληροφορίες σχετικά με το πλαίσιο ή τις αποφάσεις και τον τρόπο λήψης αυτών. Αυτοί οι κανόνες καθορίζουν τον τρόπο διαχείρισης των εγγράφων, τη διάταξή τους και τον γενικότερο τρόπο παρουσίασής τους.

Κανόνες Τεκμηρίωσης: Αυτοί οι κανόνες εφαρμόζονται σε κάθε έγγραφο απαιτήσεων στα διαφορετικά στάδια της διαδικασίας της Μηχανικής Απαιτήσεων. Στόχος αυτών των κανόνων είναι να επιβεβαιώσουν ένα επίπεδο ποιότητας επαρκώς τεκμηριωμένων απαιτήσεων, κυρίως, προκειμένου να μπορούν να χρησιμοποιηθούν αποτελεσματικά σε άλλες δραστηριότητες της Μηχανικής Απαιτήσεων (διαπραγμάτευση απαιτήσεων, επαλήθευση

απαιτήσεων). Ταυτόχρονα, οι κανόνες διευκολύνουν σημαντικά τη διαδικασία της τεκμηρίωσης με διάφορους τρόπους. Για παράδειγμα, προτείνουν συγκεκριμένα πρότυπα για την τεκμηρίωση των απαιτήσεων.

Κανόνες Προδιαγραφής: Αυτοί οι κανόνες εφαρμόζονται σε όλες τις απαιτήσεις που περιλαμβάνονται στην προδιαγραφή απαιτήσεων. Σκοπός αυτών των κανόνων είναι να επιβεβαιώσουν ένα υψηλό επίπεδο ποιότητας των απαιτήσεων που προδιαγράφονται για χρήση ως βασικές είσοδοι ή μέρη συμβάσεων σε μεταγενέστερες δραστηριότητες ανάπτυξης. Οι κανόνες προδιαγραφής μπορούν να προσδιορίζουν τη γλώσσα προδιαγραφής απαιτήσεων ή τα συντακτικά πρότυπα.

Γενικά, οι κανόνες προδιαγραφής είναι πιο περιοριστικοί σε σχέση με αυτούς της τεκμηρίωσης. Αναλόγως με την τεχνική απαιτήσεων, μπορούν να εφαρμόζονται διαφορετικοί κανόνες και πρότυπα τεκμηρίωσης και προδιαγραφής. Για παράδειγμα, μία απαίτηση μπορεί να τεκμηριώνεται με τη χρήση της φυσικής γλώσσας, προκειμένου να επιτυγχάνεται η επικοινωνία με τους τυπικούς τελικούς χρήστες , ενώ μπορεί να προδιαγράφεται με τη χρήση μίας τυπικής γλώσσας απαιτήσεων, προκειμένου να υποστηριχτεί η διαδικασία ανάπτυξης της αρχιτεκτονικής του συστήματος από τον αρχιτέκτονα του συστήματος. Όπως γίνεται αντιληπτό, τα διαφορετικά εμπλεκόμενα μέρη προτιμούν διαφορετικά πρότυπα τεκμηρίωσης και προδιαγραφής. Πολλές φορές, είναι απαραίτητη η αποτύπωση των ίδιων κανόνων σε διαφορετικά πρότυπα, προκειμένου να επικοινωνούν αποτελεσματικά με τα διάφορα εμπλεκόμενα μέρη. Αυτή η διαδικασία έχει ως προϋπόθεση την παρακολούθηση και συνεχή ανανέωση των εγγράφων.

Εξόρυξη

Ο σκοπός της δραστηριότητας εξόρυξης απαιτήσεων είναι η βελτίωση του επιπέδου κατανόησης των απαιτήσεων προκειμένου να επιτυγχάνεται πρόοδος όσον αφορά στο περιεχόμενο τους**.** Κατά τη διάρκεια της δραστηριότητας εξόρυξης, οι απαιτήσεις εκμαιεύονται από τα εμπλεκόμενα μέρη και από άλλες πηγές απαιτήσεων. Επιπλέον, αναπτύσσονται συνεργατικά νέες και καινοτόμες απαιτήσεις.

Οι πηγές απαιτήσεων που σχετίζονται με το σύστημα δεν είναι πάντα γνωστές από την αρχή της διαδικασίας. Μία βασική διεργασία της δραστηριότητας

εξόρυξης απαιτήσεων είναι η συστηματική αναγνώριση των σχετικών πηγών απαιτήσεων. Σε αυτές ανήκουν οι φορείς που εμπλέκονται στη διαδικασία, τα υπάρχοντα έγγραφα και τα υφιστάμενα συστήματα. Οι απαιτήσεις εκμαιεύονται, για παράδειγμα, μέσα από συνεντεύξεις με τα εμπλεκόμενα μέρη, ή μέσα από διερεύνηση και ανάλυση των εγγράφων ή των συστημάτων. Οι καινοτόμες απαιτήσεις αναπτύσσονται μέσα από δημιουργικές και συνεργατικές δραστηριότητες, όπως ο διαλογισμός, ενώ δεν μπορούν να εκμαιευτούν από τις πηγές απαιτήσεων (11).

Διαπραγμάτευση

Το σύστημα θα πρέπει να εξυπηρετεί τις ανάγκες και τις επιθυμίες των εμπλεκόμενων μερών. Προφανώς, οι ανάγκες και οι επιθυμίες των διαφορετικών εμπλεκόμενων μερών μπορούν να διαφέρουν. Κάθε εμπλεκόμενο μέρος διατηρεί διαφορετική προοπτική για το σύστημα που πρέπει να αναπτυχθεί. Οι διαφορετικές απόψεις των εμπλεκόμενων μερών μπορούν να έρχονται σε σύγκρουση μεταξύ τους. Ο σκοπός της δραστηριότητας διαπραγμάτευσης έχει δύο μέρη: 1) Προσδιορισμός και σαφής καθορισμός των αντικρουόμενων προοπτικών, 2) Επίλυση των διαφορών των διαφορετικών προοπτικών (στο επίπεδο που είναι εφικτή). Για την αντιμετώπιση των διαφορών των προοπτικών των ενδιαφερόμενων μερών εφαρμόζονται στρατηγικές. Κατά τα πρώτα στάδια της Μηχανικής Απαιτήσεων, συνήθως, οι προοπτικές παρουσιάζουν σημαντικές διαφορές και αντικρούσεις μεταξύ τους. Ιδανικά, στα τελικά στάδια της Μηχανικής Απαιτήσεων, οι αντικρουόμενες προοπτικές εξαλείφονται.

Όπως αναφέρθηκε παραπάνω, αναφορικά με τη Μηχανική Απαιτήσεων, η προοπτική των τριών διαστάσεων υιοθετεί το μοντέλο που συντίθεται από τις βασικές δραστηριότητες: Τεκμηρίωση, Εξόρυξη, Διαπραγμάτευση και υποστηρίζει αυτές τις δραστηριότητες με δύο άλλες: Επαλήθευση, Διαχείριση. Παρακάτω δίνεται μία περιγραφή των δύο υποστηρικτικών δραστηριοτήτων (11):

Επαλήθευση

Η δραστηριότητα της επαλήθευσης αποτελείται από τρείς υπο-δραστηριότητες:

Επαλήθευση των τεχνικών απαιτήσεων: Η επαλήθευση των τεχνικών απαιτήσεων έχει σκοπό την ανίχνευση ελαττωματικών απαιτήσεων. Μόνο οι απαιτήσεις υψηλού επιπέδου ποιότητας παρέχουν σταθερή βάση για τον αρχιτεκτονικό σχεδιασμό, την υλοποίηση του συστήματος και την ανάπτυξη των τεχνικών ελέγχου. Τα ελαττώματα στις απαιτήσεις οδηγούν σε ελαττώματα στην αρχιτεκτονική, στην υλοποίηση και στις τεχνικές ελέγχου. Οι ελαττωματικές απαιτήσεις δεν προκαλούνται μόνο από ακατάλληλη ή ανεπαρκή τεκμηρίωση. Σε πολλές περιπτώσεις, οι ελαττωματικές απαιτήσεις προκαλούνται από αστοχίες σε μία από τις τρείς βασικές διαστάσεις, όπως αυτές ορίστηκαν παραπάνω. Η επαλήθευση έχει στόχο την επικύρωση των αντικειμένων και των τεχνικών σε σχέση με το περιεχόμενο, την τεκμηρίωση και τις διαστάσεις της συμφωνίας. Μία απαίτηση θα πρέπει να χρησιμοποιείται μόνο, ως αναφορά για περεταίρω ανάπτυξη ή ως ένα μέρος μίας νομικής σύμβασης, εφόσον έχει επικυρωθεί επιτυχώς και από τις τρείς πτυχές επαλήθευσης (περιεχόμενο, τεκμηρίωση και συμφωνία).

Επαλήθευση των κύριων δραστηριοτήτων: Η δραστηριότητα της επαλήθευσης των κύριων δραστηριοτήτων (τεκμηρίωση, εξόρυξη, διαπραγμάτευση) έχει σκοπό να εξασκήσει έλεγχο όσον αφορά τη συμμόρφωση μεταξύ των δραστηριοτήτων που εκτελέστηκαν και των προδιαγραφών των διαδικασιών-δραστηριοτήτων.

Επαλήθευση της θεώρησης του πλαισίου συστήματος: Η επαλήθευση της θεώρησης του πλαισίου του συστήματος έχει ως σκοπό να εξετάσει αν η θεώρηση του πλαισίου έγινε με κατάλληλο τρόπο κατά τη διάρκεια της Μηχανικής Απαιτήσεων. Με άλλα λόγια, αυτή η υπο-δραστηριότητα σκοπεύει στην επικύρωση των προτάσεων ότι , όλα τα σχετικά εμπλεκόμενα μέρη συμμετείχαν στη διαδικασία τη στιγμή που ήταν απαραίτητο να συμμετέχουν και ότι, οι σχετικές πτυχές του πλαισίου έχουν εξεταστεί κατά τη διαδικασία της Μηχανικής Απαιτήσεων. Για παράδειγμα, με σεβασμό στην πτυχή χρήσης, θα μπορούσε να πραγματοποιηθεί επαλήθευση και να επικυρωθεί ότι, όλες οι απαραίτητες αλληλεπιδράσεις μεταξύ συστήματος και ρόλων (χρήστες και άλλα συστήματα) έχουν εξορυχτεί.

Διαχείριση

Η δραστηριότητα διαχείρισης μπορεί να διαχωριστεί στις παρακάτω υπο-δραστηριότητες:

Διαχείριση των τεχνικών απαιτήσεων: Αυτή η δραστηριότητα περιλαμβάνει τη διαχείριση των τεχνικών απαιτήσεων κατά τη διάρκεια του κύκλου ζωής του συστήματος. Στις δραστηριότητες συμπεριλαμβάνονται η ιεράρχηση των απαιτήσεων, η καταγραφή των απαιτήσεων (π.χ. αποθήκευση σε βάση δεδομένων), η διαχείριση της αλλαγής των απαιτήσεων, η διαχείριση τροποποιήσεων και παραμετροποιήσεων των απαιτήσεων και η διατήρηση της ιχνηλασιμότητας των απαιτήσεων.

Διαχείριση των δραστηριοτήτων: Αυτή η δραστηριότητα περιλαμβάνει τον προγραμματισμό και τον έλεγχο των δραστηριοτήτων της Μηχανικής Απαιτήσεων ώστε να επιβεβαιώνεται η αποτελεσματικότητα της διαδικασίας, συνολικά. Εφόσον είναι απαραίτητο, η προγραμματισμένη ροή των δραστηριοτήτων μπορεί να αλλάζει και να ευθυγραμμίζεται με την κατάσταση του έργου.

Παρατήρηση του πλαισίου συστήματος: Η παρατήρηση του πλαισίου του συστήματος έχει σκοπό να προσδιορίσει αλλαγές στο πλαίσιο του συστήματος οι οποίες σχετίζονται με το σύστημα. Μία σχετική αλλαγή πλαισίου απαιτεί την εκτέλεση μίας ή περισσότερων δραστηριοτήτων Μηχανικής Απαιτήσεων ή προγραμματισμό εκ νέου των δραστηριοτήτων .

1.5 Σύγκριση των Προσεγγίσεων

Ο στόχος και των τριών προσεγγίσεων που περιγράφηκαν παραπάνω είναι κοινός. Είναι ο **προσδιορισμός των σωστών απαιτήσεων από το σύστημα**, αποφεύγοντας πιθανή αναθεώρηση τους στην πορεία υλοποίησης του έργου και η οποία θα επιβαρύνει το χρόνο και κόστος υλοποίησης του έργου.

Το βασικότερο πλεονέκτημα της *ορθόδοξης μεθόδου* είναι ο καθορισμός σταδίων στην Μηχανική Απαιτήσεων, γεγονός που διευκολύνει στην διεξαγωγή της από τον Μηχανικό Απαιτήσεων. Παρέχει έναν «**οδικό χάρτη**» για την ανάλυση απαιτήσεων. Με την εφαρμογή της ορθόδοξης μεθόδου υπάρχει μεγαλύτερη σιγουριά πως θα διεξαχθούν οι απαραίτητες δράσεις - ενέργειες κατά την ανάλυση απαιτήσεων.

Η προσέγγιση που προτείνει ο Davis (Η Προσέγγιση Επίλυσης Προβλήματος) περιγράφοντας όλους τους πιθανούς **εμπλεκόμενους - οντότητες** που επηρεάζουν τον τελικό καθορισμό απαιτήσεων, επίσης διευκολύνει τον Μηχανικό Απαιτήσεων, καθώς κατευθύνει τις δράσεις του, «ποιούς» και «τί» θα πρέπει να λάβει υπόψη του στη διαδικασία ανάλυσης απαιτήσεων. Η προσέγγιση Davis προσδιορίζει τους βασικούς παράγοντες που επηρεάζουν τον καθορισμό απαιτήσεων.

Τέλος, η προσέγγιση που προτείνει ο Pohl (Η Προοπτική των Τριών Διαστάσεων), καθορίζει τις βασικές **διαστάσεις** που θα πρέπει να ληφθούν υπόψη κατά τη διαδικασία ανάλυσης απαιτήσεων. Η ανάλυση απαιτήσεων θα πρέπει να καταλήξει στη πλήρη κατανόηση του *περιεχομένου* όλων των απαιτήσεων από τους εμπλεκόμενους φορείς (stakeholders), στο *συμβιβασμό* μεταξύ των εμπλεκόμενων (stakeholders) σχετικά με τις τελικές απαιτήσεις του συστήματος και οπωσδήποτε σε *τεκμηρίωση* των απαιτήσεων με επίσημο τρόπο (καταγραφή σε κείμενο).

Κατά τη διαδικασία Ανάλυσης Απαιτήσεων θα πρέπει να εφαρμοστεί μία συνδυαστική προσέγγιση των τριών προσεγγίσεων που περιγράφηκαν παραπάνω. Ως οδηγός θα πρέπει να χρησιμοποιηθεί η ορθόδοξη μέθοδος. Όλες οι οντότητες που περιγράφονται στη προσέγγιση Davis θα πρέπει να συμπεριληφθούν στη διαδικασία ανάλυσης απαιτήσεων. Επίσης, ο Μηχανικός Απαιτήσεων θα πρέπει να δώσει βαρύτητα στις τρεις βασικές διαστάσεις που αναφέρει ο Pohl (διαπραγματεύσεις με τους εμπλεκόμενους παίκτες, εξόρυξη απαιτήσεων σε βάθος για την πλήρη κατανόηση των απαιτήσεων και, τέλος, καταγραφή τους με επίσημο τρόπο).

Με τη συνδυαστική εφαρμογή των τριών προσεγγίσεων, θα διασφαλιστεί σε μεγάλο βαθμό πως οι **πραγματικές ανάγκες** του πελάτη θα προσδιοριστούν πλήρως και σαφώς.

1.6 Ο Αναλυτής

Οι απαιτήσεις προέρχονται από πολλές πηγές, αλλά συνήθως, περιλαμβάνουν πελάτες και τελικούς χρήστες. Γίνεται αντιληπτό επομένως, ότι η Μηχανική Απαιτήσεων είναι μία σημαντικά κοινωνική διαδικασία. Μεγάλο μέρος της γνώσης και των πληροφοριών που απαιτούνται βρίσκεται «αποθηκευμένο» στα εμπλεκόμενα μέρη του συστήματος, δηλαδή σε ανθρώπους. Παρατηρείται λοιπόν, το εξής φαινόμενο, πολλές από τις πληροφορίες που απαιτούνται,

εκμαιεύονται αργοπορημένα, ακόμα και μετά την φάση της προδιαγραφής απαιτήσεων. Επομένως, είναι σημαντικό για τον αναλυτή, να εκμαιεύει όσο το δυνατόν περισσότερη γνώση, τηρώντας τους χρονικούς περιορισμούς, και να την παρέχει στην ομάδα υλοποίησης, με την καταλληλότερη και πιο διακριτή δυνατή μορφή. Τέλος, σε περιπτώσεις κατά τις οποίες, οι πελάτες ή οι ειδικοί δεν κατανοούν τι πραγματικά απαιτείται, ή δεν εκφράζουν σωστά αυτό που πραγματικά χρειάζεται να συμβεί, ο αναλυτής θα πρέπει να είναι σε θέση να ορίσει την πραγματική απαίτηση (12).

2 Μηχανική Απαιτήσεων και Τεχνολογία Λογισμικού

2.1 Εισαγωγή

«Τι ορίζεται ως Τεχνολογία Λογισμικού;» Δεν υπάρχει μοναδικά ορισμένη απάντηση σε αυτή την ερώτηση, ενώ υπάρχουν σχεδόν τόσες απαντήσεις, όσα και τα συγγράμματα που αναφέρονται σε αυτήν. Ακολουθεί μία αρκετά πλήρης προσέγγιση του ορισμού της Τεχνολογίας Λογισμικού:

> *Αυτό που διακρίνει την Μηχανική και επομένως τη Μηχανική Λογισμικού, ή αλλιώς Τεχνολογία Λογισμικού, από την Ad-Hoc ανάπτυξη, είναι η δυνατότητα που έχει, να ασκεί έλεγχο στην ποιότητα των προϊόντων και στις εξόδους του συστήματος. Κατ' αυτό το τρόπο, η Τεχνολογία Λογισμικού ορίζεται ως, η μελέτη ή το ερευνητικό πεδίο εκείνο, που ασχολείται με τις διαδικασίες, τις μεθόδους, τα εργαλεία και τις τεχνικές, σχετικά με την αποτελεσματικότερη άσκηση του ελέγχου της ποιότητας των προϊόντων και των εξόδων της ανάπτυξης λογισμικού.*

Ο έλεγχος των εξόδων του έργου και της ποιότητας των προϊόντων του, αναφέρεται στις εξής δυνατότητες-ικανότητες των ειδικών:

- δυνατότητα επανειλημμένου προσδιορισμού και επιτυχίας του επιθυμητού επιπέδου ποιότητας των προϊόντων,
- δυνατότητα μέτρησης και βελτιστοποίησης της ποιότητας των προϊόντων λογισμικού και
- δυνατότητα πρόβλεψης, προγραμματισμού και διαχείρισης της ανάπτυξης των προϊόντων λογισμικού.

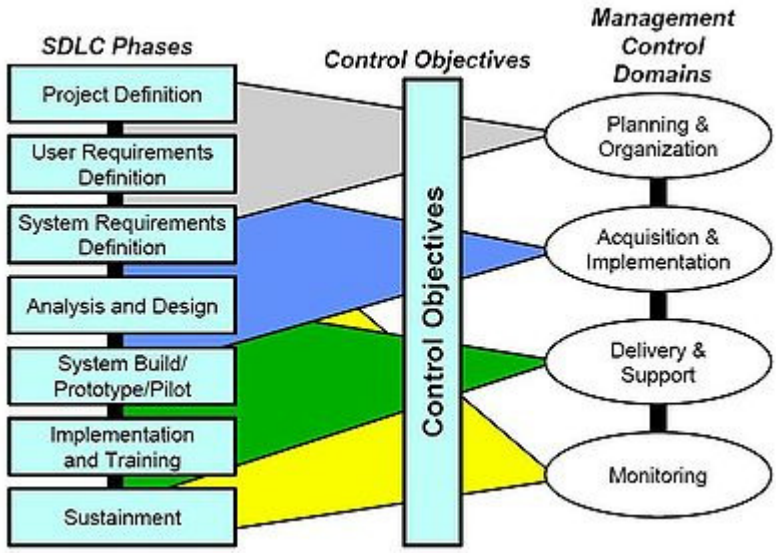
Η άσκηση του ελέγχου μπορεί να πραγματοποιηθεί σε διαφορετικά επίπεδα, όπως φαίνεται στο Σχήμα 2.1. Οι επιλογές που γίνονται σε κάθε ένα από τα επίπεδα, μπορούν να επηρεάσουν δραματικά τις εξόδους του έργου και την ποιότητα αυτών.

⇒ **Διαδικασίες Τεχνολογίας Λογισμικού** (Διαδικασίες κύκλου ζωής όπως: διαδικασίες καταρράκτη, εξελικτικού ή επαναληπτικού μοντέλου)

⇒ **Μεθοδολογίες Μηχανικής** (Μεθοδολογίες για την εφαρμογή των επιθυμητών ιδιοτήτων του λογισμικού όπως: επίδοση, αξιοπιστία, ή χρηστικότητα και μέθοδοι μηχανικής απαιτήσεων ή σχεδιασμού)

⇒ **Εργαλεία Ανάπτυξης και Περιβάλλον** (Εργαλεία περιπτώσεων όπως, προγράμματα εντοπισμού σφαλμάτων, εφαρμογές ελέγχου εκδόσεων και εργαλεία διαχείρισης διαμόρφωσης)

Τα στάδια ανάπτυξης ενός λογισμικού, αποτελούν κατευθυντήρια οδό για τη διαχείριση έργων πληροφορικής. Οι στόχοι του ελέγχου μπορούν να ομαδοποιηθούν σε τομείς (domains) και να αντιστοιχηθούν στα στάδιο της ανάπτυξης λογισμικού, με τον τρόπο που απεικονίζεται στο Σχήμα 2.1. Σε κάθε στάδιο ανάπτυξης διαφοροποιούνται οι κατηγορίες των στόχων που τίθενται (13).



Σχήμα 2.1: Σκοπός ελέγχου σε έργο ανάπτυξης λογισμικού σε κάθε στάδιο ανάπτυξης

Ορισμός 2.1 *Η Τεχνολογία Λογισμικού είναι η εφαρμογή και χρήση των αρχών της Μηχανικής Ήχου, με σκοπό την δημιουργία αξιόπιστου λογισμικού, το οποίο να λειτουργεί αποδοτικά σε πραγματικές μηχανές.*

Το IEEE ορίζει την Τεχνολογία Λογισμικού, ως εξής:

Ορισμός 2.2 *Τεχνολογία Λογισμικού: (1) Η εφαρμογή, πειθαρχημένης και μετρήσιμης προσέγγισης στην ανάπτυξη, λειτουργία και συντήρηση του λογισμικού. (2) Η μελέτη της προσέγγισης αυτής.*

Μία προοπτική που υιοθετεί ο Pfleeger (14) σχετικά με την Τεχνολογία Λογισμικού είναι η εξής:

Ορισμός 2.3 *Η Τεχνολογία Λογισμικού σχετίζεται με την χρήση της γνώσης μας, όσον αφορά τους ηλεκτρονικούς υπολογιστές και τα σχετικά υπολογιστικά συστήματα, προκειμένου να λυθούν συγκεκριμένα προβλήματα.*

2.2 Διασφάλιση Ποιότητας Λογισμικού (Software Quality Assurance)

Η διασφάλιση της ποιότητας αποτελεί μόνιμη «ανησυχία» της Μηχανικής Απαιτήσεων. Η ποιότητα αποτελεί βασικό μέλημα της διαχείρισης έργων ανάπτυξης πληροφοριακών συστημάτων και αφορά, κατά κύριο λόγο, στην ποιότητα των διαδικασιών των φάσεων ανάπτυξης, στις μεθόδους και τα εργαλεία που επιλέγονται. Τα ποιοτικά προϊόντα μπορούν να αυξήσουν το μερίδιο αγοράς και να προωθήσουν το μακροχρόνιο κέρδος για την επιχείρηση, ενώ, το αντίθετο συμβαίνει σε περιπτώσεις προϊόντων που δεν πληρούν αυξημένες προϋποθέσεις ποιότητας. Παρόλη την εστίαση στην ποιότητα, σε πραγματικά έργα, η διαδικασία προσέγγισης προσφοράς πολύ ποιοτικών υπηρεσιών, μπορεί να είναι πολύ δύσκολη υπόθεση!

Η πολυπλοκότητα των σύγχρονων συστημάτων, οι επαναληπτικές αιτήσεις για αλλαγές στις απαιτήσεις και στον σχεδιασμό και η ποικιλία των απαιτήσεων των χρηστών και των πλατφορμών δυσχεραίνουν το έργο, προκειμένου να επιτύχει ποιοτικά.

Προκειμένου να αναπτυχθεί λογισμικό, με βάση τις αρχές της μηχανικής, θα πρέπει να είμαστε σε θέση να ορίσουμε τα χαρακτηριστικά του τελικού συστήματος και να επιλέξουμε εργαλεία, τεχνικές και διαδικασίες, ώστε να επιτύχουμε την ανάπτυξη ενός συστήματος με αυτά τα χαρακτηριστικά.

Λαμβάνοντας υπ' όψη τα παραπάνω, η ποιότητα είναι ένα σύνολο χαρακτηριστικών, τα οποία, επιθυμούμε, να πληρεί το λογισμικό. Ο Μηχανικός Απαιτήσεων καλείται να αναγνωρίσει τους κύριους οδηγούς διασφάλισης ποιότητας του εκάστοτε έργου ανάπτυξης πληροφοριακού συστήματος ή λογισμικού.

«Τι είναι η Ποιότητα Λογισμικού;» Στην ερώτηση αυτή, οι απαντήσεις μπορούν να διαφέρουν, αρκετά, μεταξύ τους. Ακολουθούν οι προοπτικές του ζητήματος:

1. **Η προοπτική του τελικού χρήστη:** Αν ικανοποιεί τον σκοπό του, αν είναι αξιόπιστο, αν επιτυγχάνει επιδόσεις, η ευκολία κατά την χρήση και η βοήθεια που προσφέρει, ώστε να επιτυγχάνουν οι ίδιοι τους στόχους τους, είναι μερικά από τα χαρακτηριστικά που εξετάζουν οι τελικοί χρήστες, προκειμένου να κρίνουν, αν ένα προϊόν είναι ποιοτικό ή όχι. Πρέπει να σημειωθεί ότι, σε περιπτώσεις κατά τις οποίες, ένα προϊόν είναι απαραίτητο ή αξίζει τελικά τον κόπο να αναπτυχθεί, οι τελικοί χρήστες το θεωρούν ποιοτικό, παρά τις ενδεχόμενες δυσκολίες που μπορεί να συνάντησαν, προκειμένου να αποκτήσουν την τεχνογνωσία της χρήσης.
2. **Η προοπτική των ειδικών ανάπτυξης:** Στα χαρακτηριστικά που επικεντρώνονται οι ειδικοί του έργου, προκειμένου να το ελέγξουν ποιοτικά, συμπεριλαμβάνονται: ο αριθμός των αστοχιών του συστήματος, η ευκολία παραμετροποίησης του συστήματος, η ευκολία στον έλεγχο, η φύση του σχεδιασμού, η συμμόρφωση με τις απαιτήσεις.
3. **Η προοπτική όσων συντηρούν το σύστημα:** Αυτή η προοπτική συμπεριλαμβάνει κάποια από τα χαρακτηριστικά της προηγούμενης κατηγορίας, όσον αφορά στην ποιότητα του προϊόντος, αλλά σε αυτά προστίθενται η απλότητα και η προσαρμοστικότητα του συστήματος, η τεκμηρίωση που παράγεται από τους ειδικούς ανάπτυξης και η ευκολία κατανόησης της εφαρμογής.

Ο Μηχανικός Απαιτήσεων εργάζεται με σκοπό την επιλογή των κατάλληλων εργαλείων και τεχνικών, για την παρακολούθηση και τον έλεγχο της ποιότητας των προϊόντων ή των διαδικασιών που επιλέγονται για την ανάπτυξη του προϊόντος. Η άσκηση ελέγχου ποιότητας σε ένα προϊόν ή μία διαδικασία, συνήθως, περιλαμβάνει και την πτυχή της μέτρησης ή της αξιολόγησής, προκειμένου να προσδιοριστεί το επίπεδο ποιότητας τους. Αλλά, πώς ορίζουμε το σωστό επίπεδο ποιότητας; Πως μπορεί να μετρηθεί η ποιότητα; Ακολουθούν μερικές απόψεις για το συγκεκριμένο ζήτημα:

- **Πρότυπα (Standards):** Είναι, κοινώς, αποδεκτό ότι, η ανάπτυξη προϊόντων υψηλής ποιότητας σχετίζεται με τα πρότυπα. Τέτοια πρότυπα μπορεί να προέρχονται από το εξωτερικό περιβάλλον της

ομάδας του έργου, όπως το IEEE ή το IOS/IEC, ή και από την ίδια την ομάδα.

Στην πραγματικότητα, προκειμένου να επιτευχθεί υψηλό επίπεδο ποιότητας προϊόντων και διαδικασιών, το λογισμικό θα πρέπει να αναπτύσσεται σύμφωνα με συγκεκριμένα και αξιόπιστα πρότυπα. Η ανάπτυξη του λογισμικού σύμφωνα με πρότυπα, στοχεύει στην επιβεβαίωση ότι, κάθε μέρος του τελικού προϊόντος, υποβάλλεται στην ίδια διαδικασία ανάλυσης, σχεδιασμού, τεκμηρίωσης και προδιαγραφής, όπως κάθε άλλο μέρος. Συνεπώς, το πρότυπο του τελικού προϊόντος είναι ομοιόμορφο και ανταποκρίνεται στις προσδοκίες της οργάνωσης και της κοινότητας, ακόμα κι αν έχει αναπτυχθεί από μία ομάδα.

Ένα σχετικό παράδειγμα, είναι αυτό της προδιαγραφής απαιτήσεων σύμφωνα με το Πρότυπο IEEE-Std. 830-1998:

- **Η ικανοποίηση των αναγκών του πελάτη:** Το λογισμικό πρέπει να ικανοποιεί τον σκοπό της δημιουργίας του και τις ανάγκες των πελατών. Δίνεται έμφαση στον έλεγχο ικανοποίησης του σκοπού του λογισμικού και των αναγκών των πελατών σε κάθε φάση ανάπτυξης του λογισμικού.
- **Ικανοποίηση Ορατών απαιτήσεων:** Το τελικό προϊόν θα πρέπει να ικανοποιεί τις απαιτήσεις του έργου. Για τις ορατές απαιτήσεις, δίνεται έμφαση στον κατάλληλο σχεδιασμό και στην ιχνηλασιμότητα των απαιτήσεων μέσα από τον κώδικα. Επίσης, σημασία έχουν τα χαρακτηριστικά ποιότητας που επιλέγονται. Σύμφωνα με αυτά θα επιβεβαιώνεται ότι ο κώδικας ικανοποιεί τις απαιτήσεις. Η πτυχή της επιβεβαίωσης της καταλληλότητας του προϊόντος σε σχέση με τις απαιτήσεις του έργου όμως, δεν είναι η μόνη πτυχή που πρέπει να διερευνηθεί. Θα πρέπει να επιβεβαιωθούν και άλλα χαρακτηριστικά ποιότητας ή άλλες μη λειτουργικές απαιτήσεις.
- **Ικανοποίηση Μη-Ορατών απαιτήσεων:** Παρόλο που οι Ορατές απαιτήσεις μπορεί να ικανοποιούνται, θα πρέπει να επιβεβαιώνεται και η ικανοποίηση των Μη-Ορατών απαιτήσεων, προκειμένου να διασφαλίζεται η διαδικασία ελέγχου ποιότητας. Ο χρόνος και το

κεφάλαιο όμως, είναι περιορισμένοι πόροι. Όπως γίνεται αντιληπτό, πολλές φορές, δεν είναι εφικτή η επιτυχία υψηλού επιπέδου ποιότητας σε όλες τις απαιτήσεις του έργου. Ο Μηχανικός Απαιτήσεων θα πρέπει να αποφασίσει «ποιά» χαρακτηριστικά ποιότητας ή άλλες μη λειτουργικές απαιτήσεις είναι σημαντικές και αξίζει τον κόπο να ικανοποιηθούν.

Η ποιότητα ενός έργου δεν μπορεί να αξιολογηθεί άμεσα, εκτός αν το έργο επιθεωρείται από κάποιον, όσον αφορά στα ποιοτικά χαρακτηριστικά του. Σε αυτή την περίπτωση, το κύριο πρόβλημα είναι το γεγονός ότι, κατά την αξιολόγηση, θα ληφθεί υπόψη, μόνο η προσέγγιση του συγκεκριμένου ατόμου που θα επιθεωρεί το έργο. Για την αντιμετώπιση τέτοιων ζητημάτων, οι Μηχανικοί Λογισμικού, αναπτύσσουν μοντέλα ανάλυσης των χαρακτηριστικών που προσδιορίζουν την ποιότητα του έργου. Ένα παράδειγμα τέτοιου μοντέλου είναι το ISO standard ISO – 9126, το οποίο αναπαρίσταται γραφικά στο σχήμα 2.1 (15) (16).

2.3 Η Τεχνολογία Λογισμικού ως Διαδικασία

Όπως αναφέρθηκε και παραπάνω, προκειμένου να ελεγχθεί η ποιότητα ενός προϊόντος λογισμικού, εξετάζοντας τα (ποιοτικά) χαρακτηριστικά του, θα πρέπει να εφαρμοστούν διαδικασίες και μέθοδοι του πεδίου της Τεχνολογίας Λογισμικού. Επομένως:

Η ποιότητα του τελικού προϊόντος εξαρτάται από την ποιότητα των διαδικασιών και των μεθόδων της Τεχνολογίας Λογισμικού.

Θεωρώντας ως φάσεις του έργου το σύνολο των διαδικασιών και των μεθόδων αυτών, τότε οι φάσεις μπορούν να οριστούν, να μετρηθούν, να είναι διαχωρίσιμες και να βελτιστοποιηθούν.

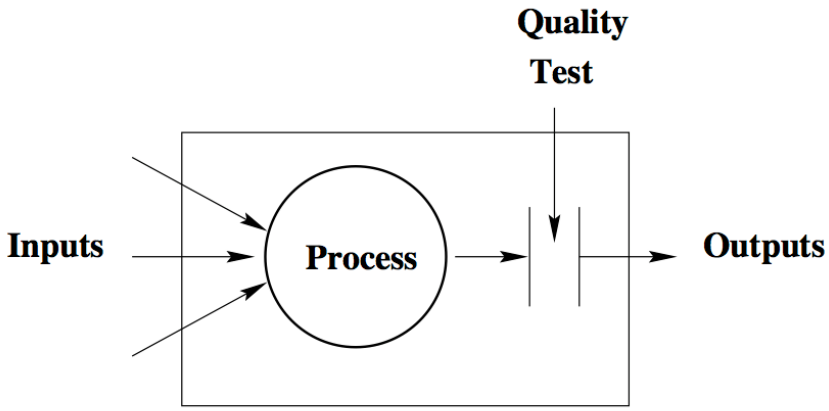
Μία πολύ σημαντική παρατήρηση που έκανε ο Royce είναι ότι, ανεξάρτητα από την διαδικασία που επιλέγεται για την ανάπτυξη του εκάστοτε λογισμικού, η διαδικασία αυτή θα περιλαμβάνει, οπωσδήποτε, τις εξής φάσεις (17):

1. Ανάλυση Απαιτήσεων: Είναι η δραστηριότητα που μπορεί να επαναλαμβάνεται συχνά, κατά την πορεία του έργου ανάπτυξης λογισμικού. Για την εκτέλεση αυτής της δραστηριότητας θα πρέπει να προσδιοριστούν οι λειτουργίες, οι περιορισμοί, οι ικανότητες και άλλες πληροφορίες, σχετικές με το σύστημα. Τέλος, όλες αυτές οι

πληροφορίες, θα πρέπει να οργανωθούν με ακριβή και σαφή τρόπο. Αυτή είναι η φάση της ανάλυσης, ως ένα μέρος της γενικότερης μηχανικής διεργασίας.

2. Σχεδιασμός: Είναι η δραστηριότητα σύνθεσης μίας λογικής δομής του συστήματος, με την οποία να ικανοποιούνται όλες οι απαιτήσεις. Η διεργασία του σχεδιασμού μετουσιώνει τις απαιτήσεις σε μία αναπαράσταση του συστήματος, ώστε να είναι εφικτός ο ποιοτικός έλεγχος και ο έλεγχος καταλληλότητας, πριν ξεκινήσει η υλοποίηση του κώδικα του λογισμικού. Οι τρόπος σχεδιασμού του συστήματος είναι πολύπλευρος. Για παράδειγμα, υπάρχουν αρχιτεκτονικές προσεγγίσεις συστημάτων, αλγοριθμικές προσεγγίσεις, προσεγγίσεις ιεραρχικού ελέγχου του συστήματος και της ροής της πληροφορίας.
3. Υλοποίηση: Είναι η δραστηριότητα με την οποία το προϊόν της φάσης του σχεδιασμού μετατρέπεται, παίρνοντας μορφή αναγνωρίσιμη από τον υπολογιστή-μηχανή. Η φάση ης υλοποίησης και του ελέγχου είναι οι πιο «χειροπιαστές» φάσεις της Μηχανικής Λογισμικού.
4. Έλεγχος: Είναι ένα σύνολο δραστηριοτήτων που αποσκοπούν στην ανίχνευση και ανάδειξη αστοχιών του συστήματος. Η φάση του ελέγχου είναι απαραίτητο να επαναλαμβάνεται καθ' όλη την διάρκεια του έργου και σε διαφορετικά επίπεδα όπως: έλεγχος μονάδας, έλεγχος ολοκλήρωσης, έλεγχος συστήματος και έλεγχος αποδοχής.
5. Συντήρηση: Είναι η διαδικασία, μέσω της οποίας, το σύστημα εξελίσσεται, διορθώνεται και βελτιώνεται. Η φάση της συντήρησης μπορεί να περιλαμβάνει και να αφορά νέες απαιτήσεις του συστήματος όπως, απαιτήσεις σχεδιασμού, υλοποίησης, ή ελέγχου.

Γενικά, ένα έργο ανάπτυξης πληροφοριακού συστήματος αποτελείται από *φάσεις*, κάθε μία από τις οποίες αποτελείται από επιμέρους δραστηριότητες. Η κάθε *φάση* έχει εισόδους, εξόδους και απαιτεί έλεγχο ποιότητας, όπως φαίνεται στο σχήμα 2.2



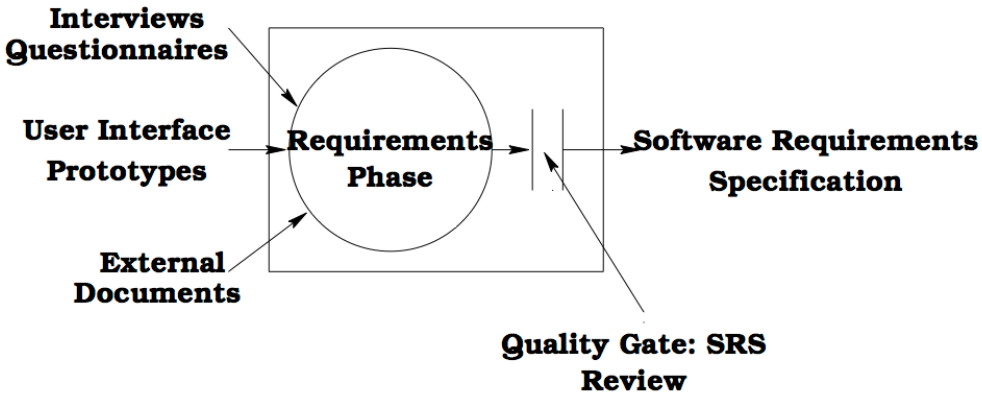
Σχήμα 2.2: Σχηματική απεικόνιση μίας τυπικής φάσης, έργου ανάπτυξης πληροφοριακού συστήματος.

Παράδειγμα 2.1 Το πρότυπο *IEEE Std. 830-1998*, που αφορά σε πρακτικές προδιαγραφής απαιτήσεων έργων ανάπτυξης λογισμικού, προσδιορίζει 8 ιδιότητες άσκησης ελέγχου ποιότητας: *(1) ορθότητα, (2) Πληρότητα, (3) Συνέπεια, (4) Ιχνηλασιμότητα, (5) Επαληθευσιμότητα, (6) Δυνατότητα παραμετροποίησης, (7) Σαφήνεια και Περιεκτικότητα της Προδιαγραφής και (8) Ταξινόμηση Απαιτήσεων.*

Προκειμένου να ασκηθεί έλεγχος στην προδιαγραφή απαιτήσεων του έργου, θα πρέπει να αναπτυχθεί η σχετική αναφορά (SRS review). Επομένως, για την εξακρίβωση του επιπέδου ποιότητας της φάσης των απαιτήσεων, θα πρέπει να εξεταστεί, αν η αναφορά SRS, πληρεί τις 8 ιδιότητες του προτύπου *IEEE Std. 830-1998.*

Γενικά, η σειρά των φάσεων, η επιλογή των χαρακτηριστικών και του τρόπου άσκησης ελέγχου ποιότητας και οι δραστηριότητες που συνθέτουν τη κάθε *φάση*, όπως αυτή παρουσιάστηκε παραπάνω, συνολικά, αποτελούν *το μοντέλο της συνολικής διαδικασίας, ή* ***το μοντέλο του έργου***. Στη σχετική βιβλιογραφία προτείνονται διαφορετικά *μοντέλα έργων*, για *διαφορετικές συνθήκες*. Ακολουθούν μερικά παραδείγματα (3):

- Μοντέλο Καταρράκτη



Σχήμα 2.3: Πιθανή έκδοση φάσης μοντέλου καταρράκτη

- Μοντέλα Προτυποποίησης
- Το Μοντέλο RAD (Rapid Applications Development)
- Τα Εξελικτικά Μοντέλα
  1) Το Σπειροειδές Μοντέλο
  2) Το Αυξητικό Μοντέλο
  3) Το Μοντέλο Ταυτόχρονης Ανάπτυξης
- Το Μοντέλο Agile
- Το Μοντέλο RUP (Rational Unified Process)

Κάθε ένα από τα παραπάνω μοντέλα, έχει τις δικές του ιδιαιτερότητες. Για παράδειγμα, το Σπειροειδές Μοντέλο συνδυάζει το Μοντέλο Προτυποποίησης με πτυχές του Μοντέλου Καταρράκτη, προκειμένου να μειωθεί ο κίνδυνος προδιαγραφής λανθασμένων ή μη αποσαφηνισμένων απαιτήσεων (17).

2.4 Η Μηχανική Απαιτήσεων στον Κύκλο Ζωής του Έργου

Οι περισσότερες διαδικασίες της Τεχνολογίας Λογισμικού χρησιμοποιούν την ανάλυση των απαιτήσεων, ή αλλιώς την διαδικασία ανάλυσης του εκάστοτε προβλήματος. Σε αυτό το μέρος της εργασίας θα γίνει μία προσπάθεια ανάδειξης των διαδικασιών της Μηχανικής Λογισμικού και του ρόλου που διαδραματίζουν οι απαιτήσεις σε αυτές (3).

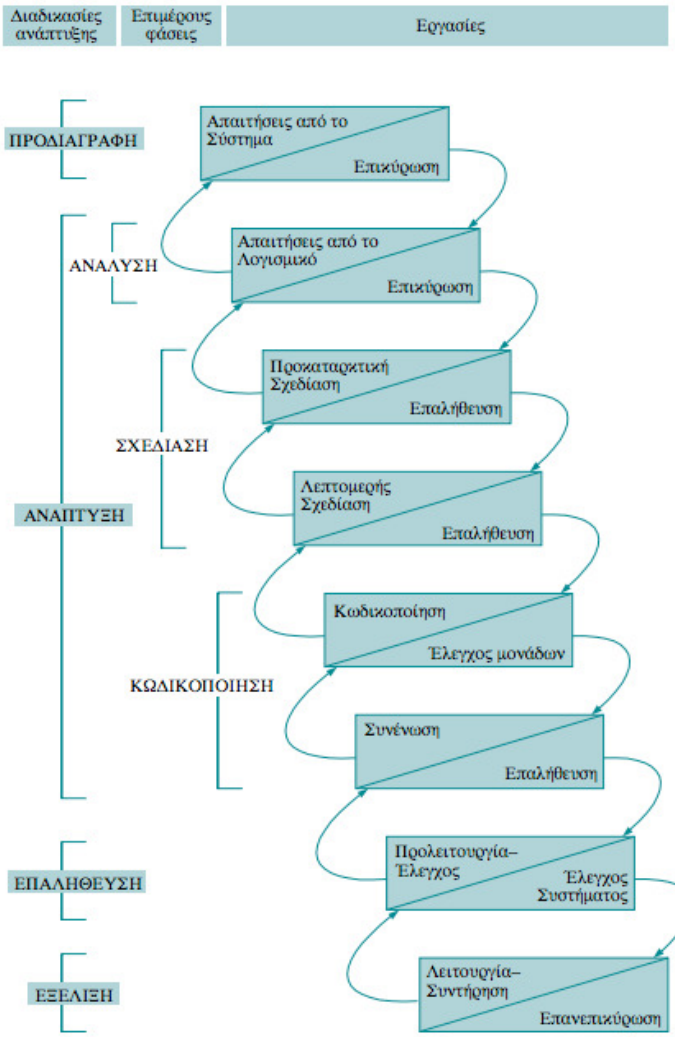
2.4.1 Το Μοντέλο του Καταρράκτη

Το Μοντέλο του Καταρράκτη είναι ένα πρότυπο διαδικασιών που πρότεινε ο Royce (18). Ο Royce, στην πραγματικότητα, δεν ήθελε να ορίσει το μοντέλο αυτό, ως μία συνολική διαδικασία ανάπτυξης λογισμικού. Αυτό που ο ίδιος

ήθελε να αναδείξει, ήταν ότι σε κάθε έργο ανάπτυξης λογισμικού, θα πρέπει να περιλαμβάνονται τουλάχιστον οι φάσεις της ανάλυσης, του σχεδιασμού, της υλοποίησης, του ελέγχου και της συντήρησης. Το σύνολο αυτών των φάσεων και των παραλλαγών τους, έγινε και αποτελεί μέχρι και σήμερα την βάση των διαδικασιών ενός έργου. Παραλλαγή σε αυτές τις διαδικασίες, μπορεί να θεωρηθεί, η εισαγωγή διαδικασιών και μεθόδων που να επιτρέπουν την ανατροφοδότηση πληροφοριών μεταξύ των φάσεων (Σχήμα 2.4), ή η εισαγωγή προσεγγίσεων διαχείρισης «Baseline», για εσωτερική επανάληψη αυστηρών διεργασιών διαχείρισης διαμορφώσεως, κατά την εκάστοτε φάση.

Το πρωταρχικό Μοντέλο του Καταρράκτη δεν επιτρέπει την εκκίνηση μίας φάσης, αν η προηγούμενη δεν έχει ολοκληρωθεί πλήρως. Στην πραγματικότητα, αυτό δεν συμβαίνει πάντα. Για παράδειγμα, η αρχιτεκτονική σχεδίαση μπορεί να ξεκινήσει, ακόμη και πριν ολοκληρωθεί η φάση των απαιτήσεων. Οι υποστηρικτές του μοντέλου αυτού ισχυρίζονται ότι, παρέχει αποτελεσματική υποστήριξη έργων, τα οποία έχουν γίνει σαφώς και ακριβώς κατανοητά, ενώ προσθέτουν ότι το Μοντέλο του Καταρράκτη είναι κατάλληλο για εκτίμηση κόστους και παρακολούθηση των φάσεων του έργου (3).

Με την ενεργοποίηση διαδικασιών ανάδρασης μεταξύ των φάσεων του συγκεκριμένου μοντέλου, μεταγενέστερες φάσεις μπορούν να μεταδίδουν πληροφορίες σε προηγούμενες και το αντίθετο, προάγοντας έτσι την αποτελεσματικότητα του έργου συνολικά. Για παράδειγμα, σε περίπτωση που ένας τεχνικός περιορισμός δεν έχει προσδιοριστεί σαφώς ή ελεγχθεί κατάλληλα, είναι απαραίτητο να επαναληφθούν οι διαδικασίες που αφορούν στον προσδιορισμό και στον έλεγχό του. Επιπλέον, προκειμένου να βελτιώνεται η αυτοπεποίθηση των ειδικών, παρατηρώντας τα προϊόντα των φάσεων, η προτυποποίηση θα μπορούσε να εισάγεται και στην φάση των απαιτήσεων ή του σχεδιασμού (3).



Σχήμα 2.4 Το Μοντέλο του Καταρράκτη με δυνατότητα ανατροφοδότησης πληροφοριών των φάσεων μεταξύ τους (19)

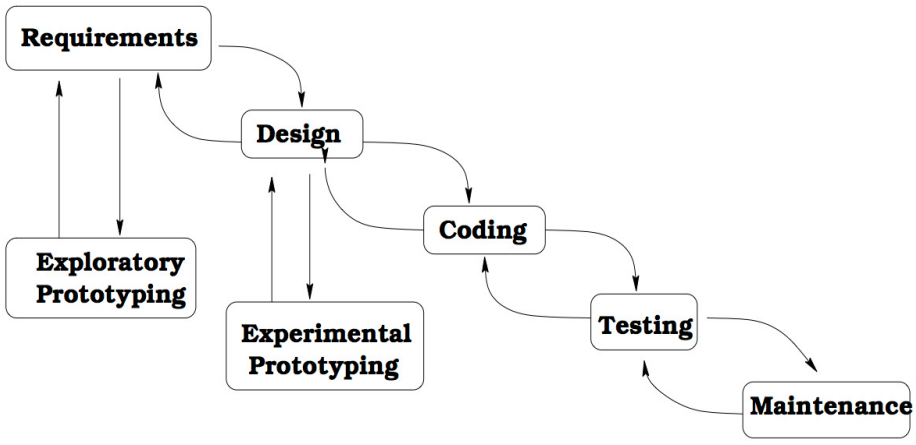
Οι επικριτές του Μοντέλου του Καταρράκτη θεωρούν ότι, το συγκεκριμένο μοντέλο δεν λαμβάνει υπόψη τους τεχνολογικούς και γενικότερους κινδύνους ενός έργου ανάπτυξης λογισμικού. Επιπλέον, υποστηρίζουν ότι, δεν επιτρέπει τον κατάλληλο αριθμό επαναλήψεων, για την αποτελεσματικότερη κατανόηση ενός έργου που εξελίσσεται κατά την διάρκεια που ένα γενικότερο έργο «διαδραματίζεται». Η μειωμένη ανάδραση που παρέχει το μοντέλο, όσον αφορά στον πελάτη είναι αρκετά σημαντικό χαρακτηριστικό του συγκεκριμένου μοντέλου. Ως αποτέλεσμα, μπορεί να περάσει αρκετά μεγάλο χρονικό διάστημα, προκειμένου, ο πελάτης, να δει ένα μέρος του συστήματος να λειτουργεί. Αυτό σημαίνει ότι, οι απαιτήσεις του έργου, για το οποίο θα υιοθετηθεί το Μοντέλο του Καταρράκτη, θα πρέπει να παραμένουν σχετικά σταθερές κατά την διάρκεια του έργου. Συμπερασματικά, αν το έργο δεν απαιτεί σημαντικές και συχνές επαναλήψεις των διαδικασιών και σχετίζεται με ένα πολύ καλά

ορισμένο και αποσαφηνισμένο πεδίο, τότε το Μοντέλο του Καταρράκτη είναι κατάλληλο (3).

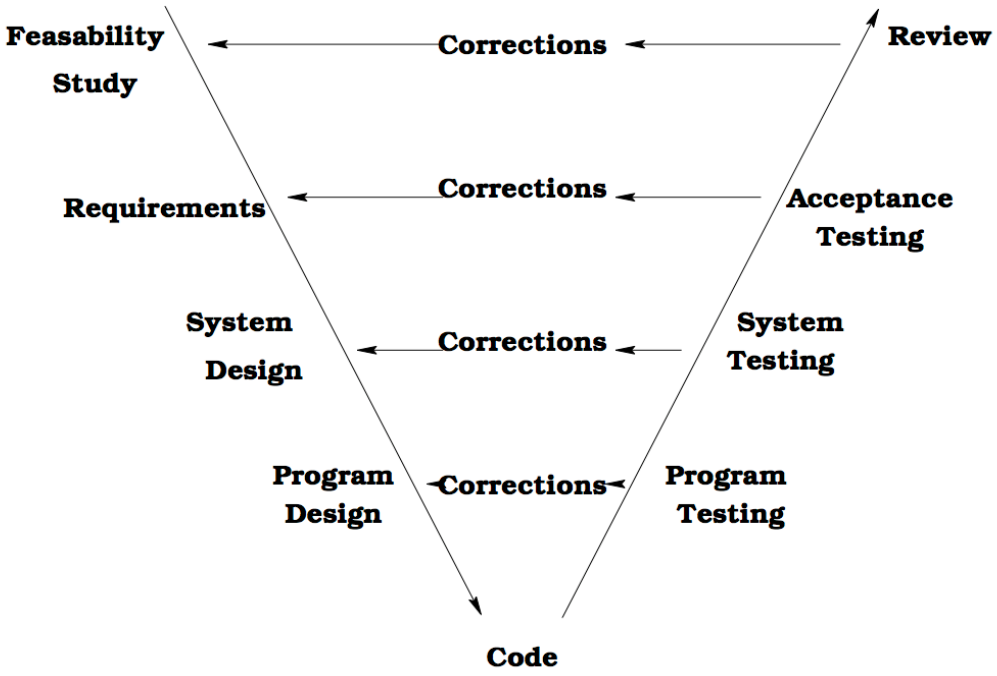
Πρέπει να σημειωθεί ότι, κάθε μία από τις κύριες φάσεις του Μοντέλου του Καταρράκτη αποδίδει κάποιες εξόδους-αποτελέσματα, όπως φαίνεται στο σχήμα 2.5. Οι αρχικές φάσεις του έργου, δεν σχετίζονται μόνο με το, «τί» πρέπει να γίνει, αλλά και με το, «τί» είναι εφικτό να γίνει"(3).

Μία άλλη έκδοση - παραλλαγή του Μοντέλου του Καταρράκτη είναι το **Μοντέλο "V"** (Σχήμα 2.6), το οποίο υιοθετείται σε περιπτώσεις έργων, με απαιτήσεις αυξημένου επιπέδου ασφαλείας, ορθότητας και βεβαιότητας. Στο Σχήμα 2.6 φαίνεται καλύτερα, πως οι απαιτήσεις δεν επηρεάζουν μόνο την επαλήθευση της διαδικασίας ανάπτυξης του έργου, αλλά και την επαλήθευση του συστήματος και την αποδοχή του (3).

Σημειώνεται ότι είναι σημαντικό να αφιερώνεται προσπάθεια στην εξόρυξη, στον καθορισμό και στην επικύρωση των απαιτήσεων, όταν ακόμα το έργο βρίσκεται σε αρχικά στάδια. Έτσι επιτυγχάνεται η αποσαφήνιση του προβλήματος και η εκμαίευση ενός σταθερού συνόλου απαιτήσεων (3).



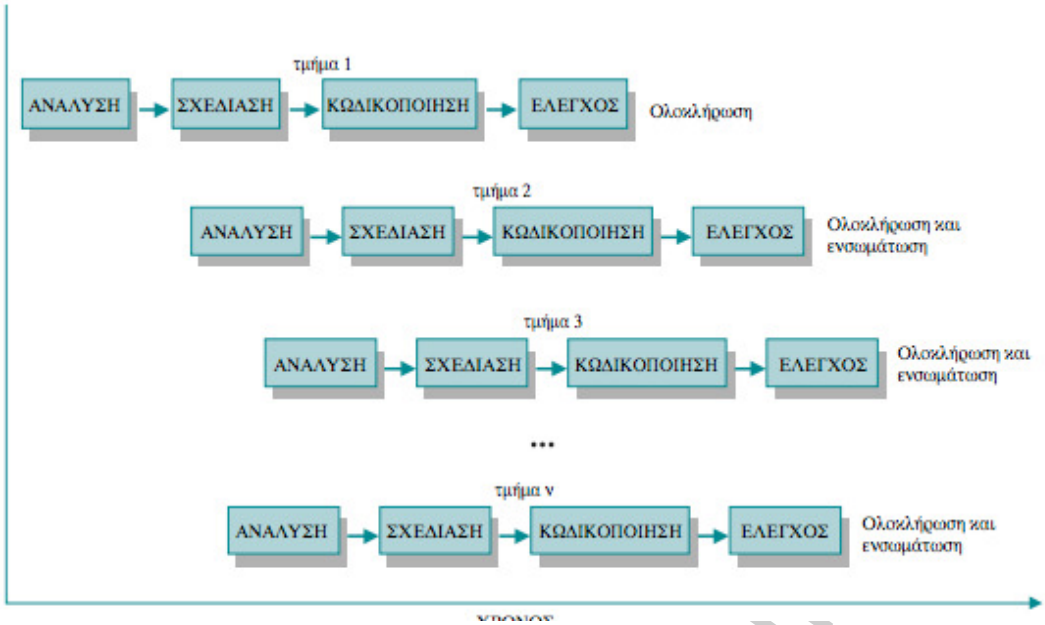
Σχήμα 2.5: Το Μοντέλο του Καταρράκτη



Σχήμα 2.6: Το Μοντέλο "V" για έργα ανάπτυξης λογισμικού

2.4.2 Το Αυξητικό Μοντέλο

Το Αυξητικό Μοντέλο διαχωρίζει το έργο σε ένα σύνολο προσαυξήσεων που βασίζονται είτε στην λειτουργικότητα ή σε άλλα κριτήρια. Συνήθως, κάθε προσαύξηση (ως μέρος του έργου-συστήματος), παραδίδεται στον πελάτη αφού ολοκληρωθεί. Το μοντέλο αυτό, απαιτεί μία ενιαία φάση απαιτήσεων, που επακολουθείται από προσαυξήσεις, κάθε μία από τις οποίες αποτελείται από φάσεις σχεδιασμού, υλοποίησης, ελέγχου και συντήρησης. Όπως έχει αναφερθεί και παραπάνω, η ανάλυση των απαιτήσεων θα πρέπει να πραγματοποιείται κατά την έναρξη του έργου, όπως προτείνεται και από το Μοντέλο Καταρράκτη. Στην πραγματικότητα όμως, η επανειλημμένη αλληλεπίδραση των χρηστών με το έργο, δημιουργεί νέες απαιτήσεις για το έργο, με αποτέλεσμα να επηρεάζονται και οι απαιτήσεις και ο σχεδιασμός, ακόμα και πρόσφατων προσαυξήσεων (3).



Σχήμα 2.7: Το Αυξητικό Μοντέλο, έργων ανάπτυξης λογισμικού (19)

Όσο οι σταδιακά επαυξημένες παραδόσεις του έργου προσδίδουν μεγαλύτερη ευελιξία στο έργο, από αυτή που προσδίδει το μοντέλο του καταρράκτη και μειώνει τους κινδύνους του συνεχώς μεταβαλλόμενου περιβάλλοντος, η συνολική διαδικασία θα πρέπει να διαχειρίζεται αποτελεσματικά. Αν οι απαιτήσεις μεταβάλλονται, είναι απαραίτητο να καταγράφονται οι απαιτήσεις που αφορούν, αντίστοιχα, σε κάποια προσαύξηση. Το ιδανικό θα ήταν, να επιβεβαιώνεται η αποτελεσματική και υψηλού επιπέδου ποιότητας διαχείριση των διαδικασιών διαμόρφωσης πριν την έναρξη της προσαυξητικής ανάπτυξης (3).

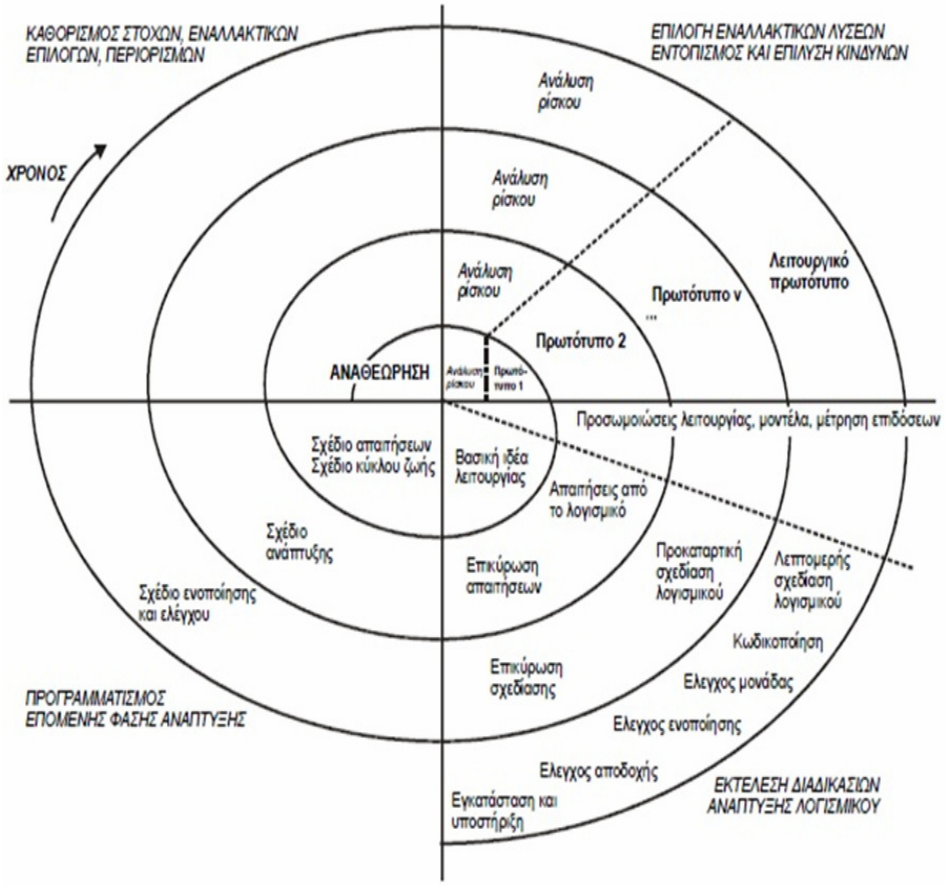
2.4.3 Το Σπειροειδές Μοντέλο

Πολλά έργα καθοδηγούνται από τον κίνδυνο. Το μοντέλο που πρότεινε ο Boehm (20)στοχεύει στον έλεγχο και τη διαχείριση των κινδύνων των έργων, κατά την διάρκεια της ανάπτυξης. Το συγκεκριμένο μοντέλο προσφέρει ευκαιρίες κατανόησης και αποσαφήνισης του προβλήματος, μέσα από την διαδικασία ανάπτυξης του συστήματος σταδιακά-αυξητικά.

Το σπειροειδές μοντέλο ξεκινά, αρχικά, με αξιολόγηση κινδύνων ή με προτυποποίηση για την αξιολόγηση του επιπέδου του συστήματος και παράγει ένα έγγραφο (concept of operations), στο οποίο περιγράφεται, πως το σύστημα θα πρέπει να λειτουργεί, σε υψηλό επίπεδο ολοκλήρωσης. Για την διευκόλυνση της διαδικασίας, θα πρέπει το σύνολο απαιτήσεων που εξάγεται και αφορά στο σύστημα, να είναι πλήρες. Με την μέθοδο της επανάληψης, το σπειροειδές

μοντέλο θα ολοκληρώσει την διαδικασία ανάπτυξης. Πρέπει να σημειωθεί ότι, κάθε επανάληψη εμπεριέχει διαδικασίες ανάλυσης κινδύνων και προτυποποίησης, προκειμένου να προσδιοριστεί η εφικτότητα των εναλλακτικών και κατόπιν ο σχεδιασμός, η υλοποίηση και ο έλεγχος (3).

Το σπειροειδές μοντέλο, όπως απεικονίζεται στο Σχήμα 2.8 επιτρέπει ολοκληρωτική επαναξιολόγηση της κατεύθυνσης του έργου μετά από κάθε σπείρα. Αυτό σημαίνει ότι οι απαιτήσεις μπορούν να επαναξιολογηθούν επίσης. Το ζήτημα που προκύπτει με αυτό το μοντέλο, αφορά στη διαχείριση των απαιτήσεων. Είναι απαραίτητο να εξακριβώνεται ότι, οι νέες ιδέες που οδηγούν σε νέες απαιτήσεις συμφωνούν και μπορούν να συμβαδίζουν με τις αρχικές.

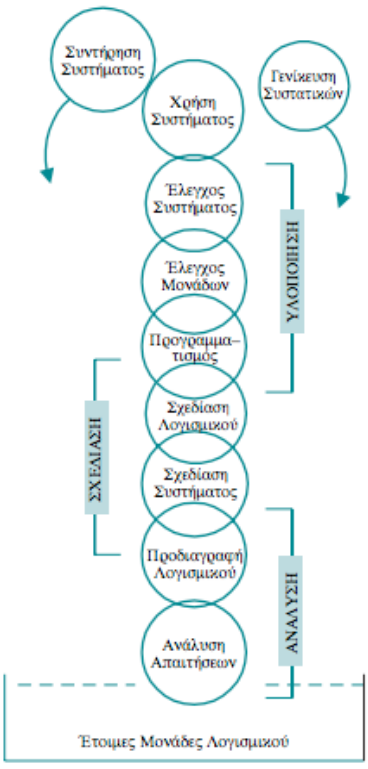


Σχήμα 2.8: Το Σπειροειδές Μοντέλο (19)

2.4.4 Το μοντέλο του Πίδακα

Αρκετά μοντέλα κύκλου ζωής που έχουν προταθεί αποτελούν παραλλαγές αυτών που αναφέρθηκαν, τα χαρακτηριστικά των οποίων υιοθετούνται από τις μεθοδολογίες ανάπτυξης. Οι πρώτες προσεγγίσεις του θέματος με βάση την αντικειμενοστρεφή (object-oriented) τεχνολογία διαφοροποίησαν το παραπάνω σχήμα βασιζόμενες σε δύο ιδιαίτερα γνωρίσματά της: πρώτον, ότι οι

έννοιες «ανάλυση - σχεδίαση – κωδικοποίηση» έρχονται στο αντικειμενοστρεφές 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 2.9. 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 αντικειμενοστρεφή λογική, ήταν δε αρκετά επίκαιρη κατά την έκρηξη ενδιαφέροντος για την αντικειμενοστρεφή τεχνολογία στα τέλη της δεκαετίας του 80 και στις αρχές της δεκαετίας του 90 (19).



Σχήμα 2.9: Το Μοντέλο του Πίδακα *(19)*

2.4.5 Το Εξελικτικό Μοντέλο RUP

Το αντικειμενοστρεφές μοντέλο Rational Unified Process έχει αναπτυχθεί από τους δημιουργούς της αντικειμενοστρεφούς γλώσσας μοντελοποίησης UML, τους Booch, Rumbaugh και Jacobson. Βασίζεται στο μοντέλο του καταρράκτη, αλλά θεωρεί ότι η ανάλυση απαιτήσεων, ο σχεδιασμός, η υλοποίηση και ο έλεγχος δεν συμπίπτουν με χρονικές φάσεις αλλά αντιπροσωπεύουν τμήματα διαδικασίας τα οποία λαμβάνουν χώρα σε διάφορες χρονικές φάσεις. Σύμφωνα με την Rational Unified Process, ένα σύστημα θα πρέπει να δημιουργείται πρώτα ως ένα πρωτότυπο, να ελέγχεται, να αναλύεται και να αξιολογείται και τέλος να εκλεπτύνεται σε επόμενες επαναλήψεις του κύκλου ζωής. Σύμφωνα με τη χρονική σειρά που διεξάγονται οι διαδικασίες, το μοντέλο ορίζει 4 χρονικές φάσεις: έναρξη, εκπόνηση μελέτης, κατασκευή και μετάβαση (21).

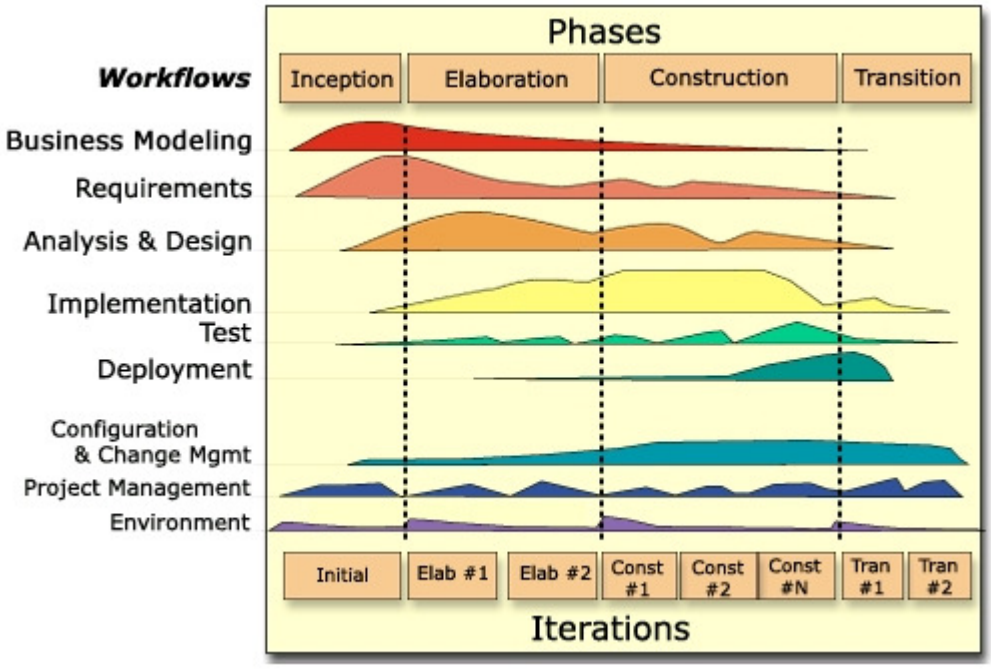
Ο κύκλος ζωής λογισμικού προτείνεται να είναι επαναληπτικός. Η ανάπτυξη δηλαδή να προχωρεί σε μια σειρά επαναλήψεων μέχρι να εξελιχθεί το τελικό προϊόν.

Η διαδικασία Unified είναι δομημένη σε δύο διαστάσεις (21):

1. Χρόνο : Χωρισμός του κύκλου ζωής σε φάσεις και επαναλήψεις.
2. Τμήματα διαδικασίας: Καλά ορισμένες ενέργειες.

Η δόμηση ενός έργου σε σχέση με το χρόνο ακολουθεί τις εξής φάσεις που έχουν σχέση με το χρόνο (21):

1. Έναρξη (Inception): Καθορίζει την προοπτική του έργου.
2. Εκπόνηση μελέτης (Elaboration): Σχεδίαση των απαιτούμενων δραστηριοτήτων και πόρων. Προσδιορισμός των χαρακτηριστικών και σχεδίαση της αρχιτεκτονικής.
3. Κατασκευή (Construction): Ανάπτυξη του προϊόντος σε μια σειρά βηματικών επαναλήψεων.
4. Μετάβαση (Transition): Προμήθεια του προϊόντος στην κοινότητα χρηστών (παραγωγή, διανομή, εκπαίδευση).



Σχήμα 2.10: Το Εξελικτικό Μοντέλο RUP (22)

2.4.6 Κωδικοποίηση και διόρθωση (code and fix)

Στο συγκεκριμένο μοντέλο δε προηγείται ανάλυση της έναρξης υλοποίησης του λογισμικού. Με την ανάληψη ενός έργου, ξεκινάει άμεσα η υλοποίησή του. Δεν υπάρχει συγκεκριμένο σχέδιο ανάπτυξης του λογισμικού. Γράφεται ο κώδικας του προγράμματος, μέχρι να ολοκληρωθεί η ανάπτυξή του (23).

Το συγκεκριμένο μοντέλο είναι χρήσιμο στην ανάπτυξη μικρών έργων πληροφορικής και δεν υπάρχουν διοικητικά έξοδα επιβάρυνσης του προϋπολογισμού του έργου. Η χρήση του σε μεγάλα έργα είναι επικίνδυνη, καθώς δεν υπάρχουν ορόσημα (milestones), ενώ υπάρχει δυσκολία στην επικοινωνία και έλλειψη μηχανισμών ελέγχου (23).

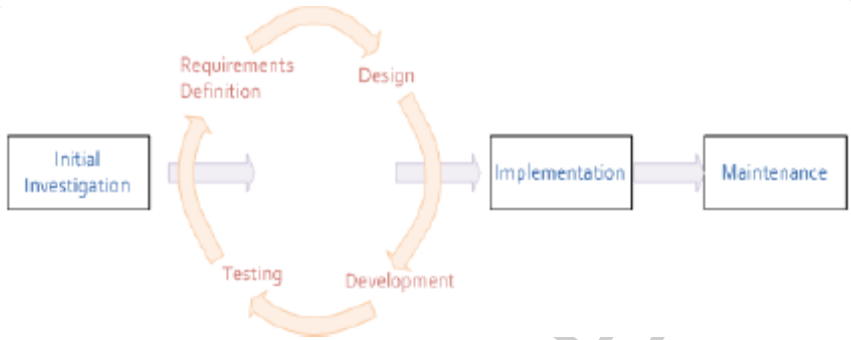
Το μοντέλο αποτελείται από δυο (2) φάσεις. Η πρώτη φάση του μοντέλου είναι η συγγραφή του κώδικα, και η δεύτερη φάση είναι η φάση διόρθωσης των σφαλμάτων στον κώδικα.. Το μεγαλύτερο μειονέκτημα του μοντέλου είναι η δυσκολία διορθώσεων με τη πάροδο του χρόνου – καθώς το μέγεθος της εφαρμογής μεγαλώνει (23).

Παρόλα αυτά το μοντέλο κωδικοποίησης και διόρθωσης χρησιμοποιείται στις μέρες μας, όταν υπάρχει πίεση για γρήγορη παράδοση προϊόντος, χωρίς να υπάρχει διαθέσιμος χρόνος για ανάλυση και σχεδίαση (23).

2.4.7 Ταχεία ανάπτυξη λογισμικού (rapid application development)

Η ταχεία ανάπτυξη λογισμικού (RAD) είναι μια μεθοδολογία που χρησιμοποιεί την προτυποποίηση στην επαναληπτική ανάπτυξη του λογισμικού

Το μοντέλο RAD προωθεί τη συνεργατικότητα. Οι εμπλεκόμενοι χρήστες συμμετέχουν ενεργά στην ανάπτυξη των προτύπων και στις δοκιμές των περιπτώσεων χρήσης (24).



Σχήμα 2.11: Μοντέλο ταχείας ανάπτυξης λογισμικού

Υπάρχουν πέντε βασικά είδη μεθοδολογιών ταχείας ανάπτυξης λογισμικού (24):

Agile: Σπάσιμο ενός έργου σε μικρότερα υποέργα. Η ανάπτυξη πραγματοποιείται με μικρές επαναλήψεις του κύκλου ανάπτυξης ενός λογισμικού. Περιλαμβάνει ελάχιστη καταγραφή και τεκμηρίωση απαιτήσεων, χωρίς να υπάρχει κάποιο πλαίσιο που καθοδηγεί τα βήματα υλοποίησης.

Extreme Programming (XP): Δεν υπάρχει στάδιο σχεδίασης. Προκειμένου να μειωθεί το κόστος, οι απαιτήσεις περιγράφονται με λίγα λόγια, περιλαμβάνονται γρήγοροι κύκλοι σπιράλ (Ανάλυση – Σχεδίαση – Υλοποίηση - Έλεγχος) στην ανάπτυξη του λογισμικού.

Joint Application Development (JAD): Από κοινού ανάπτυξη του λογισμικού με τον τελικό χρήστη ή πελάτη με τη συμμετοχή του κατά τη διάρκεια της σχεδίασης και υλοποίησης του λογισμικού.

Lean Development (LD): Υιοθετείται η άποψη ότι: «το 80% της δουλειάς σήμερα είναι καλύτερο από το 100% της δουλειάς αύριο». Η Lean Development στοχεύει στη γρήγορη ανάπτυξη ενός προτύπου με ελάχιστη λειτουργικότητα, περιορίζοντας τη διάρκεια της ανάλυσης και σχεδίασης του λογισμικού.

Scrum: Η ανάπτυξη λαμβάνει χώρα με μια σειρά από σύντομες επαναλήψεις των σταδίων ανάπτυξης λογισμικού και η πρόοδος μετριέται καθημερινά. Το μοντέλο είναι καταλληλότερο για μικρά έργα διότι δεν υιοθετεί κάποιο πλαίσιο για την ανάπτυξη του λογισμικού.

2.5 Παράδειγμα Ανάλυσης Απαιτήσεων σε έργο πληροφορικής

Αν θεωρήσουμε πως μια εταιρία έχει αναλάβει την ανάπτυξη συστήματος (ιστότοπος) κράτησης δωματίων σε ξενοδοχεία. Για την επιτυχή έκβαση της ανάπτυξης του συστήματος, σημαντικό ρόλο θα έχει η ανάλυση απαιτήσεων που θα εφαρμοστεί.

Τα βήματα ανάπτυξης του συστήματος θα είναι τα εξής:

1. **Εντοπισμός σημαντικών εμπλεκόμενων (key stakeholders):** Στο παράδειγμά μας μπορούσε να είναι η εταιρία που έχει ζητήσει την ανάπτυξη του λογισμικού, χρήστες στο διαδίκτυο που κάνουν χρήση σχετικών υπηρεσιών κ,λ,π. Οι σημαντικοί εμπλεκόμενοι μπορεί να είναι διασκορπισμένοι σε διάφορα τμήματα της εταιρίας που ζήτησε την ανάπτυξη του προϊόντος.

2. **Εντοπισμός των πραγματικών απαιτήσεων:** Ζητάμε από όλους τους εμπλεκόμενους να μας αναφέρουν τι θέλουν από το έργο, τι περιμένουν από αυτό και ποιες είναι οι απαιτήσεις τους από το νέο προϊόν. Ο εντοπισμός των πραγματικών απαιτήσεων μπορεί να γίνει με τη βοήθεια συνεντεύξεων με τους αντιπροσώπους της εταιρίας, με διαμοιρασμό ερωτηματολογίων στο διαδίκτυο σε πιθανούς μελλοντικούς χρήστες ή συναντήσεις συγκέντρωσης (focus groups). Δε θα πρέπει να ξεχνάμε πως κάθε εμπλεκόμενος βλέπει το έργο κρατήσεων από τη δικιά του οπτική γωνιά και θα πρέπει να προσπαθήσουμε να γεφυρώσουμε τις συγκρουόμενες απαιτήσεις. Πιθανόν, θα μπορούσε να αναπτυχθεί και ένα πρότυπο κρατήσεων δωματίων σε ξενοδοχεία ή υπάρχουσα αντίστοιχα λογισμικά προκειμένου να διευκολυνθεί η διαδικασία εντοπισμού των πραγματικών αναγκών.

3. **Κατηγοριοποίηση απαιτήσεων:** στη συνέχεια για τη διευκόλυνση της ανάλυσης και παρακολούθησης του έργου θα πρέπει να κατηγοριοποιήσουμε τις απαιτήσεις στις εξής κατηγορίες:

a. Λειτουργικές απαιτήσεις: Εδώ θα μπορούσαν να ταξινομηθούν οι διαθέσιμοι πιθανοί τρόποι πληρωμής, τα φίλτρα αναζήτησης στη βάση δεδομένων, κ.λ.π.

b. Μη Λειτουργικές απαιτήσεις: Εδώ θα μπορούσαν να ταξινομηθούν οι φυλλομετρητές στους οποίους θα λειτουργεί σωστά ο δικτυακός τόπος κρατήσεων δωματίων, αν θα λειτουργεί και σε έξυπνα κινητά (smartphones), κ.λ.π

c. Απαιτήσεις συμπεριφοράς: Εδώ θα μπορούσαν να ταξινομηθούν απαιτήσεις σχετικά με τη συμπεριφορά της σελίδας αν δεν έβρισκε καταλύματα, σύμφωνα με τα κριτήρια αναζήτησης του χρήστη (π.χ. παρουσίαση εναλλακτικών περιοχών).

d. Απαιτήσεις απόδοσης: Εδώ θα μπορούσαν να ταξινομηθούν περιορισμοί, όπως το μέγεθος των σελίδων, ο χρόνος φόρτωσης της σελίδας, η σειρά κατάταξης σε μηχανές αναζήτησης.

4. **Ερμηνεία και καταγραφή απαιτήσεων:** Αρχικά θα πρέπει να δούμε «ποιές» από τις απαιτήσεις που καταγράψαμε είναι εφικτό να υλοποιηθούν. Ύστερα θα πρέπει να οριστούν οι προτεραιότητες για το σύστημα (π.χ. μεγαλύτερη προτεραιότητα για αναζήτηση καταλυμάτων με τη βοήθεια χάρτη σε σχέση με αναζήτηση καταλυμάτων με χρήση κειμένου). Θα πρέπει επίσης να γίνει εντοπισμός των επιπτώσεων που θα έχει στις διαδικασίες της εταιρίας η ανάπτυξη του συστήματος (π.χ. περιορισμός τηλεφωνικών κρατήσεων, απολύσεις, προσλήψεις). Στο σημείο αυτό θα πρέπει να υπάρχει συμφωνία μεταξύ όλων των εμπλεκόμενων σχετικά με τις απαιτήσεις του συστήματος (διαπραγματεύσεις).

5. **Υπογραφή συμφωνίας:** Είναι απαραίτητο να υπάρχει έγγραφη συμφωνία για τις απαιτήσεις του συστήματος κρατήσεων, προκειμένου να μην υπάρξουν σημαντικές αλλαγές στις απαιτήσεις από μέρους του πελάτη.

Ανάλογα με το μοντέλο ανάπτυξης λογισμικού που εφαρμόζεται στον οργανισμό πληροφορικής που θα αναπτύξει το έργο, η ανάλυση θα υλοποιηθεί σε διαφορετικά στάδια:

- **Το Μοντέλο του Καταρράκτη**: η ανάλυση θα ολοκληρωνόταν πριν την έναρξη της σχεδίασης του λογισμικού. Μετά τη υπογραφή της συμφωνίας με την εταιρία που ζήτησε το λογισμικό θα ξεκινήσουμε η σχεδίαση του συστήματος, με μικρές αναθεωρήσεις των απαιτήσεων στην πορεία του έργου.

- **Το Αυξητικό Μοντέλο:** Χρησιμοποιώντας το Αυξητικό μοντέλο, η υλοποίηση θα έσπαγε σε τμήματα και η υλοποίηση κάθε υποτμήματος θα γινόταν ξεχωριστά. Κατά την υλοποίηση κάθε υποτμήματος θα υιοθετούσαμε τις αρχές του μοντέλου του Καταρράκτη, χρησιμοποιώντας ως βάση την ανατροφοδότηση από το προηγούμενο τμήμα που αναπτύχθηκε. Πιθανότατα θα δομούσαμε, αρχικά, τον σκελετό του συστήματος κρατήσεων και στην πορεία θα προσθέτονταν οι σχετικές λειτουργίες σε αυτό.

- **Το Σπειροειδές Μοντέλο:** Αρχικά θα γινόταν η καταγραφή των βασικών απαιτήσεων. Με βάση αυτές τις απαιτήσεις θα δημιουργούσαμε ένα πρωτότυπο του συστήματος κρατήσεων, θα εφαρμοζόταν έλεγχος ικανοποίησης των αρχικών απαιτήσεων και πάνω στο πρωτότυπο θα προσδιορίζονταν οι νέες απαιτήσεις που θα αποτελούσαν τη βάση δημιουργίας ενός 2ου πρωτοτύπου κοκ. Για παράδειγμα, στο 1ο πρωτότυπο θα μπορούσε να υπάρχει η δυνατότητα επιλογής ξενοδοχείων με βάση το χάρτη και δυνατότητα εκτέλεσης κράτησης δωματίων. Στο 2ο πρωτότυπο θα μπορούσαν να προστεθούν φίλτρα αναζήτησης με βάση την ονομασία του καταλύματος, έξτρα παροχές, με βάση την απόσταση από τη θάλασσα και το αεροδρόμιο.

- **Το μοντέλο του Πίδακα:** Υιοθετώντας το μοντέλο του Πίδακα, η ακολουθία των ενεργειών μας θα ήταν ίδιες με αυτές του μοντέλου του Καταρράκτη. Η κύρια διαφοροποίηση με το μοντέλο του Καταρράκτη θα ήταν η δυνατότητα επιστροφής σε προηγούμενα στάδια του κύκλου ζωής του λογισμικού. Επίσης, δεν θα πραγματοποιούσαμε εξονυχιστική ανάλυση απαιτήσεων. Με την υιοθέτηση αυτού του μοντέλου θα ήταν δυνατή η κωδικοποίηση της εφαρμογής σε αρχικό στάδιο και στην πορεία, αν υπήρχε δυσκολία στην υλοποίησή του έργου, να

πραγματοποιηθεί η σχετική παραμετροποίηση των αρχικών απαιτήσεων. Χρησιμοποιώντας το μοντέλο του Πίδακα, το πιθανότερο θα ήταν να είχαμε οδηγηθεί στην κωδικοποίηση της εφαρμογής με αντικειμενοστρεφή προγραμματισμό.

- **Το Εξελικτικό Μοντέλο RUP:** Με τη χρήση αυτού του μοντέλου στο αρχικό στάδιο θα είχαμε μια πρώτη εκτίμηση των κινδύνων, του κόστους για την ανάπτυξη του συστήματος κρατήσεων καθώς και μια πρώτη περιγραφή του έργου. Θα χρησιμοποιούνταν διαγράμματα για την αναπαράσταση των ρόλων του συστήματος (πελάτες, ξενοδόχοι, διαχειριστής συστήματος) και των λειτουργιών που θα επιτελούσε, πριν την έναρξη της υλοποίησης.

- **Κωδικοποίηση και διόρθωση:** Υιοθετώντας το συγκεκριμένο μοντέλο, θα ξεκινούσε άμεσα η συγγραφή κώδικα, χωρίς να προηγηθεί ανάλυση απαιτήσεων. Δε θα υπήρχε κάποια σειρά για τα τμήματα κώδικα προς υλοποίηση. Η ομάδα προγραμματιστών θα προχωρούσε στην υλοποίηση του συστήματος κρατήσεων με όποια σειρά έκρινε καλύτερη.

- **Ταχεία ανάπτυξη λογισμικού:** Σε αυτή τη περίπτωση, τα στάδια της ανάλυσης και σχεδίασης θα ήταν μικρά σε μέγεθος και σκοπός θα ήταν η γρήγορη υλοποίηση ενός πρωτοτύπου που θα υποστήριζε τις βασικές λειτουργίες του συστήματος. Στη συνέχεια, με τη βοήθεια της ανατροφοδότησης του πρωτοτύπου θα ακολουθούσαν επαναλαμβανόμενοι κύκλοι ανάπτυξης λογισμικού για την προσθήκη των επιπλέον λειτουργιών του συστήματος κρατήσεων .

3 Εξόρυξη Απαιτήσεων

3.1 Εισαγωγή

Πριν την ανάλυση, σχεδίαση και υλοποίηση των απαιτήσεων ενός συστήματος, θα πρέπει οι απαιτήσεις να συλλέγονται με τη βοήθεια της εξόρυξης απαιτήσεων. Η εξόρυξη απαιτήσεων αποτελεί σημαντικό κομμάτι της διαδικασίας ανάπτυξης ενός λογισμικού. Βασικός σκοπός της είναι το γεφύρωμα του σημασιολογικού χάσματος, όσον αφορά στις απαιτήσεις, μεταξύ των χρηστών και των σχεδιαστών ενός συστήματος(25).

Όπως έχει δηλώσει και ο Steve Jobs (1998) σχετικά με την Εξόρυξη Απαιτήσεων «*Πολλές φορές, οι άνθρωποι δεν γνωρίζουν τι θέλουν μέχρι να τους το δείξεις*». Με τη βοήθεια της Εξόρυξης Απαιτήσεων μπορούν να εντοπιστούν οι πραγματικές ανάγκες του πελάτη, πάνω στις οποίες θα βασιστεί η ανάλυση απαιτήσεων ανάπτυξης λογισμικού (26).

3.2 Εκκίνηση της διαδικασίας Εξόρυξης Απαιτήσεων

Η διαδικασία εξόρυξης απαιτήσεων είναι η διαδικασία κατά την οποία, προσδιορίζονται όλες οι πηγές των απαιτήσεων και ύστερα, χρησιμοποιώντας αυτές, προσδιορίζεται το πρόβλημα που πρέπει να λυθεί, καθώς και τα χαρακτηριστικά του.

Η διαδικασία της εξόρυξης απαιτήσεων είναι εξ' ορισμού επαναληπτική, αφού σε κάθε συνάντηση μεταξύ των εμπλεκόμενων μερών του έργου, συλλέγονται νέες πληροφορίες σχετικές με το έργο και επομένως προσδιορίζεται σαφέστερα το πρόβλημα που χρήζει λύσης.

Κατά τα πρώτα στάδια της διαδικασίας εξόρυξης απαιτήσεων, προτείνεται να προσδιορίζονται τα παρακάτω τέσσερα στοιχεία του έργου :

(Α) Ο σκοπός και οι στόχοι του, καθώς επίσης και το βασικό πρόβλημα που χρήζει υπολογιστικής λύσης,

(Β) το πλαίσιο και το περιβάλλον, εντός του οποίου, το έργο θα αναπτύσσεται,

(Γ) η επίδραση της επιθυμίας του πελάτη να παραλάβει το σύστημα

(Δ) το κίνητρο και οι προσδοκίες του πελάτη.

Πολλές από τις παραπάνω ιδιότητες σχετίζονται με την ποιότητα του έργου. Σε κάθε περίπτωση, ο πελάτης δεν μπορεί να γνωρίζει, ποιές από αυτές τις ιδιότητες, στις οποίες αναφερθήκαμε, είναι σημαντικές για την επιτυχία του συστήματος.

Μετά την αρχική συνέντευξη, το επόμενο βήμα είναι να προσδιοριστεί το κυρίαρχο πρόβλημα και τα μέρη του προβλήματος, που το νέο σύστημα θα κληθεί να λύσει.

Τα έργα Μηχανικής Λογισμικού είναι το αποτέλεσμα μίας ιδέας ή μίας ανάγκης κάποιων ανθρώπων, ή μία ευκαιρία, την οποία κάποιος παρατήρησε. Πριν ξεκινήσει το έργο, θα πρέπει αυτή η ιδέα να πάρει την μορφή πιο «απτής» έννοιας.

Επίσης, απαραίτητο είναι να βρεθεί το κεφάλαιο για την ανάπτυξη του έργου. Αν το λογισμικό πρόκειται να αναπτυχθεί εντός ενός οργανισμού, τότε μέσα από τον ίδιο οργανισμό θα επιβεβαιωθεί η δυνατότητα ανάλωσης του κεφαλαίου για το έργο. Αν το λογισμικό προορίζεται για εμπόριο στην ελεύθερη αγορά, τότε το κεφάλαιο μπορεί να προέρχεται από επιχειρηματικά κεφάλαια, εταιρικές χορηγίες και κρατικές επιχορηγήσεις .

Το επιπλέον ζήτημα που προκύπτει είναι ότι, προκειμένου να αποφασιστεί η ανάλωση ή μη του κεφαλαίου προς αυτό το σκοπό, θα έχει ήδη καταβληθεί προσπάθεια από ένα, πιθανώς, μεγάλο σύνολο φυσικών ή νομικών προσώπων-χαρακτήρων.

Άλλο σημαντικό ζήτημα είναι ο προσδιορισμός των προσώπων-χαρακτήρων του έργου, καθώς επίσης και του μέρους ευθύνης που αντιστοιχεί σε κάθε έναν (27) (28) (29).

Οι απαντήσεις των ερωτήσεων που ακολουθούν, σχετίζονται με τους κινδύνους του έργου, τις απαιτήσεις και την επιλογή μοντέλου-διαδικασίας (30).

1. Τι αποσκοπεί να πετύχει το νέο σύστημα; Τι αποσκοπεί να αντικαταστήσει; Τι είδους σύστημα θα επιθυμούσατε;
2. Ποιοί θα είναι οι χρήστες; Ποιοί θα είναι οι διαχειριστές του συστήματος και ποιοί οι συντηρητές του;
3. Το ήδη υπάρχον σύστημα είναι παρόμοιο; Ποιό είναι το πρόβλημα με το υπάρχον σύστημα; Πώς μπορεί να βελτιωθεί το υπάρχον σύστημα; Σε

ποιά σημεία μπορεί να βελτιωθεί; Γιατί αποφασίστηκε η αυτοματοποίηση; Τι οικονομικά αποτελέσματα αναμένονται;

4. Ποιό θα είναι το (αναμενόμενο) πραγματικό όφελος για εσάς, μετά την εγκατάσταση του νέου συστήματος;
5. Σε τι είδους περιβάλλον θα εγκατασταθεί το νέο σύστημα;
6. Μπορείτε να εκφράσετε παραδείγματα εισόδων και εξόδων του συστήματος; Μπορείτε να παράσχετε αναφορές ή δεδομένα για το υπάρχον σύστημα;
7. Τι πρότυπα χρησιμοποιεί η εταιρία σας; Τι πρότυπα θα επιδρούσαν αποτελεσματικά στο έργο;
8. Τι ιδιαίτερες ιδιότητες αναμένεται να διαθέτει το σύστημα; Ασφάλεια; , αξιοπιστία; , χρηστικότητα; , επιδόσεις;

3.3 Προσδιορισμός των ενδιαφερόμενων μερών

Είναι σημαντικό, για την διαχείριση του έργου, να προσδιορίζονται νωρίς (κατά την διαδικασία) τα ενδιαφερόμενα μέρη. Τα ενδιαφερόμενα μέρη θα αποτελέσουν πηγή απαιτήσεων σχετικά με τους περιορισμούς του έργου, που πρέπει, σύντομα, να προσδιοριστούν.

Τα ενδιαφερόμενα μέρη ενός έργου είναι οι άνθρωποι, οι ομάδες ή οι οργανισμοί που μπορεί να ενδιαφέρονται άμεσα ή έμμεσα για το έργο, ή να τους αναλογεί μέρος ευθύνης του. Συμπεριλαμβάνονται, επίσης, και εκείνοι που μπορεί στο μέλλον να επηρεαστούν από το έργο. Οι κύριες κατηγορίες ενδιαφερόμενων μερών είναι: οι πελάτες στους οποίους θα παραδοθεί το έργο, οι χορηγοί, οι τελικοί χρήστες, οι ειδικοί και οι πελάτες του οργανισμού στον οποίο θα εγκατασταθεί, ή οι πελάτες του οργανισμού που θα πωλούν το λογισμικό. Άλλα πιθανά ενδιαφερόμενα μέρη είναι, οι εμπειρογνώμονες – τεχνικοί, οι προγραμματιστές των διασυνδεδεμένων συστημάτων, οι ρυθμιστικοί φορείς και άλλοι επαγγελματικοί φορείς.

Οι τεχνικές εξόρυξης-εκμαίευσης απαιτήσεων διαφέρουν σε πολλά σημεία μεταξύ τους. Οι Συνεντεύξεις και οι Ομάδες Συγκέντρωσης, η τεχνική Άμεσης Εκτέλεσης και η τεχνική Διαλογισμού αποδίδουν πληροφορίες σχετικές με επιχειρηματικούς κανόνες και πρότυπες διαδικασίες, ηθικούς και νομικούς περιορισμούς, τις επιθυμίες των χρηστών, καθώς και τις συνήθειές τους και τέλος αποδίδουν πληροφορίες σχετικά με το, «τί» πρέπει να αλλάξει, να παρέχεται, να ολοκληρωθεί.

Πιο συντηρητικές τεχνικές, όπως το μοντέλο Οντοτήτων Συσχετίσεων, αποδίδουν πληροφορίες που σχετίζονται με την οργάνωση της εκμαιευμένης πληροφορίας και με τον εντοπισμό αντιφάσεων, ασαφειών και ελλείψεων.

Τέλος, οι καλές μέθοδοι μοντελοποίησης μας επιτρέπουν να κατανοήσουμε, συστηματικά, «πώς» κάθε επιμέρους τμήμα των πληροφοριών, εντάσσεται στη λειτουργία του συνόλου (27) (28) (29).

3.4. Τεχνικές Εξόρυξης Απαιτήσεων

Μετά τη συναίνεση και εξασφάλιση των χρηστών που θα συμμετάσχουν στη διαδικασία εξόρυξης των απαιτήσεων, θα πρέπει να επιλεγούν οι καλύτερες τεχνικές για την αποτελεσματική εξόρυξη απαιτήσεων. Στη συνέχεια παρουσιάζονται οι περισσότερο διαδεδομένες - αντιπροσωπευτικές τεχνικές εξόρυξης δεδομένων (31).

3.4.1 Συνεντεύξεις (Interviews)

Μπορούν να διαδραματίζονται με πελάτες, τελικούς χρήστες, ειδικούς-τεχνικούς ή με οποιοδήποτε άλλο μέρος των εμπλεκόμενων μερών του έργου. Αποδίδουν μεγάλο όγκο πληροφορίας, σχετικά με την προοπτική των χρηστών, όσον αφορά στο πρόβλημα που χρήζει υπολογιστικής λύσης. Η άποψη των ειδικών είναι σημαντική, αν και πολλές φορές υπερεκτιμάται, είναι όμως εξίσου σημαντική και αυτή των χρηστών.

Ιδανικά, οι πληροφορίες που αποδίδονται από μία συνέντευξη θα πρέπει να συμφωνούν με τις πληροφορίες που αποδίδουν και οι άλλες πηγές πληροφοριών. Η συγκεκριμένη εξακρίβωση, προτείνεται να πραγματοποιείται απαραίτητα! Έτσι, θα μειώνονται οι προκαταλήψεις των συνεντευξιαζόμενων. Η τεχνική των συνεντεύξεων είναι ευέλικτη και ενθαρρύνει την συμμετοχή των ενδιαφερόμενων μερών στο έργο, είναι όμως σημαντικό για την αποτελεσματικότητα της διαδικασίας, ο συνεντευκτής να διαθέτει την απαραίτητη, σχετική εμπειρία.

Από την άλλη μεριά, οι συνεντεύξεις, πολλές φορές, αποδίδουν ασαφείς πληροφορίες και κατ' επέκταση, ασαφείς απαιτήσεις. Οι άπειροι συνεντευκτές δυσκολεύονται πολύ στην εκμαίευση της κατάλληλης πληροφορίας και συνήθως λειτουργούν επιλεκτικά, συνομιλώντας με τον εκάστοτε συνεντευξιαζόμενο. Τυχαίνει επίσης συχνά, η άποψη του συνεντευκτή να επηρεάζεται από στερεότυπα χαρακτηριστικά, των απόψεων των συνεντευξιαζόμενων. Τέλος, υπάρχει κίνδυνος, η πληροφορία που εκμαιεύεται,

να μην ανταποκρίνεται στην πραγματικότητα. Η άποψη των συνεντευξιαζόμενων δεν συνάδει απαραίτητα με την πραγματική απαίτηση (32).

3.4.2 Ερωτηματολόγια (Questionnaries)

Το κυριότερο πλεονέκτημα αυτής της τεχνικής είναι η ιδιότητά της, να επιτρέπει την ταχεία συγκέντρωση στοχευόμενων πληροφοριών μεγάλου όγκου. Η τεχνική έχει χαμηλές απαιτήσεις κόστους και η αποστολή των ερωτηματολογίων προς τα ενδιαφερόμενα μέρη μπορεί να γίνει μέσω Ηλεκτρονικού Ταχυδρομείου (e-mail). Η συμπλήρωση του ερωτηματολογίου από τα εμπλεκόμενα μέρη, συνήθως, διαρκεί ελάχιστο χρόνο. Ένα άλλο πλεονέκτημα αυτής της τεχνικής είναι η δυνατότητα που παρέχει, ώστε οι πληροφορίες να συγκεντρώνονται ανώνυμα.

Από την άλλη μεριά, ο σχεδιασμός των ερωτηματολογίων αποτελεί δύσκολη διαδικασία. Επιπλέον, η έλλειψη της άμεσης επικοινωνίας δημιουργεί προφανή προβλήματα. Συνήθως, η ανταπόκριση σε τέτοιες τεχνικές δεν είναι μεγάλη και οι πληροφορίες θα πρέπει να επανελέγχονται ποιοτικά και επανειλημμένα, κατά τη διάρκεια του έργου (32).

3.4.3 Συναντήσεις Συγκέντρωσης (Focus Groups)

Πρόκειται για σχετικά άτυπες συναντήσεις μεταξύ ενός συνόλου ανθρώπων (μεταξύ πέντε και δέκα ατόμων) που αντιπροσωπεύουν μία ή περισσότερες πτυχές – προοπτικές των απαιτήσεων. Ένα παράδειγμα θα μπορούσε να είναι μία συνάντηση συγκέντρωσης σχετική με τον σχεδιασμό της διεπαφής χρήστη (user interface) και στην οποία να παρευρίσκονται τα εμπλεκόμενα, εκείνα, μέρη που πρέπει να χρησιμοποιούν το σύστημα.

Οι συναντήσεις συγκέντρωσης αποδίδουν αυθόρμητες αντιδράσεις των μελών που παρευρίσκονται. Είναι μία αποτελεσματική τεχνική εκμαίευσης πληροφοριών, οι οποίες αντιπροσωπεύουν, πραγματικά, την άποψη των παρευρισκομένων. Η “ατμόσφαιρα” της συνάντησης μπορεί να προωθεί την έμπνευση ιδεών και την ώθηση των πραγματικών απαιτήσεων, ώστε να βγουν στην επιφάνεια.

Σε περιπτώσεις επίλυσης ζητημάτων και λήψης αποφάσεων, η συγκεκριμένη τεχνική δεν είναι αποτελεσματική. Οι συναντήσεις συγκέντρωσης δεν αναδεικνύουν τις λεπτομέρειες των απαιτήσεων, ενώ, συχνά, δεν λαμβάνουν υπόψη, τους περιορισμούς του συστήματος (32).

3.4.4 Άμεση Παρατήρηση (Direct Observation)

Η μέθοδος της παρατήρησης των χρηστών κατά την εκτέλεση των καθηκόντων τους στο ήδη υπάρχον σύστημα, αποδίδει πληροφορία, σχετική με τις απαιτήσεις του νέου συστήματος.

Η συγκεκριμένη μέθοδος είναι κατάλληλη για την κατανόηση των επιμέρους διεργασιών σε κάθε κατάσταση. Αποτελεί καλό ξεκίνημα, λίγο πριν τη φάση της εκτενέστερης έρευνας. Το κόστος διεξαγωγής της είναι μικρό και αποτελεί ένα πολύτιμο εργαλείο συλλογής πληροφοριών που αντιπροσωπεύουν την πραγματική κατάσταση.

Η παρατήρηση των χρηστών θα πρέπει να πραγματοποιείται με προσοχή καθώς οι χρήστες μπορεί να επηρεαστούν από την παρουσία του παρατηρητή, ειδικά σε περιπτώσεις συνεχούς παρουσίας του τελευταίου, κατά την εκτέλεση των καθηκόντων των χρηστών (Hawthorne effect). Επιπλέον, είναι απαραίτητος ο σχεδιασμός της διαδικασίας προτού αυτή ξεκινήσει, προκειμένου ο παρατηρητής να έχει αποκτήσει μία αντικειμενική θεώρηση της εκάστοτε διεργασίας (32).

3.4.5 Άμεση Εκτέλεση (Apprenticing)

Μία βελτιωμένη εκδοχή της παραπάνω τεχνικής είναι η τεχνική άμεσης εκτέλεσης, κατά την οποία, ο Μηχανικός Απαιτήσεων εκτελεί τις διεργασίες που οι χρήστες θα καλούνται να εκτελούν με την προσαρμογή του νέου συστήματος. Με τη συγκεκριμένη τεχνική, ο Μηχανικός Απαιτήσεων αποκτά άποψη με εσωτερική προοπτική των διεργασιών του νέου συστήματος.

Είναι ιδιαίτερα δύσκολο να μπορέσει ένας αναλυτής να επικοινωνήσει πλήρως με τις διεργασίες κατά αυτόν το τρόπο. Παρόλα αυτά, αν ο αναλυτής αποτελεί αναπόσπαστο μέλος της ομάδας ανάπτυξης, τότε η συγκεκριμένη τεχνική μπορεί να αποτελέσει πολύ χρήσιμο εργαλείο κατανόησης των διεργασιών που το σύστημα θα πρέπει να εκτελεί και να υποστηρίζει (32).

3.4.6 Διαλογισμός (Brainstorming)

Η τεχνική του διαλογισμού μπορεί να αποδώσει πληθώρα ιδεών σε σύντομο χρονικό διάστημα. Το περιβάλλον στο οποίο συγκεντρώνονται όσοι συμμετέχουν στην διαδικασία διαδραματίζει σημαντικό ρόλο και πρέπει να είναι απλό, καθημερινό και όχι τόσο επίσημο. Αρχικά, τίθεται το ζήτημα για το οποίο οι παρευρισκόμενοι θα κληθούν να εκφράσουν τις ιδέες-απόψεις τους. Ύστερα, οι συμμετέχοντες επικοινωνούν τις ιδέες τους, εντός ενός ορισμένου

χρονικού περιορισμού (διαστήματος). Όλες οι ιδέες των συμμετεχόντων καταγράφονται και μετά το τέλος της συνάντησης αξιολογούνται, ώστε να επιλεγούν εκείνες οι ιδέες που επιλύουν πιο αποτελεσματικά το ζήτημα. Κατά τις συναντήσεις διαλογισμού υπάρχουν κανόνες. Ένας από αυτούς, είναι ότι δεν επιτρέπεται καμία συζήτηση επί των ιδεών, κατά τη διάρκεια της διαδικασίας. Οι σκέψεις εκφράζονται, καταγράφονται, δεν μπορεί όμως κανείς να τις σχολιάσει, τουλάχιστον μέχρι το πέρας της διαδικασίας.

Η τεχνική του διαλογισμού είναι μία αρκετά δημιουργική επιλογή για τη φάση των απαιτήσεων, ειδικά σε περιπτώσεις δύσκολων προβλημάτων ή αποφάσεων. Από την άλλη μεριά, δεν ενδείκνυται για αναλύσεις καταστάσεων, ούτε για λήψη αποφάσεων (32).

3.4.7 Επιθεώρηση εγγράφων (Inspecting Documents)

Η τεχνική επιθεώρησης εγγράφων αποτελεί χρήσιμο εργαλείο για τη συλλογή απαιτήσεων, όπου ήδη υπάρχοντα συστήματα αναπροσαρμόζονται και επισημοποιούνται. Οι πληροφορίες που συλλέγονται με άλλες τεχνικές έχουν συνήθως ποιοτικά χαρακτηριστικά, όμως και τα δεδομένα είναι, εξίσου, αξιόπιστα. Τα έγγραφα αποτελούν καλή πηγή πληροφορίας σε περιπτώσεις διεργασιών ελέγχου αξιοπιστίας ήδη συγκεντρωμένων πληροφοριών, από άλλες πηγές.

Άτυπα συστήματα και άτυπες διαδικασίες, ή εκείνα τα συστήματα και οι διαδικασίες που δεν καταγράφονται, δεν αφομοιώνονται στην πραγματική λειτουργικότητα της κατάστασης. Επίσης, ο αναλυτής απαιτήσεων θα πρέπει να παρατηρεί με προσοχή, αν τα έγγραφα είναι τόσο πρόσφατα ώστε να αντιπροσωπεύουν την υφιστάμενη κατάσταση του συστήματος.

Η επιθεώρηση των εγγράφων μπορεί να εξελιχθεί σε χρονοβόρα διαδικασία. Σε τέτοιες περιπτώσεις, οι αναλυτές θα πρέπει να είναι επιλεκτικοί. Επίσης, είναι πιθανό, τα δεδομένα να μην είναι σε μορφή που να διευκολύνει την άμεση χρήση τους ή ακόμη να μην υπάρχουν πληροφορίες σχετικές με το ζήτημα (32).

3.4.8 Κοινή Ανάπτυξη Εφαρμογών (JAD)

Μία συνεδρία κοινής ανάπτυξης εφαρμογών είναι μία δομημένη συνεδρία, στην οποία συμμετέχουν οι πελάτες, οι σχεδιαστές και άλλοι ειδικοί. Σκοπός της συνεδρίας είναι ο προγραμματισμός, η κατανόηση και η εκκίνηση του σχεδιασμού του συστήματος. Προϋπόθεση για τη διεξαγωγή μίας τέτοιας συνεδρίας είναι η δημιουργία ενός σαφή ισχυρισμού του σκοπού και των

στόχων της συνεδρίας. Η οικονομία της συνεδρίας εξαρτάται, κυρίως, από ένα άτομο που έχει τον ρόλο του διαμεσολαβητή, ώστε οι συμμετέχοντες να παραμένουν συγκεντρωμένοι. Σύμφωνα με τους κανόνες, κατά τη διάρκεια της διαδικασίας, επιτρέπεται να υπάρχουν παρατηρητές, υπό την προϋπόθεση ότι παραμένουν σιωπηλοί.

Οι συνεδρίες κοινής ανάπτυξης εφαρμογών προωθούν την ανάμειξη των χρηστών από τα πρώτα στάδια του έργου, διευκολύνοντας έτσι, την συλλογή των απαιτήσεων των χρηστών και τη διαδικασία σχεδιασμού. Οι χρήστες δεσμεύονται με το έργο και διευκολύνονται οι διαδικασίες επίλυσης ζητημάτων. Η παρουσία ενός διαμεσολαβητή αμερόληπτου ως προς τα ζητήματα, μπορεί να διευκολύνει τη διαδικασία συνολικά. Γενικώς, σε όσα έργα χρησιμοποιείται αποτελεσματικά η τεχνική των συνεδρίων κοινής ανάπτυξης εφαρμογών, επιτυγχάνεται η αποτελεσματικότερη επικοινωνία των διάφορων εμπλεκόμενων μερών (32).

3.4.9 Ενδοσκόπηση (Introspection)

Σύμφωνα με την τεχνική της ενδοσκόπησης ο Μηχανικός Απαιτήσεων ξεκινάει την εξόρυξη απαιτήσεων με βάση ό, τι πιστεύει εκείνος πως οι χρήστες χρειάζονται από το σύστημα. Παρά το γεγονός ότι χρησιμοποιείται από τους περισσότερους αναλυτές σε κάποιο βαθμό, αυτή η τεχνική θα πρέπει να χρησιμοποιείται μόνο ως σημείο εκκίνησης για άλλες τεχνικές εξόρυξης απαιτήσεων. Η ενδοσκόπηση μπορεί να είναι αποτελεσματική όταν ο αναλυτής δεν είναι μόνο γνώστης του τομέα του νέου συστήματος μα και ειδικός στις επιχειρησιακές διαδικασίες που εκτελούνται από τους χρήστες. Σε κάθε περίπτωση η τεχνική της ενδοσκόπησης θα πρέπει να χρησιμοποιείται σε συνδυασμό με άλλες τεχνικές εξόρυξης δεδομένων και ποτέ μόνη της (32).

3.4.10 Ταξινόμηση καρτών (card sorting)

Η τεχνική αυτή απαιτεί από τους χρήστες που συμμετέχουν στη διαδικασία εξόρυξης απαιτήσεων να ταξινομήσουν μια σειρά από κάρτες που αναπαριστούν τις οντότητες του συστήματος σε ομάδες σύμφωνα με τη δική τους άποψη. Επιπλέον, οι συμμετέχοντες είναι υποχρεωμένοι να εξηγήσουν το σκεπτικό της επιλογής τους. Είναι σημαντικό κατά τη τεχνική ταξινόμησης καρτών, όλες οι οντότητες του συστήματος να αναπαριστώνται από μια κάρτα.

Για την εφαρμογή της μεθόδου αυτής είναι απαραίτητο να είναι επαρκώς κατανοητό το σύστημα τόσο στον αναλυτή όσο και τον χρήστη. Σε αντίθετη

περίπτωση θα πρέπει να προηγηθεί η χρήση άλλων τεχνικών που θα διευκολύνουν την κατανόηση του συστήματος και τον εντοπισμό οντοτήτων. Σε αυτή την τεχνική οι κάρτες χρησιμοποιούνται για να αναθέτουν αρμοδιότητες για τους χρήστες και τις συνιστώσες του συστήματος (31).

3.4.11 Προτυποποίηση (prototyping)

Η παροχή στους χρήστες πρωτότυπων του συστήματος ,για την υποστήριξη της έρευνας των πιθανών λύσεων, είναι ένας αποτελεσματικός τρόπος για τη συγκέντρωση λεπτομερών πληροφορίών. Η προτυποποίηση έχει αποτελεσματικότητα όταν χρησιμοποιείται σε συνδυασμό με άλλες τεχνικές εξόρυξης, όπως συνεντεύξεις και JAD. Τα πρωτότυπα είναι συνήθως υλοποιημένα υποστηρίζοντας βασικές απαιτήσεις του συστήματος ή αποτελούν παραδείγματα παρόμοιων συστημάτων. Αυτή η τεχνική είναι χρήσιμη κατά την ανάπτυξη διεπαφών ανθρώπου-υπολογιστή, ή όταν οι ενδιαφερόμενοι δεν είναι εξοικειωμένοι με τις διαθέσιμες λύσεις.

Βασικό μειονέκτημα της προτυποποίησης είναι το υψηλό κόστος και ο απαιτούμενος χρόνος ανάπτυξης προτύπων. Ωστόσο, ένα πλεονέκτημα της χρήσης πρωτοτύπων είναι πως ενθαρρύνει τους ενδιαφερόμενους φορείς, και πιο συγκεκριμένα οι χρήστες, να διαδραματίσουν ενεργό ρόλο στον καθορισμό των απαιτήσεων (31).

4 Περιπτώσεις Χρήσης (Use Cases)

4.1 Εισαγωγή

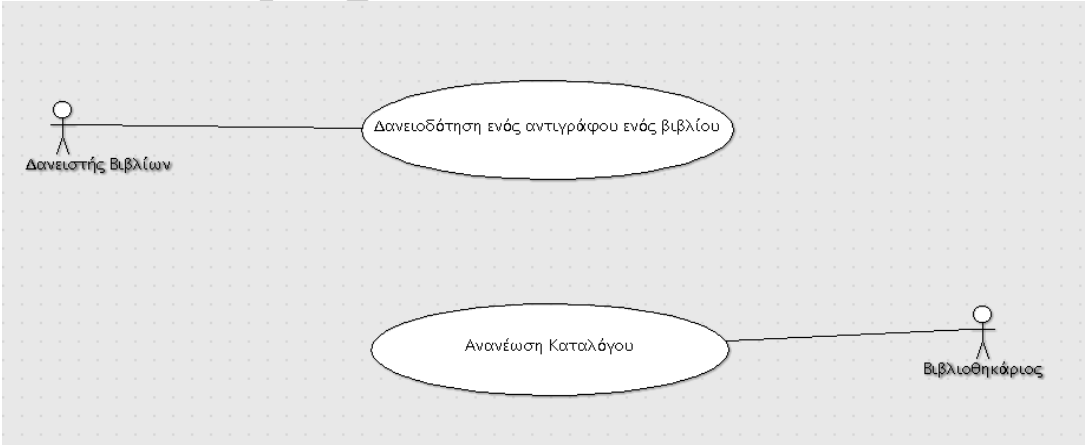
Οι περιπτώσεις χρήσης είναι μία σημειολογία, υψηλού επιπέδου αφαίρεσης, για τον καθορισμό της συμπεριφοράς ενός συστήματος και του τρόπου αλληλεπίδρασης του συστήματος με τους χρήστες και με άλλα συστήματα.

Μία περίπτωση χρήσης φανερώνει τις δραστηριότητες, ή τις διενέργειες που ένα σύστημα πρέπει να φέρει εις πέρας, προκειμένου να ικανοποιεί τις ανάγκες των πελατών. Οι περιπτώσεις χρήσης είναι τόσο γενικές ώστε, κάθε περίπτωση χρήσης μπορεί να καλύψει έναν αριθμό από διαφορετικά σενάρια. Αυτά τα σενάρια διαμορφώνουν τη συμπεριφορά του συστήματος όσον αφορά στην αλληλεπίδρασή του με εξωτερικές οντότητες. Με αυτό τον τρόπο, οι περιπτώσεις χρήσης καθορίζουν (33):

1. Τα όρια του συστήματος,
2. Τον τρόπο με τον οποίο, οι «άλλες οντότητες», αλληλεπιδρούν με το σύστημα
3. Τις διεργασίες που, το νέο σύστημα, θα πρέπει να εκτελεί

Στη γλώσσα των περιπτώσεων χρήσης, οι «άλλες οντότητες» μπορούν να είναι :

- Άνθρωποι
- Άλλες εξωτερικές συσκευές
- Άλλα υπολογιστικά συστήματα



Σχήμα 4.1 Παράδειγμα Περίπτωσης Χρήσης «Δανειστής βιβλίων»

Συνεχίζοντας την ανάλυση παρατηρείται ότι, δεν θα επωφεληθούν όλοι οι ρόλοι (ακόμη και όσοι συμμετέχουν στο σενάριο) από την περίπτωση χρήσης, παρόλο

που μπορεί να εμπλέκονται. Ένας ρόλος που απολαμβάνει αξία από μία περίπτωση χρήσης θα αναφέρεται ως «Δικαιούχος» . Ο προσδιορισμός των δικαιούχων μίας περίπτωσης χρήσης έχει αξία, όσον αφορά στην υπαγόρευση της ανάγκης για ένα συγκεκριμένο σύνολο λειτουργιών ή συμπεριφορών. Σε περίπτωση που ένα σύστημα δεν έχει δικαιούχους τότε πρόκειται για μία κατάσταση, κατά την οποία, κάποιος πιστεύει ότι θα ήταν καλό να αυτοματοποιηθεί κάποια λειτουργία. Είναι όμως πιθανό, μετά την αυτοματοποίηση της, να μην απολαμβάνει αξία κανένα από τα εμπλεκόμενα μέρη.

Τους ρόλους, μπορούν να διαδραματίζουν άνθρωποι, συσκευές, ή και εξωτερικά υπολογιστικά συστήματα. Πιο ειδικά, θα πρέπει να προσδιοριστούν οι περιπτώσεις χρήσης, σε τέτοιο επίπεδο αφαίρεσης, ώστε να αποσαφηνιστεί κάθε ενδεχόμενη ανακρίβεια, όσον αφορά στους ρόλους. Το ποντίκι και το πληκτρολόγιο, για παράδειγμα, δεν θα πρέπει να λαμβάνονται υπόψη ως εξωτερικές συσκευές. Σε περιπτώσεις χρήσης όπου συμπεριλαμβάνονται πολλές, εκτενείς και μη σχετικές λεπτομέρειες, είναι πιθανό, το μοντέλο που θα αναπτυχθεί να περιπλέκεται χωρίς ουσιαστικό λόγο. Κατευθύνσεις, σχετικά με τις πληροφορίες που πρέπει να συμπεριλαμβάνονται σε περιπτώσεις χρήσης και συγκεκριμένα, όσον αφορά τις εξωτερικές συσκευές των περιπτώσεων χρήσης, δίνονται παρακάτω. Προτείνονται τρείς περιπτώσεις, κατά τις οποίες, οι αλληλεπιδράσεις με εξωτερικές συσκευές θα πρέπει να αναδεικνύονται (33):

- Πάντα
- Όταν η εξωτερική συσκευή, ή το εξωτερικό σύστημα εκκινεί κάποια διαδικασία-επαφή
- Όταν η εξωτερική συσκευή ή το εξωτερικό σύστημα είναι ο δικαιούχος

Στη συγκεκριμένη εργασία, οι περιπτώσεις χρήσης χρησιμοποιούνται για (33):

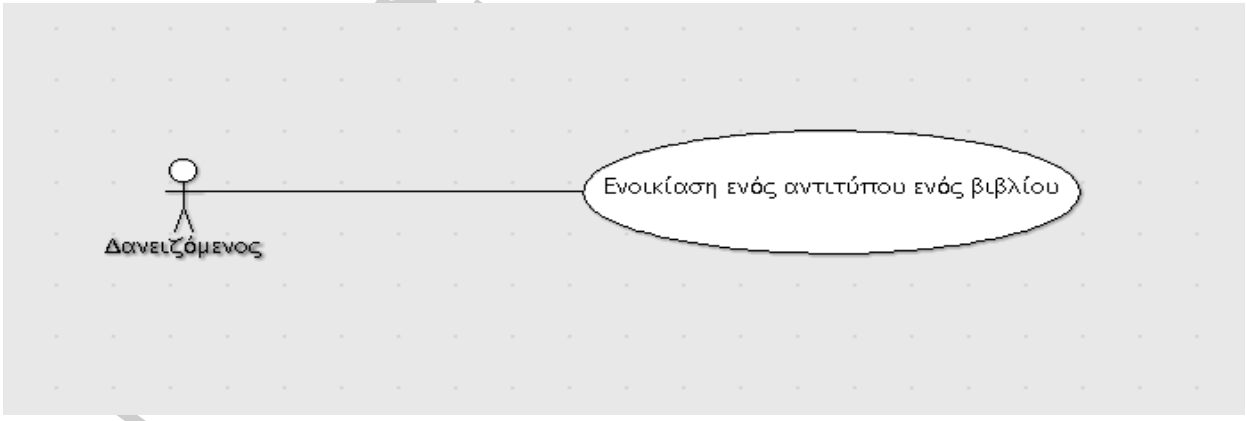
- Τη βελτίωση των απαιτήσεων
- Την ενσωμάτωση νέων απαιτήσεων
- Τον άτυπο έλεγχο της συνοχής και της πληρότητας των απαιτήσεων

Σε αυτό το σημείο, θα πρέπει να τονιστεί ότι, το ζήτημα της εκμαίευσης προέχει της προσπάθειας μοντελοποίησης, αφού, όπως είναι προφανές, η μοντελοποίηση των απαιτήσεων στηρίζεται σε μεγάλο βαθμό στις απαιτήσεις που εκμαιεύονται (33).

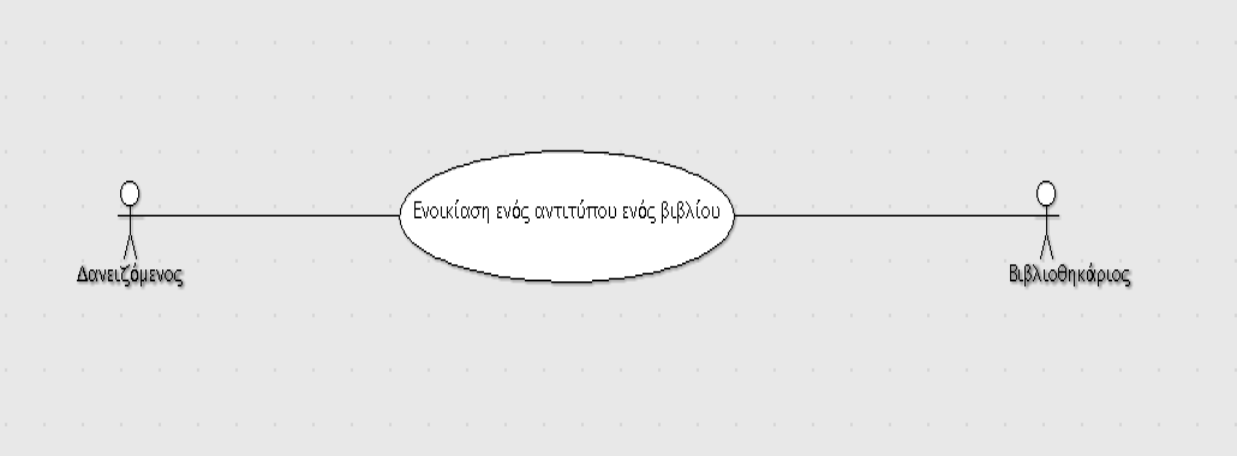
Όσον αφορά στο παράδειγμα της βιβλιοθήκης που προαναφέρθηκε, υπάρχουν τα εξής σενάρια (9):

1. Η Μαρία δανείζεται το τρίτο αντίτυπο του βιβλίου «Πόλεμος και Ειρήνη», όταν δεν οφείλει να επιστρέψει κάποιο άλλο από προηγούμενη ενοικίαση. Η ενοικίαση του βιβλίου επιτρέπεται και ο κατάλογος ανανεώνεται άμεσα.
2. Ο Γιάννης δοκιμάζει να δανειστεί το πρώτο αντίτυπο του βιβλίου «Ελ Γκρέκο», όμως η αίτησή του απορρίπτεται. Ο ίδιος, οφείλει να επιστρέψει βιβλία από προηγούμενες ενοικιάσεις και μάλιστα φαίνεται ότι έχει νοικιάσει τον μέγιστο αριθμό βιβλίων που επιτρέπεται, ενώ δεν έχει επιστρέψει κανένα.
3. Ο Δημήτρης δοκιμάζει να δανειστεί το δεύτερο αντίτυπο του βιβλίου «Το δαχτυλίδι του Αυτοκράτορα». Η βιβλιοθηκάριος παρατηρεί ότι, ο Δημήτρης έχει καθυστερήσει να επιστρέψει ένα βιβλίο από προηγούμενη ενοικίαση. Παρόλα αυτά και επειδή η συγκεκριμένη απόφαση αφορά στο ρόλο του βιβλιοθηκάριου, η άδεια δίνεται, η ενοικίαση επικυρώνεται και ο κατάλογος της βιβλιοθήκης ενημερώνεται άμεσα.

Τα δύο πρώτα σενάρια αφορούν στη περίπτωση χρήσης του σχήματος 4.2. Το τρίτο σενάριο προϋποθέτει την ανάμειξη του ρόλου «βιβλιοθηκάριος». Το Σχήμα 4.3 απεικονίζει το τρίτο σενάριο.



Σχήμα 4.2 Παράδειγμα Περίπτωσης Χρήσης «Δανειστής βιβλίων» χωρίς απαίτηση ειδικής άδειας δανεισμού



Σχήμα 4.3 Παράδειγμα Περίπτωσης Χρήσης «Δανειστής βιβλίων» με απαίτηση ειδικής άδειας δανεισμού

Παρατηρείται ότι, τα σενάρια μπορούν να γίνουν πολύπλοκα. Οι περιπτώσεις χρήσης έχουν παραλλαγές και μπορούν να ενσωματώνουν πληθώρα απαιτήσεων. Είναι, επομένως, απαραίτητη η δόμηση των περιπτώσεων χρήσης με τις εξής προϋποθέσεις (9):

- Παροχή λεπτομερειακών σεναρίων για κάθε περίπτωση χρήσης
- Καθορισμός συνθηκών, υπό τις οποίες εφαρμόζεται κάθε σενάριο

Αυτές οι πληροφορίες μπορούν να παρέχονται είτε γραπτώς (π.χ. με τη χρήση διαγράμματος ροής γεγονότων - flow of events diagram- όπως φαίνεται στο Σχήμα 4.4), είτε με τη χρήση διαγραμμάτων δραστηριότητας της γλώσσας UML.

Περίπτωση Χρήσης: *Όνομα Περίπτωσης Χρήσης*

Ρόλοι

(Λίστα των ρόλων)

Έναυσμα (Trigger)

(Λίστα των συνθηκών υπό τις οποίες ενεργοποιείται η περίπτωση χρήσης)

Διαδικασία

(Προσδιορισμός των βημάτων για τη διεξαγωγή της περίπτωσης χρήσης)

Παραλλαγές

(Λίστα των παραλλαγών των βημάτων και προσδιορισμός των περιπτώσεων κατά τις οποίες συμβαίνει κάθε παραλλαγή)

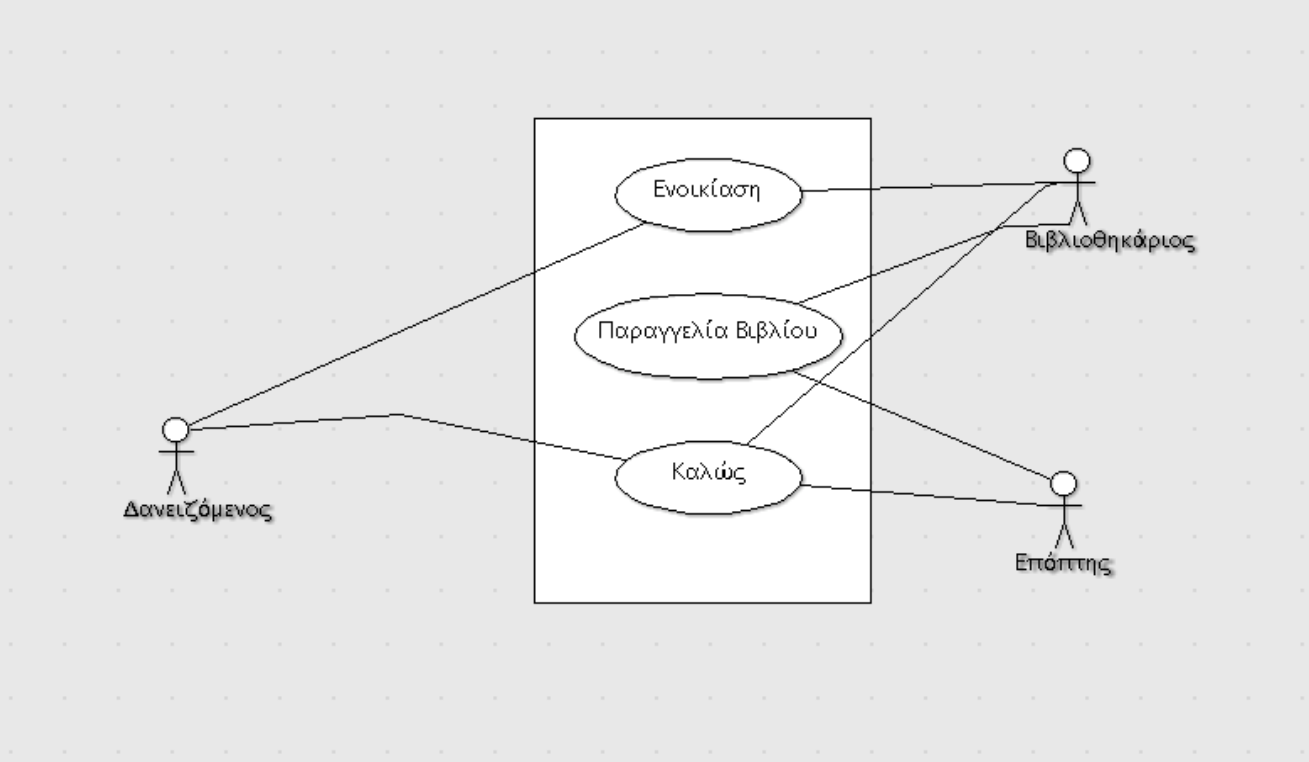
Διάλογοι

(Προσδιορισμός των διαλόγων των ρόλων με το σύστημα όσο διεξάγεται η περίπτωση χρήσης. Μπορούν να περιέχονται εικόνες-στιγμιότυπα ή διαγράμματα, προκειμένου να διευκολύνεται η διαδικασία απόδοσης της συνέχειας των διαδικασιών)

Σχήμα 4.4 Περίγραμμα προσδιορισμού περιπτώσεων χρήσης (flow of events diagram)

Η μοντελοποίηση των απαιτήσεων με τη βοήθεια περιπτώσεων χρήσης προϋποθέτει τα εξής βήματα :

1. Προσδιορισμός ρόλων και δικαιούχων,
2. Προσδιορισμός απαιτήσεων των ρόλων από το σύστημα
3. Προσδιορισμός του αναμενόμενου-επιθυμητού τρόπου αλληλεπίδρασης των ρόλων με το σύστημα
4. Προσδιορισμός των περιπτώσεων χρήσης που αποδίδουν αξία σε κάθε ρόλο και
5. Προσδιορισμός των περιπτώσεων χρήσης κατά τις οποίες, ενώ συμμετέχουν πολλοί από τους ρόλους, τελικά, δεν επωμίζονται όλοι αξία. Προσδιορισμός των ρόλων που επωμίζονται και των ρόλων που δεν επωμίζονται αξία σε κάθε περίπτωση χρήσης.



Σχήμα 4.5: Μία πιθανή περίπτωση χρήσης για ένα απλό σύστημα βιβλιοθήκης

Οι περιπτώσεις χρήσης αποτελούν σενάρια που βασίζονται σε τεχνικές εκμαίευσης και μοντελοποίησης. Ένα σύνολο από σχετικά σενάρια που αφορούν σε διεργασίες, τις οποίες ο χρήστης απαιτεί να εκτελούνται, συγκεντρώνονται σε μία περίπτωση χρήσης. Το διάγραμμα της περίπτωσης χρήσης «Βιβλιοθήκη» μπορεί να περιγραφεί ακολούθως. Ο δικαιούχος της περίπτωσης χρήσης είναι ο «δανειζόμενος». Ο «βιβλιοθηκάριος» είναι απαραίτητος, προκειμένου ο δανειζόμενος να αντλεί όφελος από την περίπτωση χρήσης «ενοικίαση». Ο «επόπτης» και ο «βιβλιοθηκάριος» είναι απαραίτητοι για την ολοκλήρωση της διεργασίας «καλώς».

4.1.1 Ονόματα (Names)

Τα μεγάλα συστήματα συμπεριλαμβάνουν μεγάλο αριθμό αντικειμένων. Για την καλύτερη οργάνωση μεγάλων αριθμών στοιχείων, χρησιμοποιούνται τα πακέτα. Τα πακέτα μπορούν να θεωρούνται υπό-συστήματα του συνολικού συστήματος. Σημειώνεται εδώ ότι, οι περιπτώσεις χρήσης δεν είναι απολύτως συνδεδεμένες με τη γλώσσα UML, επομένως μπορούν να χρησιμοποιούνται για την ανάλυση κάθε κατάστασης.

Τα ονόματα των περιπτώσεων χρήσης αντανακλούν τη δομή των υποσυστημάτων. Τα ονόματα μπορούν να είναι:

- Ονόματα που αποτελούνται από κείμενο

- Ονόματα διαδρομών ως προθέματα περιπτώσεων χρήσης από ένα πακέτο ή ένα υπό-σύστημα στο οποίο ανήκουν (34)

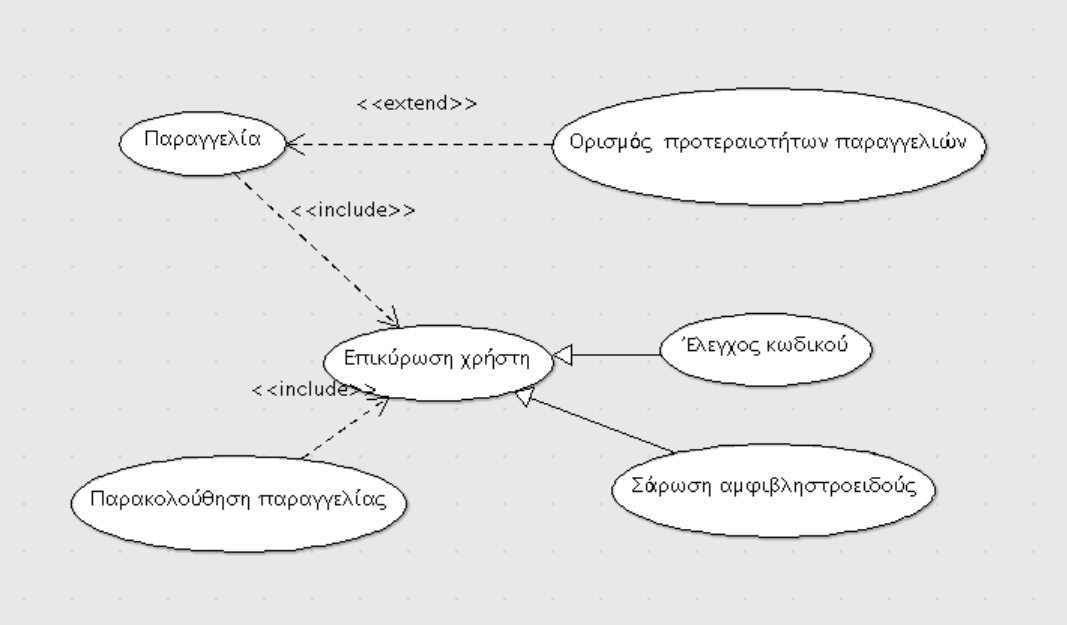
4.1.2 Γενίκευση μεταξύ ρόλων

Ένας ρόλος Α γενικεύει έναν ρόλο Β εφόσον, κάθε συμπεριφορά του Α μπορεί να εκτελείται και από τον Β. Για παράδειγμα, ο ρόλος «πελάτης» γενικεύει το ρόλο «πελάτης δανείου» διότι κάθε συμπεριφορά του δεύτερου μπορεί να εκτελείται και από τον πρώτο. Ο ρόλος «πελάτης δανείου» είναι σε θέση να εκτελέσει και άλλες συμπεριφορές που επεκτείνουν ή διαφέρουν σε συγκεκριμένα σημεία από τις συμπεριφορές του ρόλου «πελάτης».

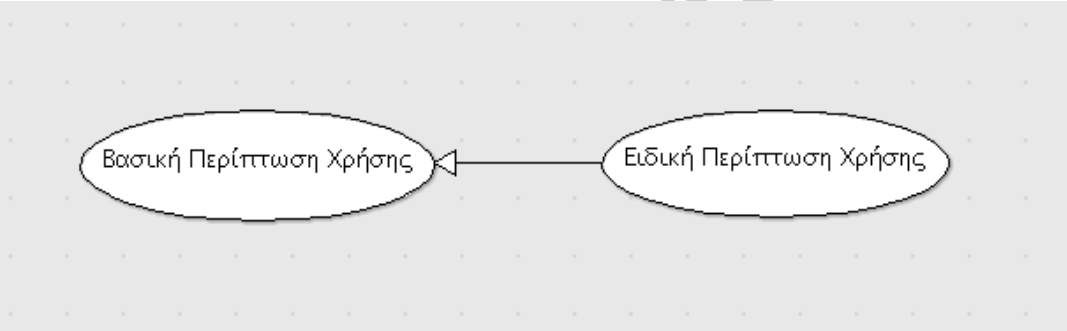
Οι περιπτώσεις χρήσης μπορούν να δομηθούν κατά τον ίδιο περίπου τρόπο, όπως ένα διάγραμμα κλάσεων. Για παράδειγμα, στις περιπτώσεις χρήσης χρησιμοποιούνται γενικεύσεις, συμπεριλήψεις και επεκτάσεις (34).

4.1.3 Γενίκευση (Generalization)

Η γενίκευση μεταξύ περιπτώσεων χρήσης είναι σχετική με τη γενίκευση μεταξύ κλάσεων στην Αντικειμενοστρεφή Ανάλυση. Η ειδική περίπτωση χρήσης (ή αλλιώς υποκλάση ή κλάση «παιδί») κληρονομεί όλες τις συμπεριφορές της γενικής περίπτωσης χρήσης (ή βασικής περίπτωσης χρήσης, ή περίπτωσης χρήσης «γονέας»). Η ειδική περίπτωση μπορεί να επεκτείνει τη συμπεριφορά της βασικής, επομένως να αλλάξει και την έννοιά της. Η σημειογραφία φαίνεται στο Σχήμα 4.8. Το βέλος επιδεικνύει, πάντα, τη βασική περίπτωση χρήσης. Η ειδική περίπτωση χρήσης μπορεί να υποκαθιστά τη βασική σε οποιοδήποτε σημείο. Για παράδειγμα, οι περιπτώσεις χρήσης «Έλεγχος κωδικού» και «Σάρωση Αμφιβληστροειδούς» είναι οι ειδικές περιπτώσεις χρήσης της βασικής περίπτωσης χρήσης «Επικύρωση Χρήστη» και έχουν ίδιες συμπεριφορές με αυτή. Τέλος, μπορούν να χρησιμοποιηθούν όπου και αυτή (34).



Σχήμα 4.7: Δόμηση περιπτώσεων χρήσης



Σχήμα 4.8: Δόμηση Περιπτώσεων χρήσης με γενίκευση

4.1.4 Συμπερίληψη-Συνυπολογισμός (Inclusion)

Η συμπερίληψη είναι μία έννοια κατά την οποία, η βασική περίπτωση χρήσης ενσωματώνει τη συμπεριφορά μίας άλλης περίπτωσης χρήσης, σε ένα συγκεκριμένο σημείο της πρώτης. Σχετικά με την απεικόνιση, το βέλος επιδεικνύει πάντα την περίπτωση χρήσης, της οποίας η συμπεριφορά ενσωματώνεται στη βασική περίπτωση χρήσης. Η συμπερίληψη χρησιμοποιείται προκειμένου να αποφεύγεται η περιγραφή της ίδιας ροής γεγονότων επανειλημμένα. Η συμπερίληψη επιτρέπει σε συμπεριφορές να μπορούν να ενσωματώνονται σε άλλες περιπτώσεις χρήσης, όπου αυτό κρίνεται απαραίτητο και μόνο(34). Συμπερασματικά, η συμπερίληψη είναι μια σχέση ανάμεσα σε μια βασική και μια ή περισσότερες ειδικές περιπτώσεις χρήσης που ορίζει ότι, η βασική περίπτωση χρήσης απαιτεί τη λειτουργία των ειδικών περιπτώσεων χρήσης.

Στο Σχήμα 4.7 οι περιπτώσεις χρήσης «Παραγγελία» και «Παρακολούθηση παραγγελίας» συμπεριλαμβάνουν τη περίπτωση χρήσης «Επικύρωση Χρήστη». Η περίπτωση χρήσης «Παρακολούθηση παραγγελίας» μπορεί να περιλαμβάνει την ακόλουθη ροή γεγονότων(34):

1. Απόκτηση και επαλήθευση αριθμού παραγγελίας
2. Συμπερίληψη (επικύρωση χρήστη)
3. Εξέταση της κατάστασης όλων των μερών της παραγγελίας και επόμενα, αναφορά στον χρήστη.

4.1.5 Επέκταση (Extension)

Οι σχέσεις επέκτασης χρησιμοποιούνται για τη μοντελοποίηση εκείνων των μερών των περιπτώσεων χρήσης, τα οποία ο χρήστης θεωρεί προαιρετικά. Η βασική περίπτωση χρήσης μπορεί να έχει υπόσταση μόνη της. Κάτω, όμως, από συγκεκριμένες συνθήκες, η συμπεριφορά της μπορεί να επεκτείνεται από άλλη περίπτωση χρήσης. Συμπερασματικά, η επέκταση ορίζει μια σχέση ανάμεσα σε περιπτώσεις χρήσης που προσδιορίζει εναλλακτικούς τρόπους διαχείρισης της βασικής περίπτωσης χρήσης.

Όσον αφορά στην περίπτωση χρήσης «Παραγγελία» του σχήματος 4.7, μπορούν να περιλαμβάνονται τα παρακάτω γεγονότα (34):

1. Συμπερίληψη (Επικύρωση Χρήστη)
2. Συλλογή των αντικειμένων του χρήστη για την παραγγελία
3. Επέκταση (Ορισμός προτεραιοτήτων παραγγελιών)
4. Υποβολή παραγγελίας

4.2 Παραδείγματα Εκμαίευσης Απαιτήσεων με τη χρήση Περιπτώσεων Χρήσης

4.2.1 Παράδειγμα «ATM»

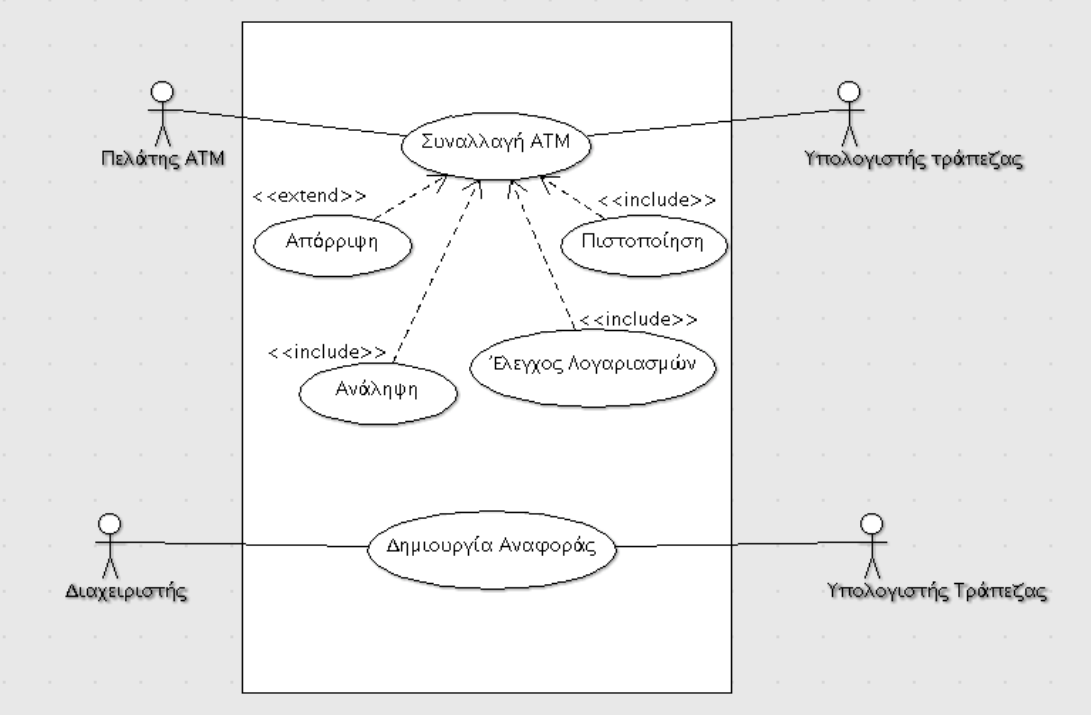
Δήλωση Προβλήματος:

Ένα σύνολο τραπεζών επιθυμεί να αναπτύξει υπολογιστικό δίκτυο συμπεριλαμβάνοντας ταμεία και ATMs (Automatic Teller Machines). Το συνολικό κόστος του συγκεκριμένου έργου θα βαρύνει τις επιμέρους τράπεζες του συνόλου (9).

- Κάθε τράπεζα διατηρεί αρχείο λογαριασμών πελατών σε υπολογιστή που βρίσκεται εντός των εγκαταστάσεών της και ανήκει σε αυτήν. Σε αυτό το αρχείο καταγράφονται και οι συναλλαγές των πελατών.

- Προκειμένου να εκκινήσει η συναλλαγή ενός πελάτη με την τράπεζα μέσω ΑΤΜ, ο πελάτης θα εισάγει τη κάρτα του σε αυτό. Ύστερα, ο χρήστης αλληλεπιδρά με το ΑΤΜ προκειμένου να καθοριστεί ο επιθυμητός τρόπος συναλλαγής με την τράπεζα. Το ΑΤΜ επικοινωνεί με το κεντρικό σύστημα των τραπεζών, ώστε να καθοριστούν οι υπολογιστές που πρέπει να αλληλεπιδράσουν (ανήκουν στο συνολικό τραπεζικό σύστημα) για να διεκπεραιωθεί η διαδικασία. Τέλος, η συναλλαγή ελέγχεται από τους υπολογιστές των τραπεζών και αν η συναλλαγή μπορεί να επικυρωθεί, τότε τα χρήματα διανέμονται μέσω του ΑΤΜ και εκτυπώνεται το αποδεικτικό της συναλλαγής.
- Όλες οι τράπεζες έχουν δικά τους ταμεία. Τα ταμεία μπορούν να επικοινωνούν απευθείας με τον υπολογιστή της αντίστοιχης τράπεζας, χωρίς τη διαμεσολάβηση του κεντρικού υπολογιστή των τραπεζών. Όταν ένας πελάτης επιθυμεί να πραγματοποιήσει συναλλαγή μέσω του ταμείου, ο ταμίας εισάγει τα στοιχεία του λογαριασμού και της συναλλαγής.
- Για την ομαλή λειτουργία του συστήματος και την αποτελεσματική διεκπεραίωση των συναλλαγών, είναι απαραίτητες οι διαδικασίες διατήρησης αρχείων συναλλαγών, ενώ θα πρέπει να πληρούνται διατάξεις ασφαλείας.
- Το σύστημα θα πρέπει να είναι σε θέση να διαχειρίζεται, χωρίς σφάλματα, περιπτώσεις ταυτόχρονης πρόσβασης στον ίδιο λογαριασμό.
- Οι υπολογιστές των τραπεζών διαθέτουν, ήδη, λογισμικό. Είναι απαραίτητη η ανάπτυξη λογισμικού για τα ATMs και για το δίκτυο.
- Το κόστος του συστήματος κοινής χρήσης επιμερίζεται στις τράπεζες, με κριτήριο τον αριθμό των πελατών στους οποίους έχει εκδοθεί κάρτα συναλλαγών.
- Οι διαχειριστές του συστήματος, θα είναι υπεύθυνοι για τη διαχείριση του δικτύου και για την έκδοση σχετικών αναφορών.

Μία πιθανή εκδοχή του διαγράμματος της περίπτωσης χρήσης του συστήματος των τραπεζών φαίνεται στο σχήμα 4.9. Το πλαίσιο μέσα στο οποίο περιλαμβάνονται οι περιπτώσεις χρήσης στο σχήμα, καθορίζει τα όρια του συστήματος ενός ΑΤΜ.



Σχήμα 4.9 Διάγραμμα περίπτωσης χρήσης συστήματος τραπεζών

Στο σχήμα 4.9 υπάρχουν δύο κύριες περιπτώσεις χρήσης: Η περίπτωση «Συναλλαγή ATM» και η περίπτωση «Δημιουργία αναφοράς». Τμήμα της περίπτωσης χρήσης «Συναλλαγή ATM» είναι η περίπτωση χρήσης «Πιστοποίηση», κατά την οποία ελέγχονται τα στοιχεία του πελάτη. Επιπλέον, όπως φαίνεται στο σχήμα, οι εφικτοί τρόποι συναλλαγής είναι δύο: Η περίπτωση «Ανάληψη» και η περίπτωση «Έλεγχος Λογαριασμών» (κατά την οποία θα ελεγχθεί η κατάσταση του συγκεκριμένου λογαριασμού). Η περίπτωση χρήσης «Απόρριψη» εκκινεί στις περιπτώσεις κατά τις οποίες, τα στοιχεία του χρήστη δεν ταυτοποιούνται με τα στοιχεία του αρχείου της τράπεζας, ή όταν η συναλλαγή δεν είναι εφικτή.

4.2.2 Παράδειγμα Συστήματος Σημείου Πώλησης

Δήλωση Προβλήματος: (9)

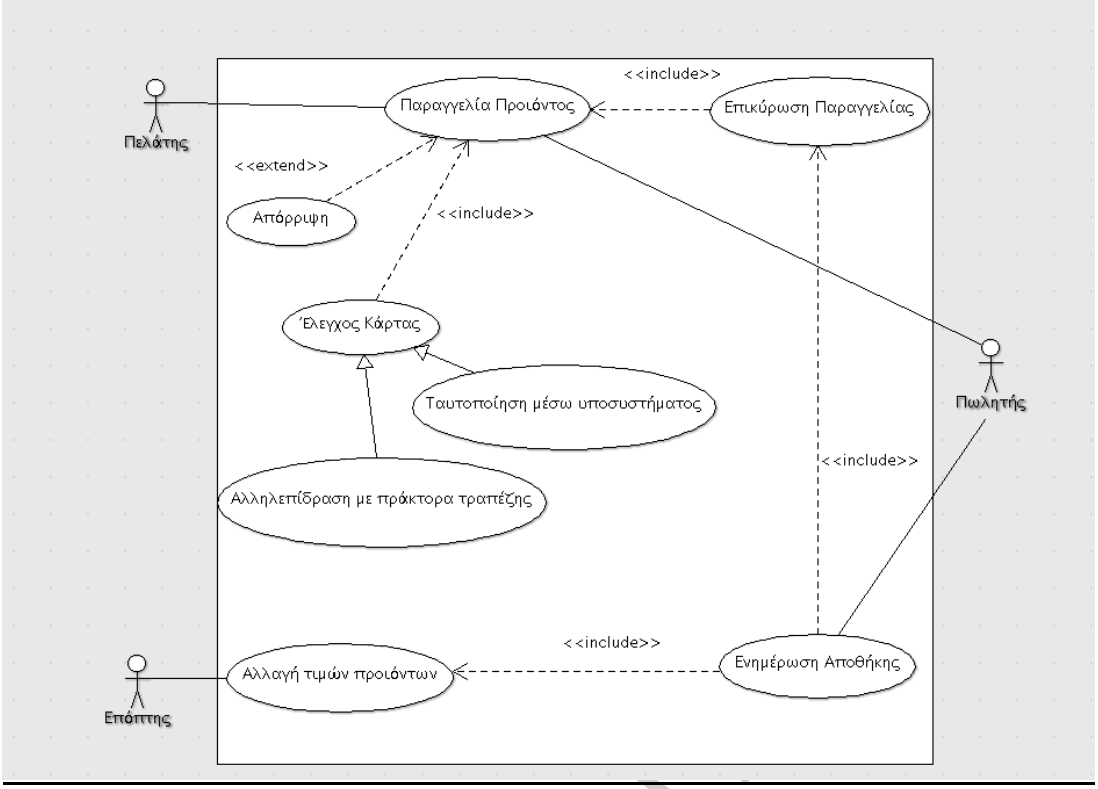
Θεωρούνται περιπτώσεις χρήσης για απομακρυσμένο σημείο πώλησης και σύστημα ελέγχου απογραφής με τα παρακάτω δεδομένα.

- Το προσωπικό των πωλήσεων μπορεί να κινείται γύρω από το σημείο πώλησης, διαθέτοντας κινητές τερματικές συσκευές POS, ώστε να αλληλεπιδρούν με το σύστημα από απόσταση. Οι πωλήσεις αντικειμένων και οι συναλλαγές με τους πελάτες, μπορούν να διεκπεραιώνονται με τη βοήθεια των συσκευών αυτών. Οι πελάτες που επιθυμούν να αγοράσουν κάποιο αντικείμενο από το συγκεκριμένο σημείο πώλησης, θα πρέπει να

διαθέτουν κάρτα πίστωσης ή χρεωστική κάρτα ή κάρτα που έχει εκδοθεί στο όνομα του κατόχου από το κατάστημα, για την εξυπηρέτηση του σκοπού του συστήματος.

- Αρχικά, ο πελάτης παρέχει την κάρτα του στον πωλητή. Αυτός ελέγχει την κάρτα με τη βοήθεια της συσκευής POS, ώστε να επικυρωθούν τα στοιχεία του πελάτη. Αν η κάρτα του πελάτη έχει εκδοθεί από τράπεζα, το σύστημα θα πρέπει να αλληλεπιδράσει με το σύστημα της τράπεζας (τον αντίστοιχο πράκτορα) για την επικύρωση της συναλλαγής. Αν η κάρτα έχει εκδοθεί από το κατάστημα και το συγκεκριμένο σημείο πώλησης, τότε αρκεί η αλληλεπίδραση της συσκευής POS με το αντίστοιχο υποσύστημα ελέγχου και επικύρωσης στοιχείων και συναλλαγών.
- Εφόσον η κάρτα και η συναλλαγή επικυρωθούν , τα χρήματα μεταφέρονται από την κάρτα στον λογαριασμό του καταστήματος πώλησης, ενώ το αντικείμενο που πουλήθηκε σαρώνεται και αφαιρείται από την αποθήκη του καταστήματος. Ο πελάτης παραλαμβάνει το προϊόν και απομακρύνεται.
- Οι πωλητές έχουν τη δυνατότητα να πραγματοποιούν καταμέτρηση των προϊόντων της αποθήκης, ενώ μπορούν να αριθμούν προϊόντα που υπάρχουν στην αποθήκη αλλά δεν αναφέρεται σε αυτά ο αριθμός του προϊόντος. Όταν ένα προϊόν σαρώνεται, η αποθήκη ελέγχεται και ανανεώνεται.
- Οι επόπτες μπορούν να χρησιμοποιούν τα δικά τους τερματικά για τον έλεγχο των πωλήσεων των πωλητών και μπορούν να αλλάζουν την τιμή των προϊόντων εφόσον, για παράδειγμα, είναι περίοδος εκπτώσεων.

Μία πιθανή εκδοχή του διαγράμματος της περίπτωσης χρήσης του συστήματος του απομακρυσμένου σημείου πώλησης φαίνεται στο σχήμα 4.10.



Σχήμα 4.10: Διάγραμμα περίπτωσης χρήσης απομακρυσμένου σημείου πώλησης

4.2.3 Παράδειγμα Συστήματος Κράτησης Θέσεων

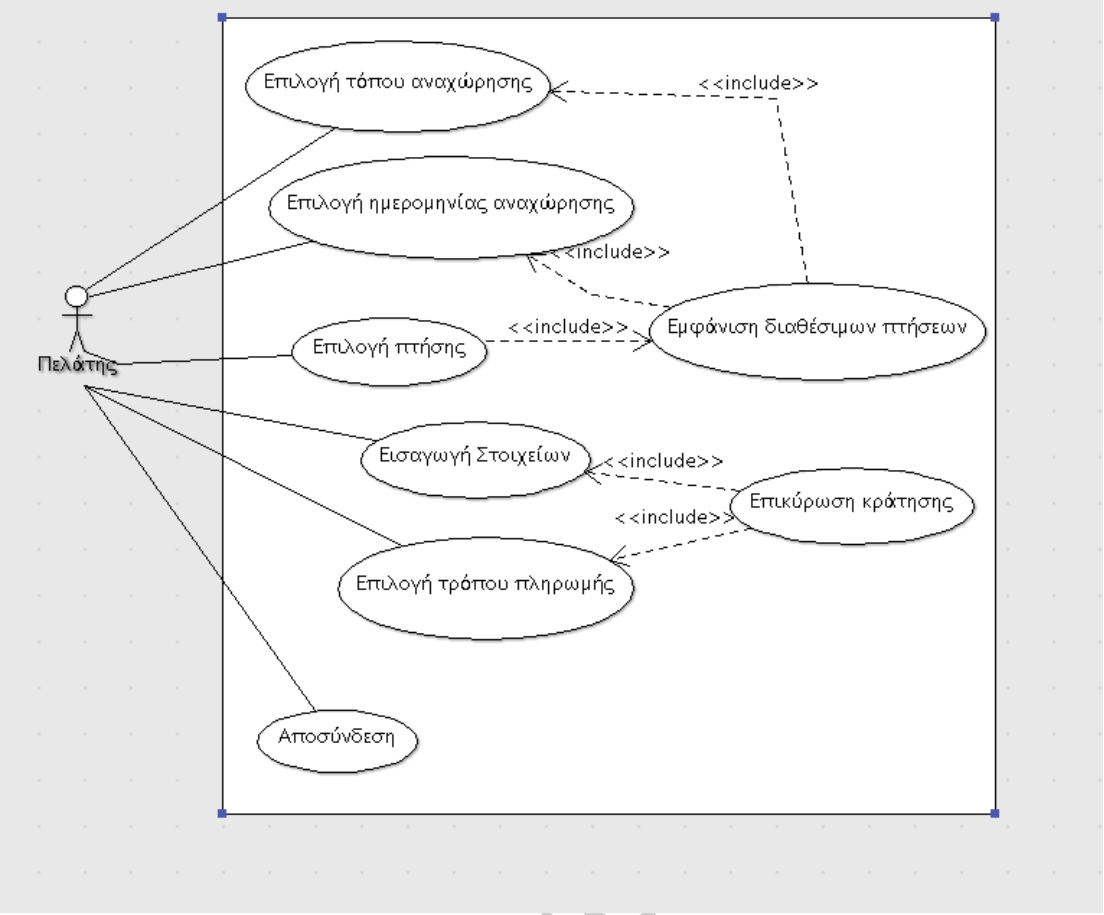
Δήλωση Προβλήματος:

Θέλουμε να υλοποιήσουμε ένα σύστημα κράτησης θέσεων. Με το σύστημα αλληλεπιδρούν απλοί ταξιδιώτες. Η κράτηση θέσης συνεπάγεται τη πληρωμή της πτήσης. Η πληρωμή μπορεί να γίνει με πιστωτική κάρτα, είτε με μετρητά.

Σενάριο:

- Ο πελάτης συνδέεται στο σύστημα
- Ο πελάτης επιλέγει τον τόπο αναχώρησης
- Ο πελάτης επιλέγει ημερομηνία αναχώρησης
- Το σύστημα εμφανίζει τις διαθέσιμες πτήσεις και τα στοιχεία τους
- Ο πελάτης επιλέγει τη πτήση που τον ενδιαφέρει
- Το σύστημα παρουσιάζει την πλήρη πληροφορία τιμολόγησης.
- Ο πελάτης εισάγει τα στοιχεία του και επιλέγει τον τρόπο πληρωμής
- Το σύστημα επιβεβαιώνει την κράτηση θέσης
- Ο πελάτης αποσυνδέεται από το σύστημα

Μία πιθανή εκδοχή του διαγράμματος της συγκεκριμένης περίπτωσης χρήσης δίνεται στο σχήμα 4.11.



Σχήμα 4.11: Περίπτωση χρήσης online συστήματος κράτησης θέσεων

4.3 Ολοκλήρωση Περιπτώσεων Χρήσης

Μία ενδιαφέρουσα προσέγγιση όσον αφορά στις περιπτώσεις χρήσης δίνεται από τους Kulak και Guiney (35). Οι συγγραφείς προσδιορίζουν ένα σύνολο από κρίσιμα ζητήματα και προτείνουν τέσσερα βήματα, τα οποία θα πρέπει να επαναλαμβάνονται, προκειμένου οι περιπτώσεις χρήσης να ολοκληρώνονται αποτελεσματικά. Αξίζει να σημειωθεί ότι κάθε ένα από τα βήματα αυτά έχει επαναληπτικό χαρακτήρα από μόνο του. Τα βήματα παρατίθενται ακολούθως (35):

(1) Πρόσοψη (Facade): Προσδιορισμός και περίγραμμα περιπτώσεων χρήσης. Παροχή υψηλού επιπέδου περιγραφής των περιπτώσεων χρήσης.

(2) Γέμιση (Filled): Διεύρυνση και εμβάθυνση των περιπτώσεων χρήσης.

(3) Συγκέντρωση (Focused): Εκλέπτυνση των περιπτώσεων χρήσης.

(4) Ολοκλήρωση (Finished): Τελικός έλεγχος, επιδιόρθωση, ολοκλήρωση.

Τα παραπάνω επαναληπτικά βήματα αποτελούν σύνολο δραστηριοτήτων που πρέπει να διενεργούνται ως μέρος μίας συνολικής διαδικασίας. Είναι σημαντικό

να τονιστεί ότι κάθε βήμα δεν αποτελεί διαδικασία από μόνο του. Ακολουθεί ειδικότερη ανάλυση για κάθε ένα από τα βήματα:

4.3.1 Πρόσοψη

Σκοπός της συγκεκριμένης δραστηριότητας είναι η αντιστοίχιση των χρηστών του προτεινόμενου συστήματος, στις θέσεις όπου θα κληθούν να λάβουν και οι οποίες αναφέρονται στις κύριες διεργασίες του συστήματος. Μία περίπτωση χρήσης γέμισης περιλαμβάνει τις απαραίτητες πληροφορίες. Σε αυτές περιλαμβάνονται: ονόματα, ρόλοι, μικρή περιγραφή αλληλεπιδράσεων.

Το επαναληπτικό βήμα της πρόσοψης αποτελείται από τα παρακάτω (35):

1) Δημιουργία της δήλωσης του προβλήματος

2) Προσδιορισμός και αναφορά της υπάρχουσας τεκμηρίωσης και του υπάρχοντος διανοητικού κεφαλαίου

3) Προσδιορισμός των χρηστών, της ομάδας διαχείρισης χρηστών, των εμπλεκόμενων μερών, των πελατών που εξυπηρετούνται από την ομάδα χρηστών και των ιδιοκτητών κάθε είδους δεδομένων και πληροφοριών

4) Συνέντευξη με τα εμπλεκόμενα μέρη,

5) Προσδιορισμός ρόλων,

6) Δημιουργία περιπτώσεων χρήσης πρόσοψης,

7) Εκκίνηση της δημιουργίας του καταλόγου των επιχειρηματικών κανόνων,

8) Ανάλυση κινδύνου,

9) Δημιουργία δήλωσης εργασίας,

10) Έντυπη έγκριση από τον εκτελεστικό χορηγό,

Η έξοδος μίας επανάληψης πρόσοψης είναι ένα σύνολο από, εσφαλμένως ορισμένες, περιπτώσεις χρήσης. Από την άλλη μεριά, αυτό το σύνολο μπορεί να χρησιμοποιηθεί, προκειμένου να οριστεί ο σκοπός του έργου και οι διεργασίες που το αποτελούν.

Παράδειγμα 4.2: Ένα παράδειγμα πρόσοψης δίνεται στο σχήμα 4.10.

4.3.2 Η περίπτωση χρήσης Γέμισης

Σε αυτό το επαναληπτικό βήμα εκτελούνται δύσκολες διεργασίες. Σκοπός του συγκεκριμένου σταδίου είναι η δημιουργία ενός περιεκτικού συνόλου περιπτώσεων χρήσης και επιχειρηματικών κανόνων για τη περιγραφή της εφαρμογής. Παρόλο που αυτές οι απαιτήσεις είναι, συχνά, τραχείς, είναι σημαντικές διότι εμπεριέχουν πολλές λεπτομέρειες όσον αφορά σε λειτουργικές και μη-λειτουργικές απαιτήσεις για το σύστημα.

Τα κρίσιμα βήματα της επανάληψης γέμισης είναι τα εξής:

1. Διάσπαση των προσόψεων και δημιουργία εκτενέστερων περιπτώσεων χρήσης,
2. Συλλογή και τεκμηρίωση μη-λειτουργικών απαιτήσεων,
3. Συμπερίληψη επιχειρηματικών κανόνων,
4. Έλεγχος των περιπτώσεων χρήσης

Παράδειγμα 4.3: Ένα παράδειγμα περίπτωσης χρήσης γέμισης δίνεται στο σχήμα 4.11. Στο παράδειγμα φαίνεται η περίπτωση κατά την οποία, τα βήματα για μία περίπτωση χρήσης διαδικασίας δανείου προστίθενται για να δημιουργήσουν μία «γεμάτη» περίπτωση χρήσης (35).

Όνομα Περίπτωσης Χρήσης:	**Παρακολούθηση Πωλήσεων Ένδυσης**
Επανάληψη:	**Πρόσοψη**
Περίληψη:	Περίπτωση χρήσης πλαισίου συστήματος. Το σύστημα απαιτεί την παρακολούθηση των πληροφοριών που σχετίζονται με πελάτες, συναντήσεις, σχέδια, παραγγελίες και πωλήσεις. Ο σκοπός του συστήματος είναι να παρέχει δεδομένα ιστορικού πωλήσεων, ώστε ο ιδιοκτήτης να μπορεί μελετήσει.
Κύρια Γεγονότα:	
Εναλλακτικές Διαδρομές:	
Διαδρομές Εξαιρέσεων:	
Σημεία Επέκτασης:	

Υποθέσεις (Assumptions):	
Προϋποθέσεις (Pre-Conditions):	
Μετ-Υποθέσεις (Post-Conditions):	
Σχετικοί Επιχειρηματικοί Κανόνες:	
Συγγραφέας (Author):	
Ημερομηνία:	

Σχήμα 4.10. Περίπτωση Χρήσης Πρόσοψης για την παρακολούθηση πωλήσεων ένδυσης

Όνομα Περίπτωσης Χρήσης:	**Διαδικασία Δανείου**
Επανάληψη:	**Γέμιση**
Περίληψη:	Ο πάροχος του δανείου και ο αγοραστής επεξεργάζονται τους όρους του δανείου. Οι όροι περιλαμβάνουν το επιτόκιο, την ασφάλεια, την εγγύηση και άλλα.
Βασικά Γεγονότα:	1. Η περίπτωση χρήσης ξεκινά όταν ο αγοραστής και ο πωλητής συμφωνήσουν στους όρους. 2. Ο αγοραστής υποδεικνύει ότι απαιτείται δάνειο. 3. Το σύστημα αποκρίνεται και έρχεται σε επαφή με τον πάροχο του δανείου, με τις λεπτομέρειες των απαιτήσεων

	του αγοραστή. 4. Ο πάροχος δανείου χρησιμοποιεί τις απαιτήσεις ως είσοδο για τη δημιουργία πρότασης δανείου. 5. Το σ΄συτημα αποκρίνεται στέλνοντας την πρόταση στον αγοραστή και τον πράκτορα του αγοραστή. 6. Ο αγοραστής κάνει αντιπροσφορά στον πάροχο δανείου. 7. Το σύστημα αποκρίνεται στέλνοντας την αντιπροσφορά στον πάροχο του δανείου. 8. Ο πάροχος αποδέχεται την αντιπροσφορά. 9. Το σύστημα αποκρίνεται στέλνοντας ειδοποίηση σχετικά με την αποδοχή του παρόχου, στον αγοραστή και στον πράκτορά του. 10. Ο αγοραστής υποδεικνύει αποδοχή του δανείου. 11. Το σύστημα αποκρίνεται καταγράφοντας την αποδοχή του δανείου από τον αγοραστή.

Σχήμα 4.11: Μία περίπτωση χρήσης γέμισης, σχετική με δάνειο

Σημειώνεται ότι, σε περιπτώσεις κατά τις οποίες, η ορολογία των περιπτώσεων χρήσης είναι ιδιαιτέρως εξειδικευμένη, θα πρέπει να περιλαμβάνεται γλωσσάρι εννοιών για την αποσαφήνισή τους (35).

4.3.3 Η Επανάληψη Συγκέντρωσης

Ο κύριος στόχος της επανάληψης συγκέντρωσης είναι η επιλογή των καταλληλότερων και πιο αποτελεσματικών περιπτώσεων χρήσης. Η δημιουργία ενός περιεκτικού συνόλου από απαιτήσεις που θα ορίζουν επαρκώς

και σαφώς το σύστημα και θα επιτρέπουν την ανάπτυξη μίας επιτυχημένης εφαρμογής .

Η επανάληψη συγκέντρωσης είναι μία δραστηριότητα που σχετίζεται με τη λήψη δύσκολων αποφάσεων σχετικά με το έργο, το σύνολο των λειτουργιών, το σκοπό, καθώς και με την εξάλειψη των στοιχείων που δεν προσδίδουν αξία στο έργο. Η ιδέα είναι ότι, πρέπει να κατανοηθεί επαρκώς το πρόβλημα, τόσο, ώστε να εξαλειφθούν οι αντιφάσεις των απαιτήσεων και να οριστεί σαφώς ο τρόπος με τον οποίο οι περιπτώσεις χρήσης υποστηρίζουν η μία την άλλη, ώστε να προσεγγίζουν τους στόχους τους.

Παράδειγμα 4.4: Στην περίπτωση χρήσης συγκέντρωσης των σχημάτων 4.12 και 4.13 αποτυπώνεται ένα μέρος των απαιτήσεων ενός συστήματος που ασχολείται με τη σύναψη συμφωνιών, σχετικά με τους όρους πώλησης ενός αυτοκινήτου. Σημειώνεται ότι, οι μη-λειτουργικές απαιτήσεις έχουν σχέση με την επόμενη επανάληψη (9).

Κατά την επανάληψη συγκέντρωσης ο στόχος είναι η επικέντρωση σε κάθε περίπτωση χρήσης και η επιβεβαίωση της σχετικότητάς της με το σύστημα. Επιπλέον, σκοπός είναι η παροχή πληροφοριών ώστε να επιτυγχάνεται ακρίβεια, πληρότητα, ορθότητα και σαφήνεια των όρων της περίπτωσης χρήσης που σχετίζονται με την κύρια ανάγκη.

Εξετάζονται (35):

- Οι διεπαφές με άλλα συστήματα
- Οι σχέσεις μεταξύ διεπαφών και περιπτώσεων χρήσης
- Η βελτίωση της διαδικασίας. Εξέταση κάθε διαδικασίας της περίπτωσης χρήσης από την προοπτική του χρήστη.
- Η ιεράρχηση των περιπτώσεων χρήσης. Η ιεράρχηση γίνεται σύμφωνα με τα κριτήρια της χρησιμότητας και του επείγοντος του χαρακτήρα των περιπτώσεων χρήσης.
- Οι είσοδοι και οι έξοδοι. Επιβεβαίωση ότι, όλες οι πληροφορίες της περίπτωσης χρήσης παρέχονται σε αυτήν μέσω κάποιας από τις διεπαφές της.

4.3.4 Η Επανάληψη Ολοκλήρωσης

Σκοπός αυτής της επανάληψης είναι η απόδοση μη-λειτουργικών απαιτήσεων, των απαιτήσεων των διεπαφών χρήστη και η συγκέντρωση των περιπτώσεων χρήσης, ώστε να μπορούν να κατανοούνται αποτελεσματικά από τους σχεδιαστές του συστήματος. Τα βασικά βήματα αυτής της επανάληψης είναι τα εξής (9):

1. Προσθήκη απαιτήσεων διεπαφής χρήστη στο μοντέλο περίπτωσης χρήσης,
2. Αφαίρεση και συνδυασμός των μη λειτουργικών απαιτήσεων. Ένωση μη λειτουργικών απαιτήσεων με τις περιπτώσεις χρήσης,
3. Λήψη αποφάσεων τελικού σκοπού,
4. Δημιουργία αναφοράς απαιτήσεων

Όνομα Περίπτωσης Χρήσης:	**Συμφωνία Όρων**
Επανάληψη:	**Συγκέντρωση**
Περίληψη:	Ο αγοραστής και ο πωλητής συμφωνούν στους όρους της πώλησης, συμπεριλαμβάνοντας τις αλλαγές στο υπάρχον ακίνητο, στα αντικείμενα, στα οικονομικά στοιχεία, και σε κάθε άλλο όρο της πώλησης. Οι πράκτορες συμβουλεύουν τους πελάτες.
Βασικά γεγονότα:	1. Η περίπτωση χρήσης ξεκινά όταν ο αγοραστής και ο πωλητής υποδεικνύουν ότι μπορεί να υπάρξει συμφωνία. 2. Το σύστημα αποκρίνεται ενημερώνοντας τον αγοραστή, τον πράκτορα του αγοραστη, τον πράκτορα του πωλητή, τον αναλυτή νόμου και τον οικονομικό αναλυτή ότι, η διαδικασία της συμφωνίας μπορει να ξεκινήσει.

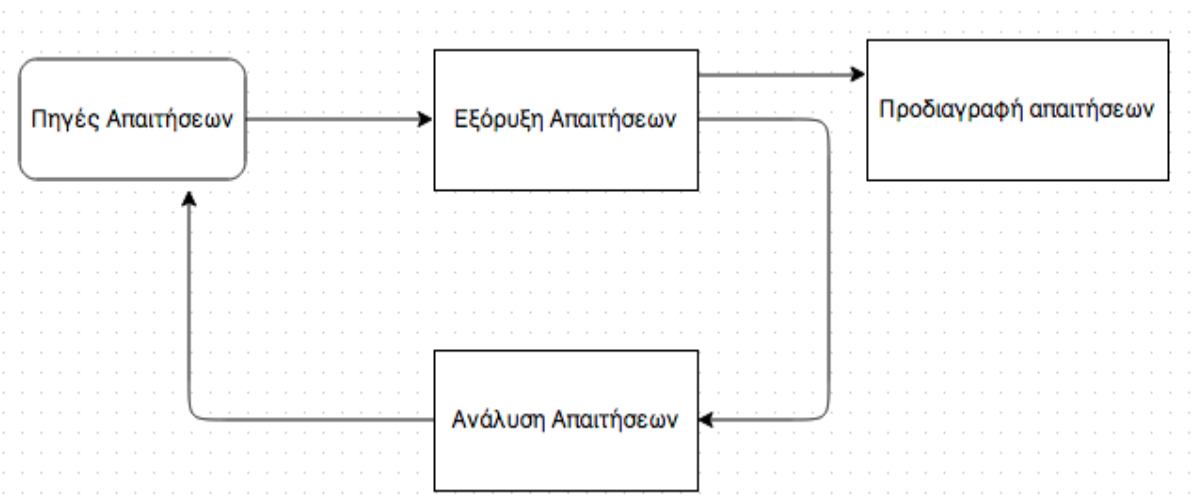
	3. Ο αγοραστής ή ο πωλητής αποδέχονται πρόταση όρων. 4. Το σύστημα αποκρίνεται επιτρέποντας σε όλους τους ρόλους (actors) την πρόσβαση στην πρόταση των όρων και τη δυνατότητα αλλαγών. 5. Οι ρόλοι κάνουν τις αλλαγές που θέλουν. 6. Το σύστημα αποκρίνεται δημοσιοποιώντας τις αλλαγές σε όλους. 7. Οι ρόλοι συζητούν τις αλλαγές και έρχονται σε συμφωνία όσον αφορά σε κάθε μία προτεινόμενη αλλαγή. 8. Το σύστημα αποκρίνεται εδραιώνοντας τη συμφωνία των αλλαγών, κάνοντας τη πρόταση των όρων δημόσια. 9. Οι ρόλοι υποδεικνύουν τη συμφωνία. 10. Η περίπτωση χρήσης ολοκληρώνεται όταν το σύστημα υποδεικνύει ότι η πρόταση των όρων έχει οριστικοποιηθεί.

Σχήμα 4.12: Πώληση ενός ακινήτου. Περίπτωση χρήσης συγκέντρωσης. Συμφωνία των όρων

5 Ανάλυση και Μοντελοποίηση με τη γλώσσα UML

5.1 Εισαγωγή

Ένα μοντέλο απαιτήσεων μπορεί να χρησιμοποιηθεί για την ανάλυση των απαιτήσεων και για την παροχή ανάδρασης και καθοδήγησης της διαδικασίας εξόρυξης (35).



Σχήμα 5.1: Χρήση Μοντέλων Ανάλυσης για ανάδραση στη διαδικασία Εξόρυξης Απαιτήσεων

Υπάρχουν πολλοί τρόποι αξιολόγησης και ανάλυσης του συνόλου των απαιτήσεων, όσο αυτές εξελίσσονται και αναπτύσσονται. Η δημιουργία μοντέλων, ή αλλιώς η μοντελοποίηση, είναι μία από τις πιο δημοφιλείς και αποδεκτές μεθόδους ανάλυσης και αξιολόγησης απαιτήσεων.

5.2 Μοντελοποίηση

Ένα μοντέλο απαιτήσεων μπορεί να χρησιμοποιηθεί στις εξής περιπτώσεις (36):

(1) έλεγχος εφικτότητας απαιτήσεων,

(2) διαδικασία κατανόησης των πτυχών του προβλήματος,

(3) αποφυγή παραλείψεων και

(4) ως μέσο εξάσκησης της φαντασίας.

Πότε ένα μοντέλο είναι αποτελεσματικό;

Ένα μοντέλο είναι αποτελεσματικό όταν μπορεί να τροποποιηθεί, επεκταθεί, συρρικνωθεί, ή μεταμορφωθεί προκειμένου να εξερευνηθούν οι απαιτήσεις. Κάθε μοντέλο έχει δυνάμεις, αδυναμίες και περιορισμούς. Επομένως, η επιλογή του μοντέλου έχει ιδιαίτερη σημασία. Πάντως, ένα καλό μοντέλο απαιτήσεων μπορεί να καθοδηγήσει την εξόρυξη απαιτήσεων και να διευκολύνει τη διαδικασία κατανόησης του κυρίαρχου προβλήματος.

Παρακάτω παρατίθενται μερικές δημοφιλείς προσεγγίσεις μοντελοποίησης απαιτήσεων.

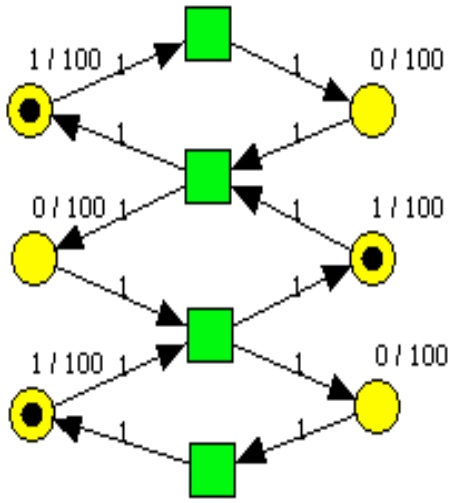
Επίσημες Μέθοδοι (Formal Methods)

Πρόκειται για μαθηματικά μοντέλα των απαιτήσεων του συστήματος. Δημοφιλείς μέθοδοι τέτοιων μοντελοποιήσεων είναι οι εξής(37):

- Z και B: Και οι δύο αποτελούν εκφράσεις της διαδικασίας προδιαγραφής απαιτήσεων με τη χρήση της λογικής και της θεωρίας. Τα μοντέλα συνήθως δημιουργούνται δίνοντας ένα σαφές μοντέλο της κατάστασης του συστήματος, αναδεικνύοντας τις ενέργειες που το σύστημα πρέπει να εκτελεί και καθορίζοντας τον τρόπο με τον οποίο κάθε ενέργεια επηρεάζει την κατάσταση του συστήματος.

  Η επιρροή στην κατάσταση του συστήματος μοντελοποιείται σε όρους των προϋποθέσεων των ενεργειών και των μετά-υποθέσεων (post-conditions).
- VDM (Vienna Development Method): Πρόκειται για μία άλλη μέθοδο μοντελοποίησης με τη χρήση της λογικής και της θεωρίας. Είναι κατάλληλα σχεδιασμένη για τη μοντελοποίηση των καταστάσεων και των μεταβάσεων (ή ενεργειών) που χρησιμοποιούν προϋποθέσεις και μετα - υποθέσεις.
- Petri Nets (Δίκτυα Petri): Η συγκεκριμένη μέθοδος χρησιμοποιεί τη μαθηματική θεωρία των γράφων. Είναι πολύ διαδεδομένη σε πολλές από τις ειδικότητες των μηχανικών και έχει χρησιμοποιηθεί ευρέως. Οι θέσεις (κίτρινοι κύκλοι) αναπαριστούν ισχυρισμούς, καταστάσεις, ή μεταβολές καταστάσεων σχετικά με το σύστημα. Όταν μία θέση καταλαμβάνεται από μία κουκίδα (κίτρινος κύκλος με μαύρη κουκίδα),

τότε ο ισχυρισμός που ανταποκρίνεται στη συγκεκριμένη θέση έχει ισχύ. Η συγκεκριμένη μέθοδος χρησιμοποιείται, κυρίως, για την κατανόηση προβλημάτων κατανομής πόρων σε πολύ-διαδικαστικά περιβάλλοντα.



Σχήμα 5.2: Παράδειγμα Μεθόδου «Petri Nets»

- Αλγεβρική Διαδικασία (Process Algebra): Αυτό το μοντέλο στηρίζεται στη μαθηματική θεωρία «automata». Οι πιο δημοφιλείς τύποι αυτού του μοντέλου είναι το CCS και το CSP. Τα συστήματα μοντελοποιούνται ως σύνολα από πράκτορες, οι οποίοι επικοινωνούν μεταξύ τους για να επιτύχουν έναν κοινό σκοπό. Κάθε πράκτορας εμπλέκεται σε μια σειρά ενεργειών που μπορούν να συγχρονιστούν με τις ενέργειες άλλων πρακτόρων.
- UML (Unified Modelling Language): Αποτελείται από διαφορετικές υποκατηγορίες γλωσσών αλλά δεν τις καθορίζει όλες τόσο αυστηρά.
- Προτυποποίηση (Prototyping): Πρόκειται για τη διαδικασία δημιουργίας εκτελέσιμων μοντέλων για την ανάλυση και καταγραφή απαιτήσεων. Τα πρωτότυπα αποτελούν εργαλείο κατανόησης του συστήματος και του περιβάλλοντός του. Το μοντέλο της προτυποποίησης είναι, εκ φύσεως, επαναληπτικό και αναδεικνύει ασαφείς και μη ακριβείς απαιτήσεις. Τέλος, το πολύ σημαντικό πλεονέκτημα αυτής της μεθόδου είναι ότι, οι πελάτες και οι τελικοί χρήστες του συστήματος μπορούν να εκπαιδευτούν επί της συμπεριφοράς του συστήματος σε συγκεκριμένες-δοσμένες συνθήκες.

5.3 UML (Unified Modelling Language):

Η UML είναι μία γλώσσα μοντελοποίησης που μπορεί να χρησιμοποιηθεί στις περισσότερες φάσεις του κύκλου ζωής του λογισμικού, συμπεριλαμβάνοντας τις φάσεις της Ανάλυσης Απαιτήσεων (με περιπτώσεις χρήσης και μοντέλα ανάλυσης), του Σχεδιασμού, της Υλοποίησης και του Ελέγχου του έργου (38). Οι πέντε κυριότερες υποκατηγορίες της γλώσσας UML που ενδιαφέρουν περισσότερο την ανάλυση του κυρίαρχου προβλήματος είναι οι εξής(39):

1. Περιπτώσεις Χρήσης (Use Cases): Χρησιμοποιούνται κατά την εξόρυξη απαιτήσεων. Η ανάλυση των περιπτώσεων χρήσης με την προοπτική Αντικειμενοστραφούς Σχεδιασμού (Object Oriented Analysis and Design Methods) πραγματοποιείται με την βοήθεια των Διαγραμμάτων Κλάσεων (Class Diagrams), των Διαγραμμάτων Αλληλεπίδρασης (Interaction Diagrams) και των Διαγραμμάτων Κατάστασης (State Charts).
2. Διαγράμματα Κλάσεων (Class Diagrams): Η αντικειμενοστραφής ανάλυση πραγματοποιείται σε όρους Αντικειμένων (Objects), Κλάσεων Αντικειμένων (Object Classes), Λειτουργίες Αντικειμένων (Object Functions) και Ενώσεις (Associations) μεταξύ Αντικειμένων. Αυτού του είδους η ανάλυση δομεί το κυρίαρχο πρόβλημα με τέτοιο τρόπο ώστε να είναι δυνατός ο έλεγχος εφικτότητας και ακρίβειας των ήδη υπαρχόντων απαιτήσεων, καθώς και η αξιολόγηση των νέων.
3. Διαγράμματα Αλληλεπίδρασης (Interaction Diagrams): Απεικονίζουν τον τρόπο επικοινωνίας μεταξύ των αντικειμένων και των ρόλων. Συγκεκριμένα, τα διαγράμματα ακολουθίας απεικονίζουν τη χρονική σειρά των μηνυμάτων που ανταλλάζουν τα αντικείμενα και οι ρόλοι. Τα διαγράμματα συνεργασίας, από την άλλη μεριά, προσδιορίζουν με ακρίβεια, τον τρόπο με τον οποίο τα αντικείμενα, οι ρόλοι και οι ενώσεις τους συνεργάζονται στο διάγραμμα κλάσεων, για την ολοκλήρωση των διεργασιών που προσδιορίστηκαν στις περιπτώσεις χρήσης. Με αυτόν το τρόπο εξακριβώνεται και η εφικτότητα και η πληρότητα των περιπτώσεων χρήσης.
4. Διαγράμματα Κατάστασης (State Charts): Αποτυπώνουν τη συμπεριφορά των αντικειμένων σε όρους καταστάσεων, μεταβολών καταστάσεων και γεγονότων. Τα διαγράμματα κατάστασης μπορούν να χρησιμοποιούνται για την αποτύπωση της συμπεριφοράς συστημάτων που βασίζονται σε γεγονότα και πτυχές συγχρονισμού.

5. Διαγράμματα Δραστηριότητας (Activity Diagrams): Χρησιμοποιούνται για την απεικόνιση δραστηριοτήτων, καταστάσεων αντικειμένων, μεταβολών καταστάσεων και γεγονότων, ενώ προσδιορίζουν τις δραστηριότητες που μπορούν να πραγματοποιούνται συγχρονισμένα. Τα διαγράμματα δραστηριότητας αποτελούν άλλον ένα τρόπο απεικόνισης διαδικασιών ή ακολουθιών διαδικαστικών βημάτων.

5.3.1 Διαγράμματα Κλάσεων (Class Diagrams)

Τα διαγράμματα χρήσης αποδίδουν πληροφορίες που συνιστούν τη βάση για τον προσδιορισμό των αντικειμένων, των ευθυνών τους και των ενώσεων εντός του συστήματος. Μετά τον προσδιορισμό των αντικειμένων και των ενώσεων ακολουθεί ο προσδιορισμός των γνωρισμάτων των αντικειμένων και των λειτουργιών που αυτά μπορούν να εκτελέσουν.

Τα αντικείμενα και οι σχέσεις μεταξύ τους αποτυπώνονται στα διαγράμματα κλάσεων. Αφού προσδιοριστούν τα αντικείμενα και οι σχέσεις τους είναι δυνατή η συγκέντρωση των αντικειμένων με τα ίδια γνωρίσματα σε κλάσεις αντικειμένων (classes). Οι κλάσεις και οι σχέσεις μεταξύ των κλάσεων συνιστούν τη βάση του Μοντέλου Ανάλυσης. Με τη βοήθεια του μοντέλου ανάλυσης της UML θα γίνει μία προσπάθεια προσδιορισμού της πληρότητας, της ορθότητας και της πυκνότητας των απαιτήσεων, ώστε να επιλεγούν αυτές που θα βελτιστοποιούν τις περιπτώσεις χρήσης.

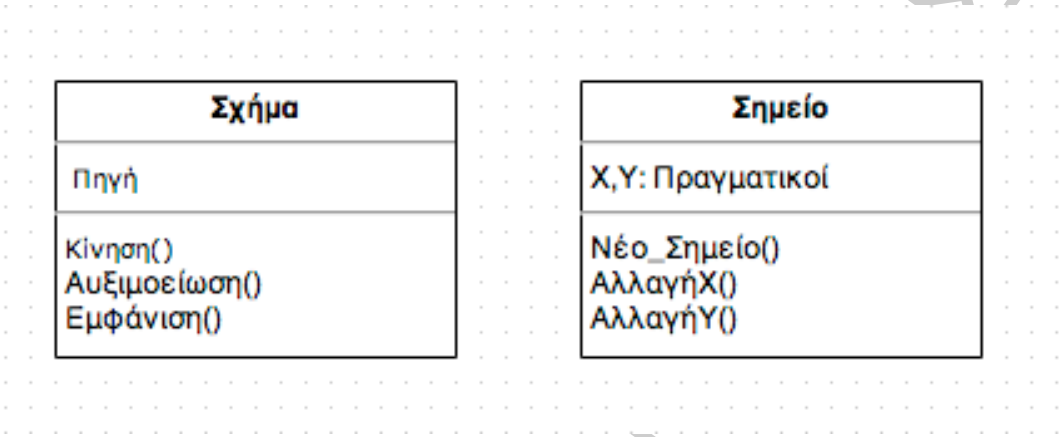
Αξίζει να σημειωθεί ότι, τα διαγράμματα κλάσεων απεικονίζουν δομημένη ή στατική πληροφορία σχετικά με το σύστημα.

Ειδικότερα, παρέχουν μία προοπτική λειτουργικών απαιτήσεων και δεδομένων του κυρίαρχου προβλήματος. Η κατανόηση της συμπεριφοράς (behavior) του συστήματος μπορεί να επιτευχθεί αποτελεσματικά με τη χρήση διαγραμμάτων συνεργασίας, διαγραμμάτων καταστάσεων και διαγραμμάτων δραστηριοτήτων.

Παρακάτω παρατίθενται μερικοί βασικοί ορισμοί (40)*:*

Αντικείμενα (Objects): Οντότητες με ακριβώς προσδιορισμένα όρια και δική τους ταυτότητα. Τα Αντικείμενα ενσωματώνουν κατάσταση και συμπεριφορά, ενώ διαθέτουν χαρακτηριστικά και λειτουργίες: Τα χαρακτηριστικά αφορούν στη κατάσταση του αντικειμένου και οι λειτουργίες στις αλλαγές της κατάστασης του μοντέλου.

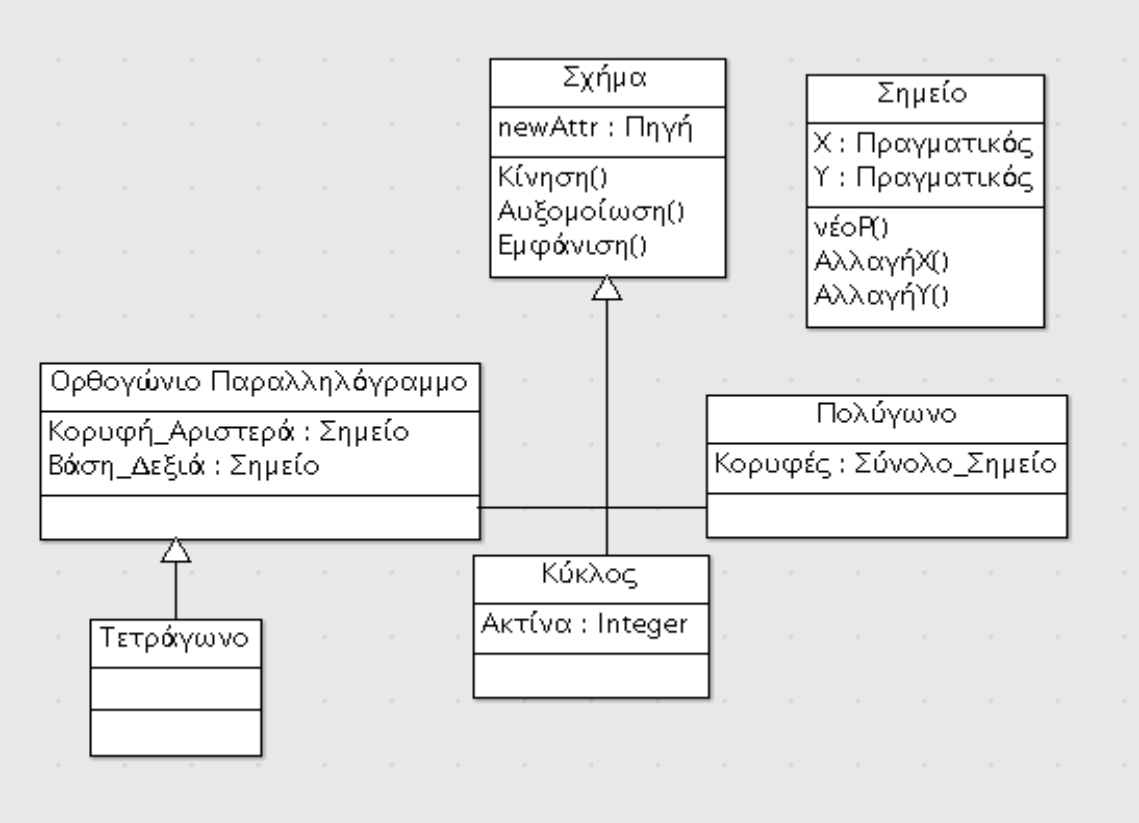
Κλάσεις (Classes): Συλλογές από αντικείμενα με κοινά χαρακτηριστικά και κοινές λειτουργίες. Κάθε αντικείμενο αντιπροσωπεύει μία κλάση αντικειμένων και κάθε κλάση αποτελείται από αντικείμενα με κοινές λειτουργίες και κοινά χαρακτηριστικά. Οι κλάσεις σχεδιάζονται σε κουτιά τεσσάρων κελιών : (1) Όνομα κλάσης, (2) Σύνολο χαρακτηριστικών, (3) Σύνολο λειτουργιών, (4) Σύνολο ευθυνών.



(Σχήμα 5.3): Δύο Κλάσεις

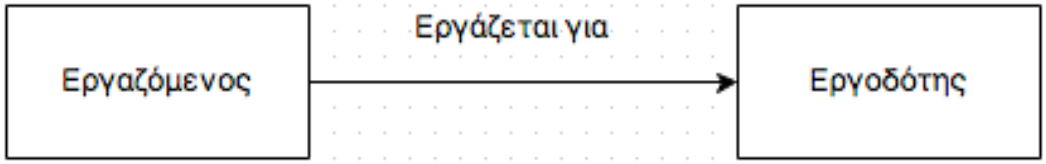
Σχέσεις: Διακρίνονται σε σχέσεις αντικειμένων και σχέσεις κλάσεων αντικειμένων. Τρία βασικά είδη τέτοιων σχέσεων από το πρίσμα της UML είναι τα εξής (40):

1. *Γενίκευση (Generalization):* Η σχέση γενίκευσης ή αλλιώς κληρονομικότητας (inheritance) ευδοκιμούν μεταξύ ενός γενικού και ενός ειδικού στοιχείου. Η ειδική κλάση κληρονομεί όλες τις ιδιότητες της γενικής κλάσης - κλάσης γονέα, όμως μπορεί να εξουδετερώνει λειτουργίες της ή και να τις επεκτείνει. Πιο συγκεκριμένα, με τη χρήση της γενίκευσης, τα αντικείμενα της ειδικής κλάσης μπορούν να χρησιμοποιούνται, όπου και τα αντικείμενα της κλάσης γονέα. Το αντίστροφο δεν αποτελεί δεδομένο. Παράδειγμα χρήσης σχέσης γενίκευσης απεικονίζεται στο σχήμα 5.4.



Σχήμα 5.3: Γενίκευση

2. *Ενώσεις (Associations):* Πρόκειται για ένα ακόμη είδος δομημένης σχέσης. Λαμβάνοντας υπόψη ενώσεις μεταξύ κλάσεων, δίνεται η δυνατότητα καθοδήγησης από ένα αντικείμενο μίας κλάσης σε μία ένωση με ένα αντικείμενο άλλης κλάσης. Οι ενώσεις μπορούν να είναι μοναδιαίες, δυαδικές ή Ν-αδικές και μπορούν να έχουν όνομα. Συνήθως, το όνομα της ένωσης περιγράφει τη φύση της ένωσης. Τέλος, οι ενώσεις μπορούν να έχουν ενσωματωμένο και ρόλο όπως στο Σχήμα 5.4.



Σχήμα 5.4: Σχέση ανάμεσα τις κλάσεις Εργαζόμενος και Εργοδότης

Οι ενώσεις, πολλές φορές, αντανακλούν πολλαπλότητα. Ένα αντικείμενο μπορεί να σχετίζεται με Ν αντικείμενα στην άλλη μεριά της σχέσης ή το αντίστροφο. Μερικές πιθανές πολλαπλότητες είναι οι εξής:

1...* (μία ή περισσότερες)

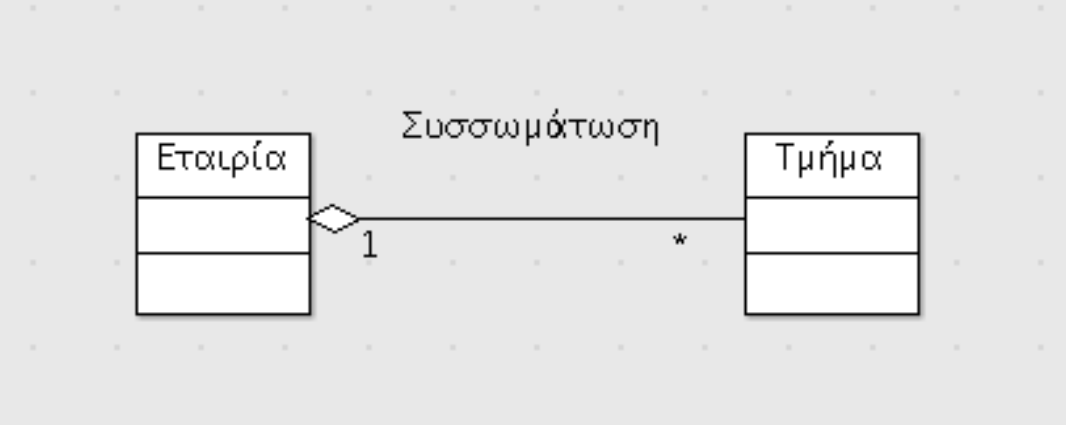
0...* (καμία ή περισσότερες)

1 (ακριβώς μία)

(0....1) (καμία ή μία)

(0...1,3..4,6..*) (πολύπλοκα εύρη)

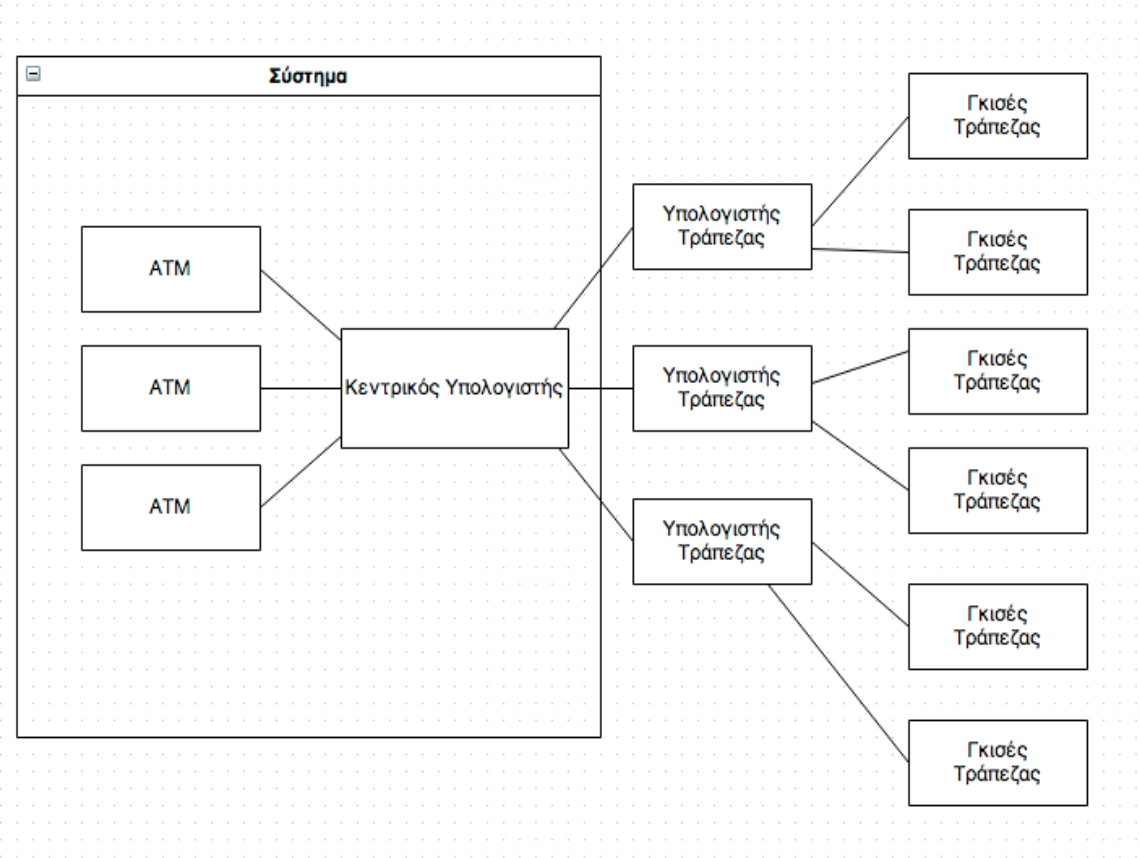
3. *Συσσωμάτωση (Aggregation):* Πρόκειται για ένα είδος ένωσης που αφορά τα μέρη στο σύνολό τους. Οι άλλες σχέσεις συνάπτονται, συνήθως, μεταξύ ομότιμων μερών. Η σχέση συσσωμάτωσης προορίζεται για αυστηρό έλεγχο της ιεραρχίας μεταξύ του συνόλου και των μερών. Στο σχήμα 5.5 απεικονίζεται μία τέτοια σχέση.



Σχήμα 5.5: Σχέση Συσσωμάτωσης. Η εταιρία είναι το «σύνολο» και το τμήμα είναι το «μέρος» του συνόλου.

5.4 Το παράδειγμα με τα ATM

Έστω ότι το σύστημα που μας ενδιαφέρει περιγράφεται από το σχήμα 5.6. Σε αυτό το σχήμα απεικονίζεται η παραμετροποίηση του υπολογιστικού συστήματος μίας τράπεζας (41).



Σχήμα 5.6: Το Δίκτυο Τραπεζών με τα ATM

Δήλωση του προβλήματος: (41)

«Σχεδιάστε λογισμικό υποστήριξης υπολογιστικού δικτύου τραπεζών συμπεριλαμβάνοντας τα ATMs (Automatic Teller Machines), ώστε να μπορούν να διαμοιράζονται πληροφορίες από μία ομάδα τραπεζών. Κάθε τράπεζα έχει τον δικό της ηλεκτρονικό υπολογιστή για τη διατήρηση αρχείου λογαριασμών και συναλλαγών (με πελάτες και με συνεργάτες) . Κάθε ταμείο επικοινωνεί απευθείας, αλλά μόνο με τον υπολογιστή της αντίστοιχης τράπεζας στην οποία βρίσκεται. Οι εργαζόμενοι στα ταμεία εισάγουν τον αριθμό λογαριασμού και τα δεδομένα συναλλαγής. Τα ATMs επικοινωνούν με έναν κεντρικό υπολογιστή που εκκαθαρίζει τις συναλλαγές με τις κατάλληλες τράπεζες. Όταν ένα ATM δέχεται μία κάρτα, αλληλεπιδρά με τον χρήστη, επικοινωνεί με το κεντρικό υπολογιστικό σύστημα ώστε να μεταφέρει τη συναλλαγή, διανέμει τα χρήματα και εκτυπώνει τα στοιχεία της συναλλαγής. Το σύστημα προϋποθέτει τήρηση αρχείων και διατάξεις ασφαλείας. Το σύστημα πρέπει, επίσης, να διαχειρίζεται την ταυτόχρονη πρόσβαση στον ίδιο λογαριασμό χωρίς σφάλματα. Οι τράπεζες διαθέτουν δικό τους λογισμικό για τους υπολογιστές τους. Καλείστε να σχεδιάσετε λογισμικό για τα ATMs και το δίκτυο. Το κόστος του συστήματος

κοινής χρήσης κατανέμεται στις επιμέρους τράπεζες του δικτύου τραπεζών με βάση τον αριθμό των πελατών στο όνομα των οποίων έχε εκδοθεί κάρτα με δυνατότητα αλληλεπίδρασης με τα ATMs.»

Περιπτώσεις Χρήσης: Έστω η περίπτωση χρήσης του σχήματος 5.7. Τα στάδια της επεξεργασίας περιλαμβάνουν όλα τα αντικείμενα και τις ενώσεις μεταξύ των αντικειμένων. Η αντικειμενοστραφής ανάλυση προσπαθεί να επεξεργαστεί τα αντικείμενα και τις ενώσεις αυτών ώστε να δημιουργήσει ένα μοντέλο κλάσεων. Οι περιπτώσεις χρήσης και η δήλωση του προβλήματος παρέχουν αξιόλογες πληροφορίες σχετικά με τα αντικείμενα και τις ενώσεις – συσχετίσεις του κυρίαρχου προβλήματος.

Επιλογή Αντικειμένων: Παρακάτω παρουσιάζεται μία απλή μέθοδος ανάπτυξης ενός μοντέλου αντικειμένων από μία περίπτωση χρήσης ή από την περιγραφή του προβλήματος.

Βήμα 1. Επιλογή δυνητικών αντικειμένων από την περίπτωση χρήσης ή την περιγραφή του προβλήματος. Τα δυνητικά αντικείμενα είναι συνήθως ¨πράγματα¨ και περιγράφονται ως ουσιαστικά. Αυτή η λίστα είναι το σύνολο των δυνητικών αντικειμένων και των δυνητικών κλάσεων αντικειμένων.

Βήμα 2. Απόρριψη όλων των παρακάτω, εφόσον έχουν ασαφή χαρακτηριστικά:

- Αν το αντικείμενο δεν είναι ορισμένο επαρκώς θα πρέπει να απορρίπτεται.
- Περιττά. Σε περιπτώσεις που κάποια από τα αντικείμενα της περιγραφής του προβλήματος αναφέρονται στην ίδια οντότητα.
- Γεγονός ή λειτουργία. Πρέπει να επιβεβαιώνεται ότι το δυνητικό αντικείμενο δεν αναφέρεται σε γεγονός ή λειτουργία.
- Εξω-συστημικά: Πρόκειται για κλάσεις αντικειμένων που δεν είναι σχετικά με το σύστημα που εξετάζεται. Στις περισσότερες περιπτώσεις η επιλογή αυτών των κλάσεων είναι αρκετά υποκειμενικό ζήτημα.
- Χαρακτηριστικά: Αν το δυνητικό αντικείμενο αναφέρεται σε κάτι απλό, χωρίς ιδιαίτερη συμπεριφορά και που μάλιστα είναι μέρος μίας άλλης κλάσης, τότε μάλλον πρόκειται για χαρακτηριστικό και όχι για αντικείμενο.
- Μετά-γλώσσα: Αν χρησιμοποιούνται ουσιαστικά για την περιγραφή των αντικειμένων θα πρέπει να επιβεβαιώνεται ότι το ουσιαστικό είναι

μέρος της γλώσσας προσδιορισμού πραγμάτων και όχι μέρος της γλώσσας του κυρίαρχου προβλήματος.

- Γενικότερα, αναζητούμε δυνητικά αντικείμενα για να δημιουργήσουμε αποτελεσματικά τις κλάσεις αντικειμένων ως εξής:

  Έρευνα στη δήλωση του Προβλήματος για:

  - Πραγματικό αντικείμενο, όπως ΑΤΜ και Χρεωστική Κάρτα
  - Ρόλος, όπως πελάτης ή εργαζόμενος
  - Γεγονός
  - Διενέργεια

Οι κατηγορίες που αναφέρθηκαν παραπάνω μπορεί να συσχετίζονται έμμεσα, όμως, έχει σημασία να τονιστεί ότι, τα αντικείμενα του μοντέλου ορίζουν το σύστημα και μία αποτελεσματική επιλογή είναι πιθανό να γλιτώσει την ομάδα του έργου από πολλή δουλειά σε επόμενη φάση.

Όνομα Περίπτωσης Χρήσης:	**Συναλλαγή ΑΤΜ**
Επανάληψη:	**Γέμιση**
Περίληψη:	Ο πελάτης εισάγει την κάρτα του σε ένα από τα ΑΤΜ του δικτύου τραπεζών. Ο πελάτης εκκινεί τη διαδικασία συναλλαγής και το δίκτυο διενεργεί τη συναλλαγή και παραδίδει στον πελάτη το αποτέλεσμα της συναλλαγής
Βασικά γεγονότα:	1. Ο πελάτης εισάγει την κάρτα του σε ένα από τα ΑΤΜ του δικτύου τραπεζών. 2. Ο πελάτης εισάγει τον προσωπικό κωδικό (pin-number). 3. Ο πελάτης επιλέγει τον τύπο της συναλλαγής. 4. Ο πελάτης εισάγει τις λεπτομέρειες της συναλλαγής στο ΑΤΜ. 5. Το ΑΤΜ αποστέλλει τα στοιχεία λογαριασμού του πελάτη, τον προσωπικό

	κωδικό και τις λεπτομέρειες της συναλλαγής στον Κεντρικό Υπολογιστή του Δικτύου Τραπεζών. 6. Ο Κεντρικός Υπολογιστής διατηρεί αρχείο της συναλλαγής και αποστέλλει τη συναλλαγή στον σωστό ηλεκτρονικό υπολογιστή της τράπεζας που διαχειρίζεται τις συναλλαγές του συγκεκριμένου πελάτη. 7. Ο ηλεκτρονικός υπολογιστής της τράπεζας ταυτοποιεί τον προσωπικό κωδικό του πελάτη (pin-number) και ελέγχει αν ο λογαριασμός του πελάτη είναι σε κατάσταση που επιτρέπει τη διενέργεια της συναλλαγής. Όλοι οι πελάτες των ΑΤΜ θα πρέπει να διατηρούν λογαριασμό σε κάποια από τις τράπεζες από τις οποίες αποτελείται το Δίκτυο Τραπεζών. 8. Ο υπολογιστής της τράπεζας αποστέλλει έναν αριθμό εξουσιοδότησης στον Κεντρικό Υπολογιστή του Δικτύου Τραπεζών. 9. Ο Κεντρικός Υπολογιστής διαγράφει τη συναλλαγή από το αρχείο του και αποστέλλει μήνυμα στο ΑΤΜ, επιβεβαιώνοντας τη διαδικασία της συναλλαγής.

Σημεία Επέκτασης:	**Βήμα 7.** Αν ο λογαριασμός του πελάτη δεν είναι σε θέση να υποστηρίξει τη συναλλαγή, ή ο προσωπικός κωδικός (pin-number) του πελάτη δεν είναι έγκυρος τότε εφάρμοσε την Περίπτωση Χρήσης **«Μη Έγκυρη Συναλλαγή»**
Σημείο Εκκίνησης:	Ο πελάτης εισάγει την κάρτα του σε ένα από τα ATMs του δικτύου τραπεζών.
Ημερομηνία:	5/1/2013.

Σχήμα 5.7 Περίπτωση Χρήσης Συναλλαγής ATM για το Πρόβλημα Δικτύου Τραπεζών

Επιλογή Ενώσεων:

Γενικώς, η κλάση Α και η κλάση Β ενώνονται όταν:

- Ένα αντικείμενο της κλάσης Α *αποστέλλει μήνυμα* σε ένα αντικείμενο της κλάσης Β.
- Ένα αντικείμενο της κλάσης Α *δημιουργεί ένα αντικείμενο* της κλάσης Β.
- Ένα αντικείμενο της κλάσης Α *έχει χαρακτηριστικά που οι αξίες τους είναι αντικείμενα* της κλάσης Β ή συλλογές αντικειμένων της κλάσης Β.
- Ένα αντικείμενο της κλάσης Α λαμβάνει μήνυμα, *του οποίου επιχείρημα είναι ένα αντικείμενο* της κλάσης Β.

Μοντέλο Κλάσεων για το Σύστημα ATM

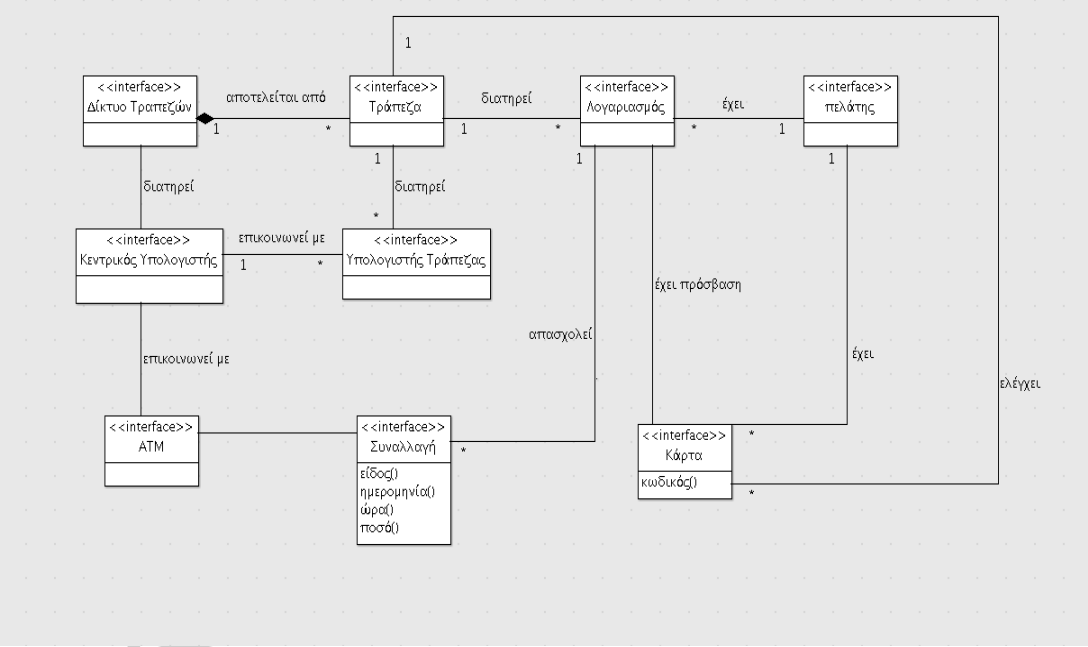
Λαμβάνοντας υπόψη τους παραπάνω κανόνες μπορούν να συνταχθούν οι ενδεχόμενες κλάσεις. Μία Κατηγοριοποίηση των κλάσεων φαίνεται στο Σχήμα 5.8.

Μοντέλο Κλάσεων: Σύστημα ATM

Καλές	Ασαφείς	Χαρακτηριστικό	Μετα-Γλώσσα	Γεγονότα	Περιττές	Εκτός Πλαισίου
Λογαριασμός	Σύστημα	Στοιχεία Λογ/μού			Χρήστης	Κόστος
ATM	Παροχή Ασφάλειας	Απόδειξη				
Τράπεζα	Διατήριση Αρχείου	Μετρητά				
	Παροχή					
Η/Υ Τράπεζας		Στοιχεία Λογ/μού				
Κάρτα		Στοιχεία Συν/γής				
Χρήστης Η/Υ						
Κεντρικός Η/Υ						
Δίκτυο Τραπ.						
Λογαριασμός						
Συναλλαγή						
Γκισές						

Σχήμα 5.8: Οι ενδεχόμενες κλάσεις του προβλήματος του δικτύου των ATM, σε κατηγορίες

Παρατήρηση 5.1: Η παραπάνω ανάγνωση του προβλήματος δεν είναι η μόνη σωστή. Οι επιλογές εξαρτώνται από το σύνολο των αντικειμένων που επιλέγονται για τη μοντελοποίηση του συστήματος.



Σχήμα 5.9: Οι ενδεχόμενες Κλάσεις του Δικτύου Τραπεζών

Ευθύνες

Για τη διευκόλυνση της διαδικασίας λήψης αποφάσεων σχετικά με τα γνωρίσματα και τις λειτουργίες των κλάσεων, είναι σημαντικό και πολύ χρήσιμο, στα διαγράμματα κλάσεων, να παρουσιάζονται οι ευθύνες κάθε κλάσης. Οι ευθύνες αφορούν στους ρόλους που διαδραματίζει κάθε κλάση, εντός του συστήματος, σε σχέση με το σύστημα. Για την αποτελεσματικότερη

δημιουργία του διαγράμματος και τη διευκόλυνση όσων θα πρέπει να το κατανοήσουν, συνηθίζεται να δημιουργείται μία άλλη κατηγορία, το κουτί των κλάσεων (class box). Σε αυτή τη κατηγορία εντάσσονται οι ευθύνες των κλάσεων.

5.5 Το Παράδειγμα με τον Εξομοιωτή Χειρουργικής Επέμβασης

Το συγκεκριμένο παράδειγμα είναι αρκετά πιο σύνθετο από το προηγούμενο.

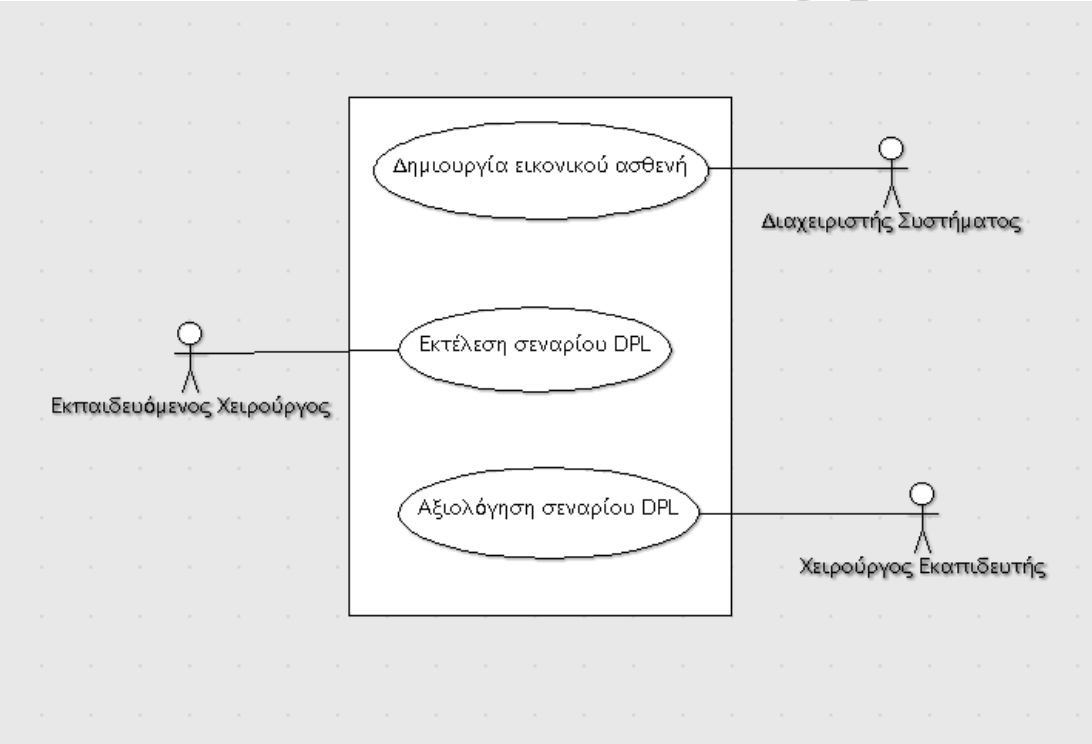
Η Δήλωση του Προβλήματος: (9)

«Η κοιλιακή χώρα του ανθρώπινου σώματος είναι μία ευαίσθητη περιοχή, στην οποία μπορεί να προκληθεί αμβλύ τραύμα με διάφορους δυνατούς τρόπους. Τροχαία ατυχήματα, πτώσεις ή αγώνες πάλης είναι μερικοί από αυτούς. Συνήθως, είναι δύσκολο να προσδιοριστεί αν υπάρχει ενδο-κοιλιακός τραυματισμός σε θύμα με αμβλύ τραύμα. Ο έλεγχος για το, δυνητικό, ενδο-κοιλιακό τραύμα πραγματοποιείται με διερευνητική επέμβαση. Όμως, η απόφαση για την εκτέλεση αυτής, είναι αρκετά δύσκολο και ριψοκίνδυνο ζήτημα. Μία διαδικασία που διευκολύνει τη διαδικασία λήψης τέτοιων αποφάσεων είναι η Διαγνωστική Περιτονιακή Πλύση (DPL-Diagnostic Peritoneal Lavage). Η DPL μπορεί να δώσει απαντήσεις σε ερωτήματα σχετικά με το, «αν υπάρχει ενδο-κοιλιακός τραυματισμός» και το «αν απαιτείται επέμβαση». Οι χειρούργοι θα πρέπει να εκπαιδεύονται σε μεθόδους, όπως η DPL, όμως η πρακτική άσκηση σε πτώματα είναι, πολλές φορές, απαγορευτική εξαιτίας του υπερβολικού κόστους της, ενώ τα ναρκωμένα ζώα μπορούν να χρησιμοποιούνται μόνο μία φορά. Ένας εξομοιωτής θα μπορούσε να διευκολύνει τη διαδικασία εκπαίδευσης με αποτελεσματικό τρόπο. Η DPL απαρτίζεται από αρκετά βήματα. Τα βασικότερα είναι τα εξής:

- *Ο χειρούργος εντοπίζει την περιοχή του τραύματος,*
- *Ο χειρούργος εφαρμόζει τοπική αναισθησία,*
- *Ο χειρούργος κάνει τομή,*
- *Ο χειρούργος εισάγει εργαλείο αποστράγγισης,*
- *Ο χειρούργος εισάγει καθετήρα,*
- *Πραγματοποιείται αναρρόφηση υγρού από την κοιλιακή χώρα,*
- *Γίνεται έλεγχος για την ύπαρξη αίματος στο υγρό που συλλέχτηκε.»*

Περιπτώσεις Χρήσης: Από την περιγραφή του προβλήματος προσδιορίζονται τρείς, κύριες περιπτώσεις χρήσης όσον αφορά στη λειτουργία του εξομοιωτή. Αυτές δίνονται στο σχήμα 5.10 που ακολουθεί.

Η περίπτωση χρήσης «Δημιουργία εικονικού ασθενή» αλληλεπιδρά με τον διαχειριστή του συστήματος, προκειμένου να δημιουργηθεί ένα μοντέλο ασθενούς με συγκεκριμένο τραύμα. Ο διαχειριστής χρησιμοποιεί ένα πρότυπο ασθενούς και προσδιορίζει σε αυτό την τοποθεσία και τις παραμέτρους του τραύματος. Ο εξομοιωτής αποκρίνεται δημιουργώντας και αποθηκεύοντας το μοντέλο «νέος ασθενής».



Σχήμα 5.10 Περιπτώσεις χρήσης εξομοιωτή DPL

Η περίπτωση χρήσης «Εκτέλεση σεναρίου DPL» είναι η κύρια περίπτωση χρήσης για τον εξομοιωτή. Η συγκεκριμένη περίπτωση χρήσης απεικονίζεται στο σχήμα 5.11. Η τελική περίπτωση χρήσης είναι η «Αξιολόγηση σεναρίου DPL», όπου ο χειρούργος-εκπαιδευτής επαναλαμβάνει και εξετάζει τις ενέργειες του εκπαιδευόμενου όσον αφορά στην περίπτωση χρήσης «Εκτέλεση σεναρίου DPL» και αξιολογεί την επίδοση του μαθητευόμενου.

Πιθανά Αντικείμενα, Κλάσεις και Ενώσεις

Η περίπτωση χρήσης είναι αρκετά επιφανειακή στην παρούσα φάση. Παρακάτω δίνονται μερικές υποθέσεις σχετικές με τον εξομοιωτή, οι οποίες,

στη συνέχεια, θα εξακριβωθούν με τη βοήθεια των πελατών και των χρηστών. Με αυτόν το τρόπο η περίπτωση χρήσης θα ολοκληρωθεί σε μεγαλύτερο βαθμό.

Πίνακας 5.1 Πιθανά Αντικείμενα για το Σενάριο DPL

Εικονικός Ασθενής	Σημείο	Κοιλιακό Τοίχωμα
Τομή	Αρχείο Ενεργειών	Τοπική Αναισθησία
Εργαλείο Τομής	Εργαλείο Σωληνίσκος	Σωληνίσκος
Κοιλιακή Περιττονία	Έντερο	Εργαλείο Οδηγός-Καλώδιο
Οδηγός-Καλώδιο	Εργαλείο Καθετήρας	Καθετήρας
Εργαλείο Σύριγγα	Σύριγγα	Υγρό

Παρατήρηση 5.2

1. Από το βήμα 3 των περιπτώσεων χρήσης και από την αλληλεπίδραση με τους χειρούργους μπορεί να εξαχθεί το συμπέρασμα ότι, το χειρουργικό νυστέρι χρησιμοποιείται για τη δημιουργία τομής.
2. Τα εργαλεία της περίπτωσης χρήσης, όπως η «κάνουλα», παρουσιάζουν παρόμοια χαρακτηριστικά μεταξύ τους και άρα απορρίπτονται από το μοντέλο ανάλυσης.
3. Ο εικονικός ασθενής συμπεριλαμβάνεται, ενώ θεωρείται ότι το κοιλιακό τοίχος είναι μέρος του εικονικού ασθενή. Επιπλέον, θεωρείται ότι, η περιττονία είναι μέρος του κοιλιακού τοίχους.
4. Θεωρείται ότι, οι σχέσεις μεταξύ των αντικειμένων προσδιορίζονται από την αλληλεπίδραση μεταξύ φυσικών αντικειμένων.

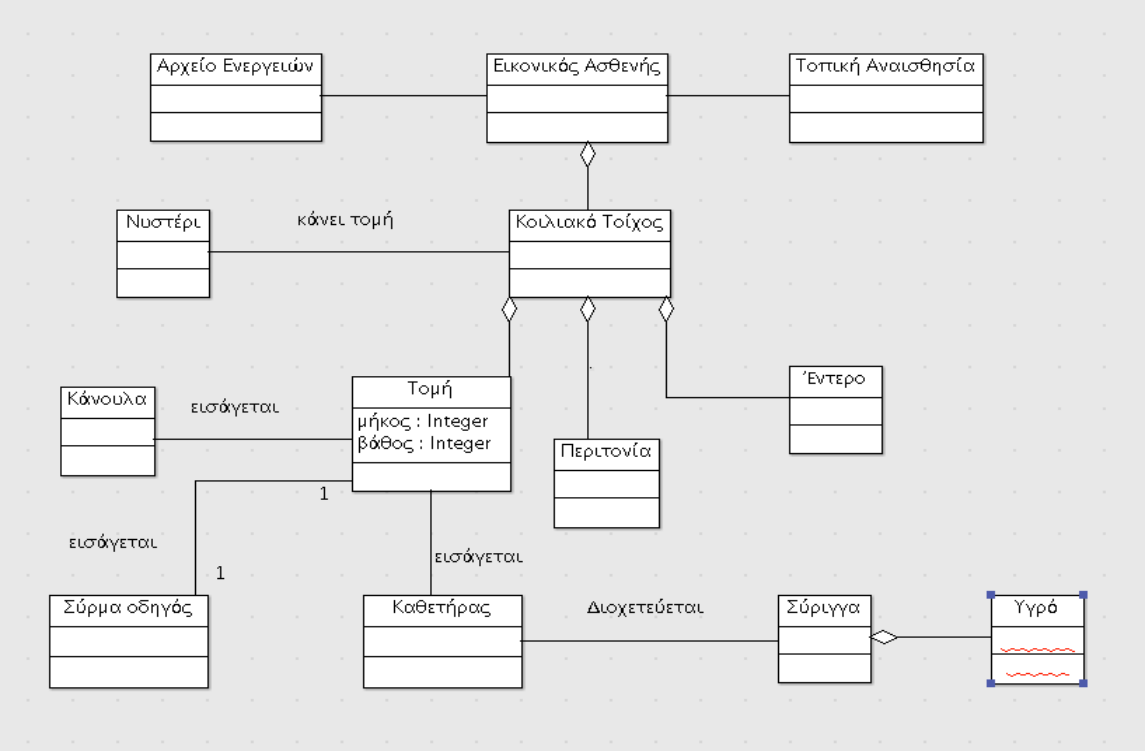
Όνομα Περίπτωσης Χρήσης:	**Σενάριο Διεξαγωγής DPL**
Επανάληψη:	**Πρόσοψη**
Περίληψη:	
Βασικά γεγονότα:	1. Ο εκπαιδευόμενος χειρούργος ή ο εκπαιδευτής φορτώνει έναν εικονικό ασθενή. 2. Ο εκπαιδευόμενος χειρούργος επιλέγει σημείο τομής στο κοιλιακό τοίχωμα του εικονικού ασθενή. Οι συντεταγμένες του

	σημείου τομής εξαρτώνται από το σημείο τραυματισμού. 3. Ο εξομοιωτής καταγράφει το σημείο προκειμένου να είναι εφικτή η περεταίρω ανάλυση του, από τον εκπαιδευτή. 4. Ο εκπαιδευόμενος χειρούργος θα πρέπει να εφαρμόσει τοπικό αναισθητικό και να περιμένει για ένα μικρό χρονικό διάστημα ώστε να επιδράσει. 5. Ο εκπαιδευόμενος χειρούργος κάνει τομή με τη χρήση του «εργαλείου τομής» που παρέχεται από τον εξομοιωτή. 6. Ο εκπαιδευόμενος χειρούργος επιλέγει το εργαλείο «σωληνίσκος» και εισάγει τον σωληνίσκο εντός της τομής. Ο εκπαιδευόμενος χειρούργος συνεχίζει να εισάγει τον σωληνίσκο μέχρι να διεισδύσει στην κοιλιακή περιτονία (μεμβράνη πίσω από το κοιλιακό τοίχωμα). Είναι σημαντικό, η εισαγωγή να γίνεται με προσοχή, ώστε να αποφεύγεται η διάτρηση του εντέρου. 7. Ο εκπαιδευόμενος χειρούργος επιλέγει το εργαλείο «οδηγός-καλώδιο» και το εισάγει στο σωληνίσκο. 8. Ο εκπαιδευόμενος χειρούργος αφαιρεί τον σωληνίσκο από τον εικονικό ασθενή. 9. Ο εκπαιδευόμενος χειρούργος επιλέγει το εργαλείο «καθετήρας» και το εφαρμόζει στον «οδηγό-καλώδιο».

	10. Ο εκπαιδευόμενος χειρούργος αφαιρεί τον «οδηγό-καλώδιο». 11. Ο εκπαιδευόμενος χειρούργος επιλέγει το εργαλείο «σύριγγα» και εισάγει τη σύριγγα στην άκρη του «καθετήρα». 12. Ο εκπαιδευόμενος χειρούργος αναρροφά υγρό από την κοιλιακή χώρα. Ο έλεγχος του υγρού για αίμα δεν είναι μέρος του σεναρίου.
Ρόλοι:	Εκπαιδευόμενος Χειρούργος
Σημείο Εκκίνησης:	Ο χειρούργος υποψιάζεται ότι υπάρχει αμβλύ κοιλιακό τραύμα. Οι σαρώσεις υπερήχων και οι CT δεν είναι διαθέσιμες.
Ημερομηνία:	5/1/2013.

Σχήμα 5.11: Η Περίπτωση Χρήσης «Διενέργεια DPL»

Το Μοντέλο Αντικειμένων της περίπτωσης παρουσιάζεται στο Σχήμα 5.12.



Σχήμα 5.12. Αρχικό μοντέλο για την DPL

Ευθύνες

Ο πίνακας 5.2 δείχνει τις ευθύνες που έχουν ανατεθεί σε κάθε αντικείμενο σύμφωνα με την υπάρχουσα θεώρηση. Η ευθύνη αποθήκευσης των ενεργειών ενός συγκεκριμένου μαθητευόμενου σε ένα συγκεκριμένο μοντέλο ασθενούς δεν αποτυπώνεται στο σχήμα, ενώ η ένωση μεταξύ του «αρχείου ασθενή» και του «εικονικού ασθενή» είναι τεχνητή. Επιπλέον, απαιτείται ένα αντικείμενο που θα έχει την ευθύνη ενίσχυσης και συντήρησης της κατάστασης της διαδικασίας. Αυτό το αντικείμενο θα ονομάζεται «Διαδικασία».

Παρακάτω γίνονται μερικές, επιπλέον, υποθέσεις:

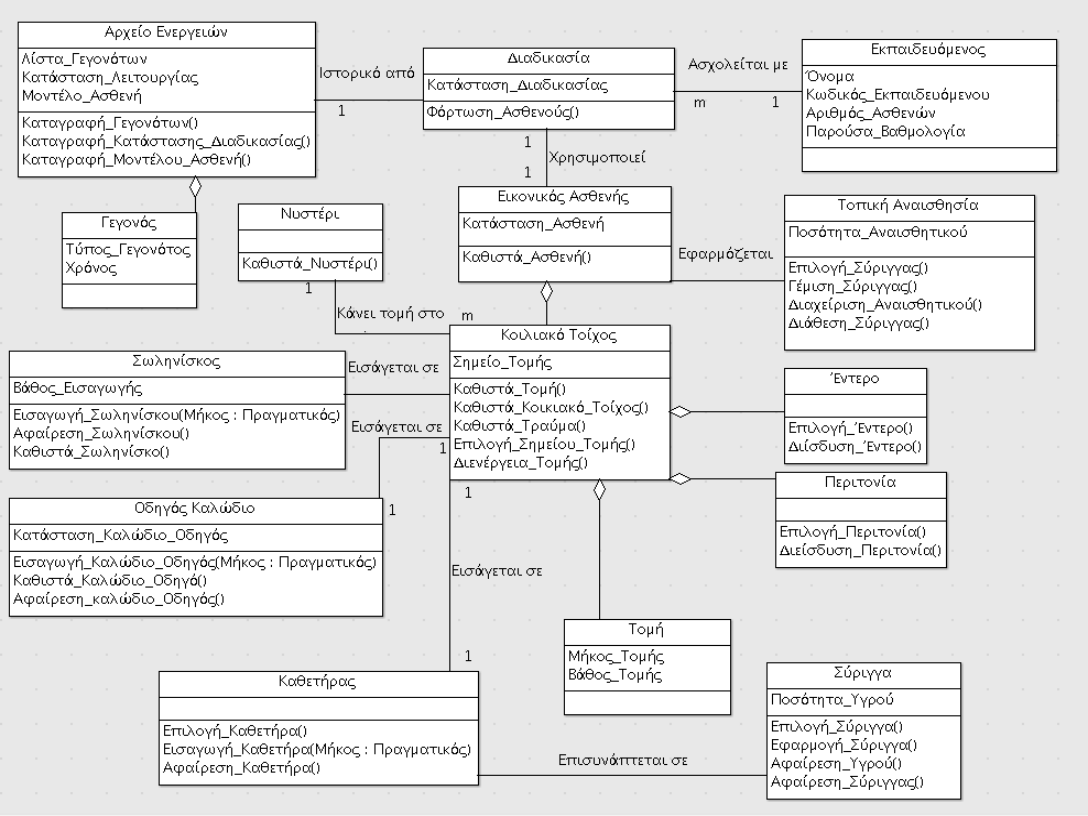
1. Κάθε ένα από τα αντικείμενα πρέπει να μπορεί να εμφανίζεται και να εκτελεί την λειτουργία για την οποία ευθύνεται.
2. Η κάνουλα πρέπει να μπορεί να είναι ορατή και να μπορεί να εισάγεται και να εξάγεται στην/από την κοιλιά.

Το μοντέλο των αντικειμένων όπως περιγράφηκε από τα παραπάνω δίνεται στο σχήμα 5.13.

Πίνακας 5.2 Ανάθεση ευθυνών στα αντικείμενα ως μέρος της διαδικασίας της ανάλυσης

Αντικείμενο/Κλάση	***Ευθύνη***
Εικονικός Ασθενής	Ευθύνες: 1. Διατηρεί τη κατάσταση του ασθενή, 2. Η παροχή ενός μοντέλου ασθενή για τη διεξαγωγή του σεναρίου. Ενέργειες: περιλαμβάνουν τη δυνατότητα να μπορεί να εμφανίζει τα κρίσιμα μέρη του εικονικού ασθενή.
Τοπική Αναισθησία	Ευθύνες: Διεξαγωγή της διαδικασίας εφαρμογής του αναισθητικού. Ενέργειες: περιλαμβάνουν τη δυνατότητα να γίνουν όλα τα βήματα, ώστε να αναισθητοποιηθεί ο ασθενής.
Κοιλιακό Τοίχωμα	Ευθύνες: 1. Παροχή ενός φυσικού μοντέλου της κοιλιακής χώρας για τη διαδικασία, 2. Η διατήρηση της τοποθεσίας και της κατάστασης του τραύματος, συμπεριλαμβάνοντας τους τύπους των υγρών που συλλέγονται από κάθε περιοχή και το σημείο τομής και την κατάσταση του ασθενή.
Τομή	Ευθύνες: Να αποτελεί μοντέλο της τομής.
Περιττονία	Ευθύνες: Να αποτελεί μοντέλο της περιττονίας. Ενέργειες: Ο έλεγχος για τη διείσδυση στην περιττονία.
Έντερο	Ευθύνες: Να αποτελεί μοντέλο του εντέρου. Ενέργειες: Ο έλεγχος για τη διείσδυση στο έντερο.
Νυστέρι	Ευθύνες: Να αποτελεί μοντέλο νυστεριού.
Σωληνίσκος	Ευθύνες: Να αποτελεί μοντέλο σωληνίσκου. Ενέργειες: Η εισαγωγή και εξαγωγή του σωληνίσκου.
Οδηγός-Καλώδιο	Ευθύνες: Να αποτελεί μοντέλο οδηγού-καλωδίου. Ενέργειες: Η εισαγωγή και εξαγωγή του οδηγού-καλώδιο.
Καθετήρας	Ευθύνες: Να αποτελεί μοντέλο καθετήρα. Ενέργειες: Η εισαγωγή και εξαγωγή του καθετήρα.
Σύριγγα	Ευθύνες: Να αποτελεί μοντέλο σύριγγας. Ενέργειες: Η επισύναψη της σύριγγας στον καθετήρα, η εξαγωγή του

	υγρού και η εξαγωγή της σύριγγας.
Αρχείο Ενεργειών	Ευθύνες: 1. Αποθήκευση σημαντικών γεγονότων και ενεργειών του χειρούργου για επανάληψη, 2. Αποθήκευση της κατάστασης μίας διαδικασίας για την επαναφορά της, αργότερα. Ενέργειες: Αποθήκευση γεγονότων, αποθήκευση της κατάστασης της διαδικασίας και διατήρηση λεπτομερειών για την παρούσα επίδοση του εκπαιδευόμενου χειρούργου.
Υγρό	Ευθύνες: Μη σαφείς.



Σχήμα 5.13: Το Μοντέλο των Αντικειμένων της περίπτωσης, πιο ολοκληρωμένο

Ερωτήσεις

Το σενάριο «Εκτέλεση σεναρίου DPL», προκειμένου να ολοκληρωθεί, απαιτείται να προσδιοριστούν μερικές λεπτομέρειες σε κάποια από τα υπο-μέρη του (9):

1. Κατά την καταγραφή των γεγονότων και των ενεργειών, «ποιές διεργασίες και ενέργειες είναι σημαντικές;», «Υπάρχουν πρότυπες διαδικασίες αξιολόγησης για τους εκπαιδευόμενους;».

2. Η προσέγγιση, όσον αφορά το αντικείμενο «Τοπική Αναισθησία» δεν είναι πλήρης. «Ποιά είναι η διαδικασία εφαρμογής τοπικής αναισθησίας;».
3. Δεν υπάρχει διαδικασία ολοκλήρωσης του σεναρίου DPL. «Ποιά είναι η διαδικασία ολοκλήρωσης;».
4. «Το μοντέλο της σύριγγας είναι σωστό;», «Απαιτούνται επιπλέον ενώσεις;», «Που πρέπει να διοχετευτεί το υγρό;».
5. Η περίπτωση χρήσης δεν παρέχει πλήρες μοντέλο δεδομένων όσον αφορά στους μαθητευόμενους, στους διαχειριστές, ή στους εκπαιδευτές χειρούργους. «Ποιοί είναι οι ρόλοι που διαδραματίζουν στο σύστημα;» και «τί πληροφορίες θα χρειαστούν;»

Όπως φαίνεται και από τα παραπάνω παραδείγματα σε ένα διάγραμμα κλάσεων (class diagram), αναπαρίστανται οι βασικές οντότητες του συστήματος και οι σχέσεις μεταξύ τους. Το διάγραμμα κλάσεων αποτελεί μια απεικόνιση της δομής ενός έργου Πληροφορικής.

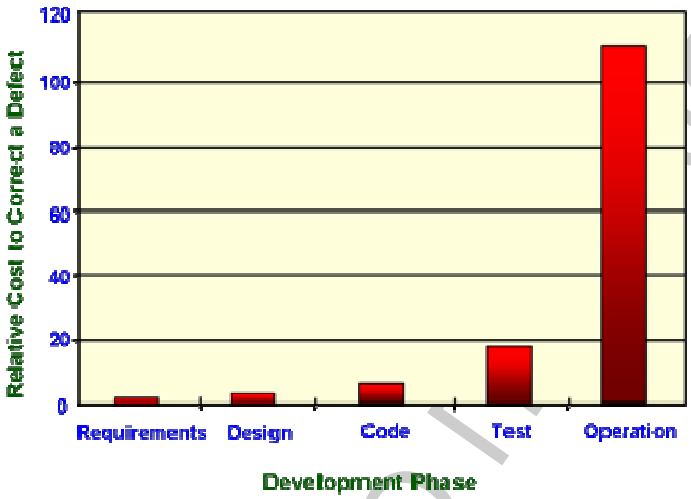
Η ανάλυση απαιτήσεων που έχει γίνει μέχρι τώρα θα πρέπει να έχει καταλήξει στην κατανόηση του επιχειρηματικού πλαισίου και των περιορισμών του πελάτη, στις λειτουργίες που θα πρέπει να εκτελεί το προϊόν, τα επίπεδα απόδοσης που θα πρέπει να τηρηθούν, καθώς και τα εξωτερικά συστήματα με τα οποία θα πρέπει να επικοινωνεί το σύστημα (42).

Αν το στάδιο της ανάλυσης απαιτήσεων διεξαχθεί αποτελεσματικά και έχουν προσδιοριστεί σωστά οι ανάγκες του πελάτες και οι απαιτήσεις από το νέο λογισμικό θα περιοριστεί η εμφάνιση λαθών (και αναθεωρήσεων) σε επόμενα στάδια του κύκλου ζωής του λογισμικού(43).

Λανθασμένος προσδιορισμός απαιτήσεων, οδηγεί σε ανάπτυξη λανθασμένου λογισμικού που δεν ικανοποιεί τις ανάγκες του πελάτη(44).

Λανθασμένες απαιτήσεις σημαίνουν επιπρόσθετη εργασία διορθώσεων. Η επιπρόσθετη εργασία δημιουργεί προβλήματα στη τήρηση χρονοδιαγραμμάτων και προκαλεί καθυστερήσεις στην παράδοση του τελικού προϊόντος. Επιπλέον, επιπρόσθετη εργασία σημαίνει μεγαλύτερές δαπάνες για την οργανισμό πληροφορικής που αναπτύσσει το έργο σπατάλη χρόνου που θα μπορούσε να είχε αποφευχθεί.

Η κύρια συνέπεια των λανθασμένων απαιτήσεων είναι να ξανακάνεις κάτι, το οποίο θεωρούσες πως έχει ήδη υλοποιηθεί. Στο Σχήμα 5.14, που βασίζεται σε δεδομένα της Hewlett-Packard, φαίνεται πως κοστίζει πολύ περισσότερο να διορθωθεί μια λανθασμένη απαίτηση που εντοπίστηκε στο τέλος του έργου, παρά στις διορθώσεις που οφείλονται σε εντοπισμό σφαλμάτων σε αρχικά στάδια στον κύκλο ζωής ενός λογισμικού.



Σχήμα 5.14: Σχετικό κόστος διορθώσεων σε ένα λογισμικό ανάλογα το στάδιο ανάπτυξης (45)

Η αποτελεσματικότητα ενός οργανισμού Πληροφορικής, εξαρτάται κυρίως από τα λάθη που γίνονται στην Ανάλυση Απαιτήσεων. **Μια Ανάλυση Απαιτήσεων χωρίς σφάλματα, έχει ως συνέπεια τη μείωση του χρόνου υλοποίησης του λογισμικού, ταχύτερη παράδοση των έργων, δυνατότητα υλοποίησης περισσότερων έργων στην ίδια μονάδα χρόνου, χαμηλότερο κόστος, καλύτερα προϊόντα και ευχαριστημένους πελάτες**.

5.6 Διαγράμματα Αλληλεπίδρασης

Τα διαγράμματα κλάσεων είναι κατάλληλα για την έκφραση της «δομής» ενός μοντέλου. Για την ολοκλήρωση της ανάλυσης θα πρέπει να αναπτυχθεί το μοντέλο έκφρασης της «συμπεριφοράς» του συστήματος. Με την έννοια «συμπεριφορά» νοείται μία σειρά γεγονότων ή λειτουργιών-ενεργειών:

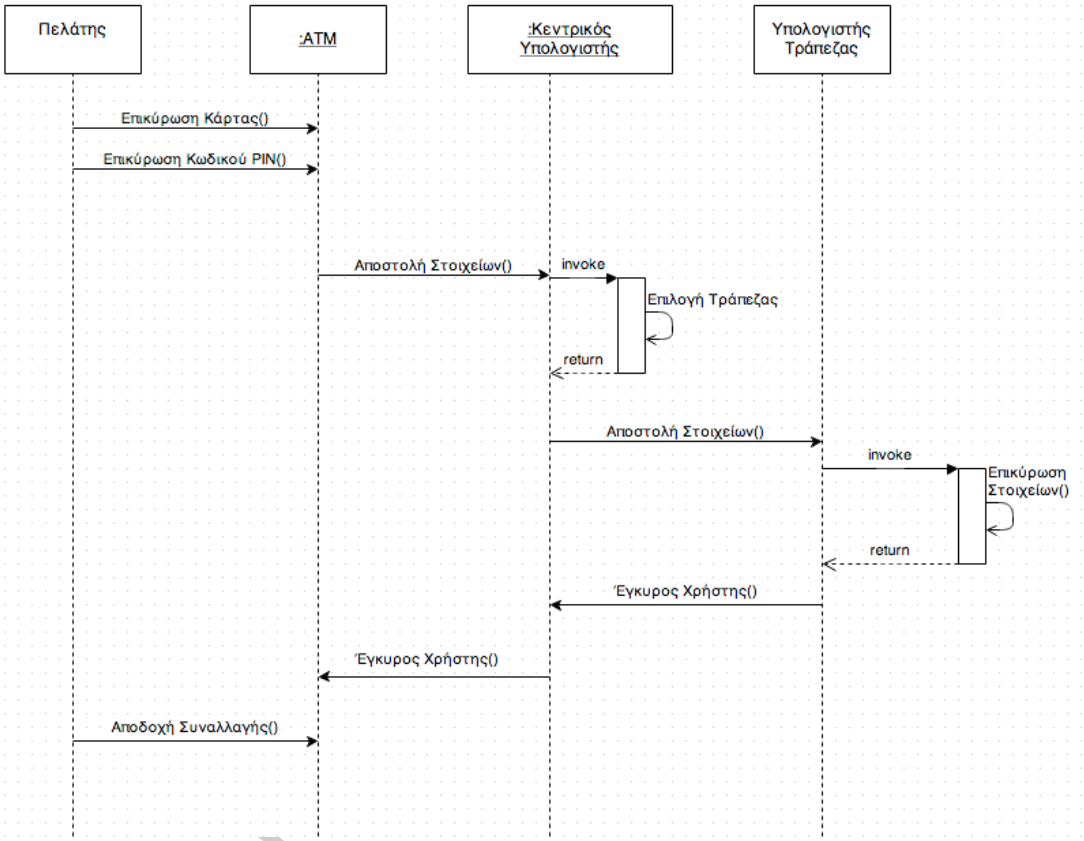
- Εξωτερικά γεγονότα που ξεκινούν από τους ρόλους
- Εσωτερικά γεγονότα ή μηνύματα που ξεκινούν από αντικείμενα
- Λειτουργίες που αναλαμβάνονται από αντικείμενα και ρόλους

Τα μοντέλα προσδιορισμού της συμπεριφοράς του συστήματος προορίζονται για την απεικόνιση του δυναμικού περιεχομένου και της εξέλιξης

του συστήματος. Τα κύρια στοιχεία της γλώσσας UML για την απεικόνιση της συμπεριφοράς του συστήματος είναι:

- Διαγράμματα Συνεργασίας, στα οποία συμπεριλαμβάνονται τα Ακολουθιακά Διαγράμματα και τα Διαγράμματα Αλληλεπίδρασης
- Διαγράμματα Δραστηριότητας και
- Διαγράμματα Κατάστασης

Στο σχήμα 5.15 δίνεται ένα παράδειγμα ενός ακολουθιακού διαγράμματος.



Σχήμα 5.15: Ένα Διάγραμμα Αλληλεπίδρασης για το παράδειγμα με τα ATM.

Τα ακολουθιακά διαγράμματα περιγράφουν τη συνεργασία των ρόλων και των αντικειμένων προκειμένου να εκτελεστούν οι διεργασίες του συστήματος. Επιπλέον αναδεικνύουν την ακολουθία των γεγονότων και των μηνυμάτων σε σχέση με το χρόνο.

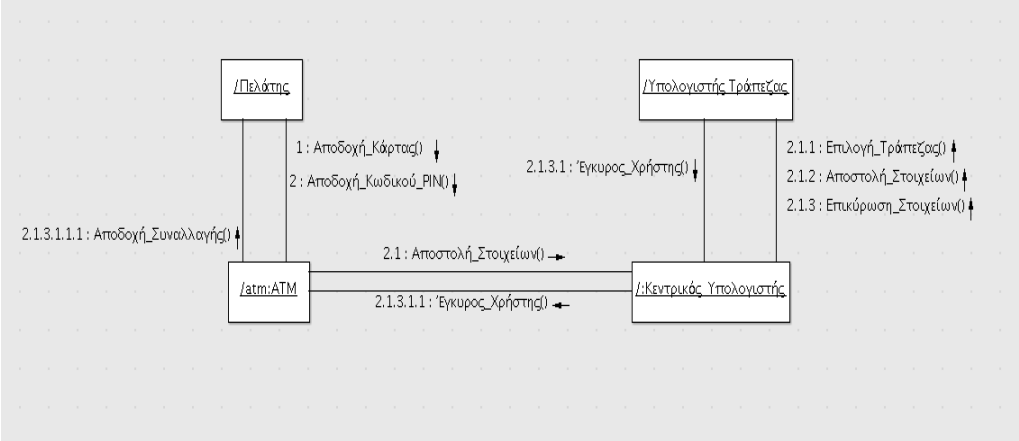
Τα αντικείμενα και οι ρόλοι δίνονται στην κορυφή του διαγράμματος. Περιλαμβάνονται:

- Ρόλοι
- Συγκεκριμένα Αντικείμενα (Πελάτης Α :Πελάτης)

- Ανώνυμα Αντικείμενα που ικανοποιούν ένα είδος δηλώσεων (:Πελάτης)

Τα μηνύματα και τα γεγονότα παρουσιάζονται από τις κλήσεις λειτουργίας στα αντικείμενα που βρίσκονται στην κορυφή του διαγράμματος, ενώ η σειρά των μηνυμάτων στο χρόνο παρουσιάζεται στο κάθετο μέρος και στο κάτω μέρος του διαγράμματος. Το βέλος προσδιορίζει το αντικείμενο στο οποίο ανήκει η λειτουργία και η ουρά του αντικειμένου προσδιορίζει το αντικείμενο που επικαλείται τη λειτουργία. Για παράδειγμα, η «Αποδοχή-Συναλλαγής» είναι μία λειτουργία στην κλάση ATM και επικαλείται από τον ρόλο «Πελάτης».

Μία άλλη μορφή του διαγράμματος που εκφράζει τον τρόπο με τον οποίο τα αντικείμενα αλληλεπιδρούν για να παράξουν συμπεριφορές είναι το Διάγραμμα Συνεργασίας. Το Διάγραμμα Συνεργασίας αντιστοιχίζεται με το Ακολουθιακό Διάγραμμα του σχήματος 5.16 στο σχήμα 5.17 (9).



Σχήμα 5.16 Ένα Διάγραμμα Συνεργασίας για το Παράδειγμα με τα ATM

Τα διαγράμματα συνεργασίας προσδιορίζουν, κυρίως, τις σχέσεις μεταξύ των αντικειμένων. Τα Ακολουθιακά Διαγράμματα, από την άλλη μεριά, προσδιορίζουν τη σειρά των γεγονότων και την ανταλλαγή μηνυμάτων στο χρόνο. Στα διαγράμματα συνεργασίας η μοντελοποίηση των γεγονότων και των ενεργειών προσδιορίζεται με τη βοήθεια αριθμών σειράς, ενώ η κατεύθυνση των μηνυμάτων δίνεται από τα βέλη. Οι γραμμές απεικονίζουν σχέσεις, ή συνδέσμους μεταξύ αντικειμένων.

Τα ακολουθιακά διαγράμματα και τα διαγράμματα συνεργασίας αντλούν την ίδια πληροφορία, όμως η αναπαράσταση της πληροφορίας διαφέρει. Τέλος, το ένα διάγραμμα μπορεί να δημιουργήσει το άλλο (40).

5.6.1 Βασικές Χρήσεις των Διαγραμμάτων Αλληλεπίδρασης

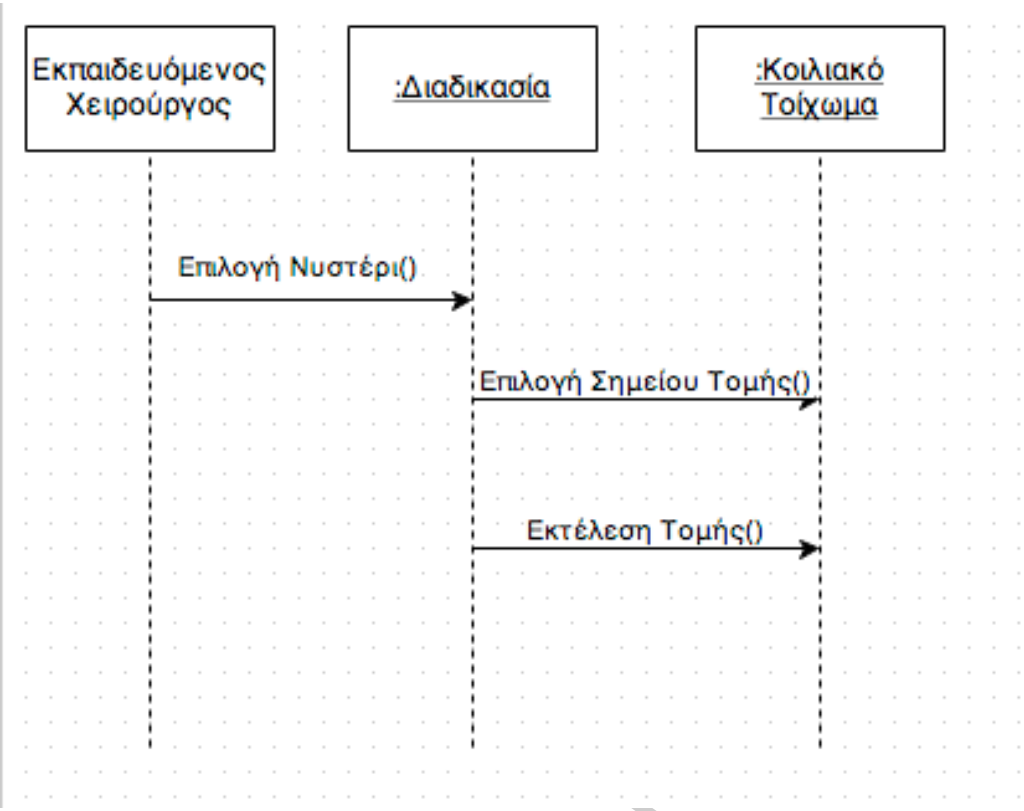
Γενικώς, υπάρχουν διάφοροι τρόποι χρήσης των διαγραμμάτων αλληλεπίδρασης, χρησιμοποιούνται όμως, κυρίως, για τη μοντελοποίηση των δυναμικών πτυχών:

- Του συστήματος, ως σύνολο
- Των υπο-συστημάτων, πακέτων, συνιστωσών
- Μίας κλάσης, ή μίας λειτουργίας

Το επίπεδο αφαίρεσης-συγκέντρωσης θα υπαγορεύσει τα αντικείμενα, τους ρόλους και τα μηνύματα που εμπλέκονται στο διάγραμμα συνεργασίας. Θα πρέπει να αποφασιστεί «αν» είναι πιο σημαντική η συγκέντρωση στη δομή της αλληλεπίδρασης και στη σειρά των μηνυμάτων (collaboration diagram), ή ο χρόνος της σειράς των μηνυμάτων (sequence diagram) (40).

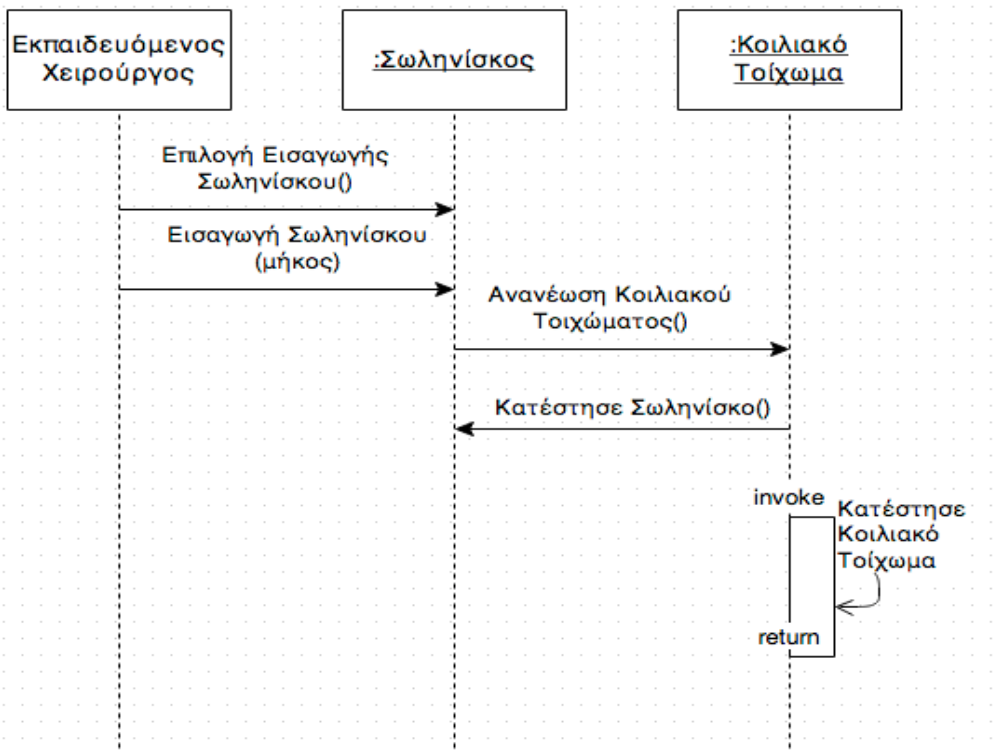
5.7 Ακολουθιακά Διαγράμματα

Στα σχήματα 5.17 και 5.18 δίνονται τα ακολουθιακά διαγράμματα της περίπτωσης του σεναρίου DPL. Παρατηρώντας το σχήμα 5.17 διαφαίνεται ότι για την ολοκλήρωση της αλληλεπίδρασης προστέθηκε η λειτουργία «Επιλογή Νυστεριού». Αυτή η λειτουργία είναι απαραίτητη διότι σε κάθε άλλη περίπτωση ο ρόλος δεν έχει λόγο να επικαλεστεί τη διαδικασία της τομής. Εξερευνώντας το μοντέλο, διαφαίνεται η παράλειψη μίας λειτουργίας για την επίκληση της διαδικασίας στο μοντέλο (σχήμα 5.15) και στην περίπτωση χρήσης. Η μελέτη της αλληλεπίδρασης με τους πελάτες θα διευκολύνει τη διαδικασία κατανόησης του τρόπου επίκλησης όλων των εργαλείων.



Σχήμα 5.17: Ένα Ακολουθιακό Διάγραμμα για την εκτέλεση τομής του Παραδείγματος «Εξομοιωτής Χειρουργικής Επέμβασης»

Το ακολουθιακό διάγραμμα που φαίνεται στο σχήμα 5.19 εμφανίζει μερικά προβλήματα του μοντέλου, όμως δεν είναι σαφές αν υπάρχει πρόβλημα με την περίπτωση χρήσης. Και πάλι ο μαθητευόμενος χειρούργος θα πρέπει να επιλέξει να εισάγει το εργαλείο κάννουλα (Επιλογή Εισαγωγής Κάννουλας), όμως επιπρόσθετα, μεταξύ των αντικειμένων «Κάννουλα» και «Κοιλιακό Τοίχος» θα πρέπει να διακινηθεί πληροφορία. Η κατάσταση της διαδικασίας θα πρέπει να μεταβληθεί προκειμένου να αντιδράσει στο γεγονός της εισαγωγής της κάννουλας. Η εισαγωγή της κάννουλας είναι μία λειτουργία που έχει διάρκεια. Απαιτείται λοιπόν η εξομοίωση να αντιδρά στα φανερά και τα κρυφά μέρη της κάννουλας για όσο εισάγεται στο κοιλιακό τοίχος (40).



Σχήμα 5.18: Ένα Ακολουθιακό Διάγραμμα για την Εισαγωγή του Σωληνίσκου

5.8 Διαγράμματα Κατάστασης

Μία πολύ αποτελεσματική περιγραφή των διαγραμμάτων κατάστασης δίνεται από τον David Harel (46).

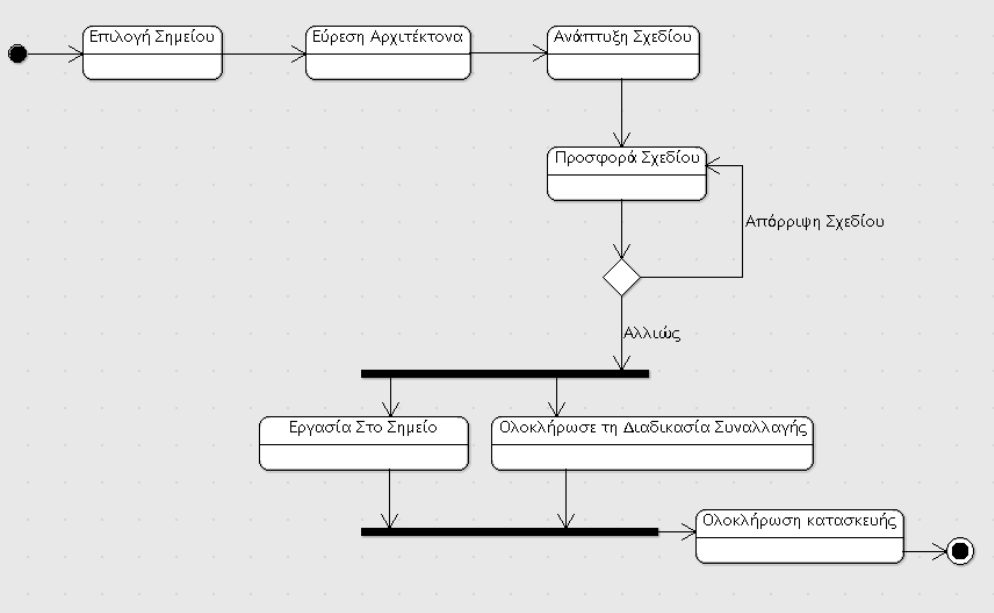
Προκειμένου να προσδιοριστεί ο τρόπος χρήσης των διαγραμμάτων κατάστασης θα γίνει αναφορά στον εξομοιωτή DPL. Οι απαιτήσεις της περίπτωσης χρήσης «DPL» μπορούν να αναλυθούν με τη βοήθεια των διαγραμμάτων κατάστασης. Συγκεκριμένες ενέργειες του χειρούργου έχουν νόημα μόνο αν η διαδικασία έχει φτάσει σε ένα ορισμένο στάδιο. Γενικά, τα διαγράμματα κατάστασης είναι ιδανικά για την περιγραφή και την ανάλυση συστημάτων, όπου είναι λογικό να γίνονται κάποιες ενέργειες όταν και εφόσον το σύστημα βρίσκεται σε μία συγκεκριμένη κατάσταση. Αυτό συμβαίνει και στην περίπτωση του εξομοιωτή DPL (46).

Η συγκεκριμένη προσέγγιση μοντελοποίησης προορίζεται για τη μοντελοποίηση γεγονότων ή ενεργειών με μεταβάσεις. Η «παραμετροποίηση» του εικονικού ασθενούς ύστερα από μία συγκεκριμένη σειρά ενεργειών από τον χειρούργο μοντελοποιείται από την κατάσταση. Για παράδειγμα, αν ο εικονικός ασθενής μπορεί να «παραμετροποιηθεί», όσον αφορά στο σημείο εισαγωγής της κάνουλας και το σύρμα-οδηγός έχει εισαχθεί στην κάννουλα. Δεν είναι

δυνατό να επισυνάπτεται η σύριγγα στον σωληνίσκο σε αυτή την κατάσταση (46).

5.9 Διαγράμματα Δραστηριότητας

Τα διαγράμματα δραστηριότητας χρησιμοποιούνται, κυρίως, προκειμένου να καθοριστούν τα δυναμικά μέρη του συστήματος. Χρησιμοποιούνται για τον καθορισμό της ροής του ελέγχου σε συστήματα και υποσυστήματα από δραστηριότητα σε δραστηριότητα. Αυτό μπορεί να οδηγήσει σε ακολουθιακή ροή του ελέγχου ή ταυτόχρονη ροή ελέγχου της υπολογιστικής διαδικασίας. Η κύρια διαφορά μεταξύ των διαγραμμάτων αλληλεπίδρασης και των διαγραμμάτων δραστηριοτήτων είναι ότι, τα πρώτα αναδεικνύουν τη ροή των μηνυμάτων μεταξύ αντικειμένων, ενώ τα διαγράμματα δραστηριότητας αναδεικνύουν τη ροή ελέγχου μεταξύ δραστηριοτήτων. Στο Σχήμα 5.19 φαίνεται ένα διάγραμμα δραστηριότητας για την περίπτωση οικοδόμησης ενός σπιτιού(40).



Σχήμα 5.19: Ένα Διάγραμμα Δραστηριότητας περίπτωσης κατασκευής σπιτιού.

Τα διαγράμματα δραστηριότητας αποτελούνται από(40):

- Δραστηριότητες οι οποίες είναι μη-ατομικές ενέργειες. Διευκολύνει πολύ τη σκέψη η παρομοίωση των δραστηριοτήτων με ενέργειες που δημιουργούνται από μία σειρά υπολογισμών.
- Κλάδους που αποτελούνται από «φύλακες», οι οποίοι επιτρέπουν στη ροή του ελέγχου να περνά κατά μήκος της κιβωτού που συνδέεται με τον φύλακα.

- Παράλληλα σύνολα δραστηριοτήτων που επισημαίνονται από την «ταυτόχρονη ένταξη». Τα παράλληλα θέματα επανέρχονται και πάλι μαζί σε ένα μεμονωμένο θέμα ελέγχου.

Όταν η σειρά των βημάτων που αντιστοιχεί σε μία δραστηριότητα ολοκληρώνεται, τότε η ροή ελέγχου περνά άμεσα στην επόμενη δραστηριότητα κατάστασης (και την πρώτη ατομική ενέργεια σε αυτή τη δραστηριότητα). Η κλαδοποίηση είναι εφικτή, επίσης, όπου η ροή ελέγχου έχει επιλογή να περάσει κατά μήκος ενός αριθμού κιβωτών. Οι κιβωτοί φυλάσσονται από εκφράσεις. Αν η έκφραση είναι αληθής, τότε η ροή ελέγχου μπορεί να περάσει κατά μήκος της κιβωτού. Τελικά, μπορούν να υπάρχουν ταυτόχρονα θέματα ελέγχου που επισημαίνονται από γραμμές «εντάξεων». Η ροή ελέγχου διαχωρίζεται από μία γραμμή και οι ακολουθίες των δραστηριοτήτων λαμβάνουν χώρα παράλληλα, έως ότου να επανασυντεθούν σε μία σειρά ένταξης. Μία σειρά ένταξης απαιτεί να μην ενταχθούν μαζί όλα τα θέματα ελέγχου(40).

6 Προδιαγραφή και Επικύρωση Απαιτήσεων

6.1 Εισαγωγή

Ως εδώ αναπτύχθηκαν ζητήματα σχετικά με την εξόρυξη απαιτήσεων, την ανάλυση απαιτήσεων και τη μοντελοποίηση με τη βοήθεια της γλώσσας UML. Η τελική φάση του πεδίου της Μηχανικής Απαιτήσεων αφορά στην Προδιαγραφή και Επικύρωση των Απαιτήσεων. Σε αυτή τη φάση, τα παράγωγα των φάσεων της εξόρυξης και της ανάλυσης παρουσιάζονται με σαφή και ξεκάθαρο τρόπο σε μορφή κατάλληλη για την διευκόλυνση της διαδικασίας του έργου ανάπτυξης.

6.2 Ο σκοπός της προδιαγραφής απαιτήσεων

Η προδιαγραφή των απαιτήσεων λογισμικού (Software Requirements Specification-SRS) χρησιμοποιείται για την καταγραφή και επικοινωνία των απαιτήσεων ανάμεσα στα διάφορα εμπλεκόμενα μέρη. Ιδανικά, εξυπηρετεί πληθώρα σκοπών (12):

- Αποτελεί τη βάση συμφωνίας μεταξύ των διάφορων εμπλεκόμενων μερών όσον αφορά στο «τί» θα κάνει το σύστημα.
- Οι λειτουργίες του συστήματος που περιγράφονται στο SRS μπορούν, δυνητικά, να διευκολύνουν τους χρήστες όσον αφορά στην επιτυχία στόχευσης των απαιτήσεων προς την κατεύθυνση των πραγματικών αναγκών των χρηστών, ενώ προσδιορίζουν τον τρόπο που το λογισμικό πρέπει να τροποποιηθεί ώστε να εξυπηρετηθούν οι ανάγκες αυτές.
- Το SRS μπορεί να διευκολύνει στην μείωση της προσπάθειας για την ανάπτυξη αποτελώντας πηγή απαντήσεων σε ερωτήσεις και πηγή αναφοράς λεπτομερειών σχετικά με το πρόβλημα που πρέπει να επιλυθεί. Στο SRS διαφαίνονται και τα κίνητρα του έργου.
- Η προετοιμασία του SRS παροτρύνει τα διάφορα εμπλεκόμενα μέρη, ώστε να εξετάσουν αυστηρά όλες τις απαιτήσεις πριν ξεκινήσει ο σχεδιασμός. Με αυτόν το τρόπο μειώνεται η ανάγκη για επανασχεδιασμό του έργου, για επαναδιαμόρφωση και επανεξέταση του κώδικα. Η προσεκτική επανεξέταση των απαιτήσεων στο SRS μπορεί να αποκαλύψει παραλείψεις, παρανοήσεις και ασυνέπειες κατά τα πρώτα στάδια του κύκλου ανάπτυξης, όταν ακόμα αυτά τα ζητήματα είναι, σχετικά, εύκολο να διορθωθούν.

- Η παροχή βάσης για εκτίμηση κόστους και πλάνων-προγραμμάτων. Η περιγραφή του προϊόντος που πρέπει να αναπτυχθεί όπως περιγράφεται στο SRS είναι η βάση για την εκτίμηση του κόστους του έργου και του χρονοπρογράμματος. Επιπλέον μπορεί να χρησιμεύσει για την αποδοχή τιμών και προσφορών.
- Το SRS μπορεί να παρέχει βάση για ανάπτυξη αναφοράς πλάνων ελέγχου. Οι ομάδες μπορούν να αναπτύξουν τα πλάνα ελέγχου πιο παραγωγικά από ένα καλό SRS που περιγράφει απαιτήσεις με κατανοητό και ελεγχόμενο τρόπο.
- Ένα καλό SRS διευκολύνει την υιοθέτηση του προϊόντος λογισμικού από νέους χρήστες ή νέες μηχανές.
- Ένα καλό SRS εξυπηρετεί ως βάση για διορθωτική συντήρηση και βελτίωση του λογισμικού. Ένα καλό SRS θα αναφέρεται στο προϊόν που αναπτύχθηκε και όχι στις διαδικασίες που το ανέπτυξαν. Το SRS εξυπηρετεί ως βάση για βελτίωση του τελικού προϊόντος και παρέχει ένα καλό σημείο εκκίνησης.

6.2.1 Περίγραμμα του Εγγράφου Προδιαγραφής Απαιτήσεων (SRS)

Οι απαιτήσεις μπορούν να προδιαγραφούν με τη χρήση φυσικής γλώσσας διαγραμμάτων ή τυπικών μεθόδων. Ιδανικά, ο σκοπός του εγγράφου απαιτήσεων είναι να μεταφέρει μία κατανόηση του προβλήματος καθώς και να προσδιορίσει, «τί» απαιτείται να παραχθεί, παρασχεθεί, μετατραπεί ή ολοκληρωθεί, προκειμένου να εξυπηρετηθούν οι ανάγκες των πελατών, όσον αφορά στην επίλυση του προβλήματος που έχουν (47).

Ένας οδηγός περιεχομένου ενός SRS δίνεται στο πρότυπο IEEE Std. 830-1998. Τα Σχήματα 6.1, 6.2, 6.3, 6.4, 6.5 δίνουν τη σειρά προτεινόμενων περιγραμμάτων για τη δημιουργία του SRS (47).

Πίνακας Περιεχομένων

1.Εισαγωγή
1.1 Σκοπός
1.2 Αντικείμενο
1.3 Ορισμοί, Ακρωνύμια, Συντομογραφίες
1.4 Αναφορές
1.5 Επισκόπηση
2. Γενική Περιγραφή Προτεινόμενου Συστήματος
2.1 Πλαίσιο Συστήματος και Προοπτικές
2.2 Λειτουργίες Συστήματος
2.3 Χαρακτηριστικά Χρηστών
2.4 Περιορισμοί
2.5 Υποθέσεις και Εξαρτήσεις
3. Ειδικές Απαιτήσεις
Παραρτήματα

Σχήμα 6.1: Περίγραμμα Εγγράφου Προδιαγραφής Απαιτήσεων (SRS). Από το Πρότυπο IEEE Std. 830-1998

3.Ειδικές Απαιτήσεις
3.1 Εξωτερική Διεπαφή Απαιτήσεων
3.1.1 Διεπαφές Χρηστών
3.1.2 Διεπαφές Υλικού
3.1.3 Διεπαφές Λογισμικού
3.1.4 Διεπαφές Επικοινωνιών
3.2 Λειτουργικές Απαιτήσεις
3.2.1 Λειτουργική Απαίτηση 1
.
.
.
3.2.N Λειτουργική Απαίτηση N
3.3 Απαιτήσεις Επίδοσης
3.4 Περιορισμοί Σχεδιασμού
3.5 Χαρακτηριστικά Λογισμικού
3.6 Άλλες Απαιτήσεις

Σχήμα 6.2: Η 3η ενότητα του Εγγράφου Προδιαγραφής Απαιτήσεων (SRS), δομημένο σύμφωνα με τις Κλάσεις Απαιτήσεων. Από το Πρότυπο IEEE Std. 830-1998

3. Ειδικές Απαιτήσεις

3.1 Εξωτερικές Διεπαφές Απαιτήσεων

3.1.1 Διεπαφές Χρηστών

3.1.2 Διεπαφές Υλικού

3.1.3 Διεπαφές Λογισμικού

3.1.4 Διεπαφές Επικοινωνιών

3.2 Λειτουργικές Απαιτήσεις

3.2.1 Λειτουργία 1

3.2.1.1 Λειτουργική Απαίτηση 1.1

.
.
.

3.2.1.N Λειτουργική Απαίτηση 1.N

3.2.2 Λειτουργία 2

.
.
.

3.2.M Λειτουργία M

3.2.M.1 Λειτουργική Απαίτηση M.1

.
.
.

3.2.M.N Λειτουργική Απαίτηση M.N

3.3 Απαιτήσεις Επίδοσης

3.4 Περιορισμοί Σχεδιασμού

3.5 Χαρακτηριστικά Λογισμικού

3.6 Άλλες Απαιτήσεις

Σχήμα 6.3: Η 3η Ενότητα του Εγγράφου Προδιαγραφής Απαιτήσεων δομημένη σύμφωνα με τις Λειτουργίες του Συστήματος. Από το Πρότυπο IEEE Std. 830-1998

3. Ειδικές Απαιτήσεις
3.1 Εξωτερικές Διεπαφές Απαιτήσεων
3.1.1 Διεπαφές Χρηστών
3.1.2 Η Διεπαφές Υλικού
3.1.3 Διεπαφές Λογισμικού
3.1.4 Διεπαφές Επικοινωνιών
3.2 Κλάσεις/Αντικείμενα
3.2.1 Κλάση/Αντικείμενο 1
3.2.1.1 Χαρακτηριστικά
3.2.1.1.1 Χαρακτηριστικό 1
.
.
.
3.2.1.1.Ν Χαρακτηριστικό Ν
3.2.1.2 Λειτουργίες (υπηρεσίες, μέθοδοι)
3.2.1.2.1 Λειτουργική Απαίτηση 1.1
.
.
.
3.2.1.2.Μ Λειτουργική Απαίτηση 1.Μ
3.2.1.3 Μηνύματα (σταλμένα ή λήψεις)
3.2.2 Κλάση/Αντικείμενο 2
.
.
.
3.2.Π Κλάση/Αντικείμενο Π
3.3 Απαιτήσεις Επίδοσης
3.4 Περιορισμοί Σχεδιασμού
3.5 Χαρακτηριστικά Λογισμικού
3.6 Άλλες Απαιτήσεις

Σχήμα 6.4: Η 3η Ενότητα του Εγγράφου Προδιαγραφής Απαιτήσεων δομημένο σύμφωνα με τα Αντικείμενα. Από το Πρότυπο IEEE Std. 830-1998

3. Ειδικές Απαιτήσεις

3.1 Εξωτερικές Διεπαφές Απαιτήσεων

3.1.1 Διεπαφές Χρηστών

3.1.2 ΗΔιεπαφές Υλικού

3.1.3 Διεπαφές Λογισμικού

3.1.4 Διεπαφές Επικοινωνιών

3.2 Περιπτώσεις Χρήσης

3.2.1 Περίπτωση Χρήσης 1

3.2.2 Περίπτωση Χρήσης 2

.

.

.

3.2.Ν Περίπτωση Χρήσης Ν

3.3 Απαιτήσεις Επίδοσης

3.4 Περιορισμοί Σχεδιασμού

3.5 Χαρακτηριστικά Λογισμικού

3.6 Άλλες Απαιτήσεις

Σχήμα 6.5: Η 3η Ενότητα του Εγγράφου Προδιαγραφής Απαιτήσεων (SRS) δομημένο σύμφωνα με τις Περιπτώσεις Χρήσης. Από το Πρότυπο IEEE Std. 830-1998

6.2.2 Ιδιότητες ενός καλού Εγγράφου Προδιαγραφής Απαιτήσεων (SRS)

Η δομή της προδιαγραφής απαιτήσεων διαφέρει από έργο σε έργο αλλά, τα πρότυπα μπορούν να αποτελέσουν πηγή έμπνευσης της δομής. Όπως έχει αναφερθεί προηγούμενα, τα πρότυπα πρέπει να επιλέγονται για το εκάστοτε έργο. Τα πρότυπα formats όπως αυτά στο (27) δεν είναι κατάλληλα για όλα τα έργα και ίσως θα έπρεπε να προστίθενται νέα τμήματα ή υπο-τμήματα με σκοπό να δημιουργείται ένα περίγραμμα σχετικό με το έργο. Το κρίσιμο σημείο είναι η πλήρης αποσαφήνιση και καταγραφή όλων των αποκλίσεων από το πρότυπο που υιοθετήθηκε κατά το ξεκίνημα της ανάπτυξης του SRS (48).

Όταν οι απαιτήσεις καταγράφονται σύμφωνα με το πρότυπο IEEE “IEEE Recommended Practice for Software Requirements Specifications” (IEEE Std 830-1993), τότε το SRS παρουσιάζει τις ακόλουθες 8 ιδιότητες(48):

1. *Ορθότητα (Correctness)*

   Η ορθότητα των απαιτήσεων, συνήθως, αναφέρεται στην ακρίβεια των απαιτήσεων και στο ενδεχόμενο, το σύστημα που ικανοποιεί αυτές τις

απαιτήσεις να ικανοποιεί τελικά και τις πραγματικές ανάγκες του πελάτη και όχι μόνο τις επιθυμίες του. Κάθε απαίτηση θα πρέπει να ελέγχεται όσον αφορά σε ακρίβεια και σε δυνατότητα να μετατρέψει, παράξει, ολοκληρώσει ή παρασχέσει ακριβώς ότι απαιτείται για μία λύση στο πρόβλημα του πελάτη και είναι εφικτή σε όρους δυνατότητας υλοποίησης της απαίτησης, εντός των χρονικών περιορισμών του έργου.

2. *Πληρότητα (Completeness)*

Για να απαντήσει κανείς στο ερώτημα: «Πότε ένα SRS θεωρείται πλήρες, τότε θα πρέπει να μπορεί να απαντήσει θετικά στις ακόλουθες δύο ερωτήσεις:

«Υπάρχουν συγκεντρωμένες όλες οι απαραίτητες απαιτήσεις για την περιγραφή της λύσης του προβλήματος του πελάτη;» και

«Το SRS εμπεριέχει όλα τα στοιχεία, τους πίνακες και τα διαγράμματα που είναι απαραίτητα για την επεξήγηση όλων των απαιτήσεων;»

3. *Συνοχή (Consistency)*

Η συνοχή αναφέρεται στα εξής: (1) λογική συνοχή των απαιτήσεων. Διερεύνηση του ενδεχομένου ύπαρξης αντικρουόμενων απαιτήσεων όσον αφορά σε ζητήματα όπως το τί θα πρέπει να ολοκληρώνει, μετατρέπει, παράγει ή παρέχει το σύστημα. Και (2) η ορολογία, τα διαγράμματα, οι πίνακες και τα σχήματα έχουν συνοχή μεταξύ τους, ώστε η προοπτική του προτεινόμενου συστήματος τελικά να έχει συνοχή.

4. *Επαληθευσιμότητα (Verifiability)*

Η επαληθευσιμότητα αναφέρεται στη δυνατότητα καταγραφής μίας διαδικασίας ώστε να ελέγχεται η ορθότητα κάθε απαίτησης κατά την υλοποίηση.

5. *Ιεράρχηση (Prioritization)*

   Κάθε απαίτηση θα πρέπει να ιεραρχείται ώστε να υπάρχει σαφής ένδειξη των απαιτήσεων που θα πρέπει να ικανοποιηθούν αρχικά και αυτών που δεν είναι τόσο σημαντικές.

6. *Ιχνηλασιμότητα (Traceability)*

   Η ιχνηλασιμότητα αναφέρεται στη δυνατότητα της εξιχνίασης υλοποιημένων μερών πίσω στις απαιτήσεις (forwards traceability) και σε εξιχνίαση απαιτήσεων πίσω στις πηγές τους (backwards traceability). Οι απαιτήσεις θα πρέπει να είναι αριθμημένες για τη διευκόλυνση της ιδιότητας της ιχνηλασιμότητας.

7. *Παραμετροποιησιμότητα (Modifiability)*

   Οι αλληλεξαρτήσεις μεταξύ απαιτήσεων θα πρέπει να διευκολύνουν τη δυνατότητα παραμετροποίησης των απαιτήσεων και τη δυνατότητα διαχείρισης των αλλαγών. Πρέπει να σημειωθεί ότι, τα εργαλεία όπως οι βάσεις δεδομένων των απαιτήσεων και τα εργαλεία διαχείρισης παραμετροποίησης, διευκολύνουν την εφαρμογή αλλαγών, όμως το κρίσιμο σημείο είναι η προσεκτική παρακολούθηση των αλλαγών των απαιτήσεων καθώς και η δομή του SRS, η οποία θα πρέπει να διευκολύνει τη διαχείριση των αλλαγών των απαιτήσεων.

8. *Σαφήνεια (Unambiguous)*

   Ιδανικά, οι απαιτήσεις θα πρέπει να ερμηνεύονται με έναν και μοναδικό τρόπο. Πολλαπλές ερμηνείες των απαιτήσεων μπορούν να οδηγήσουν σε υλοποίηση των απαιτήσεων με λάθος τρόπο. Οι απαιτήσεις θα πρέπει να καταγράφονται συνοπτικά και με σαφήνεια, όσο το επιτρέπει η γλώσσα που χρησιμοποιείται.

6.2.3 Οδηγίες Προδιαγραφής Απαιτήσεων

Όταν χρησιμοποιείται η φυσική γλώσσα για τη προδιαγραφή των απαιτήσεων είναι κρίσιμο να γίνεται προσπάθεια ώστε, οι δηλώσεις των απαιτήσεων να είναι ακριβείς και σαφείς. Το πρώτο βήμα είναι η επιβεβαίωση της αρίθμησης των απαιτήσεων, ώστε να διευκολύνεται η ιχνηλασιμότητα και η ιεράρχηση

των απαιτήσεων. Οι πιο σημαντικές απαιτήσεις θα πρέπει να έχουν προτεραιότητα.

Μερικές απλές οδηγίες όσον αφορά στον τρόπο καταγραφής απαιτήσεων ώστε να είναι σαφείς και κατανοητές, παρατίθενται παρακάτω (30):

1. Χρησιμοποιούνται πειστικές εκφράσεις όπως: «σίγουρα», «ως εκ τούτου», «προφανώς»; Αν χρησιμοποιούνται θα πρέπει να αναρωτηθεί κανείς «γιατί ;»
2. Χρησιμοποιούνται ασαφείς όροι, όπως: «μερικά», «συχνά», «συνήθως», «οι περισσότεροι». Αυτοί οι όροι θα πρέπει να ποσοτικοποιηθούν όπου είναι απαραίτητο και να επιβεβαιώνεται ότι, όλα τα αντικείμενα είναι επαρκώς κατανοητά.
3. Απαιτείται ιδιαίτερη προσοχή στις ανολοκλήρωτες λίστες, όπως στις λίστες που τελειώνουν σε: «ούτω καθ' εξής», «κλπ». Αυτές οι λίστες θα πρέπει να ολοκληρώνονται όπου είναι εφικτή, ενώ θα πρέπει να επιβεβαιώνεται ότι, όλα τα περιεχόμενα είναι πλήρως κατανοητά.
4. Θα πρέπει να επιβεβαιώνεται ότι, κάθε εύρος δεδομένων δεν συμπεριλαμβάνει κρυφές ενώσεις, όπως για παράδειγμα: «Οι είσοδοι μπορούν να πάρουν τιμές από 10 έως 100». Είναι όμως ακέραιοι αριθμοί ή πραγματικοί;
5. Απαιτείται προσοχή σε μη σαφή ρήματα όπως: «χειρίζεται», «απορρίπτεται», «επεξεργάζεται», «εξαλείφεται», «παραλείπεται». Αυτά τα ρήματα μπορούν να υπάρχουν σε πολλές περιπτώσεις, όμως θα πρέπει να αντικαθίστανται από πιο σαφείς εκφράσεις.
6. Ιδιαίτερη προσοχή απαιτείται στις αντωνυμίες, όπως: «Το τμήμα I/O επικοινωνεί με το τμήμα επαλήθευσης δεδομένων και ο έλεγχος τίθεται σε ισχύ» Ο έλεγχος ποιού τμήματος τίθεται σε ισχύ;
7. Εκφράσεις υπερβολής, όπως: «πάντα», «σε κάθε», «κανένα», «ποτέ», θα πρέπει να επανελέγχεται ώστε, να επιβεβαιώνεται η ορθότητα της χρήσης τους σε κάθε περίπτωση.
8. Όταν ένας όρος είναι ρητά καθορισμένος θα πρέπει να γίνεται προσπάθεια υποκατάστασης του σε όσα σημεία του ορισμού απαιτείται.
9. Όταν μία δομή ή ένα αντικείμενο περιγράφεται με λόγια, μία φωτογραφία κοντά του, ίσως να διευκολύνει στην κατανόησή του.

10. Όταν προσδιορίζεται ένας υπολογισμός θα πρέπει να παρατίθενται, τουλάχιστον, δύο σχετικά παραδείγματα.

Ιδιαίτερη προσοχή απαιτείται σε περιπτώσεις ασάφειας καταγεγραμμένων απαιτήσεων, είτε στον όρο είτε στη γλώσσα. Παρακάτω παρατίθενται μερικά παραδείγματα τέτοιων προβλημάτων σε όρους ή στη χρήση της γλώσσας (30):

- Το σύστημα πρέπει να ανταποκρίνεται στις διαταγές του χρήστη σε λογικό χρόνο
- Η διεπαφή χρήστη θα πρέπει να είναι διαισθητική
- Το σύστημα θα πρέπει να είναι σε θέση να μπορεί να ασχολείται με άλλους, πραγματικούς περιορισμούς
- Ο server θα είναι επαρκώς γρήγορος ώστε να μην αποτελεί τον οριακό παράγοντα στην ταχύτητα του συστήματος

Επιπλέον, σημαντικό ζήτημα είναι αυτό των απαιτήσεων που δεν έχουν οριστεί ορθά. Μερικά παραδείγματα παρουσιάζονται ακολούθως:

Το σύστημα θα ξεκινάει με:

1. Βαθμονόμηση (calibration) του υποσυστήματος Α
2. Βαθμονόμηση (calibration) του υποσυστήματος Β

Υποθέτοντας ότι γνωρίζουμε ποιά είναι τα υποσυστήματα Α και Β (από το SRS) δεν γνωρίζουμε πως προσδιορίζεται η διαδικασία της βαθμονόμησης του συστήματος. Προκειμένου να επιβεβαιώσουμε ότι αυτή η απαίτηση είναι πλήρης και σαφής θα πρέπει να προσδιορίσουμε τη διαδικασία «βαθμονόμηση» για κάθε ένα από τα δύο υποσυστήματα.

Ένα άλλο παράδειγμα αποτυχημένου ορισμού - διατύπωση απαίτησης είναι το εξής:

«Ο χρήστης θα έχει τη δυνατότητα να υποβάλει ερωτήματα στη βάση δεδομένων». Η συγκεκριμένη απαίτηση είναι πολύ γενική και ασαφής. Πέραν τούτου, θα μπορούσε να ικανοποιηθεί με πάρα πολλούς τρόπους (49).

Ένας καλύτερος τρόπος διατύπωσης της ίδιας απαίτησης θα ήταν «Το σύστημα θα περιλαμβάνει ειδική φόρμα, όπου ο χρήστης θα δηλώνει τη πόλη, τις

ημερομηνίες κράτησης (από - έως), το είδος δωματίου και θα του εμφανίζονται τα διαθέσιμα καταλύματα προς κράτηση και η αντίστοιχη τιμή κράτησης»

6.2.4 Τυπικές μέθοδοι Προδιαγραφής Απαιτήσεων

Ένας άλλος τρόπος προδιαγραφής απαιτήσεων είναι η χρήση μαθηματικής σημειογραφίας. Αυτές οι σημειογραφίες καλούνται Τυπικές Μέθοδοι (formal methods) και τυπικά, παρέχουν περισσότερες από μία σημειογραφία για την καταγραφή απαιτήσεων. Συνήθως, συμπεριλαμβάνουν εργαλεία ανάλυσης παράλληλων-ταυτόχρονων συστημάτων και της συμπεριφοράς αυτών. Μία μικρή λίστα μερικών από τις πιο δημοφιλείς μεθόδους δίνεται στο Σχήμα 6.6:

Model Based	Z, VDM, B
Axiomatic	ACT 1, ACT 2, Extended ML
Process Algebra	CCS, CSP, LOTOS, ESTELLE
Concurrensy	Petri Nets

Σχήμα 6.6: Λίστα βασικών τυπικών μεθόδων.

Μία μαθηματική σημειογραφία παρέχει ακριβή γλώσσα χωρίς τους ασαφείς όρους της φυσικής γλώσσας. Ειδικότερα, υπάρχουν τρείς κατηγορίες τυπικών μεθόδων, που ομαδοποιούνται σε “model based”, “axiomatic”, “process algebra”. Για συγχρονισμό ή χρήση πόρων, η μέθοδος “petri nets” χρησιμοποιείται εδώ και πολλά χρόνια από μηχανικούς (50).

6.3 Επικύρωση Απαιτήσεων

Όλα τα έργα μηχανικής λογισμικού θα πρέπει να παρέχουν υψηλού επιπέδου ασφάλεια –υψηλό βαθμό εμπιστοσύνης- σχετικά με το γεγονός ότι το τελικό σύστημα θα εξυπηρετεί την ανάγκη για την οποία αναπτύχθηκε. Το επίπεδο ασφαλείας εξαρτάται από πολλούς παράγοντες. Σε αυτούς συμπεριλαμβάνονται, ο τύπος του συστήματος που αναπτύσσεται, η διαδικασία που χρησιμοποιείται συνολικά, οι δραστηριότητες επαλήθευσης και επικύρωσης που χρησιμοποιούνται και η διάρκεια ανάπτυξης. Τα είδη δραστηριοτήτων άσκησης ελέγχου και παροχής ασφαλείας σε τέτοια έργα μπορούν να ομαδοποιηθούν σε δύο κατηγορίες (51):

Τομεο-Κατευθυνόμενες (Domain directed)

Αυτές οι δραστηριότητες συμπεριλαμβάνουν τον πελάτη, τους χρήστες, τους ειδικούς ή άλλα πρόσωπα που αποτελούν πηγές απαιτήσεων ή πηγές απόδοσης

λύσεων σε ζητήματα που αφορούν το μοντέλο απαιτήσεων. Σε αυτή τη κατηγορία ανήκουν και εμπειρικές δραστηριότητες που χρησιμοποιούν αποτελεσματικά το προτεινόμενο λειτουργικό περιβάλλον ως «μάντες».

Κατευθυνόμενες από τις προδιαγραφές (Specification directed)

Συμπεριλαμβάνουν την εκμαίευση πληροφοριών από περιπτώσεις χρήσης, σενάρια και εμπειρικά δεδομένα, δόμηση των απαιτήσεων και ανάλυση απαιτήσεων για την εκχύλιση περισσότερης πληροφορίας. Για παράδειγμα, η επιβεβαίωση ότι, συγκεκριμένες, επιθυμητές ιδιότητες επιτυγχάνονται. Πρέπει να σημειωθεί πάντως ότι, οι απαιτήσεις που προϋποτίθενται, όπως συγκεκριμένοι στόχοι επίδοσης, είναι μέρος των πραγματικών απαιτήσεων.

Ο κύριος στόχος της επικύρωσης του SRS είναι η επιβεβαίωση των πελατών και των εμπλεκόμενων μερών ότι, το SRS είναι αποτελεσματικό κατά τον τρόπο με τον οποίο έχει καταγραφεί. Πρέπει να σημειωθεί ότι, το πλαίσιο ασφαλείας δεν παρέχει αλάνθαστες εγγυήσεις. Κάτι τέτοιο είναι αδύνατο αφού, συνήθως, ο χρόνος είναι περιορισμένος και η πολυπλοκότητα του προβλήματος είναι τέτοια ώστε, θα έπρεπε να περάσουν πολλά έτη προκειμένου να προσδιοριστούν όλες οι πιθανότητες. Σε κάθε περίπτωση θα πρέπει να παρέχεται ένα τέτοιο επίπεδο ασφαλείας ώστε τουλάχιστον οι απαιτήσεις να είναι έγκυρες, ακριβείς, κατανοητές, σαφείς, πλήρεις, επαληθεύσιμες, κλπ

6.3.1 Αναφορές και Επιθεωρήσεις

Η αναφορά προδιαγραφής απαιτήσεων είναι η αξιολόγηση του SRS για την εξακρίβωση της ολοκλήρωσης του, της συνοχής του, της ορθότητάς του και του συγχρονισμού του με τις οκτώ ιδιότητες ενός καλού SRS, στις οποίες έγινε αναφορά παραπάνω. Οι αναφορές διεξάγονται σύμφωνα με μία τυπική διαδικασία συνάντησης η οποία διαρκεί κάποια λεπτά.

Οι άτυπες αναφορές αποτελούν συναντήσεις και έχουν σκοπό να παρέχουν αποδείξεις όσον αφορά στα εξής (52):

1.Το SRS ικανοποιεί τα κριτήρια ορθότητας, πληρότητας και συνοχής, όπως επίσης και τις υπόλοιπες ιδιότητες ενός αποτελεσματικά ανεπτυγμένου SRS.

2. Το SRS ακολουθεί τις οδηγίες και τα πρότυπα που απαιτούνται από τον οργανισμό ή το συγκεκριμένο έργο.

3. Οι αλλαγές στο SRS πραγματοποιούνται κατάλληλα και αποτελεσματικά, επιδρώντας μόνο σε εκείνες τις περιοχές που προσδιορίζονται στην προδιαγραφή αλλαγών.

Οι αναφορές είναι κατάλληλες για την επικύρωση της προδιαγραφής απαιτήσεων.

Μία πιο δομημένη μορφή επικύρωσης είναι οι επιθεωρήσεις (inspections). Οι επιθεωρήσεις επικεντρώνονται στις ακόλουθες πτυχές του SRS(52):

1. Έλεγχος για την ικανοποίηση των κριτηρίων ορθότητας, πληρότητας, συνοχής, καθώς και των υπόλοιπων ιδιοτήτων ενός SRS. Έλεγχος για την ικανοποίηση των προδιαγραφών των προϊόντων εργασίας.
2. Έλεγχος συμμόρφωσης του SRS με πρότυπα.
3. Προσδιορισμός αποκλίσεων από τα πρότυπα και των στόχων ενός καλού SRS (λίστα στόχων).
4. Συλλογή δεδομένων. Για παράδειγμα, «πόσες αποκλίσεις βρέθηκαν;»
5. Οι επιθεωρήσεις δεν γνωρίζουν, όμως εξετάζουν εναλλακτικές ή ζητήματα εμφάνισης ή βελτίωσης του SRS.

Η ανάπτυξη των σωστών απαιτήσεων για το λογισμικό μπορεί να ελεγχθεί και με τη βοήθεια των walkthrough. Το walkthrough αποτελεί μια μορφή αξιολόγησης κατά την οποία ο σχεδιαστής ή προγραμματιστής του λογισμικού παρουσιάζει βήμα – βήμα το λογισμικό που έχει αναπτυχθεί σε όλους τους ενδιαφερόμενους και χρήστες του συστήματος και οι συμμετέχοντες κάνουν ερωτήσεις, παρατηρήσεις σχετικά με πιθανά σφάλματα, προβλήματα και παραβιάσεις προτύπων ανάπτυξης (53).

Ένα walkthrough μπορεί να λάβει χώρα σε όλα τα στάδια ανάπτυξης ενός λογισμικού και έχει ως σκοπό της αξιολόγηση της απόδοσης του νέου λογισμικού και την επίτευξη των αρχικών στόχων από την ανάπτυξη του λογισμικού. Με τη βοήθειά του μπορεί ο Μηχανικός Απαιτήσεων να σιγουρευτεί πως οι σωστές απαιτήσεις λογισμικού έχουν προσδιοριστεί. Περιγράφεται το τελικό λογισμικό στους χρήστες του συστήματος και ζητείται η γνώμη τους επί του συστήματος (54).

Τα walkthrough συνήθως χρησιμοποιούνται όταν ο μηχανικός απαιτήσεων επιθυμεί να λάβει υπόψη την γνώμη των χρηστών, οι οποίοι όμως δεν έχουν τις

τεχνικές γνώσεις για να κατανοήσουν ένα έγγραφο περιγραφής των προδιαγραφών του λογισμικού. Σε ένα walkthrough εντοπίζονται πιθανά λάθη, αλλά δε διορθώνονται στα πλαίσια του walkthrough (55).

Για την επιτυχή έκβαση ενός walkthrough συνίσταται για το άτομο που έχει την ευθύνη παρουσίασης να είναι σίγουρος πως (55):

- στη συνάντηση συμμετέχουν όλοι όσοι θα πρέπει να επανεξετάσουν το λογισμικό (χρήστες, διευθυντές, προγραμματιστές κα)
- όλοι οι συμμετέχοντες γνωρίζουν το λόγο που το walkthrough λαμβάνει χώρα
- περιγράφεται με λεπτομέρεια κάθε στοιχείο που περιλαμβάνετε στην παρουσίαση του τελικού λογισμικού
- υπάρχει καθοδήγηση συζήτησης προς τον εντοπισμό ελλείψεων στο πρόγραμμα
- καταγράφονται όλα τα ζητήματα που θέτουν οι συμμετέχοντες

6.3.2 Μοντελοποίηση και Προτυποποίηση

Προηγουμένως έγινε αναφορά σε έναν από τους τρόπους επικύρωσης των απαιτήσεων με τη βοήθεια της τεχνικής μοντελοποίησης σε γλώσσα UML. Δομώντας σενάρια, περιπτώσεις χρήσης και διαγράμματα αλληλεπίδρασης, από τη σκοπιά του πελάτη, είναι συνήθως κρίσιμο για τη κατανόηση του προβλήματος. Ως συνέπεια της δόμησης του μοντέλου, άτυπα, επιτυγχάνεται και η επικύρωση των απαιτήσεων.

Ένας άλλος μηχανισμός για την επικύρωση της συνοχής, της πληρότητας και της ορθότητας των απαιτήσεων είναι αυτός της δόμησης ταχέων προτύπων (rapid prototypes). Τα ταχεία πρότυπα αποτελούν εκτελέσιμα μοντέλα των απαιτήσεων, αντίθετα με τα διαγράμματα της γλώσσας UML , τα οποία πρέπει να ενσωματώνουν δυναμική συμπεριφορά σε (στατικά) διαγράμματα. Η αναμενόμενη συμπεριφορά του συστήματος μπορεί να παρουσιαστεί στον πελάτη με την χρήση πρωτοτύπου ώστε, να αξιολογηθεί και να αλλαχθεί το συντομότερο δυνατό, εφόσον απαιτείται.

Ακολουθεί παράδειγμα πρωτοτυποποίησης:

«Η κατασκευή και ανάλυση ενός εκτελέσιμου μοντέλου που προσεγγίζει ένα προτεινόμενο σύστημα»

Η πρωτοτυποποίηση αντιμετωπίζεται, συνήθως, ως στρατηγική ελαχιστοποίησης κινδύνων, παρόλο που διευκολύνει πολύ τη κατανόηση, βελτίωση και επικύρωση των απαιτήσεων. Το κύριο πρόβλημα με τη πρωτοτυποποίηση είναι ότι, ο πελάτης μπορεί να επιθυμεί το αποτέλεσμα της πρωτοτυποποίησης και όχι τη τελική έκδοση του προϊόντος που αναπτύχθηκε μέσα από το έργο. Αυτό είναι αρκετά ριψοκίνδυνο.

Αν τα όρια μιας πρωτοτυποποίησης είναι κατανοητά, τότε η πρωτοτυποποίηση κατά τη φάση των απαιτήσεων μπορεί να επιτύχει τους εξής στόχους:

1. εκμαίευση απαιτήσεων
2. ανάλυση απαιτήσεων και δόμηση μοντέλου του προτεινόμενου συστήματος
3. επικύρωση απαιτήσεων

Τυπικά, αυτό το είδος προτυποποίησης αναφέρεται ως, «Διερευνητική Προτυποποίηση» (Exploratory Prototyping).

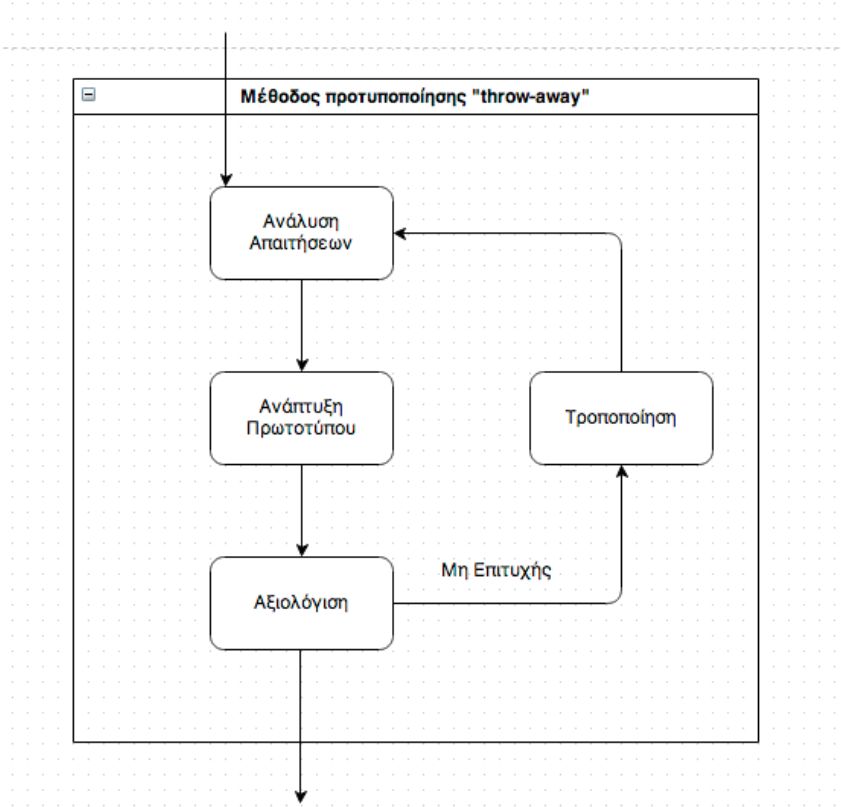
Με αυτόν το τρόπο και συμπεριλαμβάνοντας στη διαδικασία τεχνικές συνεντεύξεων, ερωτηματολογίων και ομάδων συγκέντρωσης, επιτυγχάνεται ανάδραση σχετικά με την καταλληλότητα και αποτελεσματικότητα του μοντέλου (56).

6.3.3 Προοπτικές Προτυποποίησης

Οι κυριότερες προοπτικές προτυποποίησης παρουσιάζονται παρακάτω(57):

Αναλώσιμη (Throw-Away)

Στην αναλώσιμη πρωτοτυποποίηση, τα πρωτότυπα κατασκευάζονται για το σκοπό της εξόρυξης, κατανόησης και επικύρωσης των απαιτήσεων. Μία διαδικασία βελτίωσης των απαιτήσεων με τη συγκεκριμένη προοπτική παρουσιάζεται στο Σχήμα 6.7



Σχήμα 6.7: Η μέθοδος πρωτοτυποποίησης throw-away

Τα πλεονεκτήματα και τα μειονεκτήματα της συγκεκριμένης προοπτικής παρουσιάζονται παρακάτω:

Πλεονεκτήματα:

- Το πρωτότυπο μπορεί να «τρέχει» προκειμένου να αποκαλύψει λανθασμένη συμπεριφορά
- Το πρωτότυπο μπορεί να χρησιμοποιείται ως μέσο σύγκλισης της επικοινωνίας των πελατών με τους προγραμματιστές, ώστε να μειώνεται ο κίνδυνος ενδεχόμενων παρανοήσεων.
- Διευκολύνεται η ανίχνευση απαιτήσεων που βρίσκονται στην αφάνεια.
- Διευκολύνεται η ανίχνευση πολύπλοκων ή δύσκολων υπηρεσιών, ιδιαίτερα εφόσον οι τελικοί χρήστες εμπλέκονται στη πρωτοτυποποίηση. Πρέπει να σημειωθεί εδώ ότι, τα πρωτότυπα δεν κατασκευάζονται πάντα με γνώμονα τη χρηστικότητα.
- Διευκολύνεται η ανάδειξη της σκοπιμότητας των απαιτήσεων.

Μειονεκτήματα:

- Σε περιπτώσεις λαθών σε ένα πρωτότυπο, είναι αναγκαίο να προσδιορίζεται, αν τα λάθη προκύπτουν από το πρωτότυπο ή από τις απαιτήσεις. Είναι ένα ζήτημα, επομένως, αν το πρωτότυπο ανταποκρίνεται πλήρως στις απαιτήσεις.
- Η διαχείριση της διαδικασίας πρωτοτυποποίησης μπορεί να είναι ιδιαίτερα χρονοβόρα.
- Από τη μεριά των πελατών, πολλές φορές επιθυμούν να κρατήσουν το πρωτότυπο (αντί για την τελική έκδοση) , θεωρώντας ότι τους αρκεί.
  Σημείωση: Πρέπει να γίνει σαφές ότι, το πρωτότυπο δεν ανταποκρίνεται στην τελική μορφή του λογισμικού. Υπάρχουν σημαντικές διαφορές μεταξύ τους. Πιο συγκεκριμένα, το πρωτότυπο, συνήθως, δεν περιλαμβάνει όλες τις λειτουργίες που περιλαμβάνει η τελική έκδοση. Τα πρωτότυπα, εξαιτίας της ταχύτητας με την οποία αναπτύσσονται, δεν είναι διατηρήσιμα. Τέλος, τα πρωτότυπα της αναλώσιμης προοπτικής δημιουργούνται για την εξόρυξη απαιτήσεων, με τη χρήση εργαλείων πρωτοτυποποίησης. Δεν περιλαμβάνουν ιδιότητες, όπως επίδοση.

Εξελικτική (Evolutionary)

Κατά την εξελικτική προοπτική η διαδικασία προτυποποίησης ξεκινά με τη συλλογή απαιτήσεων, αναπτύσσεται ένα πρωτότυπο και ύστερα επαναλαμβάνεται η διαδικασία έως ότου το πρωτότυπο να πάρει την τελική μορφή, αυτή δηλαδή της τελικής έκδοσης. Παρακάτω παρουσιάζονται τα προβλήματα αυτής της προοπτικής :

- Τυπικά, η εξελικτική προτυποποίηση είναι τόσο ταχεία που δεν επιτρέπει την εφαρμογή της λογικής στον σχεδιασμό. Συνεπώς, το πρωτότυπο γίνεται, πολύ σύντομα, μη διατηρήσιμο.
- Είναι πολύ δύσκολο να υιοθετούνται πρότυπα κατά τη διαδικασία της εξελικτικής προτυποποίησης.
- Ο προγραμματισμός του έργου γίνεται αδύνατος όταν δεν υπάρχει προγραμματισμός των παραδοτέων του έργου ή δεν εφαρμόζεται πειθαρχία.
- Η πρωτοτυποποίηση, η αξιολόγηση και η βελτίωση των ακολουθιών μπορούν να γίνουν ατέρμονες διαδικασίες.

- Υπάρχει πάντα ένας κίνδυνος, το πρωτότυπο που αξιολογείται να είναι ξεπερασμένο. Το γεγονός αυτό θα κάνει την αξιολόγηση να είναι ανακριβής ή χωρίς νόημα.
- Η κατάλληλη και ακριβής τεκμηρίωση είναι μία διαδικασία χρονοβόρα και δύσκολη. Επομένως, είναι πολύ δύσκολο να εκτελεστεί σε σύντομο χρονικό διάστημα.

Αυξητική (Incremental)

Η αυξητική πρωτοτυποποίηση συνδυάζει την εξελικτική με την πειθαρχία της παραδοσιακής ανάπτυξης λογισμικού. Βασίζεται στη δημιουργία απαιτήσεων και την παράδοση του συστήματος με αυξητικό τρόπο. Παρακάτω παρουσιάζονται τα πλεονεκτήματα και τα μειονεκτήματα της συγκεκριμένης προοπτικής:

Πλεονεκτήματα:

- Διατηρεί την πειθαρχία του προγραμματισμού και της διαχείρισης κάθε προσαύξησης.
- Η τεκμηρίωση μπορεί να υιοθετεί πρότυπα.
- Επιτρέπει την ανάδραση των χρηστών.

Μειονεκτήματα:

- Η ολοκλήρωση μπορεί να είναι πολύ δύσκολη διαδικασία.
- Ο σχεδιασμός θα πρέπει να ολοκληρώνεται κατά τα αρχικά στάδια.
- Βασίζεται στην απομόνωση ενός πολύ καλά κατανοητού και βασικού συνόλου απαιτήσεων.

7 Διαχείριση Απαιτήσεων

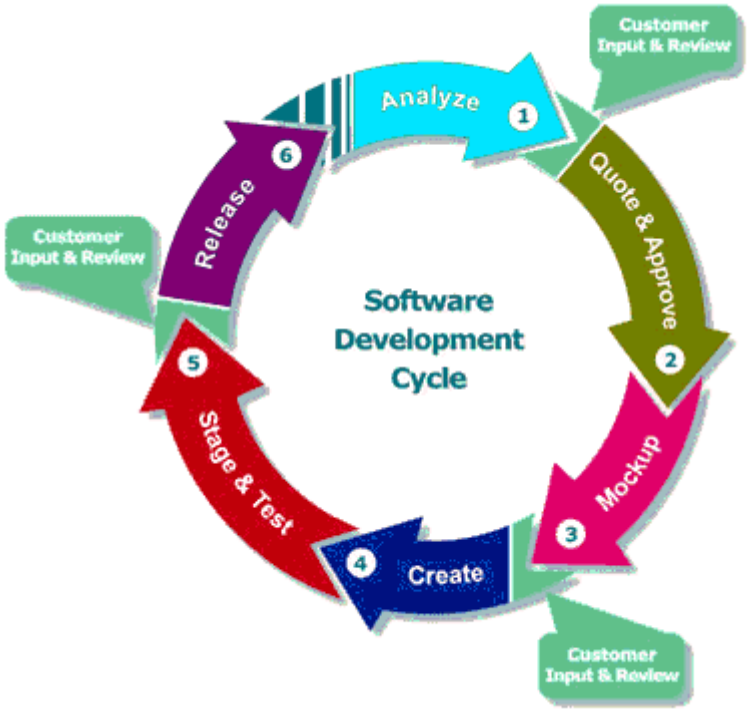
7.1 Εισαγωγή

Η βιομηχανία λογισμικού μπορεί να είναι περήφανη για αρκετά επιτεύγματα. Η παγκόσμια αγορά που αφορά σε υπηρεσίες παροχής λύσεων λογισμικού είναι μία από τις μεγαλύτερες, τουλάχιστον, στις μέρες μας. Τεχνολογικά, η βιομηχανία έχει αναπτύξει πολύ αποτελεσματικά συστήματα που έχουν βοηθήσει σημαντικά στην αύξηση της παραγωγικότητας των επιχειρήσεων. Επιγραμματικά, αναφέρονται μερικά πολύ σημαντικά παράγωγα της βιομηχανίας λογισμικού: διεπαφές χρηστών, σχεσιακές βάσεις δεδομένων, αντικειμενοστραφής προγραμματισμός, γλώσσες προγραμματισμού, διαδίκτυο, κλπ (58).

Καθώς διαδραματίζεται η πρόοδος, το πεδίο της Ανάπτυξης Λογισμικού θεωρείται περισσότερο τέχνη παρά επιστήμη. Η Ανάπτυξη Λογισμικού, ως τέχνη, έχει συγκεκριμένη αισθητική και συναισθηματική απήχηση, όμως δεν θεωρείται αξιόπιστη διαδικασία. Όπως και στις τέχνες, έτσι και στην Ανάπτυξη Λογισμικού παρατηρούνται εκρήξεις ατομικής, κυρίως, δημιουργικότητας και μηχανικής πειθαρχίας. Επιπλέον, πρέπει να σημειωθεί ότι, το λογισμικό χρησιμοποιείται για την επίλυση προβλημάτων, για τα οποία, μερικά χρόνια πριν, δεν υπήρχε υπόνοια ότι θα μπορούσαν να επιλυθούν με τη βοήθειά του (58).

Σε περιπτώσεις ανάπτυξης λογισμικού για μεγάλα και πολύπλοκα συστήματα, οι απαιτήσεις μεταβάλλονται κατά τη διάρκεια του έργου. Ένας λόγος για τον οποίο συμβαίνει αυτό, είναι ότι αυτά τα συστήματα αναπτύσσονται ώστε να επιλύσουν πολύπλοκα προβλήματα που, όμως, δεν μπορούν να οριστούν με ακρίβεια, ειδικά κατά τα πρώτα στάδια του έργου. Αν όμως το πρόβλημα δεν έχει οριστεί επαρκώς, επόμενα, οι απαιτήσεις που θα προσδιοριστούν, είναι προφανές ότι δεν θα μπορέσουν να το επιλύσουν. Κατά τη διάρκεια της Ανάπτυξης του Λογισμικού, η κατανόηση του προβλήματος εξελίσσεται και οι απαιτήσεις θα πρέπει να επηρεάζονται, ώστε να συνάδουν με τα δεδομένα και την προοπτική του προβλήματος (59).

Στο Σχήμα 7.1 απεικονίζονται τα σημεία του κύκλου ζωής ενός λογισμικού, όπου υπάρχει ανατροφοδότηση από τον πελάτη και πιθανή αναθεώρηση των απαιτήσεων του λογισμικού:



Σχήμα 7.1: Ανατροφοδότηση απαιτήσεων στον κύκλο ζωής ενός λογισμικού

Σταθερές Απαιτήσεις (Enduring Requirements): Πρόκειται για απαιτήσεις που μεταβάλλονται με μεγάλη δυσκολία. Είναι σχεδόν αμετάβλητες, καθώς προκύπτουν από τις βασικές δραστηριότητες ενός οργανισμού ή μίας επιχείρησης (59).

Ασταθείς Απαιτήσεις (Volatile Requirements): Πρόκειται για απαιτήσεις που μπορούν να μεταβάλλονται κατά τη διάρκεια του έργου ή και αφού το σύστημα αρχίσει να λειτουργεί. Αυτές οι απαιτήσεις δεν προβάλλουν ισχυρή αντίσταση στη μεταβολή τους, καθώς προκύπτουν από υποστηρικτικές δραστηριότητες που αφορούν στον τρόπο με τον οποίο διεξάγονται οι εργασίες εντός του οργανισμού ή της επιχείρησης (59).

Από τη στιγμή που ένα σύστημα εγκαθίσταται και αρχίσει να χρησιμοποιείται, νέες απαιτήσεις αναδύονται συνεχώς. Είναι δύσκολο για τους χρήστες και τους πελάτες του συστήματος να γνωρίζουν την επίδραση του νέου συστήματος όσον αφορά στον τρόπο που διεξάγονται οι διεργασίες. Όταν οι τελικοί χρήστες αποκτούν εμπειρία στη χρήση του συστήματος, τότε ανακαλύπτουν νέες ανάγκες και προτεραιότητες. Υπάρχουν πολλοί λόγοι που κάνουν τις αλλαγές αναπόφευκτες (59):

1. Το επιχειρηματικό και τεχνολογικό περιβάλλον του συστήματος μεταβάλλονται πάντα, ακόμη και μετά την εγκατάστασή και εκκίνηση

της λειτουργίας του. Η εγκατάσταση νέας υλικοτεχνικής υποδομής, η ολοκλήρωση του συστήματος και η επικοινωνία με άλλα συστήματα, η διαφοροποίηση στις επιχειρηματικές ιεραρχίες, είναι μερικές από τις αλλαγές που μπορούν να μεταβάλλουν το περιβάλλον. Το σύστημα θα πρέπει, σε κάθε περίπτωση, να συμμορφώνεται και να ανταποκρίνεται επαρκώς και αποτελεσματικά σε αυτές τις αλλαγές.

2. Οι άνθρωποι που επενδύουν στην ανάπτυξη του συστήματος και οι χρήστες του, συνήθως, δεν είναι οι ίδιοι άνθρωποι. Οι πελάτες του συστήματος επιβάλλουν κάποιες απαιτήσεις, εξαιτίας οργανωτικών και οικονομικών περιορισμών. Αυτές οι απαιτήσεις μπορεί να αντικρούουν αυτές των τελικών χρηστών και μετά την παράδοση του έργου να είναι απαραίτητη η εγκατάσταση νέων λειτουργιών για την υποστήριξη των χρηστών, προκειμένου το σύστημα να επιτελεί αποτελεσματικά τους στόχους του.
3. Τα μεγάλα συστήματα έχουν, συνήθως, ποικίλες κοινότητες χρηστών, με πολλούς χρήστες να εκθέτουν διαφορετικές απαιτήσεις και προτεραιότητες, οι οποίες μπορεί να είναι αντικρουόμενες μεταξύ τους ή αντιφατικές. Οι τελικές απαιτήσεις του συστήματος συγκεντρώνονται μετά από συμβιβασμό των παραπάνω.

Η Διαχείριση Απαιτήσεων είναι η διαδικασία της κατανόησης και του ελέγχου των αλλαγών, στις απαιτήσεις του συστήματος. Θα πρέπει να παρακολουθούνται οι απαιτήσεις ξεχωριστά και σε συνδυασμό, ώστε να μπορεί να αξιολογείται ο βαθμός επίδρασης της κάθε αλλαγής στη λειτουργία του συστήματος. Θα πρέπει να καθιερώνεται μία επίσημη διαδικασία για τη δημιουργία προτάσεων αλλαγών και για τη σύνδεση αυτών με τις απαιτήσεις του συστήματος. Η τυπική μέθοδος της διαχείρισης απαιτήσεων θα πρέπει να ξεκινά, όταν το προσχέδιο των απαιτήσεων είναι διαθέσιμο. Ωστόσο, θα πρέπει να ξεκινά και ο προγραμματισμός της διαχείρισης των αλλαγών των απαιτήσεων, παράλληλα με τη διαδικασία εξόρυξής τους (59).

7.2 Απαιτήσεις και Συναισθήματα

Σύμφωνα με τον Damasio (60), διακρίνονται τρείς, σχετικές με τις απαιτήσεις, κατηγορίες συναισθημάτων:

- Πρωτογενή Συναισθήματα (Primary Emotions)
- Συναισθήματα Ιστορικού (Background Emotions)

- Κοινωνικά Συναισθήματα (Social Emotions)

Τα πρωτογενή συναισθήματα συμπεριλαμβάνουν συναισθήματα χαράς, λύπης, φόβου, θυμού, έκπληξης και απέχθειας. Τα συναισθήματα ιστορικού περιλαμβάνουν τις αισθήσεις της ευημερίας και της κακουχίας, της γαλήνης και της έντασης, του πόνου και της ευχαρίστησης, του ενθουσιασμού και της κατάθλιψης. Τα κοινωνικά συναισθήματα περιλαμβάνουν τη ντροπή, τη ζήλεια, την ενοχή και την υπερηφάνεια (61).

Τρείς πτυχές συναισθημάτων καθιστούν δύσκολη την αντιμετώπιση ζητημάτων απαιτήσεων που προκύπτουν σε περιπτώσεις ανάπτυξης νέου λογισμικού (61):

- Ακόμη κι αν οι διαδικασίες εκπαίδευσης είναι υπεύθυνες για τις διαφορετικές εκφράσεις των συναισθημάτων και τη διαφορετικότητα όσον αφορά στην κατανόησή τους, τα συναισθήματα έχουν καθοριστεί βιολογικά. Είναι εγκεφαλικά συμπτώματα και προκαλούνται από έμφυτες συσκευές του ανθρώπινου εγκεφάλου.
- Οι εγκεφαλικές συσκευές, από τις οποίες εξαρτώνται τα συναισθήματα, μπορούν να ενεργοποιηθούν χωρίς την επίγνωση ή τη θέληση του ατόμου στο οποίο προκαλούν συναισθήματα.
- Για έναν ενήλικο, λίγα αντικείμενα και γεγονότα είναι, συναισθηματικά, ουδέτερα.

Παρακάτω αναφέρονται μερικά ενδεικτικά παραδείγματα όπου οι απαιτήσεις επηρεάζονται από τα συναισθήματα, τις αξίες και τις πεποιθήσεις των εμπλεκόμενων μερών (61):

1. Ένα σύστημα το οποίο αναπτύχθηκε για να αποθηκεύει πληροφορίες και δεδομένα σχετικά με αστοχίες και λάθη, καθώς και τις ευθύνες αυτών, δημιούργησε τόσο άγχος στους ενδεχόμενους χρήστες, σε σημείο που, τελικά, οι λειτουργίες που αφορούσαν στα λάθη αφαιρέθηκαν.

2. Για την αποτελεσματικότερη λειτουργία της βιβλιοθήκης ενός Πανεπιστημίου αποφασίστηκε η εγκατάσταση ενός κεντρικού συστήματος λογισμικού. Είχε προγραμματιστεί ότι, όλα τα μέλη του ανθρώπινου δυναμικού της βιβλιοθήκης θα είχαν πρόσβαση σε όλες τις πληροφορίες του συστήματος, με στόχους την καλλιέργεια της αυτονομίας, της δημιουργικότητας και τη

διευκόλυνση των διαδικασιών λήψης αποφάσεων. Παρόλα αυτά, το αποτέλεσμα δεν ήταν το επιθυμητό. Κάποιοι από τους χρήστες προτιμούσαν να μην έχουν πρόσβαση σε όλες τις πληροφορίες σχετικές με τις διεργασίες που τους αφορούσαν. Δεν τους ενδιέφερε να αποκτήσουν περισσότερες ευθύνες, ούτε αυτονομία. Ένιωθαν άνετα με τις εργασίες που αναλάμβαναν μέχρι τότε και οι οποίες τους επέτρεπαν να πληρώνονται χωρίς να καταβάλουν μεγάλη προσπάθεια στη κατεύθυνση της δημιουργικότητας. Όπως γίνεται αντιληπτό, οι χρήστες προέβαλαν αντίσταση στις νέες πρακτικές που θα υιοθετούνταν για την υποστήριξη του συστήματος.

3. Τα διεθνή πρότυπα που δημιουργούν κουλτούρα ανεξαρτησίας στη διαχείριση αποφάσεων και ελέγχουν τη δημιουργικότητα. Αυτή η σκέψη είναι αρκετά διαδεδομένη και χρησιμοποιείται ως επιχείρημα για την αποφυγή επιβολής προτύπων. Η σκέψη είναι η εξής: «Αν πρέπει να ακολουθήσουμε τα διεθνή πρότυπα και τις καλύτερες πρακτικές, γιατί πρέπει να μετράμε τη δημιουργικότητα;»

Μία εταιρία αποφάσισε να εγκαταστήσει ένα σύστημα ERP, προκειμένου να ικανοποιηθούν οι απαιτήσεις της. Η εταιρία προσπαθούσε να δώσει κίνητρα στο εργατικό δυναμικό, ώστε να το χρησιμοποιήσουν όσο το δυνατό περισσότερο. Ο χαρισματικός ηγέτης του τμήματος διαχείρισης πόρων θεωρούσε ότι η αυξημένη χρήση του συστήματος θα επέφερε τη μείωση της επιρροής του τμήματος το οποίο διοικούσε, προς την εταιρία συνολικά. Ως αποτέλεσμα, θεώρησε ότι κάτι τέτοιο θα σήμαινε την αναστολή της προώθησης του ως διευθυντή. Έτσι, ξεκίνησε να σαμποτάρει, ενεργά, τη χρήση του νέου συστήματος. Οι προσπάθειές του ήταν τόσο επιτυχημένες ώστε, η εγκατάσταση του συστήματος αναβλήθηκε. Κατάφερε να πείσει την εταιρία ότι το σύστημα που ήθελαν να εγκαταστήσουν δεν ήταν κατάλληλο, ενώ ταυτόχρονα ξεκίνησε η ανάπτυξη ενός άλλου συστήματος (in-house), με τους ίδιους στόχους και παρόμοιες λειτουργίες. Το πλεονέκτημα της in-house ανάπτυξης ήταν το γεγονός ότι θα δινόταν η ευκαιρία στον διοικητή του τμήματος προσωπικού και στην ομάδα που διοικούσε να εντρυφήσουν, σταδιακά, στο σύστημα. Με άλλα λόγια, το αρχικό σύστημα δεν εγκαταστάθηκε ποτέ, ενώ το σύστημα που αναπτύχθηκε, in-house, από το προσωπικό της εταιρίας, έδωσε την ευκαιρία στον διαχειριστή του ανθρώπινου δυναμικού, να υπερνικήσει τις προθέσεις της διοίκησης της εταιρίας. Το ειρωνικό σημείο της ιστορίας είναι η πρόταση που

κατέθεσε ο διαχειριστής προσωπικού, αφού πήρε προαγωγή, για την εγκατάσταση ενός νέου ERP συστήματος ώστε να μπορεί να ελέγχει πιο αποτελεσματικά την ομάδα του και τις επιδόσεις της.

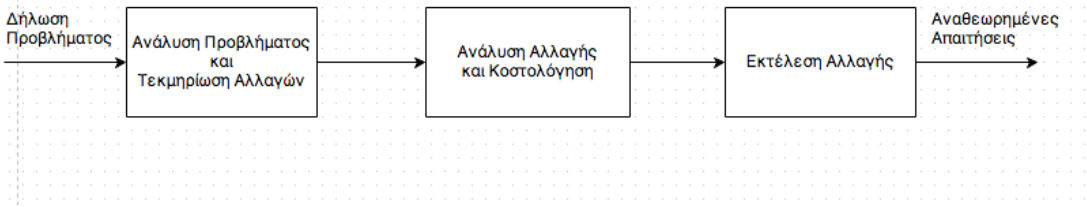
Ο Μηχανικός Απαιτήσεων, όπως διαφαίνεται από τα παραπάνω, έχει μία νέα αποστολή, εκτός από την ανάδειξη των λειτουργικών και μη-λειτουργικών απαιτήσεων ενός έργου. Θα πρέπει να αναδεικνύει, επίσης, συναισθήματα, αξίες και πεποιθήσεις που επηρεάζονται από το έργο-σύστημα, ενώ είναι σημαντικό να μπορεί να τις προσδιορίσει κατά τα αρχικά στάδια ανάπτυξης, ώστε να είναι δυνατή η αντιμετώπισή τους, όπως κάθε άλλος παράγοντας που επηρεάζει τις απαιτήσεις του συστήματος. Είναι απαραίτητο, λοιπόν, να εισάγουμε στις παραδοσιακές μεθόδους Μηχανικής Απαιτήσεων και τη δυνατότητα προσδιορισμού συναισθηματικών ζητημάτων (61).

Πώς, όμως, προσδιορίζονται συναισθηματικά ζητήματα που μπορούν να επηρεάσουν τις απαιτήσεις ενός συστήματος; Θα ήταν πολύ ευρηματικό, αν υπήρχε μία συστηματική μέθοδος που να αφορά στην ανάδειξη σχετικών ζητημάτων. Παρόλα αυτά, έχει αποδειχτεί πως δεν υπάρχει κάποια μέθοδος, η οποία, εγγυημένα, να προσδιορίζει όλα τα συναισθήματα που επηρεάζουν τις απαιτήσεις ενός συστήματος (61).

Όπως είναι προφανές, οι Μηχανικοί Απαιτήσεων θα πρέπει να είναι ενημερωμένοι για τις επιδράσεις των συναισθημάτων στις απαιτήσεις των συστημάτων. Η εκπαίδευσή τους προς αυτή τη κατεύθυνση είναι πολύ πιθανό να αποδώσει λύσεις σε πολύ κρίσιμες περιπτώσεις (61).

7.3 Προγραμματισμός Διαχείρισης Απαιτήσεων

Ο προγραμματισμός είναι ένα πολύ σημαντικό, πρώτο βήμα κατά τη διάρκεια της διαδικασίας Διαχείρισης Απαιτήσεων. Το στάδιο του προγραμματισμού καθορίζει το επίπεδο λεπτομέρειας της διαχείρισης απαιτήσεων που απαιτείται (59).



Σχήμα 7.2: Η εξέλιξη των απαιτήσεων κατά την Ανάπτυξη Λογισμικού

Κατά τη διάρκεια της διαδικασίας της διαχείρισης των απαιτήσεων θα πρέπει να ληφθούν αποφάσεις σχετικά με τα εξής (59):

1. Αναγνώριση Απαιτήσεων: Κάθε απαίτηση θα πρέπει να είναι μοναδικά καθορισμένη, ώστε να μπορεί να ελέγχεται σε σχέση με τις άλλες απαιτήσεις και να χρησιμοποιείται στις αξιολογήσεις ιχνηλασιμότητας.
2. Διαδικασία Διαχείρισης Αλλαγής: Αυτή η διαδικασία αποτελείται από δραστηριότητες αξιολόγησης της επίδρασης και του κόστους των αλλαγών.
3. Πολιτικές Ιχνηλασιμότητας: Αυτές οι πολιτικές καθορίζουν τις σχέσεις μεταξύ των απαιτήσεων, καθώς και των απαιτήσεων σε σχέση με τον σχεδιασμό του συστήματος που πρέπει να καταγραφεί. Οι πολιτικές ιχνηλασιμότητας θα πρέπει επίσης να καθορίζουν τον τρόπο («ΠΩΣ») με τον οποίο, αυτά τα αρχεία πρέπει να διατηρηθούν.
4. Εργαλεία Υποστήριξης Διαχείρισης Απαιτήσεων: Τα εργαλεία που μπορούν να χρησιμοποιηθούν μπορούν να είναι συστήματα διαχείρισης απαιτήσεων εξειδικευμένων, έγγραφα υπολογιστών, ή ακόμα και απλά συστήματα βάσεων δεδομένων.

Η διαδικασία διαχείρισης απαιτήσεων χρειάζεται αυτοματοποιημένη υποστήριξη και τα εργαλεία λογισμικού που την υποστηρίξουν, πρέπει να επιλέγονται κατά τη διάρκεια του σταδίου σχεδιασμού της. Απαραίτητα εργαλεία υποστήριξης είναι τα εξής (59):

1. Αποθήκευση Απαιτήσεων (Requirements storage): Οι απαιτήσεις θα πρέπει να διατηρούνται σε μία ασφαλή, διαχειρίσημη βάση δεδομένων, η οποία να είναι προσιτή σε όλα τα εμπλεκόμενα, στη διαδικασία Μηχανικής Απαιτήσεων, μέρη.
2. Διαχείριση Αλλαγής (Change Management): Η διαδικασία της διαχείρισης των αλλαγών απλοποιείται όταν χρησιμοποιούνται εργαλεία αυτοματοποιημένης υποστήριξης της.
3. Διαχείριση Ιχνηλασιμότητας (Traceability Management): Όπως συζητήθηκε παραπάνω, τα εργαλεία υποστήριξης ιχνηλασιμότητας διευκολύνουν τη διαδικασία εύρεσης σχετικών απαιτήσεων. Είναι διαθέσιμα αρκετά εργαλεία που χρησιμοποιούν τεχνικές φυσικής

γλώσσας για τη διευκόλυνση της διαδικασίας εύρεσης πιθανών σχέσεων μεταξύ απαιτήσεων.

Για μικρά συστήματα, μπορεί να μην είναι απαραίτητη η χρήση εξειδικευμένων εργαλείων διαχείρισης απαιτήσεων. Η διαδικασία διαχείρισης απαιτήσεων μπορεί να υποστηρίζεται με τη χρήση των διαθέσιμων λειτουργιών των επεξεργαστών κειμένου και βάσεων δεδομένων ηλεκτρονικών υπολογιστών. Ωστόσο, για μεγαλύτερα συστήματα, είναι απαραίτητη η χρήση εξειδικευμένων εργαλείων (59).

7.4 Διαχείριση Αλλαγής Απαιτήσεων

Η Διαχείριση Αλλαγής Απαιτήσεων θα πρέπει να εφαρμόζεται σε όλες τις προτεινόμενες αλλαγές στις απαιτήσεις ενός συστήματος, ύστερα από την έγκριση του εγγράφου απαιτήσεων. Η διαχείριση απαιτήσεων είναι ουσιώδης διότι, θα πρέπει να αποφασίζεται αν τα οφέλη της εισαγωγής νέων απαιτήσεων είναι αρκετά σε σχέση με τα κόστη της ικανοποίησής τους. Το πλεονέκτημα της χρήσης μίας τυπικής μεθόδου διαχείρισης απαιτήσεων είναι ότι, όλες οι προτάσεις αλλαγής αντιμετωπίζονται με συνέπεια και οι αλλαγές στο έγγραφο απαιτήσεων γίνονται με ελεγχόμενο τρόπο. Η διαδικασία Διαχείρισης της Αλλαγής χωρίζεται στα παρακάτω τρία, κύρια, στάδια (59):

1. Ανάλυση του Προβλήματος και Προδιαγραφή Αλλαγών: Η διαδικασία ξεκινά με ένα καθορισμένο πρόβλημα απαιτήσεων ή, μερικές φορές, με ένα συγκεκριμένη πρόταση αλλαγής. Κατά τη διάρκεια αυτού του σταδίου, το πρόβλημα ή η πρόταση αλλαγής αναλύονται ώστε να επικυρωθούν. Αυτή η ανάλυση δίνεται στο υποκείμενο της απαίτησης . Με τη μελέτη αυτής της ανάλυσης, το υποκείμενο, μπορεί να απαντήσει με μία πιο συγκεκριμένη πρόταση αλλαγής απαιτήσεων, ή να αποφασίσει να αναιρέσει την αρχική απαίτηση εξ' ολοκλήρου.
2. Ανάλυση Αλλαγής και Ανάλυση Κόστους: Το αποτέλεσμα της προτεινόμενης αλλαγής αξιολογείται με τη βοήθεια των πληροφοριών ιχνηλασιμότητας και γενικές γνώσεις των απαιτήσεων του συστήματος. Το κόστος πραγματοποίησης της αλλαγής εκτιμάται σε όρους τροποποιήσεων του εγγράφου απαιτήσεων και, αν είναι απαραίτητο, σε όρους σχεδιασμού και υλοποίησης του συστήματος. Με την ολοκλήρωση της συγκεκριμένης ανάλυσης γίνεται η λήψη της απόφασης σχετικά με την έγκριση ή την απόρριψη της αλλαγής απαιτήσεων.

3. Υλοποίηση Αλλαγής: Το έγγραφο απαιτήσεων και, όπου είναι απαραίτητο, ο σχεδιασμός και η υλοποίηση του συστήματος, τροποποιούνται. Το έγγραφο απαιτήσεων πρέπει να είναι οργανωμένο, ώστε οι αλλαγές σε αυτό, να μην απαιτούν τον εκτενή επανασχεδιασμό του. Με τον διαχωρισμό του εγγράφου σε ενότητες και υπό-ενότητες είναι εφικτή η τροποποίηση των μερών του εγγράφου, χωρίς να επηρεάζονται τα άλλα μέρη του.

Σε περιπτώσεις που μία απαίτηση πρέπει να εφαρμοστεί επειγόντως, διακρίνεται συχνά μία τάση, ώστε να αλλάζει πρώτα το σύστημα και ύστερα το έγγραφο των απαιτήσεων. Τέτοιες πρακτικές θα πρέπει να αποφεύγονται διότι οδηγούν αναπόφευκτα την υλοποίηση του συστήματος, εκτός σκοπού. Με την ολοκλήρωση των αλλαγών του συστήματος, δεν είναι καθόλου δύσκολο να αμελείται η καταγραφή των νέων απαιτήσεων, ή να προστίθενται ημιτελείς πληροφορίες σε σχέση με την υλοποίηση (59).

Οι διαδικασίες ανάπτυξης «Agile», όπως ο Ακραίος Προγραμματισμός (Extreme Programming), έχουν σχεδιαστεί ώστε να αντιμετωπίζουν τις απαιτήσεις που αλλάζουν κατά τη διάρκεια της διαδικασίας ανάπτυξης. Σε αυτές τις διαδικασίες, όταν ένας χρήστης προτείνει μία αλλαγή σε μία απαίτηση, η αλλαγή δεν αφορά μία τυπική διαδικασία Διαχείρισης Αλλαγής. Ο χρήστης θα πρέπει να ιεραρχήσει την αλλαγή και, εφόσον είναι υψηλής προτεραιότητας, να αποφασίσει ποιά χαρακτηριστικά του συστήματος που είχαν σχεδιαστεί για την επόμενη επανάληψη, θα πρέπει να απορριφθούν (59).

7.5 Οφέλη Εφαρμογής Τεχνικών Διαχείρισης Απαιτήσεων

Η Διαχείριση Απαιτήσεων, ως διαδικασία, αντιμετωπίζεται, συχνά, με επιφανειακό τρόπο. Δεν είναι λίγες οι περιπτώσεις κατά τις οποίες η Διαχείριση Απαιτήσεων αποτελεί δευτερεύουσας σημασίας διαδικασία, σε σχέση με αυτή της υλοποίησης. Γι αυτό το λόγο, παρακάτω, παρατίθενται μερικά από τα βασικά οφέλη της εφαρμογής τεχνικών Διαχείρισης Αλλαγής Απαιτήσεων (58):

1. *Αποτελεσματικός έλεγχος πολύπλοκων έργων:* Η έλλειψη γνώσεων σχετικά με την προβλεπόμενη συμπεριφορά του συστήματος, καθώς και των σχετικών απαιτήσεων είναι κοινοί παράγοντες σε πολλά, πολύπλοκα έργα. Τη διαχείριση των απαιτήσεων αναλαμβάνει η ομάδα του έργου, με σαφή κατανόηση σχετικά με το «τί» πρέπει να παραδοθεί,

«πότε» και «γιατί». Οι πόροι μπορούν να κατανεμηθούν με βάση τις προτεραιότητες των πελατών και τη σχετική προσπάθεια υλοποίησης. Η επίδραση των αλλαγών μπορεί έτσι να κατανοηθεί πιο εύκολα, ενώ η διαχείριση τους διευκολύνεται σε μεγάλο βαθμό.

2. *Βελτίωση Ποιότητας Λογισμικού και Ικανοποίηση Πελατών:* Το βασικό μέτρο της ποιότητας λογισμικού είναι «Αν το λογισμικό εκπληρώνει το σκοπό για τον οποίο αναπτύχθηκε». Υψηλότερο επίπεδο ποιότητας λογισμικού μπορεί να επιτευχθεί μόνο όταν οι προγραμματιστές και το προσωπικό ελέγχου κατανοούν συνοπτικά «τί» ακριβώς πρέπει να σχεδιαστεί και να ελεγχθεί.
3. *Μείωση κόστους και αποφυγή καθυστερήσεων:* Σχετικές έρευνες έχουν δείξει ότι, τα λάθη που αφορούν σε απαιτήσεις είναι αυτά που κοστίζουν και περισσότερο κατά την επιδιόρθωσή τους, ενώ προκαλούν σημαντική καθυστέρηση, εφόσον η ομάδα του έργου καθυστερήσει να τα ανακαλύψει. Μειώνοντας αυτά τα λάθη, στα αρχικά στάδια του κύκλου ζωής του έργου ανάπτυξης, μειώνεται το σύνολο των λαθών, το κόστος του έργου και ο χρόνος παράδοσής του.
4. *Βελτίωση της Επικοινωνίας των μελών της Ομάδας:* Η Διαχείριση Απαιτήσεων απαιτεί την ανάμειξη των πελατών στα αρχικά στάδια του έργου, ώστε να επιβεβαιώνεται, ότι οι ανάγκες τους ικανοποιούνται. Ένα κεντρικό αποθετήριο αναγκών και δεσμεύσεων κατασκευάζει έναν κοινό τρόπο επικοινωνίας και κατανόησης μεταξύ των χρηστών, της διοίκησης, των αναλυτών, των προγραμματιστών και του προσωπικού ελέγχου.
5. *Διευκόλυνση Συμμόρφωσης με τα πρότυπα και τους κανονισμούς:* Οι πιο μεγάλοι οργανισμοί προτυποποίησης, καθώς και οι ρυθμιστικές αρχές και οργανισμοί που ασχολούνται με το λογισμικό και τη βελτίωση των διαδικασιών ανάπτυξης κατανοούν σε βάθος το πρόβλημα της Διαχείρισης Απαιτήσεων. Για παράδειγμα, το Ινστιτούτο CMM (Capability Maturity Model) αντιμετωπίζει τη Διαχείριση Απαιτήσεων, ως ένα από τα πρώτα βήματα για τη βελτίωση της ποιότητας λογισμικού. Τα πρότυπα DOD, FDA, και ISO9000 απαιτούν από τις εταιρείες να επιδείξουν ωριμότητα και έλεγχο της διαδικασίας.

7.6 Εργαλεία Διαχείρισης Απαιτήσεων

Η Διαχείριση Απαιτήσεων αποτελεί σημαντική δραστηριότητα κατά τη διαδικασία ανάπτυξης ενός λογισμικού και περιλαμβάνει την εκμαίευση, τη μοντελοποίηση, την ανάλυση και την επικοινωνία των αντίστοιχων απαιτήσεων. Οι υπεύθυνοι της Διαχείρισης Απαιτήσεων αντιμετωπίζουν αρκετά προβλήματα που συχνά οδηγούν σε κακή εκτίμηση των απαιτήσεων(62).

Τα προβλήματα στην ανάλυση απαιτήσεων προκύπτουν συνήθως ως αποτέλεσμα των παρακάτω:

- Οι απαιτήσεις προκύπτουν από διαφορετικές οπτικές γωνίες και ενδέχεται να αλληλεπικαλύπτονται, να αλληλοσυμπληρώνονται ή και να αντιφάσκουν (63)
- Οι απαιτήσεις προκύπτουν ως αποτέλεσμα σύνθεσης διαφορετικών προσεγγίσεων – απαιτήσεων διαφορετικών ομάδων χρηστών του συστήματος, με προβλήματα επικοινωνίας των διαφορετικών προσεγγίσεων μεταξύ των χρηστών (62)
- Οι απαιτήσει είναι πολύ πιθανόν να αλλάξουν κατά τον κύκλο ανάπτυξης του λογισμικού (62)
- Οι απαιτήσεις συνήθως εκφράζονται με φυσική γλώσσα και είναι δύσκολος ο εντοπισμός και η κατανόηση της πραγματικής ανάγκης που αναφέρονται (64)

Λύση στα παραπάνω προβλήματα μπορούν να δώσουν ειδικά εργαλεία διαχείρισης απαιτήσεων που υπάρχουν στην αγορά(62).

Ο κύριος στόχος των εργαλείων διαχείρισης απαιτήσεων είναι να αυξήσουν τη πιθανότητα πως το τελικό προϊόν θα λειτουργεί όπως έχει προβλεφθεί και θα αυξήσει την αποδοτικότητα της επιχείρησης όπως έχει προβλεφθεί. Σύμφωνα με το Forrester (65), σχετίζονται με "*την αποθήκευση των απαιτήσεων, τον προσδιορισμό των σχέσεων μεταξύ των απαιτήσεων και τον έλεγχο των απαιτούμενων αλλαγών στις απαιτήσεις ή σε ομάδες απαιτήσεων*."

Ορισμένοι λόγοι υπέρ της χρήσης εργαλείων διαχείρισης απαιτήσεων είναι οι εξής (66):

- **Διαχείριση εκδόσεων λογισμικού και αλλαγών:** στο έργο πληροφορικής θα πρέπει να καθοριστούν αρχικές απαιτήσεις. Με τη βοήθεια των εργαλείων μπορούν να καταγράφονται οι αλλαγές στις απαιτήσεις, οι λόγοι της αλλαγής και είναι δυνατή η επιστροφή σε προηγούμενα καθορισμένες απαιτήσεις εάν κριθεί απαραίτητο

- **Αποθήκευση χαρακτηριστικών απαιτήσεων:** τα εργαλεία επιτρέπουν την αποθήκευση χαρακτηριστικών όπως συγγραφέας, υπεύθυνος ανάπτυξης, αιτία προέλευσης, αριθμός έκδοσης, κατάσταση, προτεραιότητα, κόστος, δυσκολία που βοηθάνε στην διαχείριση της ανάπτυξης του λογισμικού

- **Παρακολούθηση αλλαγών:** η παρακολούθηση της κατάστασης κάθε απαίτησης κατά την ανάπτυξη, διευκολύνει τη συνολική παρακολούθηση - επίβλεψη της κατάστασης του έργου

- **Σύνδεση απαιτήσεις στα άλλα στοιχεία του συστήματος:** Επιτρέπουν τον ορισμό εξάρτησης μεταξύ διαφόρων απαιτήσεων και απαιτήσεις σε διάφορα υποσυστήματα. Κατά την ανάλυση των επιπτώσεων μιας πιθανής αλλαγής απαιτήσεων, είναι δυνατή η ιχνηλάτιση των στοιχείων του συστήματος που θα επηρεαστούν

- **Προσδιορισμό υποσυστημάτων απαιτήσεων:** είναι δυνατή η ταξινόμηση, φιλτράρισμα και ενημέρωση χαρακτηριστικών της βάσης δεδομένων σύμφωνα με ερωτήματα που υποβάλλονται για απαιτήσεις με ειδικές τιμές χαρακτηριστικών

- **Έλεγχο πρόσβασης:** ορισμό δικαιωμάτων πρόσβασης σε χρήστες ή ομάδες χρηστών

- **Επικοινωνία στους χρήστες:** Τα περισσότερα εργαλεία διαχείρισης απαιτήσεων επιτρέπουν στα μέλη της ομάδας να συζητήσουμε θέματα απαιτήσεων ηλεκτρονικά. Επίσης, επιτρέπουν την άμεση ενημέρωσή τους σε περίπτωση προσθήκης νέας απαίτησης ή αναθεώρηση υπάρχουσας

Επιπροσθέτως, ένα εργαλείο διαχείρισης απαιτήσεων θα πρέπει να ικανοποιεί ορισμένες προδιαγραφές για τους προγραμματιστές, τα άτομα που διαχειρίζονται το έργο πληροφορικής και τα άτομα που διαχειρίζονται το εργαλείο διαχείρισης απαιτήσεων(67).

Το εργαλείο που θα επιλεγεί θα πρέπει να **παρέχει στους προγραμματιστές**(67):

- Μοντέλο Πληροφοριών: Δυνατότητα αλλαγών, Υποστήριξη κληρονομικότητας και επαναχρησιμοποίησης τμημάτων του μοντέλου, γραφική αναπαράσταση,
- Αναφορές (views): Πολλαπλές αναφορές - εκθέσεις για κάθε απαίτηση ή ομάδα απαιτήσεων, υποστήριξη πολλαπλών φίλτρων, γραφική αναπαράσταση
- Μορφοποίηση, εξωτερικά αρχεία: υποστήριξη χαρακτήρων από όλες τις γλώσσες, εμφάνιση εξωτερικών αρχείων, αποθήκευση κάθε μορφής δεδομένων (εικόνα, βίντεο) εσωτερικά στη βάση δεδομένων του εργαλείου
- Διαχείριση αλλαγών: Οι αιτήσεις για αλλαγή θα πρέπει να είναι της μορφής αναμονή, αποδοχή, απόρριψη. Δυνατότητα προσθήκης σχολίων
- Τεκμηρίωση αλλαγών: όλες οι αλλαγές να αποθηκεύονται στη βάση δεδομένων, προσθήκη σχολίων στις αλλαγές, εκδόσεις αλλαγών, δυνατότητα επαναφοράς, γραφική απεικόνιση, αναφορές αλλαγών και προόδου του έργου
- Γραμμή βάσης (baseline): ξεχωριστή αποθήκευση στη βάση δεδομένων για κάθε κατάσταση του έργου
- Ιχνηλασιμότητα: Το εργαλείο θα πρέπει να είναι σε θέση να επιβάλει τη δημιουργία ή αλλαγή σχέσεων στα δεδομένα κατά τη δημιουργία ή την αλλαγή απαιτήσεων, πιθανή διασύνδεση με όλα τα αντικείμενα του έργου και όχι ενός υποσυνόλου αυτών
- Λειτουργίες Ανάλυσης: πληροφορίες σχετικά με την κατάσταση και πρόοδο του έργου

- Διασύνδεση: με άλλα λογισμικά, ομοιότητα στη διασύνδεση με εξωτερικά αντικείμενα και διασύνδεση με εσωτερικά αντικείμενα.
- Εισαγωγή: δεδομένα από άλλες εφαρμογές ή από αρχεία
- Παραγωγή εγγράφων: συμπερίληψη όλων των πληροφοριών που περιέχονται στο εργαλείο, συμπερίληψη όχι μονό δεδομένων – μα κάθε μορφής δεδομένα, σταθερή μορφοποίηση, εργασία στο παρασκήνιο, ταχύτητα στη δημιουργία εκθέσεων
- Ομαδική εργασία: δυνατότητα παράλληλης εργασίας πολλών ατόμων στο ίδιο αντικείμενο
- Χρήση χωρίς σύνδεση: δυνατότητα εργασίας χωρίς σύνδεσης και ενημέρωσης κεντρικής βάσης δεδομένων με την τρέχουσα κατάσταση
- Πρόσβαση διαδικτύου: θα πρέπει να παρέχεται διεπαφή διαδικτύου, χωρίς να απαιτείται η εγκατάσταση ειδικού λογισμικού

Σχετικά με τους **διαχειριστές του έργου** θα πρέπει να παρέχει τις παρακάτω δυνατότητες(67):

- Κεντρική εγκατάσταση και διαχείριση έργων: Όλα τα έργα θα πρέπει να τηρούνται σε αν μόνο μέρος. Παροχή ιστορικού αλλαγών
- Χρήστες και δικαιώματα: Ο διαχειριστής πρέπει να είναι σε θέση να διαχειρίζεται τους χρήστες, τα δικαιώματα και τους ρόλους χρηστών κεντρικά, υψηλά επίπεδα ασφάλειας, δυνατότητα πολλαπλών ρόλων σε χρήστες
- Μέγεθος: κανένας περιορισμός στον αριθμό απαιτήσεων, χρηστών, ρόλων, μέγεθος βάσης δεδομένων
- Διαχείριση ροών: καθοδήγηση των χρηστών μέσω του συστήματος (ροή εργασιών)
- Επεκτασιμότητα: δυνατότητα χρήσης δεδομένων σε διαφορετικές εκδόσεις του ίδιου εργαλείου, προσωποποίηση

διεπαφών, επικοινωνία – ανταλλαγή δεδομένων με άλλες εφαρμογές

Τέλος, σχετικά με τον **διαχειριστή του εργαλείου** διαχείρισης απαιτήσεων θα πρέπει να παρέχει τα παρακάτω(67):

- Βάση Δεδομένων: διαθεσιμότητα βάσης δεδομένων 24 ώρες το 24ωρο, 365 μέρες το χρόνο, υποστήριξη συναλλαγών, συνοδευτικά εργαλεία εισόδου – εξόδου δεδομένων σε διαφορετικές μορφές, αντίγραφα ασφαλείας, δυνατότητες επαναφοράς

- Κρυπτογράφηση: αποθήκευση κρυπτογραφημένων δεδομένων. Ανταλλαγή δεδομένων μεταξύ χρηστών με κρυπτογραφημένο τρόπο.

Τα διαθέσιμα εργαλεία διαχείρισης απαιτήσεων μπορούν να χωριστούν σε τέσσερις (4) κατηγορίες σύμφωνα με τις απαιτήσεις του οργανισμού πληροφορικής (68):

- Επιχειρηματικά (enterprise) Εργαλεία Διαχείρισης Απαιτήσεων: απευθύνονται σε μεγάλες επιχειρήσεις και είναι ιδιαίτερα ακριβά (IBM Rational DOORS, Borland Caliber)

- Μεσαίας Αγοράς (mid-market) Εργαλεία Διαχείρισης Απαιτήσεων: παρέχουν μια ισορροπία μεταξύ των δυνατοτήτων που παρέχουν τα enterprise εργαλεία και της ευκολίας χρήσης των entry-level εργαλείων της επόμενης κατηγορίας (Accompa, Jama)

- Εισαγωγικά (entry-level) Εργαλεία Διαχείρισης Απαιτήσεων: είναι οικονομικά και επιτρέπουν τη διαχείριση απαιτήσεων με δομημένο τρόπο. Το βασικό τους μειονέκτημα είναι πως δεν επικεντρώνονται σε απαιτήσεις, αλλά περισσότερο σε θέματα ανάπτυξη και σφάλματα υλοποίησης (Atlassian JIRA, FogBugz)

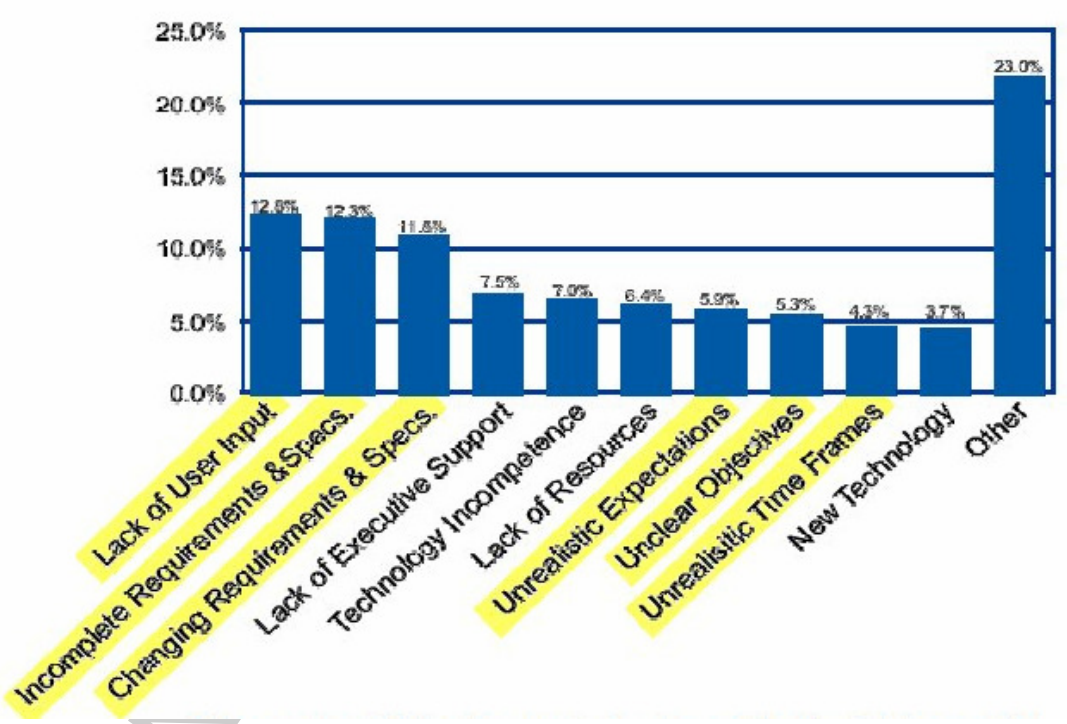
- Δωρεάν (open source) Εργαλεία Διαχείρισης Απαιτήσεων: δωρεάν εργαλεία διαχείρισης απαιτήσεων με περιορισμένες δυνατότητες σε σχέση με τα enterprise (aNimble)

Το είδος εργαλείων που θα επιλέξει ένας οργανισμός πληροφορικής σχετίζεται με το μέγεθός του, τα έργα που αναλαμβάνει και τα χρήματα που μπορεί να διαθέσει για την αγορά του (68).

7.7 Προκλήσεις Στη Μηχανική Απαιτήσεων

Η Μηχανική Απαιτήσεων είναι κρίσιμη για την επιτυχία ενός συστήματος ή έργου λογισμικού**.** Παρά το γεγονός ότι έχουν προταθεί δεκάδες μέθοδοι μηχανικής απαιτήσεων και υπάρχουν διαθέσιμες αρκετές τεχνικές στους επαγγελματίες, έχουν αποδειχθεί στη πράξη πως οι προτεινόμενες τεχνικές δεν είναι εξίσου αποτελεσματικές σε όλα τα συστήματα που έχουν εφαρμοστεί(69).

Στο Σχήμα 7.3, απεικονίζονται οι κυριότεροι λόγοι αποτυχίας έργων Πληροφορικής. Οι λόγοι που σχετίζονται με τη Μηχανική Απαιτήσεων επισημαίνονται με κίτρινο χρώμα(70):



Σχήμα 7.3: Λόγοι Αποτυχίας Έργων Πληροφορικής

Όπως μπορεί να διαπιστωθεί και από το σχήμα η **κύρια αιτία αποτυχίας ενός έργου Πληροφορικής είναι η αποτυχημένη Ανάλυση - Διαχείριση των Απαιτήσεων ενός προγράμματος.** Ο ίδιος λόγος αναγνωρίζεται ως βασικότερος λόγος αποτυχίας έργων Πληροφορικής και από άλλους ερευνητές (Hoffman & Lehner 2001;(31)). Η μηχανική απαιτήσεων πέραν της επιτυχίας ενός έργου Πληροφορικής, επηρεάζει σε μεγάλο βαθμό τη παραγωγικότητα και ποιότητα του τελικού προϊόντος(71).

Οι προκλήσεις – δυσκολίες στη Μηχανική απαιτήσεων έχουν να κάνουν κατά κύριο λόγο με τους χρήστες του συστήματος και τα άτομα που θα αναλάβουν την υλοποίησή του (κυρίως τους προγραμματιστές).

Οι **χρήστες του συστήματος** μπορούν να εμποδίσουν τη σωστή ανάλυση απαιτήσεων και καταγραφή τους, με διάφορους τρόπους, όπως αναφέρει και ο McConnell(72):

- Οι χρήστες δε γνωρίζουν ακριβώς τι θέλουν, δε γνωρίζουν ακριβώς τις απαιτήσεις του νέου λογισμικού
- Επιμένουν σε νέες αλλαγές και καθορισμό νέων απαιτήσεων μετά τη κοστολόγηση και τον προγραμματισμό ανάπτυξης του έργου
- Δεν γνωρίζουν τις διαθέσιμες τεχνολογίες στην αγορά
- Αποφεύγουν τη συμμετοχή τους σε συνεντεύξεις συλλογής πληροφοριών, καθώς δε τις θεωρούν απαραίτητες
- Υπάρχει πολλές φορές καθυστερήσεις στην επικοινωνία και στην ανταπόκριση από τους χρήστες
- Αποφεύγουν σχεδόν πάντα τη καταγραφή των απαιτήσεων και τη συμφωνία επί αυτών

Οι **μηχανικοί - προγραμματιστές** από τη μεριά τους συχνά(72):

- Ξεκινάνε το γράψιμο κώδικα και την υλοποίηση του προγράμματος, πριν την ολοκλήρωση την ανάλυση απαιτήσεων
- Οι τεχνικοί και οι χρήστες δε χρησιμοποιούν κοινό λεξιλόγιο και πολλές φορές βρίσκονται σε πλήρη συμφωνία μέχρι την ολοκλήρωση ανάπτυξης του προγράμματος, οπότε και διαπιστώνονται αποκλίσεις
- Οι μηχανικοί και οι προγραμματιστές πολλές φορές προσπαθούν να προσαρμόσουν τις απαιτήσεις του πελάτη σε υπάρχον σύστημα ή μοντέλο αντί να δημιουργήσουν ένα καινούργιο σύστημα βασισμένο αποκλειστικά στις ανάγκες του πελάτη

- Η ανάλυση συχνά πραγματοποιείται από μηχανικούς και προγραμματιστές και όχι από άτομα με ειδικές γνώσεις στον τομέα που δραστηριοποιείται ο πελάτης με αποτέλεσμα να μη μπορούν να κατανοήσουν σωστά τις ανάγκες του πελάτη

Οι τομείς όπου εντοπίζονται τα σημαντικότερα προβλήματα - προκλήσεις της Μηχανικής Απαιτήσεων σχετίζονται με:

- την **έλλειψη επικοινωνίας** και κατανόησης μεταξύ των δυο μεριών (χρήστες – οργανισμός πληροφορικής) (73)
- το **μεγάλο χρονικό διάστημα** που απαιτείται για την ολοκλήρωση της Ανάλυσης Απαιτήσεων και παρακολούθησης εξέλιξης του έργου για πιθανή αναθεώρηση των απαιτήσεων, που λειτουργεί αρνητικά για τη Μηχανική Απαιτήσεων (74)
- την υψηλή ταχύτητα αλλαγών στο επιχειρηματικό περιβάλλον και στις τεχνολογίες πληροφορικής, που έχουν ως αποτέλεσμα **μικρότερο κύκλο ζωής για τα έργα πληροφορικής**. Το γεγονός αυτό προκαλεί ακόμα περισσότερα προβλήματα στη Μηχανική Απαιτήσεων(69).
- Τους ίδιους τους χρήστες – ενδιαφερόμενους του έργου πληροφορικής. Οι χρήστες - ενδιαφερόμενοι σε ένα έργο πληροφορικής είναι συνήθως πολλοί, οι οποίοι και προσεγγίζουν το έργο από διαφορετικές οπτικές γωνίες, θέτουν διαφορετικές προτεραιότητες και δεν έχουν τις ίδιες προσδοκίες από το πρόγραμμα που θα αναπτυχθεί. Η πρόκληση έγκειται στο **γεφύρωμα των προσδοκιών μεταξύ των χρηστών** και στη διαπραγμάτευση με αυτούς (75).

Οι προκλήσεις στη Μηχανική Απαιτήσεων που σχετίζονται αποκλειστικά με τη διαχείριση των απαιτήσεων ενός λογισμικού είναι οι εξής(76):

- **Ασαφείς απαιτήσεις:** μπορούν να οδηγούν σε πλήρη αποτυχία του έργου, απώλεια χρόνου επανακαθορισμού των και οικονομική επιβάρυνση

- **Μεταβαλλόμενες απαιτήσεις:** οι απαιτήσεις ενός προγράμματος εξελίσσονται συνεχώς, λόγω της φύσης της αγοράς σήμερα. Συνεπώς θα πρέπει να υπάρχει συνεχή παρακολούθηση των απαιτήσεων που έχουν αρχικά προσδιοριστεί και τον αλλαγών στο περιβάλλον της επιχείρησης και τον ίδιο τον οργανισμό

- **Ορισμός απαιτήσεων:** απαιτείται η χρήση ειδικών εργαλείων για τη καταγραφή όλων των χαρακτηριστικών των απαιτήσεων, των συσχετίσεων μεταξύ τους και τη παρακολούθηση των αλλαγών

- **Προτεραιότητα απαιτήσεων:** όταν η λίστα των απαιτήσεων προς υλοποίηση είναι μεγάλη (και διαρκώς μεταβαλλόμενη) αποτελεί πρόκληση η επιλογή της σειράς υλοποίησης των απαιτήσεων και η πρόβλεψη της πιθανότητα αλλαγών σε κάποιες από αυτές

Λύσεις στις προκλήσεις - προβλήματα που αναφέρθηκαν παραπάνω μπορούν να δώσουν:

- Η πρόσληψη ειδικών αναλυτών στο χώρο που δραστηριοποιείται ο πελάτης για τη διεκπεραίωση του σταδίου της ανάλυσης απαιτήσεων (77)

- Η χρήση ειδικών τεχνικών στο στάδιο της ανάλυσης (όπως Προτυποποίηση, UML, Use Case) (78)

- Η χρήση ειδικών λογισμικών προσομοίωσης ή λογισμικών ορισμού απαιτήσεων, με σκοπό τη γεφύρωση του χάσματος επικοινωνίας μεταξύ των χρηστών της επιχείρησης και του οργανισμού πληροφορικής που έχει αναλάβει την υλοποίηση του λογισμικού (67)

7.8 Ευέλικτες Μεθοδολογίες (Agile) στη Διαχείριση Απαιτήσεων

Σύμφωνα με τον McPhee (79) οι εργασίες Διαχείριση των Απαιτήσεων θα πρέπει να αντιπροσωπεύουν το 25% των συνολικών εργασιών υλοποίησης ενός έργου Πληροφορικής. Επίσης, ο Taylor (80) αναφέρει πως 70% των αποτυχιών σε έργα Πληροφορικής οφείλονται σε λανθασμένο προσδιορισμό –

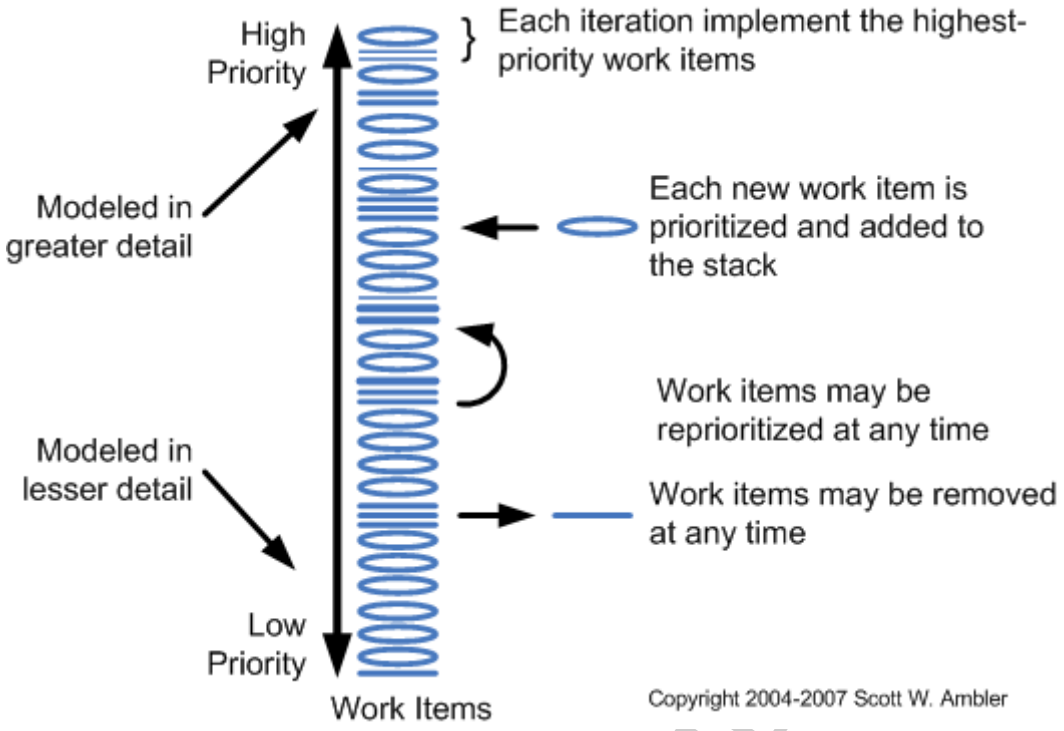
διαχείριση απαιτήσεων και πως το 80% των επιτυχημένων έργων Πληροφορικής οφείλουν την επιτυχία τους σε λεπτομερής και σωστά προσδιορισμένες απαιτήσεις.

Στις μέρες μας ο μεγάλος ανταγωνισμός και οι περίπλοκες επιχειρηματικές διαδικασίες πιέζουν τις εταιρίες Πληροφορικής για ταχεία παράδοση των έργων Πληροφορικής που αναλαμβάνουν(81). Οι ευέλικτες μεθοδολογίες ανάπτυξης λογισμικού (Agile methodologies) την τελευταία δεκαετία γίνονται ολοένα και πιο δημοφιλής. Έρευνα που διεξήχθη από τη Shine Technologies (82) σε εταιρίες ανάπτυξης λογισμικού, έδειξε πως το 92,8% των ερωτηθέντων θεωρούν πως οι ευέλικτες μεθοδολογίες ανάπτυξης λογισμικού έχουν κάνει την ομάδα τους περισσότερο παραγωγική και το 84% παρατήρησε αύξηση της ποιότητας των προϊόντων ως αποτέλεσμα των νέων μεθοδολογιών.

Ο σημαντικότερος λόγος αποτυχίας των έργων πληροφορικής είναι η έλλειψη επικοινωνίας με τον πελάτη. Συνήθως ένας πελάτης ζητά ένα σύστημα, προσδιορίζονται οι απαιτήσεις σε συνεννόηση με τον πελάτη και μετά για έξι μήνες το τμήμα Πληροφορικής είναι εξαφανισμένο. Το αποτέλεσμα είναι το τελικό προϊόν να μην είναι αυτό που πραγματικά ήθελε ο πελάτης. Με τις μεθοδολογίες Agile, υπάρχει άμεση εμπλοκή του πελάτη σε όλα τα στάδια του κύκλου ζωής ενός λογισμικού (83).

Οι ευέλικτες προσεγγίσεις (Agile) αποβλέπουν στη μείωση των εγγράφων τεκμηρίωση στο απολύτως απαραίτητο με παράλληλη αύξηση της άτυπης επικοινωνίας μεταξύ των μελών της ομάδας(84). Σε ευέλικτες προσεγγίσεις ανάπτυξης λογισμικού «οι μέθοδοι agile δεν είναι προσανατολισμένες προς την έγγραφη τεκμηρίωση, αντίθετα είναι προσανατολισμένες προς τον κώδικα της εφαρμογής. Θεωρούν ως βασικότερο κομμάτι τεκμηρίωσης ενός έργου Πληροφορικής τον ίδιο τον κώδικα της εφαρμογής»(85).

Η διαχείριση απαιτήσεων με τη χρήση ευέλικτων μεθοδολογιών απεικονίζεται στο Σχήμα 7.4:



Σχήμα 7.4: Διαχείριση απαιτήσεων με τη χρήση ευέλικτων μεθοδολογιών στην ανάπτυξη λογισμικού

Όπως φαίνεται και από το Σχήμα 7.4 οι απαιτήσεις για το λογισμικό ταξινομούνται ως προς την προτεραιότητα υλοποίηση και με βάση τη προτεραιότητά τους προσδιορίζονται - τεκμηριώνονται ή όχι σε λεπτομέρεια. Μόνο για τις υψηλής προτεραιότητας απαιτήσεις ακολουθούνται με συνέπεια τα στάδια ανάλυσης απαιτήσεων που προσδιορίστηκαν παραπάνω. Για τις υπόλοιπες απαιτήσεις αρκεί ένας πρόχειρος προσδιορισμός απαιτήσεων(86).

Αν προκύψει ανάγκη για κάποια νέα απαίτηση, αυτή ταξινομείται ως προς τη προτεραιότητα υλοποίησης και αναλόγως ακολουθεί λεπτομερή τεκμηρίωσή της ή όχι. Κατά τον κύκλο ζωής ενός λογισμικού μπορεί κάποια απαίτηση να αφαιρεθεί από το σύστημα ή να αλλάξει βαθμό προτεραιότητας(86).

Η διαχείριση απαιτήσεων σε ευέλικτες μεθοδολογίες περιορίζεται σε δυο (2) στάδια (87):

1. Αρχική Μοντελοποίηση Απαιτήσεων (Initial Requirements Modeling): στην αρχή του έργου, λαμβάνει χώρα μια περιορισμένη σε μέγεθος ανάλυση απαιτήσεων, αρκετή ώστε να σχηματιστεί η «*μεγάλη εικόνα (big picture)*» του έργου

2. Ανάδειξη Μοντέλου (Model Storming): αναλύονται οι απαιτήσεις υψηλού επιπέδου που εντοπίστηκαν κατά την αρχική φάση μοντελοποίηση απαιτήσεις. Αυτό περιλαμβάνει τη συλλογή στοιχείων με τη μορφή just-in-time. Ζητούνται από τον πελάτη περισσότερες πληροφορίες για τις ανάγκες – απαιτήσεις που έχει από το λογισμικό τη στιγμή κωδικοποίησης της αντίστοιχης απαίτησης.

Οι βασικότερη διαφοροποίηση της ευέλικτης διαχείρισης απαιτήσεων, είναι πως με αυτές οι απαιτήσεις προκύπτουν **just-in-time**, ενώ με τις παραδοσιακές μεθόδους προηγείται λεπτομερή ανάλυση πριν της έναρξης υλοποίησης του έργου. Επίσης, σε μεθόδους agile υπάρχει περιορισμός των εγγράφων τεκμηρίωσης, ενώ με τις παραδοσιακές μεθόδους προηγείται η σύνταξη του εγγράφου προδιαγραφών τεκμηρίωσης (SRS). Επίσης, στη 1η περίπτωση η έλεγχος και η επικύρωση των απαιτήσεων λαμβάνει χώρα just-in-time, ενώ στη 2η περίπτωση ως ξεχωριστή διαδικασία – στάδιο. Τέλος, στη πρώτη περίπτωση ο πελάτης είναι παρών για να απαντήσει κάθε ερώτηση της ομάδας υλοποίησης στη διάρκεια ανάπτυξης του λογισμικού, κάτι που δε συμβαίνει στη 2η περίπτωση(87).

Βέβαια σε κάθε περίπτωση, χρήσης ή μη ευέλικτων μεθοδολογιών, η ανάλυση και διαχείριση απαιτήσεων είναι ένα σημαντικό στάδιο στην ανάπτυξη ενός λογισμικού και δε θα πρέπει ποτέ να παραλείπεται(88)**.**

7.9 Κρίσιμοι παράγοντες επιτυχίας (critical success factors)

Η ανάγκη για διαχείριση απαιτήσεων είναι αναγνωρισμένη ευρέως. Η αποτελεσματική διαχείριση απαιτήσεων είναι απαραίτητη για την **υλοποίηση λογισμικού που ικανοποιεί τις ανάγκες - απαιτήσεις του πελάτη**(89).

Με τη διαχείριση απαιτήσεων αυξάνονται οι πιθανότητες για τον πελάτη να παραλάβει το λογισμικό στο χρόνο που ζήτησε και εντός προϋπολογισμού. Επίσης, η εταιρία που έχει αναλάβει την ανάπτυξη του λογισμικού, με τη βοήθεια της διαχείρισης απαιτήσεων, μπορεί να προγραμματίσει ορθά τους πόρους που θα χρειαστούν, το χρόνο υλοποίησης, τις ανάγκες για επενδύσεις και να θέσει με μεγαλύτερη ακρίβεια τα ορόσημα (milestone) για τον έλεγχο της πορείας υλοποίησης των απαιτήσεων(89).

Παρά την ανάπτυξη πολλών μοντέλων για τη διενέργεια της Μηχανικής Απαιτήσεων και την ύπαρξη προτάσεων σχετικά με τη χρήση συναφών

μεθοδολογιών, ακόμα και σήμερα μεγάλος αριθμός έργων πληροφορικής ξεπερνά τον προϋπολογισμό του έργου και παραδίδεται με μεγάλες καθυστερήσεις(90).

Σε έρευνά τους οι Nasir & Sahibuddin (91) σχετικά με τους **κρίσιμους παράγοντες επιτυχίας** της Διαχείρισης Απαιτήσεων, διαπίστωσαν πως οι οχτώ (8) παράγοντες επιτυχίας που συναντώνται συχνότερα στη διεθνή βιβλιογραφία είναι οι εξής:

1. Ξεκάθαρες απαιτήσεις και προδιαγραφές
2. Σαφείς καθορισμένους στόχους
3. Ρεαλιστικά χρονοδιαγράμματα
4. Υψηλές δεξιότητες διαχείρισης έργων από τους υπεύθυνους της διαχείρισης απαιτήσεων
5. Υποστήριξη από την κορυφή της διοίκησης
6. Συμμετοχή πελάτη
7. Αποτελεσματική επικοινωνία
8. Ρεαλιστικοί προϋπολογισμοί

Αν η διαχείριση απαιτήσεων ενός έργου πληροφορικής, χαρακτηρίζεται από τα παραπάνω, έχει πολύ μεγάλη πιθανότητα να στεφθεί με επιτυχία.

8 Μελέτη Περίπτωσης

Έχοντας ολοκληρώσει την παρουσίαση των σταδίων της ανάλυσης των απαιτήσεων και των εργαλείων – μεθόδων που μπορούν να εφαρμοστούν σε κάθε στάδιο, στη συνέχεια παρουσιάζεται μια εφαρμογή των σταδίων αυτών σε ένα έργο Πληροφορικής.

8.1 Περιγραφή μελέτης περίπτωσης

Κατά καιρούς έχει γίνει μεγάλη συζήτηση για τα μειονεκτήματα και πλεονεκτήματα, πιθανής εφαρμογής της ηλεκτρονικής ψηφοφορίας έναντι των παραδοσιακών μεθόδων ψηφοφορίας στις δημοκρατίες. Η προοπτική να είναι σε θέση οι πολίτες να ψηφίσουν "*με τις πιτζάμες τους*" όπως έχει χαρακτηριστεί για την ηλεκτρονική ψηφοφορία, εξάπτει τη φαντασία των πολιτικών αρχηγών, των προοδευτικών πολιτών και γενικότερα των ψηφοφόρων σε ολόκληρο τον κόσμο (92).

Τα πλεονεκτήματα που έχουν αναφερθεί για την ηλεκτρονική ψηφοφορία είναι τα παρακάτω (93):

- Μείωση κόστους διεξαγωγής εκλογών
- Βελτίωση διοικητικής αποτελεσματικότητας
- Αύξηση ποσοστού συμμετοχής
- Χρήση τεχνολογίας και σε άλλες διαδικασίες λήψης αποφάσεων

Μερικά από τα πιθανά μειονεκτήματα, για την ηλεκτρονική ψηφοφορία είναι τα εξής (93):

- Ακεραιότητα ψήφου
- Μυστικότητα ψήφου
- Αποκλεισμός κοινωνικών ομάδων
- Υποβιβασμός της άσκησης ενός θεμελιώδους πολιτικού δικαιώματος

Οι Kevin Daimi, Katherine Snyder, and Robert James (94) περιγράφουν τις απαιτήσεις ενός συστήματος ηλεκτρονικής ψηφοφορίας οι οποίες και παρουσιάζονται παρακάτω.

8.2 Χρήστες συστήματος

Οι Kevin Daimi, Katherine Snyder, and Robert James (94) προσδιορίζουν τα παρακάτω είδη πιθανών χρηστών του συστήματος:

Έμπειροι χρήστες: καθημερινοί χρήστες του διαδικτύου, με χρήση του διαδικτύου για ηλεκτρονικές συναλλαγές που δεν αναμένεται να έχουν κάποιο πρόβλημα με τη χρήση ενός συστήματος ηλεκτρονικής ψηφοφορίας

Τακτικοί χρήστες: περιλαμβάνει τους χρήστες που χρησιμοποιούν καθημερινά το διαδίκτυο για εργασίες ρουτίνας. Τα προβλήματα που θα αντιμετωπίσουν, αναμένονται να είναι περιορισμένα. Ωστόσο, θα χρειαστούν οδηγίες – καθοδήγηση κατά τη χρήση του συστήματος

Άπειροι χρήστες: άτομα που χρησιμοποιούν ελάχιστα ή καθόλου το διαδίκτυο. Θα χρειαστούν μεγαλύτερη βοήθεια στη χρήση του συστήματος και ανήκουν κατά κύριο λόγο στη 3η ηλικία

Κυβερνητικοί χρήστες: θα χρησιμοποιήσουν τις λειτουργίες καταμέτρησης και αποθήκευσης των αποτελεσμάτων της ψηφοφορίας. Επίσης, θα είναι υπεύθυνοι για τη δημιουργία των ψηφοδελτίων που θα χρησιμοποιήσουν οι χρήστες του συστήματος ηλεκτρονικής ψηφοφορίας

Τεχνικοί χρήστες: θα είναι υπεύθυνοι για τη σωστή λειτουργία του συστήματος και την αντιμετώπιση οποιουδήποτε προβλήματος προκύψει, κατά τη λειτουργία του συστήματος σε θέματα υλικού, λογισμικού ή ασφαλείας . Δεν θα έχουν πρόσβαση στα αποτελέσματα της ψηφοφορίας

Χρήστες ασφάλειας υπολογιστών και δικτύου: αυτή η ομάδα χρηστών είναι υπεύθυνη για την ασφάλεια του συστήματος σε θέματα υλικού, λογισμικού, δικτύου και σε φυσικό επίπεδο κατά τη διάρκεια διεξαγωγής των εκλογών.

8.3 Προβλήματα λειτουργίας συστήματος ηλεκτρονικής ψηφοφορίας

Ένα σύστημα ηλεκτρονικής ψηφοφορίας έχει να αντιμετωπίσει αρκετά θέματα – προβλήματα, που αναφέρονται παρακάτω (94):

- **Απόρρητο των ψηφοφόρων:** δε θα πρέπει να είναι γνωστό ποιος ψήφισε τι
- **Ταυτότητας των ψηφοφόρων:** οι ψηφοφόροι θα πρέπει να είναι αυτοί που ισχυρίζονται ότι είναι

- **Επαληθευσιμότητα των ψήφων:** δυνατότητα εσωτερικής παρακολούθησης των ψήφων, όλοι όσοι ψηφίζουν θα πρέπει να είναι εγγεγραμμένοι στους εκλογικούς καταλόγους
- **Ακρίβεια της συμμετοχής των ψηφοφόρων:** παρακολούθηση της διαδικασίας ψηφοφορίας, με τα δεδομένα συμμετοχής διαθέσιμα κάθε στιγμή
- **Ασφάλεια δικτύου:** ασφαλής μεταφορά των ψήφων από τον υπολογιστή του ψηφοφόρου στον κεντρικό διακομιστή
- **Ασφάλεια της ψήφου:** Η ασφάλεια κατά την εγγραφή του ψηφοφόρου και της ψήφου του πολίτη
- **Μοναδικότητα της ψήφου:** Ένα άτομο μπορεί να ψηφίσει μια μόνο φορά
- Δυνατότητα **υποβολής ψήφου και για άλλους ψηφοφόρους**, χωρίς υπέρβαση του αριθμού ψήφων
- **Άδεια κάλπη** κατά την έναρξη της ψηφοφορίας
- **Επαλήθευση της ψήφου**, πριν την οριστική αποστολή της ψήφου
- Δυνατότητα **τροποποίησης της ψήφου**, πριν την οριστική υποβολή, όσες φορές χρειαστεί
- Τα **εγχειρίδια λειτουργίας** του συστήματος θα πρέπει να παρέχονται στους ψηφοφόρους αρκετές ημέρες πριν από την εκλογή
- Η **δοκιμαστική έκδοση** θα πρέπει να είναι διαθέσιμη αρκετές ημέρες πριν από τις εκλογές
- **Ασφάλεια** σε όλες τις λειτουργίες του κεντρικού διακομιστή

Η επιτυχή αντιμετώπιση – διαχείριση των παραπάνω θεμάτων, θα διασφαλίσει την επιτυχία ενός συστήματος ηλεκτρονικής ψηφοφορίας.

8.4 Λειτουργικές απαιτήσεις

Οι βασικές λειτουργικές απαιτήσεις από ένα σύστημα ηλεκτρονικής ψηφοφορίας είναι οι εξής (94):

- Το σύστημα πρέπει να παρέχει στους ψηφοφόρους **ακριβή στοιχεία**

- **Αναφορές** για την τρέχουσα / ζωντανή κατάσταση ψήφων
- Το σύστημα θα πρέπει να επιτρέπει τη χρήση των **διαθέσιμων εργαλείων** στο διαδίκτυο
- Θα πρέπει να συμμορφώνονται με τις απαιτήσεις της κυβέρνησης και την ισχύουσα **νομοθεσία**
- **Ευκολία χρήσης**
- Ανάπτυξη σύμφωνα με **διεθνή πρότυπα**, προκειμένου να μπορεί να λειτουργήσει και σε μελλοντικές εκδόσεις λειτουργικών συστημάτων
- Συνοδευτικά **εγχειρίδια χρήσης** και διαθεσιμότητα **βοήθειας online**
- Διατήρηση **αντιγράφων ασφαλείας**, με δυνατότητα επαναφοράς
- Ζωντανή ενημέρωση του διαχειριστή του συστήματος για την **πορεία της ψηφοφορίας**
- **Ζωντανή ενημέρωση** του διαχειριστή αν παρουσιάζονται προβλήματα σε ορισμένους σταθμούς εργασίας
- **Τυποποιημένες αναφορές** για τη λήψη αποφάσεων
- **Διατήρηση αρχείου αλλαγών** στη βάση δεδομένων (ποιος έκανε τι)
- Δυνατότητα **αναβάθμισης** συστήματος
- **Έλεγχος τοποθεσίας και ταυτότητας** χρηστών
- Αυτόματος **έλεγχος για πιθανά σφάλματα**
- **Έλεγχος ακεραιότητας** αποθηκευμένων δεδομένων

8.5 Μη Λειτουργικές Απαιτήσεις

Οι μη λειτουργικές απαιτήσεις για ένα σύστημα ηλεκτρονικής ψηφοφορίας είναι οι παρακάτω (94):

- Αποδεκτός **χρόνος απόκρισης και επεξεργασίας** του συστήματος
- Λάθη στο **τοπικό αρχείο βάσης δεδομένων** ψήφου λιγότερα από 0,0001%

- Λάθη που περιέχονται στο **διακομιστή** συλλογής λιγότερα από 0.0001%
- Λάθη στη **βάση δεδομένων** λιγότερα από 0,0001%
- Κατά τον έλεγχο για πιθανά σφάλματα στη βάση δεδομένων, θα πρέπει να **ελέγχεται ολόκληρη η βάση δεδομένων** και όχι κάποιο δείγμα
- **Διαθεσιμότητα** 100% τη μέρα διεξαγωγής των εκλογών
- Μεταφορά των τωρινών και μελλοντικών δεδομένων σε Κέντρο Διαχείρισης Δεδομένων Ψηφοφορίας (**διασύνδεση** με άλλα συστήματα)
- Σε έναν αριθμό ψήφων 10 εκατομμύρια, δεν επιτρέπονται πάνω από 33 λάθη (**six sigma**)
- Δυνατότητα προσθήκης ψηφοφόρων για μεγαλύτερο **ποσοστό σύνδεσης**

8.6 Απαιτήσεις Ασφαλείας

Η ασφάλεια αποτελεί μια μεγάλη πρόκληση για τα συστήματα ηλεκτρονικής ψηφοφορίας. Πολλοί είναι αυτοί που θα είχαν κίνητρο να σπάσουν την ασφάλεια ενός συστήματος ηλεκτρονικής ψηφοφορίας (95). Συνεπώς, η **ασφάλεια αποτελεί βασικό παράγοντα επιτυχίας ενός συστήματος ηλεκτρονικής ψηφοφορίας.**

Μερικές από τις βασικότερες απαιτήσεις ασφαλείας σε σύστημα ηλεκτρονικής ψηφοφορίας είναι (94):

- Το εκλογικό σύστημα πρέπει να περιλαμβάνει ελέγχους για την πρόληψη σκόπιμης ή τυχαίας προσπάθειες αντικατάστασης (**injection attack**) κώδικα
- Το σύστημα θα πρέπει να έχει **μηδενική ανοχή** σε παρεμβάσεις
- Δε θα πρέπει να είναι δυνατή η παρέμβαση στα αποτελέσματα των εκλογών, **ούτε σε μια ψήφο**
- **Ακριβείς ρυθμίσεις** ώρας και ημερομηνίας εμφάνισης και λειτουργίας συστήματος

- Το σύστημα **δεν θα πρέπει να επιτρέψει μη προβλεπόμενες δράσεις** τόσο από τους ψηφοφόρους όσο και τους υπαλλήλους που διαχειρίζονται τη διεξαγωγή των εκλογών

- Δε θα είναι δυνατό το **φιλτράρισμα** και εμφάνιση αποτελεσμάτων **κατανομής ψήφων** σε τοπικό επίπεδο από τις πολιτικές παρατάξεις

- Το σύστημα θα πρέπει να παρέχει τα μέσα για την **προστασία** και την **αποτροπή δευτερογενούς καταμέτρησης των ψήφων**

8.7 Μοντέλο Οντοτήτων Σχέσεων

Οι οντότητες που συμμετέχουν στο σύστημα ηλεκτρονικής ψηφοφορίας είναι οι εξής:

Ψηφοφόρος: συμμετέχουν στις εκλογές με την κατάθεση της ψήφου τους. Ένα ψηφοφόρος μπορεί ψηφίσει το πολύ μια φορά σε μια εκλογική αναμέτρηση.

Εκλογές: η οντότητα των εκλογών, περιέχει στοιχεία για την εκλογική αναμέτρηση. Οι ψηφοφόροι συμμετέχουν - ρίχνουν τη ψήφο τους για κάθε εκλογική αναμέτρηση

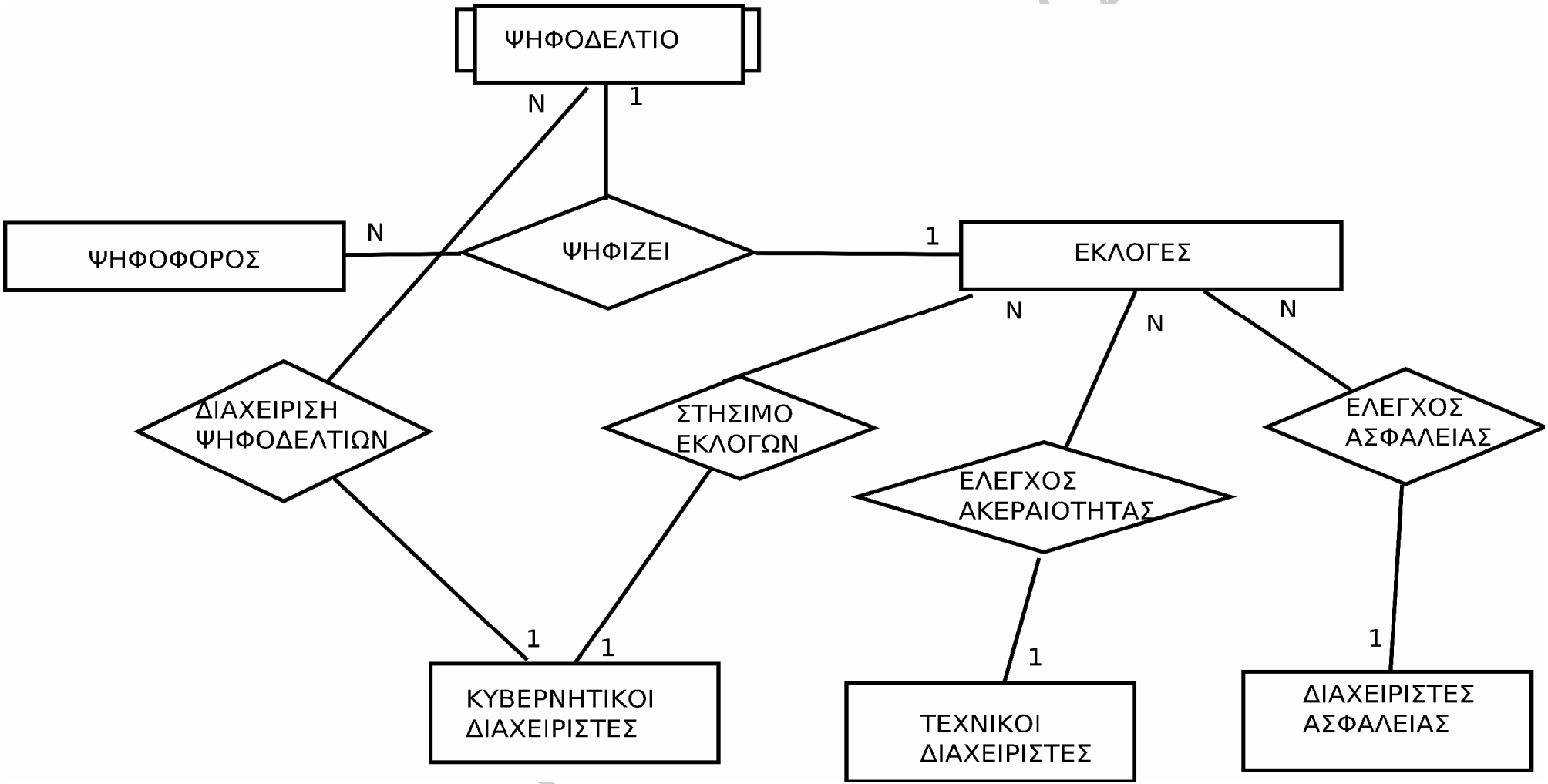
Ψήφος (ασθενής οντότητα)**:** συμμετέχει μαζί με τον ψηφοφόρο και το αντικείμενο εκλογές στην εκλογική διαδικασία. Υπάρχει ως οντότητα μόνο όταν ο ψηφοφόρος ψηφίσει σε μια εκλογική αναμέτρηση.

Κυβερνητικοί διαχειριστές: είναι τα άτομα που διαχειρίζονται τα ψηφοδέλτια (μπορούν να διαχειριστούν πολλά ψηφοδέλτια) και διαχειρίζονται τα αποτελέσματα των Ν εκλογικών διαδικασιών

Τεχνικοί Διαχειριστές: είναι υπεύθυνοι για την ορθή λειτουργία του συστήματος για κάθε εκλογική διαδικασία που μπορεί να λάβει χώρα και την ακεραιότητα των δεδομένων

Διαχειριστές ασφαλείας: είναι υπεύθυνοι για την ασφάλεια του συστήματος κατά τη διενέργεια εκλογικών διαδικασιών.

Στο Σχήμα 8.1, απεικονίζεται το μοντέλο οντοτήτων συσχετίσεων του συστήματος ηλεκτρονικής ψηφοφορίας.



Σχήμα 8.1: Μοντέλο Οντοτήτων Συσχετίσεων - μελέτη περίπτωσης

8.8 Διάγραμμα κλάσεων

Οι Kevin Daimi, Katherine Snyder, και Robert James (94) προσδιορίζουν τις λειτουργικές και μη λειτουργικές απαιτήσεις ενός συστήματος ηλεκτρονικής ψηφοφορίας καθώς και τα ζητήματα ασφαλείας που σχετίζονται με τη λειτουργία του. Δεν αναφέρονται όμως άμεσα στις οντότητες του συστήματος, τα χαρακτηριστικά τους και τις λειτουργίες του.

Ένα σύστημα ηλεκτρονική ψηφοφορίας θα πρέπει να περιλαμβάνει μια κλάση (**voter**) για τον ψηφοφόρο. Ο ψηφοφόρος θα προσδιορίζεται μοναδικά από το PIN του. Έχει πεδίο με την περιοχή (*state*) που είναι εγγεγραμμένος στο σύστημα. Η βασική λειτουργία για το αντικείμενο του ψηφοφόρου είναι η καταχώρηση ψήφου (vote). Μολονότι οι Kevin Daimi, Katherine Snyder και Robert James (94) αναφέρονται σε τρία (3) είδη χρηστών, δεν έχει νόημα να δημιουργηθούν διαφορετικές κλάσεις για κάθε είδος χρήστη (ανάλογα την ευχέρεια χρήσης διαδικτύου - υπολογιστή), καθώς η διαδικασία ψηφοφορίας θα είναι η ίδια, με παροχή περισσότερων οδηγιών στους λιγότερους έμπειρους χρήστες (λειτουργία του GUI). Επιπρόσθετες λειτουργίες για τον ψηφοφόρο είναι η είσοδος (login) και αποσύνδεση (logout). Ακόμα ο χρήστης θα μπορεί να βλέπει – επιλέγει την πολιτική παράταξη (selectParty) και τα άτομα (selectPeople) που θα ψηφίσει, να ζητάει βοήθεια (getHelp) όταν χρειάζεται και να διαβάζει τις οδηγίες χρήσης (readManual).

Η δεύτερη σημαντική κλάση για το σύστημα είναι οι ψηφοφορίες (**votingSystem**) που θα προσδιορίζονται μοναδικά από κάποιο ID. Θα έχει ως πεδία την περίοδο που θα είναι ενεργή η ψηφοφορία (startDate, endDate). Θα περιλαμβάνει ως λειτουργία την εμφάνιση των τελικών αποτελεσμάτων συνοπτικά (finalResults) και αναλυτικά (finalResultsInDetails). Επίσης, θα έχει μέθοδο για τη λήψη όλων των διαθέσιμων ψηφοδελτίων (getBallots) για εμφάνισή τους στους ψηφοφόρους.

Επίσης, θα πρέπει να υπάρχει μια κλάση (**vote**), για την καταγραφή των στοιχείων κάθε ψήφου. Θα προσδιορίζεται μοναδικά από το pin του χρήστη και το id της ψηφοφορίας. Σαν επιπλέον πληροφορίες μπορεί να περιέχει την ημερομηνία – ώρα αποστολής της ψήφου (votingDate) και τα στοιχεία της ψήφου: παράταξη (partyID) – άτομο (personID). Οι λειτουργία για τη κλάση vote περιλαμβάνουν τη δημιουργία κενού ψηφοδελτίου (createNew), την προσωρινή αποθήκευση της ψήφου (save), την οριστική αποστολή ψήφου

(send), την τροποποίηση προσωρινά αποθηκευμένης ψήφου (edit) και τη διαγραφή προσωρινά αποθηκευμένης ψήφου (delete).

Μια άλλη απαραίτητη οντότητα για το σύστημα είναι ο διαχειριστής των ψηφοφοριών (**administrator**) που προσδιορίζεται μοναδικά από το username και έχει ως επιπλέον πεδίο (απαραίτητο για τη σύνδεσή του στο σύστημα) κάποιο password. Η βασική λειτουργία του χρήστη administration είναι η διαχείριση των ψηφοφοριών (configureElections) που αναφέρεται σε αντικείμενα της κλάσης votingSystems.

Σύμφωνα με τους Kevin Daimi, Katherine Snyder καιRobert James (94) είναι τρία τα είδη διαχειριστών του συστήματος, τα οποία αντιπροσωπεύονται στο σύστημα από τρεις διαφορετικές κλάσεις που κληρονομούν από τη κλάση administrator:

- κυβερνητικοί χρήστες (**governmentAdmin**): έχει ως βασικές λειτουργίες τη διαχείριση – στήσιμο των εκλογών (configureElections), τη δημιουργία ψηφοδελτίων για τις εκλογές (setUpBallots) και τη καταμέτρηση ψήφων (countBallots).

- τεχνικοί χρήστες (**techAdmin**): υπεύθυνοι για την ακεραιότητα των δεδομένων. Βασικές λειτουργίες τους η λήψη αντιγράφων ασφαλείας (backUp), η επαναφορά συστήματος (restore) και ο έλεγχος ακεραιότητας των δεδομένων (checkConsistency).

- χρήστες διαχείρισης ασφάλειας (**securityAdmin**): υπεύθυνοι για την ασφάλεια του συστήματος. Δυο οι βασικές τους λειτουργίες η ρύθμιση των παραμέτρων ασφαλείας (setUpSecurity) του συστήματος και ο έλεγχος ασφάλειας συστήματος (checkSecurity).

Κάθε ψηφοδέλτιο αναπαρίσταται από την κλάση **ballot**. Περιλαμβάνει το πεδία partyID (κωδικός παράταξης) και μια λίστα (candidates) με τους υποψήφιους ανά παράταξη, ανάλογα με την τιμή του partyID. Οι βασικές της μέθοδοι είναι η δημιουργία ψηφοδελτίου(createNew), η τροποποίηση των στοιχείων του (update) και η διαγραφή του (delete).

Υπάρχουν δυο ακόμα κλάσεις, οι **politcalParty** και **candidate**, που αντιπροσωπεύουν αντίστοιχα τις πολιτικές παρατάξεις και τους υποψήφιους στις εκλογές. Η κλάση των πολιτικών παρατάξεων περιλαμβάνει δυο πεδία

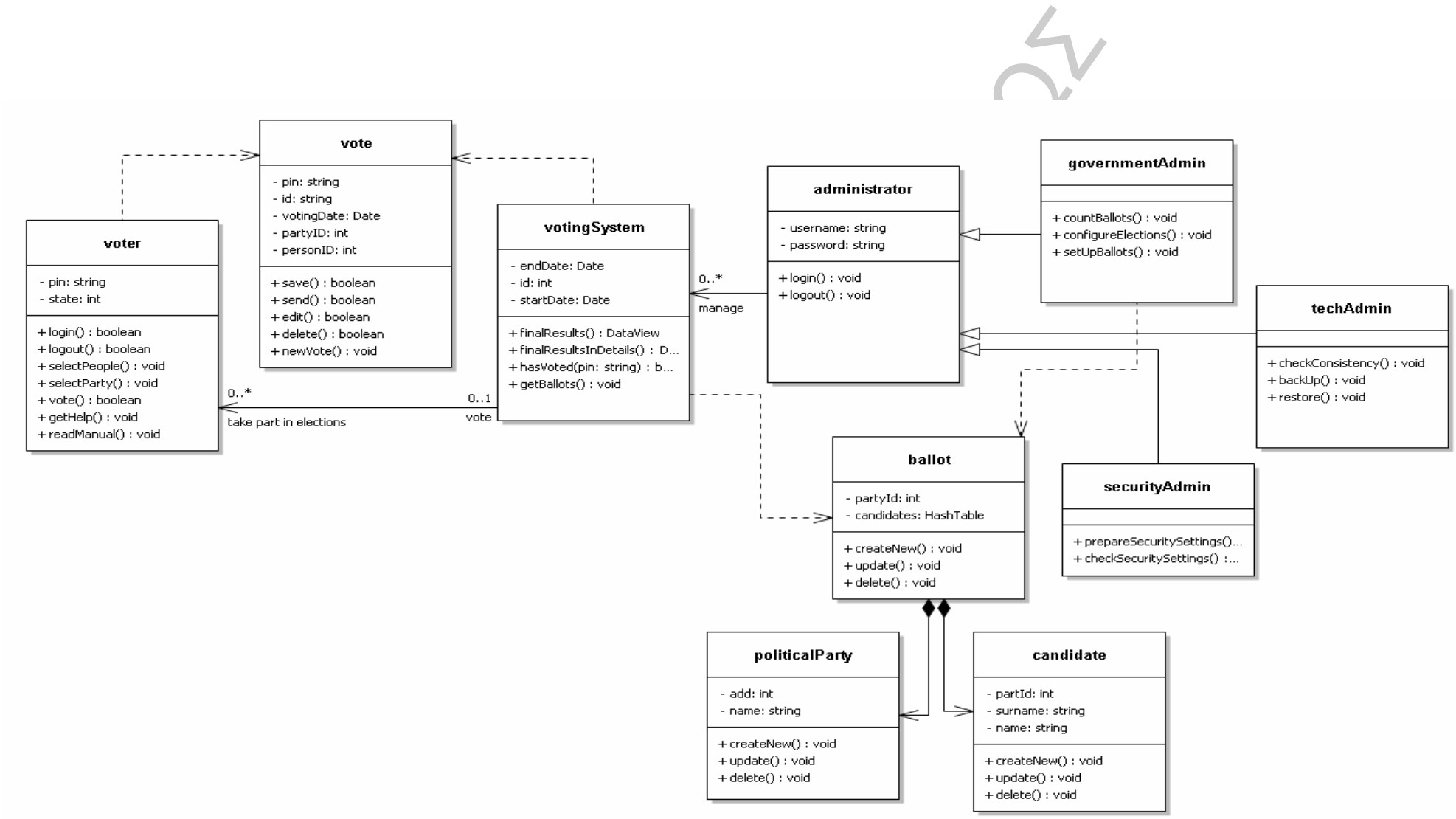
(partyId και onoma). Οι βασικές της λειτουργίες είναι η δημιουργία νέας παράταξης (cerateNew), η τροποποίηση (update) των στοιχείων της και η διαγραφή της (delete).

Ομοίως, η κλάση candidate, έχει ως πεδία τον κωδικό της παράταξης που ανήκει ο υποψήφιος (partyID) και το ονοματεπώνυμο (name, surname) του υποψηφίου. Οι μέθοδοι είναι ίδιοι με τη κλάση politicalParty: δημιουργία υποψήφιου (createNew), τροποποίηση (update), διαγραφή (delete).

Ένας ψηφοφόρος μπορεί να ψηφίσει σε περισσότερες από μια διαφορετικές εκλογικές διαδικασίες, με μια το πολύ ψήφο κάθε φορά.. Κάθε ψήφος, εκφράζεται από ένα αντικείμενο της κλάσης «vote», το οποίο δημιουργείται από τη κλάση «voter» και αποστέλλεται προς το αντικείμενο των αντίστοιχων εκλογών, κλάσης votingSystem.

Υπάρχουν τρία είδη διαχειριστών συστήματος (governmentAdmin, techAdmin, securityAdmin), που διαχειρίζονται με διαφορετικό τρόπο το αντικείμενο κλάσης «votingSystem» και κληρονομούν από τη κλάση «administrator» που περιέχει τις απαραίτητες μεθόδους διασύνδεσης κάθε είδους διαχειριστή στο σύστημα..

Στο Σχήμα 8.2, παρουσιάζεται το διάγραμμα κλάσεων για τη σύστημα ηλεκτρονικής ψηφοφορίας:



Σχήμα 8.2: Διάγραμμα κλάσεων - μελέτη περίπτωσης

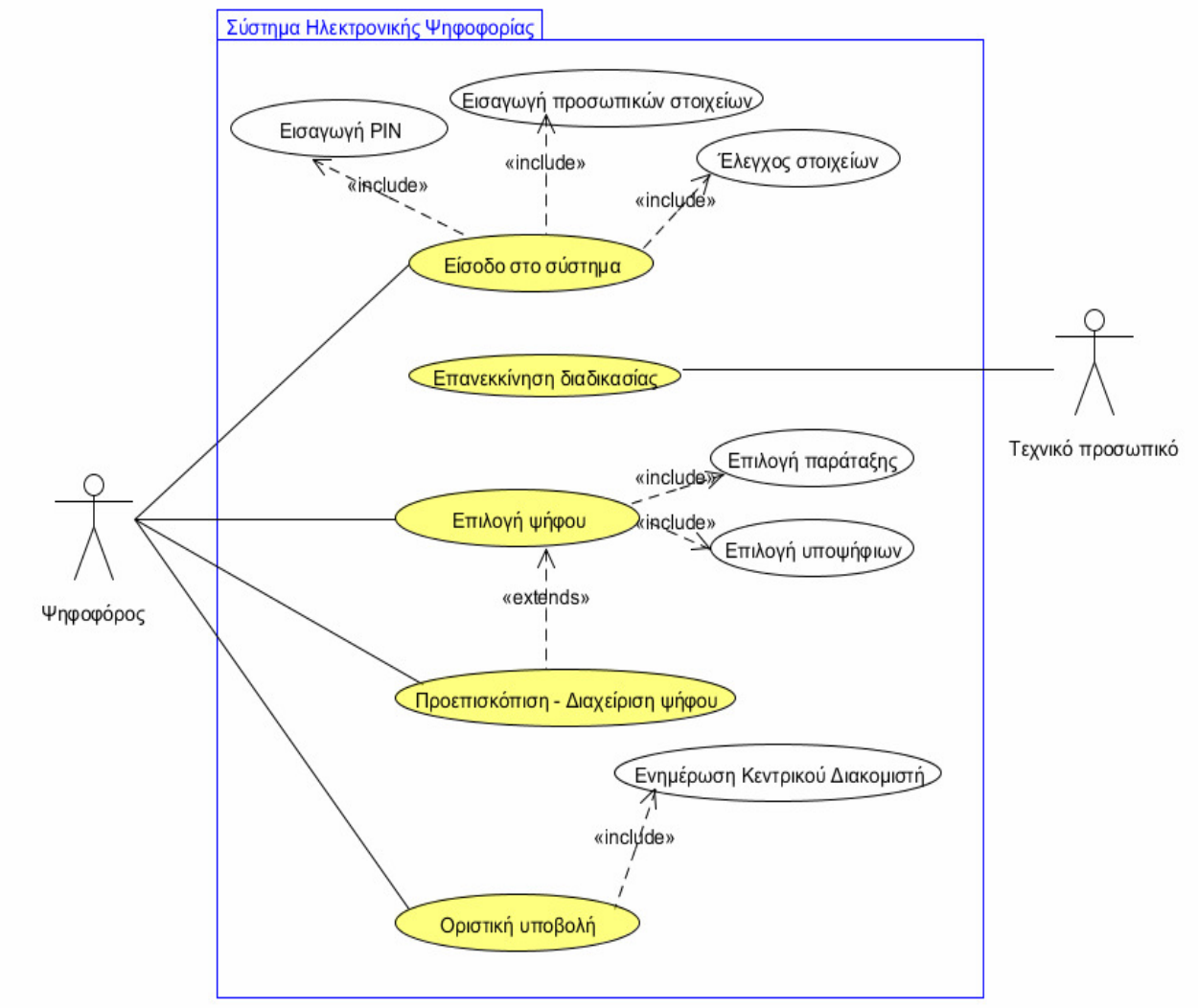
8.9 Περιπτώσεις Χρήσης

Δυο είναι οι βασικές Περιπτώσεις Χρήσης του συστήματος ηλεκτρονικής ψηφοφορίας που αναλύουν οι Kevin Daimi, Katherine Snyder, και Robert James (94), η ψηφοφορία από τους πολίτες και η ρύθμιση του συστήματος ψηφοφορίας από κυβερνητικούς παράγοντες.

Περίπτωσης Χρήσης:	**Ψηφοφορία**
Πρωτεύων χρήστης:	Κάθε πολίτης με δικαίωμα ψήφου
Δευτερεύων χρήστης:	Προσωπικό υποστήριξης διεξαγωγής εκλογών, για την επίλυση πιθανών δυσκολιών – απορριών κατά τη διαδικασία ηλεκτρονικής ψηφοφορίας
Σκοπός:	Η ασφαλής υποβολής ψήφου
Συχνότητα:	Εκτελείται, όσο υπάρχουν ψηφοφόροι που θέλουν να ψηφίσουν (ελάχιστο: μηδέν - μέγιστο: συνολικός αριθμός ψηφοφόρων)
Προαπαιτούμενα:	Ο πολίτης θα πρέπει να γνωρίζει και καταχωρίσει το προσωπικό του PIN, χωρίς αυτό δεν είναι δυνατή η σύνδεση με το σύστημα
Σενάριο:	1. ο ψηφοφόρος ανοίγει τον ιστότοπο για τη διεξαγωγή των εκλογών 2. επιλέγει τη πολιτεία – περιφέρεια που είναι καταχωρημένος 3. δυνατότητα από μέρους του χρήστη ανάγνωσης οδηγιών χρήσης του συστήματος 4. ο χρήστης εισάγει στο σύστημα το όνομα, αριθμό μητρώου, ημερομηνία γέννησης, φύλλο και κωδικό πολιτείας – περιφέρειας 5. αν τα στοιχεία που εισήγαγε ταιριάζουν με τη βάση δεδομένων του συστήματος επιτρέπετε στον πολίτη να συνεχίσει τη διαδικασία 6. ο ψηφοφόρος έχει τη δυνατότητα να επιλέξουν μία από τις δύο επιλογές: Επιλογή πολιτικής παράταξης ή επιλογή ατόμου 7. ο ψηφοφόρος ρίχνει τη ψήφο του προς την προτιμώμενη επιλογή τους στο πλαίσιο επιλογής

	8. ο ψηφοφόρος μεταβαίνει σε όλες τις σελίδες και ψηφίζει την επιλογή του σε κάθε διαθέσιμη κατηγορία ψήφου 9. εμφανίζεται στην οθόνη η επιλογή ψηφοφορίας του πολίτη 10. ο ψηφοφόρος μπορεί να αλλάξει τη ψήφο του όσες φορές επιθυμεί 11. αν ο ψηφοφόρος είναι σίγουρος για την επιλογή του ρίχνει την οριστική του ψήφο 12. αν φτάσει επιτυχώς η ψήφος – επιλογή του ψηφοφόρου στον κεντρικό διακομιστή, εμφανίζεται σχετικό μήνυμα επιτυχούς ολοκλήρωσης διαδικασίας στο χρήστη 13. ο ψηφοφόρος κάνει αποσύνδεση από το σύστημα
Εξαιρέσεις:	1. εισαγωγή λανθασμένων στοιχείων από τον ψηφοφόρο 2. ο ψηφοφόρος μπορεί να προσπαθήσει να επιλέξει περισσότερες από τις διαθέσιμες επιλογές ψήφου (πχ αριθμό σταυρών) 3. διακοπή σύνδεσης με τον κεντρικό διακομιστή πριν φτάσει η ψήφος επιτυχώς στον διακομιστή 4. διακοπή σύνδεσης με τον κεντρικό διακομιστή κατά τη διάρκεια χρήσης του συστήματος 5. μετά την επιτυχή αποστολή ψήφου, ο ψηφοφόρος πιθανόν να προσπαθήσει να ψηφίσει ξανά
Γεγονότα:	Αν ο ψηφοφόρος εισάγει τρεις φορές λανθασμένα τα στοιχεία του, δεν είναι πλέον δυνατή η σύνδεσή του στο σύστημα και απαιτείται επικοινωνία με την αρχή διεξαγωγής των εκλογών για επανεκκίνηση της διαδικασίας

Στο Σχήμα 8.3 απεικονίζεται η περίπτωση χρήσης «**Ψηφοφορία»**

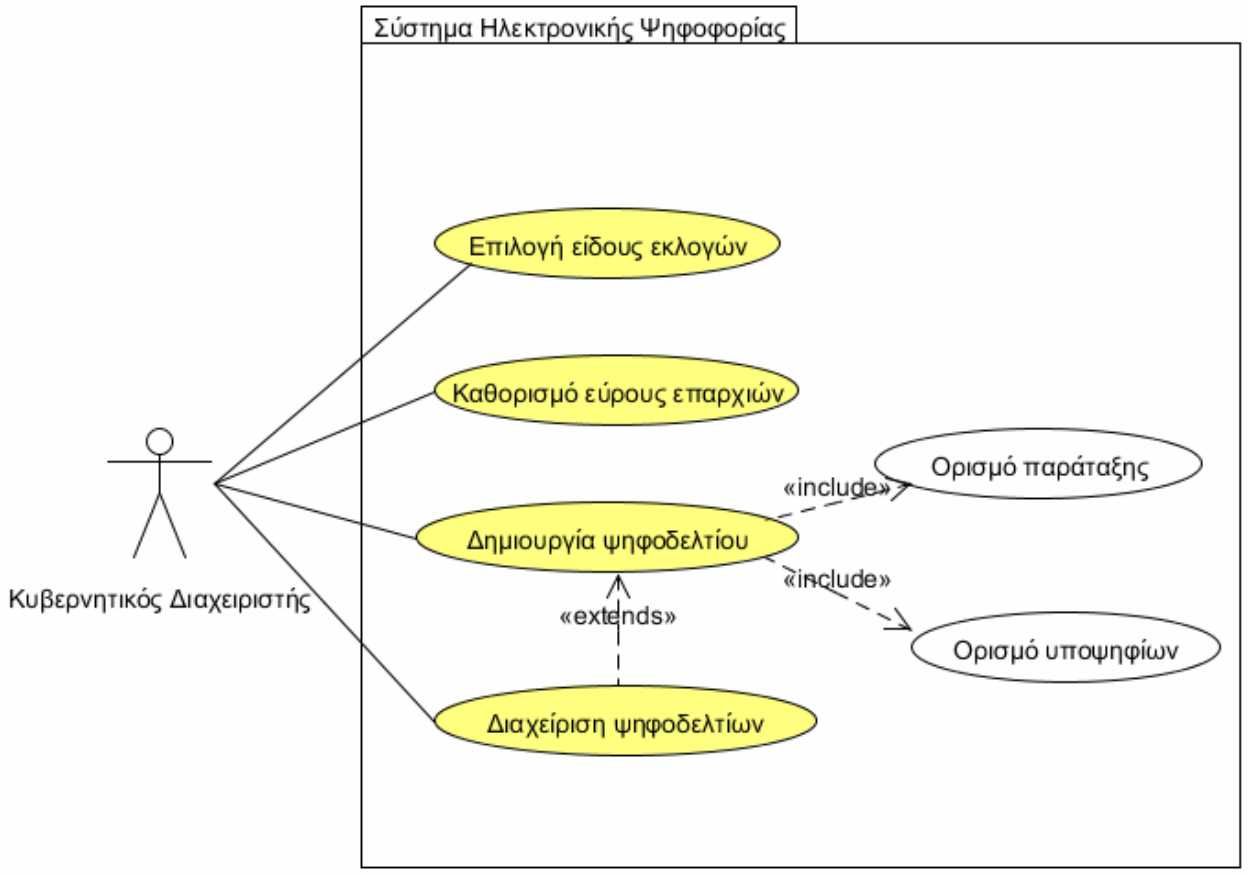


Σχήμα 8.3: Περίπτωση Χρήσης - Ψηφοφορία

Περίπτωσης Χρήσης:	**Ρύθμιση Ψηφοφορίας**
Πρωτεύων χρήστης:	Διαχειριστής (συνήθως ένα εξουσιοδοτημένο πρόσωπο της Εφορευτικής Επιτροπής).
Δευτερεύων χρήστης:	Προγραμματιστές
Σκοπός:	Εισαγωγή στο σύστημα των παρατάξεων και των υποψηφίων ανά παράταξη
Συχνότητα:	Συνήθως μια φορά
Προαπαιτούμενα:	Κανένα
Σενάριο:	1. ο χρήστης πατάει το κουμπί “Ρύθμιση” 2. ο χρήστης επιλέγει είτε "Απλή Ρύθμιση" ή "Πολλαπλή Ρύθμιση ανάλογα με το αν η εκλογές πραγματοποιηθούν σε μια επαρχία ή πολλές 3. αν επιλέξει "Πολλαπλή ρύθμιση" θα πρέπει να εισάγει τα στοιχεία των παρατάξεων ανά

	επαρχία 4. ο χρήστης δηλώνει τα κριτήρια βάσει των οποίων διακρίνονται οι επαρχίες 5. Ο χρήστης έχει τη δυνατότητα να προσθέσετε μια νέα παράταξη ή να επεξεργαστεί μια υπάρχουσα πατώντας τα αντίστοιχα κουμπιά 6. Το όνομα των παρατάξεων και του αριθμού των υποψηφίων για κάθε μια καταχωρούνται 7. Ο χρήστης κάνοντας κλικ στο κουμπί "Επόμενο" εισάγει το όνομα των υποψηφίων και τη παράταξη που ανήκουν
Εξαιρέσεις:	Καμία
Γεγονότα:	Κανένα

Στο Σχήμα 8.4 απεικονίζεται η περίπτωση χρήσης «**Ρύθμιση Ψηφοφορίας»**:



Σχήμα 8.4: Περίπτωση Χρήσης – Ρύθμιση Ψηφοφορίας

9 Συμπεράσματα– Προοπτικές

Η παρουσίαση των μοντέλων – σταδίων της Ανάλυσης Απαιτήσεων βασίστηκε στη διεθνή βιβλιογραφία. Αφού προσδιορίστηκαν τα στάδια που θα πρέπει να περιλαμβάνονται στη διαδικασία Ανάλυσης Απαιτήσεων και βοηθητικά εργαλεία διεκπεραίωσης μιας Ανάλυσης Απαιτήσεων, εξετάστηκε ο τρόπος εφαρμογής και χρήσης τους στη περίπτωση ανάπτυξης συστήματος ηλεκτρονικής ψηφοφορίας (94).

9.1 Συμπεράσματα της έρευνας

Η Ανάλυση Απαιτήσεων είναι το πρώτο στάδιο της ανάπτυξης ενός λογισμικού, ανεξαρτήτως του μοντέλου κύκλου ζωής λογισμικού που θα εφαρμοστεί. Στο στάδιο αυτό καθορίζονται οι προδιαγραφές - απαιτήσεις του λογισμικού που θα κατευθύνουν τη σχεδίαση και υλοποίηση του λογισμικού. Τελικό αποτέλεσμα της Ανάλυσης Απαιτήσεων είναι μια πρώτη εικόνα (σε σημαντικό βαθμό ολοκληρωμένη) του νέου λογισμικού.

Για την επιτυχή διεκπεραίωση της Ανάλυσης Απαιτήσεων απαιτείται η εφαρμογή των τεσσάρων βασικών της σταδίων:

1. Εξόρυξη Απαιτήσεων
2. Ανάλυση - Μοντελοποίηση Απαιτήσεων
3. Προσδιορισμός – Καταγραφή Απαιτήσεων
4. Επικύρωση Απαιτήσεων

Η απουσία κάποιου από τα παραπάνω, η μη ολοκληρωμένη υλοποίησή τους ή λάθη κατά την υλοποίησή τους με συνέπεια λανθασμένο προσδιορισμό απαιτήσεων, πιθανότατα θα οδηγήσει στην αποτυχία του έργου Πληροφορικής, όπως αναφέρει και η έκθεση του Standish Group(70).

Στη μελέτη περίπτωσης συστήματος ηλεκτρονικής ψηφοφορίας, έχει πραγματοποιηθεί από τους συγγραφείς, καταγραφή των απαιτήσεων του συστήματος. Με βάση τα αποτελέσματα της εξόρυξης απαιτήσεων, πραγματοποιήθηκε από μέρους μας η ανάλυση – μοντελοποίηση απαιτήσεων.

Ελλείψεις ή εσφαλμένα συμπεράσματα κατά την εξόρυξη απαιτήσεων θα έχουν ως συνέπεια λανθασμένη ανάλυση και μοντελοποίηση απαιτήσεων από μέρους μας.

Η υλοποίηση ενός συστήματος ηλεκτρονικής ψηφοφορίας χωρίς προηγούμενη ανάλυση απαιτήσεων, πιθανότατα δε θα λάμβανε υπόψη τους περιορισμούς ασφαλείας του συστήματος και θα είχε ως αποτέλεσμα μετέπειτα τροποποίηση – αναθεώρηση του συστήματος. Επίσης, η έλλειψη ανάλυσης, θα είχε ως πιθανότερη συνέπεια τη μη πρόβλεψη τριών ειδών διαχειριστών συστήματος (κυβερνητικοί, τεχνικοί υπεύθυνοι ασφαλείας), γεγονός που επίσης θα οδηγούσε σε μετέπειτα αναθεώρησή του.

Επιπροσθέτως, δε θα ήταν δυνατή η πρόβλεψη του τρόπου αντίδρασης του συστήματος σε κάθε σενάριο (πχ κλείδωμα σε περίπτωση τριών λανθασμένων εισαγωγών κωδικού σύνδεσης ψηφοφόρου, με ανάγκη επανεκκίνησης του συστήματος από το προσωπικό υποστήριξης του συστήματος). Η μη πρόβλεψη των πιθανών σεναρίων εκτέλεσης του λογισμικού, θα είχε ως αποτέλεσμα τις μετέπειτα τροποποιήσεις κώδικα.

Το αποτέλεσμα ελλιπούς ανάλυσης απαιτήσεις στην ανάπτυξη ενός έργου Πληροφορικής είναι η αύξηση του κόστους και του απαιτούμενου χρόνου υλοποίησής του (7). Αποτέλεσμα της αύξησης του κόστους και της αύξησης του χρόνου υλοποίησης ενός προγράμματος, είναι η υλοποίηση του έργου Πληροφορικής εκτός χρονοδιαγράμματος και εκτός προϋπολογισμού, με συνέπεια να μην είναι επιτυχής η υλοποίηση του νέου λογισμικού (91).

Ο κύριος λόγος υπέρ της υιοθέτησης μεθοδολογιών ταχείας ανάπτυξης λογισμικού, με περιορισμένη ανάλυση και διαχείριση απαιτήσεων είναι ο περιορισμός του κόστους ανάπτυξης και η ταχεία παράδοση ενός λογισμικού (81). Ωστόσο, το αποτέλεσμα της ελλιπούς ανάλυσης και διαχείρισης απαιτήσεων είναι η αύξηση του κόστους και καθυστερήσεις στη παράδοση ενός έργου πληροφορικής (70) (91). Το ζητούμενο είναι η ισορροπία – χρυσή τομή μεταξύ των δυο μεθοδολογιών.

9.2 Περιορισμοί στη μελέτη περίπτωσης

Πιθανή, **ελλιπής εξόρυξη απαιτήσεων** από μέρους των συγγραφέων (94) θα έχει ως αποτέλεσμα οι απαιτήσεις (λειτουργικές, μη λειτουργικές) του συστήματος που προσδιορίστηκαν να είναι λανθασμένες. Γεγονός, που θα είχε

ως συνέπεια όλα τα μετέπειτα στάδια να είναι ελλιπή ή λανθασμένα πχ η σχεδίαση του μοντέλου οντοτήτων – συσχετίσεων, του διαγράμματος κλάσεων και των περιπτώσεων χρήσης.

Μια αδυναμία που εντοπίστηκε στη μελέτη περίπτωσης, είναι η **έλλειψη προσδιορισμού προτεραιοτήτων** στη σειρά υλοποίησης των απαιτήσεων και στη σπουδαιότητα. Μερικά από τα οφέλη του προσδιορισμού προτεραιοτήτων είναι ο εντοπισμός των σημαντικότερων απαιτήσεων του συστήματος, η μετέπειτα ευκολία προγραμματισμού του χρονοδιαγράμματος υλοποίησης, η εκτίμηση του οφέλους από την υλοποίηση κάθε απαίτησης και μικρότερη πιθανότητα ακύρωσης του έργου – καθώς η εφαρμογή ικανοποιεί από τα πρώτα στάδια τις βασικές ανάγκες του πελάτη (96).

Επίσης, υπάρχει **έλλειψη χρονοδιαγράμματος** υλοποίησης για το σύστημα ηλεκτρονικής ψηφοφορίας, που πιθανόν μαρτυρά προχειρότητα στην ανάλυση απαιτήσεων. Η σύνταξη χρονοδιαγράμματος διευκολύνει στην αρχική εκτίμηση του απαιτούμενου χρόνου υλοποίησης και των πόρων που θα χρειαστούν, παρέχει μια γενική καθοδήγηση κατά την υλοποίηση του έργου, παρουσιάζει εναλλακτικές λύσεις κατά την υλοποίηση του συστήματος και μειώνει τις πιθανότητες αποτυχίας ενός έργου Πληροφορικής (97).

Τέλος, δεν υπάρχει **καμία πρόβλεψη για συμμετοχή των χρηστών – πολιτών** στην ανάπτυξη του συστήματος. Με τη συμμετοχή του τελικού χρήστη στη διαδικασία ανάλυση απαιτήσεων ενός έργου πληροφορικής υπάρχει μεγαλύτερη πιθανότητα για τον προσδιορισμό των πραγματικών αναγκών του χρήστη και τον προσδιορισμό των πραγματικών απαιτήσεων από το νέο σύστημα. Η συμμετοχή του χρήστη βοηθάει στην τελική επιτυχία του έργου και την ικανοποίηση του τελικού χρήστη από το τελικό αποτέλεσμα (98).

Ο προσδιορισμός προτεραιοτήτων κατά την ανάλυση απαιτήσεων, η ύπαρξη λεπτομερούς χρονοδιαγράμματος υλοποίησης των απαιτήσεων και η άμεση εμπλοκή των χρηστών στην υλοποίηση ενός έργου Πληροφορικής αποτελούν **κρίσιμους παράγοντες επιτυχίας** για μια ολοκληρωμένη ανάλυση – διαχείριση απαιτήσεων και την τελική ποιότητα ενός έργου πληροφορικής σύμφωνα με τους Nasir & Sahibuddin (91). Η έλλειψη των παραπάνω χαρακτηριστικών από την ανάλυση απαιτήσεων που χρησιμοποιήθηκε στη

μελέτη περίπτωσης πιθανών επηρεάζει την ποιότητα της μοντελοποίησης και καταγραφής των απαιτήσεων που έγινε.

9.3 Μελλοντικές επεκτάσεις

Το σύστημα ηλεκτρονικής ψηφοφορίας που μελετήθηκε στο 8ο κεφάλαιο θα μπορούσε να βελτιωθεί ακόμα περισσότερο, με την υιοθέτηση προτάσεων άλλων ερευνητών, σχετικά με τις λειτουργικές απαιτήσεις που θα πρέπει να καλύπτει ένα σύστημα ηλεκτρονικής ψηφοφορίας:

- Την εξασφάλιση της πρόσβασης στον μέγιστο αριθμό ψηφοφόρων, ιδίως όσον αφορά τα άτομα με ειδικές ανάγκες (99).
- Δυνατότητα εγγραφής στο σύστημα ακόμα και την ίδια μέρα της διεξαγωγής των εκλογών (100)
- Πολύγλωσση διεπαφή χρήστη (100)
- Όλες οι πιθανές επιλογές ψήφου θα πρέπει να εμφανίζονται σε μία μόνο οθόνη (100)
- Δυνατότητα επανάληψης καταμέτρησης ψήφων (100)
- Όλες οι ενέργειες των ψηφοφόρων και του συστήματος καταγράφονται και ελέγχονται (100)
- Το σύστημα δεν μπορεί να ρυθμιστεί εκ νέου κατά τη διάρκεια των εκλογών (100)
- Το σύστημα πρέπει να είναι ανοικτό - διαθέσιμο σε ανεξάρτητη παρακολούθηση και το έλεγχο (100)
- Εμφάνιση των ψηφοδελτίων σε τυχαία σειρά (101)

9.4 Προτάσεις για επόμενες έρευνες

Για ασφαλέστερα συμπεράσματα σχετικά με την ανάγκη και τα πλεονεκτήματα της ανάλυσης απαιτήσεων κατά την ανάπτυξη ενός συστήματος πληροφορικής θα βοηθούσε η **σύγκριση των αποτελεσμάτων** ανάπτυξης παρεμφερών συστημάτων από διαφορετικούς οργανισμούς πληροφορικής. Σύγκριση των αποτελεσμάτων της υλοποίησης παρεμφερούς λογισμικού με ολοκληρωμένη ανάλυση απαιτήσεων στη πρώτη περίπτωση και με μηδαμινή ανάλυση απαιτήσεων (rapid software development) στη δεύτερη περίπτωση.

Επιπροσθέτως, για τον προσδιορισμό του ιδανικού εύρους της ανάλυσης απαιτήσεων, θα ήταν χρήσιμη η διεξαγωγή έρευνας σχετικά με το επιθυμητό – ιδανικό μέγεθος που θα πρέπει να έχει η ανάλυση απαιτήσεων στην ανάπτυξη ενός έργου πληροφορικής. Η εύρεση δηλαδή της **ιδανικής αναλογίας - σχέσης** μεταξύ ανάλυσης απαιτήσεων και ταχείας ανάπτυξης λογισμικού.

Τέλος, θα ήταν χρήσιμη **διεξαγωγή έρευνας** συλλογής στοιχείων σχετικά με το μέγεθος και τη διάρκεια της ανάλυσης απαιτήσεων σε Ελληνικές επιχειρήσεις που δραστηριοποιούνται στον κλάδο Πληροφορικής. Η συλλογή των αντίστοιχων στοιχείων θα μπορούσε να γίνει είτε με τη χρήση ερωτηματολογίων, είτε με συνεντεύξεις του προσωπικού που εμπλέκεται στην ανάλυση απαιτήσεων. Παράλληλα, θα εξετάζονταν τα οικονομικά του.ς μεγέθη και η απόδοση των έργων Πληροφορικής που έχουν υλοποιήσει. Ο συνδυασμός και η σύγκριση των παραπάνω θα βοηθούσε στην εξαγωγή συμπερασμάτων σχετικά με τη αναγκαιότητα και σημασία της ανάλυσης απαιτήσεων στην ανάπτυξη συστημάτων Πληροφορικής.

Πιθανή συμφωνία στα αποτελέσματα των ερευνών που αναφέρονται παραπάνω, θα ενίσχυε την εγκυρότητα για τα αποτελέσματα των ερευνών.

Βιβλιογραφία

1. Laplante. What Every Engineer Should Know about Software Engineering: CRC Press; 2007.
2. Young R. The Requirements Engineering Handbook. Boston: Massachusetts: Artech House Publishers; 2004.
3. . MAGA. A Comparison Between Five Models Of Software Engineering. IJCSI International Journal of Computer Science Issues. 2010;7(5):94-101.
4. Nuseibeh BE, S., editor Requirements engineering: a roadmap. ICSE '00 Proceedings of the Conference on The Future of Software 2000; New York, NY, USA.
5. Bray I. An Introduction to Requirements Engineering: Addison-Wesley; 2002.
6. Aurum AW, C. Engineering and managing software requirements: Springer; 2005.
7. Faulk S. Software Requirements: A Tutorial in Software Engineering. IEEE Computer Soc Press. 1997:82–101.
8. Sallis P, Tate,G. & McDonell, S. Software Engineering.: Addison Wesley Publishing Co. ; 1996.
9. E. K. Requirements Engineering. Course-work Masters, The University of Melbourne2003.
10. Davis AM. Software Requirements: Objects, Functions and States.: Prentice Hall; 1993.
11. Pohl K. Requirements engineering: fundamentals, principles, and techniques. Berlin Heidelberg: Springer-Verlag 2010.
12. Wiegers K. Software Requirements. 2nd, editor: Microsoft Press; 2003.
13. Representatives USHo. Systems Development Life-Cycle Policy. 1999.
14. Pfleeger SL. Software Engineering: Theory and Practice.: Prentice-Hall; 1998.
15. Galin D. Software quality assurance: from theory to implementation. Harlow, Essex, UK: Addison- Wesley; 2004.
16. Ince D. Introduction to Software Quality Assurance and Its Implementation. New York, NY, USA: McGraw-Hill; 1995.
17. Glass R, Vessey I.& Ramesh V. Research in software engineering: an analysis of the literature. Information and Software Technology. 2000;44(8):491-506.
18. Royce W. Managing the development of large software systems. IEEE WESCON1970. p. 1-9.
19. Βεσκούκης Β. Σημειώσεις μαθήματος Τεχνολογία Λογισμικού. Πανεπιστήμιο Πειραιώς; 2000.
20. Boehm B. A spiral model for software development and enhancement. Computer. 1998;21(5):440–54.
21. Αρμάρα ΚΙ. Ανάπτυξη Μεθοδολογίας Διαχείρισης και Υποστήριξης Πληροφοριακών Συστημάτων - Έργων. Διπλωματική Εργασία - Εθνικό Μετσόβιο Πολυτεχνείο2006.
22. Research USDoHaHS-e-E. RUP Fundamentals Presentation. 2013.
23. Giunchiglia FT, A. . Students note on Software Engineering. Software Development Process: University of Trento; 2005.
24. Ruparelia NB. Software Development Lifecycle Models. ACM SIGSOFT Software Engineering Notes. 2010;35(3):8-13.

25. Doerr J, Paech, B. & Koehler, M., editor Requirements Engineering Process Improvement Based on an Information Model. IEEE Computer Society Press; 2004; Los Alamitos.
26. Sannier N. Eliciting Requirements - Specifying Requirements: Universitaire de Beaulieu, Students notes; [cited 2013]. Available from: http://nicolassannier.files.wordpress.com/2011/04/2-eliciting-requirements-specifying-requirements.pdf.
27. Zowghi DC, C. . Requirements Elicitation: A Survey of Techniques, Approaches and Tools. In: Wohlin AAaC, editor. Engineering and Managing Software Requirements. Berlin: Spriner-Verlag; 2005. p. 19-41.
28. Davis A, Dieste, O., Hickey, A., Juristo, N., & Moreno, AM, editor Effectiveness of Requirements Elicitation Techniques: Empirical Results derived from a Systematic Review. Requirements Engineering, 14th IEEE International Conference; 2006; Minneapolis/St. Paul, MN.
29. Kang K. Issues in Requirements Elicitation. Technical Report CMU/SEI–92–TR–12: Software Engineering Institute, Carnegie Mellon University; 1992.
30. Pressman R. Software Engineering: A Practitioner's Approach. 4th, editor: McGraw-Hill,; 1997.
31. Young R. Effective Requirements Practices: Effective Requirements Practices; 2001.
32. Goguen JAL, C., editor Techniques for Requirements Elicitation. International Symposium on Requirements Engineering; 1993.
33. Rumbaugh J, Jacobson, I. & Booch, G. An Introductionto Object-Oriented Analysis and Design with UML and The Unified Process

Addison- Wesley

1999.
34. I. AJN. UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design: Addison-Wesley; 2005.
35. Kulak DG, E. Use Cases – Requirements in Context. New York: Addison Wesley; 2000.
36. Starfield AM, Smith, K.A. & Bleloch, A.L. How to Model It: Problem Solving for the

Computer Age: McGraw-Hill Publishing Company; 1990.
37. Jerraya A, Romdhani, M., Le Marrec, Ph., Hessel, F., Coste, P., Valderrama, C., Marchioro, GF, Daveau, JM, Zergainoh, N.-E. Multilanguage Specification For System Design And Codesign. System Level Synthesis: NATO ASI; 1999.
38. Booch G, Rumbaugh, J. & Jacobson, I. The Unified Modelling Language User Guide: Addison-Wesley; 1999.
39. Cheesman J, & Daniels, J. UML Components : A Simple Process for Specifying Component-based Software: Addison-Wesley; 2001.
40. Pender T. UML Bible. New York: John Wiley & Sons, Inc.; 2003.
41. Rumbaugh J. Object-Oriented Modeling and Design.: Prentice-Hall; 1991.
42. Techrepublic. Five common errors in requirements analysis (and how to avoid them) 2007. Available from: http://www.techrepublic.com/article/five-common-errors-in-requirements-analysis-and-how-to-avoid-them/6146544.
43. Tran E. Requirements & Specifications: CMU; 1999. Available from: http://www.ece.cmu.edu/~koopman/des_s99~/requirements_specs/
44. IBM. Requirements definition and management - White paper, Getting requirements right: avoiding the top 10 traps: IBM 2009 [cited 2013]. Available from:

http://www.utdallas.edu/~chung/SYSM6309/Getting requirements right-avoiding the top 10 traps.pdf
45. Wiegers K. When Bad Requirements Happen to Nice People Business Analysis & Requirements Management Blog1999 [cited 2013]. Available from: http://blog.enfocussolutions.com/Powering Requirements Success/bid/111369/When-Bad-Requirements-Happen-to-Nice-People.
46. Harel D. Statecharts: A Visual Formalism for Complex Systems. Science of Computer Programming. 1987;8:231–74.
47. Vie DL. Writing software requirements specifications. JOURNAL OFOBJECT TECHNOLOGY. 2010;3(5):31-44.
48. Belfo F, editor People, Organizational and Technological Dimensions of Software Requirements Specification. 4th Conference of ENTERprise Information Systems – aligning technology, organizations and people (CENTERIS 2012); 2012.
49. Wyatt PR, M. GIS in Land and Property Management. London, UK: Taylor & Francis; 2003.
50. Kim HY, editor Validation of guidance control software requirements specification for reliability and fault-tolerance. Reliability and Maintainability Symposium; 2002; Seattle, WA.
51. Kazmierczak E, Dart, P., Winikoff, M. & Sterling, L. Verifying requirements through mathematical modeling and animation. Internation Journal of Knowledge Engineering and Software

Engineering. 2000;10(2):251-72.
52. Kirner TA, JC, editor Inspection of Software Requirements Specification Documents: A Pilot Study. 15th annual international conference on Computer documentation; 1997.
53. IEEE Std. 1028-1997 ISfSR, clause 3.8.
54. Transportation USDo. Systems Engineering Guidebook for Intelligent Transportation Systems. 3 ed2009.
55. Stellman AJ, G. . Applied Software Project Management: O'Reilly, Prentice Hall; 2005.
56. Mukasa KN, F. promise: A prototyping mantra for interactive systems engineering: Fraunhofer IESE; 2010.
57. Kordon FH, J. An overview of rapid system prototyping today. Design Automation for Embedded Systems,. 2003;8(4):275–82.
58. Leffingwell D, & Davis, A. . Using Requirements Management to Speed Delivery of Higher Quality Applications. Rational Software Corporation; 1996.
59. Sommerville I. Software Engineering. 8th, editor: Pearson Education; 2006.
60. Damasio A. The Feeling of What Happens: Body and Emotion in the Making of Consciousness. New York: Harcourt Brace; 1999.
61. Ramos I, Berry, DM. Is emotion relevant to requirements engineering? Requirements Engineering. 2005;10(3):238–42.
62. McLellan JM, Morkos, B., Mocko, G. G., Summers, J. D. , editor Requirement Modeling Systems for Mechanical Design: A Systematic Method for Evaluating Requirement Management Tools and Languages. ASME Design Engineering Technical Conference; 2010.
63. Grady JO. System Verification: Proving the Design Solutions Satisfies the Requirements Elsevier. New York: New York; 2007.
64. Nuseibeh BE, S. Requirements Engineering: A Roadmap. ACM New York, NY, USA. 2000:35-46.

65. Schwaber CS, P. Selecting The Right Requirements Management Tool - Or Maybe None Whatsoever: Forrester Research, Inc.; 2007.
66. Wiegers K. Automating Requirements Management. Software Development. 1999;7(7):S1-S5.
67. Hoffmann M, Kuhn N., Weber M., & Bittner, M., editor Requirements for requirements management tools. IEEE International Requirements Engineering Conference; 2004.
68. Shrivathsan M. 7 Requirements Management Tools, Practical Requirements Management 2012 [cited 2013]. Available from: http://www.accompa.com/requirements-management-blog/2012/05/requirements-management-tools/
69. Ramingwong L. A review of requirements engineering processes, problems and models. International Journal of Engineering Science and Technology. 2012;4(6):2997 – 3002.
70. Report TSG. Chaos. 1995.
71. Fernandez D. M.; Lochmann KP, B.; Wagner, S. , editor A case study on the application of an artefact-based requirements engineering approach. EASE'11, IET; 2011.
72. McConnell S. Rapid Development: Taming Wild Software Schedules. . Redmond, Wa: Microsoft Press; 1996.
73. John ID, J. . Elicitation of requirements from user documentation. REFSQ 2003;3:16-7.
74. Chua BBB, V.; Verner, J. M. Understanding the Use of Elicitation Approaches for Effective Requirements Gathering. ICSEA. 2010:325-30.
75. Ahmad S. Negotiation in the Requirements Elicitation and Analysis. Process ASEC 2008. 2008:683-9.
76. Chan K. Requirements Management Challenges Solved: OneDesk; 2013 [cited 2013]. Available from: http://www.onedesk.com/2013/04/requirements-management-challenges-solved/
77. Hass KB. What Are Business Analysts and Why Are They Needed? Professionalizing Business Analysis: Breaking the Cycle of Challenged Projects Management Concepts2008.
78. Desfray P. Making a success of preliminary analysis using UML: Softeam; 2004.
79. McPhee CE, A. , editor Requirements Engineering for Time-toMarket Projects. 9th Annual IEEE International Conference on the Engineering of Computer Based Systems 2002; Lund, Sweden.
80. Taylor A. IT Projects Sink or Swim. The Computer Bulletin. 2000:24-6.
81. Bose SK, M. & Ghoshal J. . Agile Methodology in Requirements Engineering. Infosys Research Publication; 2009.
82. Technologies S. Agile Methodologies Survey 2003. Available from: www.agilealliance.org/articles/reviews/ShineTechnologies1/articles/AgileSurvey2003.pdf
83. Kelly A. The Agile 10 Step Requirements Model 2011 [cited 2013]. Available from: http://www.allankelly.net/static/writing/overload/Agile10StepModel.pdf
84. Holz HM, F. . Knowledge Management Support for Distributed Agile Software Processes. 2003.
85. Fowler M. The New Methodology 2005 [cited 2013]. Available from: http://martinfowler.com/articles/newMethodology.html

86. Ambler SJ, R. Agile Modeling: Effective Practices for Extreme Programming and the Unified Process: Addison Wesley; 2002.
87. Alan MD. The Art of Requirements Triage. The Art of Requirements Triage. 2003;36(3):42-9.
88. Lahanas S. Don't Skip Requirements Management, Agile or Not 2013 [cited 2013]. Available from: http://slashdot.org/topic/bi/dont-skip-requirements-management-agile-or-not/
89. Fernández DM, Wagner, S., Lochmann, K., Baumann, A. & Carne, H Field study on requirements engineering: Investigation of artefacts, project parameters, and execution strategies. Information and Software Technology. 2012;54(2):162–78.
90. Gray CF LE. Project Management: The Managerial Process. 4th ed: Irwin/McGraw-Hill; 2008.
91. Nasir MS, S. . Critical success factors for software projects: A comparative study. Scientific Research and Essays. 2011;6(10):2174-86.
92. Baudron O, Fouque, P., Pointchevel, D., Stern, J. & G. Poupard, editor Practical Multi-Candidate Election System. The Twentieth Annual ACM Symposium on Principles of Distributed Computing; 2001; Rhode Island, USA,.
93. Development NMoLGaR. Electronic voting – challenges and opportunities. 2006.
94. Kevin Daimi KSRJ. Requirements Engineering for E-Voting Systems.
95. Rubin AD. Security Considerations for Remote Electronic Voting. CACM. 2002;45:39-44.
96. Firesmith D. Prioritizing Requirements. Journal of Object Technology. 2004;3(8):35-47.
97. Institute I. Software Project Scheduling under Uncertainties. Intaver Institute Inc; 2004.
98. Kujala S. User Involvement: A Review of the Benefits and Challenges. Behaviour & Information Technology. 2003;22(1):1–16.
99. Network AEK. Requirements for e-voting. 2013.
100. Brown J, Dickinson, D., Steinebach, C., Zhang, J. A Secure e-Voting System Project Requirements. CS 600443 Security and Privacy Spring 20032003.
101. Norwegian Government MoLG. The E-vote 2011 project

2009.