ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τομέας Ηλεκτρικών Βιομηχανικών Διατάξεων και Συστημάτων Αποφάσεων

Ανάλυση Τεχνικών Ανάπτυξης Καινοτόμων Προϊόντων Λογισμικού και Ανάπτυξη Μεθοδολογίας Εκμαίευσης και Περιγραφής Απαιτήσεων Χρήστη

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΔΗΜΗΤΡΙΟΥ ΠΙΧΛΙΑΒΑ

Επιβλέπων : **Δημήτριος Ασκούνης**
Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα,Οκτώβριος 2015

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ
ΔΙΑΤΑΞΕΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Ανάλυση Τεχνικών Ανάπτυξης Καινοτόμων Προϊόντων Λογισμικού και Ανάπτυξη Μεθοδολογίας Εκμαίευσης και Περιγραφής Απαιτήσεων Χρήστη

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΔΗΜΗΤΡΙΟΥ ΠΙΧΛΙΑΒΑ

Επιβλέπων : Δημήτριος Ασκούνης
Αναπληρωτής Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 15η Οκτωβρίου 2015.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Δημήτριος Ασκούνης	Ιωάννης Ψαράς	Βασίλειος Ασημακόπουλος
Αναπ.Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2015

.....

ΔΗΜΗΤΡΙΟΣ ΠΙΧΛΙΑΒΑΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Τα κλασικά μοντέλα ανάπτυξης εφαρμογών και υπηρεσιών πληροφορικής, όπως το «Waterfall model», είναι πλέον παρωχυμένα λόγω των σύνθετων αναγκών χρήσης και του έντονα ανταγωνιστικού περιβάλλοντος. Αντ' αυτού άλλα ευέλικτα μοντέλα, όπως το «Agile Development», αποκτούν ευρύτερης αποδοχής και πολλά σχετικά εργαλεία αναπτύσσονται πλέον για να υποστηρίζουν τις ομάδες ανάπτυξης προϊόντων. Προς αυτή την κατεύθυνση, πολλά εργαλεία προσπαθούν να καλύψουν το κενό που υπάρχει μεταξύ των ομάδων ανάπτυξης λογισμικού και των πραγματικών αναγκών των χρηστών.
Στόχος της διπλωματικής είναι να μελετήσει και να καταγράψει τις μεθόδους ανάπτυξης λογισμικού που χρησιμοποιούνται ακόμα ευρέως και τα αντίστοιχα εργαλεία υποστήριξης που χρησιμοποιούνται για επικοινωνία με τους υποψήφιους χρήστες και πελάτες. Αφού πραγματοποιηθεί μία ανάλυση των σχετικών εργαλείων, στόχος της παρούσας διπλωματικής εργασίας είναι να σχεδιαστεί ένα εργαλείο ανάπτυξης λογισμικού για καινοτόμα προϊόντα λογισμικού, να χρησιμοποιηθεί πειραματικά από μία ομάδα λογισμικού και τέλος να εξαχθούν συμπεράσματα για την επιτυχία του.
Λέξεις Κλειδιά: <<μοντέλα ανάπτυξης εφαρμογών και υπηρεσιών, διαδικασία συλλογής απαιτήσεων, διαδικαστικά μοντέλα, ευέλικτη ανάπτυξη, μοντελοποίηση απαιτήσεων, καινοτομία, εργαλείο εξαγωγής απαιτήσεων>>

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

The classic models of software development and IT services, such as «Waterfall model», is now old-fashioned due to complex user needs and the intensely competitive environment. Furthermore, new flexible models such as «Agile Development», gain wider acceptance as well as many related tools are now being developed to support the product development teams.
To this direction, many tools try to fill the gap that exists between software development teams and the real needs of users.
The scope of this thesis is to study and record the software development methods which are still widely used and the tools which support communication between prospective users and customers. After performing an analysis of these relevant tools, the scope of this thesis is to design a software development tool for innovative software products to be used experimentally by a software team and then draw conclusions about its success.

Keywords: << software development methods, requirement engineering, requirement elicitation techniques, agile development, innovation, requirement elicitation tool >>

Η σελίδα αυτή είναι σκόπιμα λευκή.

Ευχαριστίες

Με την ευκαιρία που μου δίνεται μέσω αυτής της διπλωματικής εργασίας, θα ήθελα να εκφράσω τις ευχαριστίες μου στον καθηγητή μου κ. Δημήτριο Ασκούνη για την εμπιστοσύνη που μου έδειξε για την ανάληψη της διπλωματικής εργασίας και την ευκαιρία που μου παρείχε να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα. Ακόμη, ευχαριστώ ιδιαίτερα τον υποψήφιο Διδάκτορα κ. Ιωσήφ Αλβέρτη για το χρόνο που αφιέρωσε για την καθοδηγησή μου σε όλη τη διάρκεια της εκπόνησης της παρούσας εργασίας και τις πολύτιμες συμβουλές που μου παρείχε για την επιτυχή ολοκλήρωσή της. Ευχαριστώ, επίσης, τους εργαζόμενους της νεοφυούς εταιρείας που έλαβαν μέρος στη διαδικασία της αξιολόγησης για τις ουσιαστικές απόψεις που μοιράστηκαν μαζί μου.

Τέλος, θα ήθελα να εκφράσω τις ευχαριστίες μου και την ευγνωμοσύνη μου στους γονείς μου, Πίχλιαβα Γεώργιο και Κωστούρου Μαρία, στον αδερφό μου, Πίχλιαβα Αλέξανδρο-Γεώργιο, και στους συμφοιτητές μου για την ηθική αλλά και υλική στήριξή τους σε όλη τη διάρκεια των σπουδών μου.

Πίνακας περιεχομένων

1 Εισαγωγή 1

1.1 Το πρόβλημα 2

1.2 Αντικείμενο διπλωματικής 8

1.2.1 Συνεισφορά 8

1.3 Οργάνωση κειμένου 9

2 Παρουσίαση, Ανάλυση και Αξιολόγηση Μοντέλων Ανάπτυξης Λογισμικού 11

2.1 Κατευθυντήρια μοντέλα ανάπτυξης λογισμικού 11

2.1.1 Μοντέλο Καταρράκτης (Waterfall Model) 12

2.1.2 V Μοντέλο 13

2.1.3 Προοδευτικά Εξελισσόμενο Μοντέλο (Incremental) 14

2.1.4 Μοντέλο Πρωτοτυποποίησης Προτύπου (Software Prototyping) 15

2.1.5 Το Σπειροειδές Μοντέλο (Spiral Model) 16

2.1.6 Παράλληλα Μοντέλα 18

2.2 Εξειδικευμένα Διαδικαστικά Μοντέλα 20

2.2.1 Ανάπτυξη που βασίζεται σε Συστατικά 20

2.2.2 Το Μοντέλο τυπικών μεθόδων 21

2.2.3 Ανάπτυξη λογισμικού βασισμένη σε πτυχές (Aspect-oriented development process) 22

2.2.4 Η Ενοποιημένη Διαδικασία (Unified Process) 23

2.3 Τεχνολογικά Διαδικαστικά Εργαλεία 26

2.3.1 Διαδικαστικά Εργαλεία Μοντελοποίησης 27

2.4 Ευέλικτη Ανάπτυξη (Agile Development) 28

2.4.1 Αρχές της Ευελιξίας (Principles of Agile Manifesto) 30

2.4.2 Έντονος Προγραμματισμός (XP: Extreme Programming) 33

2.4.3 Προσαρμόσιμη Ανάπτυξη Λογισμικού (ASD: Adaptive Software Development) 39

2.4.4 Μέθοδος Scrum 41

2.4.5 Δυναμική Μέθοδος Ανάπτυξης Συστημάτων 43

2.4.6 Μέθοδος Ανάπτυξης με βάση τα χαρακτηριστικά (Feature Driven Development -FDD) 45

2.4.7 Lean Ανάπτυξη Λογισμικού (Lean Software Development) 47

2.4.8 Ευέλικτη Ενοποιημένη Διαδικασία (AUP: Agile Unified Process) 48

2.5 Συμπεράσματα επί των Μεθόδων Ανάπτυξης Λογισμικού 49

2.5.1 Μοντέλο Καταρράκτης 49

2.5.2 Προοδευτικά Εξελισσόμενο Μοντέλο 52

2.5.3 Μοντέλο Πρωτοτυποποίησης Προτύπου (Prototyping) 54

2.5.4 Το Σπειροειδές Μοντέλο 57

2.5.5 Μοντέλα Ευέλικτης Ανάπτυξης 59

2.6 Μοντέλα Ανάπτυξης Λογισμικού και η σχέση τους με την Καινοτομία 62

2.7 Μοντέλα Ανάπτυξης Λογισμικού και η σχέση τους με Μεθόδους Διοίκησης Καινοτόμων Ομάδων 66

3 Απαιτήσεις Συστήματος Λογισμικού 74

3.1 Είδη Απαιτήσεων 74

3.2 Η διαδικασία της συλλογής απαιτήσεων 78

3.2.1 Εκμαίευση Απαιτήσεων (Requirements Elicitation) 81

3.2.2 Τεχνικές Εκμαίευσης Απαιτήσεων 84

3.2.3 Ανάλυση Απαιτήσεων 89

3.2.4 Προσδιορισμός Απαιτήσεων (Requirements Specification) 99

3.2.5 Επικύρωση Απαιτήσεων 105

3.3 Συμπεράσματα επι των μεθόδων εκμαίευσης απαιτήσεων 108

3.3.1 Ιστορίες Χρήσης και Περιπτώσεις Χρήσεις (User Stories Vs Use Cases) 108

3.3.2 Εκμαίευση απαιτήσεων σε νεοφυείς επιχειρήσεις 111

3.3.3 Συμπεράσματα τεχνικών εκμαίευσης απαιτήσεων 113

4 Σχετικές εργασίες σε καινοτόμα προϊόντα λογισμικού και προδιαγραφές 116

4.1 Σχεδίαση με επίκεντρο το χρήστη (User-Centered Design - UCD) 116

4.2 Πανταχού παρούσα υπολογιστική 121

4.3 Δομημένη μέθοδος σχεδίασης βασισμένη σε σενάρια χρήσης - (Structured Scenario-Based Design Method) 121

4.3.1 Διαδικασία σχεδιασμού scenarios 124

4.4 Μέθοδος σχεδίασης με επίκεντρο το όραμα - (Vision-Proposal Design Method) 125

4.4.1 Ιδιότητες σεναρίων κατά τη διαδικασία ανάπτυξης 126

4.4.2 Δομή σεναρίων 127

4.5 Μέθοδοι αυτοματοποίησης εξαγωγής απαιτήσεων 129

4.6 Συμπεράσματα επί των εργασιών σχετικών με την εκμαίευση απαιτήσεων σε καινοτόμα προϊόντα λογισμικού 131

5 Ανάπτυξη μεθοδολογίας συλλογής απαιτήσεων για ανάπτυξη καινοτόμων προϊόντων λογισμικού 133

5.1 Διαδικασία συλλογής και ανάλυσης πληροφοριών από την επιχείρηση ... 135

5.2 Διαδικασία συλλογής και ανάλυσης πληροφοριών από τους χρήστες 137

5.3 Ροή διαδικασίας εργασιών για τη συλλογή και ανάλυση πληροφοριών από τους χρήστες 141

5.4 Καθορισμός σχεδίου αξιολόγησης 142

5.5 Επικύρωση και αξιολόγηση της διαδικασίας εκμαίευσης απαιτήσεων 143

5.6 Μέθοδος εξαγωγής απαιτήσεων 144

5.7 Σύγκριση προτεινόμενου εργαλείου με αντίστοιχα εργαλεία στη βιβλιογραφία 145

6 Εφαρμογή προτεινόμενου εργαλείου εξαγωγής απαιτήσεων σε πραγματική εταιρεία και αξιολόγηση 149

6.1 Προφίλ Εταιρείας 149

6.2 Μεθοδολογία εκτέλεσης της αξιολόγησης 150
6.3 Ομάδα Καινοτομίας: Ποιοτικός τρόπος αξιολόγησης 151
6.3.1 Συμπεράσματα ποιοτικής αξιολόγησης 153
6.4 Ομάδα καινοτομίας: Ποσοτικός τρόπος αξιολόγησης 154
6.4.1 Αποτελέσματα ερωτηματολογίων 154
6.4.2 Συμπεράσματα ποσοτικής αξιολόγησης 163
6.5 Συνολική ανατροφοδότηση και προσαρμογή του προτεινόμενου εργαλείου εξαγωγής απαιτήσεων 164
6.6 Συνολική συνεισφορά της προτεινόμενης μεθόδου 165
7 Επίλογος 167
7.1 Σύνοψη και συμπεράσματα 167
7.2 Μελλοντικές επεκτάσεις 168
8 Παράρτημα Ι 170
8.1 Ερωτηματολόγιο Αξιολόγησης 170
9 Βιβλιογραφία 174

Πίνακας εικόνων

Εικόνα 1-1 Ποσοστά επιτυχίας Παραδοσιακών Μοντέλων Ανάπτυξης και Ευέλικτων Μοντέλων....7
Εικόνα 2-1 Το Μοντέλο Καταρράκτη....12
Εικόνα 2-2 Το V-μοντέλο....13
Εικόνα 2-3 Το Προοδευτικά Εξελισσόμενο Μοντέλο....15
Εικόνα 2-4 Μοντέλο Πρωτοτυποποίησης Προτύπου....16
Εικόνα 2-5 Σπειροειδές Μοντέλο....18
Εικόνα 2-6 Ένα στοιχείο παράλληλου διαδικαστικού μοντέλου....19
Εικόνα 2-7 Η Ενοποιημένη Διαδικασία....26
Εικόνα 2-8 Μοντέλα Ευέλικτης Ανάπτυξης....28
Εικόνα 2-9 Μοντέλο Έντονου Προγραμματισμού (XP)....35
Εικόνα 2-10 Προσαρμόσιμη Ανάπτυξη Λογισμικού....40
Εικόνα 2-11 Ροή διαδικασίας Scrum....43
Εικόνα 2-12 Μοντέλο ανάπτυξης που βασίζεται στα χαρακτηριστικά....46
Εικόνα 2-13 Είδη Καινοτομίας....63
Εικόνα 2-14 Η μέθοδος Customer Development....67
Εικόνα 2-15 Business Model Canvas....68
Εικόνα 2-16 Ο κύκλος διαδικασιών Lean Startup....69
Εικόνα 2-17 Lean Canvas....70
Εικόνα 2-18 Lean Analytics....71
Εικόνα 2-19 Open Product Management Workflow....73
Εικόνα 3-1 Είδη Απαιτήσεων....75
Εικόνα 3-2 Τύποι Μη-λειτουργικών Απαιτήσεων....77
Εικόνα 3-3 Η διαδικασία συλλογής απαιτήσεων....80
Εικόνα 3-4 Μία τυπική σύνταξη persona....92
Εικόνα 3-5 Το βασικό σχέδιο μιας persona....93
Εικόνα 3-6 Διάγραμμα ενος Use Case....98
Εικόνα 3-7 Το περίγραμμα ενός προκαταρκτικού εγχειριδίου χρήστη....103
Εικόνα 4-1 Βασικό μοντέλο Structured Scenario-based Design Method [51]....122
Εικόνα 4-2 Βασικό μοντέλο Vision-proposal Design Method [52]....126

Εικόνα 4-3 Συνεργατικό μοντέλο για την ανάπτυξη λογισμικού με χρήση κοινονικών μηχανών [54].....131
Εικόνα 5-1 Προτεινόμενα στάδια εκμαίευσης και εξαγωγής απαιτήσεων.....134
Εικόνα 5-2 Πεδία ενδιαφέροντος από το Business Model Canvas.....136
Εικόνα 5-3 Persona Canvas 137
Εικόνα 5-4 Ροή διαδικασίας συλλογής και ανάλυσης πληροφοριών από τους χρήστες 142
Εικόνα 5-5 Προτεινόμενο Μοντέλο Εξαγωγής Απαιτήσεων 144
Εικόνα 6-1 Στάδια διαδικασίας εφαρμογής και αξιολόγησης της προτεινόμενης μεθόδου 151
Εικόνα 6-2 Δημογραφικά χαρακτηριστικά ερωτηθέντων 155
Εικόνα 6-3 Εξοικείωση ερωτηθέντων με τις επικρατέστερες μεθόδους ανάπτυξης λογισμικού 156
Εικόνα 6-4 Προτιμήσεις ερωτηθέντων σχετικά με τη συχνότητα χρήσης μεθόδων ανάπτυξης λογισμικού 157
Εικόνα 6-5 Προτιμήσεις ερωτηθέντων σχετικά με διάφορες τεχνικές εκμαίευσης απαιτήσεων.....158
Εικόνα 6-6 Προτιμήσεις ερωτηθέντων σχετικά με διάφορες τεχνικές ανάλυσης απαιτήσεων.....158
Εικόνα 6-7 Εξοικείωση ερωτηθέντων με εργαλεία εξαγωγής απαιτήσεων.....159
Εικόνα 6-8 Συχνότητα χρήσης εργαλείων εξαγωγής απαιτήσεων 159
Εικόνα 6-9 Αξιολόγηση εργαλείου ως προς την καινοτομία.....161
Εικόνα 6-10 Αξιολόγηση εργαλείου ως προς τη χρησιμότητα.....161
Εικόνα 6-11 Αξιολόγηση εργαλείου ως προς την πληρότητα 162
Εικόνα 6-12 Αξιολόγηση εργαλείου ως προς την αποτελεσματικότητα.....162
Εικόνα 6-13 Ανάλυση επιθυμίας των ερωτηθέντων σχετικά με τη χρήση αντίστοιχων εργαλείων.....163
Εικόνα 6-14 Συνεισφορά της προτεινόμενης μεθόδου στον κύκλο ζωής ανάπτυξης προϊόντων λογισμικού 165

Πίνακας πινάκων

Πίνακας 1 Η παγκοσμία αγορά λογισμικού 2014 2
Πίνακας 2 Έσοδα αγοράς εφαρμογών προϊόντων λογισμικού για επιχειρήσεις 3
Πίνακας 3 Επενδύσεις σε προϊόντα λογισμικού για τις επιχειρήσεις από το 2009 έως και το 2014 3
Πίνακας 4 -Πρόβλεψη αύξησης επενδύσεων για το 2015 4
Πίνακας 5 Πλεονεκτήματα και Μειονεκτήματα Μοντέλου Καταρράκτη 51
Πίνακας 6 Πλεονεκτήματα και Μειονεκτήματα του Προοδευτικά Εξελισσόμενου Μοντέλου 53
Πίνακας 7 Πλεονεκτήματα και Μειονεκτήματα Μοντέλου Πρωτοτυποποίησης Προτύπου 56
Πίνακας 8 Πλεονεκτήματα και Μειονεκτήματα του Σπειροειδούς Μοντέλου 58
Πίνακας 9 Πλεονεκτήματα και Μειονεκτήματα των Μοντέλων Ευέλικτης Ανάπτυξης 61
Πίνακας 10 Μοντέλα Ανάπτυξης Λογισμικού, Εφαρμογές Χρήσης και Καινοτομία .. 66
Πίνακας 11 Συμπερασματικός πίνακας μεθόδων διοίκησης καινοτόμων ομάδων και μεθόδων ανάπτυξης λογισμικού 73
Πίνακας 12 Μετρικές Απαιτήσεων 78
Πίνακας 13 Συμπερασματικός πίνακας μεταξύ ιστοριών χρήσης και περιπτώσεων χρήσης 111
Πίνακας 14 Λόγοι επιλογής ευέλικτων μεθόδων ανάπτυξης λογισμικού από καινότομες ομάδες 113
Πίνακας 15 Συμπερασματικός πίνακας επί των τεχνικών εκμαίευσης απαιτήσεων 115
Πίνακας 16 Τεχνικές συμμετοχής χρηστών στην ανάπτυξη λογισμικού 121
Πίνακας 17 Συμπερασματικός Πίνακας Σεναριών - Vision Proposal Design Method [52] 129
Πίνακας 18 Επιχειρηματικά μοντέλα και αντίστοιχες μετρικές για κάθε στάδιο ανάπτυξης- Lean Analytics 143
Πίνακας 19 Συγκριτικός πίνακας προτεινόμενης μεθόδου και δομημένη μεθόδου σχεδίασης 147

1 *Εισαγωγή*

Η τεχνολογία λογισμικού αναφέρεται στην ανάπτυξη αξιόπιστων προϊόντων λογισμικού με μεγάλο κύκλο ζωής με στόχο την κάλυψη των απαιτήσεων των χρηστών και των πελατών. Εστιάζει στην ανάπτυξη και εφαρμογή συστηματικών μεθόδων, τεχνικών και εργαλείων που αφορούν ολόκληρο το κύκλο ζωής του προϊόντος και υποστηρίζουν την επιτυχία των παραπάνω στόχων.

Προκειμένου οι εταιρείες να διαχειριστούν τη συνεχώς αυξανόμενη ζήτηση για προϊόντα λογισμικού, δημιούργησαν μεθόδους διαχείρισης αυτών οι οποίες επικεντρώνονται στην ανάλυση και καταγραφή των αναγκών των χρηστών και στην παροχή προϊόντων τα οποία επιτυχώς ικανοποιούν αυτές τις ανάγκες. Ενώ η αγορά των προϊόντων λογισμικού έχει ωριμάσει, ακόμα και σήμερα οι υπάρχουσες μεθοδολογίες ανάπτυξης αλλά και διαχείρισης λογισμικού αποτυγχάνουν σε μεγάλο βαθμό. Η πιο κοινή αιτία αποτυχίας διαχείρησης έργων λογισμικού είναι η έλλειψη συμμετοχής των τελικών χρηστών στη διαδικασία ανάπτυξης καθώς και η ελλειπής επικοινωνία μεταξή πελατών, χρηστών και ομάδας έργου. Προκειμένου να λυθεί το προαναφερθέν πρόβλημα τα τελευταία χρόνια έχουν κάνει την εμφάνιση τους διάφορες ευέλικτες μεθοδολογίες ανάπτυξης προϊόντων λογισμικού (agile development processes) οι οποίες τυγχάνουν μεγάλης αποδοχής από την επιχειρηματική κοινότητα. Στα πλαίσια αυτής της συνεχόμενης αυξητικής τάσης επιλογής ευέλικτων μεθόδων ανάπτυξης, είναι αξιόλογο να προταθεί ένα εργαλείο υποστήριξης των συγκεκριμένων μεθόδων που στόχο έχει την σωστή και πλήρη αποσαφήνιση των αναγκών και επιθυμιών των χρηστών. Με βάση αυτό το πλαίσιο

και ανάλογα με τα ιδιαίτερα χαρακτηριστικά του αντίστοιχου προϊόντος λογισμικού θα είναι ευκολότερο να οριστούν και τελικά να ικανοποιηθούν οι ανάγκες των χρηστών και οι στόχοι της επιχείρησης που θα οδηγήσουν στην επίτευξη ενός επιτυχημένου προϊόντος το οποίο θα προσδίδει αξία και στον πελάτη-χρήστη και τελικά στην ίδια την επιχείρηση.

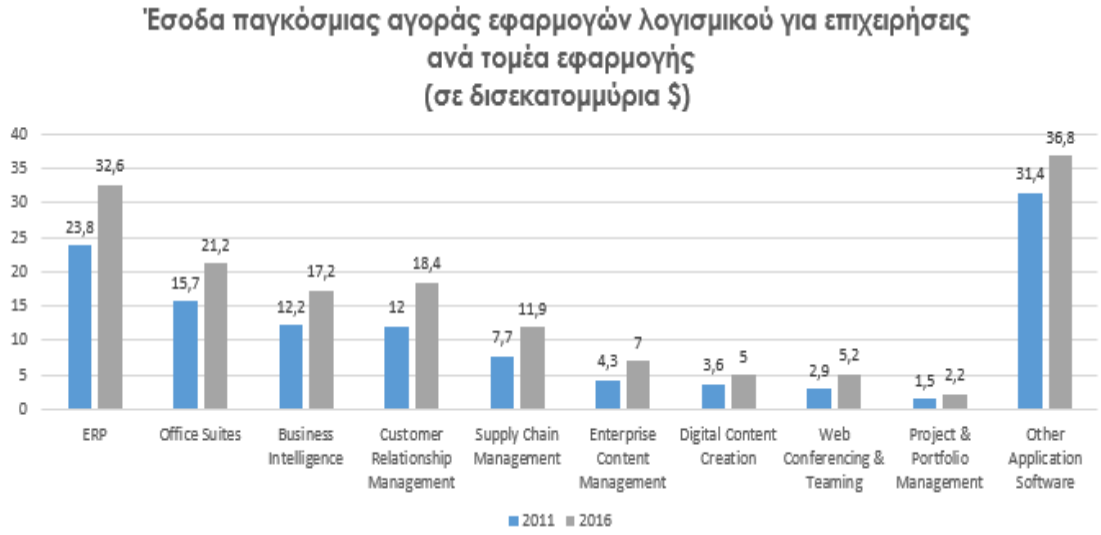
1.1 Το πρόβλημα

Η παγκοσμία αγορά των προϊόντων λογισμικού σύμφωνα με την εταιρεία Forrester Research το 2014 ανερχόταν στα 568 δισεκατομμύρια δολλάρια.Τα επιμέρους τμήματα της αγοράς παρουσιάζονται στον παρακάτω πίνακα.

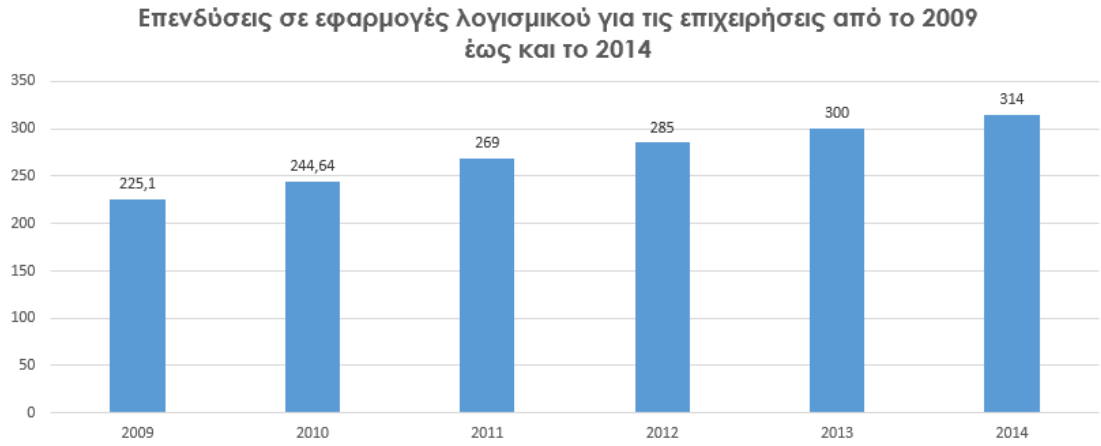
Πίνακας 1 Η παγκοσμία αγορά λογισμικού 2014[1]

[1] http://venturebeat.com/2014/01/02/global-tech-market-expected-to-grow-5-5-this-year-with-u-s-leading-the-way/

Πιο συγκεκριμένα, η αγορά λογισμικού τα τελευταία χρόνια παρουσιάζει μια συνεχώς αυξανόμενη τάση όχι μόνο όσον αφορά τις επενδύσεις στο συγκεκριμένο τομέα αλλά και τα έσοδα τα οποία παρουσιάζει.Όλα τα τμήματα της αγοράς αυξάνουν σταθερά τα μεγέθη τους όπως αυτό παρουσιάζεται και στους παρακάτω πίνακες σύμφωνα με την εταιρεία Statista καθώς και οι προβλέψεις ανάπτυξης για τα επόμενα έτη, γεγονός που οφείλεται σε μεγάλο βαθμό και στον μεγάλο αριθμό τεχνολογικών καινοτόμων startups που έχουν κάνει την εμφάνισή τους τα τελευταία χρόνια.



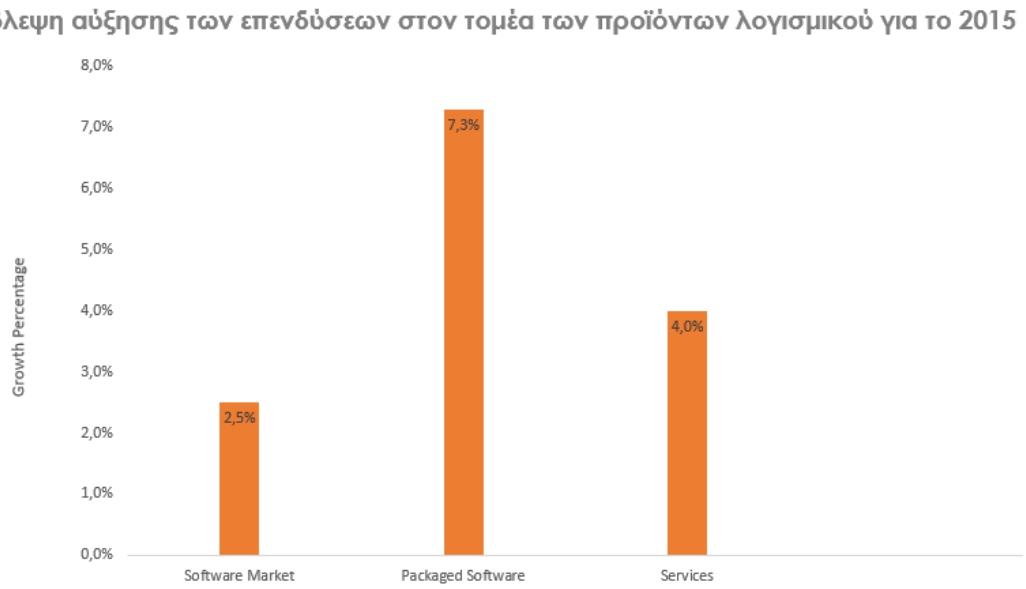
Πίνακας 2 Έσοδα αγοράς εφαρμογών προϊόντων λογισμικού για επιχειρήσεις[2]



Πίνακας 3 Επενδύσεις σε προϊόντα λογισμικού για τις επιχειρήσεις από το 2009 έως και το 2014[3]

[2] http://www.statista.com/statistics/

[3] http://www.statista.com/statistics/



Πίνακας 4 -Πρόβλεψη αύξησης επενδύσεων για το 2015[4]

Ενώ όμως τα ποσά τα οποία επενδύονται στη συγκεκριμένη αγορά αυξάνονται, τα προβλήματα τα οποία υπάρχουν και δυστυχώς δεν λύνονται με την πάροδο των χρόνων είναι αρκετά. Σύμφωνα με την εταιρία συμβουλευτικής Standish Group[5] σε έρευνα που έκανε το 2012 σχετικά με τις εταιρίες ανάπτυξης προϊόντων λογισμικού στις Η.Π.Α , ετησίως ξοδεύονται περισσότερα από 250 δισεκατομμυρία δολάρια για ανάπτυξη εφαρμογών και προϊόντων λογισμικού τα οποία αριθμούνται περίπου στα 175.000 έργα. Το μέσο κόστος ενός έργου για μια μεγάλη εταιρεία είναι $ 2.322.000, για μια μεσαία επιχείρηση είναι $ 1.331.000 και για μια μικρή εταιρεία, είναι $ 434.000. Τα ποσοστά των έργων τα οποία αποτυγχάνουν είναι εντυπωσιακά. Το 31,1% των έργων ακυρώνονται πριν καν ολοκληρωθούν. Περαιτέρω αποτελέσματα δείχνουν ότι το 52,7% των έργων κοστίζουν 189% των αρχικών εκτιμήσεών τους. Το κόστος αυτών των αποτυχιών και οι υπερβάσεις στην κοστόλογησή τους είναι μόνο η κορυφή του παγόβουνου. Τα κόστη ευκαιρίας δυστυχώς δεν μπορούν να ποσοτικοποηθούν αλλά σύμφωνα με τη Standish Group[5] εύκολα αγγίζουν τα τρισεκατομμύρια δολάρια. Ένα χαρακτηριστικό παράδειγμα του ανυπολόγιστου κόστους είναι η αποτυχία της πόλης του Denver να ανάπτυξει ένα λογισμικό το οποίο

[4] http://www.statista.com/statistics/440821/it-spending-growth-worldwide-by-segment/

[5] https://www.projectsmart.co.uk/white-papers/chaos-report.pdf

θα διαχειρίζεται τις αποσκευές στο νέο αεροδρόμιο του Denver με ημερίσιες απώλειες της τάξης των 1,1 εκατομμυρίων δολαρίων.

Με βάση αυτή την έρευνα, η Standish Group εκτιμά ότι το 2016 οι αμερικάνικες επιχειρήσεις θα δαπανήσουν περισσότερα από 90 δισεκατομμύρια δολλάρια για ακυρώσεις λογισμικού. Αυτές οι εταιρίες θα πληρώσουν επιπλέον 65 δισεκατομμύρια δολλάρια για έργα λογισμικού των οποίων το πραγματικό κόστος θα ξεπεράσει το προϋπολογισθέν.

Όσον αφορά τα επιτυχημένα προϊόντα λογισμικού μόλις το 16,2% των έργων ολοκληρώνονται μέσα στον προσχεδιασμένο χρόνο και προϋπολογισμό. Στις μεγάλες επιχειρήσεις οι αριθμοί είναι ακόμα χειρότεροι: μόνο το 9% των έργων ολοκληρώνονται εντός χρόνου και προβλεπόμενου προϋπολογισμού. Οι μεγάλες εταιρίες, λόγω των περίπλοκων δομών τους, χρησιμοποιούν κατά 85% παραδοσιακά μοντέλα ανάπτυξης λογισμικού, και τα τελικά έργα περιέχουν μόνο το 42% των αρχικών-προτεινόμενων χαρακτηριστικών που επιθυμούν οι πελάτες να έχει το λογισμικό. Οι μικρότερες επιχειρήσεις και ιδιαίτερα οι νεοφυείς (startups) πάνε πολύ καλύτερα. Το 79% αυτών των επιχειρήσεων χρησιμοποιεί ευέλικτες τεχνικές ανάπτυξης και σε ποσοστό 74,2% τα παραδοτέα προϊόντα λογισμικού περιέχουν τα πρωτότυπα χαρακτηριστικά και λειτουργίες όπως αυτές ζητήθηκαν από τους πελάτες τους.

Η Standish Group διαχειρήστηκε περαιτέρω αυτά τα αποτελέσματα από τις μεγάλες, μεσαίες και μικρές επιχειρήσεις. Στη συγκεκριμένη έρευνα μια μεγάλη εταιρία θεωρείται οποιαδήποτε εταιρία με έσοδα μεγαλύτερα από 500 εκατομμύρια δολάρια ανά έτος, μια μεσαία εταιρία ορίζεται ως έχοντας έσοδα από 100 έως και 500 εκατομμύρια δολάρια ανά έτος και μια μικρή εταιρεία θεωρείται αυτή με ετήσια έσοδα λιγότερα από 100 εκατομμύρια δολάρια.

Τα αποτελέσματα σχετικά με την αποτυχία έργων λογισμικού είναι εξίσου απογοητευτικά για επιχειρήσεις όλων των μεγεθών. Μόνο το 9% των έργων σε μεγάλες εταιρίες ήταν επιτυχημενα. Στα 16,2% και 28% αντίστοιχα, τα προϊόντα λογισμικού σε μεσαίες και μικρές επιχειρήσεις ήταν επιτυχημένα.

Η πιο σημαντική πτυχή της έρευνας αφορά τους λόγους για τους οποίους αποτυγχάνουν τόσο πολλά έργα ανάπτυξης προϊόντων λογισμικού. Για το σκοπό

αυτό η Standish Group ρωτήσε IT διευθυντικά στελέχη για τις απόψεις τους σχετικά με τους παράγοντες που οδηγούν στην αποτελεσματική και επιτυχημένη ανάπτυξη ενός προϊόντος λογισμικού. Τα αποτελέσματα της έρευνας έδειξαν ότι οι τρεις βασικοί λόγοι επιτυχίας ενός προϊόντος λογισμικού είναι:

- **η σαφής δήλωση των απαιτήσεων** (23%)
- **η συμμετοχή του τελικού χρήστη/πελάτη στη διαδικασία ανάπτυξης** (20,1%)
- **η ενεργή συμμετοχή διευθυντικών στελεχών** (17,4%).

Στην συγκεκριμένη έρευνα βρέθηκαν και άλλα κριτήρια επιτυχίας, αλλά αυτά τα τρία κριτήρια είναι τα σημαντικότερα για αυξημένες πιθανότητες επιτυχίας. Χωρίς αυτά, η πιθανότητα αποτύχιας αυξάνεται δραματικά.

Στη συνέχεια τα συμμετέχοντα στην έρευνα διευθυντηκά στελέχη IT ερωτήθηκαν σχετικά με τα στοιχεία εκείνα τα οποία δυσκολεύουν ή πολλές φορές αποτρέπουν την επιτυχή ολοκλήρωση ενός προϊόντος λογισμικού. Τα αποτελέσματα συνάδουν απόλυτα με τα κριτήρια επιτυχίας ενός έργου και τα σημαντικότερα είναι:

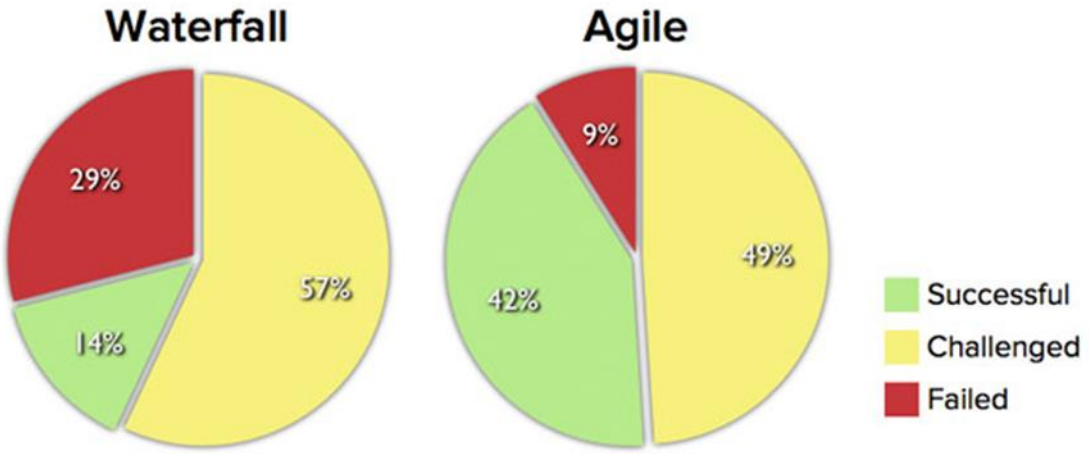
- **Η έλλειψη στοιχείων εισόδου από τους χρήστες** (user input) (19%)
- **Οι ασαφείς και ατελείς απαιτήσεις και προδιαγραφές** (16,2%)
- **Οι συνεχώς μεταβαλλόμενες απαιτήσεις και προδιαγραφές** (12,8 %)

Ακόμη σύμφωνα με την Standish Group[6] τα έργα τα οποία βασίζονται σε ευέλικτες τεχνικές ανάπτυξης *(agile)* παρουσιάζουν τριπλάσιο ποσοστό επιτυχίας από ότι τα έργα τα οποία βασίζονται σε παραδοσιακά μοντέλα ανάπτυξης και κυρίως στην μέθοδο του καταρράκτη *(waterfall).*

Το παρακάτω γράφημα δείχνει τα ποσοστά επιτυχίας με βάση τα έργα λογισμικού που υλοποιήθηκαν στις Η.Π.Α από το 2002 έως και το 2012. Αξίζει να σημειωθεί ότι ως επιτυχημένο έργο ορίζεται αυτό το οποίο παραδόθηκε στην ώρα του, σύμφωνα

[6] https://www.mountaingoatsoftware.com/blog/agile-succeeds-three-times-more-often-than-waterfall

με τα προκαθορισμένα κόστη και ανταποκρίνεται σε όλες τις προγραμματισμένες του λειτουργίες.



Εικόνα 1-1 Ποσοστά επιτυχίας Παραδοσιακών Μοντέλων Ανάπτυξης και Ευέλικτων Μοντέλων

Το συγκεκριμένο καταφανές πλεονέκτημα των ευέλικτων τεχνικών έχει οδηγήσει πολλούς οργανισμούς και εταιρείες που ασχολούνται με την ανάπτυξη καινοτόμων προϊόντων λογισμικού να στραφούν κυρίως σε ευέλικτες τεχνικές για την ανάπτυξη τέτοιων προϊόντων λογισμικού.

Σύμφωνα με τη συμβουλευτική εταιρία Actuation Consulting[7] η οποία έκανε μια μεγάλη έρευνα το 2013 σχετικά με τις τεχνικές ανάπτυξης λογισμικού που χρησιμοποιούνται σε 13.000 εταιρίες των Η.Π.Α τα αποτελέσματα ήταν εντυπωσιακά όσον αφορά την αποδοχή των ευέλικτων μεθόδων ανάπτυξης λογισμικού σε αντίθεση με τις παραδοσιακές τεχνικές οι οποίες μειώνονται ραγδαία. Η χρήση της μεθόδου καταρράκτη τα τελευταία χρόνια πέφτει συνεχώς, το 2012 σημείωσε πτώση 12%, ενώ το 2013 15%. Από την άλλη πλευρά, οι ευέλικτες τεχνικές παρουσιάζουν μια συνεχή αυξητική τάση. Το 2012 μόνο 12% των εταιρειών χρησιμοποιούσαν αποκλειστικά ευέλικτες μεθόδους ανάπτυξης, ποσοστό το οποίο εκτωξεύθηκε στο 35% το 2013. Τέλος, σημαντική παρατήρηση της έρευνας είναι ότι πολλές εταιρείες (53% των ερωτηθέντων) οι οποίες χρησιμοποιούσαν συνδυασμούς τεχνικών ανάπτυξης με κυριότερο παράδειγμα αυτό του συγκερασμού στοιχείων του καταρράκτη με στοιχεία των ευέλικτων μεθόδων ανάπτυξης στρέφονται κυρίως στις

[7] http://www.actuationconsulting.com/adoption-rates-for-various-product-development-methods/

ευέλικτες μεθοδολογίες. Τα ποσοστά τέτοιων εταιρειών μειώθηκαν κατά 20% το 2013.

Η γενικευμένη υϊοθέτηση ευέλικτων μεθόδων ανάπτυξης οφείλεται στα υψηλά ποσοστά αποτυχίας των IT έργων παγκοσμίως. Είναι πολύ σημαντικό ότι οι περισσότερες έρευνες δείχνουν πως το βασικότερο στοιχείο αποτυχίας των έργων είναι οι ασαφείς απαιτήσεις και προδιαγραφές και η μη-συμμετοχή του τελικού χρήστη/πελάτη στη διαδικασία ανάπτυξης. Σε καμία περίπτωση οι ευέλικτες μεθοδολογίες δεν έχουν εξαλείψει τα προαναφερθέντα προβλήματα, αλλά η ευελιξία και η ελαστικότητα τους στον τρόπο υλοποίησης αποτελούν τα βασικά χαρακτηριστικά για τα οποία οι εταιρείες καινοτόμων προϊόντων λογισμικού και ιδίως οι νεοφυείς τις επιλέγουν έναντι των παραδοσιακών.

1.2 Αντικείμενο διπλωματικής

Βασικός στόχος της παρούσας διπλωματικής είναι να προταθέι ένα πλαίσιο σχεδίασης ενός εργαλείου εκμαίευσης, ανάλυσης και αποσαφήνισης των απαιτήσεων για την ανάπτυξη καινοτόμων προϊόντων λογισμικού, το οποίο θα είναι συμβατό για τις περισσότερες περιπτώσεις κατά τις οποίες θα είναι πιθανό να χρειαστεί η εφαρμογή του. Με βάση το συγκεκριμένο πλαίσιο, η εκάστοτε ομάδα έργου θα είναι σε θέση να επιλέξει τα στοιχεία εκείνα τα οποία ταιριάζουν στο προϊόν το οποίο καλούνται να αναπτύξουν ακολουθώντας σταδιακά μια προκαθορισμένη διαδικασία. Με την ανάλυση που έχει διεξαχθεί νωρίτερα σε θεωρητικό επίπεδο, ορίζεται ο τρόπος με τον οποίο θα γίνει η εκμαίευση και ανάλυση των απαιτήσεων των χρηστών καθώς και τα πιθανά προβλήματα τα οποία ενδέχεται να οδηγήσουν σε αποτυχία του αρχικού στόχου. Η επιβεβαίωση της δυνατότητας χρήσης του προτεινόμενου πλαισίου γίνεται με την εφαρμογή του σε μία πραγματική επιχείρηση.

1.2.1 Συνεισφορά

Η συνεισφορά της διπλωματικής συνοψίζεται στα στάδια που ακολουθούν:

1. Αρχικά εξετάσαμε την υπάρχουσα κατάσταση στην αγορά των προϊόντων λογισμικού και εντοπίσαμε τους παράγοντες εκείνους οι οποίοι επηρεάζουν δραματικά την επιτυχία ή ακόμη και τη βιωσημότητα ενός προϊόντος λογισμικού.
2. Μελετήσαμε τα διάφορα μοντέλα ανάπτυξης λογισμικού και τα αντίστοιχα εργαλεία τους σε θεωρητικό επίπεδο, τα αναλύσαμε και αντλήσαμε χρήσιμα συμπεράσματα για κάθε ένα από αυτά.
3. Μελετήσαμε τους διάφορους τύπους καινοτομίας καθώς και τις μεθόδους διοίκησης καινοτόμων ομάδων και δημιουργήσαμε τους αντίστοιχους συσχετισμούς με τα μοντέλα ανάπτυξης λογισμικού.
4. Μελετήσαμε τη διαδικασία συλλογής απαιτήσεων σε ένα προϊόν λογισμικού, τις τεχνικές με τις οποίες πραγματοποιείται η εκμαίευση αυτών και αντλήσαμε ενδιαφέροντα συμπεράσματα.
5. Μελετήσαμε θεωρητικά εργαλεία τα οποία στοχεύουν στην μοντελοποίηση της διαδικασίας εκμαίευσης απαιτήσεων σε προϊόντα λογισμικού.
6. Προτείναμε ένα γενικό πλαίσιο σχεδίασης ενός εργαλείου εξαγωγής απαιτήσεων.
7. Εφαρμόσαμε σε μία πραγματική επιχειρήση το πλαίσιο σχεδίασης που προτάθηκε.
8. Πραγματοποιήθηκε η αξιολόγηση του προτεινόμενου εργαλείου και διεξήχθησαν συμπεράσματα.

1.3 Οργάνωση κειμένου

Συνολικά τα κεφάλαια της παρούσας διπλωματικής εργασίας είναι εννιά.

Το πρώτο κεφάλαιο αποτελέι μία γενική εισαγωγή στο θέμα της διπλωματικής εργασίας και παρουσιάζει το πρόβλημα το οποίο επιδιώκει να δώσει λύση.

Στο δεύτερο κεφάλαιο παρουσιάζεται το θεωρητικό υπόβαθρο πάνω στο οποίο στηρίζεται η εργασία και ειδικότερα αναφέρεται στα διάφορα κατευθυντήρια διαδικαστικά μοντέλα ανάπτυξης λογισμικού, στους τρόπους διοίκησης καινοτόμων ομάδων και εξάγονται συμπεράσματα για τις περιπτώσεις χρήσης των προαναφερθέντων μεθόδων.

Στο τρίτο κεφάλαιο παρουσιάζονται αναλυτικά τα είδη των διάφορων απαιτήσεων, η διαδικασία συλλογής των απαιτήσεων σε ένα προϊόν λογισμικού εξάγονται χρήσιμα συμπεράσματα σχετικά με τις τεχνικές ανάλυσης και εκμαίευσης των απαιτήσεων καθώς και με την εκμαίευση απαιτήσεων σε καινοτόμες επιχειρήσεις.

Στο τέταρτο κεφάλαιο παρουσιάζονται σχετικές εργασίες σε καινοτόμα προϊόντα λογισμικού και προδιαγραφές σε διάφορους τομείς.

Στο πέμπτο κεφάλαιο σκιαγραφείται ένα εργαλείο εξαγωγής απαιτήσεων, αναλύοντας κάθε πιθανό στοιχείο εισόδου που μπορεί να το αποτελεί και γίνεται σύγκριση με αντίστοιχες ακαδημαϊκές προσπάθειες.

Στο έκτο κεφάλαιο γίνεται η εφαρμογή του προτεινόμενου εργαλείου σε μία πραγματική επιχείρηση για την εξαγωγή απαιτήσεων σε καινοτόμες επιχειρήσεις καθώς και αξιολόγηση του εργαλείου τόσο ποιοτικά όσο και ποιοτικά.

Στο έβδομο κεφάλαιο διατυπώνονται τα γενικά συμπεράσματα, ενώ στο όγδοο κεφάλαιο παρατίθεται παράρτημα με το ερωτηματολόγιο που διαμοιράστηκε για την πραγματοποίηση της ποσοτικής αξιολόγησης.

Τέλος, η βιβλιογραφία που χρησιμοποιήθηκε για την εκπόνηση της παρούσας διπλωματικής εργασίας αποτελεί το ένατο κεφάλαιο.

2 *Παρουσίαση, Ανάλυση και Αξιολόγηση Μοντέλων Ανάπτυξης Λογισμικού*

Ένα γενικό διαδικαστικό μοντέλο για την τεχνολογία λογισμικού περιλαμβάνει ένα σύνολο πλαισίου και δραστηριότητες προστασίας, δράσεις και εργασίες έργου. Καθένα, από την ποικιλία των διαδικαστικών μοντέλων, μπορεί να περιγραφεί από μια διαφορετική ροή διαδικασίας, περιγραφή του τρόπου με τον οποίο οι δραστηριότητες πλαισίου, οι δράσεις και οι εργασίες οργανώνονται διαδοχικά και χρονολογικά. Διαδικαστικά υποδείγματα μπορούν να χρησιμοποιηθούν για την επίλυση κοινών προβλημάτων που αντιμετωπίζονται ως μέρος της διαδικασίας λογισμικού.

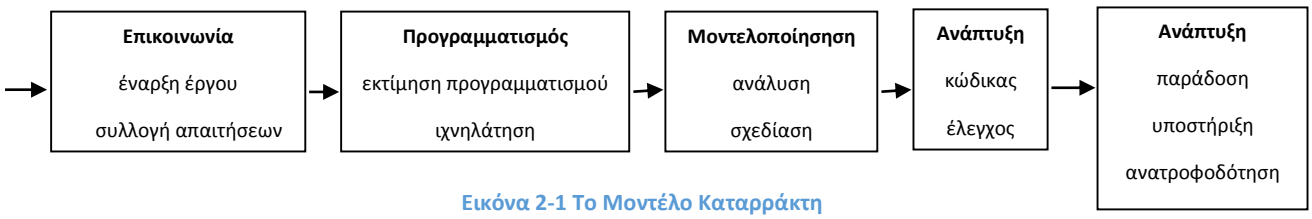
2.1 Κατευθυντήρια μοντέλα ανάπτυξης λογισμικού

Τα κατευθυντήρια μοντέλα ανάπτυξης λογισμικού αφορούν μεθοδολογίες ανάπτυξης εφαρμογών και υπηρεσιών πληροφορικής και αναφέρονται στα διαφορετικά πλαίσια που μπορούν να χρησιμοποιηθούν με στόχο τον καθορισμό της δομής, του σχεδίου, και του ελέγχου της διαδικασίας ανάπτυξης ενός πληροφοριακού συστήματος. Μια ευρεία ποικιλία τέτοιων πλαισίων έχουν εξελιχθεί με την πάροδο των ετών, το καθένα από τα οποία παρουσιάζει αναγνωρισμένα πλεονεκτήματα καθώς και αδυναμίες. Ένα μοντέλο ανάπτυξης λογισμικού δεν είναι απαραίτητα κατάλληλο για την ανάπτυξη διαφορετικών ειδών εφαρμογών και υπηρεσιών πληροφορικής. Κάθε μία από τις διαθέσιμες μεθόδολογίες είναι η καταλληλότερη για συγκεκριμένα είδη έργων, με βάση διάφορα τεχνικά και

οργανωτικά ζητήματα που αφορούν όχι μόνο τη φύση του έργου αλλά και της ομάδας. Στην παρούσα διπλωματική εργασία έγινε μια εκτενής μελέτη για κάθε ένα από τα σημαντικότερα μοντέλα που αναφέρονται στη βιβλιογραφία στο πλαίσιο μεθοδολογιών με βάση τις επιχειρήσεις, τις εφαρμογές, την οργάνωση και τα τεχνολογικά περιβάλλοντα. Ως αποτέλεσμα, παρουσιάζονται τα βασικά χαρακτηριστικά κάθε μεθοδολογίας καθώς και τα σημαντικότερα πλεονεκτήματα και μειονεκτήματα αυτών.

2.1.1 Μοντέλο Καταρράκτης (Waterfall Model)

Το μοντέλο καταρράκτη[1], γνωστό και ως ο *κλασικός κύκλος ζωής*, προτείνει μια συστηματική, γραμμική και διαδοχική προσέγγιση στην ανάπτυξη λογισμικού που αρχίζει με τον προσδιορισμό των απαιτήσεων του πελάτη και εξελίσσεται μέσα από τον σχεδιασμό, την μοντελοποίηση, την κατασκευή και την ανάπτυξη, με αποκορύφωμα τη συνεχή υποστήριξη του ολοκληρωμένου λογισμικού**.**Το μοντέλο καταρράκτη χαρακτηρίζεται από σειριακά βήματα (phases), ανάδραση ανάμεσα σε δύο γειτονικά βήματα και βασίζεται στην δημιουργία προδιαγραφών σε κάθε βήμα



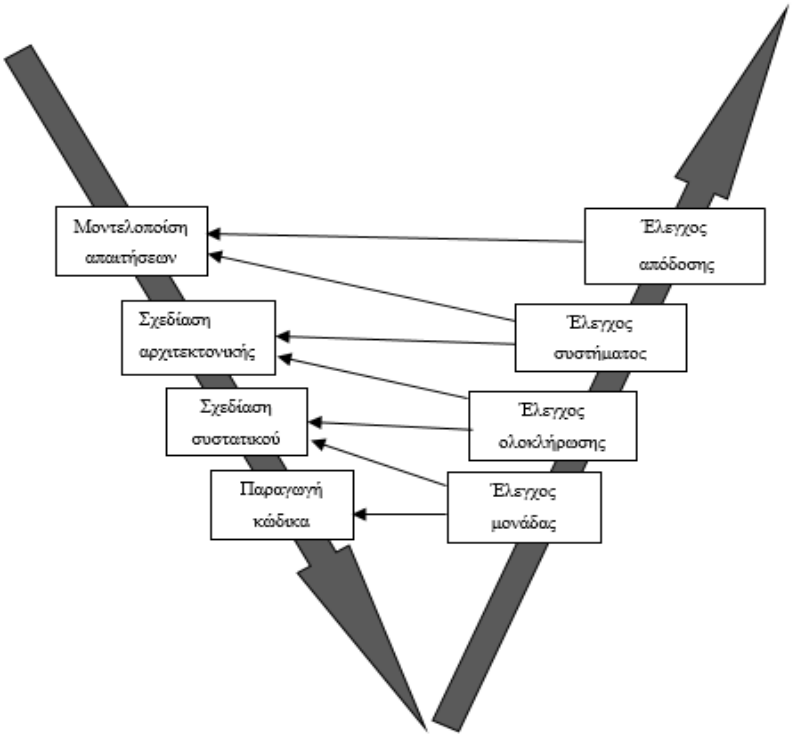
Εικόνα 2-1 Το Μοντέλο Καταρράκτη

Το μοντέλο καταρράκτη είναι το παλαιότερο πρότυπο για τους μηχανικούς λογισμικού. Ωστόσο, κατά τις τρεις τελευταίες δεκαετίες, η κριτική αυτού του μοντέλου διαδικασίας έχει προκαλέσει αμφιβολίες για την αποτελεσματικότητά του[8].

[8] https://www.mountaingoatsoftware.com

2.1.2 V Μοντέλο

Μια παραλλαγή στην αναπαράσταση του μοντέλου καταρράκτη, ονομάζεται *V-μοντέλο*[2]. Το V-μοντέλο απεικονίζει τη σχέση μεταξύ των δράσεων διασφάλισης ποιότητας και των δράσεων που σχετίζονται με την επικοινωνία, την μοντελοποίηση και τις αρχικές δραστηριότητες αναπτυξης.Καθώς μια ομάδα λογισμικού κινείται προς τα κάτω από την αριστερή πλευρά του V, βασικά προβλήματα απαιτήσεων εξευγενίζονται σταδιακά σε πιο λεπτομερείς και τεχνικές αναπαραστάσεις του προβλήματος και της επίλυσής του. Μόλις παραχθεί ο κώδικας, η ομάδα κινείται προς τα πάνω και στη δεξιά πλευρά του V, εκτελώντας οθσιαστικά μια σειρά από ελέγχους (δράσεις διασφάλισης ποιότητας) που επικυρώνει κάθε ένα από τα μοντέλα που δημιουργούνται καθώς η ομάδα κινήθηκε κάτω στην αριστερή πλευρά. Στην πραγματικότητα, δεν υπάρχει θεμελιώσης διαφορά μεταξύ του κλασικού κύκλου ζωής και του V-μοντέλου. Το V-μοντέλο αποτελεί μία μορφή απεικόνισης για τον τρόπο με τον οποίο οι δράσεις επαλήθευσης και επικύρωσης εφαρμόζονται στο αρχικό τεχνολογικό έργο.



Εικόνα 2-2 Το V-μοντέλο

2.1.3 Προοδευτικά Εξελισσόμενο Μοντέλο (Incremental)

Υπάρχουν πολλές περιπτώσεις στις οποίες οι αρχικές απαιτήσεις λογισμικού είναι αρκετά καλά καθορισμένες, αλλά το συνολικό πεδίο εφαρμογής της αναπτυξιακής προσπάθειας καθιστά αδύνατη μια καθαρά γραμμική διαδικασία[3]. Επιπρόσθετα, πολλές φορές παρουσιάζεται ανάγκη να παρέχεται ένα περιορισμένο σύνολο λειτουργικότητας λογισμικού στους χρήστες και στη συνέχεια να τελειοποιείται και να διευρύνεται η λειτουργικότητα σε μεταγενέστερες εκδόσεις λογισμικού. Σε τέτοιες περιπτώσεις, συνηθίζεται η επιλογή ενός διαδικαστικού μοντέλου που έχει σχεδιαστεί για να αναπτύσσει λογισμικό μέσω προοδευτικά εξελισσόμενης διαδικασίας[9].

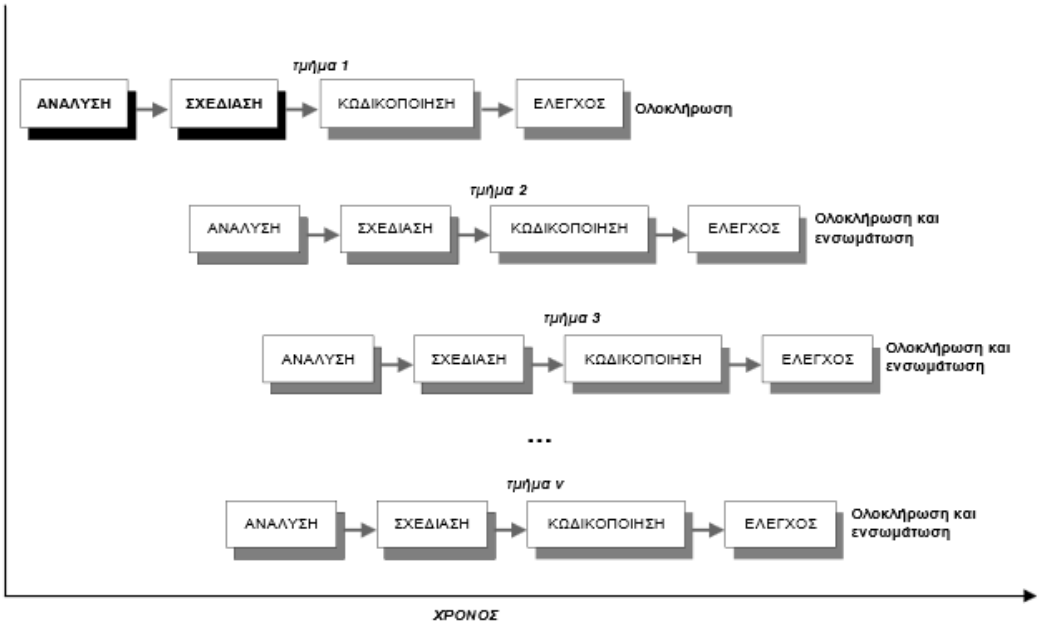
Το *προοδευτικά εξελισσόμενο μοντέλο* συνδυάζει στοιχεία της γραμμικής και της παράλληλης ροής διαδικασίας[3]. Το συγκεκριμένο μοντέλο εφαρμόζει γραμμικές ακολουθίες με έναν κλιμακωτό τρόπο. Συνεπώς, κάθε γραμμική ακολουθία παράγει παραδοτέες ‘’προσαυξήσεις’’ του λογισμικού.

Όταν ένα προοδευτικά εξελισσόμενο μοντέλο χρησιμοποιείται, η πρώτη προσαύξηση αποτελεί συνήθως ένα βασικό προϊόν. Αυτό σημαίνει, ότι ενώ οι βασικές απαιτήσεις παρουσιάζονται, συμπληρωματικά χαρακτηριστικά ακόμα δεν έχουν παραδοθεί. Το βασικό προϊόν χρησιμοποιείται από τον πελάτη και ως αποτέλεσμα της χρήσης ή της αξιολόγησης, ένα καινούργιο σχέδιο αναπτύσσεται για την επόμενη προσαύξηση. Το καινούργιο σχέδιο αναφέρεται στην τροποποίηση του βασικού προϊόντος ώστε να ανταποκρίνεται καλύτερα στις ανάγκες του πελάτη και στην παράδοση επιπλέον χαρακτηριστικών και λειτουργιών.

Το προοδευτικά εξελισσόμενο διαδικαστικό μοντέλο επικεντρώνεται στην παράδοση ενός λειτουργικού προϊόντος σε κάθε στάδιο προσαύξησης. Πρόωρες προσαυξήσεις αποτελούν ανεξάρτητες εκδόσεις του τελικού προϊόντος, αλλά παρέχουν δυνατότητα που εξυπηρετεί τον χρήστη και επίσης μια πλατφόρμα αξιολόγησης από το χρήστη.

---

9 http://istqbexamcertification.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/



Εικόνα 2-3 Το Προοδευτικά Εξελισσόμενο Μοντέλο

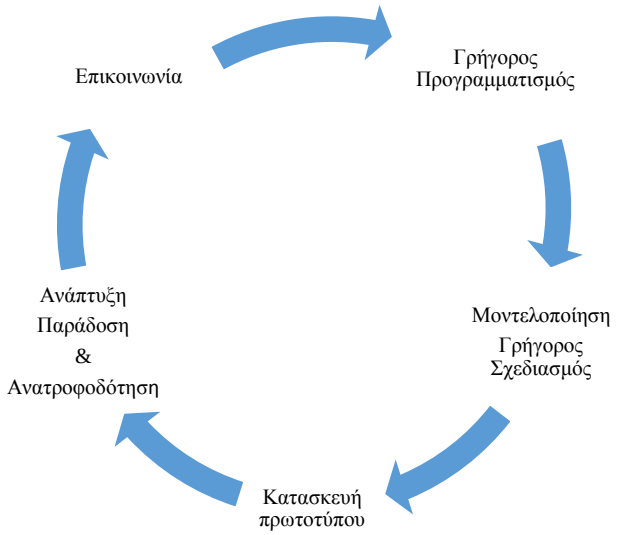
2.1.4 Μοντέλο Πρωτοτυποποίησης Προτύπου (Software Prototyping)

Οι πελάτες συχνά ορίζουν ένα σύνολο γενικών στόχων του λογισμικού, αλλά δεν ορίζουν λεπτομερώς τις απαιτήσεις για τις λειτουργίες και τα χαρακτηριστικά του λογισμικού. Είναι σύνηθες σε τέτοιες περιπτώσεις, ο υπεύθυνος για την ανάπτυξη του λογισμικού να μην είναι σίγουρος για την αποτελεσματικότητα ενός αλγορίθμου, την προσαρμοστικότητα ενός λειτουργικού συστήματος ή την μορφή την οποία θα έχει η αλληλεπίδραση μεταξύ ανθρώπου-μηχανής. Υπό αυτές τις συνθήκες [4], το *μοντέλο πρωτοτυποποίησης προτύπου* αποτελεί την καλύτερη προσέγγιση.

Αν και η πρωτοτυποποίηση μπορεί να χρησιμοποιηθεί ως ένα αυτόνομο διαδικαστικό μοντέλο, συνήθως χρησιμοποιείται ως μια τεχνική που μπορεί να εφαρμοστεί στο πλαίσιο οποιουδήποτε διαδικαστικού μοντέλου.

Το μοντέλο πρωτοτυποποίησης προτύπου αρχίζει με την επικοινωνία. Ο υπεύθυνος του έργου σε συνεργασία με τους υπόλοιπους εμπλεκόμενους καθορίζει το σύνολο των στόχων του λογισμικού, εντοπίζει ποιες απαιτήσεις είναι γνωστές και σημειώνει τις περιοχές όπου χρειάζεται περεταίρω αποσαφήνιση. Μια επανάληψη πρωτοτυποποίησης προγραμματίζεται σύντομα και η μοντελοποίηση (με τη μορφή ‘’γρήγορης σχεδίασης’’) ξεκινά. Η γρήγορη σχεδίαση επικεντρώνεται σε μια

αναπαράσταση των πτυχών του λογισμικού που θα είναι ορατές στους τελικούς χρήστες και είναι αυτή η οποία οδηγεί στην ανάπτυξη ενός πρωτότυπου. Το πρωτότυπο έχει αναπτυχθεί και αξιολογηθεί από τους εμπλεκόμενους, οι οποίοι παρέχουν ανατροφοδότηση που χρησιμοποιείται για να καλυφθούν πληρέστερα οι απαιτήσεις. Η προαναφερθείσα διαδικασία επαναλαμβάνεται προκειμένου το πρωτότυπο να προσαρμοστεί και να ικανοποιεί τις ανάγκες όλων των εμπλεκόμενων.



Εικόνα 2-4 Μοντέλο Πρωτοτυποποίησης Προτύπου

2.1.5 Το Σπειροειδές Μοντέλο (Spiral Model)

Το *σπειροειδές μοντέλο* [5], είναι ένα εξελεκτικό διαδικαστικό μοντέλο λογισμικού το οποίο συνδέει την επαναληπτική φύση των πρωτοτύπων με τις ελεγχόμενες και συστηματικές πτυχές του μοντέλου καταρράκτη και παρέχει τη δυνατότητα γρήγορης ανάπτυξης για ολοένα και πληρέστερες εκδόσεις λογισμικού.

Το σπειροειδές μοντέλο αποτελεί ένα ‘’ριψοκίνδυνο’’ διαδικαστικό μοντέλο το οποίο χρησιμοποιείται για την καθοδήγηση πολλών εμπλεκομένων σε λογισμικά παράλληλης τεχνολογίας[6].

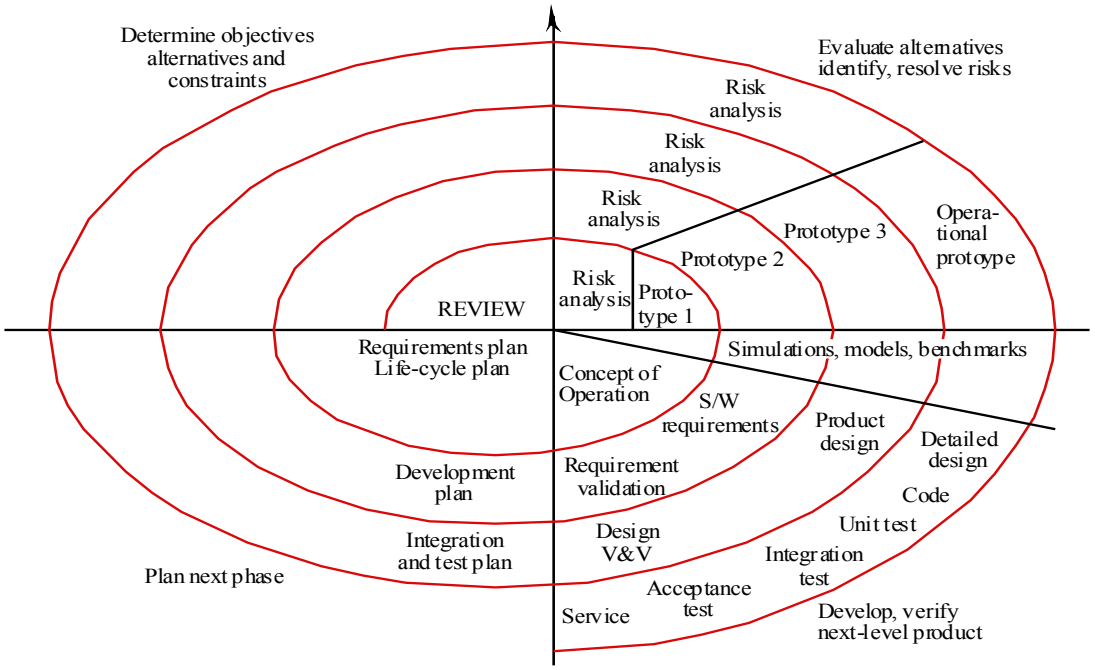
Το σπειροειδές μοντέλο παρέχει δύο βασικά γνωρίσματα. Το πρώτο είναι μια κυκλική προσέγγιση για τη σταδιακή αύξηση της έκτασης ορισμού ενός συστήματος

και της εφαρμογής αυτού καθώς μειώνεται ο βαθμός κινδύνου. Το άλλο είναι ένα σύνολο από σημεία αγκύρωσης ορόσημων, εξασφαλίζοντας τη δέσμευση των εμπλεκόμενων για εφικτές και αμοιβαίες ικανοποιητικές λύσεις.

Χρησιμοποιώντας το σπειροειδές μοντέλο, το λογισμικό αναπτύσσεται σε μια σειρά από εξελικτικές εκδόσεις. Κατα τη διάρκεια των αρχικών επαναλήψεων, η έκδοση μπορεί να αποτελεί ένα μοντέλο ή ένα πρωτότυπο. Κατά τη διάρκεια των τελικών επαναλήψεων, παράγονται ολοένα και πληρέστερες εκδόσεις του τεχνολογικού συστήματος.

Ένα σπειροειδές μοντέλο χωρίζεται σε ένα σύνολο δραστηριοτήτων πλαισίου που ορίζεται από την ομάδα τεχνολογίας λογισμικού. Κάθε μία από τις δραστηριότητες πλαισίου αντιπροσωπεύει ένα τμήμα της διαδρομής του σπιράλ. Καθώς αρχίζει η εξελικτική διαδικασία, η ομάδα λογισμικού εκτελεί τις δραστηριότητες που απαιτούνται γύρω από το σπιράλ με κατεύθυνση προς τα δεξιά, αρχίζοντας από το κέντρο. Ο κίνδυνος λαμβάνεται υπόψη με την ολοκλήρωση κάθε κύκλου. Τα σημεία αγκύρωσης ορόσημων, ένας συνδυασμός προϊόντων έργου και καταστάσεων που επιτυγχάνονται κατά μήκος της διαδρομής του σπιράλ, σημειώνονται σε κάθε εξελικτική μετάβαση.

Σε αντίθεση με άλλα διαδικαστικά μοντέλα που ολοκληρώνονται, μόλις παραδοθεί το λογισμικό, το σπειροειδές μοντέλο μπορεί να προσαρμοστεί για να εφαρμοστεί καθ’ όλη τη διάρκεια ζωής του υπολογιστικού λογισμικού.



Εικόνα 2-5 Σπειροειδές Μοντέλο[10]

2.1.6 Παράλληλα Μοντέλα

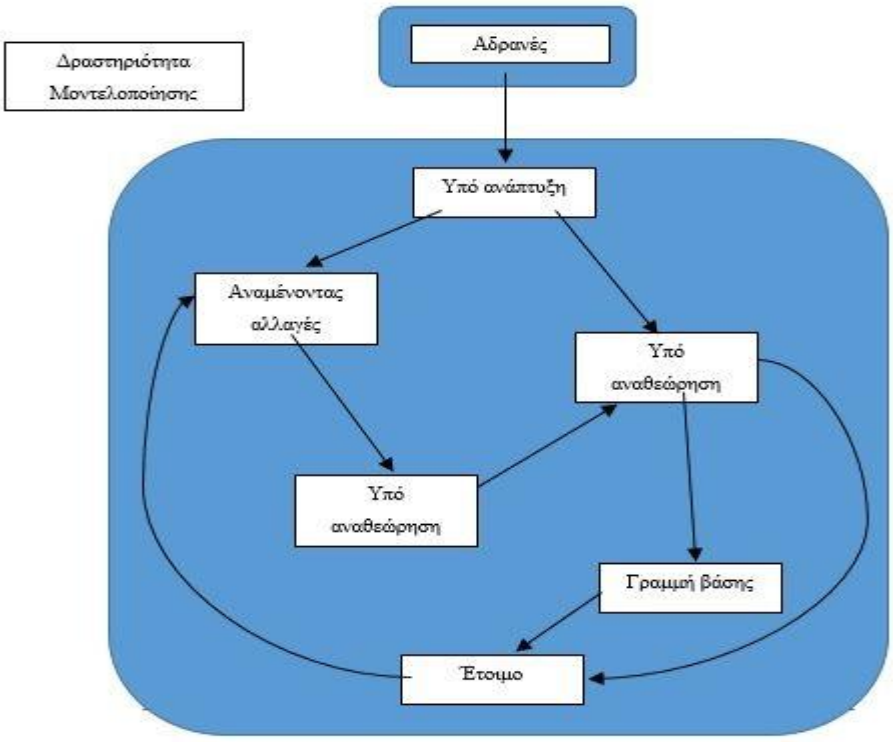
Η παράλληλη ανάπτυξη μοντέλων [3], μερικές φορές αναφέρεται ως *παράλληλη μηχανίκευση*, επιτρέπει μια ομάδα λογισμικού να παρουσιάσει επαναληπτικά και παράλληλα στοιχεία οποιασδήποτε διαδικαστικού μοντέλου.

Η *εικόνα 2-6* παρέχει μια σχηματική απεικόνιση μιας δραστηριότητας τεχνολογίας λογισμικού στο πλαίσιο της μοντελοποίησης, χρησιμοποιώντας μια παράλληλη προσέγγιση μοντελοποίησης. Η μοντελοποίηση μπορεί να είναι σε κάθε ένα από τα στάδια που σημειώνονται σε κάθε δεδομένη στιγμή. Ομοίως, άλλες δραστηριότητες, δράσεις ή εργασίες (π.χ. επικοινωνία ή κατασκευή) μπορούν να αναπαρασταθούν με ανάλογο τρόπο. Όλες οι δραστηριότητες τεχνολογίας λογισμικού εκτελούνται παράλληλα αλλά ανήκουν σε διαφορετικά στάδια.

[10] http://www.jodypaul.com/SWE/LCM/spiral.html

Αρχικά η δραστηριότητα μοντελοποίησης βρίσκεται στο στάδιο ανενεργίας, ενώ η αρχική επικοινωνία ολοκληρώθηκε και πραγματοποιεί μια μετάβαση στο στάδιο υπό ανάπτυξης.Εάν, ωστόσο, ο πελάτης δηλώσει ότι πρέπει να γίνουν αλλαγές στις απαιτήσεις, η δραστηριότητα μοντελοποίησης μετακινείται από το στάδιο υπό ανάπτυξης στο στάδιο αναμονής αλλαγών.

Η παράλληλη μοντελοποίηση ορίζει μια σειρά γεγονότων που θα ενεργοποιήσουν τις μεταβάσεις από στάδιο σε στάδιο για καθεμία από τις δραστηριότητες. Η παράλληλη μοντελοποίηση εφαρμόζεται σε όλους τους τύπους ανάπτυξης λογισμικού και παρέχει μια ακριβή εικόνα της παρούσας κατάστασης ενός έργου. Αντί να περιορίζονται οι δραστηριότητες σε μια αλληλουχία γεγονότων, το συγκεκριμένο μοντέλο ορίζει ένα διαδικαστικό δίκτυο. Κάθε δραστηριότητα υφίσταται παράλληλα με άλλες δραστηριότητες.Τέλος, γεγονότα που παράγονται σε ένα σημείο του διαδικαστικού δικτύου ενεργοποιούν μεταβάσεις μεταξύ των σταδίων.



Εικόνα 2-6 Ένα στοιχείο παράλληλου διαδικαστικού μοντέλου

2.2 Εξειδικευμένα Διαδικαστικά Μοντέλα

Τα εξειδικευμένα διαδικαστικά μοντέλα περιέχουν πολλά από τα χαρακτηριστικά ενός η περισσοτέρων παραδοσιακών μοντέλων που προαναφέρθηκαν. Τα εξειδικευμένα μοντέλα τείνουν να εφαρμόζονται όταν μια εξειδικευμένη ή σαφώς ορισμένη προσέγγιση τεχνολογίας λογισμικού πρόκειται να επιλεγεί.

2.2.1 Ανάπτυξη που βασίζεται σε Συστατικά

Τα εμπορικά συσκευασμένα (COTS: Commercial off-the-self) συστατικά λογισμικού παρέχουν στοχευμένη λειτουργικότητα με σαφώς καθορισμένες διεπαφές που επιτρέπουν το συστατικό να ενσωματωθεί στο λογισμικό που πρόκειται να κατασκευαστεί. Το αναπτυξιακό μοντέλο που βασίζεται σε συστατικά [3], ενσωματώνει πολλά χαρακτηριστικά από το σπειροειδές μοντέλο. Είναι ευέλικτο από μόνο του, απαιτώντας μία επαναληπτική προσέγγιση για την ανάπτυξη λογισμικού. Ωστόσο, το συστατικό που βασίζεται στο μοντέλο ανάπτυξης αναπτύσσει εφαρμογές από προσυσκευασμένα συστατικά λογισμικού.

Οι δραστηριότητες μοντελοποίησης και κατασκευής αρχίζουν με την αναγνώριση των υποψηφίων συστατικών. Τα συστατικά αυτά, μπορούν να σχεδιαστούν είτε ως συμβατικές ενότητες λογισμικού είτε ως αντικειμενοστραφείς κλάσεις ή πακέτα κλάσεων. Ανεξάρτητα από την τεχνολογία που χρησιμοποιείται για τη δημιουργία των συστατικών, το αναπτυξιακό μοντέλο που βασίζεται στα συστατικά, ενσωματώνει τα ακόλουθα βήματα:

1. Το διαθέσιμο προϊόν που βασίζεται στα συστατικά εξετάζεται και αξιολογείται για την εφαρμογή του
2. Εξετάζονται θέματα ενσωμάτωσης των συστατικών
3. Σχεδιάζεται η αρχιτεκτονική λογισμικού ώστε να προσαρμόσει τα συστατικά
4. Τα συστατικά ενσωματώνονται στην αρχιτεκτονική
5. Διεξάγονται εκτενείς έλεγχοι για να διασφαλιστεί η ορθή λειτουργία

Το αναπτυξιακό μοντέλο που βασίζεται στα συστατικά οδηγεί στην επαναχρησιμοπόηση λογισμικού και η επαναχρησιμοποίηση παρέχει στους

μηχανικούς λογισμικού έναν αριθμό από αξιόλογα οφέλη όπως μείωση του χρόνου του κύκλου ανάπτυξης και μείωση του κόστους έργου.

2.2.2 Το Μοντέλο τυπικών μεθόδων

Το μοντέλο τυπικών μεθόδων [3], περιλαμβάνει ένα σύνολο δραστηριοτήτων που οδηγεί σε τυπικά μαθηματικούς προσδιορισμούς του υπολογιστικού λογισμικού. Οι τυπικές μέθοδοι επιτρέπουν τον καθορισμό, την ανάπτυξη και την επικύρωση ενός υπολογιστικού συστήματος, εφαρμόζοντας μια αυστηρή μαθηματική παράσταση. Μία παραλλαγή της προσέγγισης αυτής, ονομάζεται cleanroom και εφαρμόζεται από ορισμένους οργανισμούς ανάπτυξης λογισμικού.

Όταν οι τυπικές μέθοδοι χρησιμοποιούνται κατά την ανάπτυξη, παρέχουν ένα μηχανισμό για την εξάλειψη πολλών προβλημάτων που είναι δύσκολο να ξεπεραστούν με τη χρήση άλλων προτύπων τεχνολογίας λογισμικού. Ασάφεια, ανεπάρκεια και ασυνέπεια μπορούν να ανακαλυφθούν και να διορθωθούν πιο εύκολα μέσω της μαθηματικής ανάλυσης. Τέλος, όταν οι τυπικές μέθοδοι χρησιμοποιούνται κατά τη διάρκεια της σχεδίασης λειτουγούν ως βάση για την επικύρωση του προγράμματος.

Παρόλο που δεν αποτελεί μια επικρατούσα προσέγγιση, το μοντέλο τυπικών μεθόδων υπόσχεται ένα λογισμικό απαλλαγμένο από ατέλειες. Ώστοσο, έχει εκφραστεί ενησυχία για την εφαρμογή του σε ένα επιχειρηματικό περιβάλλον που:

1. Συνήθως η ανάπτυξη τυπικών μοντέλων είναι αρκετά χρονοβόρα και δαπανηρή
2. Λίγοι προγραμματιστές λογισμικού έχουν το αναγκαίο υπόβαθρο για να εφαρμόσουν τυπικές μεθόδους μιας και η εκτεταμένη εκπαίδευση αυτών καθίσταται απαραίτητη
3. Είναι δύσκολο να χρησιμοποιηθούν τα μοντέλα ως ένας μηχανισμός επικοινωνίας για τους πελάτες που δεν έχουν πείρα σε τεχνικά θέματα

Παρόλο που υπάρχουν αυτές οι ανησυχίες, η προσέγγιση τυπικών μεθόδων έχει κερδίσει υποστηρικτές μεταξύ προγραμματιστών λογισμικού που πρέπει να αναπτύξουν κρίσιμα σε ασφάλεια λογισμικά (π.χ. οι προγραμματιστές των ηλεκτρονικών αεροσκαφών και ιατρικών μηχανημάτων) και μεταξύ

προγραμματιστών που θα υποστούν σοβαρές οικονομικές δυσκολίες σε περίπτωση που συμβούν σφάλματα.

2.2.3 Ανάπτυξη λογισμικού βασισμένη σε πτυχές (Aspect-oriented development process)

Ανεξάρτητα από τη διαδικασία λογισμικού που έχει επιλεγεί, οι υπεύθυνοι ανάπτυξης περίπλοκων λογισμικών εφαρμόζουν πάντοτε ένα σύνολο από τοπικοποιημένα χαρακτηριστικά, λειτουργίες και περιεχόμενο πληροφοριών. Τα χαρακτηριστικά αυτού του τοπικοποιημένου λογισμικού μοντελοποιούνται ως πτυχές και στη συνέχεια αναπτύσσονται στα πλαίσια ενός αρχιτεκτονικού συστήματος. Μιας και τα σύγχρονα υπολογιστικά συστήματα καθίστανται πιο εξελιγμένα, ορισμένες ανησυχίες εκτείνονται σε ολόκληρη την αρχιτεκτονική. Ορισμένες ανησυχίες αφορούν υψηλού επιπέδου ιδιότητες του συστήματος (π.χ. ασφάλεια, ανοχή σφαλμάτων). Άλλες ανησυχίες επηρεάζουν τις λειτουργίες (π.χ. η εφαρμογή των επιχειρηματικών κανόνων), ενώ άλλες χαρακτηρίζονται από μεθοδικότητα (π.χ. εργασία συγχρονισμού ή διαχείρηση μνήμης).

Όταν οι ανησυχίες προσκρούουν σε πολλές λειτουργίες του συστήματος αναφέρονται συχνά ως *εγκάρσιες ανησυχίες*. Αυτές οι εγκάρσιες ανησυχίες, ορίζονται από την ανάπτυξη λογισμικού βασισμένη σε πτυχές[7], (AOSD: Aspect-oriented software development) και έχουν αντίκτυπο σε ολόκληρη την αρχιτεκτονική του λογισμικού. Η AOSD, αναφέρεται συχνά ως προγραμματισμός βασισμένος σε πτυχές (AOP: Aspect-oriented programming), αποτελεί ένα σχετικά νέο πρότυπο τεχνολογίας λογισμικού που παρέχει μια διαδικασία και μεθοδολογική προσσέγιση για τον προσδιορισμό, τον καθορισμό, το σχεδιασμό και την ανάπτυξη των aspect μηχανισμών πέρα απο τις υπορουτίνες και την κληρονομιά της έκφρασης μιας εγκάρσιας ανησυχίας.

Το Aspect-Oriented τεχνολογικό συστατικό (AOCE: aspect-oriented component engineering) [3],[7] χρησιμοποιεί μια οριζόντια έννοια για να χωρίσει τα κάθετα αποσυνθετικά συστατικά λογισμικού, τα οποία ονομάζονται *διαστάσεις*, για να χαρακτηρίσει την εγκάρσια λειτουργικότητα και τις μη λειτουργικές ιδιότητες των

συστατικών, Οι διαδεδομένες συστηματικές διαστάσεις περιλαμβάνουν διεπαφές χρήστη, συνεργατικό έργο, κατανομή, διατήρηση, διαχείριση μνήμης, επεξεργασία συναλλαγών, ασφάλεια και ακεραιότητα. Τα συστατικά μπορεί να παρέχουν μία ή περισσότερες ‘’λεπτομέρειες διαστάσεων’’ που αφορούν μία συγκεκριμένη διάσταση οπώς μια προβολή μηχανισμού, επεκτάσιμη προσαρμοστικότητα και είδος διεπαφής, δημιουργία γεγονώτων, μεταφορά και λήψη, αποθήκευση δεδομένων και ευρετηριακή ταξινόμηση κ.ο.κ. Κάθε λεπτομέρεια διάστασης έχει έναν αριθμό από ιδιότητες, συσχετίζοντας τα λειτουργικά και μη λειτουργικά χαρακτηριστικά κάθε διάστασης.

Μία σαφής Aspect-Oriented διαδικασία δεν έχει ακόμη ωριμάσει[3]. Ωστόσο, ειναι πιθανό μια τέτοια διαδικασία να υιοθετεί χαρακτηριστικά τόσο του εξελικτικού όσο και του ταυτόχρονου μοντέλου. Το εξελικτικό μοντέλο είναι κατάλληλο όταν αναγνωρίζονται τα θέματα και στη συνέχεια αναπτύσσονται. Η παράλληλη φύση της ταυτόχρονης ανάπτυξης είναι απαραίτητη επειδή οι διαστάσεις μηχανεύονται ανεξάρτητα από τα συστατικά του τοπικοποιημένου λογισμικού και ωστόσο, οι διαστάσεις έχουν άμεσο αντίκτυπο στα συστατικά αυτά.

2.2.4 Η Ενοποιημένη Διαδικασία (Unified Process)

Η Ενοποιημένη Διαδικασία [8] αποτελεί μία προσπάθεια άντλησης των καλύτερων γνωρισμάτων και χαρακτηριστικών των παραδοσιακών διαδικαστικών μοντέλων λογισμικού και υλοποιεί πολλές απο τις αρχές της ευέλικτης ανάπτυξης (agile development) [9]. Η Ενοποιημένη Διαδικασία αναγνωρίζει τη σημαντικότητα της επικοινωνίας με τους πελάτες και βελτιστοποιεί τις μεθόδους για την περιγραφή της άποψης του πελάτη για το σύστημα (περίπτωση χρήσης). Ακόμη, η Ενοποιημένη Διαδικασία προτείνει μια ροή διαδικασίας η οποία είναι επαναλαμβανόμενη και αυξητική, παρέχοντας της προοδευτικά εξελισσόμενη αίσθηση η οποία είναι απαραίτητη στη σύγχρονη ανάπτυξη λογισμικού.

Η UP αποτελείται από πέντε βασικές δραστηριότητες [8], ή γνωστές και ως φάσεις οι οποίες χρησιμοποιούνται προκειμένου να περιγραφεί το συγκεκριμένο διαδικαστικό μοντέλο.

Η **φάση έναρξης** της UP περιλαμβάνει τόσο την επικοινωνία με τους πελάτες όσο και τις δραστηριότητες σχεδίασης. Με τη συνεργασία των εμπλεκομένων, προσδιορίζονται οι επιχειρηματικές απαιτήσεις για το λογισμικό, προτείνεται μια πρόχειρη αρχιτεκτονική για το σύστημα και ανατπύσσεται ένα σχέδιο για την επαναληπτική προοδευτικά εξελισσόμενη φύση του έργου. Οι θεμελιώδεις επιχειρηματικές απαιτήσεις περιγράφονται μέσω ενός συνόλου αρχικών περιπτώσεων χρήσης που περιγράφου τα γνωρίσματα και τις λειτουργίες της κάθε μείζονος κλάσης των επιθυμιών των χρηστών. Σε αυτό το σημείο η αρχιτεκτονική δεν είναι τίποτα περισσότερο από μια ενδεικτική περίληψη των σημαντικότερων υποσυστημάτων, που συμπληρώνει τις λειτουργίες και τα χαρακτηριστικά αυτών. Αργότερα, η αρχιτεκτονική θα τελειοποιηθεί και θα επεκταθεί σε ένα σύνολο από μοντέλα που θα παρουσιάζουν τις διαφορετικές όψεις του συστήματος. Η σχεδίαση προσδιορίζει τους πόρους, αξιολογεί τους σημαντικούς κινδύνους, καθορίζει ένα χρονοδιάγραμμα και θεσπίζει τη βάση για τα στάδια που πρέπει να εφαρμοστούν καθώς αναπτύσσεται η προσαύξηση του λογισμικού.

Η **φάση ανάπτυξης** περιλαμβάνει τις δραστηριότητες επικοινωνίας και μοντελοποίησης του γενικού διαδικαστικού μοντέλου της Ενοποιημένης Διαδικασίας. Η ανάπτυξη τελειοποιεί και επεκτείνει τις αρχικές περιπτώσεις χρήσης που αναπτύχθηκαν στο πλαίσιο της φάσης έναρξης και διευρύνει την αρχιτεκτονική αναπαράσταση ώστε να περιέχει πέντε διαφορετικές όψεις του λογισμικού, το μοντέλο περίπτωσης χρήσης, το μοντέλο απαιτήσεων, το μοντέλο σχεδίασης, το μοντέλο εφαρμογής και το μοντέλο ανάπτυξης. Σε ορισμένες περιπτώσεις, η επεξεργασία διαμορφώνει μια ‘’εκτελέσιμη γραμμή βάσης αρχιτεκτονικής’’ η οποία αναπαριστά μια ‘’πρώτη έκδοση’’ του εκτελέσιμου συστήματος. Η γραμμή βάσης αρχιτεκτονικής επιδεικνύει τη βιωσιμότητα της αρχιτεκτονικής αλλά δεν παρέχει όλα τα γνωρίσματα και τις λειτουργίες που απαιτούνται για τη χρήση του συστήματος. Επιπρόσθετα, το σχέδιο επανεξετάζεται προσεκτικά κατά την κορύφωση της φάσης επεξεργασίας για να διασφαλιστεί ότι το πεδίο εφαρμογής, οι κίνδυνοι και οι ημερομηνίες παράδοσης παραμένουν εύλογες.

Η **φάση κατασκευής** της UP χρησιμοποιώντας το αρχιτεκτονικό μοντέλο ως είσοδο,αναπτύσσει ή αποκτά τα συστατικά λογισμικού που θα καταστήσουν κάθε

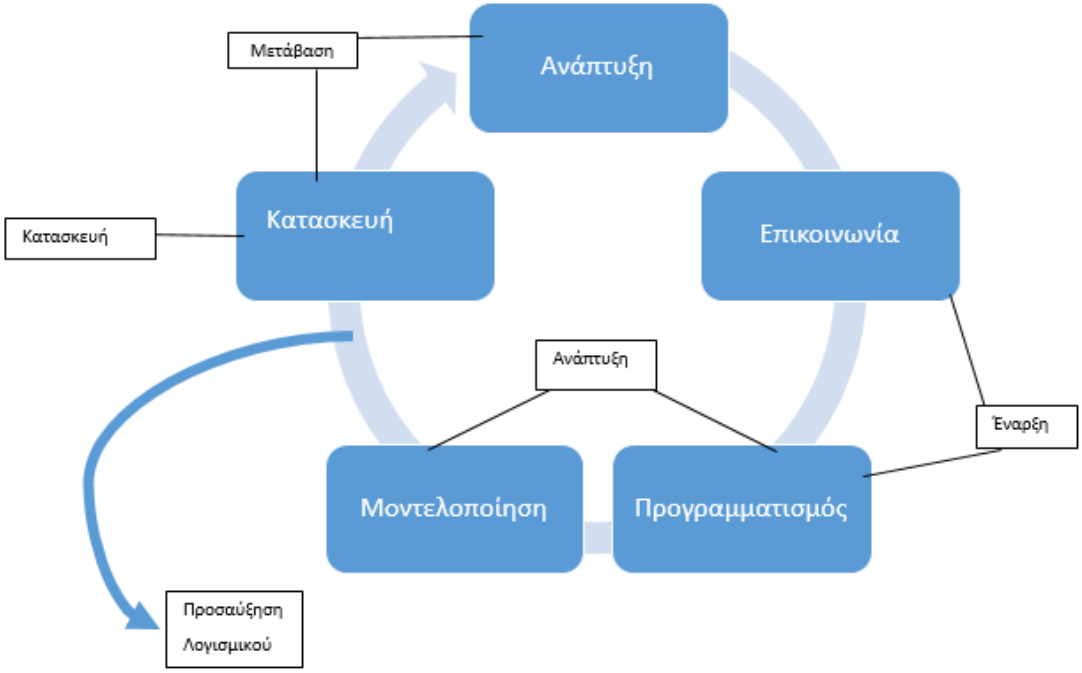
περίπτωση χρήσης λειτουργική για τους τελικούς χρήστες. Προκειμένου να επιτευχθεί αυτό, απαιτήσεις και μοντέλα σχεδίασης που ξεκίνησαν κατά τη φάση ανάπτυξης ολοκληρώνονται ώστε να σχηματίσουν την τελική έκδοση της προσύξησης λογισμικού. Όλα τα απαραίτητα και απαιτούμενα γνωρίσματα και λειτουργίες για την προσαύξηση λογισμικού εφαρμόζονται κατόπιν, σε πηγαίο κώδικα. Ακόμη διεξάγονται δραστηριότητες ενσωμάτωσης (συστατικά συναρμολόγησης και ελέγχους αποδοχής που εκτελούνται πριν από την έναρξη της επόμενης UP φάσης.

Η **φάση μετάβασης** της UP περιλαμβάνει τα τελευταία στάδια της γενικής δραστηριότητας κατασκευής και το πρώτο μέρος της γενικής δραστηριότητας ανάπτυξης (παράδοση και ανατροφοδότηση). Το λογισμικό παραδίδεται στους τελικούς χρήστες για δοκιμαστικό έλεγχο και η ανατροφοδότηση των χρηστών αναφέρει τόσο τα ελαττώματα όσο και τις απαραίτητες αλλαγές. Ακόμη, η ομάδα λογισμικού διαμορφώνει τις απαραίτητες πληροφορίες υποστήριξης (π.χ. εγχειρίδια χρήστη, οδηγοί αντιμετώπισης προβλημάτων, διαδικασίες εγκατάστασης) που απαιτούνται για την έκδοση.

Η **φάση παραγωγής** της UP συμπίπτει με τη δραστηριότητα ανάπτυξης της γενικής διαδικασίας. Κατά τη διάρκεια αυτής της φάσης, παρακολουθείται η συνεχιζόμενη χρήση του λογισμικού, παρέχεται υποστήριξη για το λειτουργικό περιβάλλον καθώς επίσης υποβάλλονται και αξιολογούνται οι αναφορές ελαττωμάτων και τα αιτήματα για αλλαγές.

Είναι πιθανό την ίδια χρονική περίοδο που εκτελείται η φάση κατασκευής, η φάση μετάβασης και η φάση παραγωγής να έχει ήδη αρχίσει η εργασία για την επόμενη προσαύξηση λογισμικού. Αυτό σημαίνει ότι οι πέντε φάσεις της UP δε λαμβάνουν χώρα διαδοχικά, αλλά περισσότερο με κατανεμημένο ταυτοχρονισμό.

Η ροή έργου της τεχνολογίας λογισμικού διανέμεται σε όλες τις UP φάσεις. Στο πλαίσιο της UP, μια ροή έργου εντοπίζει τις εργασίες που απαιτούνται για να ολοκληρωθεί μια σημαντική δραση τεχνολογίας λογισμικού και τα προιόντα έργου που παράγονται ως αποτέλεσμα της επιτυχούς ολοκλήρωσης των εργασιών.



Εικόνα 2-7 Η Ενοποιημένη Διαδικασία

2.3 Τεχνολογικά Διαδικαστικά Εργαλεία

Ένα ή περισσότερα από τα διαδικαστικά μοντέλα που εξετάστηκαν στις προηγούμενες ενότητες πρέπει να προσαρμοστούν για χρήση από μια ομάδα λογισμικού. Για να επιτευχθεί αυτό, *τεχνολογικά διαδικαστικά εργαλεία* έχουν αναπτυχθεί για να βοηθήσουν τους οργανισμούς λογισμικού να αναλύουν τις τρέχουσες διαδικασίες τους, να οργανώνουν τις εργασίες του έργου, να ελέγχουν και να παρακολουθούν την πρόοδο και να διαχειρίζονται την τεχνική ποιότητα.

Τα τεχνολογικά διαδικαστικά εργαλεία επιτρέπουν την οργάνωση του λογισμικού για την ανάπτυξη ενός αυτοματοποιημένου μοντέλου του πλαισίου της διαδικασίας, ενός συνόλου εργασιών και των δραστηριοτήτων ασφάλειας. Το μοντέλο συνήθως απεικονίζεται ως ένα δίκτυο, και στη συνέχεια αναλύεται προκειμένου να καθορίσει την τυπική ροή εργασίας και να εξετάσει εναλλακτικές δομές διαδικασίας που μπορεί να οδηγήσουν σε μείωση του χρόνου ανάπτυξης ή του κόστους.

Μόλις διαμορφωθεί μια αποδεκτή διαδικασία, άλλα τεχνολογικά διαδικαστικά εργαλεία μπορούν να χρησιμοποιηθούν για την κατανομή, την παρακολούθηση και τον έλεγχο όλων των δραστηριοτήτων. Κάθε μέλος της ομάδας λογισμικού μπορεί να χρησιμοποιήσει τέτοιου είδους εργαλεία για την αναπτυξη καταλόγου αναφοράς και

για τις δραστηριότητες διασφάλισης της ποιότητας που πρόκειται να διεξαχθούν. Τα τεχνολογικά διαδικαστικά εργαλεία μπορούν επίσης να χρησιμοποιηθούν για να συντονίσουν τη χρήση άλλων εργαλείων τεχνολογίας λογισμικού που είναι κατάλληλα για μια συγκεκριμένη διαδικασία έργου.

2.3.1 Διαδικαστικά Εργαλεία Μοντελοποίησης

Εάν ένας οργανισμός εργάζεται για να βελτιώσει μια επιχειρησιακή διαδικασία (ή λογισμικό), θα πρέπει πρώτα να την κατανοήσει. Τα διαδικαστικά εργαλεία μοντελοποίησης, γνωστά και ως εργαλεία διαχείρησης διαδικασίας, χρησιμοποιούνται για να αναπαραστήσουν τα βασικά στοιχεία της διαδικασίας, ώστε να μπορεί να γίνει καλύτερα κατανοητός ο στόχος. Τέτοιου είδους εργαλεία μπορούν να παρέχουν συνδέσμους στις περιγραφές διαδικασίας που βοηθούν όσους εμπλέκονται στη διαδικασία προκειμένου να κατανοήσουν τις δράσεις και τις εργασίες του έργου που απαιτούνται για να εκτελεστεί. Τα διαδικαστικά εργαλεία μοντελοποίησης παρέχουν επίσης συνδέσμους με άλλα εργαλεία τα οποία αποδίδουν τον καθορισμό των δραστηριοτήτων διαδικασίας.

Τα εργαλεία μοντελοποίησης επιτρέπουν σε μία ομάδα να ορίσει τα στοιχεία ενός μοναδικού μοντέλου διαδικασίας (δράσεις, εργασίες, προϊόντα έργου, QA σημεία), παρέχουν λεπτομερη καθοδήγηση για το περιεχόμενο ή την περιγραφή του κάθε στοιχείου διαδικασίας και στη συνέχεια διαχειρίζονται τη ίδια τη διαδικασία. Σε ορισμένες περιπτώσεις, τα τεχνολογικά εργαλεία διαδικασίας ενσωματώνουν πρότυπα μελέτης διαχείρησης έργων όπως εκτίμηση, χρονοδιάγραμμα, παρακολούθηση και έλεγχος.

Αντιπροσωπευτικά Εργαλεία : i) Igrafx Process Tools[11]- εργαλεία που επιτρέπουν μια ομάδα να χαρτογραφήσει, να μετρήσει και να μοντελοποιήσει μια διαδικασία λογισμικού

[11] http://www.igrafx.com/products/enterprise-modeling/process

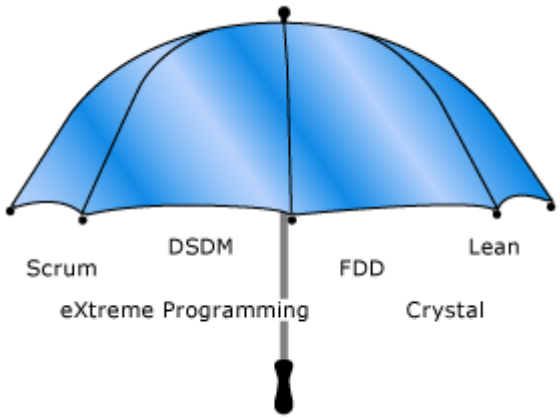
ii) Adeptia BPM Server[12]- σχεδιάστηκε για τη διαχείρηση, την αυτοματοποίηση και βελτιστοποίηση των επιχειρηματικών διαδικασιών

iii) SpeedDev Suite[13]- μια συλλογή από έξι εργαλεία με μια έντονη έμφαση στη διαχείρηση των επικοινωνιών και στις δραστηριότητες μοντελοποίησης

2.4 Ευέλικτη Ανάπτυξη (Agile Development)

Η ευέλικτη τεχνολογία λογισμικού συνδυάζει μια θεωρία και ένα σύνολο από κατευθυντήριες γραμμές ανάπτυξης [3]. Η συγκεκριμένη θεωρία προωθεί την ικανοποίηση των πελατών, την έγκαιρη προοδευτικά εξελισσόμενη παράδοση λογισμικού, τις μικρές με ισχυρά κίνητρα ομάδες έργου, τις ανεπίσημες μεθόδους, τα ελάχιστα προϊόντα έργου τεχνολογίας λογισμικού και τη συνολική απλότητα ανάπτυξης.

Οι μηχανικοί λογισμικού και άλλοι εμπλεκόμενοι του έργου όπως στελέχη, πελάτες και τελικοί χρήστες συνεργάζονται σε μια ευέλικτη ομάδα η οποία είναι αυτο-οργάνωτη. Μια ευέλικτη ομάδα προωθεί την επικοινωνία και τη συνεργασία μεταξύ όλων όσων ανήκουν σε αυτήν.



Εικόνα 2-8 Μοντέλα Ευέλικτης Ανάπτυξης

[12] https://adeptia.com/products/business-process-management-bpm.html

[13] http://www.powersim.com/main/products-services/developer-suite/

Το σύγχρονο επιχειρηματικό περιβάλλον παράγει σε μεγάλη ποσότητα υπολογιστικά συστήματα και προϊόντα λογισμικού με γρήγορο ρυθμό και συνεχώς μεταβαλλόμενες συνθήκες. Η ευέλικτη τεχνολογία λογισμικού απεικονίζει μια εύλογη εναλλακτική λύση στη συμβατική τεχνολογία λογισμικού και επιδιώκει να παραδίδει γρήγορα και επιτυχημένα συστήματα λογισμικού.

Η ευέλικτη ανάπτυξη μπορεί να ονομαστεί λιτή τεχνολογία λογισμικού [3]. Το γενικό πλαίσιο δραστηριοτήτων δηλαδή η επικοινωνία, η σχεδίαση, η μοντελοποίηση, η κατασκευή και η ανάπτυξη παραμένει, αλλά με την μορφή ενός ελάχιστου συνόλου εργασίων που ωθεί την ομάδα σχεδίασης προς την κατασκευή και την παράδοση.

Κάθε ευέλικτη διαδικασία λογισμικού χαρακτηρίζεται με τρόπο που καλύπτει ορισμένες βασικές προϋποθέσεις σχετικά με την πλειονότητα των έργων λογισμικού [10]:

1. Είναι δύσκολο να προβλεφθεί εκ των προτέρων ποιες απαιτήσεις λογισμικού θα εξακολουθούν να υφίστανται και ποιες θα αλλάξουν. Είναι εξίσου δύσκολο, να προβλεφθεί πως οι προτεραιότητες των πελατών θα αλλάξουν, καθώς εξελίσσεται το έργο.
2. Για πολλούς τύπους λογισμικού, η σχεδίαση και η κατασκευή αποτελούν παρεμβολές. Αυτό σημαίνει ότι και οι δύο δραστηριότητες θα πρέπει να εκτελούνται διαδοχικά, ώστε να αποδεικνύεται πως αναπτύχθηκαν τα μοντέλα σχεδίασης. Είναι δύσκολο να προβλεφθεί το μέγεθος της σχεδίασης που είναι απαραίτητο, προτού χρησιμοποιηθεί η κατασκευή για να αποδείξει τη σχεδίαση.
3. Η ανάλυση, η σχεδίαση, η κατασκευή και ο έλεγχος δεν καθίστανται τόσο προβλέψημα μεγέθη όσων αφορά τη σχεδιαστική πλευρά του έργου.

Από τα παραπάνω προκύπτει ότι μία ευέλικτη διαδικασία πρέπει να είναι προσαρμόσιμη προκειμένου να αντιμετωπίσει την έλλειψη προγνωσιμότητας. Ωστόσο, η συνεχής προσαρμογή χωρίς πρόοδο πετυχαίνει ελάχιστα. Ως εκ τούτου, μια ευέλικτη διαδικασία λογισμικού πρέπει να προσαρμοστεί σταδιακά. Για να επιτευχθεί η σταδιακή προσαρμογή, μια ευέλικτη ομάδα προϋποθέτει την ανατροφοδότηση των πελατών. Επομένως, μια *στρατηγική προοδευτικής αυξανόμενης ανάπτυξης* πρέπει να θεσπιστεί. Το προοδευτικά αυξανόμενο

λογισμικό, δηλαδή τα εκτελέσιμα πρωτότυπα ή τμήματα ενός λειτουργικού συστήματος, πρέπει να παραδοθούν σε σύντομα χρονικά διαστήματα, ώστε η προσαρμογή να συμβαδίζει με την αλλαγή. Η επαναληπτική αυτή προσέγγιση δίνει τη δυνατότητα στον πελάτη, να αξιολογήσει το προοδευτικά αυξανόμενο λογισμικό, να παρέχει την απαραίτητη ανατροφοδότηση στην ομάδα λογισμικού και να επηρεάσει τις προσαρμογές της διαδικασίας που γίνονται για να προσαρμόσουν την ανατροφοδότηση.

2.4.1 Αρχές της Ευελιξίας (Principles of Agile Manifesto)

Η Συμμαχία Ευελιξίας[14] ορίζει κάποιες βασικές αρχές για αυτούς που επιδιώκουν να αναπτύξουν προϊόντα λογισμικού μέσω ευέλικτων διαδικασιών. Οι αρχές είναι οι εξής:

1. Υψηλότερη προτεραιότητα είναι η ικανοποίηση του πελάτη μέσω της συνεχούς και έγκαιρης παράδοσης πολύτιμου λογισμικού
2. Είναι ευπρόσδεκτες ακόμη και οι απαιτήσεις οι οποίες εισάγονται αργά στην ανάπτυξη.
3. Γρήγορη παράδοση λογισμικού με προτίμηση στο συντομότερο χρονικό διάστημα
4. Οι άνθρωποι των επιχειρήσεων και οι υπεύθυνοι ανάπτυξης πρέπει να συνεργάζονται καθημερινά καθ'όλη τη διάρκεια του έργου
5. Ανάπτυξη έργων από άτομα που έχουν κίνητρο. Τα συγκεκριμένα άτομα χρειάζονται ένα εργασιακό περιβάλλον που να τους παρέχει την υποστήριξη και την εμπιστοσύνη που χρειάζονται προκειμένου να ολοκληρωθεί το έργο
6. Η πιο αποδοτική και αποτελεσματική μέθοδος διαβίβασης πληροφοριών σε μια ομάδα ανάπτυξης λογισμικού είναι η επικοινωνία πρόσωπο με πρόσωπο
7. Το έργο λογισμικού αποτελεί το βασικό μέτρο προόδου

[14] http://www.agilealliance.org/the-alliance/

8. Οι ευέλικτες διαδικασίες προωθούν τη βιώσιμη ανάπτυξη. Οι χορηγοί, οι υπεύθυνοι ανάπτυξης και οι χρήστες θα πρέπει να είναι σε θέση να διατηρήσουν ένα σταθερό ρυθμό επ'αόριστης προόδου
9. Η συνεχής προσοχή στη τεχνική αριστεία και καλή σχεδίαση βελτιώνει την ευελιξία
10. Η απλότητα, η τέχνη της μεγιστοποίησης του μεγέθους της εργασίας που δεν έχει εκτελεστεί, είναι απαραίτητη
11. Οι καλύτερες αρχιτεκτονικές απαιτήσεις και σχεδιάσεις προκύπτουν από αυτο-οργάνωτες ομάδες
12. Η ομάδα επανεξετάζει την αποτελεσματικότητά της σε τακτά χρονικά διαστήματα και στη συνέχεια συντονίζει και προσαρμόζει την τακτική της αναλόγως

2.4.1.1 Ευέλικτη Ανάπτυξη και Ανθρώπινοι Παράγοντες

Οι υποστηρικτές της ευέλικτης ανάπτυξης λογισμικού καταβάλουν μεγάλες προσπάθειες προκειμένου να υπογραμμίσουν τη σημασία των ''ανθρώπινων παραγόντων''. Όπως οι Cockburn και Highsmith[11] αναφέρουν: *''Η ευέλικτη ανάπτυξη εστιάζεται στα ταλέντα και στις δεξιότητες των ατόμων, μοντελοποιώντας τη διαδικασία από συγκεκριμένα άτομα και ομάδες''*. Το σημαντικό στοιχείο στη συγκεκριμένη αναφορά έγκειται στο ότι η διαδικασία διαμορφώνεται ανάλογα με τις ανάγκες των ανθρώπων και της ομάδας, όχι το αντίστροφο. Είναι αξιοσημείωτο ότι ακόμη και εταιρείες λογισμικού που χρησιμοποιούν διαφορετικά διαδικαστικά μοντέλα ανάπτυξης αναγνωρίζουν αυτή την πραγματικότητα.

Εάν τα μέλη της ομάδας λογισμικού πρόκειται να κατευθύνουν τα χαρακτηριστικά της διαδικασίας που εφαρμόζεται, ένα σύνολο βασικών χαρακτηριστικών πρέπει να υφίσταται μεταξύ των ανθρώπων μιας ευέλικτης ομάδας. Τα σημαντικότερα χαρακτηριστικά είναι τα εξής[3],[10],[11]:

- **Επάρκεια:** Σε μια ευέλικτη ανάπτυξη περιεχόμενου, η ''επάρκεια'' αναφέρεται στο έμφυτο ταλέντο, δηλαδή σε συγκεκριμένες δεξιότητες που σχετίζονται με το λογισμικό και τη συνολική γνώση της διαδικασίας που επέλεξη η ομάδα να εφαρμόσει. Οι δεξιότητες και οι γνώσεις της διαδικασίας καθ'αυτής μπορούν και

πρέπει να διδάσκονται σε όλα τα άτομα που εργάζονται ως μέλη της ευέλικτης ομάδας.

- **Κοινή Εστίαση**: Παρά το γεγονός ότι τα μέλη της ευέλικτης ομάδας μπορεί να εκτελούν διαφορετικές εργασίες και ο καθένας να κατέχει διαφορετική θέση μέσα στο έργο, όλοι πρέπει να επικεντρώνονται σε ένα στόχο: την παράδοση λειτουργικού, προοδευτικά αυξανόμενου λογισμικού στον πελάτη εντος της υποσχόμενης προθεσμίας. Για την επίτευξη αυτού του στόχου, η ομάδα θα επικεντρωθεί επίσης στις συνεχείς προσαρμογές που θα καταστίσουν τη διαδικασία σε πλήρη εναρμόνηση με τις ανάγκες της ομάδας.
- **Συνεργασία**: Η τεχνολογία λογισμικού σχετίζεται με την αξιολόγηση, την ανάλυση και τη χρήση των πληροφοριών που διαβιβάζονται στην ομάδα του λογισμικού, παρέχοντας όλα τα απαραίτητα δεδομένα που θα βοηθήσουν όλους τους εμπλεκόμενους να κατανοήσουν το έργο της ομάδας και να παρέχουν επιχειρηματική αξία στον πελάτη. Για την εκπλήρωση αυτών των εργασιών, τα μέλη της ομάδας πρέπει να συνεργάζονται – το ένα μέλος με το άλλο και όλα μαζί με τους υπόλοιπους εμπλεκόμενους.
- **Ικανότητα λήψης αποφάσεων**: Σε κάθε ικανή ομάδα λογισμικού πρέπει να δίνεται η ελευθερία να ελέγχει την πρόοδό της. Αυτό υποδηλώνει ότι δίνεται στην ομάδα αυτονομία και εξουσία λήψης αποφάσεων τόσο για θέματα τεχνικά όσο και για θέματα σχετικά με το έργο.
- **Ασαφές πρόβλημα – ικανότητα λύσης:** Οι διαχειριστές του λογισμικού πρέπει να αναγνωρίσουν ότι η ευέλικτη ομάδα θα πρέπει συνεχώς να αντιμετωπίζει ασάφειες και συνεχώς να προσαρμόζεται στις αλλαγές. Σε ορισμένες περιπτώσεις, η ομάδα πρέπει να αποδεχθεί το γεγονός ότι το πρόβλημα που προσπαθούν να λύσουν σήμερα, πιθανόν να μην είναι πρόβλημα που επιτρέπεται να λυθεί αύριο.
- **Αμοιβαία εμπιστοσύνη και σεβασμός:** Η ευέλικτη ομάδα πρέπει να γίνει αυτό που αποκαλούν οι DeMarco και Lister[12] ‘’ενωμέμενη’’ ομάδα. Μια ενωμένη ομάδα επιδεικνύει εμπιστοσύνη και σεβασμό, στοιχεία τα οποία είναι απαραίτητα για την επιτυχία της ομάδας.
- **Αυτο-οργάνωση:** Στο πλαίσιο της ευέλικτης ανάπτυξης, η αυτό-οργάνωση υποδηλώνει τρία πράγματα: (1) η ευέλικτη ομάδα οργανώνεται για να

ολοκληρωθεί το έργο, (2) η ομάδα οργανώνει τη διαδικασία για την καλύτερη προσαρμογή στο τοπικό της περιβάλλον, (3) η ομάδα οργανώνει το χρονοδιάγραμμα του έργου με σκοπό τη βέλτιστη από πλευράς ποιότητας και χρόνου παράδοση του λογισμικού. Η αυτο-οργάνωση παρουσιάζει πολλά τεχνικά οφέλη, αλλά το πιο σημαντικό είναι ότι εξυπηρετεί στη βελτίωση της συνεργασίας και της ενίσχυσης του ηθικού της ομάδας. Η ομάδα λειτουργεί με το δικό της τρόπο διαχείρησης.

2.4.2 Έντονος Προγραμματισμός (XP: Extreme Programming)

Προκειμένου να απεικονιστεί μια ευέλικτη διαδικασία πιο αναλυτικά, η ευρύτερα χρησιμοποιούμενη προσέγγιση για την ευέλικτη ανάπτυξη λογισμικού είναι αυτή του Έντονου Προγραμματισμού.

Ο Έντονος Προγραμματισμός [13] διέπεται από πέντε βασικές αξίες: επικοινωνία, απλότητα, ανατροφοδότηση(feedback), θάρρος και σεβασμό. Κάθε μία από αυτές τις αξίες χρησιμοποιείται ως οδηγός για συγκεκριμένες XP δραστηριότητες, δράσεις και εργασίες.

Προκειμένου να επιτευχθεί η αποτελεσματική επικοινωνία μεταξύ μηχανικών λογισμικού και εμπλεκόμενων, ο XP τονίζει τη στενή, όμως άτυπη συνεργασία μεταξύ των πελατών και των υπεύθυνων ανάπτυξης, την δημιουργία αποτελεσματικών προσομοιώσεων (στο πλαίσιο του XP μια προσομοίωση είναι μια ιστορία που ο καθένας πελάτης, προγραμματιστής και διαχειριστής μπορεί να πει σχετικά με το πως θεωρεί ότι λειτουργεί το σύστημα) για τις σημαντικές επικοινωνιακές έννοιες, τη συνεχή ανατροφοδότηση και την αποφυγή μεγάλων τεκμηριώσεων ως μέσο επικοινωνίας.

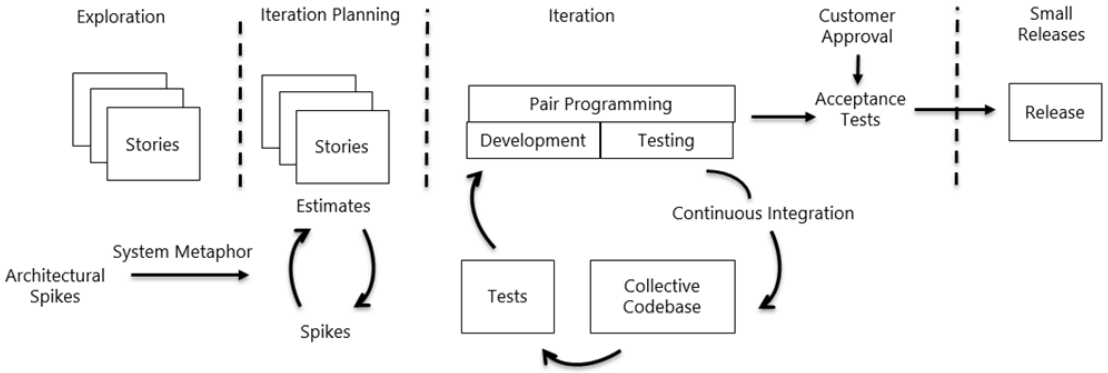
Προκειμένου να επιτευχθεί η απλότητα, ο XP περιορίζει τους υπεύθυνους ανάπτυξης να σχεδιάζουν μόνο για τις άμεσες ανάγκες, αντί να εξετάζουν τις μελλοντικές ανάγκες. Ο στόχος είναι να δημιουργηθεί μια απλή σχεδίαση που μπορεί εύκολα να εφαρμοστεί στον κώδικα. Εάν η σχεδίαση πρέπει να βελτιωθεί, μπορεί να αναδομηθεί σε μεταγενέστερο χρόνο.

Η ανατροφοδότηση(feedback) προέρχεται από τρεις διαφορετικές πηγές: από το ίδιο το εφαρμόσιμο λογισμικό, από τον πελάτη και από τα άλλα μέλη της ομάδας λογισμικού. Σχεδιάζοντας και εφαρμόζοντας μια αποτελεσματική στρατηγική ελέγχου, το λογισμικό παρέχει σημαντική ανατροφοδότηση για την ευέλικτη ομάδα. Ο XP κάνει χρήση της μονάδας ελέγχου ως βασική τακτική ελέγχου. Καθώς αναπτύσσεται κάθε κλάση, η ομάδα αναπτύσσει μια μονάδα ελέγχου για να επιδείξει κάθε λειτουργία σύμφωνα με τις καθορισμένες προδιαγραφές. Καθώς παραδίδεται μια προσαύξηση σε έναν πελάτη, οι ιστορίες χρήστη(user stories) ή περιπτώσεις χρήσης (use cases) που εφαρμόζονται από την προσαύξηση χρησιμοποιούνται ως μια βάση για τους ελέγχους αποδοχής. Ο βαθμός στον οποίο το λογισμικό υλοποιεί την παραγωγή, τη λειτουργία και τη συμπεριφορά της περίπτωσης χρήσης αποτελεί μια μορφή ανατροφοδότησης. Τέλος, καθώς προκύπτουν νέες απαιτήσεις ως μέρος του επαναληπτικού προγραμματισμού, η ομάδα παρέχει στον πελάτη μια γρήγορη ανατροφοδότηση σχετικά με το κόστος και τις επιπτώσεις στο χρονοδιάγραμμα.

Είναι αξιοσημείωτο ότι η αυστηρή τήρηση ορισμένων πρακτικών του XP προϋποθέτει θάρρος και πειθαρχία. Είναι συχνό φαινόμενο να υπάρχουν έντονες πιέσεις για τη σχεδίαση μελλοντικών απαιτήσεων. Μια ευέλικτη ομάδα XP πρέπει να έχει την πειθαρχία να σχεδιάζει για το σήμερα, αναγνωρίζοντας ότι οι μελλοντικές απαιτήσεις μπορεί να αλλάξουν δραματικά, με αποτέλεσμα να απαιτείται ουσιαστική επαναδιατύπωση της σχεδίασης και της εφαρμογής του κώδικα.

Ακολουθώντας κάθε μια από αυτές τις αξίες, η ευέλικτη ομάδα ενσταλάζει σεβασμό μεταξύ των μελών της, μεταξύ των υπόλοιπων εμπλεκομένων και των μελών της ομάδας και έμμεσα, με το ίδιο το λογισμικό. Καθώς κατορθώνουν την επιτυχή παράδοση των προσαυξήσεων λογισμικού, η ομάδα αναπτύσσει αυξανόμενο σεβασμό για την XP διαδικασία.

Ο Έντονος Προγραμματισμός χρησιμοποιεί μια αντικειμενοστραφή προσέγγιση καθώς το προτιμητέο πρότυπο ανάπτυξης περιλαμβάνει ένα σύνολο κανόνων και πρακτικών που λαμβάνουν χώρα στο πλαίσιο των τεσσάρων δραστηριοτήτων πλαισίου: προγραμματισμός, σχεδίαση, κωδικοποίηση και έλεγχος.



Εικόνα 2-9 Μοντέλο Έντονου Προγραμματισμού (XP)

Οι βασικές XP δραστηριότητες είναι οι εξής [3],[13]:

Προγραμματισμός: Η δραστηριότητα προγραμματισμού ξεκινά με την *ακρόαση*, μια δραστηριότητα συλλογής απαιτήσεων που δίνει τη δυνατότητα στα μέλη της XP ομάδας που ασχολείται με τα τεχνικά θέματα να κατανοήσουν το επιχειρηματικό πλαίσιο του λογισμικού και να ορίσουν μια ευρεία αίσθηση της απαιτούμενης απόδοσης, των σημαντικών δυνατοτήτων και της λειτουργικότητας. Η ακρόαση οδηγεί στη δημιουργία ενός συνόλου από χαρακτηριστικά καθώς και στη λειτουργικότητα του λογισμικού που πρόκειται να αναπτυχθεί. Κάθε ιστορία χρήσης γράφεται από τον πελάτη και τοποθετείται σε μια κάρτα ευρετηρίασης. Ο πελάτης εκχωρεί μια τιμή (προτεραιότητα) στην ιστορία που βασίζεται στη συνολική επιχειρηματική αξία της δυνατότητας ή της λειτουργίας. Στη συνέχεια, τα μέλη της XP ομάδας αξιολογούν κάθε ιστορία και εκχωρούν σε αυτή ένα κόστος, το οποίο μετριέται σε εβδομάδες ανάπτυξης. Εάν εκτιμάται ότι η ιστορία αυτή απαιτεί πάνω τρεις εβδομάδες ανάπτυξης, ο πελάτης καλείται να διασπάσει την ιστορία χρήσης σε μικρότερες και ο υπολογισμός της αξίας και του κόστους λαμβάνει χώρα ξανά. Είναι σημαντικό να σημειωθεί ότι νέες ιστορίες χρήσης μπορούν να γραφτούν και να προστεθούν στο σύστημα οποιαδήποτε στιγμή.

Οι πελάτες και οι υπεύθυνοι ανάπτυξης συνεργάζονται για να αποφασίσουν πως θα ομαδοποιήσουν τις ιστορίες χρήσης στην επόμενη έκδοση. Μόλις μια βασική δέσμευση (συμφωνία σχετικά με τις ιστορίες που πρέπει να περιληφθούν , την ημερομηνία παράδοσης και άλλα θέματα που αφορούν το έργο) λάβει χώρα για μία έκδοση, η XP ομάδα διευθετεί τις ιστορίες που θα αναπτυχθούν με έναν από τους

τρεις τρόπους: (1) όλες οι ιστορίες χρήσης θα εφαρμοστούν αμέσως, (2) οι ιστορίες χρήσης με μεγαλύτερη αξία θα πρέπει να μετακινηθούν προς το πάνω μέρος του χρονοδιαγράμματος και θα εφαρμοστούν πρώτες, (3) οι πιο επισφαλείς ιστορίες θα μετακινηθούν προς το πάνω μέρος του χρονοδιαγράμματος και να εφαρμοστούν πρώτες.

Αφού ολοκληρωθεί η πρώτη έκδοση του έργου και παραδοθεί στον πελάτη, η ΧΡ ομάδα υπολογίζει το ρυθμό του έργου, δηλαδή τον αριθμό των ιστοριών του πελάτη που υλοποιούνται κατα τη διάρκεια της πρώτης έκδοσης. Ο ρυθμός του έργου μπορεί μετέπειτα να χρησιμοποιηθεί για να βοηθήσει (1) στην εκτίμηση των ημερομηνιών παράδοσης και στο χρονοδιάγραμμα για τις επόμενες εκδόσεις, (2) και να καθορίσει το ενδεχόμενο να έχει γίνει υπερβολική δέσμευση για όλες τις ιστορίες χρήσης καθ'όλη τη διάρκεια ανάπτυξης του έργου. Αν μια υπερβολική δέσμευση λάβει χώρα, το περιεχόμενο των εκδόσεων τροποποιείται ή αλλάζουν οι ημερομηνίες παράδοσης.

Καθώς εξελίσσεται η ανάπτυξη του έργου, ο πελάτης έχει τη δυνατότητα να προσθέσει ιστορίες, να αλλάξει την τιμή μιας υφιστάμενης ιστορίας, να διασπάσει ιστορίες ή ακόμα και να τις εξαλείψει. Τότε, η ΧΡ ομάδα επανεξετάζει όλες τις υπόλοιπες εκδόσεις και τροποποιεί τη σχεδίαση αναλόγως.

Σχεδίαση: Η ΧΡ ακολουθεί αυστηρά την αρχή ΔΤΑ (Διατήρησέ Το Απλό). Μια απλή σχεδίαση είναι προτιμότερη από μια πιο σύνθετη απεικόνιση στον Έντονο Προγραμματισμό. Ακόμη, η σχεδίαση παρέχει κατευθυντήριες γραμμές για την εφαρμογή μιας ιστορίας που είναι ήδη γραμμένη.

Ο ΧΡ ενθαρρύνει τη χρήση καρτών CRC (Class-Responsibility Collaborator) ως έναν αποτελεσματικό μηχανισμό για τη σχεδίαση του λογισμικού σε ένα αντικειμενοστραφές πλαίσιο. Οι CRC κάρτες εντοπίζουν και οργανώνουν τις αντικειμενοστραφείς κλάσεις που σχετίζονται με την τρέχουσα προσαύξηση του λογισμικού και αποτελούν το μοναδικό προϊόν έργου σχεδίασης που παράγεται ως μέρος της ΧΡ διαδικασίας.

Εάν ένα δύσκολο πρόβλημα σχεδίασης αντιμετωπίζεται ως μέρος της σχεδίασηςμιας ιστορίας, ο XP προτείνει την άμεση δημιουργία ενός επιχειρηματικού πρωτοτύπου του εν λόγω τμήματος της σχεδίασης. Το συγκεκριμένο επιχειρηματικό πρωτότυπο ονομάζεται και επιφανειακή λύσης μιας και στόχος της δημιουργίας του είναι να ελαττωθεί ο κίνδυνος όταν ξεκινά η πραγματική εφαρμογή και να επικυρωθούν οι αρχικές εκτιμήσεις για την ιστορία που περιέχει το πρόβλημα σχεδίασης.

Ένα ακόμη πολύ σημαντικό στοιχείο του XP είναι η αναδόμηση – μια κατασκευαστική τεχνική που είναι επίσης και μια μέθοδος για τη βελτιστοποίηση της σχεδίασης. Η αναδόμηση είναι η διαδικασία της αλλαγής ενός συστήματος λογισμικού κατά τέτοιο τρόπο που δεν μεταβάλλει την εξωτερική συμπεριφορά του κώδικα αλλά βελτιώνει την εσωτερική του δομή. Αποτελεί έναν πειθαρχημένο τρόπο για να ‘’καθαριστεί’’ ο κώδικας που ελαχιστοποιεί τις πιθανότητες παρουσίασης σφαλμάτων.

Μια κεντρική έννοια του XP είναι ότι η σχεδίαση λαμβάνει χώρα τόσο πριν όσο και αφού έχει ξεκινήσει η κωδικοποίση. Η αναδόμηση δηλώνει ότι η σχεδίαση λαμβάνει χώρα συνεχώς καθώς κατασκευάζεται το σύστημα. Στην πραγματικότητα, Η δραστηριότητα κατασκευής παρέχει στην XP ομάδα καθοδήγηση για τον τρόπο βελτίωσης της σχεδίασης.

Κωδικοποίηση: Αφού αναπτύχθηκαν οι ιστορίες χρήσης και έχει γίνει η προκαταρκτική σχεδίαση του έργου, η ομάδα δεν προχωράει στην ανάπτυξη του κώδικα, αλλά αναπτύσσει μια σειρά από ελέγχους προκειμένου να ελεχθούν οι ιστορίες χρήσης που θα περιληφθούν στην τρέχουσα έκδοση (προσαύξηση λογισμικού). Μόλις ο έλεγχος μονάδας δημιουργηθεί, ο υπεύθυνος του έργου μπορεί καλύτερα να επικεντρωθεί σε αυτό που πρέπει να εφαρμοστεί για να περάσει τον έλεγχο. Τίποτα μη παρεμφερές δεν προστίθεται. Μόλις ολοκληρωθεί ο κώδικας, μπορεί να γίνει άμεσα έλεγχος μονάδας, παρέχοντας με αυτόν τον τρόπο στιγμιαία ανατροφοδότηση στους υπεύθυνους ανάπτυξης.

Μια έννοια κλειδί κατα τη διάρκει της δραστηριότητας κωδικοποίησης είναι ο προγραμματισμός σε ζεύγη. Ο XP συνιστά ότι δύο άτομα εργάζονται μαζί σε ένα σταθμό εργασίας για τη δημιουργία κώδικα σε μια ιστορία χρήσης, Αυτό παρέχει ένα

μηχανισμό για την επίλυση προβλημάτων σε πραγματικό χρόνο. Διατηρεί επίσης τους υπεύθυνους ανάπτυξης επικεντρωμένους στο προκείμενο πρόβλημα. Στην πράξη κάθε άτομο αναλαμβάνει έναν ελαφρώς διαφορετικό ρόλο. Παραδείγματος χάριν, ένα άτομο μπορεί να σκεφτεί τις λεπτομέρειες κωδικοποίησης για ένα συγκεκριμένο μέρος της σχεδίασης, ενώ ένα άλλο άτομο να διασφαλίσει ότι τα πρότυπα κωδικοποίησης ακολουθούνται ή ότι ο κώδικας για την ιστορία χρήσης θα ικανοποιήσει τον έλεγχο μονάδας που έχει αναπτυχθεί ώστε να επικυρωθεί ο κώδικας της ιστορίας.

Καθώς το ζευγάρι προγραμματιστών ολοκληρώνει τις εργασίες του, ο κώδικας που αναπτύσσουν ενσωματώνεται μέσα στο έργο των υπολοίπων. Σε ορισμένες περιπτώσεις αυτό γίνεται σε καθημερινή βάση από μια ομάδα. Σε άλλες περιπτώσεις, το ζευγάρι προγραμματιστών έχει την ευθύνη της ολοκλήρωσης. Αυτή η στρατηγική της ‘’συνεχούς ολοκλήρωσης’’ βοηθά στην αποφυγή συμβατότητας και προβλημάτων διεπαφής και παρέχει ένα πρωτόλειο περιβάλλον το οποίο βοηθά να αποκαλυφθούν σφάλματα εγκαίρως.

Έλεγχος: Η δημιουργία των ελέγχων μονάδας πριν ξεκινήσει η κωδικοποίηση αποτελεί βασικό στοιχείο της XP προσέγγισης. Οι έλεγχοι μονάδας που δημιουργούνται θα πρέπει να εφαρμοστούν χρησιμοποιώντας ένα πλαίσιο που τους επιτρέπει να αυτοματοποιηθούν. Αυτό ενθαρρύνει τη στρατηγική ελέγχων παλινδρόμησης οπότε ο κώδικας τροποποιείται.

Καθώς οι έλεγχοι μονάδας οργανώνονται σε μια ‘’καθολική ακολουθία ελέγχων’’, η ενσωμάτωση και ο έλεγχος επικύρωσης του συστήματος μπορούν να λάβουν χώρα σε καθημερινή βάση. Αυτό παρέχει στην XP ομάδα μια συνεχή ένδειξη της προόδου και μπορεί επίσης να αυξήσει τις προειδοποιητικές σημαίες εγκαίρως σε περίπτωση που συμβεί κάτι ανεπιθύμητο.

Οι XP έλεγχοι αποδοχής, οι οποίοι ονομάζονται επίσης και έλεγχοι πελάτη, καθορίζονται από τον πελάτη και επικεντρώνονται στα συνολικά χαρακτηριστικά του συστήματος και στη λειτουργικότητα που είναι ορατή και αναθεωρείται από τον

πελάτη. Οι έλεγχοι αποδοχής προέρχονται από τις ιστορίες χρήστη που έχουν υλοποιηθεί ως μέρος της έκδοσης λογισμικού.

2.4.3 Προσαρμόσιμη Ανάπτυξη Λογισμικού (ASD: Adaptive Software Development)

Η Προσαρμόσιμη Ανάπτυξη Λογισμικού (ASD) έχει προταθεί από τον Jim Highsmith ως μια τεχνική για την ανάπτυξη σύνθετου λογισμικού και συστημάτων. Οι θεωρητικές βάσεις της ASD επικεντρώνονται στην ανθρώπινη συνεργασία και την αυτο-οργάνωτη ομάδα.

Ο Highsmith [14], υποστηρίζει ότι είναι μια ευέλικτη, προσαρμόσιμη προσέγγιση ανάπτυξης που βασίζεται στη συνεργασία και αποτελεί μια πηγή διάταξης των πολύπλοκων αλληλεπιδράσεων όπως η πειθαρχία και η μηχανική. Ο ίδιος ορίζει τον ‘’κύκλο ζωής’’ του ASD που ενσωματώνει τρεις φάσεις, την πιθανολογία, τη συνεργασία και την μάθηση.

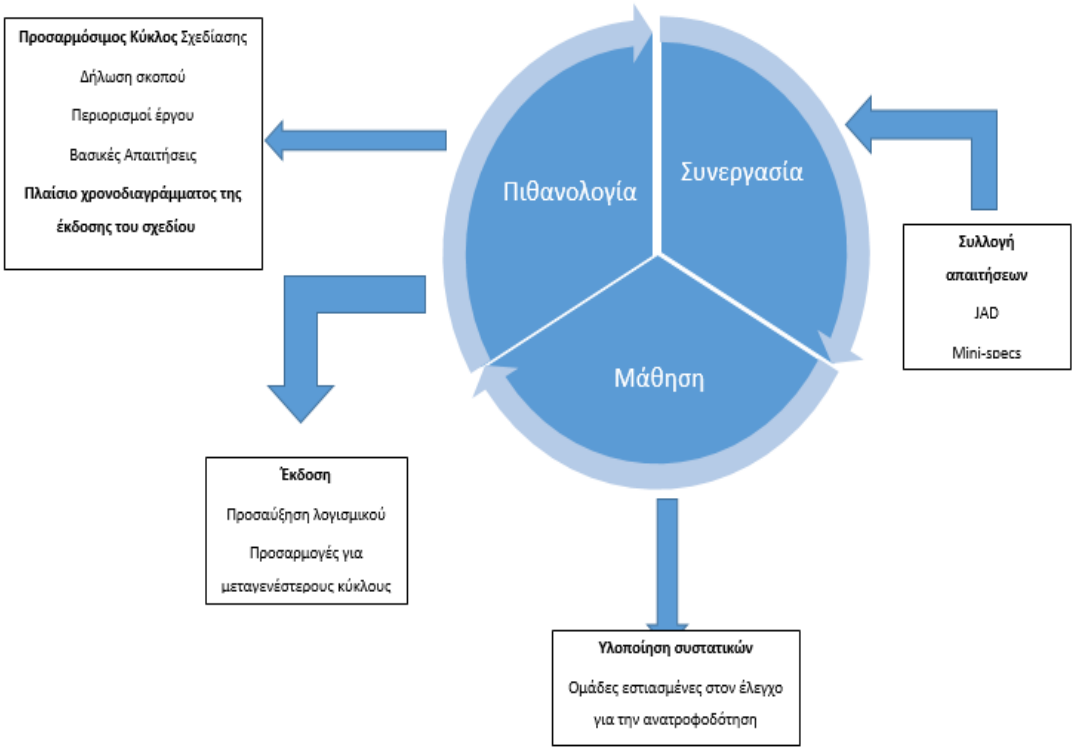
Κατα την πιθανολογία, το έργο ξεκινά και διεξάγεται ο προσαρμόσιμος κύκλος σχεδίασης. Ο προσαρμόσιμος κύκλος σχεδίασης του έργου χρησιμοποιεί πληροφορίες έναρξης, την αναφορά της αποστολής των πελατών, τους περιορισμούς του έργου και τις βασικές απαιτήσεις προκειμένου να καθοριστεί το σύνολο των κύκλων εκδόσεων που απαιτούνται για το έργο.

Ανεξαρτήτως πόσο πλήρης και διορατικός είναι ο κύκλος σχεδίασης, αλλάζει σταθερά. Με βάση τις πληροφορίες που αντλήθηκαν με την ολοκλήρωση του πρώτου κύκλου, το σχέδιο αναθεωρείται και προσαρμόζεται έτσι ώστε το προγραμματισμένο έργο να ταιριάζει καλύτερα στην πραγματικότητα στην οποία εργάζεται μια ASD ομάδα.

Άτομα με κίνητρο χρησιμοποιούν τη συνεργασία με έναν τρόπο που πολλαπλασιάζει το τελέντο της ομάδας τους και τη δημιουργική τους απόδοση. Η προσέγγιση αυτή αποτελεί ένα περιοδικό θέμα σε όλες τις ευέλικτες μεθόδους. Δυστυχώς, η συνεργασία δεν είναι κάτι εύκολο, διότι περιλαμβάνει επικοινωνία και ομαδική εργασία, αλλά υπογραμμίζει και τον ατομικισμό, διότι η προσωπική

δημιουργικότητα παίζει σημαντικό ρόλο στη συλλογική σκέψη. Πάνω από όλα, είναι θέμα εμπιστοσύνης μιας και οι άνθρωποι οι οποίοι εργάζονται μαζί πρέπει να εμπιστεύονται ο ένας τον άλλο, να ασκούν κριτική χωρίς εχθρότητα, να βοηθούν χωρίς μνησικακία, να δουλεύουν το ίδιο σκληρά, να έχουν την ικανότητα να συμβάλλουν στις εγγείς εργασίες και να μοιράζονται προβλήματα και ανησυχίες με τρόπο που να οδηγεί σε μια αποτελεσματική και ουσιαστική δράση.

Τα μέλη μιας ASD ομάδας αρχίζουν να αναπτύσσουν τα συστατικά που αποτελούν μέρος ενός προσαρμόσιμου κύκλου και δίνεται ισόποση έμφαση τόσο στη μάθηση όσο και στην πρόοδο που αφορά έναν ολοκληρωμένο κύκλο. Ο Highsmith[14] υποστηρίζει ότι οι υπεύθυνοι ανάπτυξης λογισμικού συχνά υπερεκτιμούν την αντίληψή τους καθώς και το γεγονός ότι η εκμάθησή θα τους βοηθήσει να βελτιώσουν το επίπεδο της πραγματικής κατανόησης. Οι ASD ομάδες αποκτούν γνώσεις με τρεις διαφορετικούς τρόπους: τις ομάδες εστίασης, τις τεχνικές αναθεώρησης και τις κριτικές (εκ των υστέρων) του έργου.



Εικόνα 2-10 Προσαρμόσιμη Ανάπτυξη Λογισμικού

2.4.4 Μέθοδος Scrum

Το Scrum (το όνομα προέρχεται από μια δραστηριότητα που λαμβάνει χώρα κατά τη διάρκεια ενός αγώνα ράγκμπι) αποτελεί μια ευέλικτη μέθοδο ανάπτυξης λογισμικού που επινοήθηκε από τον Jeff Sutherland και την ομάδα ανάπτυξής του στις αρχές του 1990 [3]. Τα τελευταία χρόνια έχει πραγματοποιηθεί περαιτέρω ανάπτυξη στις μεθόδους Scrum από τον Schwaber και τον Beedle [3],[15].

Οι Scrum αρχές [15], συμφωνούν με τη Διακήρυξη της ευελιξίας και χρησιμοποιούνται για να κατευθύνουν τις δραστηριότητες ανάπτυξης στο πλαίσιο μιας διαδικασίας που ενσωματώνει τις ακόλουθες δραστηριότητες πλαισίου: απαιτήσεις, ανάλυση, σχεδίαση, εξέλιξη και παράδοση. Μέσα σε κάθε δραστηριότητα πλαισίου, οι εργασίες έργου που λαμβάνουν χώρα μέσα σε ένα διαδικαστικό πρότυπο ονομάζονται *ορμή*. Το έργο που διεξάγεται μέσα στην ορμή είναι προσαρμοσμένο στα προκείμενα προβλήματα που παρουσιάζονται και αλλάζει συχνά σε πραγματικό χρόνο από την Scrum ομάδα

Το Scrum τονίζει τη χρήση ενός συνόλου διαδικαστικών προτύπων λογισμικού που έχουν αποδειχθεί αποτελεσματικά για τα έργα με στενά χρονοδιαγράμματα, μεταβαλλόμενες απαιτήσεις και επιχειρησιακή κρισιμότητα. Κάθε ένα από τα διαδικαστικά αυτά πρότυπα καθορίζει ένα σύνολο δράσεων ανάπτυξης [15]:

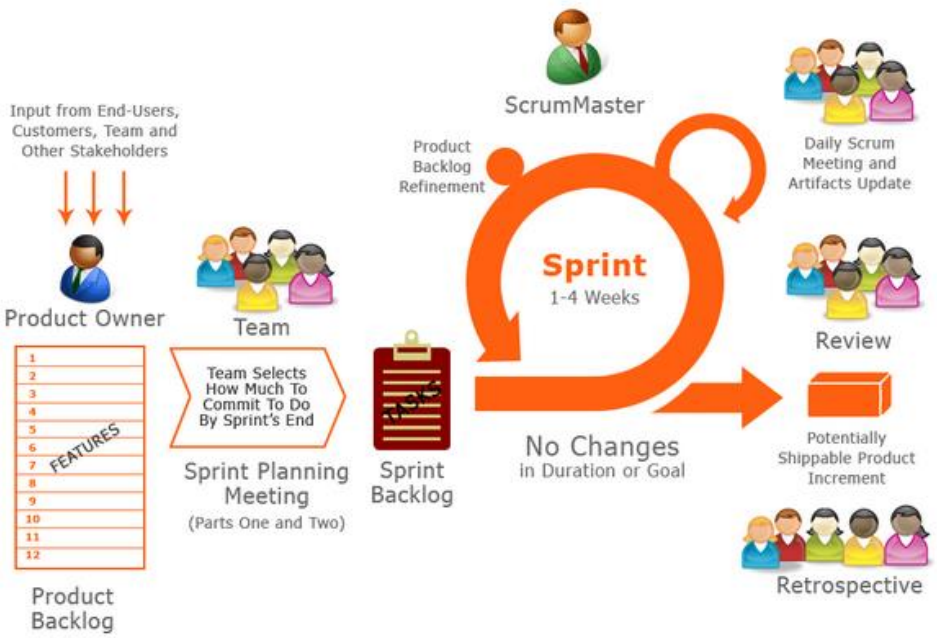
1. **Αδιεκπεραίωτες εργασίες:** Μια λίστα προτεραιοτήτων των απαιτήσεων του έργου ή χαρακτηριστικά που παρέχουν επιχειρησιακή αξία για τον πελάτη. Τα στοιχεία μπορούν να προστεθούν στις αδιεκπεραίωτες εργασίες ανά πάσα στιγμή. Ο διαχειριστής προϊόντων αξιολογεί τις αδιεκπεραίωτες εργασίες και ενημερώνει τις προτεραιότητες όπως απαιτείται.
2. **Ορμές:** Οι ορμές αποτελούνται από μονάδες έργου που απαιτούνται για να επιτευχθεί μια απαίτηση που ορίστηκε στην λίστα με τις αδιεκπεραίωτες εργασίες και πρέπει να τοποθετηθεί σε ένα προκαθορισμένο χρονικό πλαίσιο. Οι αδιεκπεραίωτες εργασίες δεν έχουν εισαχθεί κατά τη διάρκεια των ορμών με αποτέλεσμα να επιτρέπουν στα μέλη της ομάδας να εργαστούν σε ένα μικρής διάρκειας αλλά σταθερό περιβάλλον.

3. **Scrum συνεδριάσεις:** Οι εν λόγω συνεδριάσεις είναι σύντομες και διεξάγονται καθημερινά από την ομάδα. Διατυπώνονται τρεις ερωτήσεις κλειδιά και πρέπει να απαντηθούν από όλα τα μέλη της ομάδας. Οι εν λόγω ερωτήσεις είναι οι εξής:
    - Τι έχετε κάνει από την τελευταία συνεδρίαση της ομάδας?
    - Ποια εμπόδια αντιμετωπίζετε?
    - Τι σκοπεύετε να έχετε επιτύχει μέχρι την επόμενη συνεδρίαση της ομάδας?

Ένας ηγέτης της ομάδας, ο οποίος ονομάζεται κύριος Scrum (Scrum Master), κατευθύνει τη συνεδρίαση και αξιολογεί τις απαντήσεις του κάθε ατόμου. Η Scrum συνεδρίαση νοηθάει την ομάδα να ανακαλύψει πιθανά προβλήματα όσο το δυνατόν νωρίτερα. Ακόμη, οι καθημερινές αυτές συνεδριάσεις οδηγούν σε ‘’κοινωνικοποίηση της γνώσης’’ και συνεπώς, στην προώθηση μιας αυτο-οργάνωτης δομής της ομάδας.

4. **Επιδείξεις:** Παραδίδεται η προσαύξηση του λογισμικού στον πελάτη, έτσι ώστε η λειτουργικότητα που έχει υλοποιηθεί να μπορεί να αποδειχθεί και να αξιολογηθεί από τον πελάτη. Είναι ιδιαιτέρως σημαντικό ότι η επίδειξη πιθανόν να μην περιέχει όλες τις προγραμματισμένες λειτουργίες, αλλά εκείνες που μπορούν να παραδοθούν εντός του χρονικού πλαισίου που θεσπίστηκε.

Συμπερασματικά, τα Scrum διαδικαστικά πρότυπα δίνουν τη δυνατότητα σε μια ομάδα λογισμικού να λειτουργίσει με επιτυχία σε ένα περιβάλλον όπου η εξάλειψη της αβεβαιότητας είναι αδύνατη.



Εικόνα 2-11 Ροή διαδικασίας Scrum

2.4.5 Δυναμική Μέθοδος Ανάπτυξης Συστημάτων

Η Μέθοδος Δυναμικής Ανάπτυξης Συστημάτων (DSDM)[15] είναι μια ευέλικτη προσέγγιση ανάπτυξης λογισμικού που παρέχει ένα πλαίσιο για την ανάπτυξη και τη διατήρηση συστημάτων που πληρούν αυστηρούς περιορισμούς στενών χρονοδιαγραμμάτων μέσω της χρήσης της προσαυξημένης προτυποποίησης σε ένα ελεγχόμενο περιβάλλον έργου. Η DSDM θεωρία δανείζεται από μια τροποποιημένη έκδοση της αρχής Pareto, δηλαδή 80 τοις εκατό της εφαρμογής μπορεί να παραδοθεί σε 20 τοις εκατό του χρόνου που θα χρειαζόταν για να παραδωθεί η πλήρης εφαρμογή (100 τοις εκατό) [55].

Η DSDM αποτελεί μια επαναληπτική διαδικασία λογισμικού στην οποία κάθε επανάληψη ακολουθεί τον 80 τοις εκατό κανόνα. Συνεπώς, αρκετή εργασία απαιτείται για κάθε προσαύξηση ώστε να διευκολυνθεί η μετάβαση στην επόμενη προσαύξηση. Οι υπόλοιπες λεπτομέρειες μπορούν να ολοκληρωθούν αργότερα, όταν περισσότερες επιχειρησιακές απαιτήσεις είναι γνωστές ή έχουν ζητηθεί αλλαγές και έχουν προσαρμοστεί.

[15] http://www.dsdm.org/dig-deeper/book/dsdm-atern-handbook

Η DSDM Κοινοπραξία[16] είναι μία παγκόσμια ομάδα επιχειρήσεων, οι οποίες συλλογικά αναλαμβάνουν το ρόλο του ‘’φύλακα’’ της μεθόδου. Η κοινοπραξία έχει ορίσει ένα ευέλικτο διαδικαστικό μοντέλο το οποίο ονομάζεται *DSDM κύκλος ζωής* και ο οποίος ορίζει τρεις διαφορετικούς επαναληπτικούς κύκλους και ο οποίος προηγείται κατά δύο επιπλέον δραστηριότητες του κύκλου ζωής τους:

- **Μελέτη Σκοπιμότητας:** Ορίζει τις βασικές επιχειρησιακές απαιτήσεις και τους περιορισμούς που σχετίζονται με την εφαρμογή που πρόκειται να αναπτυχθεί και στη συνέχεια αξιολογεί το κατά πόσο η εφαρμογή καθίσταται βιώσιμη σύμφωνα με την DSDM διαδικασία.
- **Επιχειρησιακή Μελέτη:** Ορίζει τις λειτουργικές και τις πληροφοριακές απαιτήσεις που θα επιτρέψουν στην εφαρμογή να προσκομίσει επιχειρηματικές αξίες, τη βασική αρχιτεκτονική της εφαρμογής και προσδιορίζει τις απαιτήσεις συντήρησης για την εφαρμογή.
- **Λειτουργικό Επαναληπτικό Μοντέλο:** Παράγει ένα σύνολο προοδευτικά εξελισσόμενων πρωτοτύπων που αποδεικνύουν τη λειτουργικότητα για τον πελάτη. Ο σκοπός κατα τη διάρκεια της επανάληψης του κύκλου είναι να συγκεντρωθούν πρόσθετες απαιτήσεις εκμαιεύοντας ανατροφοδότησης από τους χρήστες οι οποίοι χρησιμοποιούν το πρωτότυπο.
- **Σχεδίαση και Ανάπτυξη Επαναληπτικών Πρωτοτύπων:** Επανεξέταση της ανάπτυξης πρωτοτύπων για να εξασφαλιστεί ότι κάθε ένα έχει αναπτυχθεί κατά τρόπο που θα του επιτρέψει να εξασφαλίσει λειτουργική επιχειρησιακή αξία για τους τελικούς χρήστες.
- **Εφαρμογή:** Τοποθετεί την τελευταία προσαύξηση λογισμικού στο λειτουργικό περιβάλλον. Είναι αξιοσημείωτο ότι η προσαύξηση πιθανόν να μην είναι 100 τοις εκατό πλήρης ή οι αλλαγές πιθανόν να ζητηθούν όταν η προσαύξηση τεθεί σε εφαρμογή. Σε κάθε περίπτωση, το DSDM αναπτυξιακό έργο συνεχίζει, επιστρέφοντας στη δραστηριότητα του λειτουργικού επαναληπτικού μοντέλου.

[16] www.dsdm.org

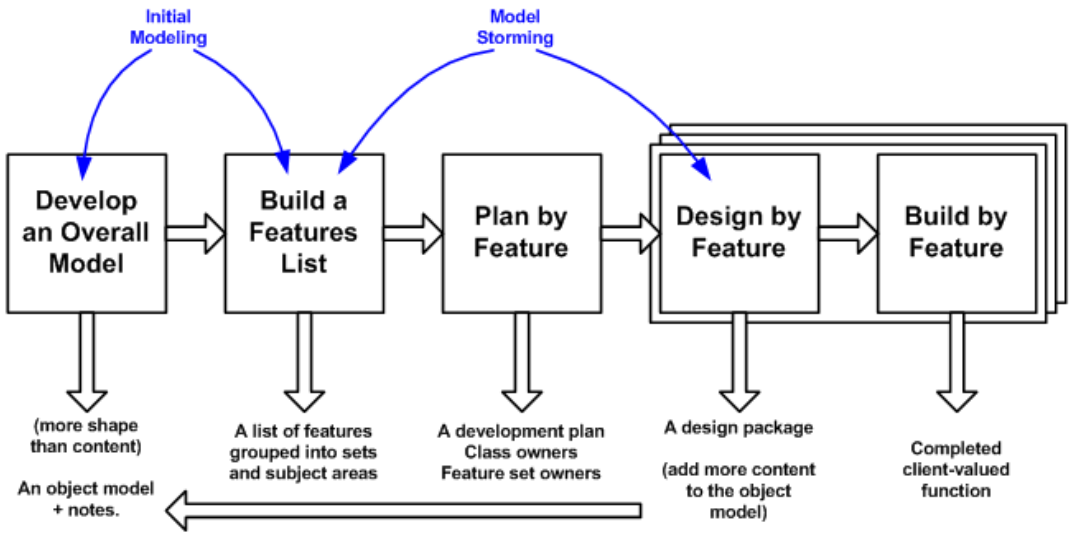
Το DSDM μπορεί να συνδυαστεί με τον XP για να παρέχει μια συνδυαστική προσέγγιση που ορίζει ένα αυστηρό διαδικαστικό μοντέλο με τις λεπτομερείς πρακτικές που απαιτούνται για την ανάπτυξη προσαυξήσεων λογισμικού. Ακόμη, η ASD έννοια της συνεργασίας και της αυτο-οργάνωτης ομάδας μπορούν να προσαρμοστούν σε ένα συνδυαστικό διαδικαστικό μοντέλο.

2.4.6 Μέθοδος Ανάπτυξης με βάση τα χαρακτηριστικά (Feature Driven Development -FDD)

Το FDD[17] μοντέλο σχεδιάστηκε για αντικειμενοστραφή τεχνολογία λογισμικού και αποτελεί μια ευέλικτη διαδικασία η οποία μπορεί να εφαρμοστεί σε μεσαίου και μεγάλου μεγέθους έργα λογισμικού.

Όπως και άλλες ευέλικτες διαδικασίες[3], το FDD υιοθετεί μια θεωρία η οποία τονίζει τη συνεργασία μεταξύ των ανθρώπων σε μια ομάδα FDD, διαχειρίζεται το πρόβλημα και την πολυπλοκότητα του έργου χρησιμοποιώντας χαρακτηριστικά που βασίζονται στην αποσύνθεση που ακολουθείται από την ενσωμάτωση των προσαυξήσεων λογισμικού και στη διαβίβαση τεχνικής λεπτομέρειας χρησιμοποιώντας κείμενο, λεκτικά και γραφικά μέσα. Το FDD τονίζει τις δραστηριότητες διασφάλισης ποιότητας λογισμικού, ενθαρρύνοντας, (1) μια προοδευτικά εξελισσόμενη στρατηγική ανάπτυξης, (2) τη χρήση της σχεδίασης και τις επιθεωρήσεις του κώδικα, (3) την εφαρμογή διασφάλισης ποιότητας λογισμικού, (4) τη συλλογή των μετρήσεων και (5) τη χρήση των πρότυπων για ανάλυση σχεδίαση και κατασκευή.

[17] http://www.featuredrivendevelopment.com/



Εικόνα 2-12 Μοντέλο ανάπτυξης που βασίζεται στα χαρακτηριστικά

Στο πλαίσιο του FDD, χαρακτηριστικό ονομάζεται μια λειτουργία η οποία είναι χρήσιμη για τους πελάτες και η οποία μπορεί να εφαρμοστεί σε δύο εβδομάδες ή και λιγότερο[3]. Η έμφαση για τον προσδιορισμό των χαρακτηριστικών παρέχει τα ακόλουθα πλεονεκτήματα:

- Επειδή τα χαρακτηριστικά είναι μικρά τμήματα παραδοτέας λειτουργικότητας, οι χρήστες μπορούν να τα περιγράψουν πιο εύκολα, κατανοούν πιο εύκολα πως συνδέονται μεταξύ τους και τα αναθεωρούν καλύτερα για ασάφεια, λάθη και παραλείψεις.
- Τα χαρακτηριστικά μπορούν να οργανωθούν σε μια ιεραρχική ομαδοποίηση, που σχετίζεται με την επιχείρηση.
- Δεδομένου ότι ένα χαρακτηριστικό αποτελεί την FDD παραδοτέα προσαύξηση λογισμικού, η ομάδα αναπτύσσει λειτουργικά χαρακτηριστικά κάθε δύο εβδομάδες.
- Επειδή τα χαρακτηριστικά είναι μικρά, η σχεδίασή τους και οι απεικονίσεις του κώδικα καθίσταται ευκολότερη προκειμένου να ελεγχθεί αποτελεσματικά.
- Η σχεδίαση, ο προγραμματισμός και η παρακολούθηση του έργου οδηγούνται από τα χαρακτηριστικά της ιεραρχίας, παρά από ένα συνολικό έργο αυθαίρετης προσαρμοσμένης τεχνολογίας λογισμικού.

Το FDD παρέχει μεγαλύτερη έμφαση στις κατευθυντήριες γραμμές και στις τεχνικές από κάθε άλλη ευέλικτη μέθοδο. Καθώς τα έργα αυξάνονται σε μέγεθος και σε πολυπλοκότητα, η ad hoc διαχείρηση του έργου είναι συχνά ανεπαρκής. Καθίσταται λοιπόν ουσιαστικής σημασίας για τους υπεύθυνους ανάπτυξης, τους διαχειριστές τους και τους υπόλοιπους εμπλεκόμενους ώστε να κατανοήσουν την κατάσταση του έργου, τι επιτεύξεις έχουν λάβει χώρα και ποια προβλήματα έχουν αντιμετωπιστεί. Εάν η προθεσμία είναι σηματνική, τότε είναι απαραίτητο να προσδιοριστεί αν είναι σωστά προγραμματισμένες οι προσαυξήσεις λογισμικού. Για να επιτευχθεί αυτό, το FDD ορίζει έξι στάδια κατά τη διάρκεια της σχεδίασης και της εφαρμογής ενός χαρακτηριστικού: ανασκόπηση σχεδίασης, σχεδίαση, επιθεώρηση σχεδίασης, κώδικας, επιθεώρηση κώδικα, προώθηση ανάπτυξης.

2.4.7 Lean Ανάπτυξη Λογισμικού (Lean Software Development)

Η Lean ανάπτυξη λογισμικού έχει προσαρμόσει τις αρχές της απλής ανάπτυξης στον κόσμο της τεχνολογίας λογισμικού. Οι απλές αρχές που εμπνέουν την LSD διαδικασία[16] μπορούν να συνοψιστούν ως η εξάλειψη των περιττών στοιχείων, ποιότητα ανάπτυξης, παραγωγή γνώσης, αναστολή δέσμευσης, γρήγορη παράδοση, σεβασμός στους ανθρώπους και βελτιστοποίηση του συνόλου.

Κάθε μια από αυτές τις αρχές μπορούν να προσαρμοστούν στη διαδικασία λογισμικού. Παραδείγματος χάριν, η εξάλειψη των περιττών στοιχείων στο πλαίσιο ενός ευέλικτου έργου λογισμικού μπορεί να ερμηνευθεί υπό την έννοια: (1) μη προσθέτοντας χαρακτηριστικά ή λειτουργίες, (2) εκτιμώντας την επίπτωση του κόστους και του χρονοδιαγράμματος μιας οποιασδήποτε απαίτησης για αλλαγή, (3) εξαλείφοντας τυχόν περιττά βήματα της διαδικασίας, (4) θεσπίζοντας μηχανισμούς για τη βελτίωση του τρόπου με τον οποίο η ομάδα συλλέγει πληροφορίες, (5) εξασφαλίζοντας ότι οι έλεγχοι αποκαλύπτουν όσο το δυνατόν περισσότερα λάθη, (6) μειώνοντας το χρόνο που απαιτείται για την έφραση μιας επιθυμίας και τη λήψη μιας απόφασης που επηρεάζει το λογισμικό ή τη διαδικασία που εφαρμόζεται για να το αναπτύξει και (7) τον εκσυγχρονισμό του τρόπου με τον οποίον οι πληροφορίες μεταδίδονται σε όλα τα άτομα που εμπλέκονται στη διαδικασία.

2.4.8 Ευέλικτη Ενοποιημένη Διαδικασία (AUP: Agile Unified Process)

Η ευέλικτη ενοποιημένη διαδικασία υιοθετεί μια ‘’διαδοχική σε μεγάλα’’ και μια ‘’επαναληπτική σε μικρά’’ θεωρία για την ανάπτυξη υπολογιστικών συστημάτων**.**Με την υιοθέτηση των κλασικών σταδίων δραστηριοτήτων της UP[8], αρχική σχεδίαση, εκπόνηση, κατασκευή και μετάβαση, η AUP[18] παρέχει μια διαδοχική επικάλυψη που επιτρέπει σε μια ομάδα να απεικονίσει τη συνολική ροή της διαδικασίας για ένα έργο λογισμικού. Ωστόσο, μέσα σε καθεμία από τις δραστηριότητες, η ομάδα πραγματοποιεί επαναλήψεις ώστε να επιτευχθεί και να παραδώσει σημαντικές προσαυξήσεις λογισμικού στους τελικούς χρήστες όσο το δυνατόν ταχύτερα. Κάθε επανάληψη της AUP εξετάζει τις ακόλουθες δραστηριότητες:

- **Μοντελοποίηση:** Αναπτύσσονται UML απεικονίσεις στους τομείς των επιχειρήσεων και των προβλημάτων. Ωστόσο, για να παραμείνουν ευέλικτα, τα μοντέλα αυτά θα πρέπει να είναι απλώς αποδοτικά ώστε να επιτρέπουν στην ομάδα να προχωρήσει.
- **Εφαρμογή:** Τα μοντέλα μεταφράζονται σε πηγαίο κώδικα.
- **Έλεγχος:** Όπως και στον XP[13], η ομάδα σχεδιάζει και εκτελεί μια σειρά ελέγχων ώστε να αποκαλύψει λάθη και να εξασφαλίσει ότι ο πηγαίος κώδικας ανταποκρίνεται στις απαιτήσεις.
- **Ανάπτυξη:** Η ανάπτυξη αυτού του πλαισίου επικεντρώνεται στη παράδοση μιας προσαύξησης λογισμικού και στην απόκτηση ανατροφοδότησης απο τους τελικούς χρήστες.
- **Διαμόρφωση και διαχείριση του έργου:** Στο πλαίσιο της AUP, η διαχείριση της διάρθωσης παρουσιάζει τη διαχείρηση των αλλαγών, τη διαχείριση των κινδύνων και τον έλεγχο κάθε επίμονου προϊόντος έργου που παράγονται από την ομάδα. Η διαχείριση του έργου παρακολουθεί και ελέγχει την πρόοδο της ομάδας και συντονίζει τις δραστηριότητες της ομάδας.

[18] http://www.ambysoft.com/unifiedprocess/agileUP.html

- **Διαχείριση περιβάλλοντος:** Η διαχείριση του περιβάλλοντος συντονίζει μια διαδικασία ανάπτυξης που περιλαμβάνει πρότυπα, εργαλεία και άλλες υποστηρικτικές τεχνολογίες, διαθέσιμες στην ομάδα.

2.5 Συμπεράσματα επί των Μεθόδων Ανάπτυξης Λογισμικού

Συμπερασματικά, τα προαναφερθέντα διαδικαστικά μοντέλα διαθέτουν κάποια βασικά χαρακτηριστικά τα οποία τα διαφοροποιούν μεταξύ τους καθώς επίσης κάθε ένα από αυτά τα μοντέλα παρουσιάζει τα δικά του πλεονεκτήματα και μειονεκτήματα, μιας και το κάθε μοντέλο είναι ιδανικό για συγκεκριμένα είδη έργων ανάπτυξης προϊόντων λογισμικού.

2.5.1 Μοντέλο Καταρράκτης

Βασικά Χαρακτηριστικά

- Το έργο χωρίζεται σε διαδοχικές φάσεις.
- Υπάρχει ανάδραση ανάμεσα σε δύο γειτονικά βήματα.
- Βασίζεται στην δημιουργία προδιαγραφών σε κάθε βήμα.
- Έμφαση δίνεται σε θέματα σχεδιασμού, χρονοδιαγράμματα, ημερομηνίες-στόχους, προϋπολογισμούς.
- Ο αυστηρός έλεγχος διατηρείται κατά τη διάρκεια ζωής του έργου με τη χρήση της εκτεταμένης τεκμηρίωσης μέσω εγγράφων, καθώς και μέσω της επίσημης ανατροφοδότησης από τον χρήστη στο τέλος των περισσότερων φάσεων πριν ξεκινώντας την επόμενη φάση.

Πλεονεκτήματα	Μειονεκτήματα
1. Ιδανικό για την υποστήριξη λιγότερο έμπειρων ομάδων έργου ή ομάδων των οποίων η σύνθεση κυμαίνεται.	1. Χρονοβόρο, δαπανηρό και περίπλοκο λόγω της άκαμπτης δομής και των αυστηρών ελέγχων.

2. Η διαδοχική σειρά των σταδίων ανάπτυξης και οι αυστηροί έλεγχοι για τη διασφάλιση της επάρκειας της τεκμηρίωσης και του σχεδιασμού διασφαλίζουν την ποιότητα, την αξιοπιστία, και συντηρησιμότητα του ανεπτυγμένου λογισμικού. 3. Η πρόοδος της ανάπτυξης του συστήματος είναι μετρήσιμη. 4. Εξοικονόμηση πόρων.	2. Το έργο προχωρά προς τα εμπρός, με μόνο μικρή κίνηση προς τα πίσω. 3. Απουσία επαναληπτικών μεθόδων-μπορεί να οδηγήσει σε προβλήματα διαχείρησης του έργου. 4. Εξαρτάται από την έγκαιρη αναγνώριση και προσδιορισμό των απαιτήσεων, αλλά οι χρήστες μπορεί να μην είναι σε θέση να ορίσουν με σαφήνεια τι χρειάζονται νωρίς στο έργο. 5. Ασυνέπεια στον προσδιορισμό των απαιτήσεων με αποτέλεσμα να υπάρχει υψηλό ρίσκο στην εξέλιξη του έργου. 6. Οι ανάγκες συχνά ανακαλύπτονται κατά τον σχεδιασμό και την κωδικοποίηση. 7. Τα προβλήματα συχνά δεν γίνονται αντιληπτά παρά μόνο στις δοκιμές του συστήματος. 8. Η απόδοση του συστήματος δεν μπορεί να ελεγχθεί μέχρι το σύστημα είναι σχεδόν πλήρως κωδικοποιημένο. 9. Δύσκολο να ανταποκριθεί στις αλλαγές. Αλλαγές που

	συμβαίνουν αργότερα στον κύκλο ζωής είναι πιο δαπανηρές και έτσι αποθαρρύνονται. 10. Παράγει υπερβολική τεκμηρίωση και η ενημέρωσή τους, καθώς το έργο εξελίσσεται, είναι χρονοβόρα. 11. Οι γραπτές προδιαγραφές είναι συχνά δύσκολο να γίνουν αντιληπτές από τους χρήστες. 12. Προάγει το χάσμα μεταξύ χρηστών και προγραμματιστών με σαφή κατανομή των αρμοδιοτήτων στους προγραμματιστές.

Πίνακας 5 Πλεονεκτήματα και Μειονεκτήματα Μοντέλου Καταρράκτη

Η χρήση του μοντέλου Καταρράκτη ενδείκνυται για:

1. Έργα τα οποία στοχεύουν στην ανάπτυξη ενός συστήματος mainframe-based ή transaction-oriented.
2. Έργα τα οποία είναι μεγάλα, δαπανηρά και πολύπλοκα.
3. Έργα τα οποία έχουν εξ'αρχής σαφείς στόχους αλλά και λύση.
4. Έργα στα οποία οι απαιτήσεις μπορούν να διατυπωθούν εξ'αρχής με πλήρη σαφήνεια και περιεκτικότητα.
5. Έργα των οποίων οι απαιτήσεις είναι σταθερές ή αμετάβλητες κατά τη διάρκεια του κύκλου ζωής του συστήματος.
6. Χρήστες οι οποίοι είναι πλήρως γνώστες του προϊόντος που καλείται η επιχείρηση να αναπτύξει προς όφελός τους.
7. Ομάδες των οποίων τα μέλη μπορεί να είναι άπειρα και η σύνθεση της ομάδας αναμένεται να διαφοροποιηθεί κατά τη διάρκεια του κύκλου ζωής.

8. Έργα στα οποία οι απαιτήσεις οφείλουν να είναι αυστηρές προκειμένου να ελέγχονται από το διαχειριστή τα προκαθορισμένα milestones.

Ωστόσο, το μοντέλο Καταρράκτης δεν ενδείκνυται για:

1. Μεγάλα έργα όπου οι απαιτήσεις δεν είναι πλήρως κατανοητές ή αλλάζουν για διάφορους λόγους, όπως εξωτερικές αλλαγές, μεταβαλλόμενες προσδοκίες, αλλαγές στον προϋπολογισμό ή γρήγορα μεταβαλλόμενη τεχνολογία.
2. Έργα ανάπτυξης πληροφοριακών συστημάτων ιστού, επειδή τα συγκεκριμένα έργα χαρακτηρίζονται από έντονη πίεση όσον αφορά το χρόνο υλοποίησης και παράδοσης σε συνδυασμό με το γεγονος ότι οι απαιτήσεις των συγκεκριμένων έργων μεταβάλλονται συνεχώς.
3. Έργα νεοφυών επιχειρήσεων όπου εμπορεύονται καινοτόμες τεχνολογίες λογισμικού (disruptive technologies).

2.5.2 Προοδευτικά Εξελισσόμενο Μοντέλο

Βασικά Χαρακτηριστικά

- Το Προοδευτικά Εξελισσόμενο Μοντέλο αποτελεί ένα συνδυασμό γραμμικών και επαναληπτικών μεθόδων ανάπτυξης λογισμικού με πρωταρχικό του στόχο να μειωθούν στο βέλτιστο βαθμό τους οι πιθανοί κίνδυνοι, με τον επιμερισμό των διαδικασίων ενός έργου σε μικρότερα τμήματα, διαδικασία η οποία αποσκοπεί στην παροχή περισσότερων βαθμών ελευθερίας όσον αφορά τις αλλαγές κατά τη διαδικασία ανάπτυξης.

Πλεονεκτήματα	Μειονεκτήματα
1. Υπάρχει δυνατότητα για την αξιοποίηση της γνώσης που αποκτήθηκε σε μια πρώιμη προσαύξηση καθώς αναπτύσσονται μεταγενέστερες προσαυξήσεις. 2. Οι εμπλεκόμενοι μπορούν να ενημερώνονται για την	1. Όταν χρησιμοποιείται μια σειρά από μίνι-καταρράκτες για ένα μικρό μέρος του συστήματος πριν από τη μετάβαση στην επόμενη προσαύξηση, υπάρχει συνήθως μια έλλειψη στη συνολική θεώρηση του προβλήματος καθώς επίσης και

κατάσταση του σχεδίου καθ 'όλη τη διάρκεια του κύκλου ζωής του έργου. 3. Επιτρέπει την παράδοση μιας σειράς εφαρμογών οι οποίες είναι σταδιακά πιο πλήρεις και μπορούν να ενσωματωθούν πιο γρήγορα στις επόμενες προσαυξήσεις. 4. Συμβάλλει στη ενσωμάτωση των κινδύνων της αρχιτεκτονικής του έργου από πρώιμα στάδια του έργου. 5. Ο έλεγχος του συστήματος διατηρείται καθ'όλη τη διάρκεια ζωής του έργου, μέσω της χρήσης των γραπτών εγγράφων και των επίσημων αναθεωρήσεων και εγκρίσεων από τους χρήστες.	στην αποσαφίνηση των επιχειρηματικών και τεχνικών απαιτήσεων για το συνολικό σύστημα. 2. Δεδομένου ότι ορισμένες ενότητες ολοκληρώνονται πολύ νωρίτερα από άλλες, σαφώς καθορισμένες διεπαφές χρήστη απαιτούνται. 3. Τα δύσκολα προβλήματα συνηθίζεται να αναβάλλονται για επόμενες προσαυξήσεις, προκειμένου η διοίκηση να προβάλλει πρόωρη επιτυχία.

Πίνακας 6 Πλεονεκτήματα και Μειονεκτήματα του Προοδευτικά Εξελισσόμενου Μοντέλου

Η χρήση του Προοδευτικά Εξελισσόμενου Μοντέλου ενδείκνυται για:

1. Μεγάλα έργα, όπου οι απαιτήσεις δεν είναι πλήρως κατανοητές ή αλλάζουν συχνά λόγω εξωτερικών αλλαγών, όπως μεταβαλλόμενων προσδοκιών και αλλαγών στον προϋπολογισμό του έργου.
2. Πληροφοριακά συστήματα Ιστού και event-driven συστήματα.
3. Έργα νεοφυών επιχειρήσεων όπου εμπορεύονται καινοτόμες τεχνολογίες λογισμικού (disruptive technologies).
4. Ωστόσο, το Προοδευτικά Εξελισσόμενο Μοντέλο δεν ενδείκνυται για:
5. Μικρά έργα μικρής διάρκειας.

6. Έργα όπου οι κίνδυνοι αρχιτεκτονικής και ενσωμάτωσης νέων διαδικασίων είναι πολύ μικροί.
7. Υψηλής διαδραστικότητας εφαρμογές όπου τα δεδομένα (input) για το έργο υπάρχουν ήδη (πλήρως ή εν μέρει).

2.5.3 Μοντέλο Πρωτοτυποποίησης Προτύπου (Prototyping)

Βασικά Χαρακτηριστικά

- Δεν αποτελεί μία αυτόνομη, πλήρης μεθοδολογία ανάπτυξης, αλλά μια προσέγγιση για το χειρισμό επιλεγμένων τμημάτων ενός μεγαλύτερου, παραδοσιακού διαδικαστικού μοντέλου ανάπτυξης.
- Επιχειρεί να μειωθούν οι κίνδυνοι με το ‘’σπάσιμο’’ ενός έργου σε μικρότερα τμήματα, η οποία οδηγεί στην παροχή μεγαλύτερης ευκολίας αλλαγής κατά τη διάρκεια της διαδικασίας ανάπτυξης.
- Οι τελικοί χρήστες συμμετέχουν σε ολόκληρη τη διαδικασία, με αποτέλεσμα να αυξάνεται η πιθανότητα αποδοχής της τελικής υλοποίησης του έργου.
- Ενώ τα περισσότερα πρωτότυπα που αναπτύσσονται τείνουν να απορρίπτονται στην πορεία του έργουμε την προσδοκία, είναι δυνατόν σε ορισμένες περιπτώσεις να εξελιχθούν από το πρωτότυπο στο σύστημα εργασίας.

Πλεονεκτήματα	Μειονεκτήματα
1. Υπογραμμίζει την αδυναμία πολλών χρηστών να καθορίσουν τις απαιτήσεις του συστήματος, και τη δυσκολία των αναλυτών του συστήματος να κατανοήσουν το περιβάλλον του χρήστη, παρέχοντας στο χρήστη ένα προσωρινό σύστημα για πειραματικούς σκοπούς το	1. Οι διαδικασίες αξιολόγησης και ελέγχου δεν είναι αυστηρές. 2. Ελλιπής ή ανεπαρκής ανάλυση των πτυχών του προβλήματος μπορεί να οδηγήσει στον εντοπισμό μόνο των προφανών και επιφανειακών αναγκών του συστήματος, με αποτέλεσμα οι τρέχουσες αναποτελεσματικές

συντομότερο δυνατό χρονικό διάστημα. 2. Μπορεί να χρησιμοποιηθεί προκειμένου να διαμορφώσει ρεαλιστικά σημαντικές πτυχές του συστήματος κατά τη διάρκεια κάθε φάσης του κύκλου ζωής των παραδοσιακών. 3. Βελτιώνει τόσο τη συμμετοχή των χρηστών στην ανάπτυξη του συστήματος όσο και την επικοινωνία μεταξύ των εμπλεκόμενων φορέων του έργου. 4. Ιδιαίτερα χρήσιμο για την ανάπτυξη έργων με ασαφείς στόχους και απαιτήσεις. 5. Ανάπτυξη και επικύρωση των απαιτήσεων των χρηστών είτε μέσω συγκρίσεων των διαφόρων σχεδιαστικών λύσεων είτε μέσω διερεύνησης της απόδοσης της αλληλεπίδρασης ανθρώπου και υπολογιστή. 6. Υπάρχει η δυνατότητα για την αξιοποίηση της ανατροφοδότησης που αποκτήθηκε σε πρώιμη επανάληψη καθώς	τεχνικές να ενσωματώνονται εύκολα στο νέο σύστημα. 3. Οι απαιτήσεις του συστήματος μπορεί να αλλάζουν συχνά. 4. Ο εντοπισμός μη λειτουργικών απαιτήσεων είναι πολύ δύσκολος. 5. Συνηθίζεται οι σχεδιαστές του συστήματος να παραμελούν την τεκμηρίωση, με αποτέλεσμα την ανεπαρκή εκ των προτέρων ανάλυση των αναγκών του συστήματος. 6. Μπορεί να οδηγήσει τον πελάτη στην λανθασμένη εντύπωση ότι το σύστημα είναι έτοιμο επειδή οι διεπαφές χρήστη είναι έτοιμες ενώ στην πραγματικότητα δεν είναι βέλτιστα λειτουργικό και χρειάζεται περαιτέρω επεξεργασία. 7. Οι επαναλήψεις αυξάνουν το κόστος προϋπολογισμού των έργων.Συνεπώς αυτό το πρόσθετο κόστος θα πρέπει να αξιολογείται έναντι των δυνητικών οφελών. Ομάδες μικρών έργων συνήθως δεν είναι σε θέση να αξιλογήσουν το

αναπτύσσονται μεταγενέστεροι κύκλοι. 7. Βοηθά στον έγκαιρο εντοπισμό δύσκολων διαδικασιών και λειτουργιών που λείπουν από το σύστημα. 8. Συμβάλλει στη δημιουργια των προδιαγραφών για μία εμπορική εφαρμογή λογισμικού. 9. Προωθεί την καινοτομία και τις ευέλικτες σχεδιαστικές τεχνικές. 10. Παρέχει γρήγορη υλοποίηση μιας ατελούς, αλλά πλήρως λειτουργικής εφαρμογής.	κόστος του επιπρόσθετου χρόνου.

Πίνακας 7 Πλεονεκτήματα και Μειονεκτήματα Μοντέλου Πρωτοτυποποίησης Προτύπου

Το μοντέλο Πρωτοτυποποίησης Προτύπου ενδείκνυται για:

1. Online πλατφόρμες λογισμικού όπου είναι έντονη η αλληλεπίδραση και η επικοινωνία μεταξύ των χρηστών καθώς επίσης και για υποτυπώδη συστήματα λήψης αποφάσεων.
2. Μεγάλα έργα με πολλούς χρήστες, αλληλεξαρτήσεις και λειτουργίες.
3. Έργα όπου οι στόχοι του είναι ασαφείς.
4. Έργα όπου οι απαιτήσεις αλλάζουν συχνά και σε σημαντικό βαθμό.
5. Έργα όπου ο πελάτης δεν είναι πλήρως γνώστης των αναγκών του συστήματος.
6. Έργα όπου ο διαχειριστής του έργου είναι εμπείρος.
7. Έργα όπου η σύνθεση της ομάδας παραμένει σταθερή καθ'όλη τη διάρκεια του έργου.
8. Έργα όπου δεν απαιτείται η ελαχιστοποίηση του κόστους και δεν υπάρχουν χρονικοί περιορισμοί.

9. Έργα όπου δεν απαιτείται αυστηρότητα όσον αφορά την έγκριση των απαιτήσεων σε καθορισμένα ορόσημα.
10. Έργα όπου δεν απαιτούνται καινοτόμες και ευέλικτες σχεδιαστικές τεχνικές σχεδίασεις για μελλοντικές επαναλήψεις.

Το μοντέλο Πρωτοτυποποίησης Προτύπου δεν ενδείκνυται για:

1. Συστήματα mainframe-based ή συστήματα transaction-oriented.
2. Συστήματα ηλεκτρονικού επιχειρείν.
3. Έργα όπου η σύνθεση της ομάδας δεν είναι σταθερή καθ'όλη τη διάρκεια του έργου.
4. Έργα όπου η μελλοντική επεκτασιμότητα (scalability) είναι ιδιαιτέρως σημαντική.
5. Έργα όπου οι στόχοι είναι σαφώς προκαθορισμένοι και ο κίνδυνος αλλαγής των απαιτήσεων είναι πολύ χαμηλός.

2.5.4 Το Σπειροειδές Μοντέλο

Βασικά Χαρακτηριστικά

- Εστίαζει στην αξιολόγηση του κινδύνου και στην ελαχιστοποίηση του, με το ''σπάσιμο'' ενός έργου σε μικρότερα τμήματα και με την παροχή περισσότερων βαθμών ελευθερίας όσον αφορά τις αλλαγές κατά τη διάρκεια της διαδικασίας ανάπτυξης.
- Κάθε κύκλος περιλαμβάνει μια εξέλικτική έκδοση μέσω της ίδιας ακολουθίας βημάτων, για κάθε τμήμα του προϊόντος και για καθένα από τα επίπεδα επεξεργασίας.
- Τα σημεία αγκύρωσης ορόσημων, ένας συνδυασμός προϊόντων έργου και καταστάσεων που επιτυγχάνονται κατά μήκος της διαδρομής του σπιράλ, σημειώνονται σε κάθε εξελικτική μετάβαση.
- Μόλις παραδοθεί το λογισμικό, το σπειροειδές μοντέλο μπορεί να προσαρμοστεί για να εφαρμοστεί καθ' όλη τη διάρκεια ζωής του υπολογιστικού λογισμικού.

Πλεονεκτήματα	Μειονεκτήματα
1. Ενισχύει την αποφυγή του κινδύνου. 2. Χρήσιμο στην επιλογή της καλύτερης μεθοδολογίας που πρέπει να ακολουθηθεί για την ανάπτυξη μιας εξελικτικής έκδοσης λογισμικού, με βάση τον κίνδυνο του έργου. 3. Μπορεί να ενσωματώσει στοιχεία από το μοντέλο Καταρράκτη, το μοντέλο Πρωτοτυποποίησης Προτύπου και το Προοδευτικά Εξελισσόμενο μοντέλο και να παρέχει πλήρη καθοδήγηση ως προς το ποιος πρέπει να είναι ο βέλτιστος συνδυασμός των μοντέλων αυτών με βάση το είδος του κινδύνου που ‘’απειλεί’’ το έργο.	1. Δυσκολία στο να καθοριστεί η ακριβής σύνθεση των μεθοδολογιών ανάπτυξης που θα χρησιμοποιηθούν για κάθε εξελικτική έκδοση. 2. Άκρως προσαρμοσμένη σε κάθε έργο, και ως εκ τούτου είναι αρκετά περίπλοκη, περιορίζοντας επαναχρησιμοποίησης. 3. Απαιτείται εξειδικευμένος και έμπειρος διαχειριστής έργου προκειμένου να καθορίσει πώς θα εφαρμόστει το μοντέλο για ένα συγκεκριμένο έργο. 4. Δεν υπάρχουν προκαθορισμένοι έλεγχοι για τη μετάβαση από τον ένα κύκλο στον άλλο κύκλο. Χωρίς ελέγχους, κάθε κύκλος μπορεί να ‘’φορτώσει’’ με επιπρόσθετη δουλειά τον επόμενο κύκλο. 5. Δεν υπάρχουν αυστηρές προθεσμίες. Ο κύκλος συνεχίζεται χωρίς σαφή κατάσταση τερματισμού, οπότε υπάρχει υψηλό ρίσκο το έργο να βγεί εκτός προϋπολογισμού.

Πίνακας 8 Πλεονεκτήματα και Μειονεκτήματα του Σπειροειδούς Μοντέλου

Το Σπειροειδές μοντέλο ενδείκνυται για:

1. Real-time και safety-critical προϊόντα λογισμικού.
2. Έργα όπου η αποφυγή του κινδύνου αποτελεί υψηλή προτεραιότητα.
3. Έργα όπου η ελαχιστοποίηση καταναλώσης πόρων δεν αποτελεί προτεραιότητα.
4. Υπεύθυνους έργων όπου είναι άκρως εξειδικευμένοι και έμπειροι.
5. Έργα όπου οι απαιτήσεις υποστηρίζουν τις αυστηρές τεκμηριώσεις μέσω εγγράφων.
6. Έργα τα οποία χρειάζονται συνδυασμό μοντέλων για την ανάπτυξή τους.
7. Έργα όπου η υψηλή ακρίβεια των διαδικασιών είναι προτεραιότητα.
8. Έργα όπου η ενσωμάτωση διαδικασιών είναι σημαντικότερη από τη λειτουργικότητα του συστήματος.

Ωστόσο, το Σπειροειδές μοντέλο δεν ενδείκνυται για:

1. Έργα όπου η αποφυγή του κινδύνου δεν αποτελεί σημαντική προτεραιότητα.
2. Έργα όπου η υψηλή ακρίβεια των διαδικασιών δεν είναι προτεραιότητα.
3. Έργα όπου η λειτουργικότητα του συστήματος ειναι σημαντικότερη από την ενσωμάτωση των διαδικασιών.
4. Έργα όπου η ελαχιστοποίηση της κατανάλωσης πόρων είναι σημαντική προτεραιότητα.

2.5.5 Μοντέλα Ευέλικτης Ανάπτυξης

Βασικά Χαρακτηριστικά

Τα μοντέλα ανάπτυξης προϊόντων λογισμικού μέσω της Ευέλικτης Ανάπτυξης υποστηρίζουν τη συνεχή ανατροφοδότηση και επιτρέπουν τις συνεχείς αλλαγές στις απαιτήσεις τους λογισμικού καθ'όλη τη διάρκεια του κύκλου ζωής ανάπτυξης λογισμικού.Ακόμη, υποστηρίζουν τη στενή συνεργασία μεταξύ των πελατών και των μηχανικών λογισμικού και προωθούν τη συχνή παράδοση στοιχείων (features) του λογισμικού ακόμα και από τους πρώτους κύκλους. Τα εν λόγω μοντέλα βασίζονται στο Μανιφέστο Ευέλικτης Ανάπτυξης το οποίο αναπτύχθηκε από μία ομάδα μηχανικών και συμβούλων λογισμικού το 2001. Σύμφωνα με το Μανιφέστο Ευέλικτης ανάπτυξης τα βασικά χαρακτηριστικά των εν λόγω μοντέλων είναι:

- Υψηλότερη προτεραιότητα είναι η ικανοποίηση του πελάτη μέσω της συνεχούς και έγκαιρης παράδοσης πολύτιμου λογισμικού
- Γρήγορη παράδοση λογισμικού με προτίμηση στο συντομότερο χρονικό διάστημα
- Οι άνθρωποι των επιχειρήσεων και οι υπεύθυνοι ανάπτυξης πρέπει να συνεργάζονται καθημερινά καθ'όλη τη διάρκεια του έργου
- Είναι ευπρόσδεκτες ακόμη και οι απαιτήσεις οι οποίες εισάγονται αργά στην ανάπτυξη.

Πλεονεκτήματα	Μειονεκτήματα
1. Ικανοποίηση των πελατών από την ταχεία και συνεχή παροχή εκδόσεων λογισμικού. 2. Δίνεται έμφαση στους ανθρώπους που εμπλέκονται στο έργο και στις αλληλεπιδράσεις μεταξύ τους παρά στα διαδικαστικά μοντέλα και στα εργαλεία που θα χρησιμοποιηθούν για την ανάπτυξη του λογισμικού.Οι τελικοί χρήστες/πελάτες και οι μηχανικοί λογισμικού αλληλεπιδρούν συνεχώς μεταξύ τους. 3. Συχνή παράδοση εκδόσεων λογισμικού(ανά εβδομάδα αντί ανά μήνα). 4. Επικοινωνία πρόσωπο με πρόσωπο. 5. Συνεχής έμφαση στα τεχνικά χαρακτηριστικά του έργου	1. Σε περίπτωση ορισμένων παραδοτέων λογισμικού, ειδικά των μεγάλων, είναι δύσκολο να εκτιμηθεί η προσπάθεια που απαιτείται κατά την έναρξη του κύκλου ζωής ανάπτυξης λογισμικού. 2. Έλλειψη έμφασης στον απαραίτητο σχεδιασμό και στην τεκμηρίωση των απαιτήσεων. 3. Το έργο μπορεί εύκολα να καθυστερήσει ή ακόμη και να ''πεθάνει'' εάν ο πελάτης δεν γνωρίζει εξ'αρχής τί ακριβώς θέλει. 4. Μόνο έμπειροι προγραμματιστές είναι σε θέση να πάρουν τις αποφάσεις που απαιτούνται κατά τη διάρκεια της διαδικασίας ανάπτυξης. Ως εκ τούτου, αρχάριοι προγραμματιστές μπορούν να

καθώς και στο σχεδιασμό του προϊόντος. 6. Τακτική προσαρμογή στις μεταβαλλόμενες συνθήκες. 7. Ακόμη και οι καθυστερημένες αλλαγές όσον αφορά τις απαιτήσεις ενσωματώνονται στο τελικό προϊόν.	ενσωματωθούν στην ομάδα έργου μόνο σε συνδυσμό με έμπειρους προγραμματιστές.

Πίνακας 9 Πλεονεκτήματα και Μειονεκτήματα των Μοντέλων Ευέλικτης Ανάπτυξης

Η χρήση των Μοντέλων Ευέλικτης Ανάπτυξης ενδείκνυται για:

1. Έργα όπου συχνά νέες αλλαγές στις απαιτήσεις του συστήματος πρέπει να εφαρμοστούν και να ενσωματωθούν στο σύστημα.
2. Έργα όπου νέα χαρακτηριστικά (features) χρειάζεται να ενσωματωθούν στο σύστημα μέσα σε λίγες μέρες.
3. Έργα όπου οι τελικοί χρήστες/πελάτες δεν γνωρίζουν εξ'αρχής τί χρειάζονται από το σύστημα.

Η χρήση των Μοντέλων Ευέλικτης Ανάπτυξης δεν ενδείκνυται για:

1. Εφαρμογές λογισμικού οι οποίες δεν έχουν εμπορικό χαρακτήρα.
2. Έργα στα οποία δεν είναι απαραίτητη η συνεχής επικοινωνία και αλληλεπίδραση με τον τελικό χρήστη/πελάτη για την ανάπτυξη του λογισμικού.
3. Έργα των οποίων οι απαιτήσεις είναι σαφώς προκαθορισμένες και αμετάβλητες εξ'αρχής.
4. Έργα όπου οι τελικοί χρήστες/πελάτες δεν παρέχουν ευελιξία στην ομάδα ανάπτυξης του προϊόντος. Αυτό οφείλεται στο γεγονός ότι οι τελικοί χρήστες/πελάτες έχουν περιορισμένους πόρους τόσο όσον αφορά τα χρήματα που είναι διατεθειμένοι να δαπανήσουν όσο και στο χρονικό περιορισμό ολοκλήρωσης του έργου.

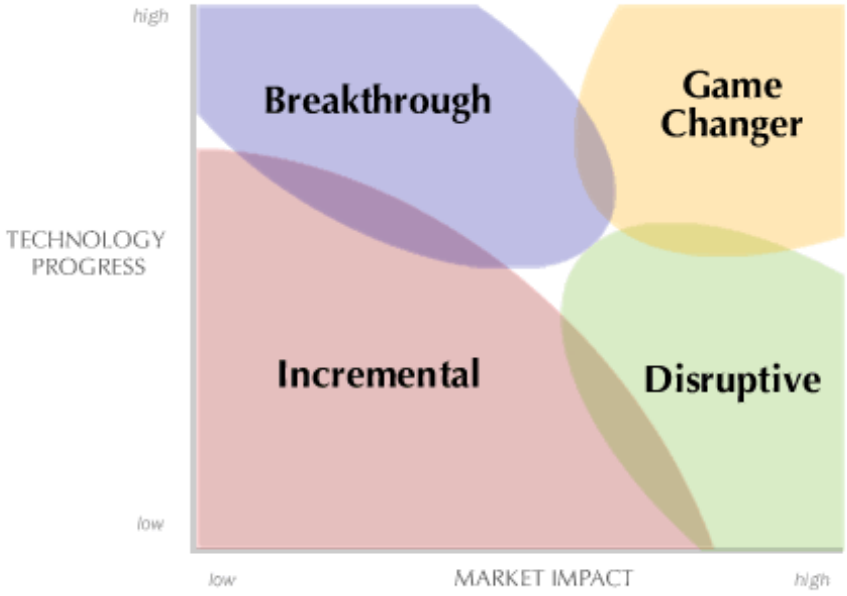
2.6 Μοντέλα Ανάπτυξης Λογισμικού και η σχέση τους με την Καινοτομία

Ένας από τους στόχους της παρούσας διπλωματικής εργασίας είναι να εντοπίσει τη σχέση μεταξύ των μοντέλων ανάπτυξης λογισμικού και της καινοτομίας.

Μιας και όρος καινοτομία μπορεί να ερμηνευθεί με πολλούς διαφορετικούς τρόπους, αρχικά παρουσιάζουμε τα διαφορετικά είδη καινοτομίας και στη συνέχεια προσδιορίζουμε τη σχέση μεταξύ είδους καινοτομίας και κατάλληλου μοντέλου ανάπτυξης.

Οι περισσότεροι ορισμοί διακρίνουν την καινοτομία σε δύο βασικές κατηγορίες. Για παράδειγμα , ο Michael Porter [17] μιλάει για ‘’συνεχείς’’ και ‘’ασυνεχείς’’ τεχνολογικές αλλαγές. Οι Tushman και Anderson [18] διακρίνουν την καινοτομία σε ‘’προοδευτική’’ *(incremental)* και ‘’επαναστατική’’ *(breakthrough)* . Οι Abernathy και Clark [19] διακρίνουν την καινοτομία σε συντηριτική (conservative) και ριζοσπαστική (radical).Τέλος, ο Christensen [20] διακρίνει την καινοτομία σε ‘’διατηρητέα’’ (sustaining) και ‘’διασπαστική’’ (disruptive).

Η καινοτομία όμως αποτελεί συνάρτηση περισσοτέρων παραγόντων. Ο παρακάτω πίνακας ταξινομεί την καινοτομία ως προς την επίδραση που έχει στην τεχνολογία καθώς και την επίδραση της καινοτομίας στη αγορά και διαχωρίζει την καινοτομία σε τέσσερις υποκατηγορίες.



Εικόνα 2-13 Είδη Καινοτομίας

- ***Προοδευτική (Incremental)***: Αναφέρεται σε καινοτομίες οι οποίες εμπεριέχουν βελτιωτικές αλλαγές σε ήδη υπάρχοντα προϊόντα και υπηρεσίες. Αυτό το είδος καινοτομίας καθιστά μια εταιρία ανταγωνιστική, μιας και προστήθενται νέα στοιχεία στο προϊόν της.
- ***Επαναστατική (Breakthrough)***: Αναφέρεται σε μεγάλες τεχνολογικές αλλαγές οι οποίες τοποθετούν το προϊόν σε πλεονεκτική θέση έναντι του ανταγωνισμού. Τέτοιου είδους καινοτομίες προέρχονται συνήθως μέσω έρευνας και ανάπτυξης (Research & Development) από εργαστήρια τα οποία στοχεύουν στο να ‘’πατεντάρουν’’ το μοντέλο τους, τη συσκευή ή και το προϊόν τους.
- ***Διασπαστική (Disruptive)***: Αναφέρεται σε καινοτομίες οι οποίες αλλάζουν τελείως την μορφή της αγοράς ή και δημιουργούν μία καινούργια. Ο Christensen ο οποίος εισήγαγε το συγκεκριμένο όρο κάνει λόγο για καινοτομίες οι οποίες στο βραχυπρόθεσμο μέλλον έχουν χαμήλη απόδοση αλλά παρέχουν μία τελείως διαφορετική πρόταση αξίας σε σχέση με τα υπάρχοντα προϊόντα. (π.χ. Uber, Transistor Radio)
- ***Ανατρεπτική (Game-Changing)***: Αναφέρεται σε καινοτομίες οι οποίες αλλάζουν τελείως την αγορά ακόμη και την ίδια την κοινωνία. Οι συγκεκριμένες καινοτομίες έχουν δραματική επίδραση στον τρόπο όπου οι

άνθρωποι δρούν, σκέφτονται και συμπεριφέρονται. (π.χ. Ford, ανακάλυψη εμβολίου για θεραπεία κάποιου είδους καρκίνου, κ.λ.π)

Μεθοδολογία Ανάπτυξης	Χρονολογία	Περιπτώσεις όπου χρησιμοποιείται	Συμβολή στην ανάπτυξη καινοτόμων προϊόντων
Waterfall	1970	Μεγάλα έργα με αυστηρά καθορισμένες και ξεκάθαρες απαιτήσεις (user and business requirements) Η ομάδα έργου αποτελείται από έμπειρους μηχανικούς λογισμικού π.χ. Military, aircraft and space aerospace systems	Incremental
Prototyping	1975	Εφαρμογές διεπαφής συστήματος - χρήστη Web-based εφαρμογές	Incremental & Disruptive
Incremental & Iterative Development	1985	Έργα στο οποία χρησιμοποιείται καινούργια τεχνολογία	Incremental & Breakthrough

		Έργα στα οποία το προϊόν πρέπει να βγεί γρήγορα στην αγορά --- artificial intelligent systems	
Spiral Model	Μέσα 1990's	Έργα όπου η υψηλή ακρίβεια των διαδικασιών είναι προτεραιότητα	Disruptive
Agile Development Processes	2001	Έργα όπου νέα χαρακτηριστικά (features) χρειάζεται να ενσωματωθούν στο σύστημα μέσα σε λίγες μέρες --- Έργα όπου νέες αλλαγές στις απαιτήσεις του συστήματος πρέπει να εφαρμοστούν και να ενσωματωθούν στο σύστημα	Incremental, Disruptive, Breakthrough & Game-changing

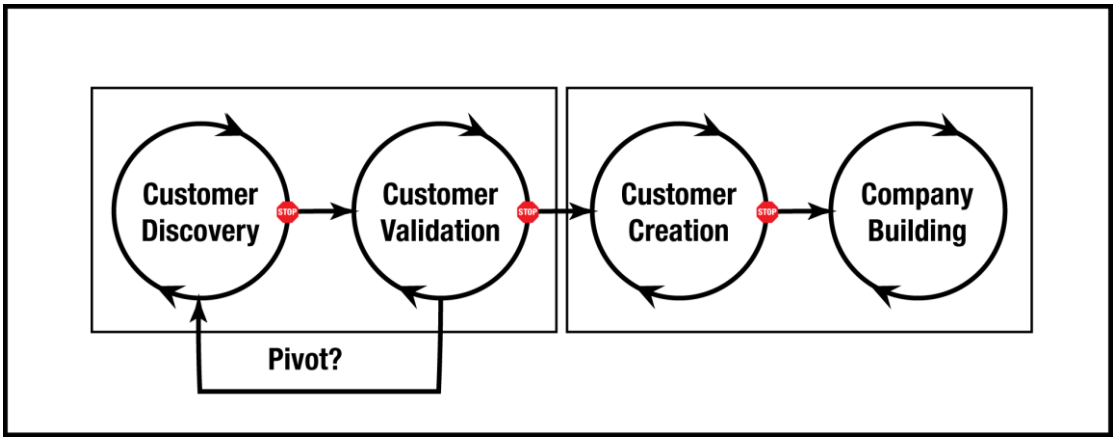
Πίνακας 10 Μοντέλα Ανάπτυξης Λογισμικού, Εφαρμογές Χρήσης και Καινοτομία

2.7 Μοντέλα Ανάπτυξης Λογισμικού και η σχέση τους με Μεθόδους Διοίκησης Καινοτόμων Ομάδων

Την τελευταία δεκαετία έχουν αναπτυχθεί αρκετές νέες μέθοδοι διοίκησης καινοτόμων ομάδων μιας και οι συνεχείς τεχνολογικές εξελίξεις έχουν οδηγήσει στη ραγδαία αύξηση των τεχνολογικών καινοτόμων επιχειρήσεων οι οποίες πρέπει να προσαρμόζονται στις συνεχείς αλλαγές του οικοσυστήματός τους. Για το λόγο αυτό έχουν αναπτυχθεί μοντέλα διοίκησης τα οποία κατά κύριο λόγο αναφερόνται σε

ομάδες οι οποίες χρησιμοποιούν ευέλικτες μεθόδους ανάπτυξης λογισμικού. Οι σημαντικότερες μέθοδοι διοίκησης καινοτόμων ομάδων είναι οι εξής:

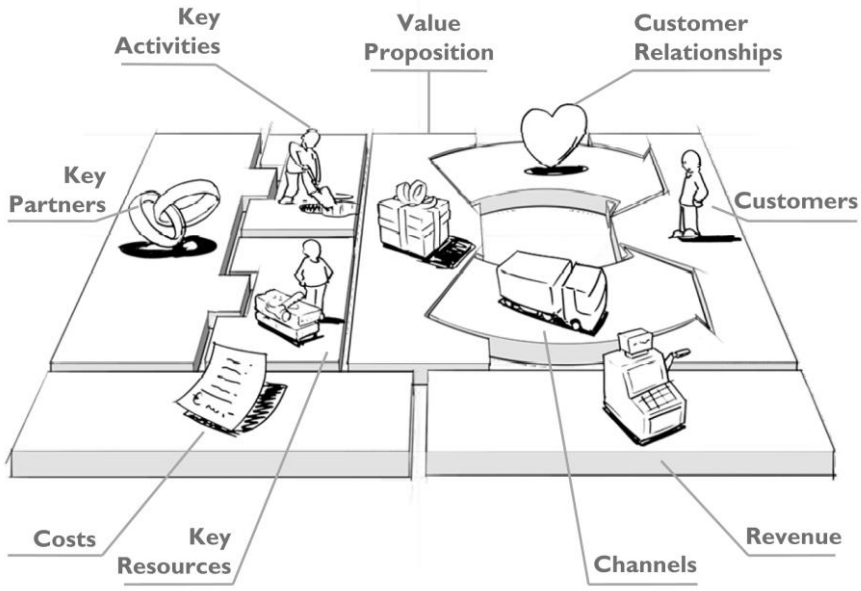
1. **Customer Development** [21]: Ο S.Blank στο βιβλίο του *''Four steps to the epiphany''* αναπτύσσει τη Customer Development μεθόδο [21], η οποία επικεντρώνεται στη δημιουργία μίας ομάδας έργου – της Customer Development Team- η οποία απαρτίζεται από μηχανικούς λογισμικού και στελέχη του εμπορικού τμήματος της ομάδας με στόχο την πληρέστερη κατανόηση των απαιτήσεων των πελατών τους και την προσαρμογή στις συνεχείς μεταβολλές. Η συγκεριμένη μέθοδος αποτελείται από τρία επαναλαμβανόμενα βήματα, τα οποία έχουν ως στόχο την πλήρη αποσαφήνιση των απαιτήσεων των αναγκών των πελατών και τον έλεγχο ότι το υπό ανάπτυξη σύστημα ικανοποιεί τις συγκεκριμένες απαιτήσεις. Τα συγκεκριμένα τρία βήματα (Customer Discovery, Customer Validation, Customer Creation) επαναλαμβάνονται μέχρι να βρεθεί πραγματική λύση για τις απαιτήσεις των πελατών και στη συνέχεια η ομάδα έργου επικεντρώνεται στην στελέχωση του τμήματος marketing και πωλήσεων. Τέλος, η συγκεκριμένη μέθοδος παρέχει αναλυτικές ροές εργασιών για κάθε ένα από τα επιμέρους βήματά της.



Εικόνα 2-14 Η μέθοδος Customer Development

2. **Business Model Canvas**: Οι A.Osterwalder και Y.Pingeur στο βιβλίο τους *''Business Model Generation''* [22] παρουσιάζουν το εργαλείο Business Model Canvas το οποίο περιγράφει τα βασικά τμήματα ενός οργανισμού. Το Business Model Canvas ενώ δεν αποτελεί ενά καθαρό εργαλείο διοίκησης καινοτόμων ομάδων, πολλές ομάδες, κυρίως σε νεοφυείς επιχειρήσεις, το

χρησιμοποιούν ως ένα μέσο επικοινωνίας και παρουσίασης νέων ιδεών, προσπαθώντας να συμπεριλάβουν κάθε αλλαγή που επηρεάζει τόσο το τεχνικό όσο και το εμπορικό κομμάτι της ομάδας.



Εικόνα 2-15 Business Model Canvas

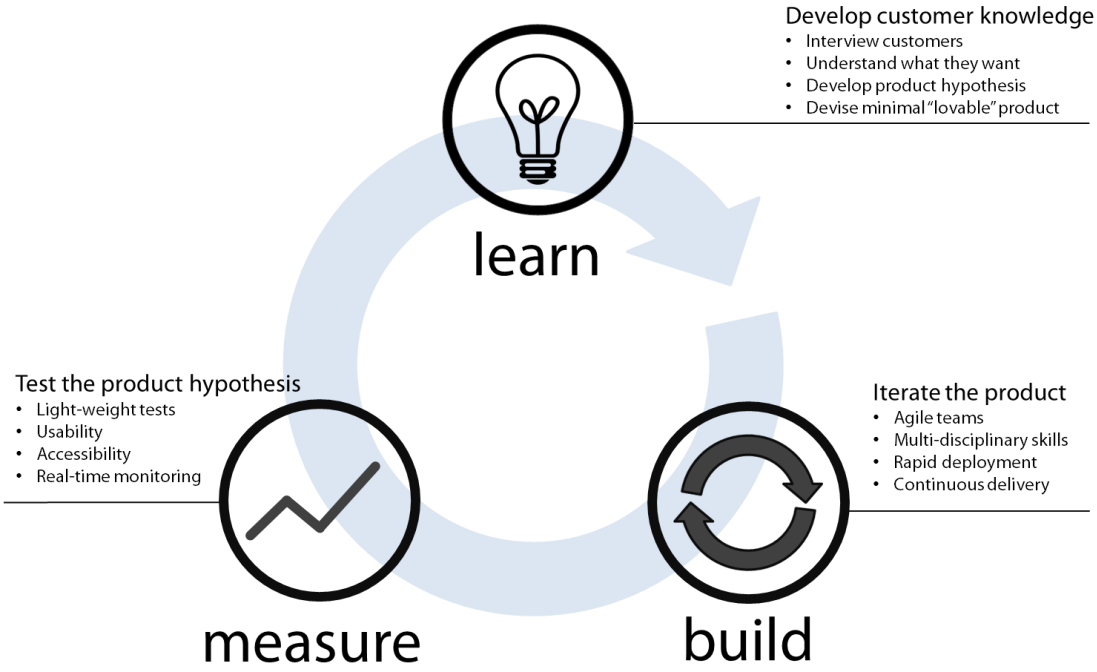
3. ***Value Proposition Canvas*** [23]: Αποτελεί μία παραλλαγή του *Business Model Canvas* [22] η οποία επικεντρώνεται στο να καταστήσει το προϊόν λογισμικού ελκυστικότερο στους πελάτες. Το Value Proposition Canvas διαχωρίζει τις ανάγκες των πελατών σε pains(-) και gains(+) και τις ταξινομεί με βάση τη σημαντικότητα της ανάγκης ως προς την ικανοποίηση των πελατών.
4. ***Empathy Map***[19]: Αποτελεί μία μέθοδο η οποία επικεντρώνεται στην κατανόηση των απαιτήσεων των πελατών από το σύστημα και την αλληλεπίδραση αυτών με το σύστημα με στόχο τη δημιουργία καλύτερων προϊόντων και υπηρεσιών.
5. ***Business Model Zen***[20]*:* Η συγκεκριμένη μέθοδος δημιουργήθηκε στην Κορέα το 2013. Πρόκειται για μία νέα μέθοδο μοντελοποίησης επιχειρηματικών διαδικασιών που βασίζεται στον προσανατολισμό της προς τον πελάτη, τη

19 http://www.entrepreneurial-insights.com/customer-profiling-using-empathy-map/

20 http://businessmodelzen.com/what-is-business-model-zen/

στρατηγική σκέψη και την επαναληπτικότητα των διαδικασιών. Το *Business Model Zen* παρέχει ένα ενιαίο πλαίσιο εργασίας όπου οι διάφορες διαδικασίες της επιχειρήσης διαμορφώνονται σταδιακά και διαχειρίζονται με προσοχή, από τη σύλληψη της ιδέας μέχρι το σχεδιασμό και την εκτέλεση.

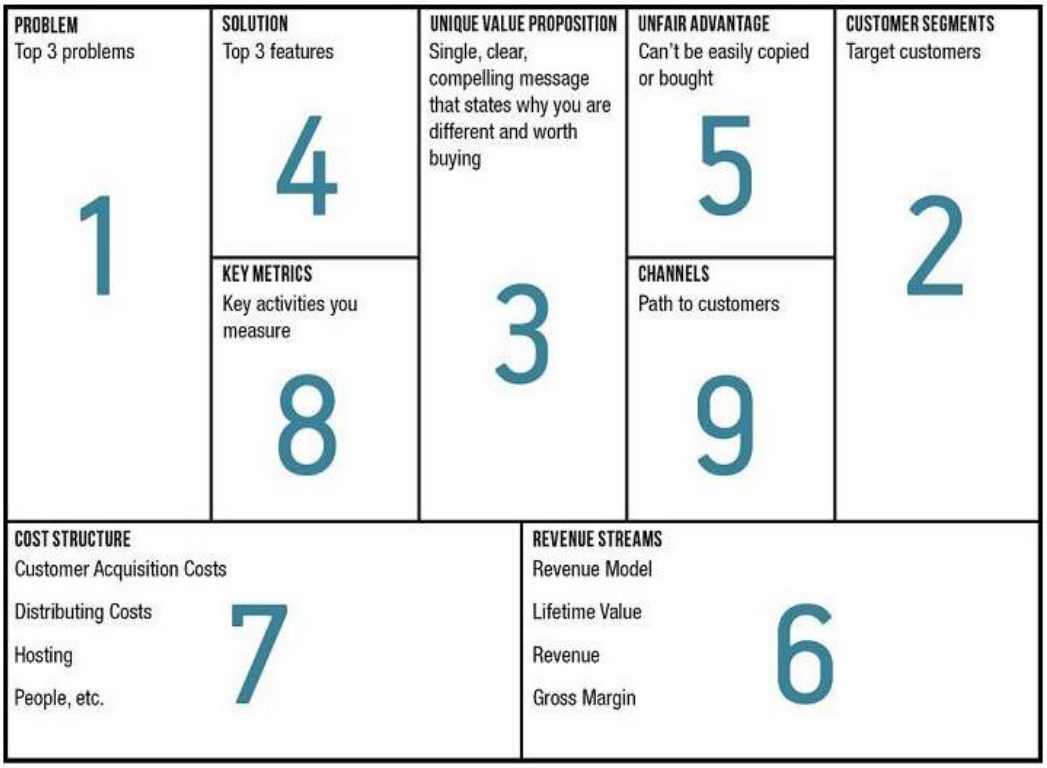
6. ***Lean Startup*** [24]: Ο Eric Ries το 2011 ανέπτυξε την μέθοδο Lean Startup η οποία αποτελεί μέθοδο διοίκησης καινοτόμων ομάδων και επικεντρώνεται στην μείωση των περιττών σπαταλών (χρόνου και χρήματος) μέσω της συνεχής επαναληπτικότητας κυκλικών διαδικασιών (*Learn, Build, Measure*) όσον αφορά το προϊόν ή την υπηρεσία που αναπτύσσει η ομάδα.



Εικόνα 2-16 Ο κύκλος διαδικασιών Lean Startup

Στο τέλος κάθε κύκλου η ομάδα αποφασίζει σύμφωνα με τα δεδομένα που έχει συλλέξει από τη διαδικασία ''Measure'' εάν χρειάζονται αλλαγές στην ιδέα που μόλις ανέπτυξαν ή εάν μπορούν να ξεκινήσουν την ανάπτυξη μίας καινούργιας ιδέας. Η συγκεκριμένη μέθοδος είναι ελλειπής όσον αφορά το επιχειρησιακό πλάνο της ομάδας σε κάθε στάδιο ανάπτυξης του λογισμικού. Για το λόγο αυτό έχει εκδοθεί μία σειρά από βιβλία τα, *''Lean Series''*, τα οποία περιγράφουν μεθόδους οι οποίες προσπαθούν να αποσαφηνίσουν πώς η Lean μέθοδος εφαρμόζεται σε κάθε στάδιο ανάπτυξης του λογισμικού. Οι μέθοδοι αυτοί είναι:

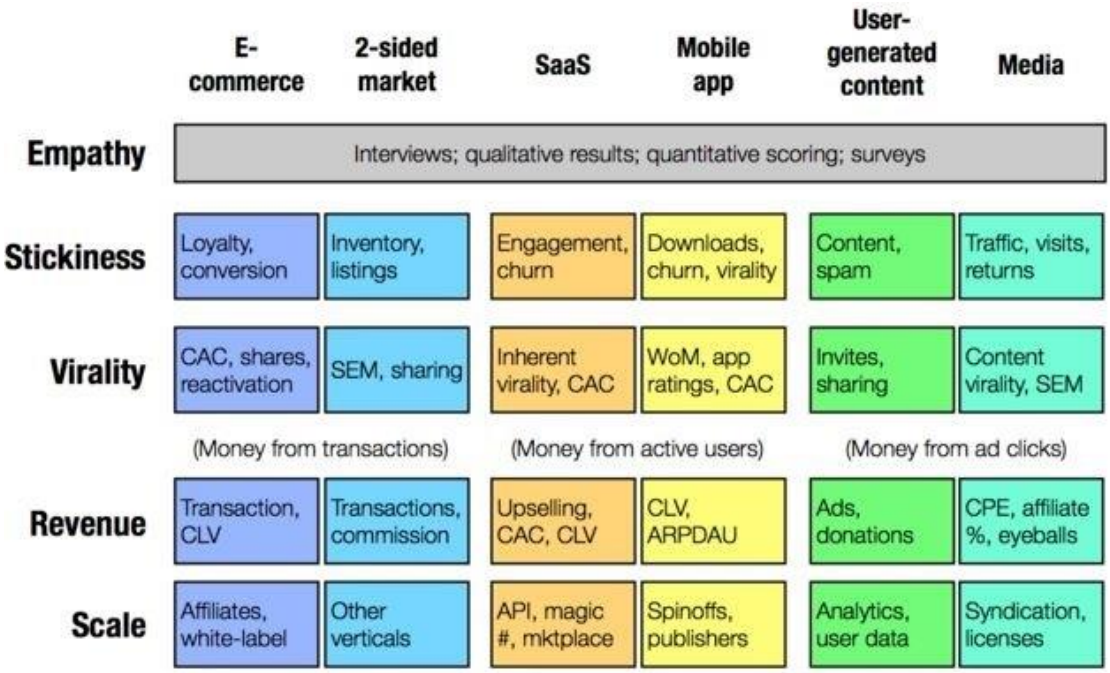
7. ***Running Lean*** [25]: Ο Ash Maurya το 2012 δημιούργησε την μέθοδο Running Lean στην οποία συνδύασε το Business Model Canvas [22] με τη Lean μέθοδο [24] και δημιούργησε το *Lean* Canvas.



Εικόνα 2-17 Lean Canvas

Το Lean Canvas βοηθάει τις ομάδες να επικεντρωθούν στις λύσεις που προσφέρει το υπό ανάπτυξη προϊόν τους στους πελάτες και στην προστηθέμενη αξία που τους παρέχουν, τα οποία αξιολογούνται μέσω μετρικών. Το Lean Canvas αποτελεί ένα εύχρηστο εργαλείο διοίκησης κυρίως στα πρώιμα στάδια μιας επιχειρηματικής καινοτόμας προσπάθειας.

8. ***Lean Analytics*** [26]: Οι Alistair Croll και Benjamin Yoskovitz το 2013 ανέπτυξαν τη συγκεκριμένη μέθοδο. Το Lean Analytics επικεντρώνεται κάθε φορά στη βελτιστοποίηση μιας μετρικής σε κάθε στάδιο ανάπτυξης του προϊόντος. Η συγκεκριμένη μέθοδος προσπαθεί να βοηθήσει την ομάδα να εντοπίσει σε κάθε στάδιο ανάπτυξης ποια είναι η σημαντικότερη μετρική και ποια τα KPIs (Key Performance Indicators) και παροτρύνει τις ομάδες να επαναλάβουν τη διαδικασία μέχρι να εντοπίσουν την μετρική την οποία πρέπει να βελτιστοποιήσουν σε αυτό το στάδιο.



Εικόνα 2-18 Lean Analytics

Το *Lean Analytics* αποτελείται από τα εξής στάδια: 1) *Empathy*: Η ομάδα προσπαθεί να κατανοήσει τις απαιτήσεις των πελατών μέσω ποιοτικών μεθόδων, 2) *Stickiness*: Η ομάδα προσπαθεί να προσδιορίσει ποιοι είναι πραγματικά οι πελάτες της, 3) *Virality*: Η ομάδα επικεντρώνεται στην ανάπτυξη στοιχείων τα οποία μπορούν να μεταδωθούν από στόμα σε στόμα, 4) *Revenue*: Η ομάδα προσπαθεί να δημιουργήσει στοιχεία τα οποία θα τις αποφέρουν έσοδα και 5) *Scale*: Η ομάδα προσπαθεί να δημιουργήσει στοιχεία τα οποία θα την μεγενθύνουν ποσοστικά.

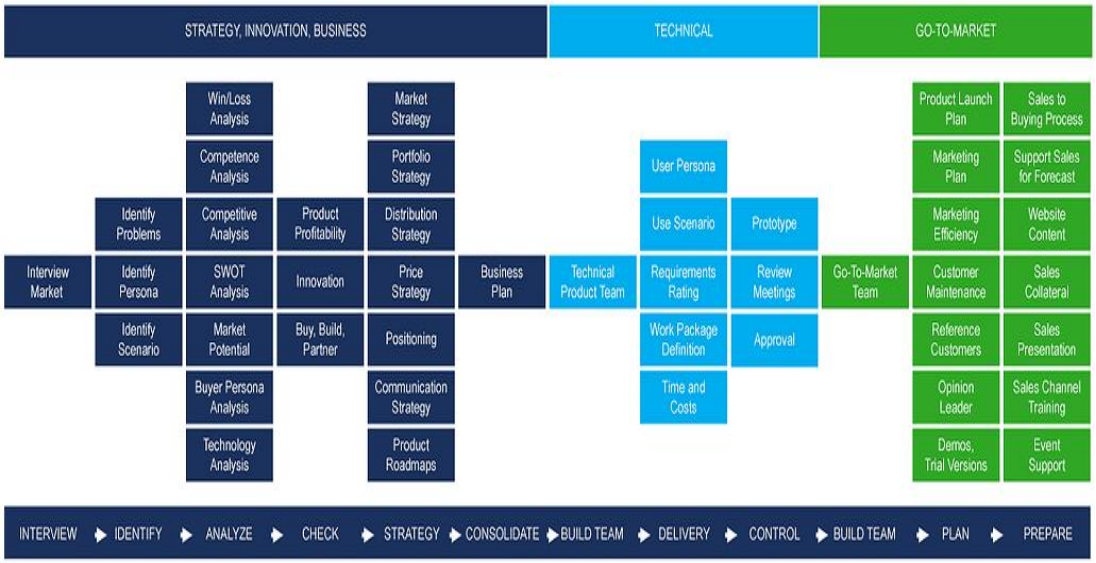
9. ***UX for Lean Startups*** [27]: Η Laura Klein ανέπτυξε τη συγκεκριμένη μέθοδο η οποία επικεντρώνεται στην εφαρμογή τεχνικών *Lean* κατά τη διαδικασία σχεδίασης UI και UX. Το βασικό στοιχείο της μεθόδου είναι η έμφαση στο χρήστη, ο προσδιορισμός του προβλήματος που καλείται να επιλύσει, η δημιουργία πιθανών λύσεων , η ανάπτυξη πρωτότυπου και ο έλεγχος του.
10. ***Design Thinking in Business***: Η συγκεκριμένη μέθοδος υποστηρίζει πως η σχεδίαση πρέπει να είναι το βασικό κριτήριο λήψης αποφάσεων για κάθε τμήμα της ομάδας. Υπάρχουν αρκετά βιβλία τα οποία παρουσιάζουν το πώς μπορεί να ενσωματωθεί το ‘’*Desing Thinking*’’ στις επιχειρησιακές διαδικασίες [28],[29]. Η βασική ιδέα της συγκεκριμένης μεθόδου είναι ότι οποιαδήποτε επικοινωνία, σχεδιασμός και διαδικασία λαμβάνει χώρα μέσα

στην ομάδα πρέπει να σχεδιαστεί και να παρουσιαστεί οπτικά. Χαρακτηριστικό παράδειγμα εταιρείας η οποία βασίζεται στο ''*Design Thinking''* είναι η Apple.

11. ***Holocracy***[21]: Πρόκειται για μία καινουργια μέθοδο διοίκησης καινοτόμων ομάδων η οποία επιδιώκει να μειώσει τον αριθμό των middle-level managers και να επιτρέψει σε όλους τους εργαζόμενους να συμμετέχουν στη λήψη αποφάσεων και στον τρόπο με τον οποία η ομάδα λειτουργεί ή ακομή και διοικείται. Η μέθοδος ''*Holocracy''* επιτρέπει στην ομάδα να λειτουργεί και να διοικείτα με βάση τη διαδικασία που λαμβάνει χώρα κάθε στιγμή. Όπως προαναφέρθηκε, πρόκειται για μία καινούργια μέθοδο και συνεπώς είναι ακόμη αρκετά νωρίς προκειμένου να αξιολογηθεί η επιτυχία εφαρμογής της.
12. ***Open Product Management Workflow***[22]: Πρόκειτα για μία ροή εργασίων όπου οι product managers μπορούν να αναφέρουν και να εντοπίσουν λεπτομερείς περιγραφές των καθηκόντων και των διαδικασιών της ομάδας. Η κύρια ροή εργασίας αποτελείται από: συντεύξεις, εντοπισμό, ανάλυση, έλεγχο απαιτούμενο για τη χάραξη στρατηγικής, παγίωση του επιχειρηματικού σχεδίου, δημιουργία της τεχνικής ομάδας, δημιουργία της εμπορικής ομάδας, σχεδιασμό και προετοιμασία. Η συγκεκριμένη μέθοδος είναι αρκετά χρήσιμη για την ανάλυση διαδικασιών σε συνδυασμό με τις δραστηριότητες διαχείρησης του προϊόντος.

[21] http://www.holacracy.org/

[22] http://open-pmw.org/



Εικόνα 2-19 Open Product Management Workflow[23]

Τέλος, έπειτα από την μελέτη που πραγματοποιήθηκε παρουσιάζουμε ένα συγκεντρωτικό πίνακα με τις μεθοδολογίες ανάπτυξης λογισμικού και τις μεθόδους διοίκησης καινοτόμων ομάδων οι οποίες μπορούν να τις υποστηρίξουν.

Μέθοδοι Διοίκησης Καινοτόμων Ομάδων	Μέθοδολογίες Ανάπτυξης Λογισμικού
Customer Development	V-model, Agile Methods
Business Model Canvas	V-model, Prototyping, Agile Methods
Lean Startup	Agile Methods
Design Thinking	Incremental & Iterative, Prototyping, Agile Methods
Holocracy	Agile Methods
Open Product Management Workflow	Waterfall, V-model, Spiral, Incremental & Iterative, Agile Methods

Πίνακας 11 Συμπερασματικός πίνακας μεθόδων διοίκησης καινοτόμων ομάδων και μεθόδων ανάπτυξης λογισμικού

[23] http://open-pmw.org/wiki/index.php/Open_Product_Management_Workflow/

3 *Απαιτήσεις Συστήματος Λογισμικού*

Έχοντας εντοπίσει από σχετικές έρευνες όπως αυτές αναφέρονται στο κεφάλαιο 1.2 πώς το βασικότερο πρόβλημα αποτυχίας έργων λογισμικού είναι η ασαφεια ως προς τις απαιτήσεις και τις αναγκές των χρηστών, στο παρόν κεφάλαιο μελετήσαμε τη σχετική βιβλιογραφία που αφορά τις απαιτήσεις ενός προϊόντος λογισμικού τόσο σε τεχνικό όσο και σε επιχειρηματικό επίπεδο.

Στο προηγούμενο κεφάλαιο, σε κάθε διαφορετικό διαδικαστικό μοντέλο ανάπτυξης λογισμικού σημειώνονταν διαφορετικά βήματα-''κλειδιά'' για την επιτυχή ανάπτυξη λογισμικού. Ειδικότερα, κάθε προτεινόμενο διαδικαστικό μοντέλο περιλαμβάνει διαφορετικές δραστηριότητες με στόχο τη συλλογή των απαιτήσεων/προδιαγραφών του συστήματος: την κατανόηση των θεμελιωδών αναγκών και προβλημάτων των τελικών χρηστών/πελατών. Μία απαίτηση μπορεί να είναι οτιδήποτε, από μια υψηλού επιπέδου αφηρημένη δήλωση μιας υπηρεσίας ή ενός περιορισμού του συστήματος μέχρι ένας λεπτομερής, μαθηματικός ορισμός μιας λειτουργίας του συστήματος. Συνεπώς, η διαδικασία προκειμένου να κατανοήσουμε τον σκοπό αλλά και τη λειτουργία του λογισμικού ξεκινάει με την εξέταση των απαιτήσεων/προδιαγραφών.Σε αυτό το κεφάλαιο, εξετάζουμε τα διαφόρα είδη απαιτήσεων, τις πηγές αυτών καθώς και τις αλληλεπιδράσεις μεταξύ τους.

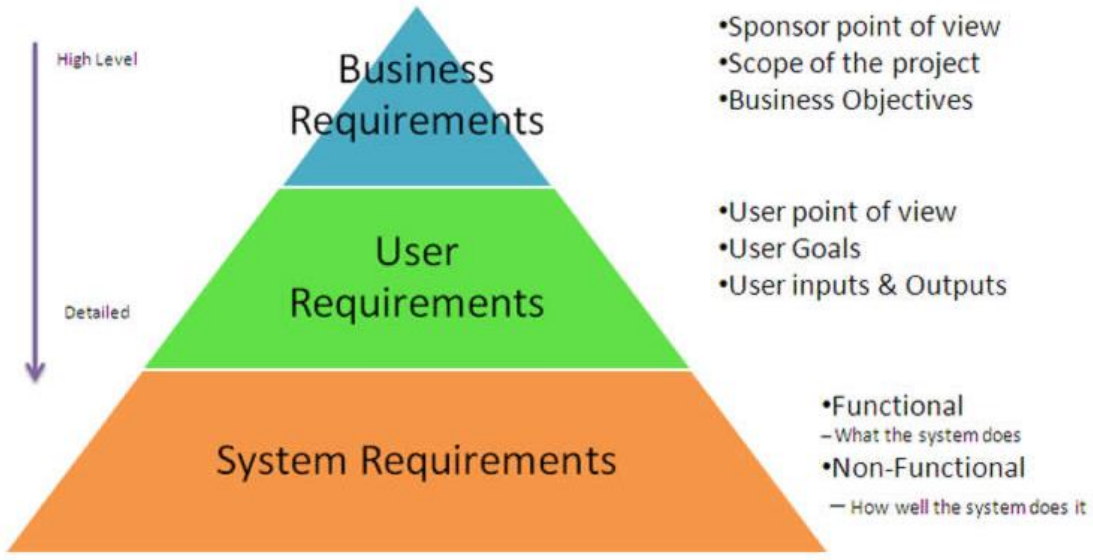
3.1 Είδη Απαιτήσεων

Με τον όρο απαίτησεις ή προδιαγραφές συστήματος εννοούμε οποιαδήποτε επιθυμητή συμπεριφορά προσδοκούμε να έχει το σύστημα το οποίο καλούμαστε να δημιουργήσουμε [30]. Οι απαιτήσεις δεν ασχολούνται μόνο με τα χαρακτηριστικά του συστήματος αλλά και με τους ανθρώπους οι οποίοι αλληλεπιδρούν με το

σύστημα (εμπλεκόμενους). Συμπερασματικά, οι απαιτήσεις αποτελούν συγκεκριμένες λειτουργίες ή χαρακτηριστικά του συστήματος

Υπάρχουν τρία βασικά είδη απαιτήσεων:

1. **Εταιρικές απαιτήσεις (Business Requirements)**: Προσδιορίζουν τον σκοπό του έργου, τους επιχειρηματικούς στόχους και τις ενέργειες που πρέπει να γίνουν προκειμένου να προστεθεί αξία στην επιχείρηση μέσω της δημιουργίας ενός καινοτόμου προϊόντος λογισμικού. Συνήθως στην τεχνολογία λογισμικού συμπεριλαμβάνονται στις ‘’ποιοτικές’’, μη-λειτουργικές απαιτήσεις, οι οποίες θα αναφερθούν παρακάτω, και πολλές φορές αναφέρονται σε περιορισμούς που πρέπει να έχει το σύστημα, όπως απόδοση, ασφάλεια κ.ο.κ στο επιχειρησιακό περιβάλλον.
2. **Απαιτήσεις χρήστη (User Requirements)**: Δηλώσεις σε φυσική γλώσσα και διαγράμματα των υπηρεσιών που παρέχει το σύστημα και των λειτουργικών περιορισμών του. Αναφέρονται στους πελάτες.
3. **Απαιτήσεις συστήματος (System Requirements)**: Ένα δομημένο έγγραφο που περιγράφει με λεπτομέρειες τις λειτουργίες, τις υπηρεσίες, και τους λειτουργικούς περιορισμούς του συστήματος. Ορίζει με ακρίβεια τι πρέπει να υλοποιηθεί ώστε να αποτελεί μέρος της σύμβασης μεταξύ πελάτη και αναδόχου.



Εικόνα 3-1 Είδη Απαιτήσεων

Ακόμη, υπάρχουν διαφορετικοί αναγνωστές των προαναφερθέντων τύπων απαιτήσεων:

Απαιτήσεις χρήστη: Διευθυντές του πελάτη, τελικοί χρήστες του συστήματος, μηχανικοί λογισμικού του πελάτη, αρχιτέκτονες του συστήματος.

Απαιτήσεις συστήματος: Τελικοί χρήστες του συστήματος, μηχανικοί λογισμικού του πελάτη, αρχιτέκτονες του συστήματος, κατασκευαστές λογισμικού.

Ο διαχωρισμός των απαιτήσεων του συστήματος γίνεται με βάση τη λειτουργικότητά τους [31]. Στη συγκεκριμένη περίπτωση εχουμε δύο τύπους απαιτήσεων: τις λειτουργικές και τις μη-λειτουργικές απαιτήσεις.

1. **Λειτουργικές Απαιτήσεις (Functional Requirements)**: Δηλώσεις που ορίζουν ποιες υπηρεσίες θα πρέπει να παρέχει το σύστημα, πώς θα πρέπει να αντιδρά σε συγκεκριμένες εισόδους και πώς θα πρέπει να συμπεριφέρεται σε συγκεκριμένες καταστάσεις. Οι λειτουργικές απαιτήσεις :
   - Περιγράφουν λειτουργικές δυνατότητες ή υπηρεσίες του συστήματος.
   - Εξαρτώνται από τον τύπο του λογισμικού, από τους αναμενόμενους χρήστες του λογισμικού και από τον τύπο του συστήματος στον οποίο χρησιμοποιείται το λογισμικό.
   - Οι λειτουργικές απαιτήσεις χρήστη μπορεί να είναι υψηλού επιπέδου δηλώσεις των δυνατοτήτων του συστήματος, αλλά οι λειτουργικές απαιτήσεις του συστήματος πρέπει να περιγράφουν με λεπτομέρειες τις υπηρεσίες του συστήματος.
2. **Μη-λειτουργικές Απαιτήσεις (Non-functional Requirements)**: Περιορισμοί στις υπηρεσίες ή στις λειτουργίες που προσφέρει το σύστημα, όπως χρονικοί περιορισμοί, περιορισμοί της διαδικασίας ανάπτυξης, πρότυπα, κ.λπ.

Οι μη-λειτουργικές απαιτήσεις:

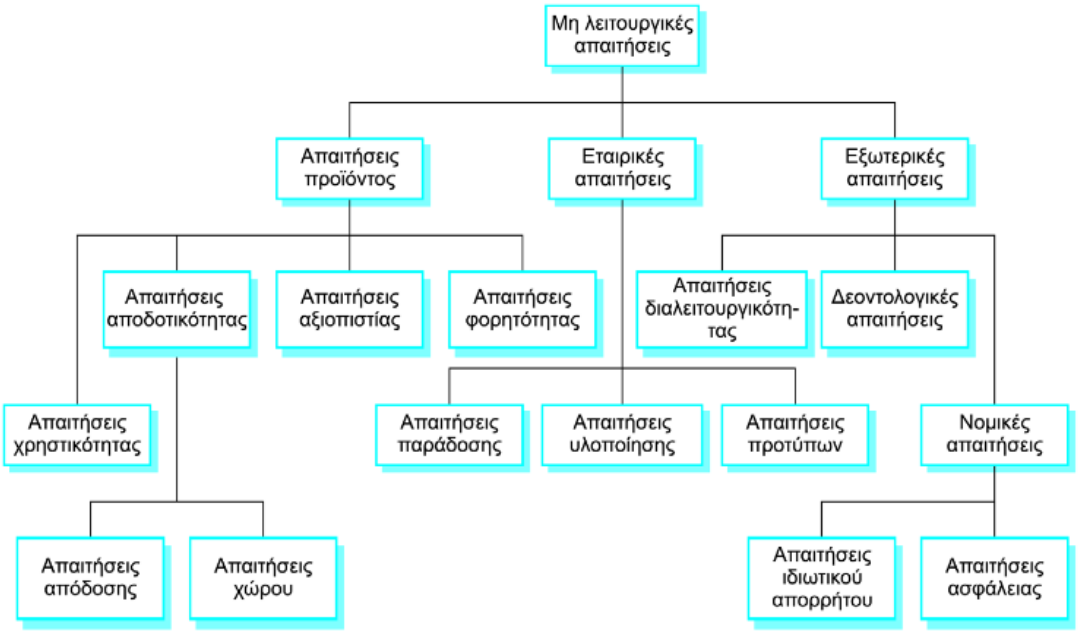
- Ορίζουν ιδιότητες και περιορισμούς του συστήματος, για παράδειγμα την αξιοπιστία, το χρόνο απόκρισης και τις απαιτήσεις σε αποθηκευτικό χώρο. Περιορισμοί μπορεί να είναι οι δυνατότητες των συσκευών εισόδου - εξόδου, οι αναπαραστάσεις του συστήματος, κ.λπ.

- Μπορεί να καθορίσουν απαιτήσεις διαδικασιών οι οποίες θα επιτάσσουν ένα συγκεκριμένο σύστημα CASE, μια συγκεκριμένη γλώσσα προγραμματισμού ή μέθοδο ανάπτυξης.
- Οι μη λειτουργικές απαιτήσεις μπορεί να είναι πιο κρίσιμες από τις λειτουργικές. Αν οι πρώτες δεν πληρούνται, το σύστημα είναι άχρηστο.

Οι μη-λειτουργικές απαιτήσεις διακρίνονται στις εξής κατηγορίες:

- Απαιτήσεις προϊόντος: Απαιτήσεις που καθορίζουν τη συμπεριφορά του τελικού προϊόντος, για παράδειγμα, την ταχύτητα εκτέλεσης, την αξιοπιστία, κ.λπ.
- Εταιρικές απαιτήσεις: Απαιτήσεις που πηγάζουν από την εταιρική πολιτική και τις εταιρικές διαδικασίες, για παράδειγμα, τα πρότυπα διαδικασιών που πρέπει να χρησιμοποιηθούν, οι απαιτήσεις της υλοποίησης, κ.λπ.
- Εξωτερικές απαιτήσεις: Απαιτήσεις που προέρχονται από παράγοντες εξωτερικούς προς το σύστημα και τη διαδικασία ανάπτυξής του, για παράδειγμα, απαιτήσεις διαλειτουργικότητας, νομικές απαιτήσεις, κ.λπ.

Το παρακάτω σχήμα παρουσιάζει τους διαφορετικούς τύπους των μη-λειτουργικών απαιτήσεων:



Εικόνα 3-2 Τύποι Μη-λειτουργικών Απαιτήσεων

Τέλος, οι απαιτήσεις προκειμένου να ελέγχονται για την ορθότητα και την αποτελεσματικότητά τους πρέπει να έχουν συγκεκριμένες μετρικές όπου θα παρέχουν ποσοτικά μεγέθη για τα διάφορα χαρακτηριστικά τους. Ο παρακάτω πίνακας παρουσιάζει τις βασικές ιδιότητες των απαιτήσεων και τις μετρικές τους.

Ιδιότητα	**Μετρική**
Ταχύτητα	Συναλλαγές/δευτερόλεπτο Χρόνος απόκρισης χρήστη/συμβάντος Χρόνος ανανέωσης οθόνης
Μέγεθος	K byte Αριθμός τσιπ μνήμης RAM
Ευχρηστία	Χρόνος εκπαίδευσης Αριθμός πλαισίων βοήθειας
Αξιοπιστία	Μέσος χρόνος μεταξύ αστοχιών Πιθανότητα μη διαθεσιμότητας Συχνότητα εμφάνισης αστοχιών Διαθεσιμότητα
Ανθεκτικότητα	Χρόνος επανεκκίνησης μετά από αστοχία Ποσοστό συμβάντων που προκαλούν αστοχία Πιθανότητα βλάβης δεδομένων από αστοχία
Φορητότητα	Ποσοστό εντολών που εξαρτώνται από το σύστημα προορισμού Αριθμός συστημάτων προορισμού

Πίνακας 12 Μετρικές Απαιτήσεων

3.2 Η διαδικασία της συλλογής απαιτήσεων

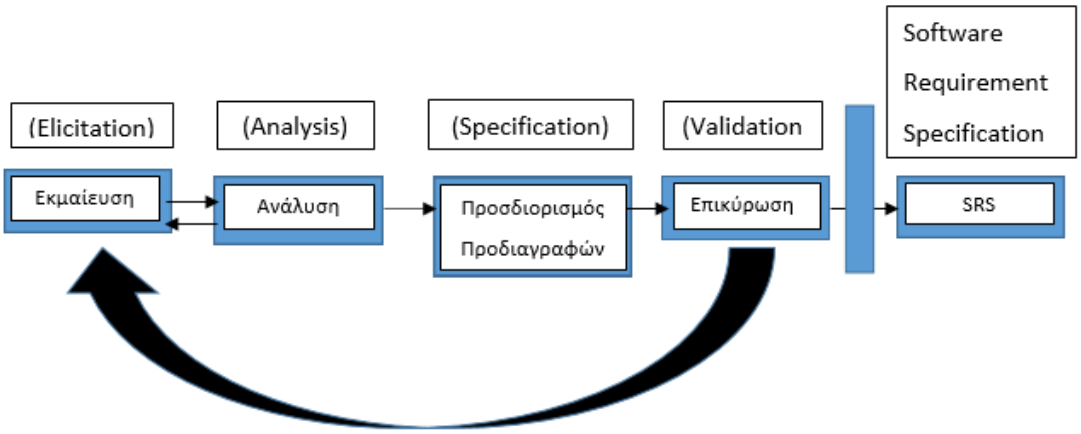
Όταν ένας πελάτης ζητήσει από μία ομάδα μηχανικών λογισμικού να φτιάξει ένα νέο σύστημα, ο πελάτης έχει μία βασική ιδέα για το τι πρέπει να κάνει το σύστημα. Συχνά, ο πελάτης θέλει να αυτοματοποιήσει μία χειρονακτική διαδικασία, όπως την πληρωμή των λογαριασμών ηλεκτρονικά από ό, τι με χειρόγραφους ελέγχους. Μερικές φορές, ο πελάτης θέλει να βελτιώσει ή να επεκτείνει μια τρέχουσα χειρονακτική διαδικασία ή ένα αυτοματοποιημένο σύστημα. Όλο και πιο συχνά, ένας πελάτης θέλει προϊόντα λογισμικού που κάνουν πράγματα τα οποία είναι καινοτόμα

και δεν προϋπήρχαν. Ασχέτως από το εάν η λειτουργικότητά του λογισμικού είναι καινοτόμα ή όχι, το προτεινόμενο σύστημα λογισμικού έχει ένα σκοπό, ο οποίος συνήθως εκφράζεται σε σχέση με τους στόχους ή επιθυμητή συμπεριφορά του συστήματος.

Ο σκοπός της συλλογής απαιτήσεων δεν είναι η ανάλυση των τεχνικών χαρακτηριστικών του συστήματος ,εκτός και εάν αυτό απαιτείται από τους τελικούς χρήστες/πελάτες, αλλά η κατανόηση των αναγκών των πελατών και των προβλημάτων που επιδιώκουν να λύσουν με το συγκεκριμένο προϊόν λογισμικού. Συχνά στη βιβλιογραφία αναφέρεται ότι οι απαιτήσεις στα διαδικαστικά μοντέλα ανάπτυξης προϊόντων λογισμικού καταδεικνύουν τη συμπεριφορά/λειτουργικότητα που επιθυμεί ο πελάτης από το σύστημα και όχι πως αυτή η λειτουργικότητα θα εφαρμοστεί στην πράξη. Οποιαδήποτε συζήτηση για τον τρόπο λειτουργίας του λογισμικού και την πρακτική εφάρμογή του στη φάση συλλογής απαιτήσεων κρίνεται ατυχής εφόσον ακόμα δεν υπάρχει πλήρης γνώση για το πρόβλημα το οποίο πρόκειται να επιλύθεί μέσω του συστήματος που η ομάδα έργου καλείται να αναπτύξει.

Η διαδικασία συλλογής των απαιτήσεων ενός συστήματος αποτελείται από τέσσερα βασικά στάδια [3], [31]:

1. **Εκμαίευση (Elicitation)**: Σε αυτό το στάδιο η ομάδα έργου συλλέγει τις απαιτήσεις των τελικών χρηστών/πελατών καθώς επίσης και τις απαιτήσεις/στόχους της εταιρείας.
2. **Ανάλυση/Επεξεργασία (Analysis)**: Σε αυτό το στάδιο η ομάδα έργου προσπαθεί να κατανοήσει τις απαιτήσεις των πελατών και να μοντελοποιήσει την επιθυμητή λειτουργία του συστήματος με βάση τις απαιτήσεις τους.
3. **Προσδιορισμός (Specification)**: Σε αυτό το στάδιο η ομάδα έργου προσπαθεί να τεκμηριώσει τη λειτουργικότητα του προτεινόμενου συστήματος λογισμικού.
4. **Επικύρωση (Validation)**: Σε αυτό το στάδιο η ομάδα έργου ελέγχει ότι οι τεκμηριώσεις (specifications) του συστήματος ταιριάζουν με τις αρχικές απαιτήσεις των τελικών χρηστών/πελατών και της εταιρείας.



Εικόνα 3-3 Η διαδικασία συλλογής απαιτήσεων

Η εικόνα 3-3 απεικονίζει την διαδικασία της συλλογής των απαιτήσεων για ένα προτεινόμενο σύστημα λογισμικού.

Η συγκεκριμένη διαδικασία χρησιμοποιείται προκειμένου η ομάδα έργου να κατανοήσει σε βάθος τις πραγματικές απαιτήσεις των τελικών χρηστών, επειδή πολλές φορές οι απαιτήσεις που αναφέρονται από τους χρήστες δεν αποτελούν και τις πραγματικές απαιτήσεις αυτών από το σύστημα.Το άτομο το οποίο ασχολείται με τη συγκεκριμένη διαδικασία από μία ομάδα έργου συνήθως ονομάζεται ‘’Αναλυτής Απαιτήσεων’’ (Requirements Analyst) ή ‘’Αναλυτής Συστήματος’’(Systems Analyst).Ο αναλυτής απαιτήσεων συνεργάζεται αρχικά με τους πελάτες προκειμένου να εξάγει τις απαιτήσεις τους χρησιμοποιώντας διάφορες τεχνικές(π.χ. ερωτήσεις, παρουσίαση παρόμοιων προϊόντων λογισμικού κ.ο.κ). Μέσω αυτών των τεχνικών ο αναλυτής καταλαβαίνει καλύτερα τη λειτουργικότητα που επιθυμούν οι πελάτες από το τελικό σύστημα και προσπαθεί να δημιουργήσει ένα αρχικό προτότυπο. Όταν οι απαιτήσεις των χρηστών γίνουν πλήρως κατανοητές από τον αναλυτή, ξεκινάει το στάδιο του προσδιορισμού των απαιτήσεων του συστήματος, δηλαδή ποιές από τις επιθυμητές απαιτήσεις των χρηστών θα εφαρμοστούν στο λογισμικό.Τα στάδια της ανάλυσης και της επικύρωσης ενέχουν πολλούς κινδύνους με αποτέλεσμα να χρειαστούν πολλές επιπλέον συναντήσεις με τους πελάτες και να επαναξεταστούν τα μοντέλα που αποφασίστηκαν για τον καθορισμό των απαιτήσεων. Το τελικό αποτέλεσμα από τη διαδικασία συλλογής απαιτήσεων ονομάζεται ‘’Προδιαγραφές Απαιτήσεων Λογισμικού’’ (Software Requirements Specification, SRS) και αποτελεί

ένα έγγραφο το οποίο δημιουργείται όταν μια λεπτομερής περιγραφή όλων των πτυχών του λογισμικού πρέπει να καθοριστέι πριν αρχίσει η σχεδίαση. Είναι σημαντικό να υπογραμμιστεί ότι ένα τυπικό SRS δεν είναι πάντα γραπτό. Στην πραγματικότητα, υπάρχουν πολλές περιπτώσεις στις οποίες η προσπάθεια που καταβλήθηκε για το SRS θα ήταν καλύτερο να είχε καταβληθεί σε άλλες δραστηριότητες της τεχνολογίας λογισμικού. Εντούτοις, όταν το λογισμικό πρόκειται να αναπτυχθεί από κάποιο τρίτο πρόσωπο, όταν η έλλειψη των προδιαγραφών θα μπορούσε να δημιουργήσει σοβαρά επιχειρησιακά προβλήματα ή όταν ένα σύστημα είναι εξαιρετικά πολύπλοκο ή επιχειρησιακά κρίσιμο, ένα SRS απαιτείται να δημιουργηθεί.

3.2.1 Εκμαίευση Απαιτήσεων (Requirements Elicitation)

Η εκμαίευση των απαιτήσεων είναι ένα από τα σημαντικότερα στάδια στη διαδικασία του καθορισμού των απαιτήσεων. Η ομάδα έργου οφείλει να χρησιμοποιήσει πολλές και διαφορετικές τεχνικές προκειμένου να αποφανθεί για το τι πραγματικά θέλουν οι τελικοί χρήστες/πελάτες και πολλές φορές είναι χρήσιμο η ομάδα έργου να δουλευεί σε συνεχή συννενόηση με τους πελάτες προκειμένου να καταλάβει σε βάθος το πρόβλημα που επικαλείται να λύσει. Στην αρχή του έργου, οι απαιτήσεις δεν είναι σαφώς αντιληπτές ούτε από την ομάδα έργου ούτε από τους ίδιους τους πελάτες και συχνά οι πελάτες δεν μπορούν να εξηγήσουν στην ομάδα έργου με σαφή τρόπο τι είναι αυτό που χρειάζονται από το σύστημα.Οι Christel και Kang [32] προσδιορίζουν έναν αριθμό προβλημάτων που συναντώνται στη διαδικασία εκμαίευσης των απαιτήσεων:

- **Προβλήματα στο πεδίο εφαρμογής**: Τα όρια του συστήματος είναι ασαφή ή οι πελάτες/χρήστες καθορίζουν περιττές τεχνικές λεπτομέρειες που πιθανόν να προκαλέσουν σύγχυση, παρά να διευκρινίσουν τους γενικούς στόχους του συστήματος.
- **Προβλήματα κατανόησης**: Οι πελάτες/χρήστες δεν είναι απόλυτα σίγουρα για το τι χρειάζονται, ίσως έχουν μια ανεπαρκή αντίληψη των δυνατοτήτων και των περιορισμών του υπολογιστικού περιβάλλοντος τους, δεν έχουν μια

πλήρη κατανόηση του τομέα του προβλήματος, προβληματίζονται για τις ανάγκες επικοινωνίας του τεχνολογικού συστήματος, παραλείπουν πληροφορίες που πιστεύουν ότι είναι ‘’προφανείς’’, ορίζουν απαιτήσεις που έρχονται σε αντίθεση με τις ανάγκες άλλων πελατών/χρηστών ή προσδιορίζουν απαιτήσεις που είναι διφορούμενες ή που δεν έχουν επαληθευτεί.

- **Προβλήματα μεταβλητότητας**: Οι απαιτήσεις αλλάζουν με την πάροδο του χρόνου. Προκειμένου να ξεπεραστούν τέτοιου είδους προβλήματα, η διαδικασία της συλλογής απαιτήσεων πρέπει να γίνεται με σαφή και οργανωμένο τρόπο.

Συνεπώς, όλοι οι εμπλεκόμενοι/ενδιαφερόμενα μέλη (stakeholders) πρέπει να βρίσκονται σε συνεχή επικοινωνία προκειμένου να καταλήξουν ομαδικά σε μία κοινή απόφαση για το ποιές είναι τελικά οι απαιτήσεις/προδιαγραφές του συστήματος.

3.2.1.1 Εμπλεκόμενοι/Ενδιαφερόμενα Μέλη (Stakeholders)

Οι Sommerville και Sawyer [31] ορίζουν τους εμπλεκόμενους ως ‘’ *οποιοσδήποτε που επωφελείται με έναν άμεσο ή έμμεσο τρόπο από το σύστημα που αναπτύσσεται.*’’

Τα ενδιαφερόμενα μέλη ενός έργου συμβάλλουν στον καθορισμό των απαιτήσεων του συστήματος και μπορεί να είναι οι εξής [30]:

- Σπόνσορες: Οι σπόνσορες είναι αυτοί που πληρώνουν για το λογισμικό που θα αναπτυχθεί. Το γεγονός ότι πληρώνουν για την ανάπτυξη του λογισμικού, τους καθιστά αυτομάτως ως τα ‘’ισχυρότερα’’ ενδιαφερόμενα μέλη και έχουν τον τελευταίο λόγο για τη λειτουργία του λογισμικού.
- Πελάτες: Είναι τα άτομα εκείνα τα οποία αγοράζουν το λογισμίκο αφού αυτό έχει αναπτυχθεί και είναι έτοιμο για χρήση. Πολλές φορές οι πελάτες και οι τελικοί χρήστες είναι τα ίδια άτομα, ενώ άλλες φορές ο πελάτης μπορεί να είναι ένα στέλεχος επιχειρήσεων ο οποίος ενδιαφέρεται να βελτιώσει την αποδοτικότητα των εργαζομένων του. Η ομάδα έργου οφείλει να κατανοήσει σε βάθος τις ανάγκες των πελατών προκειμένου να ανάπτυξει ένα προϊόν λογισμικού το οποίο

οι πελάτες θα είναι πρόθυμοι να αγοράσουν και παράλληλα θα τους προσδώσει αξία στην εργασία τους.

- Τελικοί χρήστες: Είναι τα άτομα εκείνα τα οποία θα εργασθούν με το λογισμικό το οποίο πρόκειται να αναπτυχθεί. Είναι σημαντικό να αναφερθεί ότι η ομάδα έργου πρέπει να διαχωρήσει τους τελικούς χρήστες σε διαφορετικές ομάδες, μιας και σε πολλές περιπτώσεις υπάρχουν ομάδες χρηστών διαφορετικών χαρακτηριστικών(χρήστες με ελάχιστη εξοικίωση στην χρήση Η/Υ, έμπειροι χρήστες, χρήστες με αναπηρίες, κλπ.).
- Ειδικοί του χώρου (Domain experts): Είναι εκείνα τα άτομα τα οποία βρίσκονται χρόνια στο χώρο και μπορούν να συνεισφέρουν στις απαιτήσεις/προδιαγραφές του λογισμικού, αφού γνωρίζουν πολύ καλά τους κινδύνους που ελλοχεύουν στην ανάπτυξη προϊόντων λογισμικού του συγκεκριμένου χώρου.
- Δικηγόροι: Οι δικηγόροι είναι εξοικεωμένοι με τους νόμους που σχετίζονται με την ασφάλεια του λογισμικού, την πνευματική ιδιοκτησία, τη διαχείρηση δεδομένων και τα διεθνή θεσμικά πρότυπα (π.χ. ISO) ανάπτυξης λογισμικού.
- Μηχανικοί Λογισμικού: Οι μηχανικοί λογισμικού εξασφαλίζουν ότι το προϊόν ειναι τεχνικά και οικονομικά υλοποιήσιμο.Μπορούν να εκπαιδεύσουν τους τελικούς χρήστες/πελάτες για τις καινοτομίες του προϊόντος τόσο από πλευρά υλικού (hardware) όσο και από πλευρά τεχνολογίας λογισμικού. Ακόμη είναι υπεύθυνοι για την εκτίμηση του κόστους του έργου καθώς και του χρόνου υλοποίησής του.

Κάθε ενδιαφερόμενο μέλος διαθέτει διαφορετική άποψη για τον τρόπο λειτουργίας του συστήματος και συχνά οι απόψεις των ενδιαφερόμενων μελών έρχονται σε σύγκρουση. Επιπλέον, πολλές φορές οι τελικοί χρήσες και οι προγραμματιστές μπορεί να έχουν προκαταλήψεις (σωστές ή λάθος) για το τι θεωρεί η άλλη ομάδα σημαντική προδιαγραφή για το υπό ανάπτυξη σύστημα.

Πολλές διαφορετικές προσεγγίσεις έχουν προταθεί για τη συνεργατική συλλογή απαιτήσεων [31], [35], [44]. Κάθε μία χρησιμοποιεί ένα ελαφρώς διαφορετικό σενάριο, άλλα όλες εφαρμόζουν κάποια παραλλαγή των ακόλουθων κατευθυντήριων γραμμών:

- Στις συνεδριάσεις που διεξάγονται συμμετέχουν τόσο οι μηχανικοί λογισμικού όσο και οι υπόλοιποι εμπλεκόμενοι.
- Θεσπίζονται κανόνες για την προετοιμασία και τη συμμετοχή.
- Προτείνεται μια ατζέντα που καθίσταται επαρκώς τυπική για να καλύψει όλα τα σημαντικά σημεία, αλλά αρκετά άτυπη για να ενθαρρύνει την ελεύθερη ροή των ιδεών.
- Υπάρχει ένας αμέτοχος ‘’μεσολαβητής’’ (μπορεί να είναι πελάτης, ένας υπεύθυνος ανάπτυξης ή ένα τρίτο πρόσωπο) για να ελέγχει τη συνεδρίαση.
- Χρησιμοποιείται ένας ‘’μηχανισμός ορισμού’’ (μπορεί να είναι φύλλα εργασίας, πίνακες, ή αυτοκόλλητα τοίχου ή ένας ηλεκτρονικός πίνακας ανακοινώσεων, αίθουσα συνομιλίας ή εικονικός τόπος συζητήσεων).

3.2.2 Τεχνικές Εκμαίευσης Απαιτήσεων

Υπάρχουν πολλές διαφορετικές τεχνικές για την εκμαίευση των απαιτήσεων από τα ενδιαφερόμενα μέλη. Όποια μέθοδος και εάν εφαρμοστεί από την ομάδα έργου, οι αναλυτές οφείλουν να είναι κατάλληλα εκπαιδευμένοι όσον αφορά τις τεχνικές που θα χρησιμοποιήσουν. Ακόμη η διαδικασία εκμαίευσης των απαιτήσεων είναι πιο εύκολα υλοποιήσιμη αλλά και περισσότερο αποδοτική εάν οι μηχανισμοί εκμαίευσης είναι σαφώς καθορισμένοι, τεκμηριωμένοι και τα ενδιαφερόμενα μέλη έχουν ενημερωθεί πρίν από κάθε συνεδρίαση εκμαίευσης [33]. Επίσης υπάρχουν πολλά διαφορετικά είδη πληροφοριών που θα πρέπει να εκμαιευθούν και τα διάφορα ενδιαφερόμενα μέλη θα προτιμήσουν διαφορετικές προσεγγίσεις.

Οι διάφορες τεχνικές εκμαίευσης απαιτήσεων περιλαμβάνουν δύο είδη δραστηριοτήτων, τις δραστηριότητες στις οποίες οι αναλυτές αλληλεπιδρούν με τα ενδιαφερόμενα μέλη για τις απαιτήσεις τους (facilitated activities: FA), αλλά και ανεξάρτητες δραστηριότητες στις οποίες οι αναλυτές εργάζονται μόνοι τους για να ανακαλύψουν πολύτιμες πληροφορίες για το παραγώμενο προϊόν λογισμικού.

Οι FA εστιάζονται κυρίως στην ανακάλυψη των εταιρικών απαιτήσεων καθώς και των απαιτήσεων των τελικών χρηστών. Η άμεση επικοινωνία και συνεργασία με τους τελικούς χρήστες είναι απαραίτητη επειδή οι απαιτήσεις των χρηστών είναι άμεσα συνδεδεμένες με τις εργασίες που θα πρέπει να υλοποιούν οι χρήστες μέσω τους

προϊόντος λογισμικού. Όσον αφορά τις εταιρικές απαιτήσεις, η εκμαίευση αυτών έγκειται στην συνεργασία των αναλυτών με τους πελάτες οι οποίοι πληρώνουν για την ανάπτυξή του.

Οι ανεξάρτητες δραστηριότητες συμπληρώνουν ό,τι έχει παρουσιαστεί από τους χρήστες σαν απαίτηση και στόχος τους είναι να αποκαλύψουν την απαιτούμενη λειτουργικότητα του λογισμικού την οποία οι χρήστες δεν θα μπορούσαν να προσδιορίσουν.

Τα περισσότερα έργα ανάπτυξης προϊόντων λογισμικού χρησιμοποιούν ένα συνδυασμό των προαναφερθέντων δραστηριοτήτων. Κάθε είδος δραστηριότητας προσφέρει μία τελείως διαφορετική οπτική και μπορεί να εκμαιεύσει απαιτήσεις τελείως διαφορετικές απ'ότι κάποιο άλλο είδος.

Οι ακόλουθες ενότητες περιγράφουν τεχνικές που χρησιμοποιούνται συνήθως για την εκμαίευση των απαιτήσεων.

3.2.2.1 Συνεντεύξεις (Interviews)

Ο πιο προφανής τρόπος για να μάθει κανείς τι χρειάζονται οι χρήστες ενός συστήματος λογισμικού είναι να τους ρωτήσει. Οι συνεντεύξεις είναι ένας παραδοσιακός τρόπος εκμαίευσης απαιτήσεων εισόδου για όλα τα διαφορετικά είδη προϊόντων λογισμικού. Οι περισσότεροι αναλυτές παίρνουν ατομικές συνεντεύξεις ή συνεντεύξεις με μικρών πλήθος ερωτηθέντων προκειμένου να εκμαιεύσουν απαιτήσεις αποτελεσματικότερα. Στα έργα ευέλικτης ανάπτυξης (Agile projects) η εκτεταμένη χρήση των συνεντεύξεων λειτουργεί ως ένας μηχανισμός για να λάβει η ομάδα έργου άμεση ανατροφοδότηση (feedback) των χρηστών όσον αφορά τις απαιτήσεις. Τέλος οι συντεύξεις είναι ευκολότερο να προγραμματιστούν και να αποφέρουν σημαντικά αποτελέσματα απ'ότι δραστηριότητες μεγάλου πλήθους ατόμων όπως τα εργαστήρια εκμαίευσης απαιτήσεων (requirement workshops) [34].

Τα βασικά σημεία στα οποία οφείλει να εστιάζει ο αναλύτης για την εκμαίευση απαιτήσεων μέσω των συνεντεύξεων είναι τα εξής:

- Οι συνεντεύξεις απαιτούν προετοιμασία και καλή διαχείριση της επικοινωνίας.

- Μεγάλο πλήθος συνεντεύξεων σε όσο το δυνατόν περισσότερα ενδιαφερόμενα μέλη.
- Οι συνεντεύξεις πρέπει να αποτελούνται κυρίως από ερωτήσεις οι οποίες είναι άμεσα συνδεδεμένες με το πρόβλημα που καλείται να λύσει το υπό ανάπτυξη προϊόν λογισμικού.

Τέλος, οι στόχοι της ομάδας έργου για την εκμαίευση απαιτήσεων μέσω των συνεντεύξεων είναι οι εξής:

- Η καταγραφή των πληροφοριών που πρέπει να χρησιμοποιηθούν ως είσοδος για την ανάλυση των απαιτήσεων και την μοντελοποίησή τους.
- Η εκμαίευση των απαιτούμενων πληροφοριών από τα ενδιαφερόμενα μέλη με ακρίβεια και αποτελεσματικότητα.

3.2.2.2 Ομάδες Εστίασης (Focus Groups)

Είναι συχνό φαινόμενο οι ομάδες έργου να έχουν μία μεγάλη και πολυποίκιλη βάση διαφορετικών χρηστών, οπότε η επιλογή κατάλληλων ομάδων εστίασης πρέπει να γίνει με ιδιαίτερη προσοχή γιατί είναι εξαιρετικά σημαντική για την αποτελεσματική εκμαίευση των απαιτήσεων [32].[33]. Οι συνήθεις πρακτικές επιλογής ομάδων εστίασης προτείνουν είτε τη δημιουργια ομάδων με άτομα κοινών χαρακτηριστικών (π.χ. χρήστες που έχουν χρησιμοποιήσει προηγούμενες εκδόσεις του λογισμικού ή προϊόντα παρόμοια με το υπό ανάπτυξη) είτε ομάδες εστίασης οι οποίες αποτελούνται εξίσου από όλο το φάσμα των κατηγοριών των διαφορετικών χρηστών προκειμένου κάθε κατηγορία να εκπροσωπείται εξίσου.

Η εκμαίευση απαιτήσεων μέσω των ομάδων εστίασης οδηγεί κυρίως σε μη-λειτουργικές, ποιοτικές απαιτήσεις του συστήματος. Είναι ασυνήθιστο οι αναλυτές να είναι σε θέση να λάβουν μία ποσοτική ανάλυση των απαιτήσεων από τις ομάδες εστίασης οπότε απαιτείται η περαιτέρω αξιολόγηση από μεριά τους προκειμένου να αποφανθούν για το ποιές απαιτήσεις πρέπει να μπουν σε προτεραιότητα. Τέλος, είναι σημαντικό να σημειωθεί ότι τα άτομα τα οποία παίρνουν μέρος στις ομάδες εστίασης συνήθως δεν έχουν δικαιοδοσία στη λήψη αποφάσεων για τον καθορισμό των απαιτήσεων.

3.2.2.3 Εκμαίευση απαιτήσεων μέσω των Παρατηρήσεων (Observations)

Όταν οι αναλυτές ζητούν από τα ενδιαφερόμενα μέλη και κυρίως από τους τελικούς χρήστες να περιγράψουν πώς κάνουν τη δουλεία τους ή πως χρησιμοποιούν το λογισμικό στην καθημερινότητά τους, είναι πιθανό τα ενδιαφερόμενα μέλη να μην μπορούν απαντήσουν με πλήρεις λεπτομέρειες οι οποίες πολλές φορές είναι και εσφαλμένες. Συνήθως αυτό το φαινόμενο οφείλεται στο γεγονός ότι τα καθήκοντά των χρηστών είναι πολύπλοκα και επομένως είναι δύσκολο να τα θυμούνται με κάθε λεπτομέρεια. Σε άλλες περιπτώσεις, οι χρήστες είναι τόσο εξοικειωμένοι με την εκτέλεση μιας εργασίας που δεν μπορούν καν να περιγράψουν ό,τι κάνουν. Συνεπώς πολλές φορές είναι καλό ο αναλυτής να παρακολουθεί πως οι χρήστες εκτελούν τα καθήκοντά τους προκειμένου να αποκτήσει μια περισσότερο εμπεριστατωμένη άποψη για τα καθήκοντά τους.

Οι παρατηρήσεις αποτελούν μια χρονοβόρα διαδικασία και συνεπώς δεν είναι κατάλληλες για κάθε χρήστη ή εργασία[32]. [34]. Προκειμένου ο αναλυτής να μην διαταράσσει σε τακτά χρονικά διαστήματα τους χρήστες, κάθε συνεδρίαση παρατήρησης (observation session) συνήθως δεν ξεπερνάει τις δύο ώρες. Ακόμη οι αναλυτές συνήθως επιλέγουν να παρατηρούν εργασίες υψηλής σημασίας ή υψηλού κινδύνου. Όσον αφορά τα έργα ευέλικτης ανάπτυξης (agile projects) οι αναλυτές παρατηρούν το χρήστη μόνο στις εργασίες οι οποίες είναι σημαντικές για την επόμενη προσαύξηση.

Η παρατήρηση της ροής εργασίας του χρήστη (user workflow) στο εργασιακό του περιβάλλον δίνει τη δυνατότητα στον αναλυτή να επικυρώσει πληροφορίες που σύλλεξε από άλλες πηγές, να εντοπίσει νέα θέματα συνεντεύξεων, να εντοπίσει προβλήματα στο παρόν σύστημα και να προσδιορίσει τρόπους με τους οποίους το καινούργιο σύστημα θα μπορούσε να βελτιώσει τη ροή εργασίας του χρήστη [34].

Οι παρατηρήσεις του αναλυτή μπορεί να είναι είτε σιωπηλές είτε διαδραστικές. Οι σιωπηλές παρατηρήσεις συνίστανται όταν οι χρήστες είναι πολυάσχολοι. Από την άλλη, οι FA επιτρέπουν στον αναλυτή να διακόψει το χρήστη κατά τη διάρκεια της εργασίας του και να του κάνει κάποια ερώτηση ή παρατήρηση.

3.2.2.4 Ερωτηματολόγια (Questionnaires)

Τα ερωτηματολόγια είναι ένας τρόπος ο οποίος παρέχει στον αναλυτή την ευκολία του να ερευνά μεγάλες ομάδες χρηστών προκειμένου να κατανοήσει τις ανάγκες τους. Τα ερωτηματολόγια αποτελούν μία ανέξοδη τεχνική εκμαίευσης απαιτήσεων και μπορούν να χορηγηθούν εύκολα σε πολλές γεωγραφικές περιοχές. Ακόμη τα ερωτηματολόγια μπορούν να χρησιμοποιηθούν ως είσοδος σε άλλες τεχνικές εκμαίευσης. Για παράδειγμα, μπορεί ένα ερωτηματολόγιο να χρησιμοποιηθεί για τον εντοπισμό των μεγαλύτερων αναγκών/προβλημάτων που αντιμετοπίζουν οι χρήστες και στη συνέχεια τα αποτελέσματά του να χρησιμοποιηθούν προκειμένου να αξιολογηθεί η προτεραιότητά τους από τους ιθύνοντες στις θέσεις λήψης αποφάσεων. Τέλος, η σύνθεση καλών γραπτών ερωτήσεων αποτελεί την μεγαλύτερη πρόκληση που συναντούν οι αναλυτές στη δημιουργία των ερωτηματολογίων.

3.2.2.5 Ανάλυση Τεκμηριώσεων (Documentation Analysis)

Η ανάλυση των τεκμηριώσεων συνεπάγεται την εξέταση κάθε υπάρχουσας τεκμηρίωσης για τις πιθανές απαιτήσεις λογισμικού. Η πιο χρήσημη τεκμηρίωση περιλαμβάνει τις προδιαγραφές των απαιτήσεων του σύστηματος (requirement specifications), των επιχειρηματικών διαδικασιών, εγχειρίδια χρήσης των υφιστάμενων καθώς και κάποιες εφαρμογές του λογισμικού εάν προϋπάρχει.

Οι τεκμηριώσεις μπορούν να περιγράψουν εταιρικά ή βιομηχανικά πρότυπα που πρέπει να ακολουθηθούν ή κανονισμούς με τους οποίους το προϊόν πρέπει να συμμορφώνεται. Κατά την αντικατάσταση ενός υπάρχοντος συστήματος, η παλιά τεκμηρίωση μπορεί να αποκαλύψει κάποια λειτουργικότητα που πρέπει να διατηρηθεί ή και κάποια λειτουργικότητα η οποία πρέπει να απαλοιφθεί από την επόμενη έκδοση του λογισμικού.

Η ανάλυση των τεκμηριώσεων είναι ένας τρόπος εκμαίευσης απαιτήσεων ο οποίος επιταγχύνει τις διαδικασίες με σκοπό να συγκεντρωθούν εγκαίρως όλες οι πιθανές απαιτήσεις από τα διάφορα ενδιαφερόμενα μέλη προκειμένου να αναλυθούν και τελικά να καθοριστούν.

Η ανάλυση των τεκμηριώσεων μπορεί να αποκαλύψει πληροφορίες τις οποίες τα εμπλεκόμενα μέλη να μην έχουν αναφέρει στους αναλυτές, είτε επείδη δεν το

σκέφτηκαν είτε επειδή δεν τις γνωρίζουν. Συνεπώς η ανάλυση των τεκμηριώσεων αποτελεί μία σημαντική τεχνική εκμαίευσης απαιτήσεων οι οποίες είναι πιθανό να εμφανιστούν σε κάποιο στάδιο του έργου όπου η ενσωματωσή τους να καθίσταται αδύνατη.

Ένας κίνδυνος σχετικός με τη συγκεκριμένη τεχνική είναι ότι οι υπάρχουσες τεκμηριώσεις ενδέχεται να μην είναι πλήρως ενημερωμένες, με αποτέλεσμα απαιτήσεις οι οποίες έχουν αλλάξει να μην έχουν τεκμηριωθεί πράγμα που δυσχεραίνει την αποτελεσματικότητα της ομάδας έργου.

3.2.3 Ανάλυση Απαιτήσεων

Μολονότι η επιτυχία ενός υπολογιστικού συστήματος ή ενός προϊόντος υπολογίζεται με πολλούς τρόπους, η ικανοποίηση των χρηστών βρίσκεται στην κορυφή της λίστας [35]. Εάν η ομάδα έργου κατανοήσει τον τρόπο με τον οποίο οι χρήστες θέλουν να αλληλεπιδρούν με το σύστημα, τότε βρίσκεται σε πολύ καλυτερη θέση να χαρακτηρίσει με σαφήνεια τις απαιτήσεις και να αναπτύξει εποικοδομητικά τα μοντέλα ανάπτυξης και σχεδίασης. Ως εκ τούτου, οι απαιτήσεις μοντελοποίησης με τη UML (UML: Unified Modelling Language) αρχίζει με τη δημιουργία σεναρίων με την μορφή περιπτώσεων χρήσης.

3.2.3.1 Περιπτώσεις Χρήσης, Ιστορίες Χρήσης και Σενάρια Χρήσης

Οι αναλυτές χρησιμοποιούν εδώ και αρκετά χρόνια τις περιπτώσεις χρήσης προκειμένου να εκμαιεύουν τις απαιτήσεις από τους χρήστες. Η συγκεκριμένη προσέγγιση (usage-centered scenarios) μοντελοποιήθηκε υπό την μορφή των περιπτώσεων χρήσης [36].Πρόσφατα, λόγω της εκτεταμένης χρήσης των ευέλικτων μοντέλων ανάπτυξης λογισμικού εμφανίστηκε και η αναλύση των απαιτήσεων μέσω των ιστοριών χρήσης, οι οποίες αποτελούν συνοπτικές δηλώσεις των αναγκών των χρηστών και λειτουργούν ως αφετηρία για την αποσαφήνιση των απαιτήσεων των χρηστών .

Οι περιπτώσεις χρήσης καθώς και οι ιστορίες χρήσης διαφοροποιούνται από τις παλαιές τεχνικές εκμαίευσης απαιτήσεων, οι οποίες επικεντρώνονταν στο προϊόν

λογισμικού, σε αντίθεση με τις νέες τεχνικές οι οποίες επικεντρώνονται στο τι χρειάζονται οι τελικοί χρήστες από το υπό ανάπτυξη σύστημα [36], [37]. Ο σκοπός των περιπτώσεων και ιστοριών χρήσης έγκειται στο να αποσαφηνιστεί ποιές εργασίες επιθυμεί να κάνει αποδοτικά μέσω του συστήματος ο χρήστης ή ποιές αλληλεπιδράσεις επιθυμεί με το σύστημα. Οι συγκεκριμένες τεχνικές βοηθούν τον αναλυτή να κατανοήσει την επιθυμητή λειτουργικότητα του συστήματος προκειμένου να σχεδιάσει αποτελεσματικά τις περιπτώσεις χρήσης.

Μία *περίπτωση χρήσης* περιγράφει τη συμπεριφορά του συστήματος υπό διάφορες συνθήκες καθώς το σύστημα ανταποκρίνεται σε ένα αίτημα από έναν από τους εμπλεκομενούς του [37]. Στην ουσία, μία περίπτωση χρήσης εκφράζει μια ιστορία για το πως ο τελικός χρήστης (υιοθετώντας έναν από τους πιθανούς ρόλους) αλληλεπιδρά με το σύστημα υπό ένα συγκεκριμένο σύνολο περιστάσεων. Η ιστορία μπορεί να γραφεί με αφηγηματικό κείμενο, μια περιγραφή των έργων ή αλληλεπιδράσεων, μια περιγραφή που βασίζεται σε πρότυπο ή μια διαγραμματική αναπαράσταση.

Ανεξάρτητα από τη μορφή του, ένα σενάριο χρήσης απεικονίζει το λογισμικό ή το σύστημα από την απτική γωνία του τελικού χρήστη.

Το πρώτο βήμα για τη σύνταξη μιας περίπτωσης χρήσης είναι να οριστεί ένα σύνολο από ‘’δράστες’’ *(*actors*)*, οι οποίοι θα εμπλέκονται στην ιστορία. Οι δράστες είναι οι διαφορετικοί άνθρωποι ή συσκευές που χρησιμοποιούν το σύστημα ή το προϊόν στο πλαίσιο της λειτουργίας και της συμπεριφοράς που πρόκειται να περιγραφεί. Οι δράστες αντιπροσωπεύουν τους ρόλους που οι άνθρωποι ή οι συσκευές υιοθετούν καθώς λειτουργεί το σύστημα. Συνεπώς ο δράστης είναι κάτι που επικοινωνεί με το σύστημα ή το προϊόν και ειναι εξωτερικά του συστήματος. Κάθε δράστης έχει έναν η περισσότερους στόχους όταν χρησιμοποιεί το σύστημα.

Είναι σημαντικό να σημειωθεί ότι ένας δράστης και ένας τελικός χρήστης δεν ταυτίζονται απαραίτητα. Ένας τυπικός χρήστης πιθανόν να υιοθετεί διάφορους ρόλους κατά τη χρήση ενός συστήματος, ενώ ένας δράστης αντιπροσωπεύει μια κατηγορία εξωτερικών οντοτήτων που υιοθετούν ένα ρόλο στο πλαίσιο μιας περίπτωσης χρήσης.

3.2.3.2 Personas

Ο Alan Cooper [38] ανέφερε για πρώτη φορά την έννοια των personas με στόχο την πληρέστερη κατανόηση των απαιτήσεων των χρηστών από μία ομάδα έργου όσον αφορά τη σχεδίαση του λογισμικού και ιδιαιτέρως την αλληλεπίδραση του συστήματος με τον τελικό χρήστη.

Η persona αποτελεί ένα παράδειγμα χρήστη ο οποίος αλληλεπιδρά με το σύστημα [37],[38]. Η βασική αντίληψη πίσω από την εμφάνιση των personas είναι πως προκειμένου ένα προϊόν λογισμικού να είναι αποτελεσματικό, θα πρέπει να σχεδιαστεί για ένα συγκεκριμένο χρήστη ή ομάδα χρηστών. Οι personas αποτελούν ‘’φανταστικούς’’ ανθρώπους/χρήστες του λογισμικού οι οποίοι στηρίζονται στο προφίλ των δυνητικών χρηστών του υπο ανάπτυξη λογισμικού.

Σκοπός των personas είναι να δημιουργηθούν αξιόπιστες και ρεαλιστικές αναπαραστάσεις των βασικών ομάδων χρηστών [39]. Οι αναπαραστάσεις αυτές θα πρέπει να βασίζονται τόσο σε ποιοτικά όσο και ποσοτικά χαρακτηριστικά των δυνητικών χρηστών, οι οποίες αποτελούν αποτελέσμα ενδελεχούς έρευνας [40].

Συνεπώς, οι αποτελεσματικές personas:

- Αντιπροσωπεύουν ένα μεγάλο μέρος των τελικών χρηστών του λογισμικού.
- Εκφράζουν και επικεντρώνονται στις κυριότερες αναγκές των τελικών χρηστών.
- Παρέχουν μια ξεκάθαρη εικόνα των προσδοκιών των χρηστών από το σύστημα και πώς θα επιθυμούσαν ιδανικά να αλληλεπιδρούν με αυτό.
- Βοηθούν στο να εντοπιστούν ζητήματα λειτουργικότητας.
- Περιγράφουν πραγματικούς ανθρώπους με ατομικό ιστορικό, στόχους και αξίες.

Κάποιες personas είναι ιδιαιτέρως αναλυτικές, ενώ κάποιες άλλες παρέχουν τις βασικές πληροφορίες για ένα τύπο χρήστη[39], [40]. Μία τυπική σύνταξη personas έχει την παρακάτω μορφή:

Persona:	USDA Senior Manager Gatekeeper
Photo:	
Fictional name:	Matthew Johnson
Job title/ major responsibilities:	Program Staff Director, USDA
Demographics:	• 51 years old • Married • Father of three children • Grandfather of one child • Has a Ph.D. in Agricultural Economics.
Goals and tasks:	He is focused, goal-oriented within a strong leadership role. One of his concerns is maintaining quality across all output of programs. Spends his work time: • Requesting and reviewing research reports, • preparing memos and briefs for agency heads, and • supervising staff efforts in food safety and inspection.
Environment:	He is comfortable using a computer and refers to himself as an intermediate Internet user. He is connected via a T1 connection at work and dial-up at home. He uses email extensively and uses the web about 1.5 hours during his work day.
Quote:	"Can you get me that staff analysis by Tuesday?"

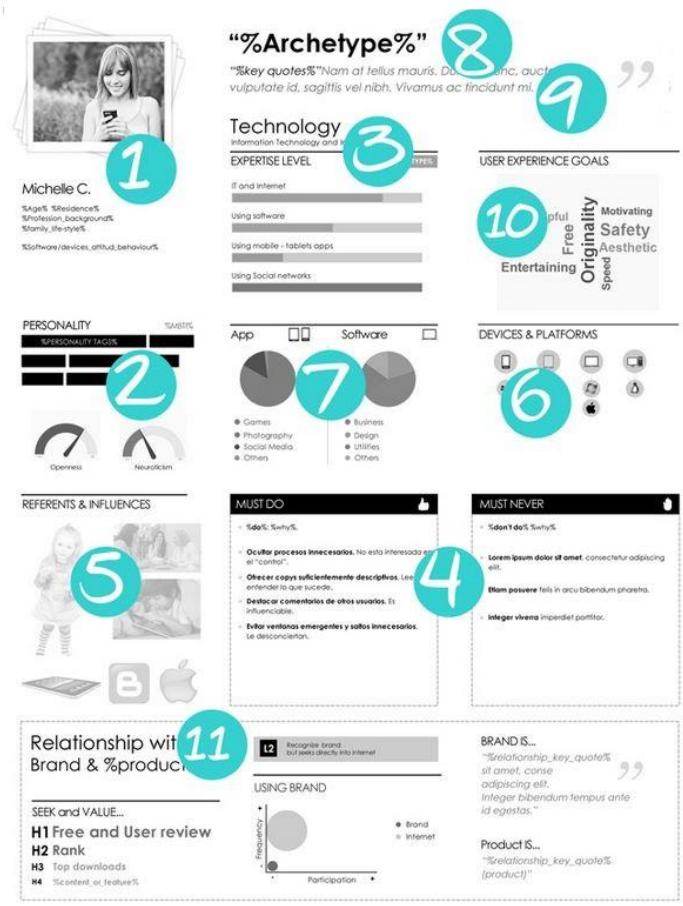
Εικόνα 3-4 Μία τυπική σύνταξη persona[24]

Όσον αφορά τις νεοφυείς επιχειρήσεις οι οποίες ασχολούνται με την ανάπτυξη καινοτόμων προϊόντων λογισμικού, αξίζει να σημειωθεί ότι η χρήση των personas είναι ευρέως διαδεδομένη μιας και οι συγκεκριμένες επιχειρήσεις, τουλάχιστον στα αρχικά τους στάδια, δεν έχουν ακόμα πελάτες όποτε η αποτελεσματική δημιουργία και χρήση αντιπροσωπευτικών personas βοηθάει πάρα πολύ στην κατανόηση των διαφορετικών ειδών δυνητικών χρηστών που καλούνται να ικανοποιήσουν μέσω του καινοτόμου προϊόντος τους.

[24] http://www.usability.gov/how-to-and-tools/methods/personas.html

3.2.3.3 Πρότυπο Persona

Στο διαδικτυο υπάρχουν πολλές μεθοδολογίες για την ανάπτυξη personas. Οι περισσότερες έχουν κάποια κοινά βασικά στοιχεία τα οποία είναι απαραίτητα για την ανάπτυξη κατάλληλων personas. Η UX Lady[25], η οποία είναι έμπειρη UX Designer, προσπάθησε να μοντελοποιήσει τη σύνταξη των personas**.** Το μεθοδολογικό πλαίσιο το οποίο πρόκειται να αναπτύξουμε στο Κεφάλαιο 5 βασίζεται στη συγκεκριμένη μέθοδο για την μοντελοποίηση της σύνταξης των personas.



Εικόνα 3-5 Το βασικό σχέδιο μιας persona[26]

[25] http://www.ux-lady.com/diy-user-personas

[26] http://www.ux-lady.com/introduction-to-user-personas/

Οι βασικές πληροφορίες τις οποίες καλούμαστε να συμπληρώσουμε είναι οι εξής:

1. **Βασικά στοιχεία (Profile area):** Στο συγκεκριμένο πεδίο συμπληρώνουμε τα βασικά δημογραφικά χαρακτηριστικά της persona, όπως ηλικία, τόπος διαμονής, επάγγελμα, κ.λ.π.
2. **Στοιχεία προσωπικότητας (Personality elements):** Στο συγκεκριμένο πεδό συμπληρώνουμε τα βασικά χαρακτηριστικά της προσωπικότητας της persona που πρόκειται να ανάπτυξουμε. Επειδή το συγκεκριμένο πεδίο είναι αρκετά δύσκολο να συμπληρωθεί αντιπροσωπευτικά συνίσταται η χρήση της μεθόδου MBTI [41]και του 5Factor μοντέλου [42], προκειμένου να βοηθήσουν το σχεδιαστή στην πλήρη αποσαφήνιση των χαρακτηριστικών της προσωπικότητας της persona.
3. **Πραγματογνωμοσύνη (Expertise):** Στο συγκεκριμένο πεδίο συμπληρώνουμε το επίπεδο εξοικείωσης της persona με τον τομέα εφαρμογής του προϊόντος.
4. **Πάντα/Ποτέ (Must Does/Must Never):** Στο συγκεκριμένο πεδίο συμπληρώνουμε τί επιθυμεί η persona από το προϊόν και τί την ενοχλεί στο προϊόν.
5. **Επιρροές (Referents & Influences):** Στο συγκεκριμένο πεδίο συμπληρώνουμε τους ανθρώπους, τα brands και τα προϊόντα τα οποία επηρεάζουν τη σχέση της persona με το διαδίκτυο, τους υπολογιστές, τις εφαρμογές κ.λ.π.
6. **Συσκευές και Πλατφόρμες (Devices & Platforms):** Στο συγκεκριμένο πεδίο συμπληρώνουμε τις συσκευές με τις οποίες η persona αλληλεπιδρά με το προϊόν.
7. **Χρησιμοποιούμενο προϊόν ή υπηρεσία (Used product/service):** Το συγκεκριμένο πεδίο αφορά το τομέα (domain) του προϊόντος.
8. **Αρχέτυπο (Archetype):** Στο συγκεκριμένο πεδίο συμπληρώνουμε μία περίληψη των χαρακτηριστικών της persona.
9. **Χαρακτηριστικά σχόλια (Key Quotes):** Στο συγκεκριμένο πεδίο συμπληρώνουμε σχόλια της persona που σχετίζονται με την persona ως χρήστη.

10. **Experience Goals:** Στο συγκεκριμένο πεδίο συμπληρώνουμε ποιες είναι οι επιδιώξεις και οι προτιμήσεις της persona κατά την αλληλεπίδρασή της με το σύστημα/προϊόν.
11. **Σχέση με το προϊόν (Brand Relationship):** Στο συγκεκριμένο πεδίο περιγράφουμε τη σχέση μεταξύ του brand και του προϊόντος.
12. **Φωτογραφία (Picture):** Τοποθετούμε μία φωτογραφία της persona για καλύτερη οπτικοποίησή της.
13. **Είδος χρήστη (User type):** Στο συγκεκριμένο πεδίο συμπληρώνουμε τι είδος χρήστη είναι η persona όπου δημιουργούμε.

3.2.3.4 Διαγράμματα Περιπτώσεων Χρήσης

Οι περιπτώσεις χρήσης και η χρήση διαγραμμάτων περιπτώσεων χρήσης UML βοηθούν στον καθορισμό της λειτουργικότητας και των χαρακτηριστικών του λογισμικού από την πλευρά του χρήστη. Τα διαγράμματα περιπτώσεων χρήσης παρέχουν μία υψηλού επιπέδου οπτική αναπαράσταση των απαιτήσεων των χρηστών [43].Προκειμένου να γίνει κατανοητό το πώς λειτουργούν οι περιπτώσεις χρήσης και τα διαγράμματα περιπτώσεων χρήσης, παραθέτεται ένα παράδειγμα δημιουργίας εφαρμογής λογισμικού για τη διαχείρηση των ψηφιακών μουσικών αρχείων. Μερικές από τις εργασίες που μπορεί να εκτελέσει το εν λόγω λογισμικό περιλαμβάνουν:

- Κατεβάστε ένα αρχείο μουσικής MP3 και αποθηκεύστε το στη βιβλιοθήκη της εφαρμογής.
- Καταγράψτε μια συνεχή ροή μουσικής και αποθηκεύστε την στη βιβλιοθήκη της εφαρμογής.
- Διαχειριστείτε τη βιβλιοθήκη της εφαρμογής.
- Γράψτε μια λίστα με τραγούδια της βιβλιοθήκης σε ένα CD.
- Φορτώστε μια λίστα με τα τραγούδια της βιβλιοθήκης σε μία συσκευή αναπαραγωγής MP3.

- Μετατρέψτε ένα τραγούδι από την μορφή MP3 σε AAC και το αντίστροφο.

Η παραπάνω λίστα δεν είναι εκτενής, αλλά αρκεί για να κατανοήσουμε το ρόλο των περιπτώσεων χρήσης και των διαγραμμάτων περιπτώσεων χρήσης.

Ένα διάγραμμα περίπτωσης χρήσης UML παρέχει μια επισκόπηση όλων των περιπτώσεων χρήσης και του τρόπου με τον οποίο σχετίζονται. Τα συγκεκριμένα διαγράμματα παρέχουν μία εκτενή διορατικότητα της λειτουργικότητας του συστήματος. Ένα διάγραμμα περίπτωσης χρήσης για μία ψηφιακή εφαρμογή μουσικής παρουσιάζεται στην εικόνα 3-6.

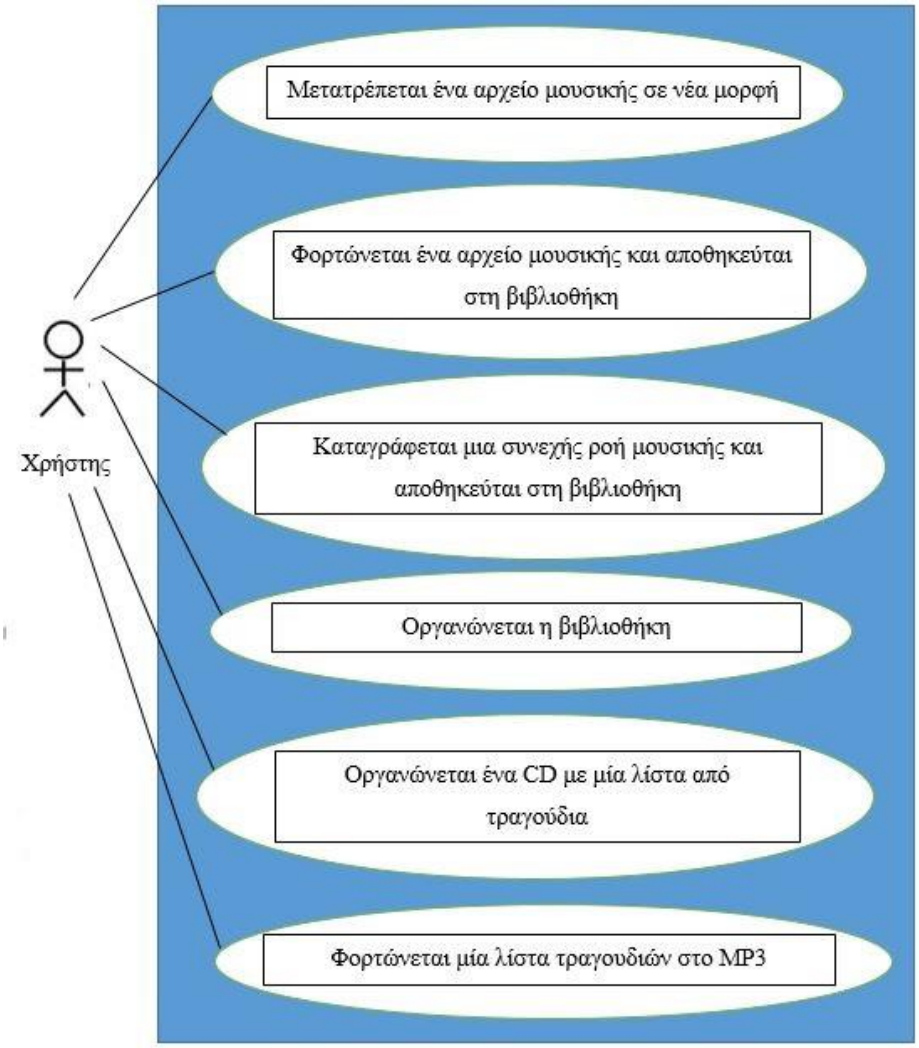
Σε αυτό το διάγραμμα, το ανθρωπάκι απεικονίζει ένα δράστη που σχετίζεται με μία κατηγορία χρηστών ή ένα άλλο στοιχείο αλληλεπίδρασης. Τα πολύπλοκα συστήματα συνήθως έχουν περισσότερους από έναν δράστες. Για παράδειγμα, μια εφαρμογή ενός μηχανήματος αυτόματης πώλησης μπορεί να έχει τρεις δράστες αντιπροσωπεύοντας τους πελάτες, το προσωπικό επισκευής και τους πωλητές οι οποίοι ανατροφοδοτούν το μηχάνημα.

Στο διάγραμμα περίπτωσης χρήσης, εμφανίζονται οι περιπτώσεις χρήσης σε ωοειδή σχήματα. Οι δράστες συνδέονται με γραμμές στις περιπτώσεις χρήσης που διεξάγουν. Σημειώνεται ότι κανένα από τα στοιχεία για τις περιπτώσεις χρήσης που περιλαμβάνονται στο διάγραμμα στο διάγραμμα δεν αποθηκεύονται σε αυτό και αντ' αυτού πρέπει να αποθηκεύονται ξεχωριστά. Σημειώνεται επίσης ότι οι περιπτώσεις χρήσης τοποθετούνται σε ένα ορθογώνιο, αλλά όχι οι δράστες. Αυτό το ορθογώνιο αποτελεί μια οπτική υπενθύμιση των ορίων του συστήματος και ότι οι δράστες βρίσκονται εκτός του συστήματος.

Ορισμένες περιπτώσεις χρήσης σε ένα σύστημα ενδέχεται να σχετίζονται μεταξύ τους. Για παράδειγμα, υπάρχουν παρόμοια βήματα στο γράψιμο μιας λίστας τραγουδιών σε ένα CD και στη φόρτωση μιας λίστας τραγουδιών σε μια συσκευή αναπαραγωγής MP3. Και στις δύο περιπτώσεις, ο χρήστης δημιουργεί πρώτα μία κενή λίστα και στη συνέχεια προσθέτει τραγούδια από τη βιβλιοθήκη στη λίστα. Για να αποφευχθούν οι επικαλύψεις σε περιπτώσεις χρήσης, συνήθως είναι καλύτερα να δημιουργείται μια νέα περίπτωση χρήσης που αντιπροσωπεύει τη δραστηριότητα επικάλυψης και στη συνέχεια επιτρέπει στις άλλες περιπτώσεις χρήσης να

περιλαμβάνουν αυτή τη νέα περίπτωση για να χρησιμοποιηθεί ως ένα από τα βήματά τους.

Ένα διάγραμμα περίπτωσης χρήσης επειδή εμφανίζει όλες τις περιπτώσεις χρήση, είναι χρήσιμο βοήθημα για την εξασφάλιση ότι η ομάδα έργου έχει καλύψει όλες τις λειτουργίες του συστήματος. Για τον διοργανωτή της ψηφιακής μουσικής σίγουρα θα χρειαζόμασταν περισσότερες περιπτώσεις χρήσης, όπως είναι η περίπτωση χρήσης για την αναπαραγωγή ενός τραγουδιού από την βιβλιοθήκη. Τέλος η πιο πολύτιμη συμβολή των περιπτώσεων χρήσης για τη διαδικασία ανάπτυξης του λογισμικού είναι η περιγραφή κειμένου της κάθε περίπτωσης χρήσης και όχι το συνολικό διάγραμμα περίπτωσης χρήσης [44]. Μέσα από τις περιγραφές η ομάδα έργου είναι σε θέση να σχηματίσει μια σαφή κατανόηση των σκοπών του συστήματος που καλείται να αναπτύξει.



Εικόνα 3-6 Διάγραμμα ενος Use Case

3.2.3.5 Εντοπισμός περιπτώσεων χρήσης

Υπάρχουν πολλές τεχνικές σωστής επιλογής και ακριβή εντοπισμού των περιπτώσεων χρήσης. Οι κυριότερες είναι οι εξής [45]:

- Προσδιορισμός των δραστών και εν συνεχεία καθορισμός των διαδικασίες που θα υποστηρίζει το σύστημα καθώς και των περιπτώσεων χρήσης όπου δράστες και διαδικασίες αλληληλεπιδρούν.
- Δημιουργία ενός σεναρίου χρήσης για μία συγκεκριμένη διαδικασία και εν συνεχεία προσδιορισμός των περιπτώσεων χρήσης και των δραστών που συμμετέχουν σε κάθε μία.
- Προσδιορισμός των εξωτερικών παραγόντων στους οποίους το σύστημα πρεπει να ανταποκριθεί και εν συνεχεία συσχετισμός αυτών με τους κατάλληλους δράστες με σκοπό τη δημιουργία διαφορετικών περιπτώσεων χρήσης.

Οι προαναφερθείσες τεχνικές ελλοχεύουν πολλούς κινδύνους οι οποίοι μπορούν να οδηγήσουν σε εσφαλμένο προσδιορισμό περιπτώσεων χρήσης με αποτελέσμα την μη αποδοτική ανάπτυξη του συστήματος. Οι βασικοί κίνδυνοι τους οποίους μία ομάδα έργου οφείλει να προσέχει είναι οι εξής [44], [45]:

- Πάρα πολλές περιπτώσεις χρήσης: Πολλές φορές οι αναλυτές δημιουργούν ξεχωριστές περιπτώσεις χρήσης για κάθε ένα πιθανό σενάριο με αποτελέσμα να κάνουν την ανάλυση των απαιτήσεων δύσκολη και περίπλοκη διαδικασία.
- Πολύπλοκες περιπτώσεις χρήσης: Είναι συχνό φαινόμενο οι περιπτώσεις χρήσης να είναι πολυσέλιδες με πολύπλοκα διαγράμματα χρήσης, τα οποία εκτός από τη φυσιολογική ροή εργασίας παραθέτουν και πολλές εναλλακτικές ροές, γεγονός που τις καθιστά ακατανόητες.
- Συνδυασμός γραφικών και περιπτώσεων χρήσης: Ο ρόλος των περιπτώσεων χρήσης έγκειται στην αποσαφήνιση των αναγκών των χρηστών και όχι στο τρόπο παρουσίασης της διεπαφής συστήματος και χρήστη. Η τάση να συμπλεριλαμβάνεται σε μία περίπτωση χρήσης και η διεπαφή συστήματος-

χρήστη ελλοχεύει τον κίνδυνο άστοχου προσδιορισμού των απαιτήσεων του χρήστη.

- Περιπτώσεις χρήσης που οι χρήστες αδυνατούν να κατανοήσουν: Εάν οι χρήστες δεν μπορούν να καταλάβουν μία περίπτωση χρήσης που σχετίζεται με τις ανάγκες τους, είναι καταφανές ότι υπάρχει κάποιο πρόβλημα στον καθορισμό των απαιτήσεων από την ομάδα έργου. Οι περιπτώσεις χρήσης ενδείκνυται να γράφονται από την πλευρά του χρήστη και όχι από την πλευρά του συστήματος.

3.2.4 Προσδιορισμός Απαιτήσεων (Requirements Specification)

3.2.4.1 Προδιαγραφές Απαιτήσεων

Μετά την εξαγωγή και την ανάλυση των απαιτήσεων ακολουθεί η δραστηριότητα της σύνταξης των προδιαγραφών των απαιτήσεων.

Αποτέλεσμα της σύνταξης των προδιαγραφών είναι η έκδοση ενός κειμένου που περιγράφει τις απαιτήσεις και από αυτό το κείμενο είναι δυνατόν να κατανοήσει κάποιος που δε συμμετείχε στη σύνταξη των απαιτήσεων το τι θα κάνει το λογισμικό και με ποιους περιορισμούς [3], [44]. Για να μπορέσουμε να περιγράψουμε τις απαιτήσεις με τη μορφή προδιαγραφών, θα πρέπει καταρχάς να τις έχουμε επαρκώς κατανοήσει και προσδιορίσει, στη συνέχεια να μπορέσουμε να οργανώσουμε τη δομή του κειμένου που θα περιγράφει τις προδιαγραφές και τέλος να διατυπώσουμε-συντάξουμε τις προδιαγραφές αυτές.

Οι δραστηριότητες της εξαγωγής και της ανάλυσης των απαιτήσεων θεωρούμε ότι μας βοήθησαν να προσδιορίσουμε και να κατανοήσουμε τις απαιτήσεις που έχουν οι ενδιαφερόμενοι (stakeholders) από το σύστημα. Κατά τη δραστηριότητα της σύνταξης των προδιαγραφών των απαιτήσεων θα επιχειρήσουμε τη διατύπωση των απαιτήσεων.

Το στάδιο προσδιορισμού των απαιτήσεων αφορά στις ενεργείες των προγραμματιστών της ομάδας έργου να μεταφέρουν τις απαιτήσεις των πελατών οι οποίες αναφέρονται σε αντικείμενα, καταστάσεις, γεγονότα και δραστηριότητες σε μορφή απαιτήσεων που αφορούν τη διασύνδεση του συστήματος. Προκειμένου να επιτευχθεί αυτό, οι προγραμματιστές ξαναγράφουν τις απαιτήσεις σε κατάλληλη

μορφή προκειμένου αυτές να ανιχνεύονται και να σχετίζονται άμεσα με το υπό ανάπτυξη σύστημα.

Οι προγραμματιστές χρησιμοποιούν τις παρακάτω τεχνικες προκειμένου να προσδιορίσουν βέλτιστα τις απαιτήσεις του σύστηματος [44], [45]. Αυτές είναι:

- Λεπτομερής περιγραφή όλων των εισόδων (input) και αποτελεσμάτων (output) στην οποία συμπεριλαβάνονται οι πηγές των εισόδων (source of inputs) , ο σκοπός των αποτελεσμάτων, τα διαφορετικά είδη εισαγωγής και εξαγωγής δεδομένων, τα πρωτόκολλα που διέπουν τη σειρά με την οποία πρέπει να ανταλλάσονται δεδομένα από την είσοδο και την έξοδο του συστήματος καθώς και όλοι οι χρονικοί περιορισμοί.
- Είναι σημαντικό να σημειωθεί ότι πολλές άπειρες ομάδες λογισμικού περιορίζονται στο να περιγράφουν στο συγκεκριμένο στάδιο μόνο τον τρόπο λειτοργίας της διεπαφής χρήστη, ενώ πολύ συχνά το σύστημα αλληλεπιδρά όχι μόνο με τον τελικό χρήστη αλλά και με άλλα συστατικά λογισμικού όπως άλλα συστήματα, βάσεις δεδομένων, το Διαδίκτυο κ.ο.κ.
- Επαναδιατύπωση της λειτουργικότητας όσον αφορά τις εισροές και τις εκροές των διασυνδέσεων. Σε αυτό το σημείο η ομάδα έργου μπορεί να χρησιμοποιήσει ένα λειτουργικό διάγραμμα σημειογραφείας (*functional notation diagram)* ή ένα διάγραμμα ροής για την χαρτογράφηση των δεδομένων είσοδου και εξόδου έιτε ακόμη και την τεκμηρίωση και διαχωρισμών των απαιτήσεων σε ‘’προϋποθέσεις και μετα-συνθήκες’’ *(pre-conditions & post-conditions)*. Ακόμη πολλές ομάδες χρησιμοποιούν διαγράμματα οντότητας (entity-relationship diagrams) για τη συλλογή των συναφών δραστηριοτήτων και λειτουργιών με σκοπό την ομαδοποίησή τους σε κατηγορίες. Τέλος, κάθε προδιαγραφή θα πρέπει να είναι πλήρης, πράγμα που σημαίνει ότι θα πρέπει να να προσδιορίζει μια έξοδο για κάθε πιθανή δραστηριότητα των εισροών. Συνεπώς με αυτο τον τρόπο, η ομάδα έργου ελέγχει την εγκυρότητα των εισροών και εκροών στις διαφόρες απόκρισεις του συστήματος όταν παραβιάζονται οι προϋποθέσεις.

Τέλος, η ομάδα έργου καταστρώνει κατάλληλα κριτήρια για κάθε μία από τις ποιοτικές, μη-λειτουργικές απαιτήσεις των πελατών, έτσι ώστε να είναι δυνατόν να ελεγχθεί εάν το σύστημα πληρεί τις συγκεκριμένες μη-λειτουργικές απαιτήσεις.

3.2.4.2 Σύνταξη των απαιτήσεων

Η σύνταξη των απαιτήσεων γίνεται σε μια γλώσσα που ανήκει σε μια από τις παρακάτω κατηγορίες [3], [44]:

- **Οι άτυπες** (informal). Σε αυτή την κατηγορία ανήκουν οι φυσικές γλώσσες ή οι δομημένες φυσικές γλώσσες (φυσικές γλώσσες χωρίς επίθετα και επιρρήματα).
- **Οι ημιτυπικές** (semiformal). Μείγμα φυσικών γλωσσών με διαγράμματα και μαθηματικούς συμβολισμούς.
- **Οι τυπικές** (formal). Είναι γλώσσες που έχουν αυστηρή σύνταξη και σημασιολογία και κάνουν εκτεταμένη χρήση των μαθηματικών. Ίσως η πλέον γνωστή τυπική γλώσσα για τη σύνταξη προδιαγραφών είναι η γλώσσα Z, γνωστή και ως ‘’Z notation’’. Η γλώσσα OCL που χρησιμοποιείται σε μοντέλα UML είναι επίσης μία τυπική γλώσσα.

Όσον αφορά το ύφος της διατύπωσης, ένας συνήθης τρόπος καταγραφής των απαιτήσεων είναι οι δηλωτικές απαιτήσεις *(declarative requirements*). Οι δηλωτικές απαιτήσεις δεν είναι τίποτα άλλο από δηλώσεις της μορφής «το σύστημα (ή το λογισμικό) θα …». Ένα πρώτο επίπεδο δηλωτικών απαιτήσεων είναι τα λειτουργικά χαρακτηριστικά. Έχοντας τα βασικά λειτουργικά χαρακτηριστικά ως οδηγό, γίνεται λεπτομερέστερη περιγραφή για το τι θα κάνει το λογισμικό. Οι δηλωτικές απαιτήσεις είναι η υπόσχεση που δίνουμε στον πελάτη για τη λειτουργικότητα του λογισμικού, χωρίς να μας απασχολεί το πώς θα παρασχεθεί η λειτουργικότητα αυτή. Κάθε απαίτηση μπορεί να εκλεπτύνεται ιεραρχικά σε επιμέρους δηλωτικές απαιτήσεις, οι οποίες αποσαφηνίζουν κάθε υπόσχεση.

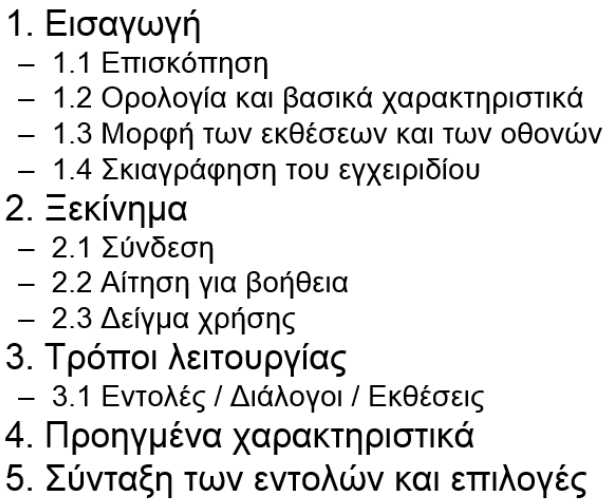
Το αποτελέσμα που προκυπτεί είναι μια περιγραφή του τί οι προγραμματιστές θα πρέπει να παράξουν, γραμμένο με επαρκείς λεπτομέρειες ώστε να είναι ξεκάθαρη η διάκριση μεταξύ των αποδεκτών και των μη αποδεκτών λύσεων, χωρίς ομως να

περιγράφει το πως το προτεινόμενο σύστημα θα σχεδιάσει και θα υλοποιήσει αυτές τις λύσεις.

3.2.4.3 Έγγραφα Απαιτήσεων

Τα έγγραφα των απαιτήσεων χωρίζονται στις εξής κατηγορίες [3], [44], [45]:

- **Το έγγραφο προοπτικής** (vision document) συνδέει το προϊόν του λογισμικού με τους επιχειρησιακούς στόχους και τις επιχειρησιακές απαιτήσεις. Παρέχει μία συνοπτική περιγραφή των λειτουργικών χαρακτηριστικών του προϊόντος, της εμβέλειάς τους, του προφίλ των ενδιαφερομένων, του περιβάλλοντος λειτουργίας και άλλων υψηλού επιπέδου απαιτήσεων.
- **Το μοντέλο περιπτώσεων χρήσης** χρησιμοποιείται κατά την εξαγωγή των απαιτήσεων και τεκμηριώνει τις λειτουργικές απαιτήσεις του λογισμικού από τη σκοπιά του χρήστη.
- **Το προκαταρκτικό εγχειρίδιο χρήστη** (preliminary user's manual) απευθύνεται στους χρήστες του υπό ανάπτυξη συστήματος και στοχεύει να περιγράψει πώς θα λειτουργεί το σύστημα, όταν ολοκληρωθεί η ανάπτυξή του. Η περιγραφή αυτή θεωρείται ότι διευκολύνει τους χρήστες να διατυπώσουν τις παρατηρήσεις τους επί των απαιτήσεων λογισμικού, πριν προχωρήσει η ομάδα έργου στην σχεδίαση. Πολλοί χρήστες, όταν διαβάζουν το προκαταρκτικό εγχειρίδιο χρήστη, διατυπώνουν ενστάσεις περί της ορθότητας ή αποδοτικότητας του λογισμικού, ενώ δεν έχουν συνήθως συστάσεις επί του εγγράφου των απαιτήσεων, επειδή αυτό το θεωρούν δυσνόητο για μη ειδκούς στον προγραμματισμό.Φυσικά, οι ενστάσεις των χρηστών είναι προτιμότερες κατά τη φάση του προσδιορισμού των απαιτήσεων παρά κατά τη φάση παράδοσης του συστήματος. Το εγχειρίδιο αυτό παρέχει στον μελλοντικό χρήστη του συστήματος μια εικόνα του πώς θα δουλεύει το λογισμικό και πώς θα χειρίζεται όλες τις εισόδους και εξόδους. Τέλος, το περίγραμμα ενός προκαταρκτικού εγχειριδίου χρήστη είναι όπως παρακάτω.



Εικόνα 3-7 Το περίγραμμα ενός προκαταρκτικού εγχειριδίου χρήστη

- **Το έγγραφο προδιαγραφών απαιτήσεων λογισμικού**_περιγράφει τις προδιαγραφές ενός συγκεκριμένου προϊόντος λογισμικού που εκτελεί συγκεκριμένες λειτουργίες σε ένα συγκεκριμένο περιβάλλον. Τα κύρια ζητήματα που απαντά είναι:
    - Η λειτουργικότητα, τι θα κάνει το λογισμικό.
    - Οι εξωτερικές διεπαφές, πώς το λογισμικό θα αλληλεπιδρά με τους χρήστες, άλλα συστήματα υλικού ή και λογισμικού.
    - Οι επιδόσεις, όπως ταχύτητες, διαθεσιμότητα, χρόνοι απόκρισης κ.τ.λ.
    - Τα χαρακτηριστικά ποιότητας, όπως αξιοπιστία, μεταφερσιμότητα, συντηρησιμότητα, ασφάλεια κ.τ.λ.
    - Οι περιορισμοί σχεδίασης, όπως τήρηση προτύπων, υλοποίηση σε συγκεκριμένες γλώσσες, πολιτικές ολοκλήρωσης, όρια πόρων, λειτουργικά περιβάλλοντα κ.τ.λ.

3.2.4.3.1 Οφέλη του εγγράφου προδιαγραφών λογισμικού

Ένα καλό έγγραφο προδιαγραφών απαιτήσεων λογισμικού προσφέρει σημαντικά οφέλη σε όλους τους ενδιαφερομένους (stakeholders) των απαιτήσεων [3], [30], [45], όπως:

- Δημιουργεί τη βάση της συμφωνίας μεταξύ του πελάτη και της ομάδας ανάπτυξης για το τι θα κάνει το λογισμικό.

- Η πλήρης περιγραφή των λειτουργιών που θα εκτελεί το λογισμικό, όπως προδιαγράφονται στο ΕΠΑΛ, δίδουν τη δυνατότητα στους εν δυνάμει χρήστες του να επιβεβαιώσουν ότι αυτό θα ικανοποιεί τις ανάγκες τους ή να τις τροποποιήσουν, ώστε να ικανοποιεί τις ανάγκες τους.
- Μειώνει το κόστος ανάπτυξης, αφού «υποχρεώνει» όλους τους ενδιαφερομένους να προσδιορίσουν έγκαιρα τα χαρακτηριστικά τού υπό ανάπτυξη λογισμικού, ελαχιστοποιώντας με αυτό τον τρόπο την ανάγκη σημαντικών διορθώσεων, όταν το έργο θα έχει πλέον προχωρήσει.
- Εξασφαλίζει μια βάση για την ακριβή εκτίμηση κοστών και χρονοδιαγραμμάτων, αφού εξασφαλίζει ένα καλό προσδιορισμό του αντικειμένου του έργου ανάπτυξης.
- Εξασφαλίζει μια αναφορά για την επικύρωση και επαλήθευση των παραδοτέων του έργου.
- Διευκολύνει τη συντήρηση και αναβάθμιση του παραγόμενου λογισμικού στο μέλλον.

Τέλος, ένα καλό έγγραφο προδιαγραφών λογισμικού πρέπει να είναι:

1. **Σωστό** (correct)
2. **Σαφές** (unambiguous)
3. **Πλήρες** (complete)
4. **Συνεπές** (consistent)
5. **Επιβεβαιώσιμο** (verifiable)
6. **Τροποποιήσιμο** (modifiable)
7. **Ιχνηλατήσιμο** (traceable)

3.2.4.3.2 Πρότυπα εγγράφων προδιαγραφών λογισμικού

Επιδίωξη κάθε προτύπου είναι να παράγει έγγραφα που χαρακτηρίζονται από κατανοητότητα, τροποποιησιμότητα, συνέπεια, σαφήνεια και πληρότητα. Τα πρότυπα αυτά είναι ένας οδηγός για τη συγγραφή απαιτήσεων λογισμικού. Περιγράφουν τα απαραίτητα περιεχόμενα ενός εγγράφου απαιτήσεων λογισμικού, τη δομή και οργάνωσή τους.

Διάφοροι οργανισμοί, όπως ο IEEE, η Ευρωπαϊκή Ένωση και το Υπουργείο Άμυνας των Η.Π.Α έχουν ποικίλα πρότυπα για τον τρόπο που προσδιορίζονται και εγγράφονται οι απαιτήσεις όσον αφορά το περιεχόμενο και την μορφή τους. Τα πρότυπα του IEEE παρέχουν πολλούς τρόπους για την οργάνωση μιας προδιαγραφής απαιτήσεων κατά τάξεις ή αντικείμενα καθώς και την οργάνωσή τους σύμφωνα με τη λειτουργία του λογισμικού και την κατηγορία των χρηστών. Οι περισσότεροι προγραμματιστές συμβουλεύονται τα συγκεκριμένα πρότυπα για να προετοιμάσουν τα έγγραφά για τις προδιαγραφές των απαιτήσεων στα έργα τους. Τα πλέον γνωστά είναι τα πρότυπα:

- DoD (Department of Defence) [DoD 88a, DoD 88b, DoD 88c, DoD 88d, DoD 94, DoD 95a, DoD 95b, DoD 95c]
- NASA [NASA 89a, NASA 89b]
- NATO [NATO 01]
- ESA [ESA 87]
- IEEE [IEEE 98]

3.2.5 Επικύρωση Απαιτήσεων

Η *επικύρωση* των απαιτήσεων (validation) απαντά στο ερώτημα αν χτίζουμε το σωστό λογισμικό και έαν αυτό επικενρώνεται στις απαιτήσεις [3], [30].

Η *επαλήθευση* των απαιτήσεων (verification) απαντά στο ερώτημα εάν χτίζουμε σωστά το λογισμικό και αν επικεντρώνεται στο σχέδιο, στον κώδικα κι στα άλλα προϊόντα του λογισμικού [3], [30].

Κατά τη διαδικασία επικύρωσης των απαιτήσεων λαμάνουν χώρα οι παρακάτω ενέργειες:

- Επανεξέταση των τεθέντων στόχων για το σύστημα.
- Σύγκριση των απαιτήσεων με τους τεθέντες στόχους και επιβεβαίωση ότι όλες οι απαιτήσεις είναι αναγκαίες.
- Περιγραφή του περιβάλλοντος στο οποίο θα λειτουργήσει το σύστημα. Εξέταση των διεπαφών μεταξύ του συστήματος και άλλων συστημάτων και επιβεβαίωση ότι αυτές είναι σωστές και πλήρεις.

- Επανεξέταση των ροών δεδομένων για να επιβεβαιωθεί ότι οι απαιτήσεις ανταποκρίνονται επακριβώς στις επιταγές και προθέσεις του πελάτη.
- Επανέλεγχος όλων των απαιτήσεων για τυχόν παραλήψεις, ελλείψεις πληρότητας και ασυνέπειες.
- Εάν υπάρχει οποιοσδήποτε κίνδυνος κατά την ανάπτυξη ή τη λειτουργία του συστήματος, επισημαίνεται και τεκμηριώνεται.
- Προσδιορισμός τρόπου επαλήθευσης και επικύρωσης των απαιτήσεων κατά τη φάση του ελέγχου.

3.2.5.1 Μέθοδοι επικύρωσης των απαιτήσεων

Οι συχνότερα χρησιμοποιούμενες μέθοδοι για τη διασφάλιση της ποιότητας του λογισμικού είναι οι τεχνικές ανασκοπήσεις (technical reviews) και χρησιμοποιούνται για την επικύρωση και επαλήθευση των ενδιάμεσων προϊόντων [35]. Τα προϊόντα αυτά μπορεί να είναι τα κείμενα των απαιτήσεων, όπως το ΕΠΑΛ, σχέδια λογισμικού, εγχειρίδια χρήσης, κώδικας κ.ά.

Με τις ανασκοπήσεις γίνεται λεπτομερής εξέταση των ενδιάμεσων προϊόντων με άτυπες ή τυπικές διαδικασίες. Οι άτυπες ανασκοπήσεις μπορεί να πραγματοποιούνται σε μόνιμη βάση, εξετάζοντας συνεχώς τα παραγόμενα προϊόντα. Εντούτοις, οι τυπικές ανασκοπήσεις που είναι πιο χρονοβόρες, πραγματοποιούνται σε επιλεγμένα ορόσημα στον κύκλο ζωής του λογισμικού.

Το πλεονέκτημα των ανασκοπήσεων είναι ότι ανακαλύπτουν σφάλματα πριν από την υλοποίηση [30]. Είναι ιδιαίτερα αποδοτικές ακόμα και στον έλεγχο των προγραμμάτων και χρησιμοποιούνται ως συμπλήρωμα του κλασικού ελέγχου [35].

3.2.5.1.1 Περιηγήσεις (Walkthroughs)

Μία περιήγηση (walkthrough) είναι μία άτυπη διαδικασία τεχνικής ανασκόπησης. Ο συντάκτης ενός προϊόντος μαζί με ένα ή περισσότερα μέλη της ομάδας ανάπτυξης το εξετάζουν με στόχο τη βελτίωση της ποιότητάς του [31], [35].

Η διαδικασία καθοδηγείται από το συντάκτη ο οποίος εξηγεί στους υπόλοιπους συμμετέχοντες της συνάντησης το περιεχόμενο του προϊόντος. Οι υπόλοιποι

συμμετέχοντες εξετάζουν το προϊόν και κάνουν διάφορες παρατηρήσεις και προτάσεις για τη βελτίωση της ποιότητάς του.

Οι περιηγήσεις δεν εστιάζονται στον εντοπισμό σφαλμάτων και έχουν σχετικά μία ελεύθερη δομή. Δεν προϋποθέτουν κάποια ειδική προετοιμασία και ολοκληρώνονται σχετικά γρήγορα.

3.2.5.1.2 Επιθεωρήσεις (Inspections)

Μία τυπική διαδικασία τεχνικής ανασκόπησης είναι η επιθεώρηση (inspection). Οι επιθεωρήσεις είναι τυπικές διαδικασίες, επειδή ακολουθούν μία συγκεκριμένη διαδικασία με προδιαγεγραμμένα βήματα και δραστηριότητες [31], [33]. Οι επιθεωρήσεις εκτελούνται από μία μικρή ομάδα προσώπων που εμπλέκονται στην ανάπτυξη του λογισμικού.

Οι βασικοί ρόλοι σε μία επιθεώρηση είναι ο μεσολαβητής (moderator) ο οποίος ηγείται της επιθεώρησης και συντονίζει τις απαραίτητες ενέργειες, ο συντάκτης του ενδιάμεσου προϊόντος και οι επιθεωρητές οι οποίοι μαζί με το συντάκτη αναλαμβάνουν τον εντοπισμό σφαλμάτων.

Μία επιθεώρηση εκτελείται τυπικά σε έξι βήματα τα οποία είναι:

- **Σχεδιασμός**: αφορά τους στόχους της επιθεώρησης, τα πρόσωπα που θα παίξουν το ρόλο των επιθεωρητών, στις τεχνικές που θα ακολουθηθούν και το χρονοδιάγραμμα της επιθεώρησης
- **Επισκόπηση**: σε μία κοινή συνεδρίαση ο συντάκτης του ενδιάμεσου προϊόντος, παρουσιάζει στην ομάδα της επιθεώρησης το περιεχόμενο που προϊόντος, έτσι ώστε όλοι να αποκτήσουν την απαραίτητη οικειότητα με αυτό.
- **Προετοιμασία**: κάθε επιθεωρητής μελετά το προϊόν ξεχωριστά με στόχο την αναζήτηση των σφαλμάτων στο προϊόν. Τα σφάλματα αυτά μπορεί να είναι λογικά σφάλματα στις απαιτήσεις, λάθη στη σχεδίαση ή και σφάλματα στον κώδικα.
- **Συνεδρίαση επιθεώρησης**: σε μία κοινή συνεδρίαση συλλέγονται τα σφάλματα που έχουν εντοπιστεί από κάθε επιθεωρητή και συζητούνται στην

ομάδα. Η συνεδρίαση καταλήγει σε έναν κατάλογο θεμάτων τα οποία θα πρέπει να διορθώσει ο συντάκτης του προϊόντος.

- **Διόρθωση**: Ο συντάκτης προβαίνει στις κατάλληλες ενέργειες για τη διόρθωση των σφαλμάτων.
- **Κλείσιμο**: Ο κύκλος της επιθεώρησης ολοκληρώνεται με μία τελική επαλήθευση. Είναι αποτέλεσμα συνεργασίας του μεσολαβητή με το συντάκτη για να επιβεβαιωθεί ότι έχουν γίνει όλες οι απαραίτητες αλλαγές για τη διόρθωση του προϊόντος.

3.2.5.1.3 Ιχνηλάτηση απαιτήσεων

Η ιχνάλητηση των απαιτήσεων είναι η εγκαθίδρυση ενός μηχανισμού μονοπατιών εξάρτησης μεταξύ των απαιτήσεων και των υπόλοιπων στοιχείων του λογισμικού. Η βασική τεχνική ιχνηλάτησης των απαιτήσεων είναι οι πίνακες ιχνηλάτησης (traceability tables) [33], [35].

Με τους πίνακες ιχνηλάτησης δημιουργούμε αμφίδρομα μονοπάτια ιχνηλάτησης μεταξύ απαιτήσεων και σχεδίου ή κώδικα. Ο πίνακας ιχνηλάτησης μας καθοδηγεί από κάθε απαίτηση σε εκείνο το στοιχείο του σχεδίου ή της μονάδας λογισμικού που την καλύπτει.

Ο πίνακας χρησιμοποιείται και αντίστροφα. Μας δίνει το μονοπάτι από μία μονάδα λογισμικού ή από κάποιο στοιχείο του σχεδίου προς τις απαιτήσεις που του το προδιαγράφουν.

3.3 Συμπεράσματα επι των μεθόδων εκμαίευσης απαιτήσεων

3.3.1 Ιστορίες Χρήσης και Περιπτώσεις Χρήσεις (User Stories Vs Use Cases)

Υπάρχει μία σχετική σύγχυση όσον αφορά τις διαφορές που υπάρχουν μεταξύ των περιπτώσεων χρήσης, των ιστοριών χρήσης και των σεναρίων χρήσης για την ανάλυση των απαιτήσεων [43], [44], [45]. Πολλές φορές παρατηρείται ότι αυτές οι έννοιες ταυτίζονται μιας

και δεν διευκρινίζονται οι διαφορές τους, γεγονός που καθυστερεί την υλοποίηση και μπορεί

να αποβεί καθοριστικός παράγοντας αποτυχίας κυρίως για νεοφυείς επιχειρήσεις όπου οι

χρηματικοί πόροι είναι περιορισμένοι και τα περιθώρια λάθους πολύ μικρά.

Εντούτοις, οι βασικές διαφορές αυτών είναι οι εξής:

- Μία ιστορία χρήσης αποτελεί ένα ανεπίσημο έγγραφο το οποίο χρησιμοποιείται για να διευκολύνει την επικοινωνία μεταξύ των εμπλεκόμενων μελών και δεν εμπεριέχει όλες τις απαιτούμενες λεπτομερίες που χρειάζονται για την ανάλυση των απαιτήσεων.
- Μια πέριπτωση χρήσης αποτελεί ένα επίσημο έγγραφο το οποίο περιγράφει όχι μόνο την ‘’φυσιολογική’’ ροή των βημάτων που πρέπει να ακολουθηθούν για μία αλληλεπίδραση αλλά και ‘’εναλλακτικές’’ ροές. Μία περίπτωση χρήσης εμπεριέχει όλες τις απαιτούμενες λεπτομερείες που χρειάζονται για την ανάλυση των λειτουργικών απαιτήσεων.
- Ένα σενάριο χρήσης περιγράφει ένα πραγματικό παράδειγμα αλληλεπίδρασης ενός ή πολλών ανθρώπων με το υπό ανάπτυξη σύστημα. Περιγράφει τα βήματα, τα γεγονότα και τις ενέργειες που λαμβάνουν χώρα κατά τη διάρκεια της αλληλεπίδρασης. Ένα σενάριο χρήσης μπορεί να είναι ιδιαιτέρως λεπτομερές, αναφέροντας επακριβώς πως κάποιος χρήστης αλληλεπιδρά με το σύστημα ή ακομά και επιχειρηματικές κινήσεις που πρέπει να γίνουν για τη βελτίωση της αλληλεπίδρασης, δηλαδή περιγράφει μη-λειτουργικές, ποιοτικές απαιτήσεις όμως δεν περιγράφει τις τεχνικές λεπτομέρειες που αφορούν την αλληλεπίδραση.

Συμπερασματικά:

	Ιστορίες Χρήσης (User Stories)	**Περιπτώσεις Χρήσης (Use Cases)**
Ομοιότητες	Διατυπώνονται στην καθημερινή γλώσσα των χρηστών.	Διατυπώνονται στην καθημερινή γλώσσα των χρηστών για τη διευκόλυνση της

	Βοηθούν τον αναγνώστη να κατανονήσει ποιο είναι το υπό ανάπτυξη σύστημα. Πρέπει να συνοδεύονται από διαδικασίες δοκιμών αποδοχής για την αποσαφήνιση της συμπεριφοράς του συστήματος όπου αυτή είναι διφορούμενη.	επικοινωνίας μεταξύ των ενδιαφερόμενων μελών. Πρέπει να συνοδεύονται και να επαληθεύονται από περιπτώσεις δοκιμής.
Διαφορές	Αναλύουν τις απαιτήσεις με βασικό κριτήριο το χρονικό προγραμματισμό τους.Οι ιστορίες αναλύονται σε μικρότερες μέχρι να αποτελέσουν κομμάτι επόμενης έκδοσης. Παρέχουν μία μικρής κλίμακας παρουσίαση των πληροφοριών χωρίς τεχνικές λεπτομέρειες, γεγονός το οποίο μπορεί να οδηγήσει σε παρερμηνείες. Συνήθως γράφονται σε μικρά κομμάτια χαρτιού.	Οργανώνουν τις απαιτήσεις με τέτοιο τρόπο προκειμένου να σχηματίσουν μία αφήγηση του πως οι χρήστες συνδέονται και χρησιμοποιούν το σύστημα. Περιγράφουν αλληλουχίες αλληλεπιδράσεων με επίσημους όρους.Μία περίπτωση χρήσης καλείται να παρέχει επαρκείς πληροφορίες προκειμένου να είναι κατανοητή από μόνη της.

	Είναι αναλυτικές όσον αφορά τις απαιτήσεις των χρηστών διότι πρέπει να είναι εξ ολοκλήρου άρτιες προκειμένου να εισαχθούν στην επόμενη έκδοση.	Συνήθως αποτελούν αυτόνομα έγγραφα και παρουσιάζονται μέσω διαγραμμάτων UML. Δεν αναφέρουν μη λειτουργικές απαιτήσεις. Μία περιπτώση χρήσης μπορεί να αντιστοιχεί εξ ολοκλήρου σε μία ιστορία χρήσης.Ωστόσο μία ιστορία χρήσης μπορεί να εμπεριέχει μία ή περισσότερες περιπτώσεις χρήσης.

Πίνακας 13 Συμπερασματικός πίνακας μεταξύ ιστοριών χρήσης και περιπτώσεων χρήσης

3.3.2 Εκμαίευση απαιτήσεων σε νεοφυείς επιχειρήσεις

Τα σλόγκαν ‘’Done is better than perfect’’ και ‘’Move fast and break things’’ είναι εκφράσεις όπου οποιοσδήποτε έχει έρθει σε επαφή με το οικοσύστημα των startups τις γνωρίζει. Αυτό που κρύβεται πίσω από αυτές τις εκφράσεις δεν είναι τίποτε περισσότερο από μια περιλήψη των πρακτικών που εφαρμόζονται στην αγορά. Αναλύσαμε τις συγκεκριμένες πρακτικές προκειμένου να εντοπίσουμε που υπάρχουν κενά στην μεθοδολογία τους και χρειάζεται επιπλέον έρευνα.

Η διαχείρηση των διαδικασιών (process management) αντιπροσωπεύει όλες τις δραστηριότητες της μηχανικής απαιτήσεων (requirement engineering) που χρησιμοποιούνται για τη διαχείρηση της ανάπτυξης του προϊόντος σε νεοσύστατες επιχειρήσεις. Η ικανότητα της ευελιξίας σε συνεχείς αλλαγές είναι απαραίτητη για μία νεοσύστατη εταιρεία, οπότε οι ευέλικτες τεχνικές ανάπτυξης προϊόντων (agile development methods) θεωρούνται οι πιο βιώσιμες τεχνικές επειδή διευκολύνουν

τις αλλαγές και επιτρέπουν την ανάπτυξη του προϊόντος να προσαρμοστεί στην επιχειρηματική στρατηγική. Η γρήγορη έκδοση αρχικών εκδόσεων με επαναληπτικές και εξελικτικές προσεγγίσεις επιτρέπει την μείωση του χρόνου από την αρχική σύλληψη της ιδέας μέχρι και την τελική παραγωγή του προϊόντος λογισμικού. Μία παραλλαγή της ευέλικτης ανάπτυξης είναι η Lean ανάπτυξη η οποία εστιάζει στον εντοπισμό των επικίνδυνων σημείων κατά τη διαδικασία ανάπτυξης του προϊόντος λογισμικού και στη γρήγορη δημιουργία κάποιου αρχικού συστήματος το οποίο συστηματικά ελέγχεται και τροποποιείται μέχρι την επόμενη προσαύξησή του. Ενώ υπάρχουν πολλές μεθοδολογίες οι οποίες υποστηρίζουν την ταχεία πρωτοτυποποίηση, καμία μεθοδολογία δεν ακολουθείται αυστηρά από τις νεοφυείς επιχειρήσεις. Η αβεβαιότητα και οι γρήγορες αλλαγές στις ανάγκες αυτών των επιχειρήσεων της οδηγούν στο να χρησιμοποιούν περιστασιακά κάποια μεθοδολογία που αφορά την υλοποίηση των short-term στόχων της και να την προσαρμόζουνε έιτε ακόμα και να επιλέγουν άλλη μέθοδο καθώς μαθαίνουν περισσότερα για τους χρήστες που ανήκουν στην αγορά στην οποία επιθυμούν να εισάγουν το προϊόν τους. Συνοψίζοντας, η ad hoc επιλογή μεθοδολογιών ανάπτυξης στις νεοφυείς επιχειρήσεις οφείλεται στους παρακάτω λόγους:

Λόγοι	Περιγραφή
Έλλειψη Πόρων	Οι οικονομικοί, φυσικοί και ανθρώπινοι πόροι είναι περιορισμένοι.
Υψηλή Δραστικότητα	Οι στόχοι και οι ανάγκες των νεοφυών επιχειρήσεων μεταβάλλονται συχνά λόγω των γρήγορων αλλαγών που αφορούν την τεχνολογία, την αγορά και το προϊόν τους.
Καινοτομία	Δεδομένου του υψηλά ανταγωνιστικού οικοσυστήματος, οι νεοφυείς επιχειρήσεις διαρκώς αναζητούν τμήματα της αγοράς στα οποία μπορούν να καινοτομήσουν.

Γρήγορη Εξέλιξη	Οι νεοφυείς επιχειρήσεις έχουν ως στόχο τη γρήγορη ανάπτυξη (scale up)
Πίεση Χρόνου	Το εξωγενές περιβάλλον συχνά αναγκάζει τις startups να βγάζουν συχνά prototypes και να δουλεύουν υπό πίεση.
Στήριξη από τρίτους	Λόγω έλλειψης πόρων οι startups στηρίζονται σημαντικά σε τρίτους προκειμένου να αναπτύξουν το προϊόν τους.
Μικρή Ομάδα	Οι startups συνήθως ξεκινάνε ως μία ομάδα με λίγα μέλη, γεγονός που δυσκολεύει το διαχωρισμό εργασιών.
Έλλειψη Εμπειρίας	Συνήθως οι startups αποτελούνται από μηχανικούς λογισμικού οι οποίοι είτε δεν έχουν αποφοιτήσει είτε έχουν λιγότερη από τριετή εμπειρία.

Πίνακας 14 Λόγοι επιλογής ευέλικτων μεθόδων ανάπτυξης λογισμικού από καινότομες ομάδες

3.3.3 Συμπεράσματα τεχνικών εκμαίευσης απαιτήσεων

Ο παρακάτω πίνακας παρουσιάζει συνοπτικά τα βασικά χαρακτηριστικά κάθε τεχνικής εκμαίευσης καθώς και τα πλεονεκτήματα και μειονεκτήματα αυτών.

Τεχνική Εκμαίευσης	Ενδείκνυται για	Είδος δεδομένων (input)	Πλεονεκτήματα	Μειονεκτήματα
Συνεντεύξεις	Την αποσαφήνιση		Ο αναλυτής μπορεί να	

(Interviews)	πολλών και διαφορετικών ζητημάτων που αφορούν την ανάπτυξη του συστήματος	Ελάχιστα ποσοτικά δεδομένα, αλλά πολλά χρήσιμα ποιοτικά	καθοδηγήσει τον συνεντευξιαζόμενο. Βελτιώνει την επικοινωνία μεταξύ προγραμματιστών και χρηστών.	Σπαταλάται πολύτιμος χρόνος. Συνεντεύξεις μέσω Η/Υ μπορεί να αποθαρρύνουν τον συνεντευξιαζόμενο
Ομάδες Εστίασης (Focus Groups)	Τη συλλογή πολλών διαφορετικών απόψεων	Ελάχιστα ποσοτικά δεδομένα, αλλά πολλά χρήσιμα ποιοτικά	Εστιάζει σε ζητήματα είτε ομοφωνίας είτε έντονης διαφωνίας. Βελτιώνει την επικοινωνία προγραμματιστών και χρηστών.	Η πιθανότητα ύπαρξης ανθρώπων με επιβλητικούς χαρακτήρες (*dominant characters*)
Ερωτηματολόγια (Questionnaires)	Τη συλλογή απαντήσεων για συγκεκριμένα ζητήματα	Ποσοτικά και ποιοτικά δεδομένα	Μπορεί να απευθυνθεί σε μεγάλο έυρος ατόμων με μικρό κόστος	Ο σχεδιασμός του είναι πολύ σημαντικός για την επιτυχία και την αξιοπιστία του

				Οι απαντήσεις μπορεί να είναι ελάχιστες και ίσως όχι αυτές που χρειάζετα ο αναλυτής
Παρατήρηση (Naturalistic Observation)	Την αποσαφήνιση του τρόπου χρήσης του προϊόντος από τον τελικό χρήστη	Ποιοτικά δεδόμενα	Η παρατήρηση παρέχει ποιοτικές πληροφορίες όπου οι άλλες τεχνικές αδυνατούν	Πολύ χρονοβόρα διαδικασία --- Μεγάλος όγκος δεδομένων
Ανάλυση Τεκμηριώσεων (Documentation Analysis)	Την κατανόηση των υπάρχουσων διαδικασιών και κανόνων	Ποσοτικά δεδομένα	Δεν χρειάζεται οι χρήστες να δαπανήσουν χρόνο	Η καθημερινή χρήση του λογισμικού διαφερεί από τις διαδικασίες των τεκμηριώσεων

Πίνακας 15 Συμπερασματικός πίνακας επί των τεχνικών εκμαίευσης απαιτήσεων

4 *Σχετικές εργασίες σε καινοτόμα προϊόντα λογισμικού και προδιαγραφές*

Όπως αναφέρθηκε στις ενότητες 1.2 και 3.3.2 οι νεοφυείς επιχειρήσεις καινοτόμων προϊόντων λογισμικού προτιμούν τις ευέλικτες μεθοδολογίες ανάπτυξης λογισμικού. Το συγκεκριμένο φαινόμενο εμφανίζεται να απασχολεί ιδιαιτέρως την ακαδημαϊκή κοινότητα η οποία τα τελευταία χρόνια έχει δείξει ιδιαίτερο ενδιαφέρον για τον τρόπο δημιουργίας καινοτόμων προϊόντων λογισμικού και έχει εστιάσει την έρευνά της στη δημιουργία μεθόδων άντλησης, ανάλυσης και μοντελοποίησης των απαιτήσεων σε συνεχώς μεταβαλόμενα περιβάλλοντα όπως αυτά των νεοφυών επιχειρήσεων μιας και οι απαιτήσεις των τελικών χρηστών ενός λογισμικού αποτελούν το σημαντικότερο κριτήριο για την επιτυχία ή την αποτυχία ενός προϊόντος λογισμικού.

4.1 Σχεδίαση με επίκεντρο το χρήστη (User-Centered Design - UCD)

Η σχεδίαση με επίκεντρο το χρήστη, γνωστή ως UCD, είναι ένας ευρέως γνωστός όρος ο οποίος περιγράφει τις διαδικασίες σχεδιασμού απαιτήσεων στις οποίες οι τελικοί χρήστες επηρεάζουν τον τρόπο σχεδιασμού του λογισμικού [46].Επειδή είναι

μία αρκετά γενική φιλοσοφία σχεδιασμού, υπάρχουν ποικίλες μέθοδοι υλοποίησής της. Για παράδειγμα, ορισμένες μέθοδοι UCD συμβουλεύονται τους χρήστες σχετικά με τις ανάγκες τους και τους εντάσσουν στην ομάδα έργου σε συγκεκριμένες χρονικές στιγμές κατά τη διάρκεια της ανάπτυξης και συνηθέστερα κατά τη διαδικασία εκμαίευσης των απαιτήσεων και κατά διαδικασία ελέγχου χρησιμότητας (usability testing). Από την άλλη πλευρά, υπάρχουν μέθοδοι UCD στις οποίες οι χρήστες έχουν μια βαθιά και ουσιαστική επίδραση στη σχεδίαση με το να συμμετέχουν σε όλη τη διαδικασία ανάπτυξης του προϊόντος, τεχνικές στις οποίες θα επικεντρωθούμε μιας και χρησιμοποιούνται ευρύτερα από ομάδες ανάπτυξης καινοτόμων προϊόντων λογισμικού.

Ο όρος UCD προέρχεται από τον Donald Norman, ερευνητή στο University of California San Diego, τη δεκαετία του 1980 [47].

Σύμφωνα με τον Donald Norman βασικά στοιχεία τα οποία πρέπει να λαμβάνονται υπόψη κατά τη διαδικασία σχεδίασης είναι τα εξής:

1. Ευκολία στον καθορισμό των δράσεων/ενεργειών οι οποίες είναι δυνατόν να υλοποιηθούν ανά πάσα στιγμή.
2. Όλες οι ενεργείες κατά τη διαδικασία ανάπτυξης να είναι οροτές συμπεριλαμβανομένων του εννοιολογικού μοντέλου του συστήματος (conceptual system model), των εναλλακτικών ενεργειών καθώς και των αποτελεσμάτων των δράσεων.
3. Ευκολία στην αξιολογήση του συστήματος ανά πάσα στιγμή.
4. Υπάρξη οπτικής αναπαράστασης των σχέσεων μεταξύ στόχων/αναγκών και επιθυμητών ενεργείων, ενεργειών και επιθυμητών αποτελεσμάτων.

Οι συγκεκριμένες ενέργειες τοποθετούν τον χρήστη στο επίκεντρο της σχεδίασης. Ο ρόλος του σχεδιαστή/αναλυτή είναι να διευκολύνει το έργο για το χρήστη και να σιγουρευτεί ότι ο χρήστης είναι σε θέση να χρησιμοποιήσει το προϊόν όπως ήταν αρχικά προσχεδιασμένο με την ελάχιστη προσπάθεια προκειμένου να μάθει πώς να το χρησιμοποιεί. Ο Norman επισημαίνει πως τα μακροσκελή και ακατανόητα εγχειρίδια χρήσης δεν επικεντρώνονται στο χρήση και δεν τον βοηθούν στη χρήση του προϊόντος λογισμικού.

Είναι πολύ σημαντικό οι σχεδιαστές να καταλάβουν ποιοι είναι οι επίδοξοι χρήστες του τελικού προϊόντος λογισμικού. Προφάνως χρήστες είναι και τα άτομα τα οποία χρησιμοποιούν το τελικό προϊόν λογισμικού για κάποια εργασία, αλλά δεν είναι μόνο αυτά. Οι άνθρωποι που διαχειρίζονται τους χρήστες έχουν και αυτοί απαιτήσεις από το σύστημα. Ακόμη τα εμπλεκόμενα μέλη της επιχείρησης που αναπτύσσει το σύστημα και επηρεάζονται έμμεσα από αυτό έχουν απαιτήσεις οι οποίες επιβάλλεται να συμπεριληφθούν κατά τη διαδικασία σχεδίασης. Ο Eason περιγράφει τρία διαφορετικά είδη χρηστών: τους πρωταρχικούς (primary), τους δευτερεύοντες (secondary) και τους τριτογενείς (tertiary) [48]. Πρωταρχικοί χρήστες είναι αυτοί που χρησιμοποιούν πραγματικά το προϊόν λογισμικού. Δευτερεύοντες είναι εκείνοι που χρησιμοποιούν περιστασιακά το προϊόν ή το χρησιμοποιούν μέσω τρίτων, και τριτογενείς είναι οι χρήστες οι οποίοι θα επηρεαστούν από τη χρήση του προϊόντος λογισμικού και συμμετέχουν στη λήψη αποφάσεων όσον αφορά την αγορά του. Η επιτυχής σχεδίαση ενός προϊόντος λογισμικού θα πρέπει να λαμβάνει υπόψη το ευρύ φάσμα των εμπλεκόμενων μελών. Κατά τη διαδικασία σχεδίασης δεν είναι απαραίτητο όλα τα είδη εμπλεκόμενων μελών να εκπροσωπούνται στην ομάδα σχεδιασμού, αλλά το αποτελέσμα της σχεδίασης οφείλει να τους εμπεριέχει [47], [48].

Μόλις τα εμπλεκόμενα μέλη έχουν εντοπιστεί και έχει διενεργηθεί η απαραίτητη έρευνα για τις ανάγκες/απαιτήσεις τους μέσω αναλύσεων, οι σχεδιαστές είναι σε θέση να αναπτύξουν εναλλακτικές λύσεις σχεδίασεις οι οποίες εν συνεχεία να αξιολογηθούν από τους τελικούς χρήστες. Αυτές οι λύσεις σχεδίασης μπορούν να είναι απλά γραμμένες σε χαρτή με σχήματα που απεικονίζουν τον τρόπο αλληλεπίδρασης του χρήστη με τη διεπαφή στην αρχή της διαδικασίας σχεδίασης.

Ακούγοντας τους χρήστες να συζητούν τις εναλλακτικές λύσεις σχεδίασης, οι σχεδιαστές μπορούν να κατανοήσουν το επιδιωκόμενο αποτελέσμα που επιθυμούν οι χρήστες και να αντλήσουν σημαντικές πληροφορίες για τις απαιτήσεις των χρηστών οι οποίες δεν εκμαιεύτηκαν από τις αρχικές συνεντεύξεις ή παρατηρήσεις. Καθώς ο κύκλος σχεδιασμού εξελίσσεται, πρωτότυπα παράγονται από την ομάδα σχεδιασμού και δίνονται στους χρήστες προς δοκιμή. Σε αυτό το σημείο, οι σχεδιαστές πρέπει να δώσουν ιδιαίτερη σημασία στις αξιολογήσεις των χρηστών,

δεδομένου ότι θα βοηθήσει στον εντοπισμό των μετρικών ευχρηστίας. Οι μετρικές ευχρηστίας αναφέρονται σε ζητήματα που σχετίζονται με την αποδοτικότητα, την αποτελεσματικότητα, τη χρησιμότητα και την ασφάλεια του λογισμικού και με την υποκειμενική ικανοποίηση των χρηστών από αυτό. Σύμφωνα με τον Preece είναι ιδιαιτέρως δύσκολο για τους σχεδιαστές να γνωρίζουν είτε ακόμα να φαντάζονται όλα τα κριτήρια ευχρηστίας τα οποία είναι σημαντικά για τους χρήστες. Μόνο μέσω των ανατροφοδοτήσεων και με συμμετοχή των χρηστών στη σχεδίαση του προϊόντος λογισμικού μπορούν να αξιολογηθούν τα κριτήρια ευχρηστίας τους. Σύμφωνα με τη UCD, ο παρακάτω πίνακας παρουσίαζει τις τεχνικές με τις οποίες η ομάδα έργου συμπεριλαμβάνει τους χρήστες στη σχεδίαση και στην ανάπτυξη του προϊόντος λογισμικού.

Τεχνική	**Σκοπός**	**Στάδιο στον κύκλο σχεδίασης του προϊόντος**	**Σε ποιους απευθύνεται**
Άτυπες συνεντεύξεις και ερωτηματολόγια	Συλλογή δεδομένων σχετικών με τις ανάγκες και προσδοκίες των χρηστών --- Αξιολόγηση των διάφορων εναλλακτικών σχεδίασης και των πρωτοτύπων	Αρχή του έργου	Πρωταρχικούς, Δευτερεύοντες & Τριτογενείς Χρήστες
Συνεντεύξεις εργασίας και ερωτηματολόγια	Συλλογή δεδομένων σχετικών με την ακολουθία των	Αρχή του κύκλου σχεδίασης	Πρωταρχικούς χρήστες

	εργασίων που θα εκτελεί το προϊόν λογισμικού	του προϊόντος	
Ομάδες εστίασης	Συζήτηση πάνω σε ζητήματα που αφορούν τις απαιτήσεις	Αρχή του κύκλου σχεδίασης του προϊόντος	Όλα τα εμπλεκόμενα μέλη
Παρατήρηση (On-site Observation)	Συλλογή πληροφοριών σχετικά με το περιβάλλον στο οποίο θα χρησιμοποιείται το προϊόν λογισμικού	Αρχή του κύκλου σχεδίασης του προϊόντος	Πρωταρχικούς χρήστες
Περιηγήσεις (Walkthroughs)	Αξιολόγηση των διάφορων τεχνικών σχεδίασης --- Συλλογή πληροφορίων σχετικών με τις ανάγκες και προσδοκίες των χρηστών --- Αξιολόγηση του πρωτότυπου	Αρχή του έργου --- Στην μέση του κύκλου σχεδίασης του προϊόντος --- Έπειτα από κάθε προσάυξηση	Όλα τα εμπλεκόμενα μέλη
	Συλλογή ποσοτικών δεδομένων που		Πρωταρχικούς, Δευτερεύοντες

Έλεγχος χρησιμότητας (Usability Testing)	αφορούν την ευχρηστία του προϊόντος (μετρικές ευχρηστίας)	Τελικό στάδιο σχεδίασης	& Τριτογενείς χρήστες
Συνεντεύξεις και ερωτηματολόγια	Συλλογή ποιοτικών δεδομένων σχετικών με την ικανοποίηση των χρηστών από το προϊόν λογισμικού	Τελικό στάδιο σχεδίασης	Πρωταρχικούς, Δευτερεύοντες & Τριτογενείς Χρήστες

Πίνακας 16 Τεχνικές συμμετοχής χρηστών στην ανάπτυξη λογισμικού

4.2 Πανταχού παρούσα υπολογιστική

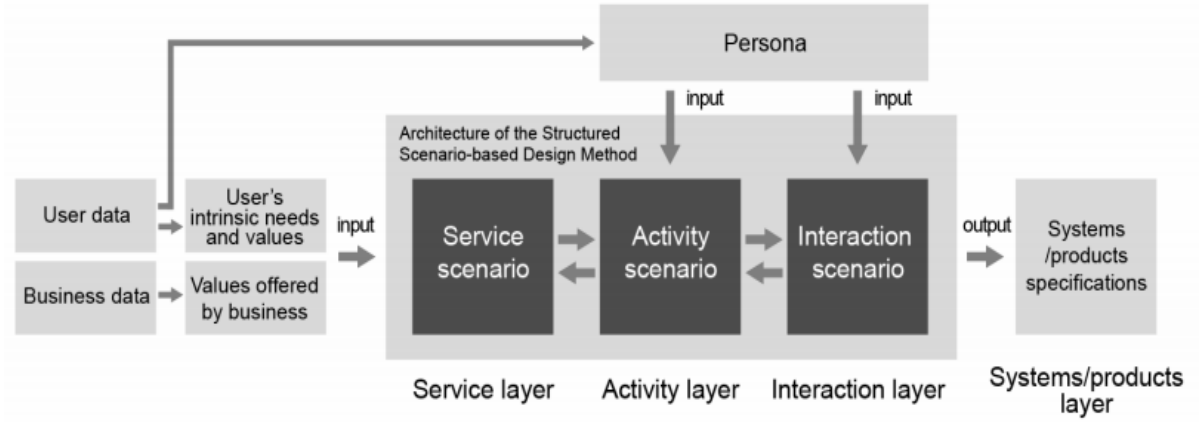
Η έννοια της πανταχού παρούσας υπολογιστικής (ubiquitous computing) που θεσπίστηκε από τον Mark Weiser το 1993 πλεόν αποτελεί πραγματικότητά [49]. Οι περισσότερες νεοφυείς επιχειρήσεις δημιουργούν προϊόντα τα οποία συμπεριλαμβάνονται στην έννοια αυτή.Η συγκεκριμένη έννοια αναφέρεται στην τεχνολογία λογισμικού, όπου η υπολογιστική εμφανίζεται παντού [50]. Σε αντίθεση με το desktop computing, η πανταχού παρούσα υπολογιστική μπορεί να υπάρξει σε οποιαδήποτε συσκευή, σε οποιαδήποτε θέση και σε οποιαδήποτε μορφή. Ένας χρήστης αλληλεπιδρά με τον υπολογιστή με πολλές διαφορετικές μορφές, συμπεριλαμβανομένων των φορητών υπολογιστών και των tablets ή ακομά και με terminals σε καθημερινά αντικείμενα, όπως ένα ψυγείο ή ένα ζευγάρι γυαλιά. Οι βασικές τεχνολογίες για την υποστήριξή του περιλαμβάνουν το διαδίκτυο, διάφορα λειτουργικά συστήματα, κώδικα για κινητά τηλέφωνα, αισθητήρες και μικροεπεξεργαστές.

4.3 Δομημένη μέθοδος σχεδίασης βασισμένη σε σενάρια χρήσης - (Structured Scenario-Based Design Method)

Οι Koji Yanagida, Yoshihiro Ueda, Kentaro Go, Katsumi Takahashi,Seiji Hayakawa, και Kazuhiko Yamazaki από το Kurashiki University δημιούργησαν το 2009 μία καινοτόμα

μέθοδο για τη σχεδίαση καινοτόμων προϊόντων λογισμικού η οποία επικεντρωνεται στην εκμαίευση των απαιτήσεων των διάφορων εμπλεκόμενων ενός συστήματος λογισμικού βασισμένοι στη δημιουργία σεναρίων και personas [51].

Σύμφωνα με το μοντέλο τους όσον αφορά τη σχεδίαση συστημάτων με επίκεντρο το χρήστη υπάρχουν δύο διαφορετικές προσεγγίσεις. Η πρώτη προσέγγιση αφορά τη σχεδίαση η οποία εστιάζεται στην επίλυση προβλημάτων (problem solving approach), όπου η χρησιμότητα των υπάρχοντων συστημάτων αξιολογείται και στη συνέχεια τα υπάρχοντα προβλήματα λύνονται. Η δεύτερη προσέγγιση η οποία μας ενδιαφέρει κυρίως, γιατί αφορά την καινοτομία και τις νεοφυείς επιχειρήσεις, εστιάζεται στη σχεδίαση συστημάτων με επίκεντρο το όραμα του συστήματος (vision design approach) δηλαδή τη δυνητική προστιθέμενη αξία όπου θα παρέχει στους χρήστες ένα σύστημα, όπου νέες υπηρεσίες οι οποίες ακόμα δεν υπάρχουν δημιουργούνται με βάση τις υπάρχουσες ανάγκες και αξίες των χρηστών. Το συγκεκριμένο μοντέλο μπορεί να εφαρμοστεί από καινοτόμες επιχειρήσεις οι οποίες προσπαθούν να δημιουργήσουν προϊόντα λογισμικού. Το βασικό μοντελό είναι το εξής:



Εικόνα 4-1 Βασικό μοντέλο Structured Scenario-based Design Method [51]

1. **Η χρήση τεσσάρων επιπέδων με σκοπό τη διευρεύνηση της διαδικασίας ανάπτυξης του σύστήματος.**

   Η πρώτη βαθμίδα, αυτή των υπηρεσιών (Service layer) χρησιμοποιείται με σκοπό να συζητηθούν οι αξίες και οι ανάγκες των χρηστών. Η δεύτερη βαθμίδα, αυτή της δραστηριότητας (Activity layer) χρησιμοποιείται προκειμένου να αποσαφηνιστούν εις βάθος οι ενέργειες των χρηστων. Η τρίτη βαθμίδα αύτη της αλληλεπίδρασης (Interaction layer) χρησιμοποιείται προκειμένου να αποσαφηνιστούν οι

αλληλεπιδράσεις των χρηστών με το σύστημα και τέλος, η τέταρτη βαθμίδα αφορά τα τεχνικά χαρακτηριστικά του υπό ανάπτυξη συστήματος (Service/Product layer).

2. **Η χρήση σεναρίων και personas**

   Η σχεδίαση με επίκεντρο το χρήστη προϋποθέτει ότι οι εμπειρίες του χρήστη είναι άρρηκτα συνδεδεμένες με το υπό ανάπτυξη σύστημα. Για το σκοπό αυτό, το συγκεκριμένο μοντέλο χρησιμοποιεί σενάρια σε κάθε στάδιο της διαδικασίας ανάπτυξης. Επίσης χρησιμοποιεί personas προκειμένου να αναπαραστήσει του χρήστες. Ο συνδυασμός σεναρίων και personas επιτρέπει στην ομάδα έργου να αποσαφηνίσει τους στόχους και τις ανάγκες των χρηστών με ιδιαίτερη λεπτομέρεια.

3. **Η έξοδος δομημένων σεναρίων και πληροφορίων όπου ορίζουν τις προδιαγραφές του συστήματος**

   Στο συγκεκριμένο μοντέλο οι ανάγκες και οι αξίες των χρηστών, οι αξίες και οι στόχοι της επιχείρησης καθώς και οι personas χρησιμοποιούνται ως πληροφορίες εισόδου. Συζητήσεις μεταξύ των εμπλεκόμενων μελών διενεργόνται σε κάθε βαθμίδα και τρία διαφορετικά σενάρια δημιουργουνται (υπηρεσιών, δραστηριότητας και αλληλεπίδρασης). Αυτά τα σενάρια χρησιμοποιούνται ως χρήσιμες πληροφορίες προκειμένου να οριστούν οι προδιαγραφές του συστήματος. Ο παρακάτω πίνακας παρουσιάζει τα χαρακτηριστικά του κάθε σεναρίου.

4. **Η διαδικασία σχεδιασμού συνδέει τις υπηρεσίες και τις προδιαγραφές του συστήματος μέσω της εμπειρίας του χρήστη**

   Το συγκεκριμένο μοντέλο προτείνει μία διαδικασία σχεδίασης όπου η συζήτηση σχετικά με τις ανάγκες και τις αξίες των χρηστών ξεκινά στο επίπεδο των υπηρεσιών το οποίο αποτελεί την πρώτη από τις τέσσερεις βαθμίδες. Έπειτα, οι ανάγκες και οι αξίες των χρηστών μετουσιώνονται σε δραστηριότητες και αλληλεπιδράσεις του χρήστη με το σύστημα προκειμένου να αποσαφηνιστούν οι καινούργιες υπηρεσίες. Τέλος, οι πληροφορίες και τα δεδομένα που έχουν συλλεχθεί χρημοποιούνται προκειμένου να δημιουργηθούν οι προδιαγραφές του συστήματος οι οποία θα ικανοποιούν τις ανάγκες του χρήστη. Το συγκεκριμένο μοντέλο σχεδίασης επιτρέπει στην ομάδα έργου να συμπεριλαμβάνει καθ'όλη τη διαδικασία ανάπτυξης του συστήματος τις απαιτήσεις και τις ανάγκες των χρηστών.

4.3.1 Διαδικασία σχεδιασμού scenarios

Η συγκεκριμένη διαδικασία σχεδιασμού αποτελείται από τέσσερα βασικά βήματα.

1. **Δημιουργία δύο στοιχείων εισόδου προκειμένου να ξεκινήσει η διαδικασία σχεδιασμού.**

- **Ανάγκες χρήστη και αξίες της επιχείρησης**

  Οι ανάγκες χρήστη συλλέγονται με πολλούς και διαφορετικούς τρόπους οι οποίοι έχουν αναφερθεί σε προηγούμενα κεφάλαια. Οι αξίες που προσδίδει η επιχείρηση στο σύστημα μπορούν να προσδιοριστούν λαμβάνοντας υπόψη τα δεδομένα της επιχείρησης όπως τις ήδη υπάρχουσες τεχνολογίες της, την επιχειρησιακή της στρατηγική και το επιχειρηματικο περιβάλλον στο οποίο αναπτύσσεται.

- **Personas**

  Οι personas δημιουργούνται από τα δεδομένα και τις πληροφορίες που έχουν συλλεχθεί από τους χρήστες. Οι personas χρησιμοποιούνται προκειμένου ένα σενάριο υπηρεσίας να μετατραπεί σε σενάριο δραστηριότητας και αλληλεπίδρασης. Οι personas όπως έχει προαναφερθεί χρησιμεύουν στην πληρέστερη αποσαφήνιση της συμπεριφοράς των χρηστών μέσα στα σενάρια. Στο παρόν στάδιο μία ή περισσότερες personas δημιουργούνται.

2. **Η δημιουργία σεναρίου υπηρεσιών με βάση τις ανάγκες των χρηστών και τις αξίες της επιχείρησης**

   Το σενάριο υπηρεσιών παρουσιάζει μία συγκεκριμένη υπηρεσία του συστήματος και παρέχει μία πληρέστερη εικόνα για αυτή. Στο συγκεκριμένο στάδιο τα διαφορετικά είδη χρηστών καθώς και το τελικό σύστημα δεν έχουν αποσαφηνιστεί ακόμα. Εντούτοις ένα προφίλ χρήστη αναλύεται σε υποκατηγορίες. Η αξιολόγηση του σεναρίου λαμβάνει υπόψην την πλευρά του χρήστη αλλά και την πλευρά της επιχείρησης. Όσον αφορά τον χρήστη, ιδιαίτερη έμφαση δίνεται στην ικανοποίηση του από το σύστημα.

3. **Η δημιουργία σεναρίου δραστηριότητας με βάση το σενάριο υπηρεσιών και τις personas**

   Το σενάριο δραστηριότητας είναι ένα σενάριο που δείχνει τις δραστηριότητες των χρηστών, απεικονίζοντας σκηνές από το σενάριο υπηρεσιών και από τους

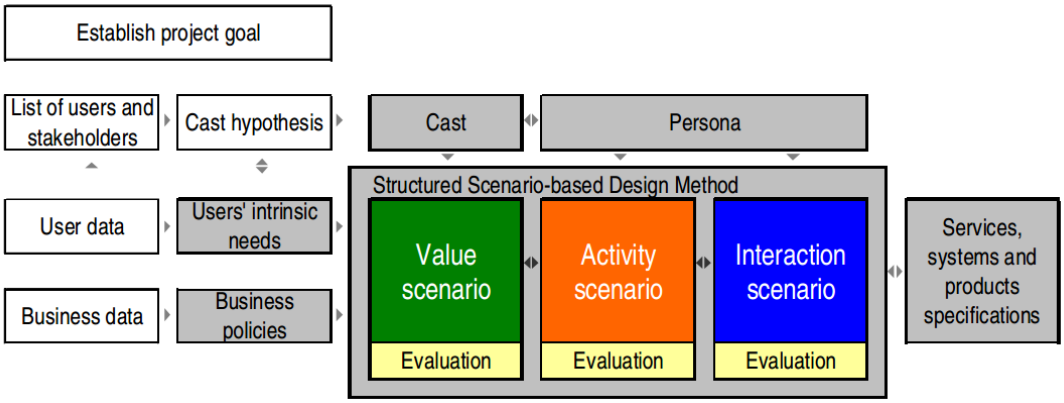
στόχους της περσόνας και περιγράφει τη ροή της δραστηριότητας του χρήστη και τα συναισθήματα του κατά τη διάρκεια όλων των σταδίων της ροής. Στο συγκεκριμένο σενάριο δεν περιγράφεται το σύστημα αλλά κύρια έμφαση δίνεται στις εμπειρίες της περσόνας. Για την αξιολόγηση του σεναρίου ιδιαίτερη έμφαση δίνεται στην αποτελεσματικότητα.

4. **Η δημιουργία σεναρίου αλληλεπίδρασης με βάση το σενάριο δραστηριότητας και τις personas**

   Το σενάριο αλληλεπίδρασης είναι ένα σενάριο που καταδεικνύει την αλληλεπίδραση των χρηστών με το σύστημα, όπου κάθε αλληλεπίδραση του ατόμου που αναπτύχθηκε στο σενάριο δραστηρίοτητας περιγράφεται με όρους αλληλουχίας χρόνου καθώς το σύστημα χρησιμοποιείται. Στο συγκεκριμένο σενάριο το λογισμικό(software), και το υλικό (hardware) που θα χρησιμοποιηθεί από το σύστημα πρέπει να αναφερθούν. Για την αξιολόγηση του σεναρίου, ιδιαίτερη έμφαση δίνεται στην αποδοτικότητα.

4.4 Μέθοδος σχεδίασης με επίκεντρο το όραμα - (Vision-Proposal Design Method)

Η συγκεκριμένη μέθοδος σχεδίασης επιτρέπει την ανάπτυξη καινοτόμων προϊόντων λογισμικού καθώς και τη βελτίωση υπάρχοντων προϊόντων. Οι Koji Yanagida και Yoshihiro Ueda οι οποίοι τη δημιουργήσαν υπογραμμίζουν ότι η χρησιμότητά της έγκειται στο γεγονός ότι η σχεδίαση του προϊόντος στηρίζεται όχι στα τεχνικά χαρακτηριστικά του συστήματος αλλά στην αξία που πρόκειται να προσδώσει το σύστημα στους χρήστες [52].Η συγκεκριμένη μέθοδος χρησιμοποιεί σενάρια για το σχεδιασμό του συστήματος. Περιέχει τρεις διαφορετικές βαθμίδες σεναρίων: το σενάριο αξίας (value scenario), το σενάριο δραστηριότητας (activity scenario) και το σενάριο αλληλεπίδρασης (interaction scenario) .



Εικόνα 4-2 Βασικό μοντέλο Vision-proposal Design Method [52]

4.4.1 Ιδιότητες σεναρίων κατά τη διαδικασία ανάπτυξης

Όταν δημιουργηθεί το γενικό περίγραμμα και τα χαρακτηριστικά των υπηρεσιών, των συστημάτων είναι σημαντικό να αποσαφηνιστεί πλήρως η αξία που μπορεί να ληφθεί αξιοποιώντας τα. Εξηγώντας τις πρακτικές διαδικασίες και σχέσεις, θα δημιουργηθεί φυσικά μια ιστορία. Συνεπώς το σενάριο χρησιμοποιείται κατά τη δημιουργία ιδεών σε ένα πρώιμο στάδιο της ανάπτυξης. Έτσι, το σενάριο αρχικά χρησιμοποιείται εντός των ατόμων της επιχείρησης που εμπλέκονται στην ανάπτυξη. Στη συνέχεια, όταν οι υπηρεσίες και τα συστήματα που παρέχονται στον χρήστη αποκτήσουν μία πρώιμη μορφή, το σενάριο χρησιμοποιείται για να εξηγήσει στους χρήστες πώς να τα χρησιμοποιούν.

Όσον αφορά την επικοινωνία των τεχνικών χαρακτηριστικών του συστήματος χρησιμοποιείται η τεχνική του ‘’πότε, πού, ποιος, τι, γιατί και πώς’’ (when, where, who,what, why and how) . Όσον αφορά ενά σενάριο που αναφέρεται σε προϊόν λογισμικού το ‘’πού’’ και το ‘’πότε’’ αναφέρονται στην κατάσταση, το ‘’ποιος’’ στο χρήστη, το ‘’τι’’ στις υπηρεσίες του συστήματος, το ‘’γιατί’’ στους στόχους και στις προσδοκίες των χρηστων και το ‘’πώς’’ παρουσιάζει τον τρόπο με τον οποίο χρησιμοποιείται το σύστημα. Απαντώντας στο ‘’ποιος’’ δημιουργείται η persona και έπειτα σχεδιάζονται διάφορα σενάρια με βάση αυτή την persona. Το σενάριο μπορεί να παρουσιαστεί στο χρήστη και κατά τη αρχή της σχεδίασης του προϊόντος προκειμένου η ομάδα έργου να λάβει χρήσιμη ανατροφοδότηση.

4.4.2 Δομή σεναρίων

Τα σενάρια πριν δομηθούν κατηγοροιοποιούνται σύμφωνα με τα κριτήρια ιεραρχίας της μεθόδου. Τα κριτήρια ιεραρχίας είναι: 1) η αξία, 2) η δραστηριότητα και 3) η αλληλεπίδραση. Αξίζει να σημειωθεί ότι δεν υπάρχει προτεραιότητα στα συγκεκριμένα κριτήρια, η σειρά τους διαφοροποιείται ανάλογα με το σύστημα που καλούμαστε να αναπτύξουμε.

Το σενάριο αξίας αντιστοιχεί στην περιγραφή που δίνεται για την ιεραρχία αξίας. Το συγκεκριμένο σενάριο περιγράφει μια εγγενή αξία για τους χρήστες και την αξία της επιχείρησης ως παρόχου. Εδώ απεικονίζονται οι στόχοι της επιχείρησης και οι προσδοκίες των χρηστών. Σε αυτήν την ιεραρχία, οι κατηγορίες χρηστών σχηματίζονται με σχετικά υψηλό επίπεδο αφαίρεσης. Όσον αφορά την αξιολόγηση του συγκεκριμένου σεναρίου λαμβάνεται υπόψην μόνο οι στόχοι και η άποψη της επιχείρησης. Επίσης όσον αφορά του χρήστες η αξιολόγηση της ικανοποίησής τους αξιολογείται μέσω της καινοτομίας του συστήματος. Οι βασικές σκοπιές από τις οποίες αξιολογείται ένα σενάριο αξίας είναι οι εξής:

- Εάν ο στόχος του έργου έχει επιτευχθεί
- Εάν οι αρχικές απαιτήσεις των χρηστών έχουν ικανοποιηθεί
- Έαν το προϊόν λογισμικού ανταποκρίνεται στο μοντέλο παροχής του προϊόντος όπου έχει σχεδιάσει η επιχείρηση

Το σενάριο δραστηριότητας ανταποκρίνεται στην περιγραφή που δίνεται για την ιεραρχία των δραστηριοτήτων. Αυτό το σενάριο εμπεριέχει σκηνές όπου το σύστημα χρησιμοποιείται,, οπτικοποιεί τη δραστηριότητα του χρήστη και απεικονίζει τη ροή δραστηριότητας και τα συναισθήματα του χρήστη ειδικά. Στο συγκεκριμένο σενάριο περιγράφεται και η άποψη της επιχείρησης με κύρια αντικείμενα της περιγραφής της να είναι η αποτελεσματικότητα που παρέχει το σύστημα στον χρήστη. Οι βασικές σκοπιές από τις οποίες αξιολογείται ένα σενάριο δραστηριότητας είναι οι εξής:

- Εάν ο στόχος του έργου έχει επιτευχθεί
- Εάν η σκηνή που περιγράφεται στο σενάριο αξίας είναι υλοποιήσημη
- Εάν οι personas ανταποκρίνονται αποτελεσματικά στους χρήστες

Το σενάριο αλληλεπίδρασης περιγράφει τις πρακτικές λειτουργίες του συστήματος σε σχέση με το χρήστη. Περιγράφει τους τρόπους με τους οποίους το σύστημα μέσω της λειτουργίας τους ικανοποιεί τις αξίες και τις ανάγκες των χρηστών. Βασικό κρίτηριο αξιολόγησης του συγκεκριμένου σεναρίου είναι η αποδοτικότητα του συστήματος. Στον παρακάτω πίνακα συνοψίζονται τα βασικά χαρακτηριστικά κάθε σεναρίου. Οι βασικές σκοπίες από τις οποίες αξιολογείται ένα σενάριο αλληλεπίδρασης είναι οι εξής:

- Εάν ο στόχος του έργου έχει επιτευχθεί
- Εάν οι εργασίες που περιγράφονται στο σενάριο δραστηριότητας είναι ρεαλιστικές
- Εάν οι personas ανταποκρίνονται πραγματικά στους χρήστες που αλληλεπιδρούν με το σύστημα

Στο σενάριο αλληλεπίδρασης όλα τα κριτήρια αξιολόγησης λαμβάνονται υπόψην. Σημαντικότερα όμως είναι τα κριτηρία που αφορούν την αποδοτικότητα του συστήματος από την πλευρά του χρήστη καθώς και κατά πόσο η ανάπτυξη του προϊόντος είναι τελικά εφικτή από την πλευρά της επιχειρήσης, κριτήρια τα οποία δεν περιλαμβάνονται σε κάποιο άλλο σενάριο. Συμπερασματικά για το συγκεκριμένο μοντέλο:

Είδος Σεναρίου	Στόχοι Επιχείρησης Προσδοκίες Χρήστη	Συγκεκριμένη ''εικόνα'' για τη δραστηριότητα των χρηστών	Τεχνικά Χαρακτηριστικά Αντικειμένου	Τεχνολογία	Βασικό στοιχείο Αξιολόγησης
Σενάριο Αξίας	Καλά	Ελάχιστα	Ελάχιστα	Ελάχιστα	Σχεδίαση με επίκεντρο το χρήστη Καινοτομία
Σενάρια Δραστηριότητας	Μέτρια	Καλά	Ελάχιστα	Ελάχιστα	Σχεδίαση με επίκεντρο το χρήστη

					Χρησιμότητα
Σενάρια Αλληλεπίδρασης	Μέτρια	Μέτρια	Καλά	Ελάχιστα	Σχεδίαση με επίκεντρο το χρήστη Αποτελεσματικότητα
Πρόταση προς υλοποίηση	Μέτρια	Μέτρια	Μέτρια	Καλά	Τεχνολογία που χρησιμοποιείται Σκοπιμότητα

Πίνακας 17 Συμπερασματικός Πίνακας Σεναριών - Vision Proposal Design Method [52]

4.5 Μέθοδοι αυτοματοποίησης εξαγωγής απαιτήσεων

Τα τελευταία χρόνια παρατηρείται στην επιστημονική κοινότητα μία τάση για την ανάπτυξη αυτόματων ροών εργασίας για την εκμαίευση απαιτήσεων από τους χρήστες. Οι περισσότερες αυτοματοποιημένες μέθοδοι όπου έχουν αναπτυχθεί στηρίζονται στο crowdsourcing ή αλλίως πληθοπορισμό. Οι Estelles-Arolas και Gonzalez Landron-de-Guevara έχουν ορίσει τον πληθοπορισμό ως εξής [53]: ‘’ *Ο πληθοπορισμός είναι μία μορφή συλλογικής διαδικτυακής δραστηριότητας στην οποία ένα άτομο, ένα ίδρυμα, ένας μη κερδοσκοπικός οργανισμός ή μία εταιρεία προτείνει σε μία ομάδα ατόμων με ποίκιλλες γνώσεις, ετερογένεια και αριθμό, μέσω μίας ανοιχτής πρόσκλησης, να αναλάβουν εθελοντικά μια εργασία. Η ανάληψη της εργασίας, η οποία ποικίλλει σε πολυπλοκότητα και στο βαθμό στον οποίο είναι χωρισμένη και στην οποία το πλήθος πρέπει να συμμετάσχει με προσωπική εργασία,*

χρήματα, γνώση, εμπειρία, περιλαμβάνει πάντοτε αμοιβαίο όφελος και για τις δύο πλευρές. Οι χρήστες λαμβάνουν την ικανοποίηση κάποιας ανάγκης τους, είτε αυτή είναι οικονομική, είτε κοινωνική αναγνώριση, προσωπική ικανοποίηση, ανάπτυξη ατομικών ικανοτήτων σε κάποιο τομέα, ενώ ο εκκινητής της πρωτοβουλίας (πληθοποριστής) αποκτά και χρησιμοποιεί προς όφελός του, αυτά που έχει συνεισφέρει ο χρήστης στο εγχείρημα, τα οποία εξαρτώνται από τη δραστηριότητα που έχει αναλάβει ο χρήστης.''

Ο βασικός στόχος των συγκεκριμένων μεθόδων είναι η αύξηση της συμμετοχής των χρηστών στη διαδικασία συλλογής των απαιτήσεων. Η Kathryn T.Stolee χρησιμοποίησε το Amazon Mechanical Turk[27] το οποίο είναι μια διαδικτυακή αγορά (Crowdsourcing) η οποία δίνει τη δυνατότητα σε ιδιώτες και σε επιχειρήσεις να αιτηθούν και να συντονίσουν τη χρήση της ανθρώπινης ευφυΐας για την εκτέλεση εργασιών οι οποίες δεν είναι σήμερα δυνατόν να εκτελεστούν από τους υπολογιστές. Έπειτα από την έρευνά της αποφάνθηκε ότι στη συγκεκριμένη μέθοδο εκμαίευσης απαιτήσεων ήταν δύσκολη η κατάλληλη επιλογή αντικείμενων προσώπων καθώς και η αλληλεπίδραση του ερευνητή με τα άτομα τα οποία συμμετέχουν στη διαδικασία.

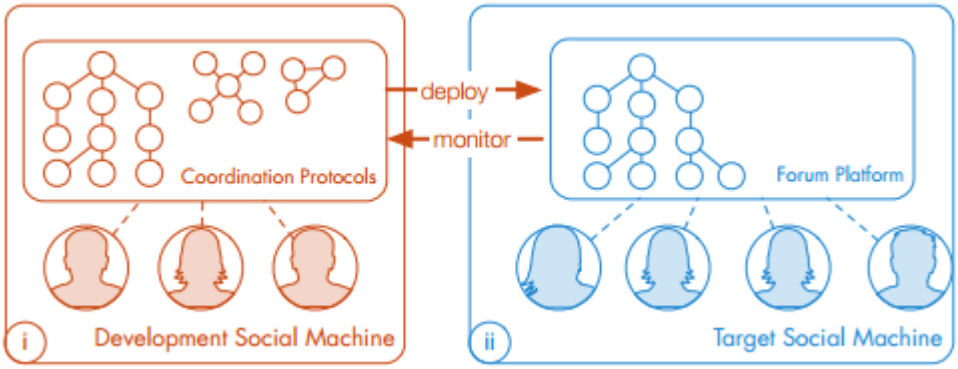
Ακόμη, η Soo Ling Lim το 2010 δημιούργησε το StakeSource[28] το οποίο είναι μία πλατφόρμα η οποία στοχεύει στην ανάλυση των εμπλεκόμενων μελών και στην εκμαίευση των απαιτήσεών τους μέσω crowdsourcing. Το StakeSource επίσης στοχεύει στη μείωση του φόρτου εργασίας των αναλυτών. Οι αναλυτές αρχικά προσδιορίζουν τα χαρακτηριστικά κάθε εμπλεκόμενου μέλους και στη συνέχεια τους ζητούν να συμπληρώσουν ειδικές φόρμες τις οποίες τις ομαδοποιούν σε συγκεκριμένα profiles και εν τέλει δημιουργούν κάποιου είδους personas.

Τέλος, ο Murray-Rust D. [54] το 2014 δημιούργησε ένα εννοιολογικό μοντέλο για το συνδυασμό διαδικαστικών μοντέλων ανάπτυξης λογισμικού με crowd-sourced ομάδες προκειμένου να δημιουργήσουν αντικείμενα λογισμικού (software artefacts) με τη βοήθεια δυναμικών κοινοτήτων μέσω του SCU (Social Compute Unit) το οποίο

[27] https://www.mturk.com/mturk/welcome

[28] http://www.stakesource.co.uk/

συνδυάζει χρήστες του λογισμικού και προγραμματιστές με τους αναλυτές οι οποίοι ελέγχουν και βάζουν σε προτεραιότητα τις απαιτήσεις των χρηστών.



Εικόνα 4-3 Συνεργατικό μοντέλο για την ανάπτυξη λογισμικού με χρήση κοινονικών μηχανών [54]

4.6 Συμπεράσματα επί των εργασιών σχετικών με την εκμαίευση απαιτήσεων σε καινοτόμα προϊόντα λογισμικού

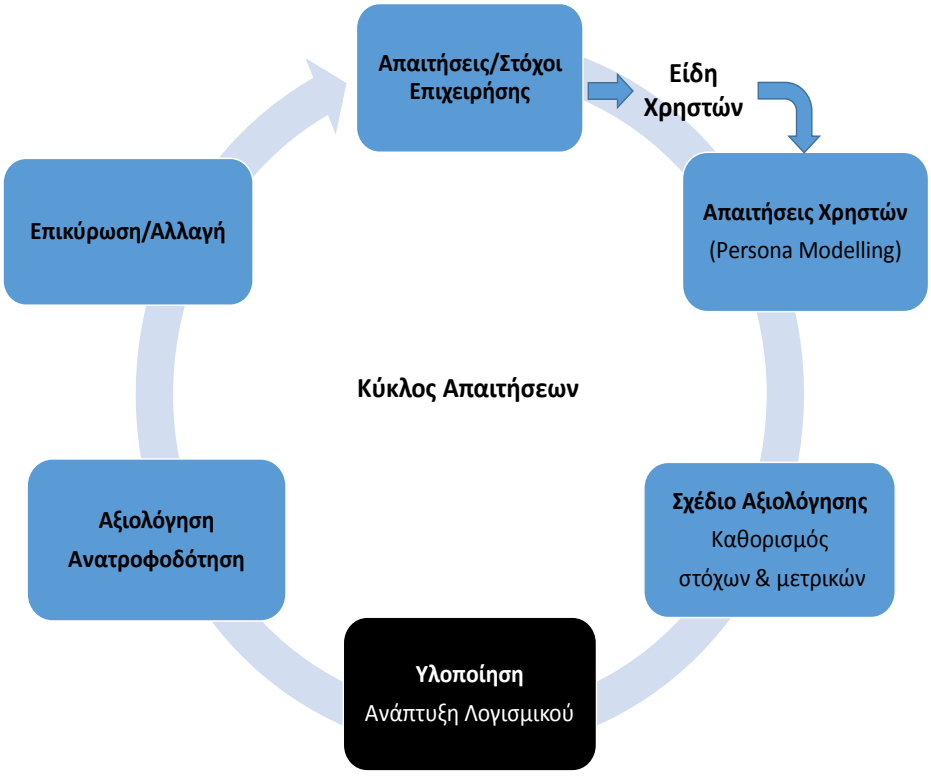
Η μελέτη των προαναφερθέντων εργασιών οδήγησε στην εξαγωγή σημαντικών συμπερασμάτων σχετικά με τις μεθόδους ανάπτυξης προϊόντων λογισμικού και ιδιαιτέρως με τη διαδικασία εκμαίευσης και συλλογής των απαιτήσεων από τους χρήστες.Παρατηρείται πως υπάρχει ένα σημαντικό κενό μεταξύ της διαχείρησης των επιχειρηματικών αναγκών και των τεχνικών αναγκών. Ενώ οι προαναφερθείσες εργασίες προσπαθούν να προωθήσουν μεθόδους οι οποίες να προάγουν τη σχεδίαση προϊόντος με βάση το χρήστη, δεν συμπεριλαμβάνουν έναν ενιαίο τρόπο διαχείρησης μεταξύ των επιχειρηματικών και τεχνικών απαιτήσεων με αποτελέσματα να χάνονται πολύτιμες πληροφορίες όχι μόνο για τη βελτίωση των επιχειρηματικών διαδικασιών αλλά πολύ περισσότερο για την αποτελεσματική σχεδίαση καινοτόμων προϊόντων με επίκεντρο το χρήστη.

Η καινοτομία σε ένα μεγάλο βαθμό στηρίζεται στους νέους ανθρώπους οι οποίοι δεν έχουν την απαραίτητη εμπειρία και εκπαίδευση για να χρησιμοποιούν μεθόδους οι οποίες να μην είναι αυστηρά προκαθορισμένες. Οι εργασίες που μελετήθηκαν ενώ

αναγνωρίζουν το συγκεκριμένο ζήτημα δεν παρέχουν μία βηματική διαδικασία η οποία να διευκολύνει την ομάδα έργου ανεξαρτήτως ηλικίας, μεγέθους της εταιρείας και αγοράς στην οποία απευθύνεται να δημιουργήσουν καινοτόμα προϊόντα λογισμικού.

5 *Ανάπτυξη μεθοδολογίας συλλογής απαιτήσεων για ανάπτυξη καινοτόμων προϊόντων λογισμικού*

Μελετώντας την υπάρχουσα βιβλιογραφία και τις διάφορες τεχνικές και μεθοδολογίες εκμαίευσης απαιτήσεων για την ανάπτυξη καινοτόμων προϊόντων λογισμικού τόσο σε επίπεδο χρηστών όσο και σε επίπεδο επιχειρήσης και αναγνωρίζοντας σημαντικά κενά στη σωστή αποσαφήνιση και καταγραφή των απαιτήσεων προτείνεται η παρακάτω μεθοδολογία συλλογής απαιτήσεων και ελπίζουμε πως θα βοηθήσει τις καινοτόμες ομάδες να δημιουργήσουν προϊόντα λογισμικού τα οποία να αναπτύσσονται με επίκεντρο τους χρήστες και την ικανοποίηση των αναγκών τους. Το εννοιολογικό πλαίσιο της μεθόδου αποτελείται από 6 στάδια τα οποία επαναλαμβάνονται μέχρι να αποσαφηνιστούν πλήρως οι απαιτήσεις των χρηστών αλλά και της επιχείρησης και γραφικά αναπαριστάται σαν ένας κύκλος προκειμένου να απεικονιστεί η επαναληπτική διαδικασία.



Εικόνα 5-1 Προτεινόμενα στάδια εκμαίευσης και εξαγωγής απαιτήσεων

- Συλλογή πληροφορίων από την επιχείρηση
- Συλλογή πληροφοριών από τους χρήστες με στόχο την πληρή αποσαφήνιση των στόχων, των αξιών και των απαιτήσεών τους
- Ανάλυση των πληροφοριών που συλλέχθηκαν και κατάστρωση σχεδίου αξιολόγησης αυτών
- Υλοποίηση – Ανάπτυξη τμήματος/τμημάτων λογισμικού με στόχο την ικανοποίηση των αναγκών/επιθυμιών που συλλέχθηκαν στα προηγούμενα βήματα
- Αξιολόγηση μέσω μετρικών του λογισμικού που αναπτύχθηκε και συζήτηση για αναγνώριση λαθών ή βέλτιστων πρακτικών.
- Ανατροφοδότηση για την αποτελεσματικότητα του τμήματος λογισμικού που αναπτύχθηκε ως προς τους στόχους/μετρικές όπου έθεσε η επιχείρηση για την αποσαφήνιση των απαιτήσεων.
- Επικύρωση των απαιτήσεων ή επανεκκίνηση του κύκλου απαιτήσεων για διόρθωση λαθών.

5.1 Διαδικασία συλλογής και ανάλυσης πληροφοριών από την επιχείρηση

Η διαδικασία συλλογής πληροφορίων από την επιχειρήση είναι ιδιαιτέρως σημαντική προκειμένου να αποσαφηνιστούν οι στόχοι της επιχειρήσης όχι μόνο όσον αφορά το προϊόν, αλλά και την αξιολόγησή του, την κοστολόγησή του, τα κανάλια διανομής κ.λ.π.

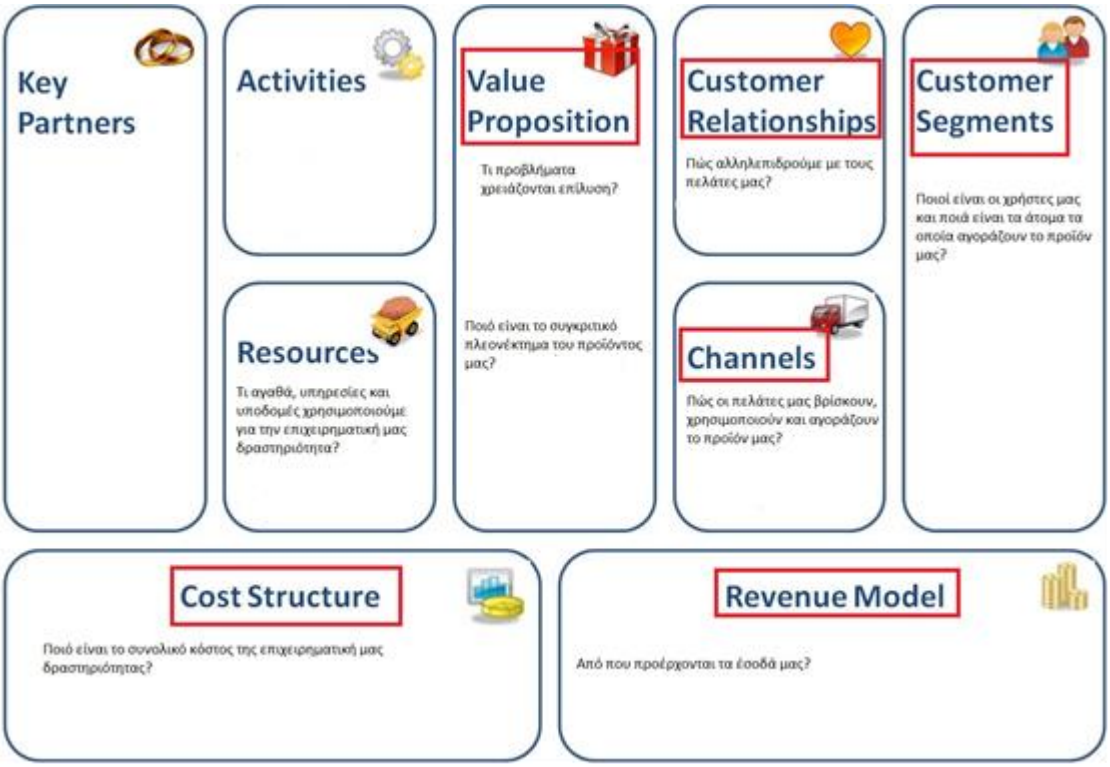
Όπως αναφέραμε στην ενότητα 3.3.3 υπάρχουν ποικίλες μέθοδοι διοίκησης καινοτόμων ομάδων κάθε μία από τις οποίες επικεντρώνεται σε διαφορετικά στοιχεία της επιχειρησιακής διαδικασίας. Η μέθοδός μας επικεντρώνεται στο συνδυασμό των εργαλείων ‘’Business Model Canvas’’ και ‘’Lean Analytics’’ προκειμένου η ομάδα να συγκεντρώσει όσο το δυνατόν περισσότερα στοιχεία σχετικά με τους στόχους της όσον αφορά τη διαδικασία ανάπτυξης του καινοτόμου προϊοντος λογισμικού.

Αρχικά, η επιχείρηση καλείται να χρησιμοποιήσει το εργαλείο ‘’Business Model Canvas’’, μιας και το εν λόγω εργαλείο οργανώνει έτσι την επιχειρηματική στρατηγική ώστε να έχουμε τη δυνατότητα να αντλήσουμε σημαντικές πληροφορίες που αφορούν τους στόχους αλλά και τις απαιτήσεις της επιχείρησης από την ανάπτυξη ενός προϊόντος λογισμικού το οποίο αποτελεί άλλη μία επιχειρηματική δραστηριότητα.Συνεπώς τα πεδία τα οποία χρειαζόμαστε από το ‘’Business Model Canvas’’ για την ανάλυσή μας με σειρά προτεραιότητας είναι τα εξής:

1. **Πρόταση Αξίας (Value Proposition)** : Από το συγκεκριμένο πεδίο αντλούμε τα πιο σημαντικά στοιχεία που αφορούν τους στόχους και τις επιδιώξεις της επιχειρήσης μιας και σε αυτό το πεδίο αναγράφονται τα συγκριτικά πλεονεκτήματα του προϊόντος έναντι του ανταγωνισμού καθώς και τα προβλήματα τα οποία το προϊόν στοχεύει να επιλύσει.
2. **Είδη Πελατών (Customer Segments) :** Από το συγκεκριμένο πεδίο αντλούμε όλες τις πληροφορίες οι οποίες αφορούν τα διαφορετικά είδη πελατών που έχει η επιχείρηση. Στη συγκεκριμένη περίπτωση όπου η προτεινόμενη μεθοδολογία αναφέρεται σε προϊόντα λογισμικού, αναφερόμαστε στα

διαφορετικά είδη των χρηστών του λογισμικού τα οποία θα χρησιμοποιηθούν στη συνέχεια για τη δημιουργία personas προκείμενου η ανάπτυξη του προϊόντος να γίνει με επίκεντρο την ικανοποίηση των αναγκών των χρηστών/πελατών της.

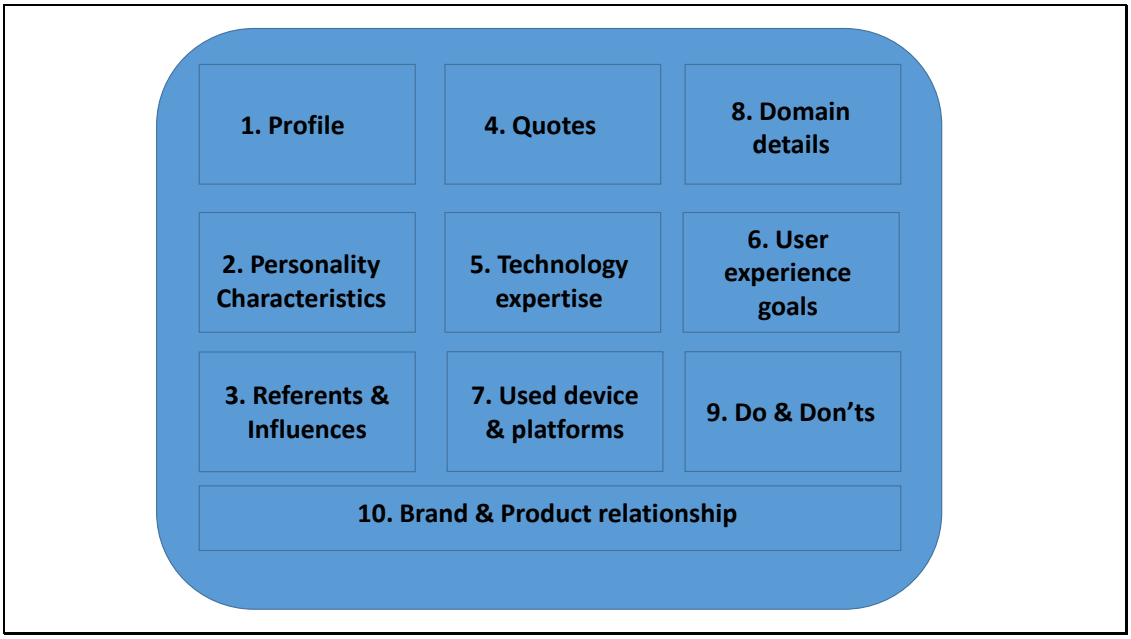
3. **Σχέση με τους πελάτες (Customer Relationships):** Στο παρόν πεδίο συμπληρώνουμε τους τρόπους με τους οποίους αλληλεπιδρά η επιχείρηση με τους πελάτες της όπως τους τρόπους με τους οποίους παρέχει το προϊόν της κ.λ.π.
4. **Κανάλια διανομής (Channels):** Στο παρόν πεδίο συμπληρώνουμε τους τρόπους με τους οποίους οι πελάτες βρίσκουν, χρησιμοποιούν και αγοράζουν το προϊόν της επιχειρήσης.
5. **Μοντέλο Εσόδων (Revenue Model):** Στο παρόν πεδίο συμπληρώνουμε τους τρόπους με τους οποίους η επιχείρηση λαμβάνει έσοδα από την παροχή του προϊόντος της.
6. **Δομή Κοστολόγησης (Cost Structure):** Στο παρόν πεδίο συμπληρώνουμε τα κόστη/έξοδα τα οποία απαιτούνται για την επιχειρηματική δραστηριότητα.



Εικόνα 5-2 Πεδία ενδιαφέροντος από το Business Model Canvas

5.2 Διαδικασία συλλογής και ανάλυσης πληροφοριών από τους χρήστες

Όσον αφορά τους χρήστες, αναλύουμε τα διαφορετικά είδη αυτών από το ‘’Business Model Canvas’’ και δημιουργούμε personas προκειμένου να τα συμπεριλάβουμε στην ανάλυσή μας για το λογισμικού που πρόκειται να υλοποιήσουμε. Η διαδικασία της δημιουργίας κατάλληλων personas προκειμένου να είναι πλήρως αντιπροσωπευτικές των πελατών βασίζεται στο πρότυπο persona που αναφέρεται στην ενότητα *3.2.3.3*. Στο παρόν σημείο αναφέρεται αναλυτικά η διαδικασία δημιουργίας αντιπροσωπευτικής persona προκειμένου να διευκολύνουμε τη δημιουργία της από την ομάδα.



Εικόνα 5-3 Persona Canvas

Η παραπάνω εικόνα απεικονίζει τις πληροφορίες που πρέπει να περιλαμβάνει η *persona* όπως αναφέραμε στην ενότητα *3.2.3.3*. Με βάση τη συγκεκριμένη τεχνική προτείνεται η παρακάτω διαδικασία συμπλήρωσης των αριθμημένων σημείων, κάθε ένα από τα οποία περιέχει διαφορετικές αλλά εξίσου σημαντικές πληροφορίες για το χρήστη.

Η διαδικασία είναι η εξής:

1. **Συμπλήρωση των βασικών στοιχείων που αφορούν την persona.** Στο παρόν σημείο καλούμαστε να συμπληρώσουμε τις παρακάτω πληροφορίες:
    - Δημογραφικά χαρακτηριστικά: Ηλικία, φύλο, οικογενειακή κατάσταση, εισόδημα, εκπαίδευση, εργασία.
    - Γεωγραφικά στοιχεία: Η persona που κατοικεί και εργάζεται? Πώς είναι εκεί? Κατοικεί σε μεγαλούπολη ή χωριό?
    - Ψυχογραφικά στοιχεία: Κοινωνική τάξη, Δραστηριότητες, Απόψεις, Στοιχεία της προσωπικότητας
    - Είδος χρήστη: π.χ Αρχάριος, Μέτρια εξικοιωμένος με τις τεχνολογίες, Έμπειρος, κ.λ.π.
    - Σχέση πελάτη με το προϊόν: Συχνότητα χρήσης, Brand Loyalty, παρούσα κατάσταση χρήστη (επίδοξος πελάτης, πρώτη-φορά πελάτης, συχνός)
    - Όνομα persona

    Τις συγκεκριμένες πληροφορίες προτείνεται η ομάδα να τις γράφει σε μορφή ιστορίας συνοψίζοντας το προαναφερθέντα βασικά σημειά προκειμένου η ομάδα να αντιληθφεί τον τρόπο ζωής της persona και το υπόβαθρό της.

2. **Συμπλήρωση στοιχείων της προσωπικότητας της persona.**
    - Όπως προαναφέρθηκε στην ενότητα 3.2.3.3 η εκμαίευση πληροφοριών για την προσωπικότητα της persona συνίσταται να γίνει μέσω του 5-factor model και του MBTI.
    - Η μοντελοποίηση των χρηστών θα είναι ελλειπής δίχως η ομάδα να γνωρίζει τα βασικά χαρακτηρτιστικά της προσωπικότητας της persona, δεδομένου ότι οι περισσότερες ανθρώπινες αποφάσεις στηρίζονται στα χαρακτηριστικά της προσωπικότητάς μας.

    Η συγκεκριμένη καταγραφή βοηθάει την ομάδα στη δημιουργία περισσότερο ρεαλιστικών ιστοριών χρήσης και ενισχύει την κατανόηση των απαιτήσεων του χρήστη.

3. **Συμπλήρωση στοιχείων σχετικών με τις επιρροές της persona όσον αφορά τους ανθρώπους, την τεχνολογία και τα λογισμικά που χρησιμοποιεί.**

- Με τις συγκεκριμένες πληροφορίες μπορεί εύκολα η ομάδα να κατανοήσει τις επιρροές και την εξοικείωση με τις τεχνολογίες της persona.

Τις συγκεκριμένες πληροφορίες προτείνεται να τις αναπαριστά η ομάδα μέσω εικόνων προκειμένου να έχουν οπτική επαφή με τον ‘’υποθετικό πελάτη’’ και να είναι και πιο εύκολες στην ανάγνωση και στην κατανόηση.

4. **Συμπλήρωση χαρακτηριστικών παραδειγμάτων όσον αφορά φράσεις ή προτιμήσεις της persona.**
   - Οι συγκεκριμένες πληροφορίες αντικατοπτρίζουν τα σχόλια της persona και πρέπει να απαντούν στην ερώτηση: ‘’ Τι είναι αυτό που αναμένει, θέλει ή φοβάται η persona?’’.
   - Μπορούν να εκμαιευτούν σημαντικές πληροφορίες όσον αφορά όχι μόνο το προϊόν λογισμικού αλλά και το brand.
   - Τις συγκεκριμένες πληροφορίες προτείνεται η ομάδα να τις συλλέγει μέσω συνεντεύξεων. Αφού ολοκληρωθούν οι συνεντεύξεις και συλλεχθούν πληροφορίες η ομάδα έχει την ευχέρεια να ομαδοποιήσει personas με παρεμφερείς απαντήσεις και χαρακτηριστικά προσωπικότητας.
5. **Συμπλήρωση στοιχείων σχετικών με την εξοικείωση της persona με την τεχνολογία που χρησιμοποιείται από το προϊόν λογισμικού για την αλληλεπίδραση με την persona.**
   - Προτείνεται η ανάλυση εξοικείωσης να γίνεται σε τέσσερις διαφορετικές βαθμίδες - άπειροι (inexpert), βασικών γνώσεων (medium), προχωρημένοι (advanced) και έμπειροι (experts) προκειμένου να γίνεται αποτελεσματικότερη ταξινόμηση των κατηγοριών των personas.
6. **Συμπλήρωση στοιχείων σχετικών με τις επιδιώξεις των personas σχετικά με την εμπειρία χρήσης.**
   - Οι συγκεκριμένες πληροφορίες θα βοηθήσουν την ομάδα να δώσει προτεραιότητα στις απαιτήσεις των χρηστών όσων αφορά την αλληλεπίδραση.
   - Τα συγκεκριμένα στοιχεία είναι πολύ σημαντικά για τη λήψη αποφάσεων σχετικά με τη διαδικασία σχεδίασης του προϊόντος και αποτελούν

καθοριστικό παράγοντα για την προσθήκη ενός νέου χαρακτηριστικού (feature) στο προϊόν.

- Προτείνεται η ομάδα να εκμαιεύσει τις συγκεκριμένες πληροφορίες μέσω ερωτηματολογίων και ερευνών αγοράς προκειμένου η ομάδα να θέσει τις απαιτήσεις του χρήστη όσον αφορά την εμπειρία χρήσης από την πλευρά των personas.

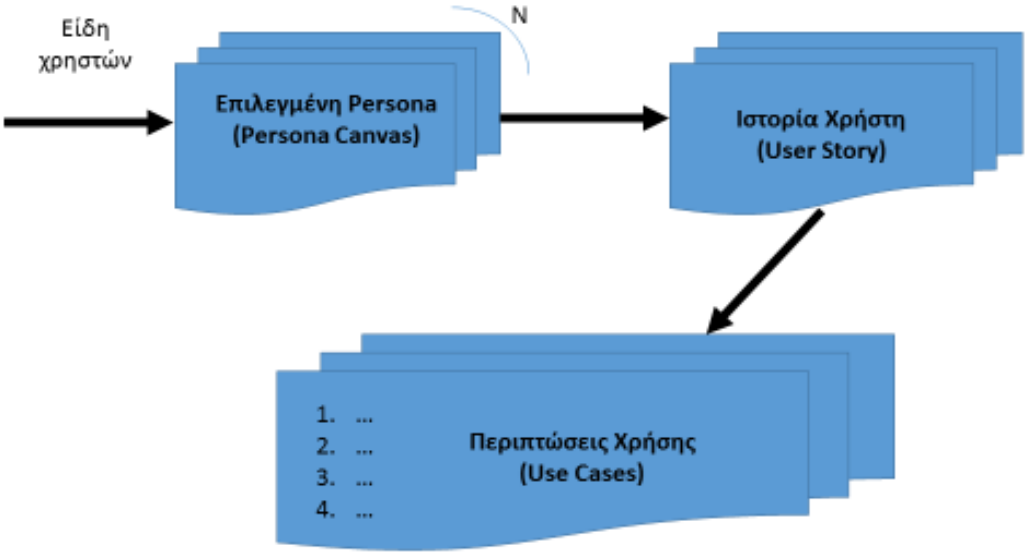
7. **Συμπλήρωση στοιχείων σχετικών με τις συσκευές και τις πλατφόρμες λογισμικού που γνωρίζει να χρησιμοποιεί η persona.**
   - Στο παρόν σημείο προτείνεται η ομάδα να αναφέρει την εμπειρία ή την εξοικείωση της persona με τις συσκευές (π.χ smartphone, laptop, tablet) και τις πλατφόρμες λογισμικού (π.χ Windows, Linux).
8. **Συμπλήρωση στοιχείων σχετικών με τις συνήθειες χρήσης της persona σχετικά με τα προϊόντα λογισμικου και τις συσκευές που συνηθίζει να χρησιμοποιεί.**
   - Οι συγκεκριμένες πληροφορίες επιτρέπουν στην ομάδα να κατανοήσει τα ενδιαφέροντα και τις συνήθειες της persona ως καταναλωτή προϊόντων λογισμικού.
   - Προτείνεται η ομάδα να παρουσιάζει τις συγκεκριμένες πληροφορίες σε μορφή pie charts προκειμένου μέσω της οπτικοποίησης οι πληροφορίες αυτές να γίνουν πλήρως κατανοητές από τους σχεδιαστές και τους προγραμματιστές της ομάδας.
9. **Συμπλήρωση στοιχείων σχετικών με τις βασικές οδηγίες αλληλεπίδρασης της persona με το προϊόν λογισμικού.**
   - Τα συγκεκριμένα στοιχεία παρέχουν πληροφορίες στην ομάδα σχετικά με το τί επιθυμεί η persona από το προϊόν και τί την ενοχλεί στο προϊόν.
10. **Συμπλήρωση στοιχείων σχετικών με τη σχέση της persona με το brand ή και το προϊόν λογισμικού.**
    - Στο παρόν σημείο η ομάδα προσπαθεί να αξιολογήσει την επιρροή του brand και του προϊόντος για την κάθε persona.
    - Η ομάδα εκμαίευει σημαντικές πληροφορίες για τη στρατηγική marketing.

5.3 Ροή διαδικασίας εργασιών για τη συλλογή και ανάλυση πληροφοριών από τους χρήστες

Συνοψίζοντας, η συλλογή στοιχείων με τις προτεινόμενες μεθόδους εκμαίευσης όπως αυτές περιγράφηκαν στην ενότητα 5.2 είναι ιδιαιτέρως σημαντική για την ομάδα προκειμένου στη συνέχεια να δημιουργήσει τις ιστορίες χρήσης (user stories) οι οποίες θα χρησιμοποιηθούν ως input στο μοντέλο μας προκειμένου έπειτα από ανάλυση η ομάδα να καταφέρει να προσδιορίσει πλήρως τις απαιτήσεις των χρηστών τόσο σε λειτουργικό αλλά και μη-λειτουργικό, ποιοτικό επίπεδο.

Αρχικά μέσω του persona canvas η ομάδα έργου συντάσσει μία ή και περισσότερες ιστορίες χρήσης για κάθε μία persona.Αφού πραγματοποιηθεί η σύνταξη των ιστοριών χρήσης, εν συνεχέια πραγματοποιείται ανάλυση αυτών με στόχο την αποσαφήνιση όλων των διαφορετικών περιπτώσεων χρήσης (use cases) που αφορούν την persona. Αφού ολοκληρωθεί αυτή η διαδικάσια για όλες τις personas, προτείνεται η συγκριτική ανάλυση όλων των διαφορετικών περιπτώσεων χρήσης προκειμένου να εντοπιστούν ομοιότητες και να μπουν σε ιεραρχία ως προς την υλοποίησή τους.

Η διαδικασία παρουσιάζεται συνοπτικά στο επόμενο διάγραμμα ροής εργασιών.



Εικόνα 5-4 Ροή διαδικασίας συλλογής και ανάλυσης πληροφοριών από τους χρήστες

5.4 Καθορισμός σχεδίου αξιολόγησης

Έχοντας λοιπόν εντοπίσει στα προηγούμενα βήματα τις απαιτήσεις/ανάγκες των χρηστών καθώς και τους στόχους της επιχειρήσης στο παρόν σημείο πρέπει να καθοριστεί ένα σχέδιο αξιολόγησης προκειμένου η ομάδα έργου να αποσαφηνίσει εάν οι απαιτήσεις οι οποίες εντόπισε είναι αυτές οι οποίες πραγματικά έχουν σημασία και προτεραιότητα. Για το σκοπό αυτό το προτεινόμενο μοντέλο χρησιμοποιεί τη μέθοδο ‘’Lean Analytics’’ [26].

Όσον αφορά το εργαλείο ‘’Lean Analytics’’ [26], χρησιμοποιείται προκειμένου να διευκολυνθεί η εύρεση κατάλληλων μετρικών για το κάθε διαφορετικό είδος επιχειρηματικού μοντέλου (SaaS, E-commerce platform, κ.λ.π.) σε συνδυασμό με το στάδιο ανάπτυξης της επιχείρησης αλλά και του προϊόντος προκειμένου να μπορούμε να αξιολογήσουμε τη κάθε διαδικασία ανάπτυξης και το αντίκτυπο που έχει στους χρήστες. Συνοψίζοντας η ομάδα πρέπει να διαλέξει ποιο επιχειρηματικό μοντέλο ακολουθεί και κατά συνέπεια ποιες μετρικές είναι εκείνες στις οποίες

οφείλει να δώσει ιδιαίτερη προσοχή. Τα παραπάνω συνοψίζονται στον παρακάτω πίνακα.

	E-commerce	2-sided market	SaaS	Mobile app	User-generated content	Media
Empathy	Interviews; qualitative results; quantitative scoring; surveys					
Stickiness	Loyalty, conversion	Inventory, listings	Engagement, churn	Downloads, churn, virality	Content, spam	Traffic, visits, returns
Virality	CAC, shares, reactivation	SEM, sharing	Inherent virality, CAC	WoM, app ratings, CAC	Invites, sharing	Content virality, SEM
	(Money from transactions)		(Money from active users)		(Money from ad clicks)	
Revenue	Transaction, CLV	Transactions, commission	Upselling, CAC, CLV	CLV, ARPDAU	Ads, donations	CPE, affiliate %, eyeballs
Scale	Affiliates, white-label	Other verticals	API, magic #, mktplace	Spinoffs, publishers	Analytics, user data	Syndication, licenses

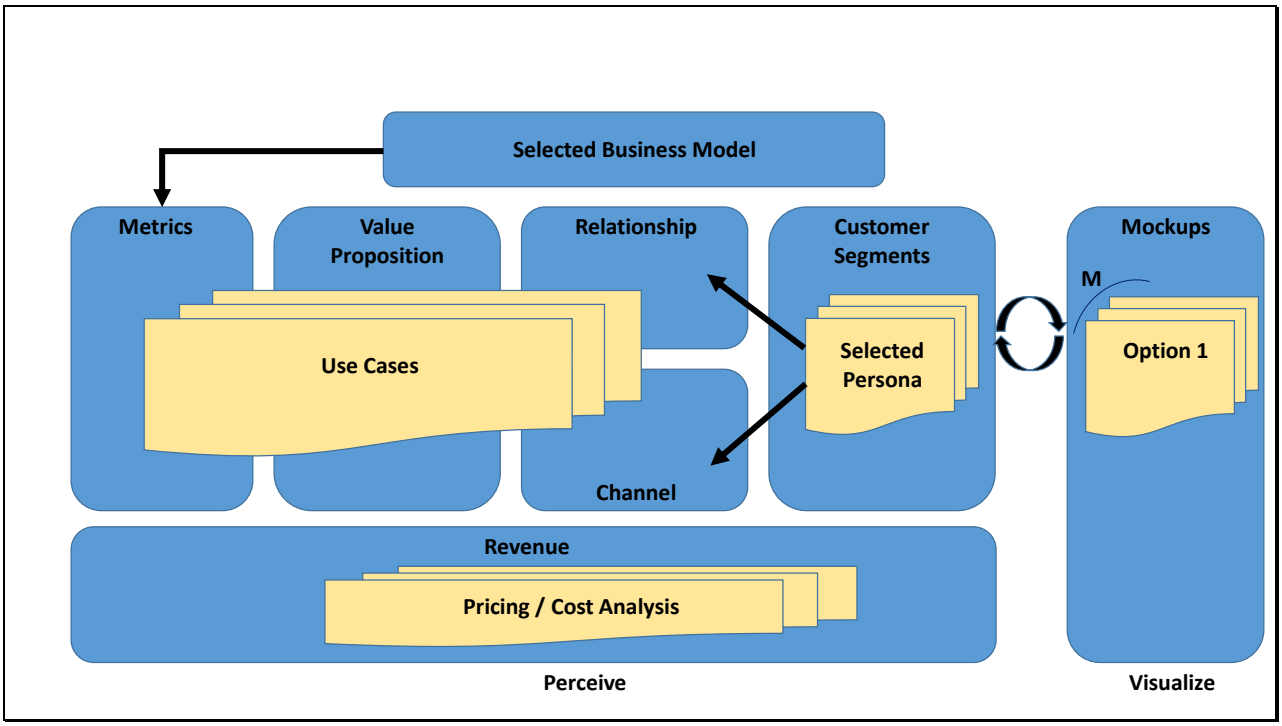
Πίνακας 18 Επιχειρηματικά μοντέλα και αντίστοιχες μετρικές για κάθε στάδιο ανάπτυξης- Lean Analytics

5.5 Επικύρωση και αξιολόγηση της διαδικασίας εκμαίευσης απαιτήσεων

Αφού η ομάδα έχει συλλέξει απαιτήσεις με τους προαναφερθέντες τρόπους προτείνεται η συμμετοχή όλων των εμπλεκόμενων μελών σε μία συζήτηση η οποία θα αφορά τα είδη των απαιτήσεων τα οποία συλλέχθηκαν, την αποτελεσματικότητα της ομαδοποίησης και ιεράρχησης αυτών καθώς και εάν εγκρίνεται η προσάυξηση συγκεκριμένων λειτουργιών που θα ικανοποιούν τις απαιτήσεις των χρηστών έπειτα από την ανάλυση. Εάν δεν εγκριθεί από τα εμπλεκόμενα μέλη η προσαύξηση αυτών, προτείνεται να επαναληφθεί η διαδικασία προκειμένου να διορθωθούν τυχόν λάθη και έπειτα από αξιολόγηση η ομάδα έργου να περάσει στο στάδιο υλοποίησης.

5.6 Μέθοδος εξαγωγής απαιτήσεων

Έχοντας συλλέξει όλες τις απαιτούμενες πληροφορίες που χρειάζεται με τις τεχνικές που προαναφέρθηκαν, η ομάδα καλείται να τις τοποθετήσει στο συγκεκριμένο μοντέλο/εργαλείο προκειμένου να ξεκινήσει η διαδικασία της εξαγωγής απαιτήσεων. Το προτεινόμενο μοντέλο παρουσιάζεται σχηματικά στην παρακάτω εικόνα.



Εικόνα 5-5 Προτεινόμενο Μοντέλο Εξαγωγής Απαιτήσεων

Αρχικά, έχοντας δημιουργήσει τα persona canvas η ομάδα είναι σε θέση να συμπληρώσει τα παρακάτω πεδία:

- Είδη Πελατών (Customer Segments)
- Σχέση Πελάτη – Επιχείρησης (Relationship)
- Κανάλη Διανομής Προϊόντος (Channel)

Αφότου έχει γίνει η ανάλυση των διάφορων πελατών και χρηστών καθώς και η ομαδοποίησή τους ανάλογα με τις απαιτήσεις, τις συνήθειες και τις προσδοκίες τους, η ομάδα καλείται να συμπληρώσει σύμφωνα με το Business Model Canvas της τα παρακάτω πεδία:

- Πρόταση Αξίας (Value Proposition)
- Έσοδα (Revenue)

Επίσης, λαμβάνοντας υπόψή το επιχειρηματικό της μοντέλο και τα ‘’Lean Analytics’’, η ομάδα προσδιορίζει τις μετρικές που χρειάζεται για την αξιολόγηση της διαδικασίας εξαγωγής απαιτήσεων.

Σε αυτό το σημείο η ομάδα καλείται να συνυπολογίσει τις πληροφορίες που συνέλεξε από τους πελάτες και χρήστες της καθώς και από την εσωτερική της αξιολόγηση. Λαμβάνοντας υπόψη κάθε φορά ένα persona canvas και το αντίστοιχο user story η ομάδα αντιλαμβάνεται το είδος πελάτη που καλείται να ικανοποιήσει, το κανάλι με το οποίο θα του παρέχει το προϊόν λογισμικού καθώς και τα μη-λειτουργικά ποιοτικά χαρακτηρτιστικά που χρειάζεται ο πελάτης.

Στη συνέχεια με βάση το επιχειρησιακό της μοντέλο και την αντίστοιχη μετρική καθώς και το business model canvas η ομάδα προσπαθεί να αξιολογήσει την συγκεκριμένη μετρική τόσο σε επίπεδο εμπειρίας χρήστη όσο και σε τεχνικό επίπεδο από άποψη πόρων. Αφού γίνει η συγκεκριμένη ανάλυση και κοστολογηθεί δημιουργούνται τόσα use cases όσα και τα γεγονότα τα οποία καλείται να υλοποιήσει η ομάδα προκειμένου να ικανοποιήσει τη συγκεκριμένη persona. Στο τέλος αυτού του κύκλου, προτείνεται η οπτικοποίηση του αποτελέσματος και η άμεση ανατροφοδότηση από την πλευρά των χρηστών αλλά και της επιχείρησης προκειμένου να αξιολογηθεί μέσω των μετρικών εάν ικανοποιούνται οι στόχοι και οι απαιτήσεις τόσο των χρηστών όσο και της επιχείρησης. Εάν κάτι τέτοιο δεν συμβεί τότε η διαδικασία επαναλαμβάνεται ξανά για την ίδια persona μέχρι τα αποτελέσματα των μετρικών να είναι ικανοποιητικά για την ομάδα.

Αυτή η διαδικασία επαναλαμβάνεται για όλες τις personas και λαμβάνοντας υπόψη τα αποτελέσματα των μετρικών οι απαιτήσεις μέσω των use cases μπαίνουν σε προτεραιότητα προκειμένου η ομάδα να γνωρίζει ποιες από αυτές πρέπει να υλοποιήσει προκειμένου να αυξήσει την ικανοποίηση των πελατών/χρηστών της.

5.7 Σύγκριση προτεινόμενου εργαλείου με αντίστοιχα εργαλεία στη βιβλιογραφία

Έχοντας εντοπίσει στην βιβλιογραφία μεθόδους οι οποίες στοχεύουν στην εξαγωγή απαιτήσεων κατά τη διαδικασία ανάπτυξης προϊόντων λογισμικού όπως αυτές

παρουσιάζονται στο Κεφάλαιο 4, ακολουθεί η σύγκριση του προτεινόμενου από μεριάς μας εργαλείου με τις συγκεκριμένες μεθόδους.

Η δομημένη μέθοδος σχεδίασης βασισμένη σε σενάρια χρήσης (Structured Scenario-Based Design Method) [51] αποτελεί τη βασική μέθοδο πάνω στην οποία στηρίχτηκαν και οι υπόλοιπες μέθοδοι που αναφέρονται στο κεφάλαιο 4, μιάς και όλες οι μέθοδοι που παρουσιάζονται στο εν λόγω κεφάλαιο αποτελούν βελτιώσεις της δομημένης μεθόδου σχεδίασης και προϊόν ακαδημαϊκής έρευνας της ίδιας ομάδας.

Ενώ ο βασικός στόχος όλων των μεθόδων και των εργαλείων είναι ο ίδιος, δηλαδή η σχεδίαση προϊόντων με επίκεντρο το χρήστη, οι συγκεκριμένες μέθοδοι παρουσιάζουν σημαντικές διαφορές από το προτεινόμενο εργαλείο εξαγωγής απαιτήσεων, μιας και το προτεινόμενο εργαλείο αποτελεί μία πρακτική εφαρμογή εξαγωγής απαιτήσεων σε αντίθεση με τις μεθόδους της βιβλιογραφίας οι οποίες αποτελούν εννοιολογικά πλαίσια θεωρητικού χαρακτήρα χωρίς να παρέχουν συγκεκριμένες πληροφορίες για την υλοποίηση του κάθε προτεινόμενου βήματός τους.Ο παρακάτω πίνακας παρουσιάζει τις διαφορές μεταξύ της δομημένης μεθόδου σχεδίασης βασισμένη σε σενάρια και του προτεινόμενου εργαλείου.

	Προτεινόμενη Μέθοδος	**Δομημένη Μέθοδος Σχεδίασης (Structured Scenario-Based Design Method)**
Στόχοι επιχειρήσης	Εξαγωγή πληροφοριών μέσω του εργαλείου Business Model Canvas	Δεν περιγράφονται τρόποι εξαγωγής πληροφοριών από την επιχείρηση
Απαιτήσεις χρήστη	Εκμαίευση πληροφοριών μέσω της βηματικής διαδικασίας (Persona Canvas→Ιστορίες	Δημιουργία personas χωρίς περιγραφή του τρόπου δημιουργίας αυτών

	Χρήσης→Περιπτώσεις Χρήσης Οι περιπτώσεις χρήσης δημιουργούν μεμονομένες απαιτήσεις οι οποίες χρήζουν υλοποίησης	Δημιουργία σεναρίων βασισμένων στις personas Δεν υπάρχει ιεράρχηση των απαιτήσεων Δεν παρέχει τρόπους αξιολόγησης των απαιτήσεων
Στόχος εργαλείου	Σχεδίαση προϊόντων λογισμικού με επικέντρο το χρήστη σε συνδυασμό με το στάδιο ανάπτυξης της επιχειρήσης και τους στόχους της	Σχεδίαση προϊόντων λογισμικού με επίκεντρο το χρήστη
Γενικά σχόλια	Παρέχει μία βηματική διαδικασία (step-by-step process) για την εξαγωγή απαιτήσεων	Παρέχει ένα εννοιολογικό πλαίσιο στο οποίο κάθε βήμα μπορεί να υλοποιηθεί με ποικίλους τρόπους μιας και η συγκεκριμένη μέθοδος δεν παρέχει κάποιο συγκεκριμένο εργαλείο εξαγωγής απαιτήσεων

Πίνακας 19 Συγκριτικός πίνακας προτεινόμενης μεθόδου και δομημένη μεθόδου σχεδίασης

6 *Εφαρμογή προτεινόμενου εργαλείου εξαγωγής απαιτήσεων σε πραγματική εταιρεία και αξιολόγηση*

Η διαδικασία αξιολόγησης συντελείται με δύο τρόπους, ποιοτικά και ποσοτικά.Ο ποιοτικός τρόπος, ο οποίος περιλάμβανε την παρουσίαση του εργαλείου, χρησιμοποιήθηκε από την πλευρά των εργαζόμενων της εταιρείας στο γραφείο του Δουβλίνου, το οποίο αποτελείται από πέντε μηχανικούς λογισμικού και δύο στελέχη στρατηγικής και επιχειρηματικής ανάπτυξης. Ο ποσοτικός τρόπος, ο οποίος περιλάμβανε τη συμπλήρωση ερωτηματολογίων, χρησιμοποιήθηκε για την εξέταση του εργαλείου από τα εργαζόμενους της εταιρείας από την σκοπιά των χρηστών εργαλείων εξαγωγής απαιτήσεων.Αξίζει να σημειωθεί ότι προτάθηκε και η συμπλήρωση ερωτηματολογίων από χρήστες της υπηρεσίας όμως κάτι τέτοιο ήταν αδύνατο.Ο λόγος που χρησιμοποιήθηκαν δύο διαφορετικοί τρόποι αξιολόγησης είναι γιατί επιδιωκόταν διαφορετικό είδος ανατροφοδότησης.

6.1 Προφίλ Εταιρείας

Προκειμένου να ελεγχθεί η ορθότητα και η αποτελεσματικότητα του προτεινόμενου εργαλείου εξαγωγής απαιτήσεων, το προαναφερθέν εργαλείο παρουσιάστηκε σε μία νεοφυή εταιρεία λογισμικού με έδρα το Λονδίνο και γραφεία σε Βερολίνο και Δουβλίνο η οποία με την μορφή εφαρμογής παρέχει στους χρήστες της ένα on-demand πληντύριο και στεγνοκαθαριστήριο όπου συλλέγει, πλένει και παραδίδει τα ρούχα μέσα σε 24 ώρες. Η συγκεκριμένη εταιρεία παρέχει τις υπηρεσίες της είτε μέσω του ιστότοπού της είτε μέσω της εφαρμογής της για κινητά και επιστρέφει τα

ρούχα στον πελάτη μέσα σε ένα προκαθορισμένο διάστημα μισής ώρας την επομένη της παραλαβής.

Πιο συγκεκριμένα, για τους εργαζόμενους της εταιρείας στο Δουβλίνο ήταν σημαντικό το εργαλείο να εξεταστεί σε βάθος και μάλιστα από άτομα εξειδικευμένα στα μοντέλα ανάπτυξης λογισμικού και στις μεθόδους εξαγωγής απαιτήσεων.Τέλος, η συλλογή δεδομένων μέσω των ερωτηματολογίων είναι φανερό πως αποτελεί σημαντικό εργαλείο αξιολόγησης για την άποψη των εργαζομένων ως χρηστών. Οι ενέργειες που ακολουθήθηκαν παρουσιάζονται αναλυτικά παρακάτω.

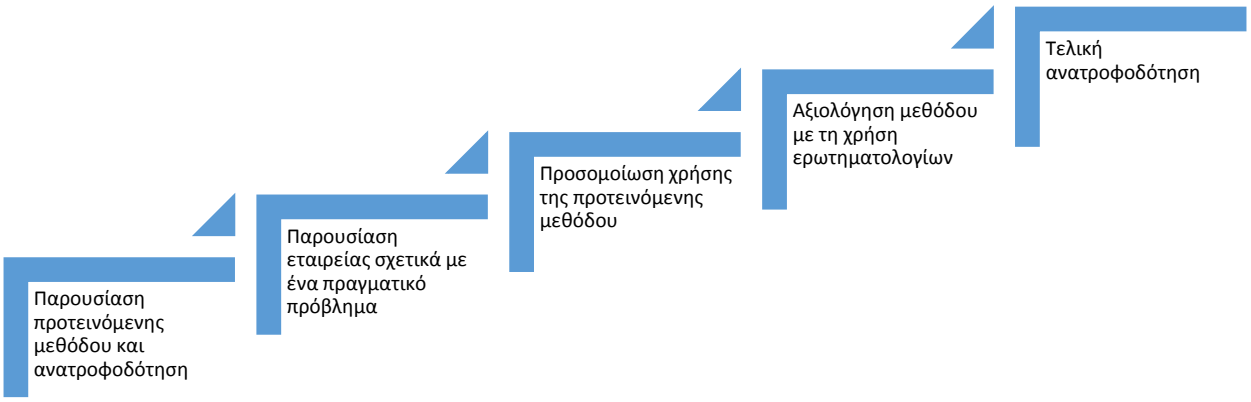
6.2 Μεθοδολογία εκτέλεσης της αξιολόγησης

Αρχικά προκαθορίστηκε μία ημερομηνία συνάντησης όπου η προτεινόμενη μέθοδος παρουσιάστηκε στην εταιρεία.Στη συγκεκριμένη συνάντηση έγινε μία επικοδομητική συζήτηση σχετικά με τις απαιτήσεις ενός προϊόντος λογισμικού, τους τρόπους διαχείρησης των επιχειρηματικών αλλά και τεχνικών απαιτήσεων ενός προϊόντος τέτοιας φύσεως καθώς και της χρησιμότητας/αποτελεσματικότητας της προτεινόμενης μεθόδου.

Εν συνεχεία, συμφωνήθηκε με την εταιρεία μία επόμενη συνάντηση προκειμένου να προσομοιωθεί η χρήση της προτεινόμενης μεθόδου σε ένα συγκεκριμένο είδος χρηστών της εταιρείας που τυγχάνει τη συγκεκριμένη χρονική περίοδο να αποσχολεί την ομάδα έργου ιδιαιτέρως. Για τη συγκεκριμένη συνάντηση συμφωνήθηκε να γίνει μία παρουσίαση της εταιρείας σχετικά με το εν λόγω πρόβλημα και να συμπληρωθουν τα πεδία του εργαλείο Business Model Canvas για την επιτάγχυνση της διαδικασίας.

Τέλος η αξιολόγηση της μεθόδους έγινε χρήση ερωτηματολογίων για την εκμαίευση ποσοτικών δεδομένων καθώς και μία τελική συζήτηση με την εταιρεία για την ανατροφοδότηση σχετικών με ποιοτικά χαρακτηριστικά της μεθόδου.

Η εν λόγω διαδικασία παρουσιάζεται συνοπτικά στην παρακάτω εικόνα.



Εικόνα 6-1 Στάδια διαδικασίας εφαρμογής και αξιολόγησης της προτεινόμενης μεθόδου

6.3 Ομάδα Καινοτομίας: Ποιοτικός τρόπος αξιολόγησης

Η ποιοτική αξιολόγηση του εργαλείου έγινε με την αναλυτική παρουσίαση του στην ομάδα εργασίας που εδρεύει στο Δουβλίνο και αποτελείται από πέντε μηχανικούς λογισμικού και δύο στελέχη στρατηγικής και επιχειρηματικής ανάπτυξης. Τα άτομα αυτά είναι γνώστες της έννοιας της καινοτομίας, των διαφορετικών μοντέλων ανάπτυξης λογισμικού καθώς και διάφορων τεχνικών εξαγωγής απαιτήσεων. Βασικός στόχος ήταν να ληφθεί η απαραίτητη ανατροφοδότηση και να εξεταστεί εάν το προτεινόμενο εργαλείο ανταποκρίνεται στις προσδοκίες και στους επιμέρους σκοπούς της εταιρείας. Η παρουσίαση η όποια διενεργήθηκε μέσω skype στις 18/9/2015 διήρκησε περίπου μία ώρα, κατά την οποία δόθηκε ο γενικός σκοπός του εργαλείου και ακολούθησε η αναλυτική επεξήγηση χρήσης του σε βήματα (step-by-step process). Στη συνέχεια ακολούθησαν ερωτήσεις της ομάδας κυρίως αναφορικά με διευκρινήσεις για τη λειτουργία του εργαλείου. Ιδιαίτερη έμφαση δόθηκε στον τρόπο εκμαίευσης και ανάλυσης των απαιτήσεων των χρηστών καθώς και στον καθορισμό μετρικών οι οποίες να συνδυάζουν την αξιολόγηση κάθε προσαύξησης τμημάτων λογισμικού τόσο σε προστηθέμενη αξία για το χρήστη καθώς και για την

επιχείρηση. Αφού τελείωσε η διαδικασία της παρουσίασης και μέσα από επικοδομητική συζήτηση ορίστηκε μία νέα ημερομηνία όπου η εταιρεία θα παρουσίαζε το επιχειρηματικό της πλάνο καθώς και τα χαρακτηριστικά των διαφορετικών χρηστών της προκειμένου να διενεργηθεί μία προσομοίωση χρήσης του προτεινόμενου εργαλείου.

Στις 26/9/2015 πραγματοποιήθηκε η παρουσίαση του επιχειρηματικού πλάνου της εταιρείας καθώς και η συμπλήρωση του ‘’Business Model Canvas’’ προκειμένου να υλοποιηθεί η προσομοίωση χρήσης του προτεινόμενου εργαλείου. Έμφαση δόθηκε σε ένα συγκεκριμένο είδος πελατών το οποίο παρουσιάζει ιδιαίτερο ενδιαφέρον για την εταιρεία, μιας και είναι το πιο απαιτητικό είδος τόσο σε επίπεδο εμπειρίας χρήστη (ως προς την εφαρμογή) όσο και σε επίπεδο υπηρεσίας (ποιότητα, χρόνος παράδοσης).

Η προσομοίωση χρήσης του εργαλείου είχε θετικό αντίκρισμα. Τα σημεία στα οποία εστίασε η ομάδα ήταν η ανταλλαγή πληροφορίας μεταξύ των μελών της μιάς και ο καθένας έχει την επιλογή εργασίας από το σπίτι. Σημείωσαν ότι το προτεινόμενο εργαλείο θα ήταν καλό να υλοποιηθεί σε μία αυτοματοποιημένη διαδικασία όπου το κάθε μέλος της ομάδας να είναι σε θέση να προσθέτει στοιχεία όπου θεωρεί σημαντικά καθώς και να παρατηρεί τις προσθήκες των συναδέλφων του. Πράγματι το προτεινόμενο εργαλείο παρέχει ένα εννοιολογικό πλαίσιο στο οποίο οι ομάδες εργασίας είναι σε θέση να ακολουθήσουν μία συγκεκριμένη διαδικασία για την εξαγωγή απαιτήσεων και η δημιουργία αυτού σε μία οπτικοποιημένη και αυτοματοποιημένη διαδικασία θα ενίσχυε την επικοινωνία και την αποτελεσματικότητα της ομάδας κατά τη διαδικασία ανάλυσης των απαιτήσεων.

Η ομάδα και ειδικότερα τα στελέχη στρατηγικής και επιχειρηματικής ανάπτυξης έδειξαν ιδιαίτερο ενδιαφέρον για τη συμμετοχή τους στη διαδικασία ανάλυσης και εξαγωγής των απαιτήσεων μιας και το οργανωτικό μοντέλο της εταιρείας δεν επιτρέπει τη συμμετοχή αυτών των στελεχών στην ανάλυση των απαιτήσεων παρά μόνο σε απαιτήσεις οι οποίες έχουν να κάνουν με την ποιότητα των παρεχόμενων υπηρεσιών και όχι των τεχνικών χαρακτηριστικών της υπηρεσίας. Συνολικά η ομάδα φάνηκε να ενθουσιάζεται από τη χρήση του εργαλείου μιας και όλα τα μέλη της ήταν

σε θέση να εκφράσουν διαφορετικές απόψεις οι οποίες να απορρέουν τόσο από το τεχνικό όσο και από το επιχειρηματικό πλαίσιο στο οποίο λειτουργούν.

Οι μηχανικοί λογισμικού έδειξαν ιδιαίτερο ενδιαφέρον για τον τρόπο εκμαίευσης απαιτήσεων από τους χρήστες μιας και η διαδικασία την οποία ακολουθούν είναι αρκετά διαφορετική από την προτεινόμενη. Έδειξαν θετική διάθεση στη χρήση του ‘’persona canvas’’ και στη βηματική διαδικασία ανάλυσης αυτού σε ιστορίες χρήσης και εν συνεχεία σε περιπτώσεις χρήσης. Σχολίασαν πως η συγκεκριμένη βηματική διαδικασία τους βοήθησε να καταλάβουν περισσότερα πράγματα για το χρήστη τον οποίο καλούνται να ικανοποιήσουν καθώς και η βηματική ανάλυση των στοιχείων των χρηστών σε ιστορίες χρήσης και εν συνεχεία σε περιπτώσεις χρήσης τους βοήθησε στο να εντοπίσουν περισσότερες απαιτήσεις κυρίως τεχνικής φύσεως εν συγκρίση με την μέθοδο όπου ήδη χρησιμοποιούσαν.

Ακόμη, οι μηχανικοί λογισμικού σχολίασαν θετικά την ομαδοποίηση στοιχείων στα τμήματα που προτείνει το υπό αξιολόγηση εργαλείο αφού όπως είπαν τους διευκολύνει στην αποσαφήνιση απαιτήσεων σχετικών με τη σχεδίαση του περιβάλλοντος αλληλεπίδρασης με το χρήστη. Τα στελέχη στρατηγικής και επιχειρηματικής ανάπτυξης σχολίασαν θετικά το συγκερασμό στοιχείων εισόδου τόσο επιχειρηματικού όσο και τεχνικού χαρακτήρα δίνοντας ιδιαίτερη έμφαση στις πληροφορίες που μπορούν να αντλήσουν σχετικά με την ανάπτυξη και επέκταση της εταιρείας τους.

6.3.1 Συμπεράσματα ποιοτικής αξιολόγησης

Τα παραπάνω σημεία αποτελούν τις βασικές επισημάνσεις της ομάδας αναφορικά με το προτεινόμενο εργαλείο εξαγωγής απαιτήσεων, έτσι ώστε να ληφθούν υπόψη κάποιες αλλαγές, οι οποίες πιθανώς να χρειάζονταν μετά την κυκλοφορία του εργαλείου για τη βελτίωση της λειτουργίας του.

Το γενικό συμπέρασμα από τη διαδικασία της ποιοτικής αξιολόγησης ήταν πως το εργαλείο θα ήταν αποδεκτό από επιχειρήσεις που ασχολούνται με την ανάπτυξη καινοτόμων προϊόντων λογισμικού με την βελτίωση κάποιων σημείων που προκάλεσαν προβληματισμούς. Οι προβληματισμοί και οι αλλαγές που τέθηκαν δεν επηρεάσαν τόσο το γενικό πλαίσιο, αλλά επιμέρους χαρακτηριστικά της διαδικασίας.

Περισσότερο οι απόψεις επηρέασαν τον τρόπο με τον οποίο πρέπει να ορίζονται οι μετρικές στο μοντέλο αναφορικά με τις επιχειρηματικές απαιτήσεις. Το τελικό σενάριο και πλαίσιο έγινε αποδεκτό ομόφωνα.

6.4 Ομάδα καινοτομίας: Ποσοτικός τρόπος αξιολόγησης

Προκειμένου να γίνει ποσοτικά η αξιολόγηση του εργαλείου που προτάθηκε, χρησιμοποιήθηκε η διαδικασία των ερωτηματολογίων. Τα ερωτηματολόγια της παρούσας φάσης είναι διαθέσιμα στο **Παράρτημα Ι**, όπου φαίνεται η μορφή τους και οι ερωτήσεις που κλήθηκε να απαντήσει η ομάδα καινοτομίας. Τα ερωτηματολόγια διαμοιράστηκαν ηλεκτρονικά.

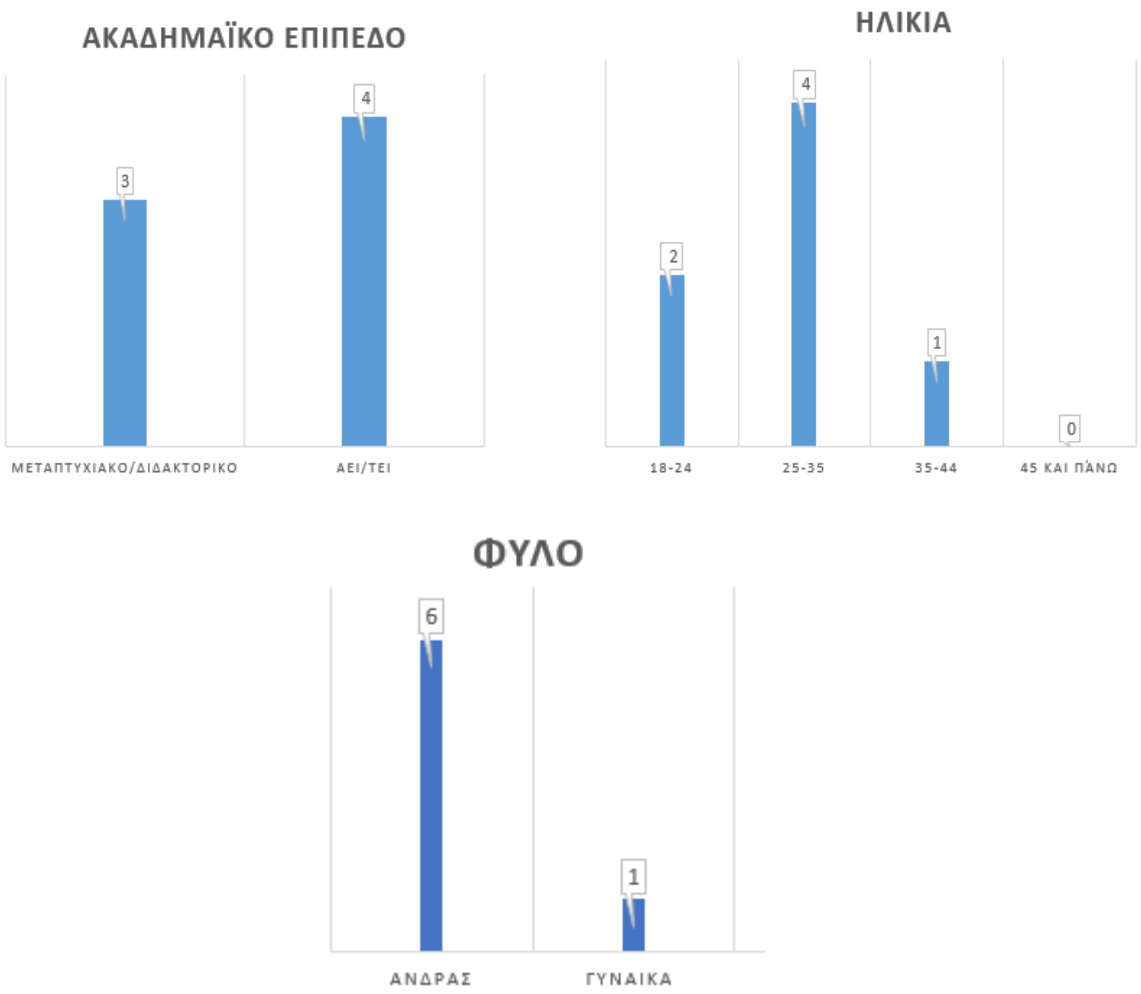
Οι ερωτήσεις που περιλαμβάνει κάθε ερωτηματολόγιο χωρίζονται σε τέσσερα μέρη.Το πρώτο μέρος περιλαμβάνει ερωτήσεις σχετικές με τα βασικά δημογραφικά χαρακτηριστικά του κάθε ερωτηθέντος. Το δεύτερο μέρεος έχει ως στόχο να ανακαλύψει τη σχέση του με την καινοτομία. Το τρίτο τμήμα αποτελείται από ερωτήσεις σχετικές με την εξαγωγή απαιτήσεων και εξετάζει το ενδιαφέρον του ερωτηθέντος να συμμετέχει στη χρήση παρόμοιων εργαλείων με το προτεινόμενο, ενώ το τέταρτο μέρος αφορά την εμπειρία χρήσης του ερωτηθέντος σε σχέση με το προτεινόμενο εργαλείο εξαγωγής απαιτήσεων. Ο βασικός στόχος των ερωτήσεων είναι να εντοπιστούν τα στοιχεία τα οποία θα έδιναν κίνητρο σε ομάδες καινοτομίας με διαφορετικά χαρακτηριστικά να χρησιμοποιούν εύχρηστα εργαλεία για την εξαγωγή απαιτήσεων κατά τη διαδικασία ανάπτυξης προϊόντων λογισμικού.

6.4.1 Αποτελέσματα ερωτηματολογίων

Τα διαγράμματα που ακολουθούν προέρχονται από τα δεδομένα που συλλέχθηκαν βάσει των απαντήσεων που δόθηκαν.

<u>Γενικά στοιχεία</u>

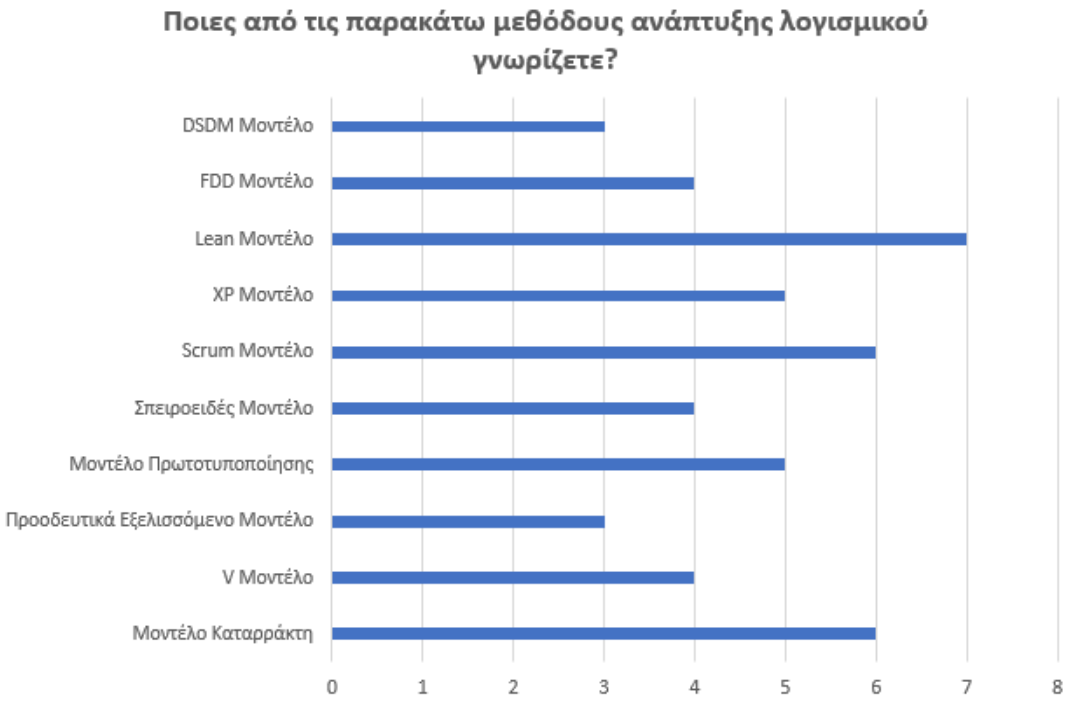
- Η πλειοψηφία των ερωτηθέντων άνηκε σε κλίμακα ηλικιών 25-35,απόφοιτοι ΑΕΙ/ΤΕΙ.Τα δημογραφικά χαρακτηριστικά τους συνοψίζονται στα παρακάτω διαγράμματα.



Εικόνα 6-2 Δημογραφικά χαρακτηριστικά ερωτηθέντων

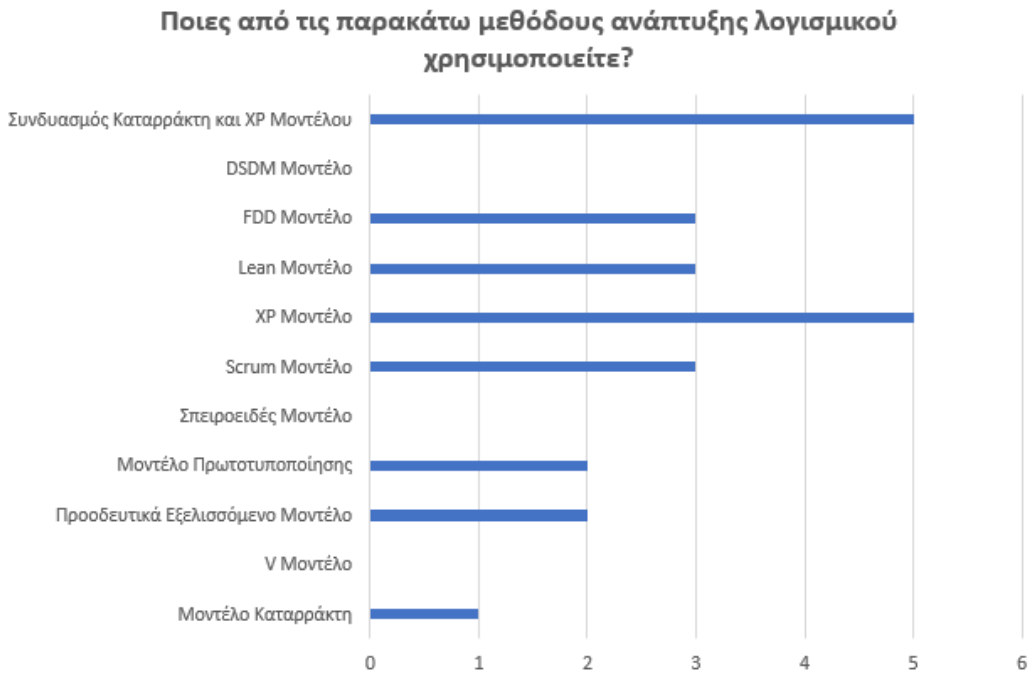
Καινοτομία

- Όσον αφορά τη σχέση των ερωτηθέντων με την καινοτομία στην ανάπτυξη προϊόντων λογισμικού, παρατηρήθηκε ότι η πλειοψηφία των ερωτηθέντων γνωρίζει το μοντέλο του καταρράκτη, ένα παραδοσιακό και αυστηρά προκαθορισμένο από άποψης διαδικασιών μοντέλο ανάπτυξης, αλλά και εύελικτα μοντέλα ανάπτυξης τα οποία έχουν κάνει την εμφάνισή τους τα τελευταία χρόνια με επικρατέστερο αυτό της Lean μεθόδου το οποίο συνδυάζει τεχνικές αλλά και επιχειρηματικές τεχνικές, πράγμα που επιβεβαιώνει την επικράτηση τέτοιων μεθόδων στις νεοφυείς επιχειρήσεις.Τέλος, παρατηρείται πως όλοι οι μηχανικοί λογισμικού της εταιρείας γνωρίζουν το XP μοντέλο, το οποίο χρησιμοποιείται συχνά από ομάδες μηχανικών λογισμικού.



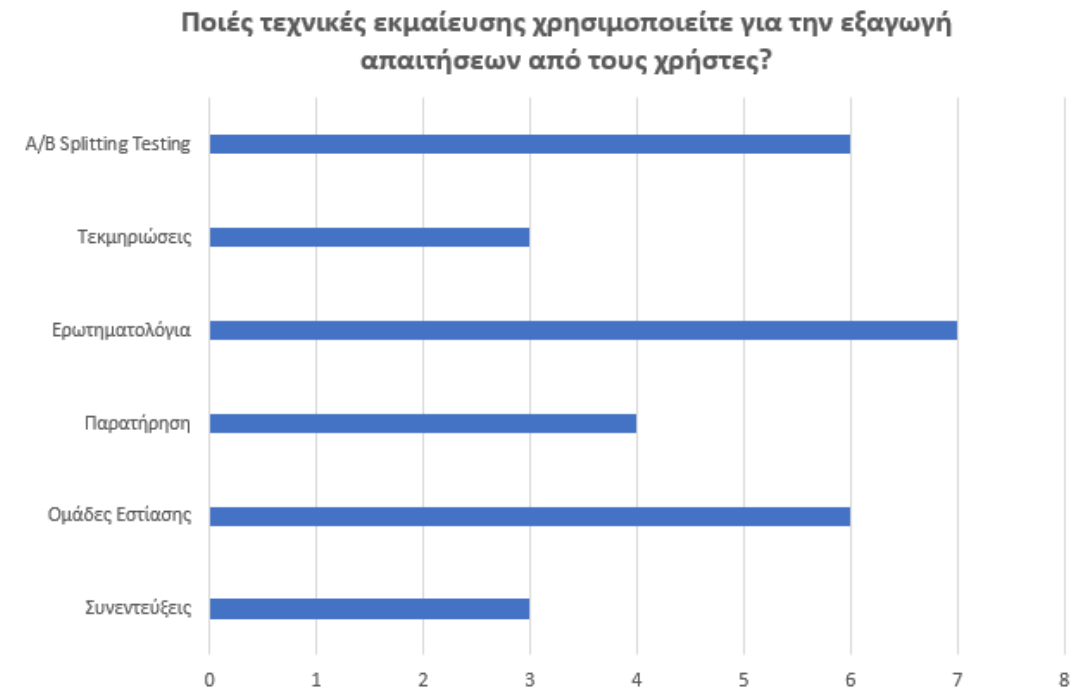
Εικόνα 6-3 Εξοικείωση ερωτηθέντων με τις επικρατέστερες μεθόδους ανάπτυξης λογισμικού

- Όσον αφορά την πρακτική εφαρμογή των μεθόδων ανάπτυξης λογισμικού, παρατηρείται πώς επικρατεί η χρήση του XP μοντέλου καθώς και όπως συμπλήρωσαν οι ερωτηθέντες χρησιμοποιούν στην ανάπτυξη του προϊόντος τους ένα συνδυασμό στοιχείων του μοντέλου καταρράκτη και στοιχείων του XP μοντέλου.Οι τάσεις αυτές διακρίνονται στο διάγραμμα της παρακάτω εικόνας.

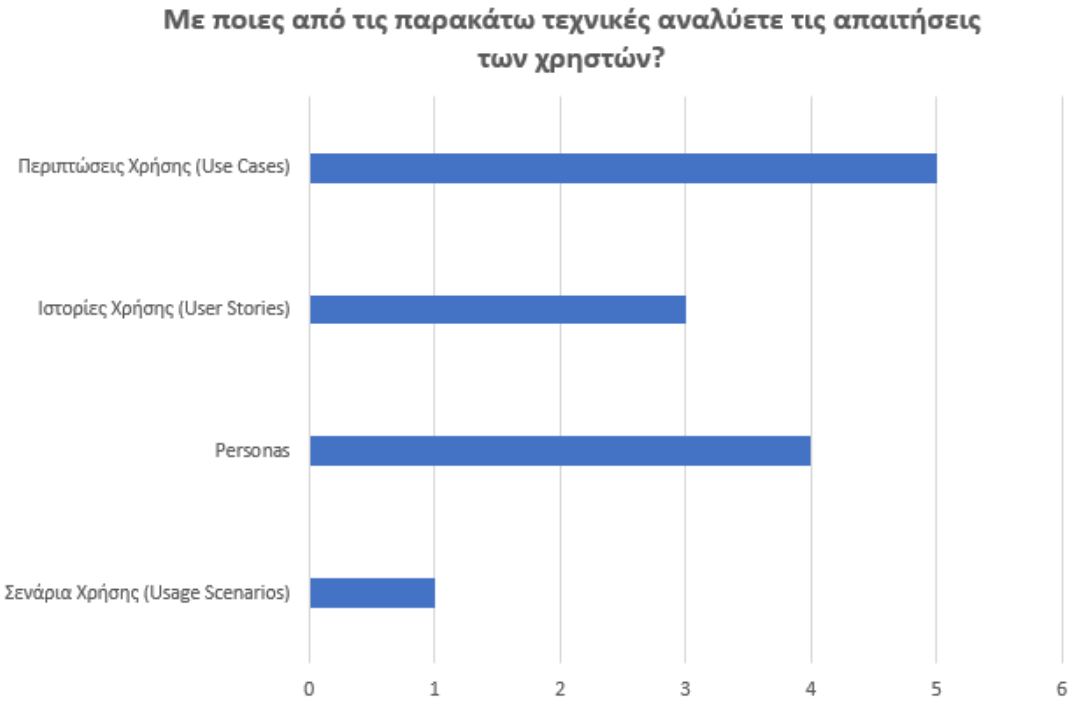


Εικόνα 6-4 Προτιμήσεις ερωτηθέντων σχετικά με τη συχνότητα χρήσης μεθόδων ανάπτυξης λογισμικού

- Όσον αφορά τις τεχνικές εκμαίευσης και ανάλυσης των απαιτήσεων, η πλειοψηφία των ερωτηθέντων φαίνεται πως χρησιμοποιεί για την εκμαίευση απαιτήσεων τεχνικές όπως τα ερωτηματολόγια, τις ομάδες εστίασης (focus groups) και τα A/B splitting tests τα οποία με ενδιαφέρον παρατηρήσαμε να χρησιμοποιούν, μιας και στην ανάλυση που έγινε σχετικά με τις τεχνικές εκμαίευσης απαιτήσεων δεν είχαν συμπεριληφθεί. Επίσης στην ερώτηση σχετικά με τις τεχνικές ανάλυσης των απαιτήσεων φαίνεται πως η πλειοψηφία των ερωτηθέντων (5 στους 7) χρησιμοποιούν περιπτώσεις χρήσης για την ανάλυση των απαιτήσεων, πρακτική η οποία είναι ιδιαιτέρως διαδεδομένη στους μηχανικούς λογισμικού, ενώ ιδιαίτερο ενδιαφέρον μας παρέχει η πληροφορία πως τέσσερις από τους επτά ερωτηθέντες φάνηκε να χρησιμοποιούν personas προκειμένου να αναλύσουν απαιτήσεις με ιδιαίτερη έμφαση στην ανάλυση των απαιτήσεων των χρηστών, καταδεικνύοντας ότι η χρήση των personas δεν γίνεται μόνο από μηχανικούς λογισμικού αλλά και από στελέχη επιχειρηματικής και στρατηγικής ανάπτυξης προκειμένου να αναλύσουν απαιτήσεις των χρηστών οι οποίες ξεπερνούν τον τεχνικό χαρακτηρά του προϊόντος.



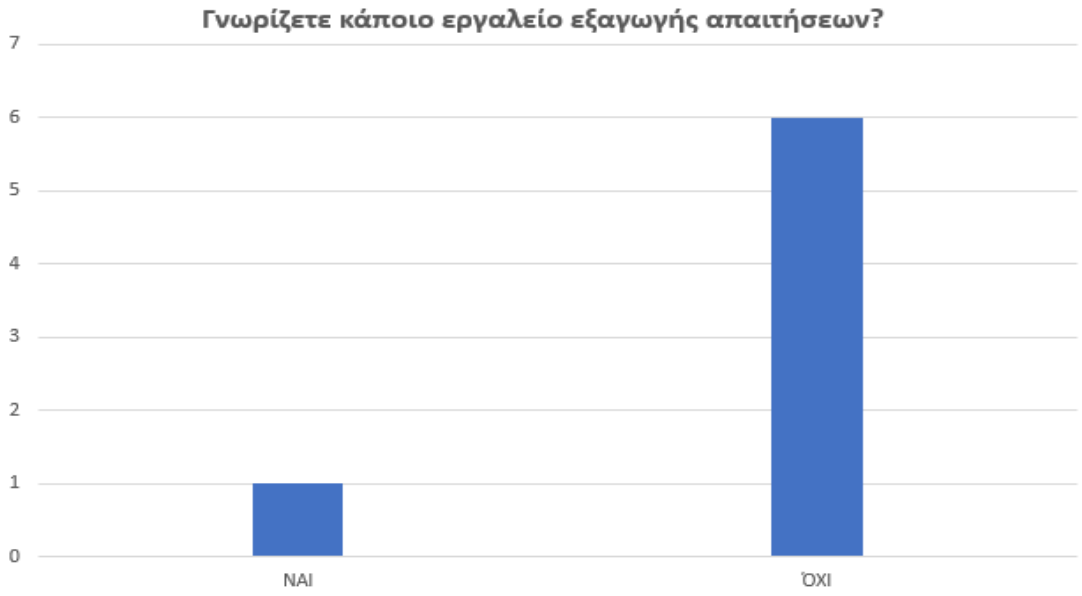
Εικόνα 6-5 Προτιμήσεις ερωτηθέντων σχετικά με διάφορες τεχνικές εκμαίευσης απαιτήσεων



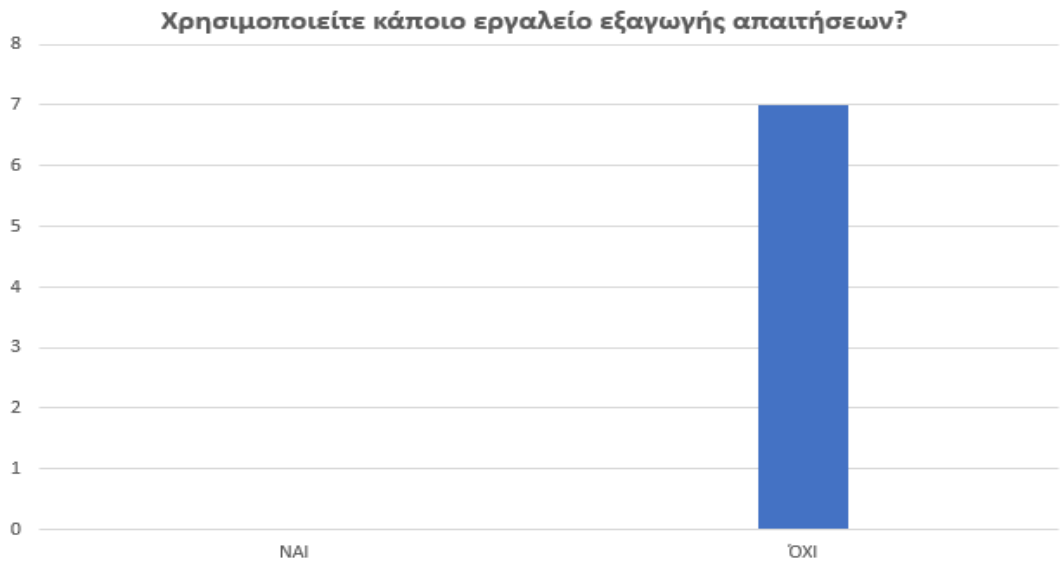
Εικόνα 6-6 Προτιμήσεις ερωτηθέντων σχετικά με διάφορες τεχνικές ανάλυσης απαιτήσεων

Εργαλεία εξαγωγής απαιτήσεων

- Όσον αφορά τη σχέση των ερωτηθέντων με εργαλεία εξαγωγής απαιτήσεων, παρατηρήθηκε ότι η πλειοψηφία αυτών δεν γνωρίζει την ύπαρξη κάποιου εργαλείου εξαγωγής απαιτήσεων, ενώ ακόμη πιο συντηρητικά ήταν ποσοστά στην ερώτηση εάν χρησιμοποιούν κάποιο εργαλείο εξαγωγής απαιτήσεων όπου η απόλυτη πλειοψηφία απάντησε ότι δεν χρησιμοποιεί κάποιο αντίστοιχο εργαλείο.

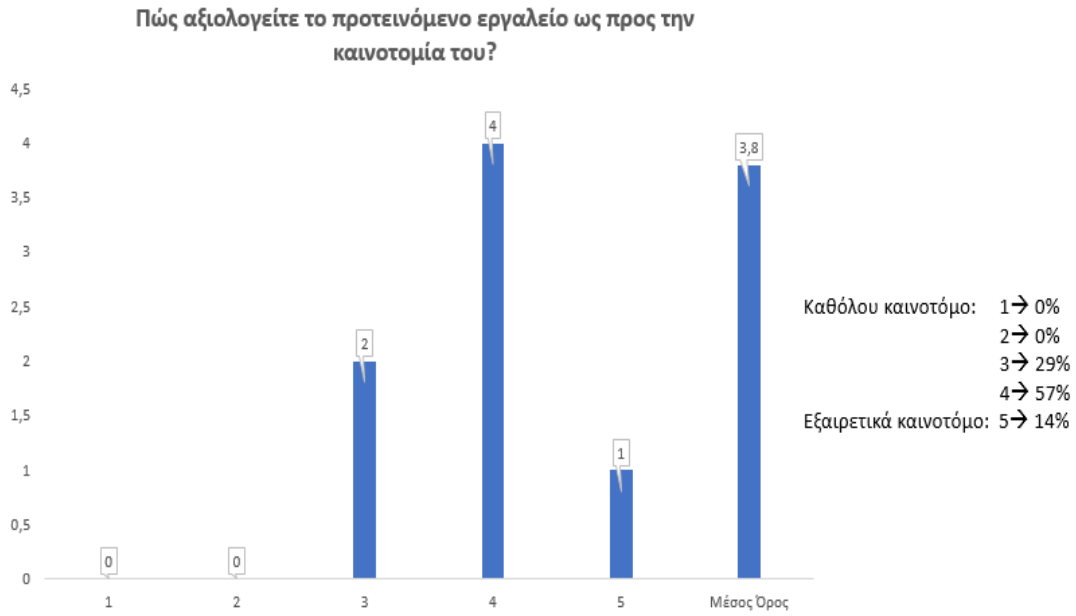


Εικόνα 6-7 Εξοικείωση ερωτηθέντων με εργαλεία εξαγωγής απαιτήσεων

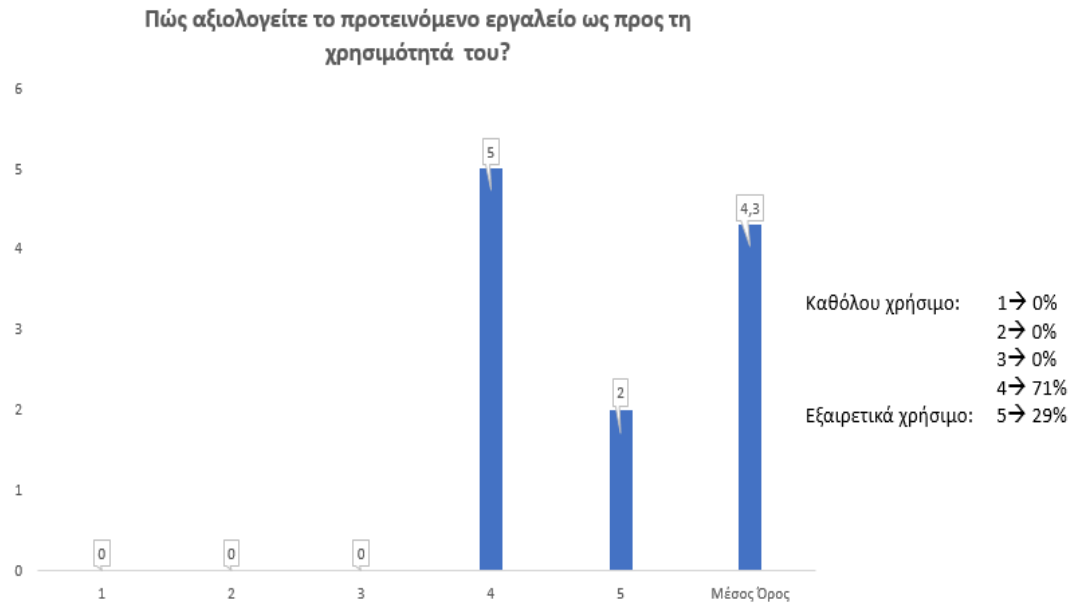


Εικόνα 6-8 Συχνότητα χρήσης εργαλείων εξαγωγής απαιτήσεων

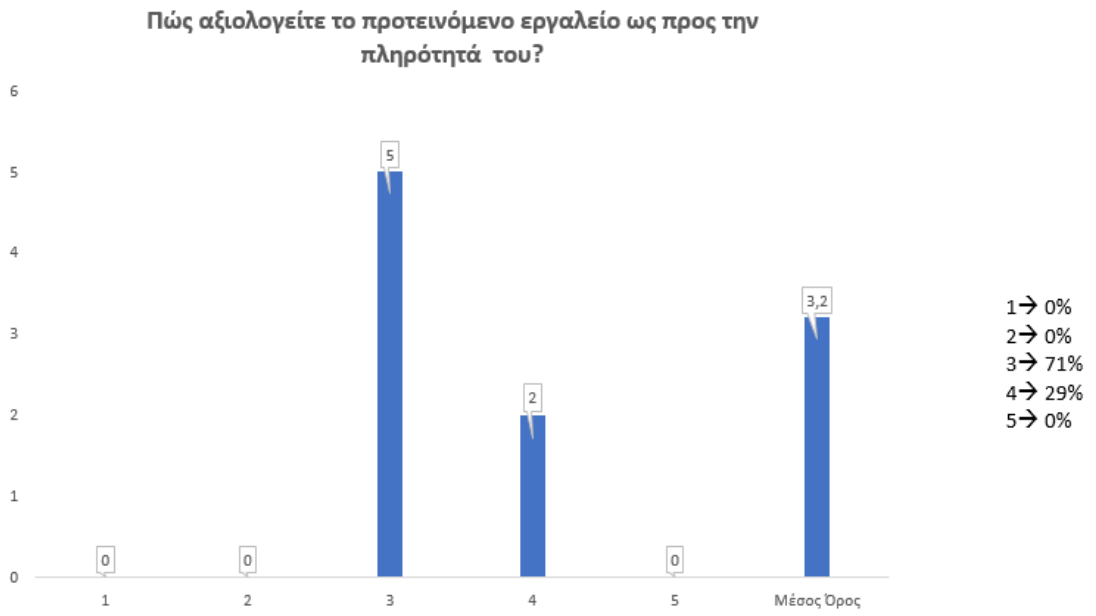
- Όσον αφορά το προτεινόμενο εργαλείο εξαγωγής απαιτήσεων οι ερωτηθέντες καλέστηκαν να το αξιολογήσουν με κριτήριο τέσσερις βασικούς άξονες, την καινοτομία, τη χρησιμότητα, την πληρότητα και την αποτελεσματικότητά του και να το βαθμολογήσουν σε μία κλίμακα από ένα έως και πέντε.Οι απαντήσεις αυτών ήταν αρκετά ενθαρρυντικές για το προτεινόμενο εργαλείο το οποίο αποτελέσε βασικό συστατικό της παρούσας διπλωματικής εργασίας. Ως προς την καινοτομία του εργαλείου, η πλειοψηφία των ερωτηθέντων το βαθμολόγησε με τέσσερις μονάδες ενώ ο μέσος όρος της βαθμολογίας άγγιξε το 3,8. Ως προς τη χρησιμότητα του εργαλείου, η πλειοψηφία των ερωτηθέντων το βαθμολόγησε με τέσσερις μονάδες ενώ ο μέσος όρος της βαθμολογίας ήταν 4,3/5. Ως προς την πληρότητά του τα αποτελέσματα ήταν λιγότερο ενθαρρυντικά, γεγονός το οποίο ήταν αναμενόμενο αφού και από την ποιοτική αξιολόγηση είχε εκφραστεί η επιθυμία των ερωτηθέντων να αυτοματοποιηθεί το προτεινόμενο εργαλείο προκειμένου να μπορούν όλοι οι συμμετέχοντες να αλληλεπιδρούν κατά τη διαδικασία εξαγωγής των απαιτήσεων. Ο μέσος όρος βαθμολογίας του εργαλείου ως προς την πληρότητα του ήταν 3,2/5, ο οποίος αποτελεί και το χαμηλότερο μέσο όρο.Τέλος, όσον αφορά την αποτελεσματικότητα του εργαλείου οι ερωτηθέντες στην πλειοψηφία τους το βαθμολογήσανε με τέσσερα στα πέντε, γεγονός που καταδεικνύει τη σημαντικότητα του εν λόγω εργαλείου στη διαδικασία εξαγωγής απαιτήσεων.



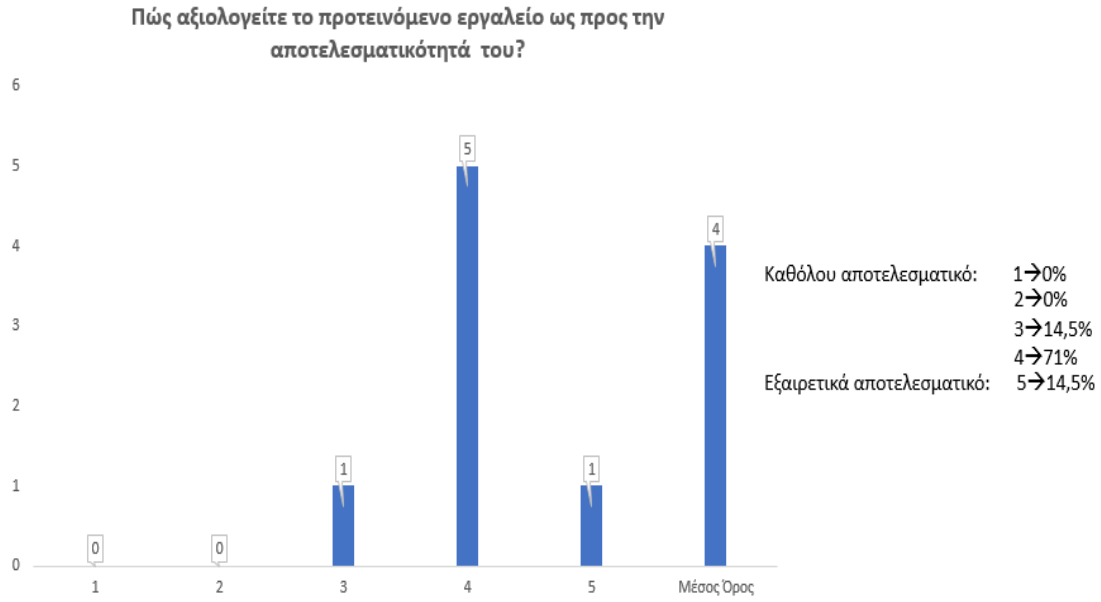
Εικόνα 6-9 Αξιολόγηση εργαλείου ως προς την καινοτομία



Εικόνα 6-10 Αξιολόγηση εργαλείου ως προς τη χρησιμότητα

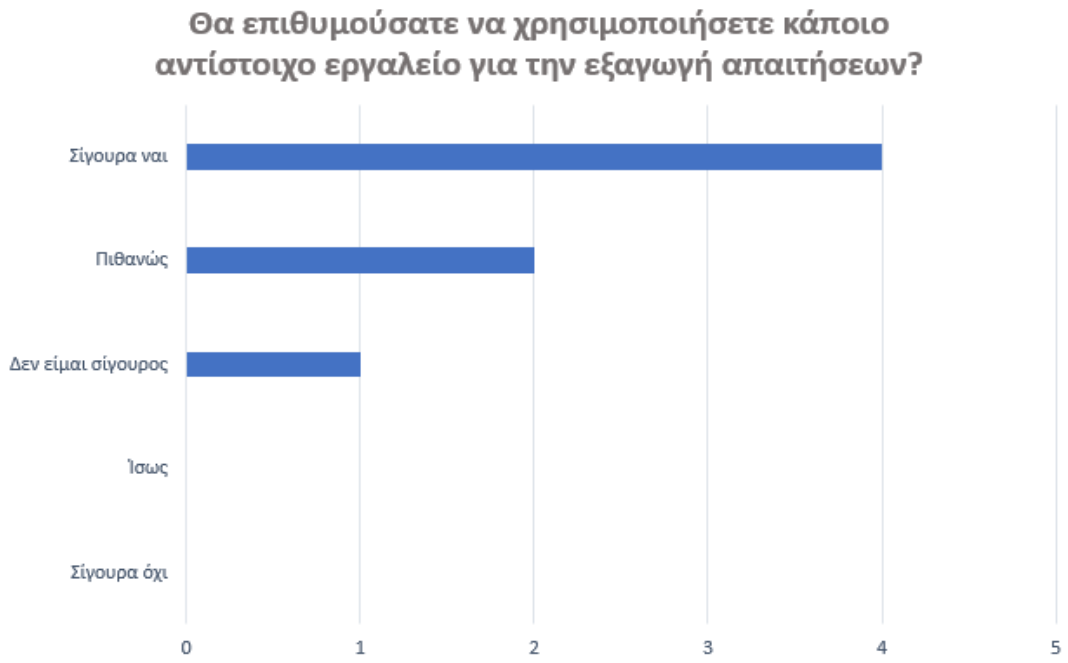


Εικόνα 6-11 Αξιολόγηση εργαλείου ως προς την πληρότητα



Εικόνα 6-12 Αξιολόγηση εργαλείου ως προς την αποτελεσματικότητα

- Τέλος, όσον αφορά το εάν οι ερωτηθέντες επιθυμούσαν να χρησιμοποιήσουν κάποιο αντίστοιχο εργαλείου και να το ενσωματώσουν στη διαδικασία ανάπτυξης του προϊόντος τους, η πλειοψηφία αυτών απάντησε πως σίγουρα θα επιθυμούσε τη χρήση κάποιου εργαλείου εξαγωγής απαιτήσεων στη διαδικασία ανάπτυξης ενός προϊόντος λογισμικού.



Εικόνα 6-13 Ανάλυση επιθυμίας των ερωτηθέντων σχετικά με τη χρήση αντίστοιχων εργαλείων

6.4.2 Συμπεράσματα ποσοτικής αξιολόγησης

Εξετάζοντας τα συνολικά αποτελέσματα των ερωτηματολογίων που συλλέχθηκαν είναι δυνατό να εξαχθούν ορισμένα συμπεράσματα.Οι διαφοροποιήσεις μεταξύ των δύο φύλων ήταν πολύ μικρές αναφορικά με τις απαντήσεις που δόθηκαν, καθώς η γενικότερη εικόνα είναι κοινή για άντρες και γυναίκες. Όσον αφορά την καινοτομία φαίνεται πως ενώ υπάρχει το ακαδημαϊκό υπόβαθρο όσον αφορά τα διάφορα μοντέλα ανάπτυξης λογισμικού, στην πράξη χρησιμοποιούνται μοντέλα τα οποία έχουν πολύ ξέκαθαρα χαρακτηριστικά τόσο για τους μηχανικούς λογισμικού όσο και τα επιχειρηματικά στελέχη. Όπως έγινε αντιληπτό οι μηχανικοί λογισμικού προτιμούν τη χρήση μοντέλων τα οποία απευθύνονται μόνο στο τεχνικό επίπεδο της επιχειρηματικής διαδικασίας που δεν είναι άλλο από την ανάπτυξη του λογισμικού. Έγινε αντιληπτό συνεπώς ότι δεν προτιμούνται μοντέλα τα οποία απαιτούν τη συνεργασία μηχανικών λογισμικού και επιχειρηματικών στελεχών.Ακόμη όσον αφορά τον τομέα των απαιτήσεων και των προδιαγραφών φαίνεται πως η εκμαίευση αυτών γίνεται με περισσότερους τρόπους από αυτούς που μελετήθηκαν στην παρούσα εργασία καθώς η πλειοψηφία των ερωτηθέντων ανέφερε την τεχνική του A/B splitting test. Όσον αφορά τις τεχνικές ανάλυσης των απαιτήσεων φαίνεται πως οι μηχανικοί λογισμικού προτιμούν παραδοσιακές τεχνικές όπως αυτές των

περιπτώσεων χρήσης, ενώ ιδιαίτερο ενδιαφέρον παρουσίασε το γεγονός ότι τα επιχειρηματικά στελέχη προτιμούν τη χρήση των personas για την ανάλυση των απαιτήσεων.Επίσης, αναφορικά με τα εργαλεία εξαγωγής απαιτήσεων τα στοιχεία που λάβαμε από τους ερωτηθέντες δεν ήταν ιδιαίτερως ενθαρρυντικά καθώς σχεδόν κανείς δεν γνωρίζει ή χρησιμοποιεί κάποιο εργαλείο εξαγωγής απαιτήσεων σε αντίθεση με την ιδιαίτερη προθημεία τους να χρησιμοποιήσουν κάποιο αντίστοιχο με το προτεινόμενο εργαλείο κατά τη διαδικασία εξαγωγής των απαιτήσεων. Τέλος όσον αφορά το προτεινόμενο εργαλείο η βαθμολόγηση του ως στοιχείο ποσοτικής αξιολόγησης έδειξε θετικά σημάδια, αφού η πλειοψηφία των ερωτηθέντων το βαθμολόγησε θετικά.

6.5 Συνολική ανατροφοδότηση και προσαρμογή του προτεινόμενου εργαλείου εξαγωγής απαιτήσεων

Συνολικά, τα σχόλια στο προτεινόμενο εργαλείο εξαγωγής απαιτήσεων αναφορικά με την τελική του μορφή ήταν τα εξής:

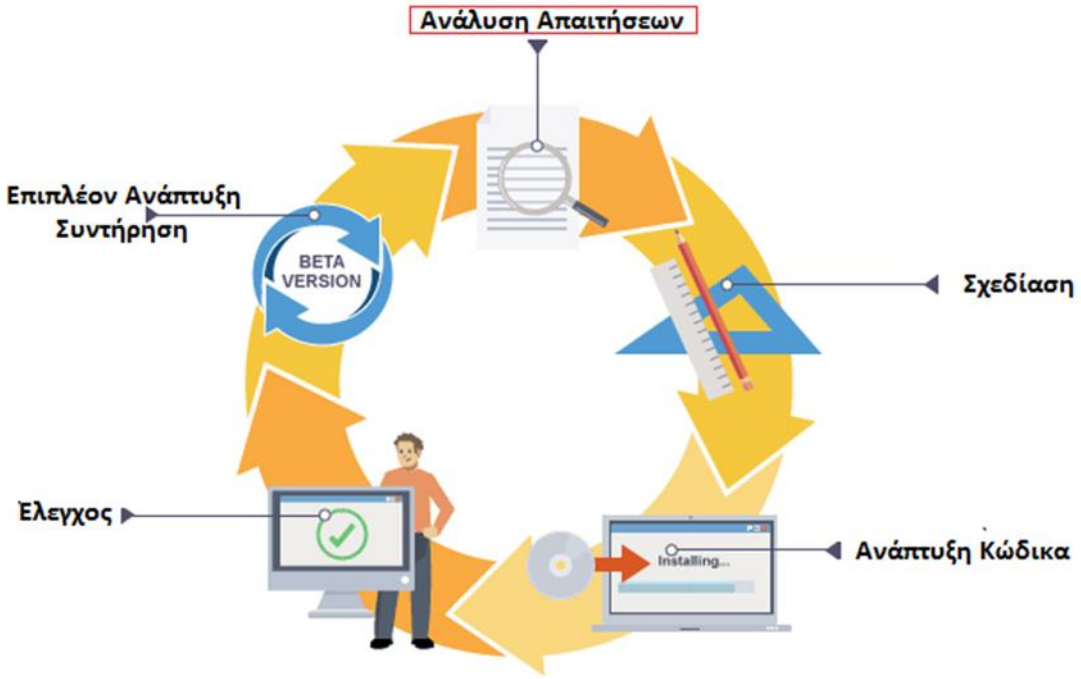
Βάσει της ποιοτικής αξιολόγησης:

- Η συνύπαρξη δύο ειδών μετρικών οι οποίες να αναφέρονται οι μεν στις επιχειρηματικές απαιτήσεις και οι δε στις τεχνικές
- Αυτοματοποίηση του προτεινόμενου εργαλείου για τη διευκόλυνση της αλληλεπίδρασης μεταξύ των μελών της ομάδας εργασίας

Βάσει της ποσοτικής αξιολόγησης:

- Προσθήκη της τεχνικής ανάλυσης απαιτήσεων A/B splitting test
- Έμφαση στην αλληλεπίδραση όλων των μελών της ομάδας εργασίας

6.6 Συνολική συνεισφορά της προτεινόμενης μεθόδου



Εικόνα 6-14 Συνεισφορά της προτεινόμενης μεθόδου στον κύκλο ζωής ανάπτυξης προϊόντων λογισμικού

Το εργαλείο το οποίο αναπτύχθηκε και παρουσιάστηκε στις προηγούμενες ενότητες του παρόντος κεφαλαίου επικεντρώνεται σε ένα συγκεκριμένο στάδιο του κύκλου ζωής ανάπτυξης ενός προϊόντος λογισμικού και πιο συγκεκριμένα σε ένα από τα πρωταρχικά στάδια του κύκλου, αυτό της ανάλυσης των απαιτήσεων. Όπως αναφέρθηκε στο Κεφάλαιο 1 τα περισσότερα έργα ανάπτυξης λογισμικού αποτυγχάνουν λόγω της έλλειψης στοιχείων εισόδου από τους χρήστες και από τις ασαφείς ή ακόμα και ατελείς προδιαγραφές/απαιτήσεις των χρηστών. Για το λόγο αυτό, το προτεινόμενο μοντέλο επικεντρώνεται στην εξαγωγή απαιτήσεων προκειμένου να προτείνει μία εναλλακτική λύση στο πρόβλημα. Επίσης η εφαρμογή του δίνει ιδιαίτερη σημασία και στις επιχειρηματικές πρακτικές προκειμένου το μοντέλο να αποτελεί ένα ουσιαστικό πρακτικό εργαλείο για τις επιχειρήσεις.

Όσον αφορά αντίστοιχες προτάσεις που μελετήθηκαν από την υπάρχουσα βιβλιογραφία δεν παρατηρήθηκε κάποιο αντίστοιχο εργαλείο. Ωστόσο, τα τελευταία χρόνια (2009 και μετά) παρατηρείται αυξανόμενο ενδιαφέρον στη δημιουργία εργαλείων εξαγωγής απαιτήσεων, αν και οι περισσότερες προτεινόμενες μέθοδοι παρέχουν ένα θεωρητικό εννοιολογικό πλαίσιο χωρίς την ύπαρξη αυστηρά καθορισμένων διαδικασιών προκειμένου να τεθούν σε πρακτική ισχύ από κάποια

επιχειρήση. Το προτεινόμενο από μεριάς μας εργαλείο αποσκοπεί στην κάλυψη του συγκεκριμένου κενού προκειμένου να συστηματοποιηθεί η διαδικασία εξαγωγής απαιτήσεων δίνοντας έμφαση στις καινοτόμες επιχειρήσεις όπου το περιβάλλον τους είναι ιδιαιτέρως ανταγωνιστικό και ασταθές.

7 Επίλογος

Μετά την αναλυτική προσέγγιση των μεθόδων ανάπτυξης λογισμικού, των διαδικασιών εξαγωγής απαιτήσεων σε καινοτόμα προϊόντα λογισμικού, την εφαρμογή και τη δοκιμή του προτεινόμενου εργαλείου εξαγωγής απαιτήσεων, παρουσιάζονται συνολικά τα αποτελέσματα της διπλωματικής εργασίας, ενώ διατυπώνονται πιθανές μελλοντικές επεκτάσεις.

7.1 Σύνοψη και συμπεράσματα

Οι καινοτόμες εταιρείες προϊόντων λογισμικού είναι εταιρείες χωρίς ιστορικό λειτουργίας και υποχρεούνται να παράξουν πολύ γρήγορα προϊόντα λογισμικού τελευταίας τεχνολογικής αιχμής. Αυτές οι εταιρείες αναπτύσσουν λογισμικό κάτω από εξαιρετικά αβέβαιες συνθήκες, με ελάχιστους πόρους και καλούνται να αντιμετωπίσουν ταχέως αναπτυσσόμενες αγορές.Οι νεοφυείς εταιρείες παρουσιάζουν ένα μοναδικό συνδυασμό χαρακτηριστικών τα οποία ελοχεύουν πολλούς κινδύνους όσον αφορά τις μεθόδους ανάπτυξης λογισμικού, δημιουργώντας προβλήματα στους μηχανικούς λογισμικού. Ενώ τα χρηματικά ποσά τα οποία επενδύονται στη δημιουργία προϊόντων λογισμικού συνεχώς αυξάνονται, τα ποσοστά αποτυχίας αντίστοιχων προϊόντων παραμένουν υψηλά, με κυριότερο λόγο αποτυχίας τους την έλλειψη σαφούς καθορισμού των απαιτήσεων. Με βάση τη συγκεκριμένη παρατήρηση η παρούσα διπλωματική εργασία μελέτησε τόσο από τεχνική όσο και από επιχειρηματική σκοπιά όλο το πλαίσιο στο οποίο αναπτύσσονται διάφορα προϊόντα λογισμικού, δίνοντας ιδιαίτερη έμφαση στο καθορισμό, στην ανάλυση και στην εξαγωγή των απαιτήσεων.Με τη βοήθεια ερευνητικών μελετών και παραδειγμάτων προτάθηκε ένα εργαλείο εξαγωγής απαιτήσεων, το οποίο θα μπορούσε να αποτελέσει μία κοινή βάση για την εξαγωγή απαιτήσεων σε καινότομες εταιρείες λογισμικού με διαφορετικά χαρακτηριστικά. Κατά την εφαρμογή του

διαπιστώθηκε ζήτηση για την ύπαρξη αντίστοιχων εργαλείων και μάλιστα υπό την μορφή αυτοματοποιημένων διαδικασίων στις οποίες τα μέλη μιας εταιρείας να μπορούν να αλληλεπιδρούν. Οι μετατροπές οι οποίες πιθανώς να χρειάζονταν για την ενσωμάτωση σε ένα γενικότερο πλαίσιο είναι πολύ μικρές και αυτό καταδεικνύει πως το προτεινόμενο εργαλείο αποτελεί ένα πλαίσιο ακόμη και για εταιρείες οι οποίες δεν έχουν την απαραίτητη εμπειρία στο σχεδιασμό καινοτόμων προϊόντων λογισμικού.Τέλος, η αξιολόγηση του κατέδειξε την ανάγκη της επιχειρηματικής κοινότητας για την ύπαρξη αντίστοιχων εργαλείων.

7.2 Μελλοντικές επεκτάσεις

Έχει παρατηρηθεί ότι τα ποσοστά αποτυχίας των έργων σχετικών με ανάπτυξη λογισμικού παραμένουν ιδιαίτερα υψηλά και βασικός λόγος αποτυχίας τους είναι η έλλειψη σαφώς καθοριμένων απαιτήσεων. Ενώ η διαδικασία ανάλυσης και εξαγωγής απαιτήσεων είναι ένας ιδιαίτερα δημοφιλής κλάδος της τεχνολογίας λογισμικού τα ποσοστά αποτυχίας παρουσιάζουν αυξητικές τάσεις.Επομένως, είναι σημαντικό να υπάρξει συστηματική και ολοκληρωμένη μελέτη ώστε να διασφαλίζεται αποδοτικότερα η επιτυχία της διαδικασίας καθορισμού των απαιτήσεων. Η παρούσα διπλωματική εργασία θα μπορούσε να αποτελέσει έναυσμα για επιπλέον έρευνα και να χρησιμοποιηθεί το προτεινόμενο πλαίσιο ως βάση για νέα εργαλεία εξαγωγής απαιτήσεων. Μία επέκταση που θα μπορούσε να λάβει χώρα θα ήταν η αυτοματοποίηση της σχεδίασης από το γενικό πλαίσιο σε σύστημα.Μία επιπλέον προσπάθεια μπορεί να αποτελέσει η ενδυνάμωση δευτερευόντων μηχανισμών και η εισαγωγή νέων. Αυτό θα είχε ως αποτέλεσμα την ανάγκη επανεξέτασης του προτεινόμενου πλαισίου και διεύρυνσής του. Οι τομείς στους οποίους θα ήταν δυνατό να ενσωματωθούν εργαλεία εξαγωγής απαιτήσεων είναι πολλοί, ωστόσο ως μελλοντικός στόχος θεωρείται η επέκτασή του και σε επιπλέον περιοχές. Αυτό θα συμβεί εάν ξεπεραστούν πιθανοί ενδοιασμοί που αποτρέπουν τη χρήση του καθώς και επιμέρους δραστηριότητες που δύσκολα προκαλούν ενδιαφέρον.

8 Παράρτημα Ι

8.1 Ερωτηματολόγιο Αξιολόγησης

Ερωτηματολόγιο Αξιολόγησης

Γενικά Στοιχεία

Φύλο

- Άντρας
- Γυναίκα

Ηλικία

- 18-24
- 25-34
- 45 και πάνω

Ακαδημαϊκό Επίπεδο

- Μεταπτυχιακό/Διδακτορικό
- ΑΕΙ/ΤΕΙ

Καινοτομία

Ποιές από τις παρακάτω μεθόδους ανάπτυξης λογισμικού γνωρίζετε?

- ○ Μοντέλο Καταρράκτη
- ○ V Μοντέλο
- ○ Προοδευτικά Εξελισσόμενο Μοντέλο
- ○ Μοντέλο Πρωτοτυποποίησης
- ○ Σπειροειδές Μοντέλο
- ○ Scrum Μοντέλο
- ○ XP Μοντέλο
- ○ Lean Μοντέλο
- ○ FDD Μοντέλο
- ○ DSDM Μοντελο
- ○ Άλλο: [ ]

Ποιές από τις παρακάτω μεθόδους ανάπτυξης λογισμικού χρησιμοποιείτε?

- ○ Μοντέλο Καταρράκτη
- ○ V Μοντέλο
- ○ Προοδευτικά Εξελισσόμενο Μοντέλο
- ○ Μοντέλο Πρωτοτυποποίησης
- ○ Σπειροειδές Μοντέλο
- ○ Scrum Μοντέλο
- ○ XP Μοντέλο
- ○ Lean Μοντέλο
- ○ FDD Μοντέλο
- ○ DSDM Μοντελο
- ○ Άλλο: [ ]

Ποιές τεχνικές εκμαίευσης χρησιμοποιείτε για την εξαγωγή απαιτήσεων από τους χρήστες?

- ○ Συνεντεύξεις
- ○ Ομάδες Εστίασης
- ○ Παρατήρηση
- ○ Ερωτηματολόγια
- ○ Τεκμηριώσεις
- ○ Άλλο: [ ]

Με ποιές από τις παρακάτω τεχνικές αναλύετε τις απαιτήσεις των χρηστών?

- ○ Σεναρια Χρήσης
- ○ Personas
- ○ Ιστορίες Χρήσης
- ○ Περιπτώσεις Χρήσης
- ○ Άλλο: [ ]

Εργαλεία Εξαγωγής Απαιτήσεων

Γνωρίζετε κάποιο εργαλείο εξαγωγής απαιτήσεων?

- ○ Ναι
- ○ Όχι

Χρησιμοποιείτε κάποιο εργαλείο εξαγωγής απαιτήσεων?

- ○ Ναι
- ○ Όχι

Πώς αξιολογείτε το προτεινόμενο εργαλείο ως προς την καινοτομία του?

	1	2	3	4	5	
Καθόλου Καινοτόμο	○	○	○	○	○	Εξαιρετικά Καινοτόμο

Πώς αξιολογείτε το προτεινόμενο εργαλείο ως προς τη χρησιμότητά του?

	1	2	3	4	5	
Καθόλου Χρήσιμο	○	○	○	○	○	Εξαιρετικά Χρήσιμο

Πώς αξιολογείτε το προτεινόμενο εργαλείο ως προς την πληρότητά του?

1	2	3	4	5
○	○	○	○	○

Πώς αξιολογείτε το προτεινόμενο εργαλείο ως προς την αποτελεσματικότητά του?

	1	2	3	4	5	
Καθόλου Αποτελεσματικό	○	○	○	○	○	Εξαιρετικά Αποτελεσματικό

Θα επιθυμούσατε να χρησιμοποιήσετε κάποιο αντίστοιχο εργαλείο για την εξαγωγή απαιτήσεων?
(1 - Σίγουρα όχι) (2 - Ίσως) (3 - Δεν είμαι σίγουρος) (4 - Πιθανώς) (5 - Σίγουρα ναι)

1	2	3	4	5
○	○	○	○	○

9 Βιβλιογραφία

[1]R. Winston, 'Managing the Development of Large Software Systems', *Proceeding of IEEE WESCON*, vol. 26, pp. 1-9, 1970.

[2]K. Forsberg and H. Mooz, 'The Relationship of System Engineering to the Project Cycle', *INCOSE International Symposium*, vol. 1, no. 1, pp. 57-65, 1991.

[3]R. Pressman, *Software engineering*. New York: McGraw-Hill Higher Education, 2010, pp. 41-42.

[4]M. Smith, *Software prototyping*. London: McGraw-Hill, 1991.

[5]B. Boehm, 'A spiral model of software development and enhancement', *Computer*, vol. 21, no. 5, pp. 61-72, 1988.

[6]A. Nilsson and T. Wilson, 'Reflections on Barry W. Boehm's β€A spiral model of software development and enhancement', *Int J Managing Projects in Bus*, vol. 5, no. 4, pp. 737-756, 2012.

[7]M. AksMit and S. Matsuoka, *ECOOP '97*. Berlin: Springer, 1997.

[8]K. Scott, *The unified process explained*. Boston: Addison-Wesley, 2002.

[9]S. Ambler, *Agile modeling*. New York: J. Wiley, 2002.

[10]C. Larman, *Agile and iterative development*. Boston: Addison-Wesley, 2004.

[11]J. Highsmith and A. Cockburn, 'Agile software development: the business of innovation', *Computer*, vol. 34, no. 9, pp. 120-127, 2001.

[12]T. DeMarco, 'All Late Projects Are the Same', *IEEE Softw.*, vol. 28, no. 6, pp. 104-104, 2011.

[13]K. Beck, *Extreme programming eXplained*. Reading, MA: Addison-Wesley, 2000.

[14]J. Highsmith, *Adaptive software development*. New York: Dorset House Pub., 2000.

[15]K. Schwaber and M. Beedle, *Agile software development with Scrum*. Upper Saddle River, NJ: Prentice Hall, 2002.

[16]M. Poppendieck and T. Poppendieck, *Leading lean software development*. Upper Saddle River, NJ: Addison-Wesley, 2010.

[17]M. Porter, *Clusters of innovation initiative*. Washington, D.C.: Council on Competitiveness, 2002.

[18]M. Tushman and P. Anderson, *Managing strategic innovation and change*. New York: Oxford University Press, 1997.

[19]W. Abernathy and K. Clark, *Innovation*. Boston, Mass. (Soldiers Field, Boston 02163): Division of Research, Graduate School of Business Administration, Harvard University, 1983.

[20]C. Christensen, *The innovator's dilemma*. Boston, Mass.: Harvard Business School Press, 1997.

[21]S. Blank, *The Four Steps to the EpiphanyThe Four Steps to the Epiphany*. Menlo Park: K&S Ranch Publishing LLC, 2013.

[22]A. Osterwalder, Y. Pigneur and T. Clark, *Business model generation*. 2010.

[23]A. Osterwalder, Y. Pigneur, G. Bernarda and A. Smith, *Value proposition design*. Hoboken, New Jersey: Wiley, 2014.

[24]E. Ries, *The lean startup*. New York: Crown Business, 2011.

[25]A. Maurya, *Running lean*. Sebastopol, CA: O'Reilly, 2012.

[26]A. Croll and B. Yoskovitz, *Lean analytics*. Sebastopol, CA: O'Reilly, 2013.

[27]L. Klein, *UX for lean startups*. Farnham: O'Reilly, 2013.

[28]T. Lockwood, *Design thinking*. New York, NY: Allworth Press, 2010.

[29]I. Mootee, *Design thinking for strategic innovation*. 2013.

[30]S. Robertson and J. Robertson, *Mastering the requirements process*. Upper Saddle River, NJ: Addison-Wesley, 2006.

[31]I. Sommerville and P. Sawyer, *Requirements engineering*. Chichester: Wiley, 1997.

[32]M. Christel and K. Kang, *Issues in requirements elicitation*. Pittsburgh, Pa.: Carnegie Mellon University, Software Engineering Institute, 1992.

[33]B. Berenbach, *Software & systems requirements engineering*. New York: McGraw-Hill, 2009.

[34]I. Alexander and L. Beus-Dukic, *Discovering requirements*. Chichester, West Sussex, England: Wiley, 2009.

[35]A. Lamsweerde, *Requirements engineering*. Chichester, England: John Wiley, 2009.

[36]P. Bra, A. Kobsa and D. Chin, *User modeling, adaptation, and personalization*. Berlin: Springer-Verlag, 2010.

[37]A. Cockburn, *Writing effective use cases*. Boston: Addison-Wesley, 2001.

[38]A. Cooper, *The inmates are running the asylum*. Indianapolis, Ind.: Sams, 1999.

[39] Austin Govella., *Information Architecture*. New Riders, 2009.

[40]L. Nielsen, *Personas -- user focused design*. London: Springer, 2013.

[41]I. Myers and P. Myers, *Gifts differing*. Palo Alto, Calif.: Davies-Black Pub., 1995.

[42]J. Wiggins, *The five-factor model of personality*. New York: Guilford Press, 1996.

[43]H. Podeswa, *UML for the IT business analyst*. Boston, MA: Thomson Course Technology PTR, 2005.

[44]M. Fowler, *UML distilled*. Boston: Addison-Wesley, 2004.

[45]D. Kulak and E. Guiney, *Use cases*. Reading, Mass: Addison-Wesley, 2000.

[46]T. Lowdermilk, *User-centered design*. Boston, USA: Preece, 2004.

[47]D. Norman and S. Draper, *User centered system design*. Hillsdale, N.J.: Lawrence Erlbaum Associates, 1986.

[48]C. Righi and J. James, *User-centered design stories*. Amsterdam: Elsevier/Morgan Kaufman, 2007.

[49]M. Weiser, 'The computer for the 21 st century', *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 3, no. 3, pp. 3-11, 1999.

[50]M. Kurosu, *Human Centered Design*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[51]K. Yanagida, Y. Ueda, K. Go, K. Takahashi, S. Hayakawa and K. Yamazaki, 'Structured Scenario-Based Design Method', *Human Centered Design*, pp. 374-380, 2009.

[52]M. Kurosu, *Human centered design*. Heidelberg: Springer, 2011.

[53]E. Estelles-Arolas and F. Gonzalez-Ladron-de-Guevara, 'Towards an integrated crowdsourcing definition', *Journal of Information Science*, vol. 38, no. 2, pp. 189-200, 2012.

[54]D. Murray-Rust, O. Scekic, H. Truong, D. Robertson and S. Dustdar, 'A Collaboration Model for Community-Based Software Development with Social Machines', *Proceedings of the 10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2014.

[55]A. Craddock, B. Fazackerley, S. Messenger, B. Roberts and J. Stapleton, *DSDM Atern*, 2009.