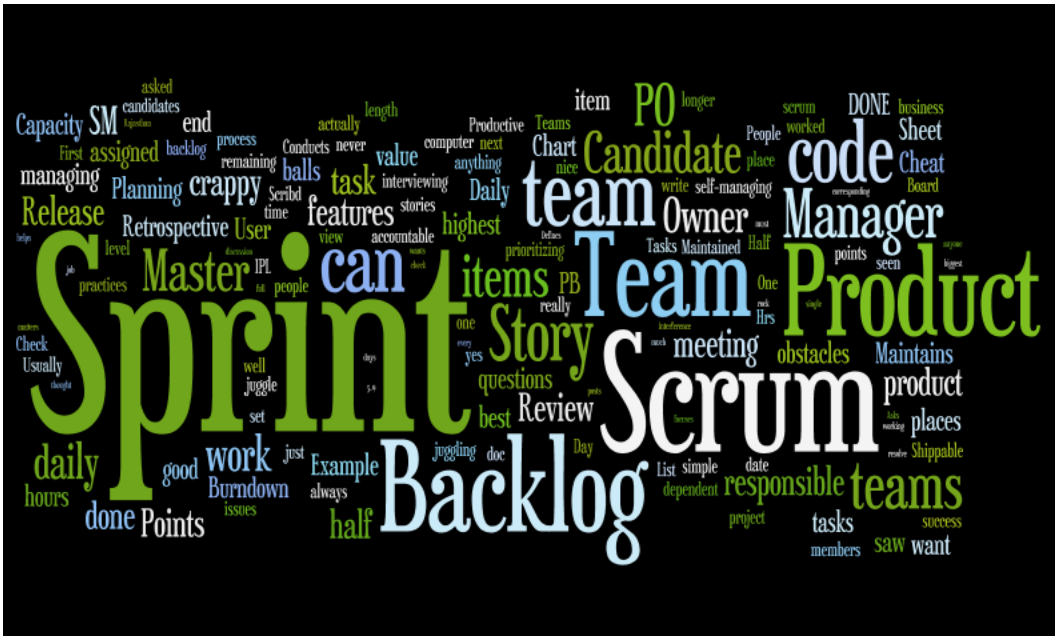
ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Αξιολόγηση των προτύπων και πρακτικών διασφάλισης της ποιότητας στα Ευέλικτα Έργα



Του φοιτητή

Κεφάλα Ευστράτιου
Αρ. Μητρώου: 02/2071

Επιβλέπων Καθηγητής

Παναγιώτης Σφέτσος

Θεσσαλονίκη 2011

Πρόλογος

Λίγοι θα διαφωνήσουν με τη θέση ότι το λογισμικό αποτελεί ένα από τα πιο σημαντικά εργαλεία που ο άνθρωπος παρήγαγε. Από καιρό ξέρουμε ότι οι ειδικευόμενοι άνθρωποι είναι οι πιο κατάλληλοι για την ανάπτυξη λογισμικού. Όμως ο καθένας μας μπορεί να φανταστεί κάποιους παράγοντες για να βελτιώσει την παραγωγικότητα του λογισμικού. Ωστόσο χρειάστηκαν πάνω από 10 χρόνια οι ευέλικτες μέθοδοι για να μπορέσουν να εδραιώσουν την άποψη του ότι πρέπει να δοθεί παραπάνω έμφαση στους ανθρώπους και την αλληλεπίδραση τους. Από τότε είμαστε μάρτυρες μιας μετακίνησης που έχει επιταχυνθεί περισσότερο από οποιαδήποτε άλλη καινοτομία στον τομέα της τεχνολογίας λογισμικού.

Οι ευέλικτες μέθοδοι άλλαξαν δραματικά τις διαδικασίες ανάπτυξης λογισμικού. Οι ευέλικτες διαδικασίες ανάπτυξης λογισμικού όπως η Exreme Programming, η Scrum και άλλες στηρίζονται σε καλύτερες πρακτικές που υπολογίζεται ότι θα βελτιώσουν την ποιότητα στην ανάπτυξη λογισμικού. Οι υποστηρικτές των ευέλικτων μεθόδων υποστηρίζουν ότι λόγο του μεγάλου συνόλου των ιδιοτήτων αυτών των μεθόδων, η ποιότητα στα ευέλικτα έργα λογισμικού πρέπει να είναι φυσικό επακόλουθο των εφαρμόσιμων μεθόδων. Η ποιότητα λογισμικού στην ευέλικτη ανάπτυξη δεν είναι ένα απλό θέμα. Δεν είναι λοιπόν εύκολο να μπορέσουμε να προσφέρουμε μια απάντηση που στόχο να έχει να λύση αυτό το θέμα. Σαν συνέπεια η διασφάλιση της ποιότητας στην ευέλικτη ανάπτυξη λογισμικού ελπίζουμε, περιμένουμε και υποθέτουμε να είναι λίγο πολύ ενσωματωμένη στις ευέλικτες διαδικασίες, καθώς οι πρακτικές διασφάλισης της ποιότητας του λογισμικού ολοκληρώνονται κατά την διάρκεια ολόκληρου του κύκλου ζωής της ανάπτυξης από το στάδιο των απαιτήσεων μέχρι και την τελική απελευθέρωση του προϊόντος.

Περίληψη

Ένας από τους στόχους αυτής της πτυχιακής εργασίας είναι να μας εξηγήσει τι είναι η ποιότητα και γιατί είναι τόσο σημαντική σε όλα τα στάδια της ανάπτυξης του λογισμικού και σύμφωνα με ποια κριτήρια την επιτυγχάνουμε. Γιατί πρέπει να την διασφαλίσουμε κατά την ανάπτυξη και πως αυτό μπορεί να επιτευχθεί με διάφορες τεχνικές

Αναφέρουμε τα πιο σημαντικά ευέλικτα μοντέλα ανάπτυξης λογισμικού και βλέπουμε πως μπορούμε να χρησιμοποιήσουμε τις διάφορες πρακτικές διασφάλισης της ποιότητα λογισμικού μέσα σε αυτά ώστε τα προϊόντα που θα παραχθούν από αυτά να πληρούν τις προϋποθέσεις του πελάτη.

Γίνεται μια αντιπαράθεση μεταξύ παραδοσιακού τρόπου ανάπτυξης λογισμικού και ευέλικτων μεθόδων και παρουσιάζονται παραδείγματα με πίνακες και σχέδια για το πώς θα αναπτυσσόταν ένα λογισμικό με βάση το παραδοσιακό μοντέλο καταρράκτη και με βάση ευέλικτα μοντέλα.

Παρουσιάζονται επιπλέον τα διάφορα στάδια από τα οποία θα περάσει η ανάπτυξη ενός λογισμικού σύμφωνα με τις ευέλικτες μεθόδους.

Τέλος αναφερόμαστε στα εργαλεία που χρησιμοποιούνται για την ανάπτυξη λογισμικού τα οποία ακολουθούν την λογική των ευέλικτων μεθόδων και παρουσιάζουμε μερικά από αυτά χωρίζοντας τα χαρακτηριστικά τους και σε πλεονεκτήματα και μειονεκτήματα.

Περιεχόμενα

Πρόλογος..... 3

Περίληψη..... 4

Εισαγωγή..... 10

1. Ποιότητα και Πρότυπα διασφάλισης ποιότητας

1.1 Τι είναι ποιότητα..... 12
1.2 ISO/IEC 12207.....13
1.3 ISO/IEC 9126.....16
1.4 Καθορισμός των αμφισβητούμενων εννοιών στο ISO/IEC 9126-1..... 18

2. Ευέλικτες μέθοδοι ελέγχου ποιότητας και αξιοπιστίας λογισμικού

2.1 Εισαγωγή..... 22
2.2 Ορισμός της ευέλικτης μεθόδου.....23
2.3 Θεωρητικός ορισμός..... 24
2.4 Λειτουργικός ορισμός.....27
2.5 Εκτίμηση της ποιότητας στις Ευέλικτες διαδικασίες.....30
2.6 Τεχνικές..... 32
2.7 Το πλαίσιο αξιολόγησης της ευέλικτης μεθοδολογίας..... 33
2.8 Διαδικασίες και Μετρικές Ποιότητας Προϊόντος στην παραδοσιακή και Ευέλικτη Ανάπτυξη Λογισμικού..... 34
2.9 Ποιότητα και Διαδικασία Βελτίωσης Λογισμικού.....34
2.10 Τι να μετρήσουμε και πώς να βελτιώσουμε την ποιότητα στην ευέλικτη ανάπτυξη.....37
2.11 Μετρήσεις του προϊόντος ή της διαδικασίας ποιότητας στην ανάπτυξη ευέλικτου λογισμικού;..... 37
2.12 Παράγοντες Δοκιμής, Βελτίωσης και επανακατασκευής στον Παραδοσιακό και Ευέλικτο κύκλο ζωής των διαδικασιών..... 38

3. Extreme Programming (XP)

3.1 Εισαγωγή……40
3.2 XP vs Πρότυπο Ικανότητας Ωρίμανσης (XP vs Capability Maturity Model)……41
3.3 XP vs Sommerville-Sawyer Πρότυπο……43
3.4 Γεφύρωμα της XP με την τεκμηρίωση απαιτήσεων……43
3.5 Πολλαπλοί εκπρόσωποι των πελατών……44
3.6 Τροποποίηση του σχεδιασμού της τακτικής(Modified Planning Game)……45
3.7 Πρακτικές XP……46

4. Ο κύκλος Ζωής των XP έργων(The XP projects lifecycle)

4.1 Πλάνο του χρήστη……49
4.2 Αρχιτεκτονική Αιχμή(Architectural Spike)……49
4.3 Προγραμματισμός Έκδοσης(Release Planning)…… 50
4.4 Επαναλήψεις(Iterations)…… 50
4.5 Αποδεκτές Δοκιμές(Acceptance Testing)…… 51
4.6 Εκδόσεις(Release)…… 51

5. Pair Programming

5.1 Εισαγωγή……52
5.2 Επικοινωνία……53
5.3 Ανατροφοδότηση……53
5.4 Τόλμη……54
5.5 Απλότητα……54
5.6 Σεβασμός……55

6. Θεωρητικό Πλαίσιο 5-Α μοντέλο(5-A Model

6.1 Ανάλυση…… 57

7. Ανάλυση των Πρακτικών της XP

7.1 Πρακτικές διαχείρισης έργου……..61
7.2 Εφαρμογή των πρακτικών…….....62
7.3 Πρακτικές επαλήθευσης και επικύρωσης(Verification & Validation practices)…….. 63
7.4 Διαμόρφωση των πρακτικών διαχείρισης(Configuration management practices)……64

8. Η μέθοδος SCRUM

8.1 Εισαγωγή……66
8.2 Τι είναι Scrum…… 68
8.3 Το αντίθετο της μεθόδου Καταρράκτη(Waterfall)……71
8.4 Βασικά στοιχεία της Scrum(Scrum Basics)……..... 72
8.5 Διαχείριση έργου σύγκριση Παραδοσιακής μεθόδου με Scrum 72
8.6 Οι ρόλοι της Scrum(Scrum Roles)……..79
8.7 Η Διαδικασία…….....79
8.8 Τεχνάσματα της Scrum(Scrum Artifacts)……..80
8.9 Γιατί η επαναληπτική ανάπτυξη δουλεύει……..86

9. Διασφάλιση Ποιότητας Λογισμικού

9.1 Εισαγωγή……88
9.2 Μοντέλο Καταρράκτη εναντίων Ευέλικτων μεθόδων…….. 89
9.3 Τεχνικές διασφάλισης ποιότητας…… 90
9.4 Το μοντέλο Καταρράκτη με SQA και V&V……91
9.5 Ευέλικτοι μέθοδοι με QA……92
9.6 Ευέλικτοι μέθοδοι, Τεχνικές ποιότητας……92

10. Σχεδιασμός και εργαλεία Ευέλικτων Έργων

10.1 Εισαγωγή……96
10.2 Υποψήφια Εργαλεία…….....97
10.3 Συνολική Σύγκριση……97

11. Agilefant

11.1 Η Ιδέα(Concept)…… .101
11.2 Βιωσιμότητα, Υποστήριξη & Τεκμηρίωση……....101
11.3 Ευχρηστία……..101
11.4 Πλεονεκτήματα……....102
11.5 Αδυναμίες……..102
11.6 Γενική εκτίμηση…….....102

12. IceScrum

12.1 Η Ιδέα(Concept)……...107
12.2 Βιωσιμότητα, Υποστήριξη & Τεκμηρίωση……107
12.3 Ευχρηστία……108
12.4 Πλεονεκτήματα……108
12.5 Αδυναμίες…… 108
12.6 Γενική εκτίμηση…… 108

13. Agilo

13.1 Η Ιδέα(Concept)……...113
13.2 Βιωσιμότητα, Υποστήριξη & Τεκμηρίωση……113
13.3 Ευχρηστία……114
13.4 Πλεονεκτήματα……114
13.5 Αδυναμίες…… 114
13.6 Γενική εκτίμηση……114

14. eXplainPMT

14.1 Η Ιδέα(Concept)……...120
14.2 Βιωσιμότητα, Υποστήριξη & Τεκμηρίωση……120
14.3 Ευχρηστία…… ……120
14.4 Πλεονεκτήματα……120
14.5 Αδυναμίες……121
14.6 Γενική εκτίμηση……121

15. XPlanner

15.1 Η Ιδέα(Concept)…… …… 125
15.2 Βιωσιμότητα, Υποστήριξη & Τεκμηρίωση……125
15.3 Ευχρηστία…… 125
15.4 Πλεονεκτήματα……126
15.5 Αδυναμίες…… 126
15.6 Γενική εκτίμηση……126
15.7 Τελικό Συμπέρασμα ……130

Επίλογος……131

Βιβλιογραφία – References…… ……132

Πίνακας Σχημάτων-Πινάκων

Σχήμα 1: Η διαδικασία του Κύκλου Ζωής..... 14
Σχήμα 2: Η δομή της Διαδικασίας 15
Σχήμα 3: Εξωτερικό/Εσωτερικό μοντέλο ποιότητας του προτύπου ISO/IEC 9126-1.....18
Σχήμα 4: Διαφορές μεταξύ ευέλικτων και παραδοσιακών μεθόδων.....24
Σχήμα 5: Οι 4 κύριες διαδικασίες στην ευέλικτη μεθοδολογία.....26
Σχήμα 6: Μοντέλο Lego.....55
Σχήμα 7: Μοντέλο 5-Α.....58
Σχήμα 8: Κύκλος ζωής της Scrum..... 67
Σχήμα 9: Δείγμα Λίστα Προϊόντος.....81
Σχήμα 10: Δείγμα Ταχείας Λίστας..... 81
Σχήμα 11: Ταχεία Λίστα-Καμία εργασία δεν πραγματοποιήθηκε.....82
Σχήμα 12: Ταχεία Λίστα μετά από εργασία που έχει γίνει.....82
Σχήμα 13: Καταγραφή διαγράμματος μετά από εργασία που έχει πραγματοποιηθεί..... 83
Σχήμα 14: Διάγραμμα μετά από εργασία που έχει πραγματοποιηθεί αλλά όχι αρκετά γρήγορα..... 84
Σχήμα 15: Διάγραμμα μετά από εργασία που έχει πραγματοποιηθεί αλλά πολύ γρήγορα.....85
Σχήμα 16: Κέρδος και απώλειες χρησιμοποιώντας τον παραδοσιακό τρόπο ανάπτυξης..... 86
Σχήμα 17: Κέρδος και απώλειες χρησιμοποιώντας την επαναληπτική ανάπτυξη.....87
Σχήμα 18: Ολοκληρωμένο μοντέλο ποιότητας.....91
Σχήμα 19: Ευέλικτες μέθοδοι και Διασφάλιση Ποιότητας(QA)..... 92
Σχήμα 20: SQA Χρονοδιάγραμμα..... 95
Πίνακας 1: Ποιοτικά χαρακτηριστικά ορισμένα από ISO/IEC 9126-1..... 17
Πίνακας 2: Παράμετροι Ποιότητας Λογισμικού στις Ευέλικτες Τεχνικές..... 31
Πίνακας 3: Αξίες στην παραδοσιακή SPI και στα ευέλικτα πρότυπα.....35
Πίνακας 4: Ποιοτικοί παράγοντες στις βασικές αρχές και αξίες της SPI και των ευέλικτων προτύπων.....36
Πίνακας 5: Πρακτικές και πιθανές βελτιώσεις..... 65
Πίνακας 6: Οι βασικές Ερωτήσεις της Scrum 72
Πίνακας 7: Σύγκριση πρακτικών και προϊόντων της Μεθόδου Καταρράκτη με την Scrum..... 72
Πίνακας 8: Συνολική Σύγκριση Ευέλικτων Εργαλείων..... 97
Εικόνα 1έως Εικόνα 9: Οθόνες του προγράμματος Agilefant 102-106
Εικόνα 10 έως Εικόνα 17: Οθόνες του προγράμματος IceScrum.....109-112
Εικόνα 18 έως Εικόνα 26: Οθόνες του προγράμματος Agilo.....115-119
Εικόνα 27 έως Εικόνα 33: Οθόνες του προγράμματος eXplainPMT.... 121-124
Εικόνα 34 έως Εικόνα 42: Οθόνες του προγράμματος XPlanner.....126-130

Εισαγωγή

Σκοπός αυτής της πτυχιακής είναι η αξιολόγηση των προτύπων και πρακτικών για την διασφάλιση της ποιότητας στα ευέλικτα έργα.

Για να μπορέσει αυτό να γίνει εφικτό πρέπει να αναφέρουμε τι είναι ποιότητα και πως αυτή ορίζεται. Έτσι στο πρώτο κεφάλαιο παρουσιάζονται κάποιο ορισμοί της ποιότητας και αναφέρονται τα πρότυπα ISO/IEC 12207, όπου παρουσιάζονται οι βασικές αρχές του, και το ISO/IEC 9126 όπου και σε αυτό αναφέρονται τα χαρακτηριστικά του. Στη συνέχεια προσπαθούμε να ξεκαθαρίσουμε ορισμένες αμφισβητούμενες έννοιες στο πρότυπο ISO 9126-1 και αναφέρουμε τα πιο ανακριβή σημεία του.

Στη συνέχεια στο δεύτερο κεφάλαιο αναφερόμαστε γενικά στις ευέλικτες μεθόδου ελέγχου ποιότητας και την αξιοπιστία λογισμικού δίνοντας τους διάφορους ορισμούς που υπάρχουν. Γίνεται προσπάθεια για την εκτίμηση της ποιότητας στις ευέλικτες μεθόδους και μελετάμε τις τεχνικές που σύμφωνα με τις διάφορες παραμέτρου μπορούν να εφαρμοστούν για να πετύχουμε κάτι τέτοιο. Έπειτα γίνεται μια αντιπαραβολή των διαδικασιών ποιότητας της παραδοσιακής ανάπτυξης λογισμικού με την ευέλικτη ανάπτυξη λογισμικού.

Στο τρίτο κεφάλαιο εξετάζουμε μια από τις πιο σημαντικές μεθόδους της ευέλικτης ανάπτυξης την Extreme Programming και βλέπουμε διάφορα χαρακτηριστικά της. Κάνουμε μια σύγκριση της XP με διάφορα άλλα πρότυπα και προσπαθούμε να γεφυρώσουμε το χάσμα που υπάρχει σχετικά με την τεκμηρίωση. Τέλος εξετάζουμε και αναλύουμε τις διάφορες πρακτικές της.

Στο τέταρτο κεφάλαιο κάνουμε αναφορά στον κύκλο ζωής των έργων που αναπτύσσουν λογισμικά σύμφωνα με την μέθοδο Extreme Programming. Κάθε ένα από τα 6 αυτά στάδια αναλύονται για την καλύτερη κατανόηση τους.

Αμέσως μετά στο πέμπτο κεφάλαιο κάνουμε μια μικρή αναφορά και στην μέθοδο Pair Programming εξηγώντας την φιλοσοφία αυτής της μεθόδου και σε ποιους άτυπους κανόνες πρέπει να υπακούν τα μέλη που την χρησιμοποιούν ώστε η συνεργασία τους να είναι ικανοποιητική και να υπάρχει ένα καλό αποτέλεσμα στο προϊόν το οποίο θα κατασκευάσουν.

Στη συνέχεια στο έκτο κεφάλαιο αναφερόμαστε στο θεωρητικό πλαίσιο του 5-A μοντέλου για το πώς δημιουργείτε η γνώση αναλύοντας τους 5 τρόπους που υπάρχουν για να πετύχουμε κάτι τέτοιο μιας και είναι ένα σημαντικό κομμάτι στον συνεχώς εξελισσόμενο κόσμο της τεχνολογίας.

Στο έβδομο κεφάλαιο γίνεται ανάλυση των πρακτικών της Extreme Programming και κάθε μια από αυτές αναλύεται αφού πρώτα έχει κατηγοριοποιηθεί σύμφωνα με το στάδιο το οποίο μπορεί να χρησιμοποιηθεί.

Στη συνέχεια όμως και στο κεφάλαιο οκτώ δεν ξεχνάμε την μέθοδο Scrum όπου εκεί γίνεται αναφορά στα 6 βασικότερα πλεονεκτήματα της χρήσης της. Απαντάμε στο ερώτημα τι είναι Scrum και κάνουμε μια αντιπαράθεση των χαρακτηριστικών της Scrum με την παραδοσιακή μέθοδο Καταρράκτη. Έπειτα ακολουθεί ένας πίνακας που δείχνει την διαχείριση ενός έργου σύμφωνα με τις δυο μεθόδους. Και τέλος γίνεται μια παρουσίαση και των διαφόρων τεχνικών που χρησιμοποιεί αυτή η μέθοδος.

Αμέσως μετά στο ένατο κεφάλαιο παρουσιάζουμε την διασφάλιση της ποιότητας, τι είναι και γιατί είναι σημαντική. Κάνουμε μια σύγκριση πάλι της διασφάλισης της ποιότητας μεταξύ του μοντέλου καταρράκτη και των ευέλικτων μεθόδων και αναφερόμαστε στις τεχνικές της διασφάλισης της ποιότητας. Παρουσιάζουμε την διασφάλιση της ποιότητας στις ευέλικτες μεθόδους και τις τεχνικές που πρέπει να ακολουθηθούν για να πετύχουμε κάτι τέτοιο.

Και στο κλείσιμο και στα κεφάλαια δέκα έως δεκαπέντε παρουσιάζουμε τον σχεδιασμό και κάποια εργαλεία που χρησιμοποιούνται για την ανάπτυξη λογισμικού σύμφωνα με τις ευέλικτες μεθόδους δίνοντας κάποια χαρακτηριστικά τους ομαδοποιώντας τα σε πλεονεκτήματα και μειονεκτήματα και παρουσιάζοντας έναν γενικό πίνακα συγκρίνοντας αυτά τα χαρακτηριστικά.

1. Ποιότητα και Πρότυπα διασφάλισης Ποιότητας

1.1Τι είναι η ποιότητα;

Παρόλο που η ποιότητα λογισμικού[1] είναι κρίσιμη για την επιτυχία στην παραγωγή ενός λογισμικού σαν έννοια είναι δύσκολο να το προσδιορίσει, να το περιγράψει, να το κατανοήσει και να το μετρήσει κάποιος.

Η ποιότητα λοιπόν σύμφωνα με το ISO 8402 είναι: ***Το σύνολο των χαρακτηριστικών ενός προϊόντος ή μιας υπηρεσίας που αφορούν την δυνατότητα του να μπορεί να ικανοποιήσει τις δηλωμένες και αυτές που υπονοούνται ανάγκες***. Το Ίδρυμα Ηλεκτρικών και Μηχανικών Ηλεκτρονικής(The Institute of Electrical and Electronics Engineers IEEE) όρισε την ποιότητα σαν: ***Ο βαθμός στον οποίο ένα σύστημα, ένα μέρος αυτού ή οι διαδικασίες συναντούν συγκεκριμένες απαιτήσεις και ανάγκες ή προσδοκίες πελατών/χρηστών***. Και οι δύο ορισμοί συγκεντρώνονται στην ικανοποίηση των αναγκών του πελάτη για το λογισμικό.

Για να αντιμετωπιστεί το ζήτημα της διαδικασίας λογισμικού και της ποιότητας του προϊόντος στις ευέλικτες μεθόδους, θα πρέπει να λάβουμε υπόψη 2 καλά ορισμένα βιομηχανικά πρότυπα, το ISO/IEC 12207 και το ISO/IEC 9126 αντίστοιχα.

Ο διεθνής οργανισμός για τα πρότυπα(ISO International Organization for Standardization)[2] έχει ορίσει ένα σύνολο από ISO και ISO/IEC(International Electronical Commission) πρότυπα για την ποιότητα του λογισμικού. Πρώτα απ' όλα θα έπρεπε να αναφέρουμε το ISO 9000-3 που είναι η κατευθυντήρια οδηγία για να ισχύσει το ISO 9001 πρότυπο το οποίο σχετίζεται με την διασφάλιση της ποιότητας των διαδικασιών για την ανάπτυξη, τροφοδότηση, εγκατάσταση και διατήρηση του λογισμικού. Συνεπώς το πρότυπο ISO/IEC 9126 για την ποιότητα του προϊόντος, πρέπει να συνδυαστεί με το ISO/IEC 14598 για την αξιολόγηση των λογισμικών προϊόντων. Άλλα πρότυπα που μπορούν να συσχετιστούν ή μπορούν να συνδυαστούν με το ISO/IEC 9126 και το ISO/IEC 14598 είναι:

- ISO/IEC 12119 – Ποιοτικές απαιτήσεις για πακέτα λογισμικού.
- ISO/IEC 12207 – Διαδικασίες του κύκλου ζωής του λογισμικού.
- ISO/IEC 14143 – Μετρικές λογισμικού
- ISO/IEC 15271 – Οδηγία για το ISO/IEC 12207
- ISO/IEC 15504- Εκτίμηση των διαδικασιών ενός λογισμικού(επίσης γνωστό και ως Spice)
- ISO/IEC 15939 – Μετρικές διαδικασιών λογισμικού.

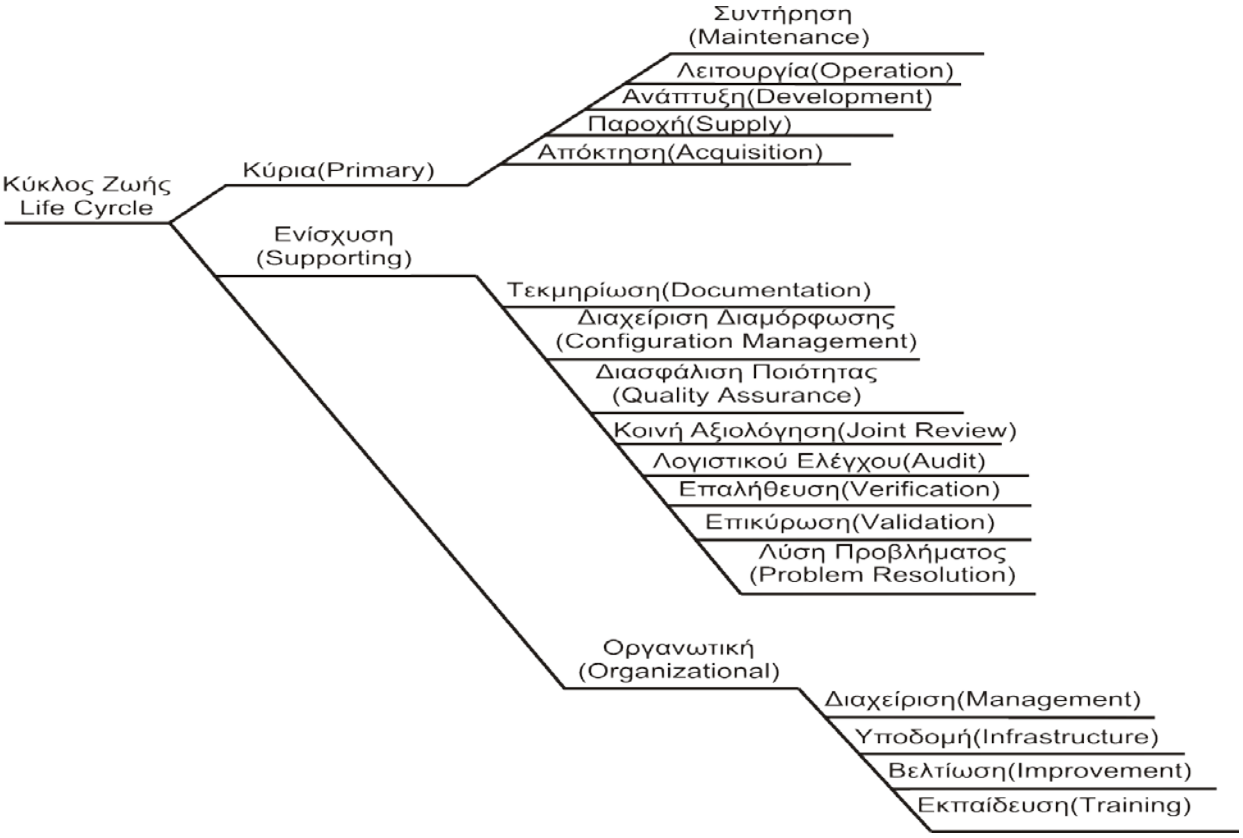
1.2 ISO/IEC 12207

Το ISO/IEC 12207 πρότυπο [5] παρέχει ένα πλαίσιο για τον κύκλο ζωής των διαδικασιών ενός λογισμικού. Εστιάζουμε μόνο στον τομέα των διαδικασιών ανάπτυξης σε αυτό το πρότυπο διότι οι περισσότερες ευέλικτες πρακτικές μπορούν να απεικονιστούν απευθείας στις δραστηριότητες αυτού του τομέα των διαδικασιών. Ο σχεδιασμός της τακτικής(Planning game) και ο προγραμματισμός ροής(sprint planning) περιλαμβάνουν ενέργειες που μπορούν να απεικονιστούν σε δραστηριότητες για τον ορισμό των απαιτήσεων στον τομέα της διαδικασίας ανάπτυξης, ενώ ο προγραμματισμός σε ζεύγη(pair programming) και οι συνεχείς ενσωματώσεις(continuous integration) ανήκουν στις δραστηριότητες εφαρμογής και δοκιμής αυτού του τομέα.

Το ISO 12207 [6] καθιερώνει μια διαδικασία για τον κύκλο ζωής του λογισμικού, περιέχει διαδικασίες και δραστηριότητες που εφαρμόζονται κατά την διάρκεια της απόκτησης και διαμόρφωσης των παροχών του συστήματος. Κάθε διαδικασία έχει ένα σύνολο από αποτελέσματα που συνδέονται με αυτή. Υπάρχουν 23 Διαδικασίες, 95 Δραστηριότητες, 325 Εργασίες, 224 Αποτελέσματα.

Το πρότυπο έχει σαν κύριο αντικειμενικό στόχο την υποστήριξη μιας κοινής δομής έτσι ώστε οι αγοραστές, οι προμηθευτές, οι κατασκευαστές, οι χρήστες, οι διευθυντές και οι τεχνικοί που συμμετέχουν στην ανάπτυξη ενός λογισμικού να χρησιμοποιούν μια κοινή γλώσσα. Αυτή η κοινή γλώσσα καθιερώθηκε στη μορφή μιας καλά ορισμένης διαδικασίας. Η δομή του προτύπου επρόκειτο να συλληφθεί σε ένα εύκαμπτο, τμηματικό τρόπο ώστε να είναι προσαρμόσιμη στις ανάγκες όποιου το χρησιμοποιεί. Το πρότυπο έχει βασιστεί σε 2 βασικές αρχές: της τμηματοποίησης(modularity) και της ευθύνης(responsibility).

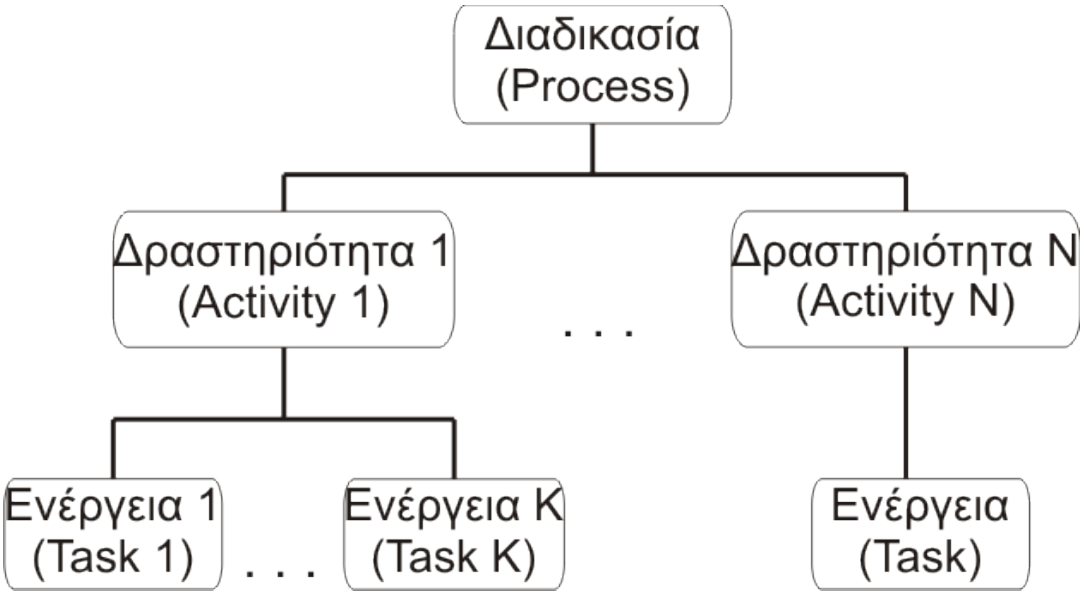
Οι διαδικασίες του κύκλου ζωής[5]: Οι διαδικασίες ομαδοποιούνται σε τρεις ευρείς κλάσεις: την κύρια(primary), της ενίσχυσης(supporting) και της οργανωτικής(organizational) Οι κύριες διαδικασίες είναι οι πρωταρχικοί υποκινητές στο κύκλο ζωής, είναι απόκτηση(acquisition), παροχή(supply), ανάπτυξη(development), λειτουργία(operation), συντήρηση(maintenance). Οι ενισχυτικές διαδικασίες είναι τεκμηρίωσης(documentation), διαχείρισης διαμόρφωσης(configuration management), διασφάλισης ποιότητας(quality assurance), κοινής αξιολόγησης(joint review), λογιστικού ελέγχου(audit), επαλήθευσης(verification), επικύρωσης(validation), και λύσης προβλήματος(problem resolution). Η ενισχυτική διαδικασία υποστηρίζει μια άλλη διαδικασία για την εκτέλεση μιας εξειδικευμένης λειτουργίας. Οι οργανωτικές διαδικασίες είναι διαχείρισης(management,), υποδομής(infrastructure), βελτίωσης(improvement,) και εκπαίδευσης(training). Ένας οργανισμός μπορεί να υιοθετήσει μια οργανωτική διαδικασία για να καθιερώσει ελέγξει, και βελτιώσει την διαδικασία του κύκλου ζωής.



Σχήμα 1: Η διαδικασία του Κύκλου Ζωής

Η δομή της διαδικασίας του κύκλου ζωής[5]: Κάθε διαδικασία είναι περεταίρω σχεδιασμένη από την άποψη των συστατικών δραστηριοτήτων της, κάθε μια από τις οποίες είναι περεταίρω σχεδιασμένη από την άποψη των εργασιών της. Μια δραστηριότητα κάτω από μια διαδικασία είναι ένα σύνολο από ενιαίες εργασίες.

Φύση των εργασιών[5]. Μια εργασία είναι ένα σύνολο από στοιχειώδες ή ατομικές ενέργειες. Μια εργασία καταναλώνει τις εισόδους(δεδομένων, πληροφοριών, ελέγχου) και παράγει εξόδους(δεδομένων, πληροφοριών, ελέγχου). Εκφράζεται υπό την μορφή της αυτό-δήλωσης, απαίτησης, σύστασης ή επιτρεπόμενης ενέργειας. Για αυτό τον σκοπό, το πρότυπο χρησιμοποιεί προσεκτικά συγκεκριμένα βοηθητικά ρήματα(θα, πρέπει και μπορεί) για να υπάρχει διαφοροποιήσει μεταξύ των ευδιάκριτων μορφών μιας εργασίας. Το “ΘΑ” χρησιμοποιείται για να εκφράσει μια αυτό-δήλωση του σκοπού ή της πρόθεσης από ένα συμβαλλόμενο κομμάτι, το “ΘΑ ΠΡΕΠΕΙ” για να εκφράσει μια υπόδειξη μεταξύ άλλων πιθανοτήτων και το ”ΜΠΟΡΕΙ” για να δείξει ένα σχέδιο δράσης επιτρεπόμενο μέσα στα όρια του προτύπου. Το “ΠΡΕΠΕΙ” που υποδηλώνει μια υποχρεωτική ενέργεια δεν χρησιμοποιείται ποτέ στο πρότυπο.



Σχήμα 2: Η δομή της Διαδικασίας

Η φύση της αξιολόγησης[5]: Στο πρότυπο αυτό η αξιολόγηση είναι η στοιχειώδη λειτουργία και χρησιμοποιείται με αρκετούς τρόπους κατά την διαδικασία. Οι αξιολογήσεις πραγματοποιούνται στις διάφορες οντότητες με τους δοσμένους σκοπούς σε σύγκριση με τα δοσμένα κριτήρια. Μια οντότητα μπορεί να είναι μια διαδικασία, μια δραστηριότητα ή μια εργασία, ένα σχέδιο, μια συμφωνία ή μια αναφορά, ένα δεδομένο, μια πληροφορία ή ένα προϊόν ή άλλα. Ένας σκοπός μπορεί να ελέγξει, να αναθεωρήσει, να παρακολουθήσει, να επιβεβαιώσει, να επικυρώσει ή να βελτιώσει, όπου το κίνητρο και ο αντικειμενικός στόχος πίσω από τις αντίστοιχες αξιολογήσεις διαφέρουν στην πράξη. Ένα κριτήριο μπορεί να είναι για παράδειγμα η ανιχνευσιμότητα του σχεδίου στις απαιτήσεις του ή η ορθότητα του σχεδίου.

Συνολική διαχείριση της ποιότητας[5]: Το πρότυπο εφαρμόζει τις αρχές της συνολικής διαχείρισης της ποιότητας.

Για αρχή τα πρότυπα μεταχειρίζονται όλες τις δραστηριότητες συμπεριλαμβανομένου και αυτές που σχετίζονται με την ποιότητα σαν ένα ζωτικής σημασίας κομμάτι του κύκλου ζωής του λογισμικού. Κατά συνέπεια η ποιότητα εξετάζεται αυτόματα από την αρχή.

Σαν επόμενο βήμα[5], οι δραστηριότητες που σχετίζονται με την ποιότητα στον κύκλο ζωής προσαρμόζονται στην κάθε διαδικασία. Κάθε διαδικασία εξοπλίζεται με έναν ενσωματωμένο κύκλο “σχεδίασης, πράξης, ελέγχου, ενέργειας” ("plan-do-check-act" (PDCA) cycle). Κατά συνέπεια κάθε διαδικασία και στο υπεύθυνο πρόσωπο για την εκτέλεση της διαδικασίας ορίζονται σχετικές με αυτή διαδικασίες, εσωτερικές δραστηριότητες συσχετιζόμενες με την ποιότητα, συμπεριλαμβανομένου και των αξιολογήσεων. Επιπλέον δυο ειδικές διαδικασίες, της επαλήθευσης και της επικύρωσης, παρέχονται για να συμπληρώσουν τις προαναφερθείσες διαδικασίες εσωτερικής αξιολόγησης με επιθυμητό βαθμό ανεξαρτησίας και αντικειμενικότητας.

Μια ιδιαίτερη διαδικασία[5], η διασφάλιση της ποιότητας, είναι αφιερωμένη στην διασφάλιση της προσάρμοσης των προϊόντων και των υπηρεσιών με τις ιδιαίτερες απαιτήσεις τους. Τα άτομα που είναι υπεύθυνα για αυτή την διαδικασία παρέχονται με οργανωτική ελευθερία και εξουσία για να επηρεάσουν την προσάρμοση. Οργανωτική ελευθερία σημαίνει η ελευθερία που έχει κάποιος ο οποίος έχει την άμεση ευθύνη της διαχείριση για την παραγωγή του προϊόντος ή της παροχής των υπηρεσιών, ενώ εξουσία σημαίνει η εξουσία του να επιτρέψει κάποιος τις σχετικές αξιολογήσεις και να θέτει σε εφαρμογή τις σχετικές διορθωτικές ενέργειες. Σε αυτή τη διαδικασία δεν παραχωρείται η εξουσία του να επιβάλει αυτές οι διορθωτικές ενέργειες.

Τέλος[5], το πρότυπο παρέχει μια βελτιωμένη διαδικασία για διαχείριση, έλεγχο και βελτίωση των καθιερωμένων διαδικασιών και της απόδοσης τους, πέρα από τις συμφωνημένες δεσμεύσεις.

1.3 ISO/IEC 9126

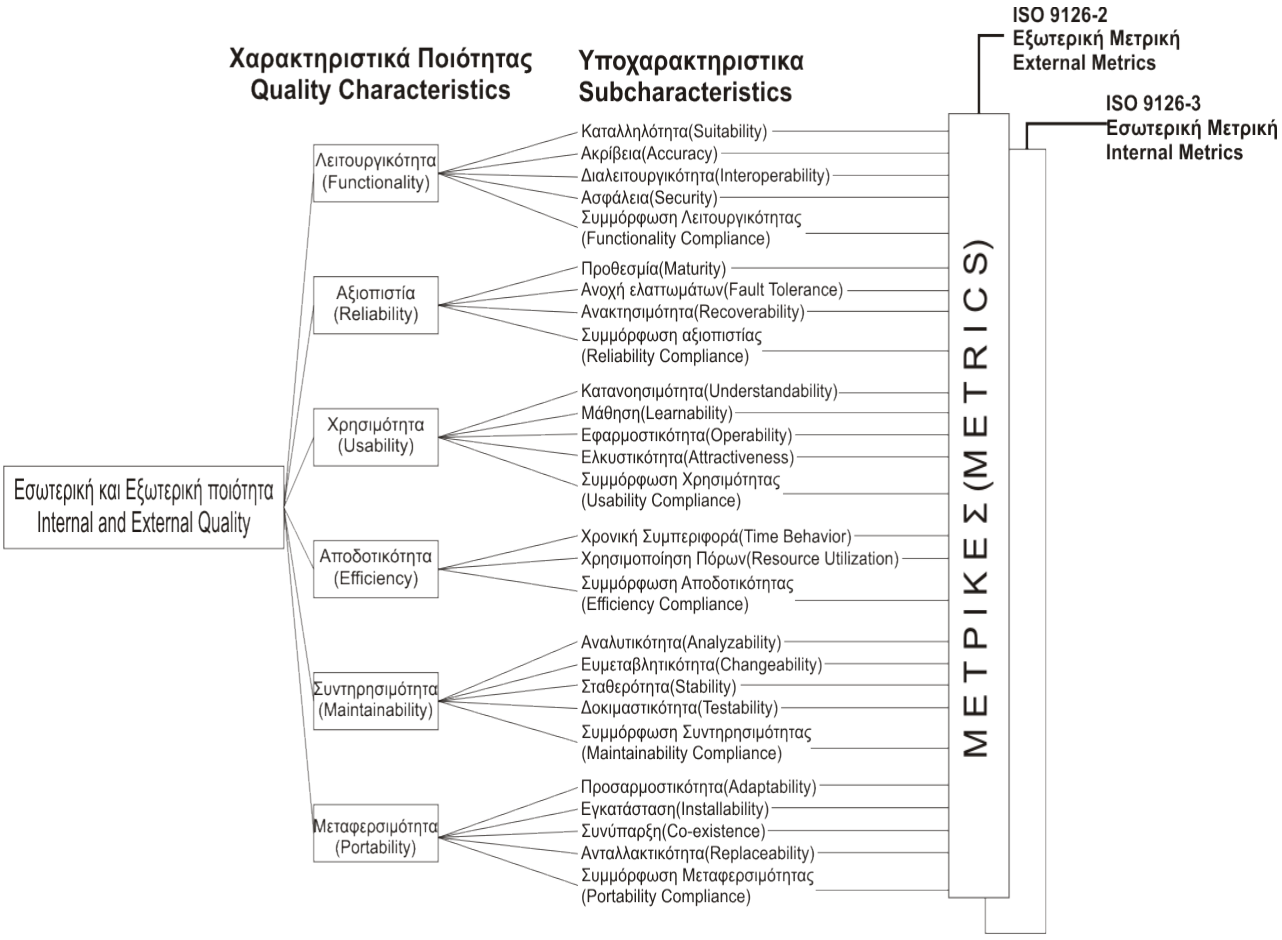
Το ISO/IEC 9126-1 [3] κατευθύνει συγκεκριμένα τον καθορισμό των μοντέλων ποιότητας. Η κύρια ιδέα πίσω από αυτό το πρότυπο είναι ο καθορισμός ενός μοντέλου ποιότητας και η χρήση του για την αξιολόγηση λογισμικού. Ένα μοντέλο ποιότητας σύμφωνα με το ISO/IEC 9126-1 ορίζεται από την έννοια των γενικών χαρακτηριστικών του λογισμικού, τα οποία είναι περεταίρω τελειοποιημένα μέσα στα υποχαρακτηριστικά, τα οποία με την σειρά τους αποσυντίθεται σε ιδιότητες, που παράγουν μια πολυεπίπεδη ιεραρχία, μπορεί να εμφανιστούν ενδιάμεσες ιεραρχία από τα e υποχαρακτηριστικά και τις ιδιότητες. Στο τέλος της ιεραρχίας εμφανίζεται η αξιολόγηση των ιδιοτήτων του λογισμικού, της οποίας η αξία υπολογίζεται από μετρικές. Για την διευκόλυνση μας στη συνέχεια θα αναφερόμαστε στα χαρακτηριστικά, στα υποχαρακτηριστικά και στις ιδιότητες ως παράγοντες ποιότητας. Οι ποιοτικές απαιτήσεις μπορεί να ορίζονται σαν περιορισμοί στο μοντέλο ποιότητας. Η 2001 έκδοση του προτύπου παρουσιάζει τα υποχαρακτηριστικά σαν πρότυπα, βασιζόμενο στα ενημερωτικά υποχαρακτηριστικά που παρουσιάζονται στη 1991 έκδοση.

Το ISO/IEC 9126 [2] πρότυπο κάνει μια διάκριση μεταξύ εσωτερικής ποιότητας και εξωτερικής ποιότητας, και παρουσιάζει το λεγόμενο ποιότητα σε χρήση(quality in use). Αυτά τα μοντέλα κατηγοριοποιούν τις ποιοτικές ιδιότητες του λογισμικού σε χαρακτηριστικά. Οι ιδιότητες που μπορούν να μετρηθούν κατά την διάρκεια της διαδικασίας ανάπτυξης αναφέρονται σαν εσωτερικές. Η εξωτερική συμπεριφορά μπορεί να μετρηθεί κατά την διάρκεια της διαδικασίας δοκιμών, και τέλος η άποψη του χρήση για την ποιότητα παρουσιάζεται από την μέτρηση των ιδιοτήτων της ποιότητας σε χρήση.

Το μοντέλο ποιότητας που εισάγεται στο πρότυπο είναι κοινό για την εξωτερική και εσωτερική ποιότητα(παρόλο που θα καθοριστούν με διαφορετικούς τρόπου), ενώ ένα άλλο μοντέλο για την ποιότητα σε χρήση θα εισαχθεί. Παρακάτω βλέπουμε την απαρίθμηση των 6 ποιοτικών χαρακτηριστικών ορισμένα στο ISO/IEC 9126-1 εσωτερικό/εξωτερικό μοντέλο ποιότητας και τον διαχωρισμό τους σε υποχαρακτηριστικά. Μόνο για να αναφερθεί το μοντέλο ποιότητας για την ποιότητας σε χρήση χωρίζεται σε 4 κατηγορίες: Αποτελεσματικότητα(Effectiveness), Παραγωγικότητα(Productivity), Ασφάλεια(Safety) και Ικανοποίηση(Satisfaction).

Πίνακας 1:Ποιοτικά χαρακτηριστικά ορισμένα από ISO/IEC 9126-1

Χαρακτηριστικά(Characteristics)	Υποχαρακτηριστικά(Subcharacteristic)
Λειτουργικότητα(Functionality)	Καταλληλότητα(Suitability)
	Ακρίβεια(Accuracy)
	Διαλειτουργικότητα(Interoperability)
	Ασφάλεια(Security)
	Συμμόρφωση Λειτουργικότητας (Functionality Compliance)
Αξιοπιστία(Reliability)	Προθεσμία(Maturity)
	Ανοχή ελαττωμάτων(Fault Tolerance)
	Ανακτησιμότητα(Recoverability)
	Συμμόρφωση αξιοπιστίας (Reliability Compliance)
Χρησιμότητα(Usability)	Κατανοησιμότητα(Understandability)
	Μάθηση(Learnability)
	Εφαρμοστικότητα(Operability)
	Ελκυστικότητα(Attractiveness)
	Συμμόρφωση Χρησιμότητας (Usability Compliance)
Αποδοτικότητα(Efficiency)	Χρονική Συμπεριφορά(Time Behavior)
	Χρησιμοποίηση Πόρων(Resource Utilization)
	Συμμόρφωση Αποδοτικότητας (Efficiency Compliance)
Συντηρησιμότητα(Maintainability)	Αναλυτικότητα(Analyzability)
	Ευμεταβλητικότητα(Changeability)
	Σταθερότητα(Stability)
	Δοκιμαστικότητα(Testability)
	Συμμόρφωση Συντηρησιμότητας (Maintainability Compliance)
Μεταφερσιμότητα(Portability)	Προσαρμοστικότητα(Adaptability)
	Εγκατάσταση(Installability)
	Συνύπαρξη(Co-existence)
	Ανταλλακτικότητα(Replaceability)
	Συμμόρφωση Μεταφερσιμότητας (Portability Compliance)



Σχήμα 3: Εξωτερικό/Εσωτερικό μοντέλο ποιότητα του προτύπου ISO/IEC 9126-1

1.4 Καθορισμός των αμφισβητούμενων εννοιών στο ISO/IEC 9126-1[2]

Η έννοια του μοντέλου ποιότητας ορίζεται με διαφορετικούς τρόπους από διαφορετικούς οργανισμούς και συγγραφείς. Άλλωστε ακόμα και ένας συγκεκριμένος ορισμός μπορεί να περιέχει ασάφειες και (και μερικές φορές σκόπιμες) ατέλειες οι οποίες γίνονται προφανείς μόλις το μοντέλο ποιότητας πρέπει να κατασκευαστεί. Πιστεύουμε ότι οι απολαβές από τις επενδύσεις μπορούν να αυξηθούν με τον καθορισμό αυτών των ασαφειών και ατελειών σε υψηλότερο βαθμό.

Αυτό συμβαίνει στο ISO/IEC 9126-1. Όπως αναφέρθηκε πριν το πρότυπο διατυπώνει τρείς προσεγγίσεις της ποιότητας(εσωτερική ποιότητα, εξωτερική ποιότητα και ποιότητα σε χρήση). Εσωτερική και εξωτερική ποιότητα μοιράζονται μια λίστα με έξι υψηλού επιπέδου παράγοντες ή χαρακτηριστικά (Λειτουργικότητα (Functionality), Αξιοπιστία (Reliability), Χρησιμότητα (Usability), Αποδοτικότητα (Efficiency), Συντηρησιμότητα (Maintainability), Μεταφερσιμότητα(Portability)) που κάθε μια από αυτές διαιρείται σε μερικά υποχαρακτηριστικά. Η ποιότητα σε χρήση καθορίζεται από ένα σύνολο τεσσάρων χαρακτηριστικών (Αποτελεσματικότητα (Effectiveness), Παραγωγικότητα (Productivity), Ασφάλεια (Safety) και Ικανοποίηση (Satisfaction)). Το πρότυπο επίσης ορίζει σαν τελικό στόχο τον καθορισμό των χαμηλότερων επιπέδων παραγόντων ή ιδιοτήτων μαζί με τις μετρικές που χρησιμοποιούνται για την μέτρηση. Ωστόσο το πρότυπο δεν είναι ακριβές σε κανένα από τα παρακάτω σημεία.

- *Επιτρέπεται η ιεράρχηση των υποχαρακτηριστικών και των ιδιοτήτων?* Σύμφωνα με τα προηγούμενα έχουμε αποφασίσει να το επιτρέπουμε, για τον μη περιορισμό από την άποψη των ιεραρχιών. Από την μια μεριά τα υποχαρακτηριστικά μπορούν να αποσυντίθενται σε άλλα υποχαρακτηριστικά ή εναλλακτικά σε ιδιότητες. Από την άλλη μεριά, υπάρχουν οι ιδιότητες που μπορούν να παραχθούν και υπάρχουν και οι βασικές. Η παραγωγή ιδιοτήτων μπορεί να τις αποσυνθέσει σε άλλες ιδιότητες , όπου οι βασικές ιδιότητες δεν μπορούν.
- *Μπορούν τα υποχαρακτηριστικά και οι ιδιότητες να συσχετιστούν με παραπάνω από έναν παράγοντες ποιότητας στα υψηλότερα επίπεδα ιεράρχησης? Ή πρέπει να σχηματίζουν μια τέλεια ιεραρχία?* Σύμφωνα με τα προηγούμενα αποφασίσαμε να περιορίσουμε την ιεραρχία των χαρακτηριστικών και των υποχαρακτηριστικών σε μια απόλυτη μορφή ιεραρχίας. Ωστόσο επιτρέπουμε την επικάλυψη στην ιεραρχία των ιδιοτήτων και των σχέσεων τους με τα υποχαρακτηριστικά. Χειριζόμαστε με διαφορετικό τρόπο τις ιδιότητες από τότε που συγκεκριμένες ιδιότητες μπορούν να συνδυαστούν με πάνω από ένα υποχαρακτηριστικά.

  •*Πρέπει τα χαρακτηριστικά και τα υποχαρακτηριστικά να έχουν ορισμένες μετρικές για την μέτρηση τους?* Σύμφωνα με τα προηγούμενα, τα χαρακτηριστικά είναι μη μετρήσιμοι παράγοντες και χρησιμοποιούνται μόνο για να διευκρινίσουν τα ανώτερα επίπεδα των υποχαρακτηριστικών. Αντίθετα επιτρέπουμε να μετρηθούν τα υποχαρακτηριστικά όπου χρειάζεται. Παρόλα αυτά σε αυτή την περίπτωση δεν είναι δυνατόν να αναθέσουμε σε αυτά αντικειμενικές μετρικές(πχ. επίσημοι και ακριβείς τρόποι να μετρηθούν από την άποψη των τιμών που παίρνουν τα υποχαρακτηριστικά ή οι ιδιότητες στις οποίες αποσυντίθενται), πιστεύουμε ότι μπορούμε να τις δώσουμε υποκειμενικές μετρικές(πχ. οι τρόποι να μετρηθούν εξαρτώνται από την αντίληψη των ανθρώπων που συμμετέχουν στην διαδικασία μέτρησης).

•*Στην περίπτωση που έχουμε ιεράρχηση των ιδιοτήτων, μετριούνται όλες οι ιδιότητες με τον ίδιο τρόπο?* Σύμφωνα με τα προηγούμενα έχουμε τις παραγόμενες και τις βασικές ιδιότητες. Όλες είναι μετρήσιμοι παράγοντες ποιότητας. Συγκεκριμένα μετρικές για τις βασικές ιδιότητες πρέπει να είναι αντικειμενικές, από τι στιγμή που πρέπει να μετρηθούν με έναν επίσημο και ακριβή τρόπο ανεξάρτητα από τις σκέψεις και την αντίληψη των ανθρώπων που συμμετέχουν στην μέτρηση. Για τις παραγόμενες ιδιότητες μερικές φορές δεν είναι δυνατόν να βρεθεί μια αντικειμενική μετρική ώστε να μετρηθούν από την άποψη των τιμών των ιδιοτήτων στις οποίες έχουν αποσυντεθεί. Σε αυτή την περίπτωση απαιτείται μια υποκειμενική μετρική.

Οι ορισμοί των διαφορετικών τύπων των ποιοτικών παραγόντων που συμπεραίνονται από τις παραπάνω δηλώσεις είναι οι εξής:

- Τα χαρακτηριστικά σε ένα μοντέλο ποιότητας είναι μη-μετρήσιμοι παράγοντες ποιότητας που χρησιμοποιούνται με στόχο την ταξινόμηση των υποχαρακτηριστικών στα ανώτερα επίπεδα του μοντέλου.
- Τα υποχαρακτηριστικά σε ένα μοντέλο ποιότητας είναι παράγοντες ποιότητας που μπορεί υποκειμενικά να μετρηθούν όταν απαιτείται, και μπορεί να αποσυντίθενται σε άλλα υποχαρακτηριστικά ή εναλλακτικά σε ιδιότητες που βοηθούν στη μέτρηση τους. Ένα χαρακτηριστικό και τα υποχαρακτηριστικά του έχουν σαν αποτέλεσμα μια τέλεια ιεραρχία.
- Οι ιδιότητες μπορεί να είναι παραγόμενες ή βασικές. Οι ιδιότητες μπορούν να σχετίζονται με περισσότερα από έναν παράγοντες ποιότητας στα ανώτερα επίπεδα της ιεραρχίας.
- Οι βασικές ιδιότητες είναι αντικειμενικά μετρήσιμοι παράγοντες ποιότητας που δεν μπορούν να αποσυντεθούν παραπάνω.
- Οι παραγόμενες ιδιότητες είναι μετρήσιμοι παράγοντες ποιότητας που μπορούν να αποσυντεθούν σε μια οι περισσότερες ιδιότητες. Μερικές φορές δεν είναι δυνατών να βρεθούν αντικειμενικές μετρικές για τον προσδιορισμό των τιμών των ιδιοτήτων στις οποίες αποσυντίθενται. Σε αυτή την περίπτωση μια υποκειμενικά μετρική απαιτείται.

Το ISO/IEC 9126 πρότυπο [4], στοχεύει να διασφαλίσει την ποιότητα όλων των προϊόντων του λογισμικού, καθορίζει τα χαρακτηριστικά και τα υπό-χαρακτηριστικά της ποιότητας του λογισμικού και τις συνδεδεμένες μετρικές. Παρέχει ένα πλαίσιο για τους οργανισμούς ώστε να ορίσουν ένα μοντέλο ποιότητας για ένα λογισμικό ώστε να προσδιορίσουν τις αξίες που στοχεύουν για τις μετρικές ποιότητας. Οι εξωτερικές μετρικές εφαρμόζονται σε λογισμικά τα οποία είναι σε λειτουργία, ενώ οι εσωτερικές μετρικές είναι αυτές οι οποίες δεν στηρίζονται στην εκτέλεση του λογισμικού. Η ποιότητα με την χρήση μετρικών είναι μόνο διαθέσιμη όταν το τελικό προϊόν χρησιμοποιείτε σε πραγματικές συνθήκες

Οι ευέλικτες μέθοδοι[1] προωθούν εξελικτικές αλλαγές μέσα από τις διαδικασίες ανάπτυξης λογισμικού. Στηρίζονται σε ένα σύνολο από τις καλύτερες πρακτικές οι οποίες λαμβάνονται υπόψη για να αυξήσουν την διασφάλιση και τον έλεγχο της

ποιότητας. Μπορεί να ειπωθεί ότι η συλλογή αυτών των καλύτερων πρακτικών διαμορφώνει μια πειθαρχημένη διαδικασία με ενσωματωμένη ποιότητα. Η διασφάλιση της ποιότητας και ο έλεγχος των διαδικασιών ολοκληρώνεται κατά την διάρκεια ολόκληρου του κύκλου ανάπτυξης, από τις απαιτήσεις μέχρι την τελική έκδοση του προϊόντος. Οι ευέλικτες μέθοδοι χτίζουν ποιότητα μέσα στα προϊόντα συνδυάζοντας τις καλύτερες πρακτικές, παρακινώντας μια διαφορετική προοπτική της διαχείρισης της ποιότητας.

Η ευέλικτη[1] ανάπτυξη ολοκληρωτικά επαναπροσδιορίζει την δουλειά της διασφάλιση της ποιότητας, από τα τυπικά καθήκοντα έως τις καθημερινές δραστηριότητες. Οι αξίες δημιουργούνται και η ποιότητα διασφαλίζεται μέσο όλων των σταδίων ανάπτυξης, όταν όλα τα μέρη ενώνονται σε ένα ενιαίο. Οι κατασκευαστές ακολουθούν ένα σύνολο από τις καλύτερες πρακτικές όπως ο προγραμματισμός σε ζεύγη(pair programming) και την ανάπτυξη καθοδηγούμενη από τις δοκιμές(test-driven development), με αποτέλεσμα την παράδοση λογισμικού ανώτερης ποιότητας, γρηγορότερα και με υψηλότερη αποδοχή από τον τελικό χρήστη. Ο προγραμματισμός σε ζεύγη σαν μια έντονη κοινωνική και συνεργατική δραστηριότητα προσφέρει στους κατασκευαστές μοναδικές ικανότητες, εμπειρίες, ιδιοσυγκρασίες και χαρακτήρες. Αυτές οι πρακτικές εξυπηρετούν όπως ένα συνεχές πλάνο σχεδιασμού και τα αποτελέσματα της διαδικασίας αναθεώρησης κώδικα βοηθούν στην μείωση των ελαττωμάτων και την βελτίωση του σχεδιασμού και την ποιότητα του κώδικα. Η ανάπτυξη καθοδηγούμενη από δοκιμές(TDD) ή η ανάπτυξη πρώτα με δοκιμές(TFD) και η επαναπαραγωγή είναι μια επαναληπτική και επαυξητκή προσέγγιση για προγραμματισμό. Στην TDD οι κατασκευαστές γράφουν αυτόματα εκτελέσιμα τεστ(περιπτώσεις δοκιμών) πριν από την εγγραφή του κώδικα που θα δοκιμάσουν. Οι κατασκευαστές κάνουν λεπτομερή σχεδίαση και σκέφτονται για νέες λειτουργίες πριν γράψουν τον κώδικα. Σε συνδυασμό με τα αποδεκτά τεστ, τα οποία χρησιμοποιούνται σαν απαιτήσεις-τεχνάσματα και την επαναπαραγωγή του κώδικα οι κατασκευαστές περιμένουν να πετύχουν υψηλό επίπεδο ποιότητας. Η επαναπαραγωγή είναι ένας πειθαρχημένος δρόμος για να γίνουν μικρές αλλαγές στον πηγαίο κώδικα βελτιώνοντας το σχεδιασμό του χωρίς να αλλάξει η εξωτερική συμπεριφορά του. Κατά την διάρκεια της επαναπαραγωγής οι κατασκευαστές επανακατασκευάζουν τον κώδικα μέσο της εξέτασης του κώδικα και την προσπάθεια μείωσης των λαθών. Οι ευέλικτες μέθοδοι απαιτούν ότι ο πελάτης θα εμπλακεί σε όλα τα στάδια ανάπτυξης, μια πρακτική που προσφέρει προοπτική με την μορφή του υψηλού επιπέδου απαιτήσεων, βασισμένη σε αποδεκτά κριτήρια και εμφανή ικανοποίηση για το τελικό προϊόν. Με την χρήση τέτοιων πρακτικών στην εξελικτική, επαναληπτική και επαυξητική διαδικασία ανάπτυξης, ο χρήστης της επιχείρησης γίνετε ένας δυνατός συνέταιρος στην διασφάλιση της ποιότητας. Στις ευέλικτες μεθόδους η ποιότητα δεν είναι δουλειά ενός μόνον ατόμου, όλοι οι χρήστες της επιχείρησης είναι υπεύθυνοι για να διασφαλίσουν ότι η εφαρμογή έφτασε στο στόχο.

2. Ευέλικτες μέθοδοι ελέγχου ποιότητας και αξιοπιστίας λογισμικού

2.1Εισαγωγή[7]

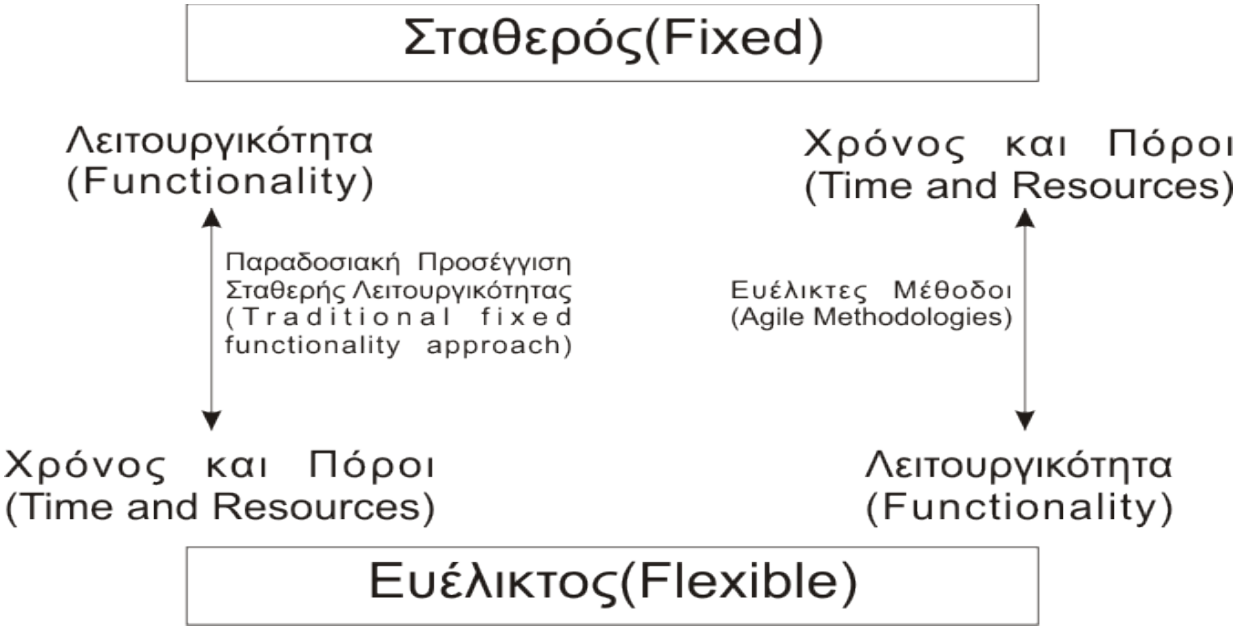
Οι ευέλικτες μέθοδοι άλλαξαν δραστικά την ανάπτυξη λογισμικού. Τα λογισμικά που χρησιμοποιούν ευέλικτες μεθόδους όπως extreme programming, scrum κ.α. βαδίζουν σε καλύτερες πρακτικές οι οποίες ενδιαφέρονται να βελτιώσουν την ποιότητα ανάπτυξης λογισμικών. Μπορούμε να πούμε ότι οι πρακτικές αυτές στοχεύουν στο να επηρεάσουν την Διασφάλιση της Ποιότητας του Λογισμικού(SQA Software Quality Assurance) των project των οποίων βοηθούν. Οι υπερασπιστές των ευέλικτων μεθόδων ισχυρίζονται ότι λόγο της πολύ καλής ουσίας αυτών των μεθόδων, η ποιότητα στα προγράμματα που χρησιμοποιούν ευέλικτες μεθόδους πρέπει να είναι φυσικό επακόλουθο των εφαρμοσμένων μεθόδων. Σαν συνέπεια αυτών τα ευέλικτα προγράμματα ανάπτυξης ασφάλειας ποιότητας(agile software development quality assurance, ASDQA) αναμένονται και υποτίθεται ότι θα είναι λίγο ή πολύ πιο εμπεδωμένα στις διαδικασίες των ευέλικτων προγραμμάτων, καθώς οι πρακτικές της ασφάλειας ανάπτυξης λογισμικού ολοκληρώνονται σε όλη την διαδικασία ανάπτυξης, από τις απαιτήσεις μέχρι την τελική απελευθέρωση. Έτσι οι ευέλικτες μέθοδοι παρουσιάζουν μια διαφορετική όψη στην ασφάλεια της ποιότητας στην ανάπτυξη λογισμικού.

Οι ευέλικτες πρακτικές αναμένεται να χειριστούν ασταθείς και ευμετάβλητες απαιτήσεις κατά την διάρκεια του κύκλου ανάπτυξης, ώστε να παραδώσουν προγράμματα με όσο το δυνατόν λιγότερα ελαττώματα και σφάλματα, σε μικρότερο χρονικό διάστημα, και κάτω από προκαθορισμένους μειωμένους προϋπολογισμούς. Οι επαναληπτικοί και συνεχώς εναλλασσόμενοι τρόποι ανάπτυξης επιτρέπουν και τους μηχανισμούς αναθεώρησης των απαιτήσεων του πελάτη και ενεργή συμμετοχή του πελάτη στην διαδικασία λήψης αποφάσεων. Η συμμετοχή του πελάτη εξασφαλίζει την ανάγκη του μηχανισμού ανατροφοδότησης, εγγυώμενη την ικανοποίηση για συνεχή παρατήρηση του πελάτη μέχρι την ολοκλήρωση του τελικού προϊόντος. Είναι επίσης γνωστό ότι οι ευέλικτες μέθοδοι δημιουργούν χρήστες κλειδιά με πολύ στενή επαφή στην διασφάλισης της ποιότητας. Παρά να αφεθεί η ποιότητα στους επαγγελματίες, τα ευέλικτα προγράμματα έκαναν αυτούς τους χρήστες κλειδιά υπεύθυνους ώστε να διασφαλίσουν ότι η εφαρμογή ταιριάζει με τον σκοπό. Στην ευέλικτη ανάπτυξη χρησιμοποιούνται τα συνεχή τεστ κατά την ανάπτυξη και τεστ του πρώτου σχεδίου, και τα δυο ερχόμενα από το μέτωπο των καλών πρακτικών, φέρνοντας τις ευέλικτες μεθόδους στην κορυφή της ανάπτυξης, ελαχιστοποιώντας τα λάθη και τα ελαττώματα στο τελικό προϊόν. Μερικές άλλες πρακτικές όπως ο απλός σχεδιασμός και ανάπτυξη, ο συνδυαστικός προγραμματισμός, οι περιορισμένοι επαναληπτικοί κύκλοι, οι μικρές απελευθερώσεις, οι συνεχής ενσωματώσεις και τα κοινά κομμάτια κώδικα ενίσχυσαν την ασφάλεια της ποιότητας.

2.2Ορισμός της ευέλικτης μεθόδου[7]

Στην μεθοδολογία των προγραμμάτων ευέλικτης ανάπτυξης δόθηκε το όνομα ευέλικτα όταν ένα group από εργαζόμενους στην ανάπτυξη λογισμικού ένωσαν και δημιούργησαν την Ευέλικτη Συμμαχία(έναν οργανισμό από εργαζόμενους στην ανάπτυξη προγραμμάτων που δημιούργησαν και τυποποίησαν την ευέλικτη μεθοδολογία) τον Φεβρουάριο του 2001. Η ευέλικτη στρατηγική μπορούσε να διακρίνει την εμφάνιση μιας νέας μηχανικής ιεραρχίας που άλλαξε θέση στις σχέσεις των διαδικασιών ανάπτυξης προγραμμάτων από μηχανική(καθοδηγούμενες από διαδικασίες και κανόνες της επιστήμης) σε οργανική(καθοδηγούμενες από τα πιο μικρά θέματα των ανθρώπων και τις αλληλεπιδράσεις τους). Αυτό συνεπάγετε αλλαγές στην μηχανική σύνθεση των λογισμικών συστημάτων στο εργασιακό περιβάλλον τα οποία είναι πολύ δυναμικά και απρόβλεπτα. Το κοινό θέμα σχετικά με τις ευέλικτες μεθόδους είναι ότι όλες εστιάζουν στην προσπάθεια να παράγουν μια λειτουργική λύση και να μπορούν να ανταποκριθούν στις αλλαγές των απαιτήσεων του χρήστη/πελάτη. Σίγουρα οι παραδοσιακές μέθοδοι ανάπτυξης προσπαθούν επίσης να αναπτύξουν λειτουργικές λύσεις, αλλά εστιάζουν στις αλλαγές των απαιτήσεων που η ανομοιότητα του πυρήνα δημιουργεί. Για να το καταλάβουμε ας παρατηρήσουμε το επόμενο διάγραμμα.

Αυτό το διάγραμμα προσπαθεί να διευκρινίσει τις διαφορές των ευέλικτων μεθόδων συγκρινόμενες με τις περισσότερες παραδοσιακές μεθόδους. Αυτές είναι, σε μια ευέλικτη μέθοδο ο διαθέσιμος χρόνος και οι διαθέσιμοι πόροι, πρέπει να υπολογίζονται ώστε να μπορέσουν να καθοριστούν, ενώ η λειτουργικότητα υπολογίζεται σαν κάτι πιο ευπροσάρμοστο. Κατά συνέπεια, ο στόχος είναι συνήθως να οριστεί μια προκαθορισμένη ημερομηνία στην οποία κάτι θα παραδοθεί και θα καθοριστεί η λειτουργικότητας που πρέπει να υλοποιηθεί, έτσι ώστε ότι πρέπει να υλοποιηθεί, να υλοποιηθεί, αλλά με την παραδοχή ότι μπορεί να μην παραδοθούν τα πάντα. Σε αντίθεση, πολλά έργα ανάπτυξης λογισμικού έχουν υπερβεί τον χρόνο διότι ενώ η διαχείριση υπολόγιζε πως η λειτουργικότητα θα φτιαχτεί, ο διαθέσιμος χρόνος και ο αριθμός των πόρων μεταβλήθηκε. Υπάρχουν ωστόσο περιπτώσεις όπου ολοκληρώθηκε όλη η λειτουργικότητα, αλλά σε πολλές (αν όχι στις περισσότερες) περιπτώσεις, η λειτουργικότητα ποικίλει σε αυτό που πρέπει να έχουμε, αυτό που είναι καλό να έχουμε, αυτό που μπορεί ενδεχόμενος να είναι χρήσιμο και σε αυτό που δεν θα χρησιμοποιηθεί καθόλου. Κατά συνέπεια, πρέπει να δοθεί προτεραιότητα σε τέτοιες λειτουργικότητες, που μπορεί να οδηγήσουν σε μια πιο αδύνατη αλλά περισσότερο αποτελεσματική λύση η οποία θα παραδοθεί μέσα στο χρονικό διάστημα που έχει συμφωνηθεί και μέσα στα πλαίσια του προϋπολογισμού. Για παράδειγμα μια θεμελιώδη αρχή της DSDM είναι ότι τίποτα δεν χτίζεται τέλεια την πρώτη φορά, αλλά ένα εύχρηστο και χρήσιμο 80% του προτεινόμενου συστήματος μπορεί να παραχθεί στο 20% του χρόνου που χρειάζεται για να παραχθεί όλο το σύστημα. Η επίπτωση αυτού είναι ότι το 20% μπορεί ποτέ να μην ζητηθεί ή ακόμα χειρότερα να ζητηθεί στις μετέπειτα παραδόσεις.



Σχήμα 4: Διαφορές μεταξύ ευέλικτων και παραδοσιακών μεθόδων

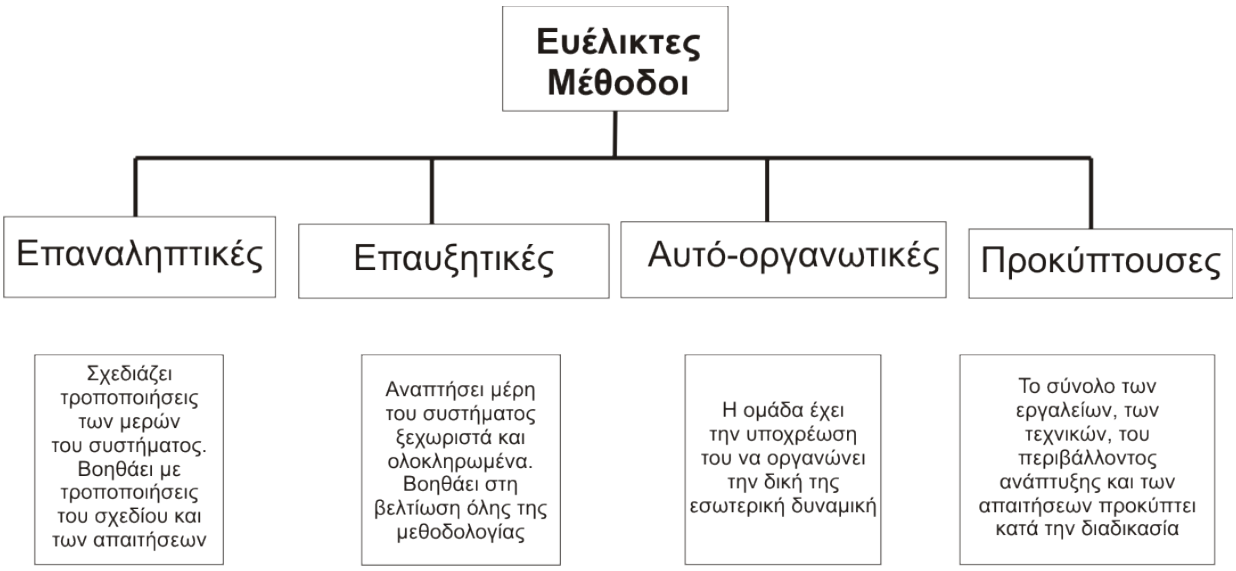
2.3Θεωρητικός ορισμός[8]

Συνοψίζοντας ο υπάρχον ορισμό της ευέλικτης μεθοδολογίας ορίζεται σαν ένα group από διαδικασίες προγραμμάτων ανάπτυξης, αυτές είναι επαναληπτικές, επαυξητικές, αυτό-οργανωτικές και προκύπτουσες.

1. **Επαναληπτικές:** Η λέξη επαναληπτικός προέρχεται από την επανάληψη. Στην περίπτωση της ευέλικτης μεθοδολογίας, δεν μιλάμε μόνο για επαναλήψεις αλλά και για μια απόπειρα να λύσουμε ένα προγραμματιστικό πρόβλημα βρίσκοντας επιτυχή προσέγγιση προς την λύση του προβλήματος ξεκινώντας από ένα αρχικό μικρό σύνολο από απαιτήσεις. Αυτό σημαίνει ότι ο σχεδιαστής σχεδιάζει ένα πλήρες σύστημα από την αρχή και μετά αλλάζει την λειτουργικότητα του κάθε υποσυστήματος με κάθε νέα έκδοση καθώς οι απαιτήσεις αναβαθμίζονται στην κάθε προσπάθεια. Αυτή η προσέγγιση έρχεται σε αντίθεση με τις παραδοσιακές μεθόδους, οι οποίες προσπαθούν να λύσουν το πρόβλημα με μια προσπάθεια. Η επαναληπτική προσέγγιση είναι πιο κατάλληλη για τα σύγχρονα προβλήματα ανάπτυξης λογισμικού τα οποία χαρακτηρίζονται από υψηλή πολυπλοκότητα και γρήγορες αλλαγές στις απαιτήσεις. Επίσης με την αρχή της επανάληψης συνδέεται και η έννοια των συνεχόμενων μεταβολών στην ανάπτυξη.

2. **Επαυξητικές:** Κάθε υποσύστημα είναι κατασκευασμένο με τέτοιο τρόπο ώστε να επιτρέπει να μαζευτούν περισσότερες απαιτήσεις και να χρησιμοποιηθούν ώστε να αναπτυχθούν υποσυστήματα στηριζόμενα στα προηγούμενα. Αυτή η διαδικασία θέλει να διαχωρίσει το συγκεκριμένο σύστημα σε μικρότερα υποσυστήματα βάση της λειτουργίας τους και να προσθέσει νέες λειτουργίες σε κάθε έκδοση. Κάθε έκδοση είναι ένα πλήρως ελεγμένο και έτοιμο για χρήση υποσύστημα με συγκεκριμένες λειτουργίες βασιζόμενο στις υλοποιημένες απαιτήσεις. Καθώς η ανάπτυξη προχωράει, οι έτοιμες για χρήση λειτουργίες αυξάνονται μέχρις ότου ένα πλήρες σύστημα δημιουργηθεί.
3. **Αυτό-οργανωτικές:** Αυτός ο όρος παρουσιάζει μια σχετικά ξένη έννοια στον χειρισμό των επιστημονικών διαδικασιών. Η συνηθισμένη προσέγγιση είναι η οργάνωση ομάδων με βάση την ικανότητα και ανάλογα με τα καθήκοντα τους, να αναφέρονται στη διαχείριση μιας ιεραρχικής δομής. Στην δομή της ευέλικτης ανάπτυξης, η σκέψη της αυτό-οργάνωσης δίνει στην ομάδα την αυτονομία να οργανωθεί από μόνη της ώστε με τον καλύτερο τρόπο να τελειώσει τα θέματα στα οποία εργάζεται. Αυτό σημαίνει ότι η υλοποίηση των θεμάτων όπως η αλληλεπίδραση μεταξύ της ομάδας, η δυναμικότητα της ομάδας, οι ώρες εργασίας, η συναντήσεις σχετικά με την πρόοδο, η αναφορές προόδου και άλλα αφήνονται να αποφασιστούν από την ομάδα για το πόσο καλύτερα μπορούν να γίνουν. Μια τέτοια προσέγγιση είναι κάπως εκκεντρική με τον τρόπο κατά τον οποίο είναι εκπαιδευμένοι οι διαχειριστές των έργων και απαιτεί οι διαχειριστές των έργων να αλλάξουν όλοι μαζί τα διαχειριστικά πρότυπα τους. Αυτή η τεχνική απαιτεί ότι τα μέλη της ομάδας σέβονται το ένα το άλλο και συμπεριφέρονται επαγγελματικά όταν προκύπτει αυτό που σχεδιάστηκε στο χαρτί. Με άλλα λόγια η διαχείριση και ο πελάτης δεν πρέπει να εξαιρούνται από την αποτυχία στο να ικανοποιηθούν οι δεσμεύσεις και δεν πρέπει αδίκως να απαιτούν επεκτάσεις. Ο ρόλος του διαχειριστή του έργου σε μια τέτοια δομή είναι να διευκολύνει την ήπια λειτουργία της ομάδας με το να προσπαθεί να επιτύχει καλή διαχείριση και να αφαιρεί τα διάφορα εμπόδια όποτε αυτό είναι εφικτό. Η προσέγγιση της αυτό-οργάνωσης επομένως συνεπάγεται ότι πρέπει να υπάρχει μια καλή επικοινωνιακή πολιτική μεταξύ διαχειριστή και ομάδας ανάπτυξης.

4. **Προκύπτουσες:** Η λέξη υποδηλώνει 3 πράγματα. Πρώτον, βασίζεται στην φύση των συνεχών μεταβολών της προσέγγισης της ανάπτυξης που το σύστημα επιτρέπει να εμφανιστούν μια σειρά από προσαυξήσεις. Δεύτερον, βασίζεται στην αυτό-οργανωτική φύση μια μέθοδος που προκύπτει σταδιακά καθώς η ομάδα εργάζεται. Τρίτων, καθώς το σύστημα προκύπτει και η μέθοδος εργασίας προκύπτει ένα πλαίσιο των τεχνολογιών ανάπτυξης θα προκύψει επίσης. Η προκύπτουσα φύση των ευέλικτων μεθοδολογιών σημαίνει ότι τα προγράμματα ευέλικτης ανάπτυξης είναι στην ουσία μια διαδικασία συνεχούς εκμάθησης για το κάθε έργο και θα παραμείνει μια διαδικασία συνεχούς εκμάθησης διότι κάθε έργο συμπεριφέρεται διαφορετικά με την εφαρμογή της επαναληπτικής, της επαυξητικής, της αυτό-οργανωτικής και προκύπτουσας τεχνικής.



Σχήμα 5: Οι 4 κύριες διαδικασίες στην ευέλικτη μεθοδολογία

Η αξίας της ευελιξίας είναι στην άδεια των εννοιών να καθορίζονται από νωρίτερα για να μεταλλαχθούν μέσα από τις παραμέτρους των ευέλικτων τιμών και αρχών. Υπάρχει πάντα ο πειρασμός να καθοριστεί ένα πλαίσιο για την ανάπτυξη λογισμικού εάν η επιτυχία είναι επαναλαμβανόμενη, αλλά αυτό θα κατέστρεφε την καινοτομία που έρχεται με την ευέλικτη ανάπτυξη.

2.4Λειτουργικός ορισμός[8]

Θα ορίσουμε τώρα τις ευέλικτες μεθοδολογίες σύμφωνα με τον τρόπο που κάποιοι ευέλικτοι επαγγελματίες τις έχουν καταλάβει καθώς τις χρησιμοποιούν πρακτικά στον πραγματικό κόσμο.

Ο όρος ευέλικτος μεταφέρει πορίσματα από την ευκαμψία, την ευκινησία, ετοιμότητα για μηχανισμούς, ενεργητικότητα, επιδεξιότητα στους μηχανισμούς και προσαρμοστικότητα.

- **Ευκαμψία:** Αυτή η λέξη υποδηλώνει ότι οι κανόνες και οι διαδικασίες στην ευέλικτη ανάπτυξη μπορεί εύκολα να ταιριάξουν στις δεδομένες καταστάσεις χωρίς να χρειάζεται να διαχωριστούν. Με άλλα λόγια ο ευέλικτος τρόπος ανάπτυξης προγραμμάτων επιτρέπει την προσαρμοστικότητα και μεταβλητότητα.
- **Ευκινησία:** Αυτό σημαίνει ότι στην ευέλικτη ανάπτυξη προγραμμάτων πρέπει να υπάρχει γρήγορη παράδοση του προϊόντος. Αυτό συνήθως γίνεται μέσο των νέων εκδόσεων των υποσυστημάτων μέσα σε χρονική περίοδο που κυμαίνεται από μια εβδομάδα έως τέσσερις εβδομάδες. Αυτό δίνει καλά εκμεταλλεύσιμα υποπροϊόντα καθώς ο πελάτης θα αρχίσει να χρησιμοποιεί το σύστημα πριν αυτό ολοκληρωθεί.
- **Ετοιμότητα μηχανισμού:** Στην ευέλικτη ανάπτυξη, η γενική πρόθεση είναι να μειωθούν όλες οι εργασίες και τα υλικά τα οποία μπορεί είτε να επιβραδύνουν την ανάπτυξη είτε να αυξήσουν την γραφειοκρατία.
- **Ενεργητικότητα:** Αυτό περιλαμβάνει, ότι το πραγματικό γράψιμο του κώδικα, αντιτίθεται σε όλο τον σχεδιασμό του προγράμματος που μερικές φορές καταναλώνει τον περισσότερο χρόνο στην ανάπτυξη του λογισμικού.
- **Επιδεξιότητα στο μηχανισμό:** Αυτό σημαίνει ότι πρέπει να υπάρχει μια αφθονία από ικανότητες στην δραστηριότητα της ανάπτυξης κώδικα. Οι ικανότητες αυτές αναφέρονται στις πνευματικές ικανότητες που εξοπλίζουν τους σχεδιαστές για τις προγραμματιστικές προκλήσεις και τις δυναμικές της ομάδας.
- **Προσαρμοστικότητα:** Αυτό έχει δύο πτυχές, πρώτον πρέπει να υπάρχει χώρος για αλλαγές στο σύνολο των δραστηριοτήτων και των τεχνολογιών που αποτελούν την διαδικασία της ευέλικτης ανάπτυξης, δεύτερον οι απαιτήσεις, ο κώδικας και το σχέδιο/αρχιτεκτονική πρέπει να επιτρέπουν τις αλλαγές προς όφελος του πελάτη.

Επίσης οι ευέλικτες μεθοδολογίες είναι ασυνεπείς, αποτελεσματικές, χαμηλού ρίσκου, εύκαμπτες, προβλέψιμες, επιστημονικές και ένας διασκεδαστικός τρόπος για να αναπτύξεις ένα λογισμικό.

•**Ασυνεπείς:** υπονοεί την ελαχιστοποίηση όλων αυτών που πρέπει να γίνουν στην διαδικασία της ανάπτυξης(πχ τεκμηρίωση, απαιτήσεις) προκειμένου να αυξήσει την ταχύτητα και την αποτελεσματικότητα στην ανάπτυξη. Η ιδέα της ελαχιστοποίησης της τεκμηρίωσης είναι ακόμα αμφισβητούμενη, δεδομένου ότι μερικοί υποθέτουν ότι η ευελιξία σημαίνει καθόλου τεκμηρίωση. Τέτοιες απόψεις ωστόσο δεν είναι αβάσιμες διότι μερικοί ακραίοι υποστηρικτές της ευέλικτης μεθόδου έχουν ρητά συμπεράσματα μηδενικής τεκμηρίωσης αξιώνοντας ότι ο κώδικας είναι επαρκώς τεκμηριωμένος. Καθώς οι ευέλικτες μεθοδολογίες φτάνουν σε πιο υψηλά επίπεδα ωριμότητας η ελαχιστοποίηση της τεκμηρίωσης έχει εξελιχθεί και συνήθως συνεπάγεται ότι όσο περισσότερη τεκμηρίωση θέλει ο πελάτης τόσο πρόθυμος πρέπει να είναι για να πληρώσει σε χρόνο και χρήμα.

- **Αποτελεσματικές:** σημαίνει να κάνεις μόνο την δουλειά η οποία θα προσφέρει το επιθυμητό προϊόν με όσο το δυνατόν λιγότερα έξοδα όσο είναι αυτό πρακτικά πιθανό.
- **Χαμηλού ρίσκου:** συνεπάγεται εκμετάλλευση των χρήσιμων κατευθύνσεων και εγκατάλειψη των αγνώστων έως ότου γίνουν γνωστά. Το πραγματικό γεγονός είναι ότι όλες οι μεθοδολογίες ανάπτυξης λογισμικού έχουν ως σκοπό να μειώσουν το ρίσκο της αποτυχίας του έργου. Κατά περιόδους πάρα πολύ προσπάθεια έχει σπαταληθεί στην θεωρητική αφαίρεση του προβλήματος σε μια προσπάθεια ώστε να ρυθμιστεί ο κίνδυνος.
- **Προβλέψιμες:** σημαίνει ότι οι ευέλικτες μεθοδολογίες βασίζονται στο τι οι επαγγελματίες κάνουν όλη της ώρα, με άλλα λόγια ο κόσμος της ασάφειας μειώνεται. Αυτό ωστόσο δεν σημαίνει ότι ο προγραμματισμός, ο σχεδιασμός και η αρχιτεκτονική του λογισμικού είναι προβλέψιμα. Σημαίνει ότι η ευελιξία επιτρέπει την ανάπτυξη λογισμικού με τους πιο φυσικούς τρόπους τους οποίους οι εκπαιδευμένοι κατασκευαστές έχουν καθορίσει εκ των προτέρων βασιζόμενοι σε ιδικές γνώσεις.
- **Επιστημονικές:** σημαίνει ότι οι ευέλικτες μεθοδολογίες ανάπτυξης λογισμικού βασίζονται σε πλήρης και αποδεδειγμένες επιστημονικές αρχές. Παρόλα αυτά παραμένει η ευθύνη στον ακαδημαϊκό κόσμο ώστε να συνεχίσει να μαζεύει εμπειρικά στοιχεία σχετικά με την ευέλικτή διαδικασία επειδή οι περισσότεροι από τους επαγγελματίες που συνέγραψαν την ευέλικτη μεθοδολογία δεν είχαν τον χρόνο για να κάνουν τέτοιου είδους έρευνα.

•**Διασκεδαστικές:** διότι επιτέλους οι κατασκευαστές επιτρέπεται να κάνουν αυτό που θέλουν πιο πολύ(πχ να καταναλώσουν τον περισσότερο από τον χρόνο τους γράφοντας καλό κώδικα που δουλεύει). Για τους κατασκευαστές, η ευελιξία προσφέρει μια μορφή ελευθερίας ώστε να είναι δημιουργικοί και καινοτόμοι χωρίς ο πελάτης να είναι υποχρεωμένος να πληρώσει για αυτό, αντιθέτως ο πελάτης επωφελείται από αυτό.

Η ευέλικτη μεθοδολογία ορίζεται σαν μια αντίθετη πορεία στις 30ετής όλο και περισσότερο καταπιεστικές διαδικασίες που σημαίνει ότι αλλάζει τον απλό προγραμματισμό των υπολογιστών σε επιστήμη του λογισμικού καθιστώντας το τόσο εύχρηστο και προβλέψιμο όσο είναι οποιαδήποτε άλλη επιστήμη εφαρμοσμένης μηχανικής.

Σε μια πρακτική προοπτική, οι ευέλικτες μεθοδολογίες προκύπτουν από μια κοινή ανακάλυψη μεταξύ των επαγγελματιών των οποίων η πρακτική απομακρύνθηκε σταδιακά από την παραδοσιακή βαριά τεκμηρίωση και το επίκεντρο της διαδικασίας ανάπτυξης προσεγγίζει περισσότερο το ανθρωποκεντρικό και λιγότερο την καθοδηγούμενη από τεκμηρίωση προσέγγιση. Υπάρχει μια γενική παρερμηνεία στο ότι στην ευέλικτη διαδικασία δεν υπάρχει καθόλου ή λίγος σχεδιασμός. Αυτό οφείλεται στο γεγονός ότι στις δημόσιες λίστες της ευελιξίας σαν μια από τις τέσσερις αξίες τους είναι η προτίμηση για ανταπόκριση στην αλλαγή μετά από ένα πλάνο. Στην πραγματικότητα, ο σχεδιασμός σε ευέλικτα έργα θα μπορούσε να είναι περισσότερο ακριβής από ότι στις παραδοσιακές διαδικασίες αυτό γίνεται αυστηρά για κάθε αύξηση και από την προοπτική σχεδιασμού ενός έργου οι ευέλικτες μεθοδολογίες παρέχουν μια προσέγγιση μείωσης του κινδύνου όπου η περισσότερο σημαντική αρχή του ευέλικτου σχεδιασμού είναι η ανατροφοδότηση. Μπορούμε να απαριθμήσουμε τους κινδύνους σε: κίνδυνοι παρανόησης στην λειτουργικότητα των απαιτήσεων, κίνδυνοι για βαθιά ρήγματα στην αρχιτεκτονική, κίνδυνοι από την μη αναμενόμενη αλληλεπίδραση με τον χρήστη, κίνδυνοι από λάθος ανάλυση και κατασκευή μοντέλου, κίνδυνοι όχι καλής κατανόησης της επιλεγμένης τεχνολογίας από την ομάδα. Η ανατροφοδότηση εξασφαλίζεται με την κατασκευή μιας λειτουργικής έκδοσης του συστήματος σε τακτά χρονικά διαστήματα ή με την συνεχόμενη επαύξηση της προηγούμενης σχεδιαστικής προσπάθειας.

Παρά να ασχοληθεί με τους πιο σχετικούς κινδύνους της ανάπτυξης λογισμικού μέσω της επαυξητικής ανάπτυξης, η ευέλικτη μεθοδολογία επιτίθεται με την προϋπόθεση ότι τα πλάνα, τα σχέδια, οι αρχιτεκτονικές και οι απαιτήσεις είναι προβλεπόμενα και επομένως μπορούν να σταθεροποιηθούν. Οι ευέλικτες μέθοδοι επιτίθενται επίσης με την προϋπόθεση ότι οι διαδικασίες είναι επαναλαμβανόμενες. Οι δύο αυτές προϋποθέσεις είναι μέρη των θεμελιωδών αρχών πάνω στις οποίες χτίστηκαν οι παραδοσιακές μεθοδολογίες και τυχαίνει να είναι και τα κύρια εμπόδια των παραδοσιακών μεθοδολογιών.

2.5Εκτίμηση ποιότητας στις Ευέλικτες διαδικασίες[8]

Το ερώτημα που τίθεται εδώ είναι: Μπορούμε να εκτιμήσουμε την ποιότητα στις ευέλικτες διαδικασίες; Αυτό μπορούμε να το επιτύχουμε μέσω:

- Της παροχής της λεπτομερείς γνώσης σχετικά με συγκεκριμένα θέματα ποιότητας των ευέλικτων διαδικασιών.
- Προσδιορισμό των καινοτόμων τρόπων για βελτίωση της ποιότητας στην ευέλικτη διαδικασία.
- Προσδιορισμό συγκεκριμένων ευέλικτων τεχνικών ποιότητας για συγκεκριμένες ευέλικτες μεθοδολογίες.

Η λογοτεχνία μας δείχνει ότι ο Huo et al(2004) ανέπτυξε μια τεχνική σύγκρισης που στόχος της ήταν να παρέχει μια συγκριτική ανάλυση μεταξύ της ποιότητας στην μέθοδο ανάπτυξης του μοντέλου του καταρράκτη(σαν εκπρόσωπο των παραδοσιακών μεθόδων) και της ποιότητας στα γκρουπ των ευέλικτων μεθοδολογιών. Τα αποτελέσματα της ανάλυσης δείχνουν ότι πραγματικά υπάρχει διασφάλιση της ποιότητας στην ευέλικτη ανάπτυξη, αλλά έχει πραγματοποιηθεί με διαφορετικούς τρόπους απ’ ότι στις παραδοσιακές διαδικασίες. Οι περιορισμοί του εργαλείου του Huo et al ωστόσο είναι ότι η ανάλυση:

- Επιλέγει δυο κύριες πτυχές της διαχείρισης της ποιότητας ειδικότερα την διασφάλιση και επαλήθευση της ποιότητας και την επικύρωση.
- Παραβλέπει άλλες ζωτικής σημασίας τεχνικές που χρησιμοποιούνται στις ευέλικτες διαδικασίες για να πετύχει υψηλότερη διαχείριση ποιότητας.
- Η ευέλικτη διασφάλιση της ποιότητας πηγαίνει ένα βήμα πιο μπροστά τα θέματα της ποιότητας απ’ ότι τα παραδοσιακά λογισμικά διασφάλισης της ποιότητας.

Μια άλλη πρόκληση της τεχνικής του Huo et al είναι ότι ενώ ο κύριος στόχος αυτής της ανάλυσης ήταν να δείξει ότι υπάρχει διασφάλιση της ποιότητας στην ευέλικτη διαδικασία δεν δείχνει καθαρά ότι η συνέχεια θα είναι. Οι υπερασπιστές των ευέλικτων μεθόδων δεν φαίνεται να ανησυχούν για την σύγκριση μεταξύ των ευέλικτων και των παραδοσιακών διαδικασιών όπως μερικοί από τους πιο ενθουσιώδης “ευελικτιστές” πίστευαν ότι δεν υπάρχει τρόπος οι παραδοσιακές μέθοδοι μπορούν να ταιριαστούν με τις ευέλικτες μεθόδους σε καμία κατάσταση.

Πίνακας 2: Παράμετροι Ποιότητας Λογισμικού στις Ευέλικτες Τεχνικές

Παράμετροι Λογισμικών Ποιότητας	Ευέλικτες Τεχνικές	Πιθανές Βελτιώσεις
Ακρίβεια(Correctness)	Γραφή κώδικα από ελάχιστες απαιτήσεις Οι προδιαγραφές επιτυγχάνονται με την άμεση επικοινωνία με τον πελάτη. Ο πελάτης μπορεί να αλλάξει τις προδιαγραφές.	Λαμβάνοντας υπόψη την πιθανότητα της χρήσης επίσημης προδιαγραφής στην ευέλικτη ανάπτυξη, Πιθανή χρήση γενικών σεναρίων για ορισμό των προδιαγραφών(εδώ πρέπει να σημειώσουμε ότι κάποιες ομάδες ανάπτυξης κάνουν ήδη χρήση αυτής της τεχνικής)
Ευρωστία(Robustness)	Δεν υπάρχει άμεση αντιστοίχηση στις ευέλικτες μεθόδους.	Περιλαμβάνει πιθανές ακραίες συνθήκες στις απαιτήσεις.
Επεκτασιμότητα (Extendibility)	Ένα γενικό χαρακτηριστικό όλων των ΟΟ εφαρμογών ανάπτυξης. Έμφαση βρίσκεται στις άψογες τεχνικές και το καλό σχεδιασμό. Η Έμφαση επίσης επιτυγχάνει καλύτερη κατασκευή.	Χρήση της τεχνικής μοντελοποίησης για την αρχιτεκτονική λογισμικού.
Επαναχρησιμοποίηση (Reusability)	Ένα γενικό χαρακτηριστικό όλων των ΟΟ εφαρμογών ανάπτυξης. Υπάρχουν κάποιες διαφωνίες ενάντια στην ικανότητα επαναχρησιμοποίη- ση ευέλικτων προϊόντων.	Ανάπτυξη προτύπων για τις ευέλικτες εφαρμογές.
Συμβατότητα(Compatibility)	Ένα γενικό χαρακτηριστικό όλων των ΟΟ εφαρμογών ανάπτυξης.	Μπορούν επιπλέον χαρακτηριστικά να προστεθούν για χάρη της συμβατότητας ακόμα και αν δεν είναι απαραίτητα. Αυτό μπορεί να αντίκρουση την αρχή της απλότητας.
Αποδοτικότητα(Efficiency)	Προσθέτει καλά πρότυπα κωδικοποίησης.	Προτρέπει τον σχεδιασμό που βασίζεται σε πιο αποδοτικούς αλγόριθμους.
Φορητότητα(Portability)	Πρακτική συνεχούς ενσωμάτω-σης στον ακραίο προγραμματισμό.	Κάποιες ευέλικτες μέθοδοι δεν έχουν άμεση αντιστοίχηση προβλημάτων στην ανάπτυξη προϊόντος. Η λύση αυτού του προβλήματος μπορεί να αποτελεί το πλεονέκτημα της ευελιξίας.

Επικαιρότητα(Timeliness)	Το πιο δυνατό χαρακτηριστικό της ευελιξίας. Μικροί κύκλοι, γρήγορη παράδοση κ.α.	
Ακεραιότητα(Integrity)	Όχι άμεση αντιστοίχιση στην ευέλικτη ανάπτυξη.	
Επιβεβαιωσημότητα (Verifiability)	Η μέθοδο της Test-Driven ανάπτυξης είναι άλλο ένα δυνατό σημείο της ευελιξίας.	
Άνεση στη χρήση(Ease of use)	Εφόσον ο πελάτης είναι μέρος της ομάδας και οι πελάτες ανατροφοδοτούν συχνά, θα συστήσουν πιθανότατα ένα σύστημα που είναι πιο εύχρηστο.	Σχεδιάστηκε για τον λιγότερο ικανό χρήστη στην οργάνωση.

2.6 Τεχνικές(Techniques)[8]

Οι παράμετροι που μπορούν να ορίσουν την ποιότητα του λογισμικού από ένα υψηλό επίπεδο μπορούν να είναι κάπως αφηρημένες. Ωστόσο η προτεινόμενη τεχνική επιλέγει κάθε μια από τις παραμέτρους και προσδιορίζει τις αντίστοιχες ευέλικτες τεχνικές οι οποίες εφαρμόζουν τις παραμέτρους με τον ένα τρόπο ή τον άλλο. Πιθανές βελτιώσεις στην συγκεκριμένη πρακτική έχουν προταθεί από την ανάλυση του τρόπου που δουλεύουν οι χρήστες των ευέλικτων τεχνικών. Το πιο σημαντικό από αυτή την ανάλυση είναι η αναθεώρηση μερικών από τις διαισθητικές πρακτικές που οι κατασκευαστές εφαρμόζουν συνήθως και οι οποίες μπορεί να μην είναι τεκμηριωμένες. Μπορεί να αναρωτιόμαστε πόση αντικειμενικότητα μπορεί να υπάρχει σε τόσες πληροφορίες. Το θέμα είναι ότι οι κατασκευαστές λένε τις επιτυχίες τους σε διάφορα επαγγελματικά forums και μερικές από αυτές τις συμβουλές-επιτυχίες από κάποιες τέτοιες μελέτες έχουν συλληφθεί σε αυτές τις τεχνικές χωρίς να έχει ακολουθηθεί καμία τυπική μεθοδολογία συλλογής δεδομένων. Κάποιοι ερευνητές πιστεύουν ότι η μη τυποποιημένη συλλογή μη επεξεργασμένων δεδομένων επηρεάζει την ισορροπία των γεγονότων ειδικά στην περίπτωση που οι κατασκευαστές μιλούν για τις τεχνικές τους.

Σύμφωνα με την επίσημη διαχείριση της ποιότητας του λογισμικού, οι δραστηριότητες για την διασφάλιση της ποιότητας πραγματοποιούνται και σαν στόχο έχουν να διασφαλίσουν ότι κάθε μια από τις παραμέτρους που βρίσκονται στον πίνακα 2 θα προχωρήσει μέχρι κάποιο συγκεκριμένο σημείο στον κύκλο της ανάπτυξης λογισμικού του έργου που μας ενδιαφέρει. Μια σύντομη περιγραφή της κάθε παραμέτρου δίνεται στη συνέχεια σύμφωνα με τον Meyer(2000):

- **Ακρίβεια(Correctness):** Η δυνατότητα ενός συστήματος να λειτουργεί σύμφωνα με ορισμένες προδιαγραφές.

- **Ευρωστία(Robustness):** Η κατάλληλη συμπεριφορά ενός συστήματος κάτω από ακραίες συνθήκες. Αυτή η παράμετρος συμπληρώνει την Ακρίβεια.
- **Επεκτασιμότητα (Extendibility):** Η παράμετρός που μας φανερώνει το πόσο εύκολα ένα σύστημα μπορεί να προσαρμοστεί σε νέες προδιαγραφές.
- **Επαναχρησιμοποίησης (Reusability):** Η παράμετρος που μας δείχνει ότι ένα σύστημα είναι φτιαγμένο από στοιχεία τα οποία μπορούν να χρησιμοποιηθούν για την κατασκευή διαφορετικών εφαρμογών.
- **Συμβατότητα(Compatibility):** Η παράμετρος που μας δείχνει ότι ένα σύστημα είναι φτιαγμένο από στοιχεία τα οποία εύκολα μπορούν να συνδυαστούν με άλλα στοιχεία.
- **Αποδοτικότητα(Efficiency):** Η ικανότητα που έχει σύστημα να αναθέτει μερικές απαιτήσεις όσο τα υλικά του(hardware) του επιτρέπουν όπως η μνήμη, ο χρόνος επεξεργασίας και το εύρος ζώνης της επικοινωνίας.
- **Φορητότητα(Portability):** Η παράμετρος που μας δείχνει πόσο εύκολα μπορεί να εγκατασταθεί το λογισμικό σε διαφορετικού τύπου υλικά(hardware) και πλατφόρμες.
- **Επικαιρότητα(Timeliness):** Η έκδοση του λογισμικού πριν ή ακριβώς όταν το έχει ανάγκη ο χρήστης.
- **Ακεραιότητα(Integrity):** Η παράμετρος που μας δείχνει πόσο καλά το λογισμικό προστατεύει τα προγράμματα του και τα δεδομένα του ενάντια σε μη εξουσιοδοτημένη προσπάθεια πρόσβασης.
- **Επιβεβαιωσημότητα(Verifiability):** Η παράμετρος του πόσο εύκολο είναι να εξεταστεί το λογισμικό.
- **Άνεση στη χρήση(Ease of use):** Η παράμετρος του πόσο εύκολα οι άνθρωποι διαφορετικών υπόβαθρων μπορούν να μάθουν και να χρησιμοποιήσουν το λογισμικό.

2.7 Το πλαίσιο αξιολόγησης της ευέλικτης μεθοδολογίας[8]

Όλες οι ευέλικτες μεθοδολογίες έχουν εντυπωσιακές ομοιότητες σχετικά με τις διαδικασίες τους, διότι βασίζονται σε 4 ευέλικτες αξίες και 12 αρχές. Είναι ενδιαφέρον να σημειωθεί ότι ακόμα και οι ίδιοι οι δημιουργοί των ευέλικτων μεθοδολογιών δεν δίνουν έμφαση στα ευέλικτα όρια των μεθοδολογιών τους και είναι πρόθυμοι να χρησιμοποιήσουν πρακτικές από άλλες ευέλικτες μεθοδολογίες αν αυτές τους εξυπηρετούν για τις καταστάσεις που τους έχουν δοθεί. Μια λεπτομερή αναθεώρηση των ευέλικτων μεθοδολογιών αποκαλύπτει ότι οι ευέλικτες διαδικασίες αντιμετωπίζουν τα ίδια θέματα χρησιμοποιώντας διαφορετικά πραγματικής ζωής μοντέλα.

Οι τεχνικές αξιολόγησης για παράδειγμα η LD(Lean Development Αδύνατη Ανάπτυξη) παρουσιάζει την ανάπτυξη λογισμικού να χρησιμοποιεί μια βιομηχανική και ανάπτυξη προϊόντος μεταφορά. Η Scrum παρουσιάζει τις διαδικασίες ανάπτυξης λογισμικού να χρησιμοποιούν μια ελεγχόμενη μηχανική μεταφορά. Η XP παρουσιάζει τις δραστηριότητες ανάπτυξης λογισμικού σαν μια κοινωνική ενέργεια που οι κατασκευαστές κάθονται μαζί.

2.8 Διαδικασίες και Μετρικές Ποιότητας Προϊόντος στην παραδοσιακή και Ευέλικτη Ανάπτυξη Λογισμικού[8]

Η αποτυχία και η επιτυχία των IT/IS έργων συζητήθηκε από τις αρχές της δεκαετίας του 70΄ όταν οργανισμοί άρχισαν να χρησιμοποιούν την τεχνολογία των υπολογιστών για να ελέγξουν την ικανότητα των πληροφοριακών συστημάτων τους. Η αποτυχία των IT έργων πολύ έντονα αναφέρθηκε και μελετήθηκε. Στατιστικές που παρουσιάστηκαν αποκάλυψαν ότι πέντε στα έξι έργα λογισμικού χαρακτηρίστηκαν μη επιτυχείς, και περίπου το ένα τρίτο των έργων ακυρώθηκαν. Τα εναπομείναντα έργα παρέδωσαν λογισμικά περίπου στο διπλάσιο από τον υπολογιζόμενο προϋπολογισμό και χρονικό περιθώριο που αρχικά είχε σχεδιαστεί.

Σημαντικές πρόοδοι έχουν επιτευχθεί με την παρουσίαση μεθόδων και εργαλείων για τον συστηματικό έλεγχο και σχεδιασμό της διαδικασίας της ανάπτυξης. Παρά αυτές τις προσπάθειες τα συστήματα συνέχιζαν αν αποτυγχάνουν με δραματική συχνότητα. Οι μεθοδολογίες για την ανάπτυξη συστημάτων προτάθηκαν και χρησιμοποιήθηκαν για να εξεταστούν τα προβλήματα των διφορούμενων προδιαγραφών των χρηστών, των μη φιλόδοξων σχεδιασμών συστημάτων, των χρονικών περιθωρίων που δεν ικανοποιήθηκαν, των προϋπολογισμών που ξεπεράστηκαν, της φτωχής ποιότητας λογισμικού με τα πολλά λάθη και της φτωχής τεκμηρίωσης. Αυτοί αλλά και άλλοι παράγοντες έκαναν τα IS αδιάλλακτα σε μελλοντικές αλλαγές και δύσκολα στη συντήρηση.

2.9 Ποιότητα και Διαδικασία Βελτίωσης Λογισμικού[8]

Η διαδικασία βελτίωσης λογισμικού(Software process improvement SPI) έγινε ένα πρακτικό εργαλείο για τις εταιρίες όπου η ποιότητα του λογισμικού έχει μεγάλη σημασία. Σε μια τεχνική αναφορά με αποτελέσματα από 13 οργανισμούς και με τον αριθμό της αναφοράς μειονεκτημάτων της μετα-απελευθέρωσης χρησιμοποιημένη σαν μέτρο σύγκρισης οι Herbsleb, Carleton, Rozum, Siegel και Zubrow έδειξαν ότι μέσο της διαδικασίας βελτίωσης λογισμικού τα προϊόντα και η αξία της επιχείρησης(ειδικά οι επιστρεφόμενες επενδύσεις) βελτιώθηκαν. Εξετάζεται γενικά ότι μια καλά τεκμηριωμένη και μια συνεχόμενη διαδικασία είναι ουσιαστική για την ανάπτυξη λογισμικών προγραμμάτων με υψηλή ποιότητα. Υπάρχουν ακόμα στοιχεία ότι η χρήση προτύπων και μοντέλων εκτίμησης διαδικασιών έχουν μια θετική επίπτωση στην ποιότητα του τελικού προϊόντος.

Η κοινότητα τεχνολογίας λογισμικού σταδιακά μετακινήθηκε από τις διορθωτικές μεθόδους που βασίζονται στην ποιότητα του προϊόντος στις προληπτικές μεθόδους που βασίζονται στις διαδικασίες, μετατοπίζοντας κατά συνέπεια την έμφαση από την βελτίωση της ποιότητας των προϊόντων στην βελτίωση της ποιότητας των διαδικασιών. Οι επιθεωρήσεις στο τέλος της γραμμής παραγωγής έχουν πλέον αντικατασταθεί από τα περάσματα των σχεδίων και τις ενσωματωμένες τεχνικές διασφάλισης ποιότητας σε όλο τον κύκλο ζωής την ανάπτυξης. Η ποιότητα είναι μια ασαφής, πολύπλοκη έννοια και η ποιότητα του λογισμικού εξαρτάται από τις απόψεις και τις τοποθετήσεις των εμπλεκομένων που ενδιαφέρονται για διαφορετικά ποιοτικά χαρακτηριστικά.

Τα στοιχεία για την ποιότητα της διαδικασίας είναι επίσης ότι οι πιστοποιήσεις ISO δεν πιστοποιούν την ποιότητα του προϊόντος αλλά λαμβάνουν υπόψη ότι μια διατυπωμένη και τεκμηριωμένη διαδικασία ακολουθήθηκε. Ομοίως γνωστά και αποδεκτά μοντέλα αξιολόγησης όπως το CMM/CMMI, BOOT-STRAP και SPICE/ISO 15504 εστιάζουν στην αξιολόγηση της ποιότητας της διαδικασίας και όχι στην ποιότητα του τελικού προϊόντος.

Αυστηρά εμφανίζοντας και συνοψίζοντας τις αξίες και τις αρχές της παραδοσιακής διαδικασία βελτίωσης λογισμικού και ευέλικτων προτύπων οι πίνακες 3 και 4 μας παρέχουν μια συγκριτική αναφορά για τις τάσεις και τις τεχνικές που οι δυο προσεγγίσεις βασίζονται. Αρχικά ο πίνακας 3 μας παρουσιάζει μια σύντομη περιγραφή των αξιών της SPI και της ευέλικτης προσέγγισης. Στον πίνακα 4 συνεχίζεται η έκθεση των ποιοτικών παραγόντων και ειδικών χαρακτηριστικών/αρχών μέσω των οποίων η ποιότητα ενσωματώνεται στην διαδικασία βελτίωσης λογισμικού, και/ή στην παραγωγή λογισμικού.

Πίνακας 3: Αξίες στην παραδοσιακή SPI και στα ευέλικτα πρότυπα.

SPI	Ευέλικτα
Διαδικασίες και εργαλεία	Άτομα και Αλληλεπιδράσεις
Περιεκτική Τεκμηρίωση	Εφαρμόσιμο Λογισμικό
Διαπραγμάτευση Συμβολαίων	Συνεργασία Πελατών
Αλλαγή μέσω ενός σχεδίου	Αλλαγή μέσω γρήγορης αντίδρασης

Πίνακας 4: Ποιοτικοί παράγοντες στις βασικές αρχές και αξίες της SPI και των ευέλικτων προτύπων.

Ποιοτικοί Παράγοντες	SPI	Ευέλικτα Πρότυπα
Ιδεολογία	Ενδυνάμωση	Καινοτόμα-Συμμετοχική-Ενδυναμωτική
Τρόπος Ζωής	Προσανατολισμός βασιζόμενος στην Εργασίας	Προσανατολισμός βασιζόμενος στη Ζωή
Προσέγγιση	Οδηγούμενη βάση σχεδίου και ρυθμιστικές διαδικασίες Διαδικασίες καθοδηγούμενες-άκαμπτες-γραφειοκρατικές	Εύκαμπτη, Εξελικτική, Προσαρμοστική, Επαναληπτική, Επαυξητική
Κατευθυντήριες Δυνάμεις	Διαχειριστική υποχρέωση και ηγεσία	Τεχνικά ικανοί και παρακινούμενοι Κατασκευαστές
Εμπλοκή Πελάτη	Αρχικό και προχωρημένο στάδιο στον κύκλο ζωής	Καθ' όλη την διάρκεια του κύκλου ζωής
Συμμετοχή Πελάτη	Ενθαρρυμένη- Εστίαση Πελάτη	Επιτακτική συμμετοχή Χρηστών
Επικοινωνία	Επίσημη	Ανεπίσημη
Ομάδες	Δια-ομαδικός Συντονισμός	Αυτό-οργανωτικές ομάδες
Ανταπόκριση	Γραφειοκρατικές Καθυστερήσεις	Γρήγορη Αντίδραση
Δημιουργία Γνώσης	Σιωπηρή-Επίσημη-Ρητή	Σιωπηρή-Ανεπίσημη-Ρητή
Μοίρασμα Γνώσης	Επιθυμητή-Επίσημη	Προστακτική-Ανεπίσημη
Τεκμηρίωση	Μέγιστη	Ελάχιστη
Μεταβαλλόμενες Απαιτήσεις	Οι διαδικασίες πρέπει να ακολουθηθούν	Προσαρμοστικότητα στις αλλαγές καθ' όλη την διάρκεια της διαδικασίας ανάπτυξης
Δοκιμές	Αργά στον Κύκλο Ζωής	Πρώτα Δοκιμές
Ανίχνευση Λαθών	Επιθεώρηση	Pair Programming
Αναφορές Προόδου	Επίσημες όμοιες αναφορές	Συνεχόμενες όμοιες αναφορές
Απόσπαση Απαιτήσεων	Προσχεδιασμένες και Σπάνιες	Καθημερινές συναντήσεις
Υποστήριξη Εργαλείων	Τα εργαλεία υποστηρίζουν διαφορετικές φάσεις του κύκλου ζωής- Χωρισμένες	Αυτοματοποιημένα εργαλεία ελέγχου
Παράδοση Προϊόντος	Προγραμματισμένη	Συχνά – Ευέλικτος Σχεδιασμό
Λειτουργία Διασφάλισης Ποιότητας	Τυποποιημένη - Χωρισμένη	Ανεπίσημη – που Ενσωματώνεται

2.10Τι να μετρήσουμε και πώς να βελτιώσουμε την ποιότητα στην ευέλικτη ανάπτυξη.

Έχει πλέον αναγνωριστεί ότι η μετρήσεις παρέχουν διορατικότητα στις αδυναμίες και στα δυνατά χαρακτηριστικά και της διαδικασία και του προϊόντος. Το να μετράς σημαίνει να γνωρίζεις. Ότι δεν μπορείς να μετρήσεις, δεν μπορείς να το αποδείξεις. Άρα το ερώτημα “Γιατί να μετράμε;” είναι κατανοητό και αποδεκτό, αλλά οι αποφάσεις του “Τι πρέπει να μετρηθεί”, ”Πως πρέπει να μετρηθεί”, “Πότε πρέπει να μετρηθεί” και ιδιαίτερα “Ποιος πρέπει να μετρήσει” και “Για ποιόν” γίνονται κρίσης σημασίας, αν τα οφέλη πρόκειται να κερδηθούν από συγκεκριμένες ομάδες. Στη συνέχεις πρέπει να καταλάβουμε ποιοι συμμετέχοντες και γιατί θα επωφεληθούν από τις μετρήσεις και από τις βελτιώσεις, και μακροπρόθεσμα ποιος είναι ο πιο κατάλληλος για την μέτρηση και ποιος θα προκαλέσει τις μεταρρυθμίσεις, τις αλλαγές και τις βελτιώσεις στα λογισμικά προϊόντα και στις λογισμικές διαδικασίες.

2.11Μετρήσεις του προϊόντος ή της διαδικασίας ποιότητας στην ανάπτυξη ευέλικτου λογισμικού;[8]

Θα μπορούσαν οι ευπροσάρμοστες ευέλικτες ποιοτικές μετρικές να υπάρξουν για τις διαδικασίες ευέλικτης ανάπτυξης λογισμικού και για τα παράγωγα τους; Η ευελιξία αναφέρεται στην ικανότητα να μπορεί συνέχεια να προσαρμόζει και το κόστος και την προσπάθεια χρησιμοποιώντας παραμέτρους όπως τα λειτουργικά σημεία και τις γραμμές κώδικα.

Ο Gilb(2006) πρότεινε ότι τα πρότυπα των ευέλικτων μεθόδων θα μπορούσαν να επωφεληθούν αν περιείχαν τις μετρικές των συμμετεχόντων για να εστιάσουν στις κρίσιμες απαιτήσεις, στην μέτρηση των διαδικασιών και στην διευκόλυνση της απάντησης για ανατροφοδότηση. Στην πρόταση του ο Gilb διαβεβαίωσε την ανάγκη ότι η ευέλικτη ανάπτυξη πρέπει να εστιάσει στις πιο ποσοτικές και μετρήσιμες απαιτήσεις των συμμετεχόντων και στην ανάγκη για να κρατιέται μια διαδρομή αλλαγών και πλάνων για καλύτερη πρόοδο σε μικρότερους κύκλους έργων. Ωστόσο οι αυστηρές και ευμετάβλητες ευέλικτες διαδικασίες μέτρησης μπορεί να είναι δύσκολο να καθιερωθούν για τις απαιτήσεις της ευέλικτης ανάπτυξης λογισμικού. Ενδεικτικά, δεν υπάρχει καμία συγκεκριμένη καθιερωμένη επιστημονική διαδικασία για να το κάνει αυτό, και επιπλέον ούτε μεγάλη βιομηχανική ούτε ακαδημαϊκή ουσιαστική απόδειξη για την φύλαξη των ποσοτικών προσδιορισμών μέτρων στα ευέλικτα έργα ανάπτυξης λογισμικού.

Μερικές ευέλικτες μετρικές για παράδειγμα έχουν αναφερθεί και χρησιμοποιηθεί από μια ερευνητική ομάδα του Ισραηλινού στρατού. Τα αναφερόμενα αποτελέσματα υποστηρίζουν τον αρχικό στόχο, ο οποίος είναι να “επιτρέπεται μια ακριβής και επαγγελματική διαδικασία λήψης αποφάσεων και για βραχυπρόθεσμους και για μακροπρόθεσμους στόχους”.

Από την άλλη μεριά οι ευέλικτες μετρικές δεν είναι και πολύ διαφορετικές από τις παραδοσιακές μετρικές. Η διαφορά της ευελιξία πιθανότατα αναφέρεται στην μεθοδολογία και στην ταχύτητα της έρευνας. Από τότε που οι ευέλικτες μέθοδοι υποστηρίζουν την αδύνατη τεκμηρίωση και τις συχνές παραδόσεις προϊόντων, στην ευελιξία η εκτίμηση και η συνεχής προσαρμογές είναι απαραίτητες έτσι ώστε τα έργα να παραδίδονται και στην ώρα τους και μέσα στον προϋπολογισμό. Για να πετύχει λοιπόν αυτή η επαναληπτική διαδικασία πρέπει να αποκτήσουμε μια πρακτική μιας ώριμης διαδικασίας μέσα στην οργάνωση.

2.12Παράγοντες Δοκιμής, Βελτίωσης και Επανακατασκευής στον Παραδοσιακό και Ευέλικτο κύκλο ζωής των διαδικασιών.[8]

Οι προσπάθειες για να μοντελοποιηθούν οι διαδικασίες μας πρόσφεραν πολυάριθμα μοντέλα, ISD μεθοδολογίες, ποιοτικά μοντέλα, και πρότυπα διαδικασίας ωριμότητας. Μια έρευνα έγινε για τα διάφορα μοντέλα δίνοντας συγκεκριμένη έμφαση στην δοκιμή όλου του κύκλου ζωής. Η έρευνα μέχρι στιγμής αναγνώρισε ότι περισσότεροι ώριμοι κύκλοι ζωής και συνεπώς περισσότερες ώριμες διαδικασίες τεχνολογίες λογισμικού, κινούνται στις δοκιμές και άλλες τεχνικές διασφάλισης της ποιότητας εφαρμόζονται στα νωρίτερα στάδια του κύκλου ζωής της ανάπτυξης.

Μια παράθεση από μια ιστορική παρουσίαση των μοντέλων στην CMM κλίμακα μας δείχνει ότι μεταξύ του 1970 και του 2000 καθώς κινούμαστε από το μοντέλο καταρράκτη(το οποίο εμφανίζεται κατά το 1956) στα αυξητικά μοντέλα η ωριμότητα των υποκείμενων διαδικασιών αυξήθηκε. Στην περίπτωση των μοντέλων ευέλικτης ανάπτυξης οι δραστηριότητες ελέγχου είναι πιο συχνές και επικεντρώνονται στην ικανοποίηση των συμμετεχόντων. Υπάρχουν ακόμα και προσεγγίσεις με έλεγχο των απαιτήσεων των συμμετεχόντων πριν την ανάπτυξη, συνεπώς η ωριμότητα των διαδικασιών ανάπτυξης λογισμικού συνεχώς αυξάνεται με επίγνωση και με τις εξελιγμένες πρακτικές δοκιμών.

Παρόλα αυτά τα περιοδικά μοντέλα ανάπτυξης, η αυξητική προτυποποίηση και η γρήγορη εφαρμογή της ανάπτυξης πρόσφεραν μηχανισμούς για τον έλεγχο, τον συγχρονισμό και την ποιότητα των προϊόντων. Η προτυποποίηση αυξάνει την πιθανότητα της επίτευξης της ολοκλήρωσης λόγο της αρχής της παράδοσης λειτουργικών προϊόντων σύμφωνα με τις απαιτήσεις του χρήστη. Η ανάγκη για την συνέχιση της βελτίωσης ευρέως αναγνωρίστηκε και ενθυλακώθηκε από την ιδέα του Kaizen που στα Ιαπωνικά σημαίνει “πολλές μικρές βελτιώσεις”.

Σύμφωνα με τον Beck τα μοντέλα με παραδοσιακούς κύκλους ζωής είναι ανεπαρκής και πρέπει να αντικατασταθούν από αυξητικά σχέδια και γρήγορη προτυποποίηση, χρησιμοποιώντας ένα μοντέλο από την αρχή της σύλληψης της ιδέας μέχρι τον κώδικα μέσο της ανάλυσης, του σχεδιασμού και της εφαρμογής. Με άλλα λόγια η σχεδίαση αρχίζει ενώ η ανάλυση δεν έχει τελειώσει ακόμα και η κωδικοποίηση ξεκινάει αμέσως μετά την αρχή της λεπτομερείς σχεδίασης. Τα στάδια του σχεδίου ελέγχονται μαζί με την εφαρμογή. Η Extreme Programming για παράδειγμα είναι μια από τις ευέλικτες ή ελαφριές μεθοδολογίες για να δείξει την απομάκρυνση από τους πολλούς κανόνες και πρακτικές.

Η προσέγγιση της XP θεωρείτε επιτυχής από τότε που έδωσε έμφαση στις απαιτήσεις του πελάτη πάνω στο σχεδιασμό, στον έλεγχο, στην εφαρμογή, στην μελλοντική και τρέχων συμμετοχή του πελάτη και στην ανατροφοδότηση. Η συγκεκριμένη μέθοδο σχεδιάστηκε για να παραδίδει το λογισμικό στον πελάτη με τις απαιτήσεις του πελάτη και στην ώρα που το χρειάζεται ο πελάτης. Η XP (όπως και οι περισσότερες άλλες ευέλικτες μέθοδοι) δίνει δικαιώματα στους κατασκευαστές να ανταποκριθούν με βεβαιότητα και να ελέγξουν τις αλλαγές στις απαιτήσεις του πελάτη ακόμα και αργά μέσα στον κύκλο ζωής. Η αποδοχή των αλλαγών επίσης δίνει δικαιώματα και στους πελάτες και στους τελικούς χρήστες, αλλά η πρόσθεση απαιτήσεων μήπως αθετεί τις συμφωνημένες απαιτήσεις και μήπως ακυρώνει τις προσεκτικά ελεγμένες και τεκμηριωμένες απαιτήσεις από τις προηγούμενες προσπάθειες μοντελοποίησης; Εκτός βέβαια αν και οι συνεχείς αναδομήσεις και επανακατασκευές των προκαθορισμένων απαιτήσεων ενισχύουν αντιθέτως την τεχνική μελέτη και την επαναπαραγωγή.

3. Extreme Programming (XP)

3.1 Εισαγωγή[9]

Η Extreme Programming μέθοδος είναι μια ελαφριά διαδικασία ανάπτυξης λογισμικού για μικρές ομάδες οι οποίες ασχολούνται σε την ασάφεια ή με την γρήγορη αλλαγή των απαιτήσεων. Η XP διαχωρίζεται σε έναν αριθμό από παραδοσιακές πρακτικές της τεχνολογίας λογισμικού. Πρώτον, η τεκμηρίωση είναι σχεδόν εξολοκλήρου ανύπαρκτη. Η μόνη τεκμηρίωση είναι μια συλλογή από ενδεικτικές κάρτες στις οποίες τα μέλη της ομάδας γράφουν πρόχειρα τα χαρακτηριστικά του συστήματος. Εκτός από αυτό, ο πηγαίος κώδικας είναι η μόνη τεκμηρίωση. Η λογική είναι ότι γράφοντας την τεκμηρίωση καθυστερεί τους κατασκευαστές και παραμελείται συνήθως. Δεύτερον, δεν υπάρχει καμία προδιαγραφή για το λογισμικό. Εκτελέσιμες δοκιμαστικές περιπτώσεις, γράφονται προτού ο κώδικας αναπτυχθεί και χρησιμοποιούνται σαν υποκατάστατο. Τρίτον, δεν υπάρχει διαχωρισμός μεταξύ σχεδιασμού και δοκιμαστικής φάσης. Αντί αυτού, ο σχεδιασμός, η υλοποίηση και η δοκιμή γίνονται ταυτόχρονα με μικρές προσαυξήσεις. Τέταρτον, υπάρχει μια σαφή απαγόρευση ενάντια στο σχέδιο για αλλαγή, μόνο το πιο απλό σχέδιο που ικανοποιεί τα χαρακτηριστικά την δεδομένη στιγμή πρέπει να υλοποιηθεί. Πέμπτον, δεν υπάρχει καμία επίσημη αναθεώρηση ή επιθεώρηση. Η extreme Programming μέθοδος καθορίζει μια συλλογή από πρακτικές και αξίες που παρατηρήθηκαν κατά την εργασία καλών ομάδων ανάπτυξης λογισμικού, που χρησιμοποιήθηκαν σε ακραίες καταστάσεις και με έναν συντονισμένο τρόπο εργασίας. Οι απαιτήσεις και ο βαθμός ανάπτυξης των χαρακτηριστικών αποφασίζονται σταδιακά. Η Extreme Programming μέθοδος είναι σχεδιασμένη για μια ομάδα κατασκευαστών λογισμικού που να τους κάνει μια παραγωγική μονάδα που δέχεται τις αλλαγές και τις ενσωματώνει γρήγορα σε ένα αναπτυσσόμενο σύστημα.

Η Extreme Programming μέθοδος όμως έχει και κάποιες αδυναμίες. Από την μεριά της Μηχανικής του λογισμικού οι πιο σημαντικές είναι:

- Στην XP μέθοδο υπάρχουν τρεις πηγές γνώσης σχετικά με το λογισμικό ώστε να συντηρηθεί: ο κώδικας, οι δοκιμές, και η μνήμη των προγραμματιστών. Εάν το λογισμικό παραμένει χωρίς αλλαγές για μια μεγάλη χρονική περίοδο, οι προγραμματιστές θα ξεχάσουν πολλά σημαντικά πράγματα και ακόμα χειρότερα κάποιοι από τους προγραμματιστές μπορεί να μην είναι διαθέσιμοι(για παράδειγμα μπορεί να έχουν μετακομίσει σε άλλη χώρα).Τότε, η μόνη βάση για συντήρηση είναι ο κώδικας και οι δοκιμές που μπορούν να κάνουν την συντήρηση δύσκολη. Θα ήταν πιο εύκολο αν κάποιος είχε τις απαιτήσεις τεκμηριωμένες. Τότε οι προγραμματιστές θα μπορούσαν να τις χρησιμοποιήσουν σαν οδηγό στην ανάγνωση του κώδικα και στις δοκιμές. Κατά συνέπεια προκύπτει η ακόλουθη ερώτηση: είναι δυνατόν να έχουμε ευέλικτη μεθοδολογία και τεκμηριωμένες απαιτήσεις;

- Η XP μέθοδος βασίζεται στην προφορική επικοινωνία. Όμως η προφορική επικοινωνία είναι επιρρεπής σε λάθη μνήμης. Μετά από μερικές φορές κάποιος μπορεί να έχει προβλήματα στο να επαναφέρει από την μνήμη του ποια εναλλακτική ήταν τελικά επιλεγμένη και γιατί, ειδικά όταν ένα σύστημα είναι πολύπλοκο, υπάρχουν πολλές απόψεις, πολλές αντικαταστάσεις και δισταγμοί.
- Η XP μέθοδος υποθέτει ότι υπάρχει μόνο ένας αντιπρόσωπος πελατών. Εάν υπάρχουν πολλοί υποθέτει ότι μιλούν με μια φωνή. Τι γίνεται όμως αν έχουν διαφορετικές απόψεις; Πως πρέπει να τροποποιηθεί η XP ώστε αυτό να το λάβουμε υπόψη;
- Στην XP η προοπτική του σχεδιασμού είναι σχετικά μικρή, μόνο μια έκδοση(αν είναι δυνατόν μια έκδοση να παίρνει όχι πάνω από 2 μήνες για να βγει. Κάποιοι το θεωρούν αυτό σαν ένα μειονέκτημα. Χρόνος και χρήμα μπορεί να εξασφαλιστεί εάν αυτοί που συμμετέχουν(αντιπρόσωποι πελατών και κατασκευαστές) σπαταλήσουν από την αρχή του έργου, μια με δυο βδομάδες συζητώντας τον σκοπό και τι είναι εφικτό για όλο το έργο. Άρα πως είναι δυνατών να τροποποιηθεί η XP ώστε να εξασφαλίσει μια πιο ευρεία προοπτική ενός έργου;
- Η XP εξαρτάται από την συχνότητα με την οποία γίνονται αυτοματοποιημένες δοκιμές. Αυτοματοποιημένες περιπτώσεις δοκιμών πρέπει όχι μόνο να κατασκευάζονται αλλά να συντηρούνται κιόλας. Η συντήρηση των περιπτώσεων δοκιμών είναι ένα σοβαρό πρόβλημα ειδικά στην περίπτωση των συχνών αλλαγών. Πως μπορεί να υποστηριχθεί η συντήρηση των περιπτώσεων δοκιμών;

3.2XP Vs Πρότυπο Ικανότητας Ωρίμανσης (XP vs. Capability Maturity Model)[9]

Πιθανόν το δημοφιλέστερο πρότυπο που έχει σχέση με την διαδικασίας της αξιολόγησης του λογισμικού είναι το Πρότυπο Ικανότητας Ωρίμανσης(Capability Maturity Model εν συντομία CMM). Σε αυτό το μοντέλο υπάρχουν πέντε στάδια ωρίμανσης και το κάθε επίπεδο περιλαμβάνει έναν αριθμό από Βασικές Περιοχές Διαδικασίας(Key Process Areas εν συντομίας KPA). Στο δεύτερο επίπεδο που ονομάζεται Επαναληπτικό, υπάρχουν έξι KPA και ένα από αυτά είναι η Διαχείριση των Απαιτήσεων. Πρόσφατα παρουσιάστηκε μια ενδιαφέρον αποτίμηση της XP από την άποψη του CMM. Σε αυτήν η διαχείριση των απαιτήσεων, όπως είναι ορισμένες στο CMM, είναι σε μεγάλο βαθμό εξετασμένες στην XP.

Η XP εξετάζει την διαχείριση των απαιτήσεων του 2ου επιπέδου μέσο της χρήσης των πλάνων του πελάτη και της συνεχής ενσωμάτωσης. Το πλάνο ενός χρήστη αποτελείται από δυο συστατικά, την γραπτή κάρτα και μια σειρά από συζητήσεις που γίνονται αφότου γραφούν οι κάρτες. Οι γραφόμενες κάρτες είναι απλά υποσχέσεις για συζητήσεις και δεν χρειάζεται να ολοκληρωθούν ή να καθοριστούν πλήρως. Εάν κάτι δεν είναι ξεκαθαρισμένο, θα διευκρινιστεί με τις συζητήσεις με τον πελάτη. Εάν ένα πλάνο είναι επιτυχώς υλοποιημένο μπορεί η κάρτα αυτού του πλάνου να καταστραφεί. Το τρίτο στοιχείο, η συνεχή ενσωμάτωση είναι πιο σημαντική για την διασφάλιση της ποιότητας παρά για την διαχείριση των απαιτήσεων. Το ενδιαφέρον στο πλαίσιο των απαιτήσεων είναι οι μικρές εκδόσεις, οι οποίες επιτρέπουν γρήγορες ανατροφοδοτήσεις, το οποίο είναι ο καλύτερος τρόπος για την επικύρωση των απαιτήσεων.

Η κύρια αδυναμία της προσέγγισης της XP στην διαχείριση των απαιτήσεων είναι η έλλειψη της τεκμηρίωσης των απαιτήσεων. Από μια άποψη η XP μοιάζει με την εξελικτική διαμόρφωση πρωτοτύπου. Όπου ένα πρότυπο αναπτύσσεται ραγδαία στο τελικό προϊόν.

Η γραφή τεκμηριωμένων απαιτήσεων είναι επίσης σημαντική για το CMM. Στο 2ο επίπεδο υπάρχουν μεταξύ άλλων τα ακόλουθα κριτήρια που αφορούν μια σωστή διαδικασία διαχείρισης απαιτήσεων.

- **Οι προσδιορισμένες απαιτήσεις είναι τεκμηριωμένες:** Δυστυχώς στην XP δεν υπάρχει τεκμηρίωση απαιτήσεων. Οι κάρτες με το πλάνο είναι σαν παγόβουνο, αυτό που βλέπεις είναι ένα μικρό κομμάτι από αυτό που θα πάρεις.
- **Αλλαγές σαν αποτέλεσμα αλλαγών στις προσδιορισμένες απαιτήσεις είναι τεκμηριωμένες:** Στην XP οι αλλαγές δεν είναι τεκμηριωμένες. Εάν η επιχείρηση αντιληφθεί ότι χρειάζεται ένα νέο πλάνο στο μέσον της ανάπτυξης μια έκδοσης, μπορεί να γράψει το πλάνο αυτό. Η ανάπτυξη εκτιμάει το πλάνο, μετά η επιχείρηση αφαιρεί τα πλάνα με ισοδύναμη εκτίμηση από το υπόλοιπο σχέδιο και εισάγει το νέο πλάνο.

Η προσέγγιση της XP στη διαχείριση των απαιτήσεων, η οποία βασίζεται κυρίως στην προφορική επικοινωνία δεν είναι αποδεκτή από την μεριά του CMM. Η στέρηση της τεκμηρίωσης περισσότερο πλήττει την συντήρηση. Υπάρχουν δύο είδη συντήρησης: η συνεχής και η ειδική. **Η συνεχή συντήρηση** σημαίνει ότι οι εκθέσεις ατέλειας και οι διαταγές αλλαγών εμφανίζονται αρκετά συχνά και είναι λογικό να έχει καθοριστεί μια ομάδα για την εργασία της συντήρησης. Μπορούν να παράγουν νέες διορθώσεις και να απελευθερώνουν νέες εκδόσεις του λογισμικού σε σχεδόν περιοδική βάση. Εάν το λογισμικό δεν είναι πολύ μεγάλο τότε η XP είναι μια καλή επιλογή. **Στην ειδική συντήρηση** οι εκθέσεις ατέλειας και οι διαταγές αλλαγών εμφανίζονται σπάνια. Όταν μια ανάγκη προκύψει ένα σχέδιο ενεργοποιείται. Πολλά από τα 2000ων χρόνων προβλήματα έχουν ξεχωριστά σχέδια συντήρησης. Εάν η χρονική απόσταση από ένα σχέδιο συντήρησης σε ένα άλλο είναι αρκετά μεγάλη, τότε η έλλειψη τεκμηρίωσης μπορεί να γίνει σοβαρό πρόβλημα.

3.3XP VS Sommerville-Sawyer Πρότυπο [9]

Η ωριμότητα της διαδικασίας της μηχανικής των απαιτήσεων εφαρμοσμένη σε μια οργάνωση μπορεί να καθοριστεί σύμφωνα με το Sommerville-Sawyer πρότυπο. Αυτό είναι ένα απλό τρίτου επιπέδου μοντέλο βασισμένο σε καλές πρακτικές. Οι Sommerville και Sawyer έχουν αναγνωρίσει 66 πρακτικές και τις διαχώρισαν σε 3 ομάδες: βασικές, ενδιάμεσες και προηγμένες. Κάθε πρακτική μπορεί να έχει 0 έως 3 πόντους, ανάλογα με το πόσο συχνά χρησιμοποιούνται σε μια οργάνωση(3 πόντους αν η πρακτική είναι πρότυπο της οργάνωσης, 2 αν είναι συχνά χρησιμοποιούμενη, 1 αν η χρήση της είναι προαιρετική, 0 αν η πρακτική δεν χρησιμοποιείται). Το υψηλότερο επίπεδο ωρίμανσης λέγεται **Ορισμένο.** Οι οργανώσεις σε αυτό το επίπεδο έχουν πάνω από 85 πόντους στις βασικές πρακτικές και πάνω από 40 στις ενδιάμεσες και προηγμένες πρακτικές. Το ενδιάμεσο επίπεδο λέγεται **Επαναλαμβανόμενο** και οι οργανώσεις σε αυτό το επίπεδο έχουν πάνω από 50 πόντους στις βασικές πρακτικές. Το κατώτερο επίπεδο λέγεται **Αρχικό** και μια αρχική οργάνωση έχει λιγότερους από 55 πόντους στις βασικές πρακτικές.

Χρησιμοποιήσαμε το Sommerville-Sawyer πρότυπο για να εκτιμήσουμε την προσέγγιση στη διαχείριση των απαιτήσεων που προτείνει η XP. Υπάρχουν μόνο επτά βασικές πρακτικές στο Sommerville-Sawyer πρότυπο που είναι πλήρως υποστηριζόμενες από την XP:

1. Αξιολόγηση της επιτευξιμότητας του συστήματος.
2. Χρήση των ενδιαφερόντων της επιχείρησης για την απόκτηση των απαιτήσεων.
3. Σχέδιο για τις διαμάχες και την ανάλυση αυτών.
4. Αντιστοίχηση προτεραιοτήτων στις απαιτήσεις.
5. Διαμόρφωση της αρχιτεκτονικής του συστήματος.
6. Μεμονωμένη αναγνώριση κάθε απαίτησης.
7. Καθορισμός επιβλεπόντων για την διαχείριση απαιτήσεων.

3.4Γεφύρωμα της XP με την τεκμηρίωση απαιτήσεων [9]

Με την πρώτη ματιά η XP και η γραφή τεκμηρίωσης δεν ταιριάζουν μαζί καθόλου. Όπως έχουμε αναφέρει και πιο πάνω η XP είναι μια ελαφριά μεθοδολογία η οποία προτιμάει την προφορική επικοινωνία παρά την γραπτή. Ωστόσο ένα ερώτημα προκύπτει: Τι σημαίνει ότι μια μεθοδολογία ανάπτυξης είναι ελαφριά; Για ποίον είναι ελαφριά; Η XP δεν είναι σίγουρα μια ελαφριά μεθοδολογία από την μεριά του πελάτη αφού η XP χρειάζεται την εκ όψεως επαφή με τον πελάτη που θα εργάζεται με τους υπεύθυνους της ανάπτυξης(η κλασσική προσέγγιση είναι πολύ πιο ελαφριά από αυτή την άποψη). Το κύριο πλεονέκτημα της XP είναι οι προγραμματιστές, η μόνη απασχόληση τους είναι να γράψουν και να δοκιμάσουν περιπτώσεις και κώδικα. Κατά συνέπεια αν κάποιος θέλει να γεφυρώσει μια ελαφριά μεθοδολογία με την γραπτή τεκμηρίωση των απαιτήσεων, δεν μπορεί να ρίξει το βάρος της τεκμηρίωσης στους προγραμματιστές.

Μια καλή πρόταση είναι να κάνει τους δοκιμαστές υπεύθυνους για την τεκμηρίωση και την διαχείριση των απαιτήσεων.Σύμφωνα με κάποιους, οι δοκιμαστές στην XP είναι υπεύθυνοι ώστε να βοηθούν τον πελάτη να διαλέξει και να γράψει λειτουργικά τεστ και να τα πραγματοποιεί σε συχνή βάση. Οι απαιτήσεις και η έγκριση των τεστ είναι στο ίδιο επίπεδο έτσι οι δοκιμαστές φαίνεται να είναι τα καταλληλότερα άτομα να αναλάβουν την ανάλυση της τεκμηρίωσης και της διαχείρισης των απαιτήσεων. Μπορούμε αυτό το άτομο να το ονομάζουμε δοκιμαστή και αναλυτή. Συγχωνεύοντας τον ρόλο του δοκιμαστή και του αναλυτή μοιάζει λογικό γιατί οι περιπτώσεις δοκιμών πρέπει να διαχειρίζονται όπως και να έχει. Είναι ιδιαίτερα δύσκολο όταν οι απαιτήσεις προχωρούν αργά και το σύστημα είναι αρκετά πολύπλοκο(όσο πιο πολύπλοκο είναι το σύστημα τόσο πιο αργά προχωρούν οι απαιτήσεις, καθώς κανένας δεν μπορεί να τις φτιάξει από την αρχή του έργου). Μια καλή πρακτική είναι να συνδεθούν οι περιπτώσεις δοκιμών με τις απαιτήσεις. Εάν οι απαιτήσεις αλλάξουν, κάποιος θα ξέρει ποιες περιπτώσεις δοκιμών πρέπει να αναθεωρηθούν. Λόγω αυτών ένας καλά οργανωμένος δοκιμαστής θα μπορεί να διαχειριστεί τις απαιτήσεις. Εάν οι απαιτήσεις δεν είναι τεκμηριωμένες, η δουλεία του είναι πιο δύσκολη. Κάποιες προτάσεις για τον δοκιμαστή/αναλυτή στην XP είναι οι εξής:

- **Χρήση προηγμένων εργαλείων χωρίς φόβο.**
- **Οργάνωση των απαιτήσεων σε πολλαπλά επίπεδα.**
- **Διαθεσιμότητα των συλλεγόμενων απαιτήσεων στην ομάδα και στον πελάτη μέσω του διαδικτύου.**
- **Προσεκτική επιλογή χαρακτηριστικών για τις απαιτήσεις.**
- **Μεταφορά όλων των συνημμένων που παρέχονται από τον πελάτη σε ηλεκτρονική μορφή.**

3.5Πολλαπλοί εκπρόσωποι των πελατών[9]

Μια από τις πρακτικές της XP είναι ο εκ όψεως πελάτης. Είναι ένας πραγματικός πελάτης (τελικός χρήστης) και αναμένεται να κάτσει με την ομάδα, απαντώντας σε ερωτήσεις και να πάρει τις αποφάσεις για την επιχείρηση. Ιδιαίτερα, συμμετέχει στο σχεδιασμό της τακτικής η οποία αποσκοπεί στην επόμενη έκδοση και στην απόφαση της επόμενης αναδρομής.

Η XP υποθέτει, ότι υπάρχει μόνο ένας αντιπρόσωπος του πελάτη για το έργο. Στην περίπτωση που υπάρχουν πολλοί αντιπρόσωποι του πελάτη υποτίθεται ότι έχουν την ίδια γνώμη. Κάποιες συμβουλές από την Sommerville και Sawyer μέθοδο η οποία δείχνει την ανάγκη για να ληφθούν υπόψη οι πολλοί αντιπρόσωποι του πελάτη είναι:

- **Αναγνώριση και υπολογισμός των ατόμων που συμμετέχουν στο σύστημα.** Είναι σημαντικό να κάνουμε τους ανθρώπους να αισθάνονται ότι συμμετέχουν στη δημιουργία του συστήματος.
- **Συλλογή απαιτήσεων από διαφορετικές οπτικές πλευρές.**

- **Ευαισθησία στην οργάνωση και την πολιτική της εκτίμησης.** Εάν ένα σύστημα προορίζεται να εξυπηρετήσει πολλά τμήματα σε έναν οργανισμό, μια κοινή πρακτική είναι να συμπεριληφθούν στην διαδικασία των αποφάσεων ένας αντιπρόσωπος από κάθε ένα τμήμα.
- **Σχέδιο για τις διαμάχες και την ανάλυση αυτών.** Οι συγκρούσεις είναι αναπόφευκτες εάν ένα σύστημα εξυπηρετεί πολλούς ανθρώπους με διαφορετικές προσδοκίες και φόβους.

Οι προτάσεις που έχουν παρουσιαστεί για το πώς μπορούμε να συμπεριλάβουμε περισσότερους από έναν αντιπρόσωπους στη διαδικασία της ανάπτυξης είναι:

- **Οι δοκιμαστές/αναλυτές απαντούν στις ερωτήσεις των προγραμματιστών.** Εάν υπάρχουν πολλοί αντιπρόσωποι πελατών δεν μπορούν να καθίσουν με την ομάδα περιμένοντας να απαντήσουν σε ερωτήσεις των προγραμματιστών. Ο δοκιμαστής/αναλυτής είναι ο καλύτερος άνθρωπος για την δουλειά αυτή.
- **Οι δοκιμαστές/αναλυτές έχουν καθημερινή επικοινωνία με τους αντιπροσώπους του πελάτη.** Αναπτύσσει με αυτούς αποδεκτά τεστ και τους παρουσιάζει τις αναφορές για την πρόοδο που γίνεται (επιτυχία ή αποτυχία στα αποδεκτά τεστ).
- **Οι αντιπρόσωποι των πελατών συμμετέχουν στην τροποποίηση του σχεδιασμού της τακτικής.**

3.6Τροποποίηση του σχεδιασμού της τακτικής(Modified Planning Game)[9]

Υποθέτουμε ότι από την μεριά του πελάτη υπάρχουν πολλοί αντιπρόσωποι του πελάτη και ένας κύριος/ανώτερος αντιπρόσωπος του πελάτης. Κάθε αντιπρόσωπος του πελάτη έχει κάποια συγκεκριμένη γνώση και θα είναι ο τελικός χρήστης του συστήματος. Είναι πιθανόν οι απαιτήσεις του να έρχονται σε αντιπαράθεση με τις απαιτήσεις των άλλων αντιπροσώπων του πελάτη.

Ο κύριος/ανώτερος αντιπρόσωπος του πελάτης εκπροσωπεί τα ενδιαφέροντα όλου του οργανισμού. Μπορεί να μην έχει την κύρια γνώση που οι αντιπρόσωποι του πελάτη έχουν ούτε τον χρόνο για να κάτσει να συζητήσει τις λεπτομέρειες του συστήματος που φτιάχνεται. Γι’ αυτό το λόγο είναι απαραίτητοι και ο κύριος/ανώτερος και οι αντιπρόσωποι του πελάτη. Ο κύριο ρόλος του κύριου/ανώτερου αντιπροσώπου του πελάτη είναι να λύνει τις διαφορές μεταξύ υπολοίπων αντιπροσώπων του πελάτη όταν αυτό χρειάζεται.

Όπως και στον κύριο σχεδιασμό της τακτικής, η πρώτη κίνηση ανήκει στον οργανισμό πχ στους αντιπροσώπους του πελάτη. Φέρνουν μαζί τις κάρτες με το πλάνο τους.

Η δεύτερη κίνηση γίνεται από τους υπεύθυνους ανάπτυξης. Όπως και στον κύριο σχεδιασμό της τακτικής, οι υπεύθυνοι ανάπτυξης προσπαθούν να υπολογίσουν τον ιδανικό χρόνο ανάπτυξης(σε ώρες) και να αξιολογήσουν τον κίνδυνο που συνδέεται με το κάθε πλάνο.

Η τρίτη κίνηση είναι του οργανισμού που θα αποφασίσει για τον στόχο. Αυτή η κίνηση τροποποιείται εάν υπάρχουν πολλοί αντιπρόσωποι του πελάτη αντί για ένας. Κάθε αντιπρόσωπος του πελάτη παίρνει έναν υπολογισμό του χρόνου που περίπου η ομάδα καταναλώσει δουλεύοντας μαζί του. Ο υπολογισμός αυτός εκφράζεται σε ώρες. Αυτό προσδιορίζεται στον κάθε αντιπρόσωπο του πελάτη από τον κύριο/ανώτερο αντιπρόσωπο. Κάθε πελάτης αγοράζει το πιο καλό πλάνο από τους υπεύθυνος ανάπτυξης. Εάν χ αντιπρόσωποι του πελάτη αγοράζουν ένα πλάνο, κάθε ένας θα πληρώσει το 1/χ της τιμής.

Εάν κάποιος αντιπρόσωπος δεν είναι διαθέσιμος να αγοράσει ένα καλό πλάνο, μπορεί να επιστρέψει τον προϋπολογισμό του στον κύριο/ανώτερο εκπρόσωπο. Τότε ο κύριος/ανώτερος εκπρόσωπος προσφέρει αυτά τα χρήματα στους υπόλοιπους εκπροσώπους. Αυτό είναι ένα είδος πλειοδοσίας. Το άτομο που θα προσφέρει την υψηλότερη τιμή θα πάρει αυτά τα χρήματα για να μπορέσει αγοράσει και άλλο πλάνο. Η τιμή αυτή θα καταβληθεί στον επόμενο σχεδιασμό τακτικής στο άτομο που παραιτήθηκε από αυτούς.

Εάν υπάρχουν πολλοί αντιπρόσωποι είναι λογικό να χρειάζεται περισσότερο χρόνο η κάθε έκδοση(πχ 3 μήνες διάρκεια) να μπορέσει να υλοποιήσει τα προτεινόμενα πλάνα. Στο επαυξητικό επίπεδο είναι χρήσιμο ο σχεδιασμός της επαύξησης ενός πλάνου να προέρχεται από τον ίδιο αντιπρόσωπο.

3.7Πρακτικές XP[16]

Όπως αναφέρθηκε και πιο πάνω η Extreme Programming μέθοδος είναι μια συλλογή από πρακτικές και αξίες. Αυτές οι πρακτικές και αξίες είναι: Αλληγορία(Metaphor), Συλλογικός Κώδικας(Collective Code Ownership), Απλός Σχεδιασμός(Simple Design), Επαναπαραγωγή(Refactoring), Προγραμματισμός Έκδοσης(Release Planning), Μικρές Εκδόσεις(Small Releases), Συνεχείς Ενσωματώσεις(Continuous Integration), Εκ όψεως επαφή με τον πελάτη(On Site Costumer), Δοκιμές(Testing), Προγραμματισμός με ζευγάρι(Pair Programming), Πρότυπα κωδικοποίησης(Coding standard), 40 ώρες την Εβδομάδα(40 hour Week).

Αλληγορία(Metaphor)[16]: Η XP(Extreme Programming) υιοθετεί ότι κάθε εφαρμογή έχει μια εννοιολογική ακεραιότητα βασισμένη σε μια απλή αλληγορία, και αυτό εξηγεί την ουσία για το πως το σύστημα δουλεύει. Η ιδέα της αλληγορίας του συστήματος είναι πολύ αφηρημένη και συχνά μυστηριώδη στους έμπειρους επαγγελματίες.

Συλλογικός Κώδικας(Collective Code Ownership) [16]: Σε μια ομάδα ανάπτυξης που χρησιμοποιεί ΧΡσε κανένα μεμονωμένο προγραμματιστή δεν ανήκει κανένα κομμάτι κώδικα. Μόλις μπει κάτι στη βάση του κώδικα, σε κάθε μέλος τις ομάδας ανήκει ο κώδικας και μπορεί να αλλάξει τον κώδικα χωρίς να πάρει την άδεια κανενός.

Απλός Σχεδιασμός(Simple Design) [16]: Η ΧΡ μέθοδος πιέζει τους προγραμματιστές να μην επιχειρήσουν να προβλέψουν μελλοντικές ανάγκες και να σχεδιάσουν ανάλογα. Άλλη μια πλευρά του Απλού Σχεδιασμού είναι «Ο πηγαίος κώδικας είναι το σχέδιο». Δεν απαιτείται τεκμηριωμένο σχέδιο, σε αντίθεση με το SPMP(Software Project Management Plan) έγγραφο , το οποίο απαιτείται για το έργο από την πρώτη φάση.

Επαναπαραγωγή(Refactoring) [16]: Είναι η διαδικασία κατά την οποία βελτιώνεται η δομή του κώδικα καθώς διαφυλάσσονται οι λειτουργίες του. Η ΧΡ υποστηρίζει τον επαναπαραγωγικό κώδικα συνεχώς και κατηγορηματικά.

Προγραμματισμός Έκδοσης(Release Planning) [16]: Πρώτον, οι απαιτήσεις του πελάτη είναι γραμμένες σε φυσική γλώσσα, άτυπα οι κάρτες με το πλάνο του χρήστη(User Story Cards), παρομοιάζονται με τις περιπτώσεις χρήσης(Use Case). Οι καταγεγραμμένοι χρόνοι των υπεύθυνων για την ανάπτυξη λογισμικού υπολογίζονται για την κάθε κάρτα και για τις προσδιορισμένες προτεραιότητες του πελάτη σε κάθε κάρτα. Στο σχεδιασμό της τακτικής(Planning Game) ο πελάτης επιλέξει εκείνες τις κάρτες με το πλάνο του χρήστη που περιλαμβάνουν τα πιο σημαντικά περιεχόμενα για μικρό χρονικό διάστημα και αφού προστεθούν παραδίδονται σε διάστημα περίπου ενός μηνός. Αυτά τα μικρά κομμάτια προγράμματος που προστίθενται είναι αποδεκτά και τα δοκιμάζει ο πελάτης. Ύστερα οι υπόλοιπες κάρτες επανεξετάζονται για πιθανές απαιτήσεις και/ή γίνεται αλλαγή προτεραιοτήτων, και ο σχεδιασμός της τακτικής ξανά-ενεργοποιείται για την επόμενη εφαρμογή της πρόσθεσης νέων κομματιών.

Συνεχείς Ενσωματώσεις(Continuous Integration) [16]: Η κωδικοποίηση χωρίζεται σε μικρές εργασίες, κατά προτίμηση όχι περισσότερο από μια ημέρα. Όταν κάθε εργασία ολοκληρωθεί, ενώνεται με την κοινή βάση κώδικα. Αυτό έχει σαν αποτέλεσμα πολλά προϊόντα να κατασκευάζονται καθημερινά.

Εκ όψεως επαφή με τον πελάτη(On Site Costumer) [16]: Οι πελάτες είναι πάντα πρόθυμα διαθέσιμοι και προσβάσιμοι στους κατασκευαστές με σκοπό να διευκρινίσουν και να επικυρώσουν τις απαιτήσεις. Προτιμότερο είναι αυτές οι επαφές να γίνονται πρόσωπο με πρόσωπο.

Δοκιμές(Testing) [16]: Εκτεταμένες, αυτοματοποιημένες περιπτώσεις δοκιμών λευκού κουτιού είναι γραμμένες πριν ακόμα παραχθεί ο κώδικα και προστεθεί στη βάση του κώδικα. Πριν οι προγραμματιστές μπορέσουν να ενσωματώσουν τον κώδικα τους στη βάση, πρέπει να περάσουν 100% τις δικές τους περιπτώσεις δοκιμών και 100% οποιασδήποτε δοκιμής είχαν γράψει για την βάση του κώδικα. Δυστυχώς, έχει αναφερθεί ότι συχνά δεν μπορεί να γραφούν οι περιπτώσεις δοκιμών πριν γραφεί ο κώδικας κάτι το οποίο έρχεται σε αντίθεση με την ΧΡ.

Πρότυπα κωδικοποίησης(Coding standard) [16]: Για να μπορούν οι κατασκευαστές να καταλαβαίνουν ο ένας τον κώδικα του άλλου, ακολουθείται ένα συμφωνημένο ανώτερο πρότυπο κωδικοποίησης.

40 ώρες την Εβδομάδα(40 hour Week) [16]: Η XP συνηγορεί ότι οι προγραμματιστές δεν πρέπει να εξαντλούνται από την υπερβολική εργασία. Η ομάδα αποφασίζει να δουλεύει με ρυθμό ο οποίος είναι βολικός για όλα τα μέλη της ομάδας. Οι κατασκευαστές παρατήρησαν ότι κατά την διάρκεια κρίσιμων περιόδων που δούλευαν υπερορίες, τα προϊόντα που παρήγαγαν ήταν φτωχά.

4. Ο κύκλος ζωής των XP έργων(The XP projects lifecycle)

Η XP παρέχει ένα πρότυπο κύκλου ζωής στην ανάπτυξη λογισμικού όπως επίσης και οδηγίες για την οργάνωση μίας ομάδας ανάπτυξης λογισμικού.

4.1Πλάνο του Χρήστη(User Stories)[18]

Τα πλάνα του χρήστη είναι παρόμοια με τις περιπτώσεις χρήστη στο ότι και τα δυο στοχεύουν στον να συλλάβουν το πώς ο χρήστης θα χρησιμοποιήσει το σύστημα. Ωστόσο γράφονται από τον χρήστη και όχι από την ομάδα ανάπτυξης. Πρέπει επίσης να σημειωθεί ότι ο χρήστης δεν πρέπει να αποκοπεί και να περιγράψει μόνο το περιβάλλον εργασίας του χρήστη. Είναι προτιμότερο το πλάνο του χρήστη να πρέπει να περιγράψει πως ο χρήστης θα χρησιμοποιήσει το σύστημα για να επιτύχει κάτι.

Δεν ωφελεί σε τίποτα το ότι τα πλάνα του χρήστη δεν είναι λεπτομερείς απαιτήσεις. Αυτές θα αποκτηθούν κατά τις επαναλήψεις, όταν και αν η προοπτική του συστήματος συμπεριληφθεί από ένα λεπτομερές πλάνο χρήστη που πρόκειται να εφαρμοστεί. Αντί αυτού, τα πλάνα του χρήστη πρέπει να είναι τόσο συγκεκριμένα όσο να επιτρέπουν να γίνει μια λογική εκτίμηση του πόσος χρόνος θα χρειαστεί για να ολοκληρωθούν κάτω από ιδανικές συνθήκες μετά από την προγραμματισμένη συνάντηση.

Πρέπει να είναι σύντομα και πρέπει να έχουν την ορολογία που χρησιμοποιεί ο χρήστης και όχι η ομάδα ανάπτυξης. Αυτά τα πλάνα χρήστη πρέπει να ενσωματωθούν μέσα στις προγραμματισμένες συναντήσεις και στα τεστ που είναι αποδεκτά από τον χρήστη.

4.2Αρχιτεκτονική αιχμή(Architectural Spike)[18]

Η αιχμή σαν όρος της XP είναι μια προσπάθεια για να μειωθεί το ρίσκο το οποίο σχετίζεται με μια άγνωστη περιοχή του συστήματος, της τεχνολογίας ή της εφαρμογής. Μια αιχμή μπορεί να περιλαμβάνει κάποια έρευνα, αναζήτηση και πιθανόν κάποιο λογισμικό για να αξιολογήσει ή να εξαλείψει το λάθος/πρόβλημα. Τα αποτελέσματα από τις περισσότερες αναζητήσεις είναι συνήθως όχι και τόσο καλά ώστε να κρατηθούν και έτσι πρέπει να απορρίπτονται. Έγκαιρα λοιπόν στην συνολική αρχιτεκτονική του συστήματος που πρέπει να αναπτυχθεί αποτελούν ένα σοβαρό θέμα που πρέπει να λυθεί. Επομένως σε αυτό το στάδιο της έρευνας και της ανάλυσης της αρχιτεκτονικής που θα χρησιμοποιηθεί πρέπει να εφαρμοστούν και να προωθηθούν στις προγραμματισμένες συναντήσεις. Άλλες αιχμές χρησιμοποιούνται κατά την διάρκεια της φάσης του προγραμματισμού του έργου ώστε να προσδιορίσουν άλυτα θέματα.

4.3Προγραμματισμός Έκδοσης(Release Planning)[18]

Η συνάντηση για τον προγραμματισμό της νεότερης έκδοσης είναι για να δημιουργηθεί ένα σχέδιο σχετικά με την έκδοση, το οποίο θα διευθέτει το συνολικό έργο. Το σχέδιο σχετικά με την νέα έκδοση δείχνει ποια πλάνα χρήστη θα εφαρμοστούν και σε ποια έκδοση αυτό θα συμβεί. Δείχνει επίσης πόσες επαναλήψεις έχουν προγραμματιστεί και πότε η κάθε επανάληψη θα παραδοθεί. Αυτό καθορίζεται με διαπραγμάτευση μεταξύ των ενδιαφερόμενων πλευρών χρησιμοποιώντας τις εκτιμήσεις των παραδόσεων των πλάνων των χρηστών. Οι εκτιμήσεις γίνονται από το τεχνικό μέρος της ομάδας εκμεταλλευόμενοι και τις πληροφορίες των πλάνων. Οι χρήστες δίνουν προτεραιότητα στα πλάνα τους πιθανόν και με εισηγήσεις από το τεχνικό τμήμα της ομάδας. Από αυτά, το χρονοδιάγραμμα του έργου, η ημερομηνία παράδοσης των διαφόρων επαναλήψεων και η παράδοση του τελικού συστήματος είναι όλα διαπραγματεύσιμα. Εάν οι διαχειριστές ή οι πελάτες δεν είναι ικανοποιημένοι με τις προτεινόμενες ημερομηνίες παράδοσης, τότε ένα ή περισσότερα από τα χαρακτηριστικά του συστήματος, οι διαθέσιμοι πόροι ή το χρονικό διάστημα πρέπει να τροποποιηθούν μέχρι ότου όλοι οι συμμετέχοντες να είναι ικανοποιημένοι.

4.4Επαναλήψεις(Iterations)[19]

Κάθε επανάληψη που συμπεριλαμβάνεται στο έργο προσθέτει ευελιξία στην ομάδα ανάπτυξης. Αυτό συμβαίνει γιατί στην αρχή της κάθε επανάληψης αλλαγές μπορεί να συμβούν σχετικά με το τι θα συμβεί και το πότε θα συμβεί. Όσο πιο μικρές είναι οι επαναλήψεις τόσο η ομάδα ανάπτυξης μπορεί να ανταποκριθεί στις αλλαγές. Στην έναρξη της κάθε επανάληψης μια προγραμματισμένη συνάντηση καθορίζει ακριβώς τι θα συμβεί σε αυτή την επανάληψη. Αυτός ο ακριβής προγραμματισμός είναι υπεύθυνος για το ότι η ομάδα προλαβαίνει τις αλλαγές των απαιτήσεων του χρήστη. Εάν ωστόσο φαίνεται ότι η επανάληψη δεν θα καταφέρει να πετύχει όλα αυτά τα οποία είχαν προγραμματιστεί, τότε μια άλλη συνάντηση πρέπει να γίνει για να ανταποκριθεί σε αυτά τα ζητήματα. Αυτές οι συναντήσεις μπορεί να χρειαστεί να επανεκτιμήσουν τις τρέχοντες και μελλοντικές εργασίες και να προσδιορίσουν τι μπορεί και τι όχι να συμβεί.

4.5Αποδεκτές Δοκιμές(Acceptance Testing)[19]

Οι αποδεκτές δοκιμές δημιουργούνται από τα πλάνα των χρηστών και γράφονται στην αρχή του έργου. Κάθε επανάληψη εφαρμόζει ένα ή περισσότερα πλάνα χρηστών, αυτά τα πλάνα θα μεταφραστούν σε μια σειρά από αποδεκτές δοκιμές κατά την διάρκεια της επανάληψης. Για να συμβεί αυτό ο πελάτης πρέπει να καθορίσει σενάρια για να δοκιμάσει ότι ένα πλάνο χρήστη έχει σωστά υλοποιηθεί. Ένα πλάνο χρήστη δεν περιορίζεται σε ένα μόνο αποδεκτό έλεγχο αλλά μπορεί να σχετίζεται με πολλούς αποδεκτούς ελέγχους(εξαρτάται από την πολυπλοκότητα του σεναρίου). Ενδιαφέρον έχει ότι είναι οι χρήστες αυτοί που έχουν πρόσβαση στα αποτελέσματα των αποδεκτών ελέγχων και αξιολογούν τα αποτελέσματα των ελέγχων ώστε να αποφασίσουν ποιοι από τους αποτυχόντες ελέγχους είναι υψηλότερης προτεραιότητας. Συνεπώς οι αποδεκτές δοκιμές πρέπει να είναι αυτοματοποιημένες ώστε να μπορούν να λειτουργούν πιο συχνά. Επιπλέον σε κάθε επανάληψη όλοι οι προηγούμενοι αποδεκτοί έλεγχοι θα πρέπει να τρέχουν ώστε να διασφαλίσουν ότι τίποτα δεν χάλασε.

4.6Εκδόσεις(Release)[19]

Η XP προωθεί τις μικρές εκδόσεις(small releases) και τις συχνές εκδόσεις(release often). Οι συναντήσεις για τον προγραμματισμό των εκδόσεων πρέπει να προσδιορίζει την σημασία της λειτουργικότητας του συστήματος, αυτό έχει επιχειρησιακό νόημα σε σχέση με τους χρήστες και την κατάσταση συστήματος στα διάφορα στάδια που αυτό περνάει. Αυτές οι περιεκτικές εκδόσεις πρέπει να είναι διαθέσιμες στους χρήστες όταν διατεθούν. Αυτό θα επιτρέψει νωρίτερες και συχνότερες ανατροφοδοτήσεις από τους χρήστες παρά στο να στηριχθεί σε μια μεγάλη και απότομη προσέγγιση και μετά να μας ενδιαφέρουν οι συνέπειες.

5. Pair Programming

5.1 Εισαγωγή[20]

Ο συνεργατικός προγραμματισμός(Pair Programming) απαιτεί μια εργασία ανά ζεύγη όπου όλος ο παραγόμενος κώδικας γράφεται από δύο άτομα που κάθονται μαζί στον ίδιο υπολογιστή, μοιραζόμενοι ένα ποντίκι, ένα πληκτρολόγιο και μια οθόνη. Σε κάθε στιγμή, ένα από τα άτομα συγκεντρώνεται στο να γράφει με λεπτομέρεια ενώ το άλλο παρατηρεί τι ο συνεργάτης του κάνει ώστε να έχει την γενική εικόνα, κάνοντας ερωτήσεις όπως, είναι αυτό ότι οι απαιτήσεις εννοούν; Τηρούμε τα πρότυπα κωδικοποίησης και τις πρακτικές της ομάδας; Υπάρχει απλούστερη υλοποίηση; Οι δυο συνεργάτες αλλάζουν συχνά ρόλους, και αυτό βοηθάει στο να συνεχιστεί ο διάλογος, κάποιες φορές κάνεις ερωτήσεις και κάποιες φορές απαντάς σε ερωτήσεις, άλλες φορές προφορικά και άλλες φορές γράφοντας τες. Οι άνθρωποι μαθαίνουν πιο γρήγορα όταν δουλεύουν μαζί και προσπαθούν να αναπτύξουν καλύτερες λύσεις, είναι κοινός αποδεκτό ότι τις περισσότερες φορές δυο μυαλά είναι καλύτερα από ένα. Είναι επίσης πιο δύσκολο τα λάθη να μην παρατηρηθούν όταν δυο άνθρωποι συνεχώς ελέγχουν ο ένας την δουλειά του άλλου. Τι καλύτερο από την αλληλεπίδραση δυο ανθρώπων παράγει κώδικα που είναι πιο εύκολος στην κατανόηση και την συντήρηση. Η συνεργασία προσφέρει ένα καθαρό πλεονέκτημα σε αυτούς τους τομείς κλειδιά στην ανάπτυξη λογισμικού, οπότε, γιατί ακόμα οι προγραμματιστές δουλεύουν μόνοι τους; Είναι απλά ένα θέμα προσωπικής συνήθειας ή έχοντας λάθος πεποίθηση για την οικονομική αποτελεσματικότητα. Προκειμένου να εφαρμοστεί η συνεργασία, χρειάζονται αρκετά άτομα για να εναλλάσσονται ώστε κάθε υπεύθυνος για την ανάπτυξη να δουλέψει με κάποιον άλλον συνεργάτη το λιγότερο, κάποιες φορές την ημέρα. Αυτό όχι μόνο μας εξασφαλίζει ότι οι διάλογοι παραμένουν φρέσκοι και ενημερωτικοί, αλλά και ότι η γνώσει διαχέεται μέσα στην ομάδα. Οι συχνές αλλαγές μας εξασφαλίζουν ότι τα μέλη γνωρίζουν ολόκληρη την δομή του κώδικα παρά να ειδικεύονται σε συγκεκριμένα κομμάτια. Αυτό επίσης σημαίνει ότι οι συμβουλές και τα τεχνάσματα μαθαίνονται γρήγορα. Η συνεργασία δεν σημαίνει ότι δεν μπορεί κάποιος ποτέ να δουλέψει μόνος ώστε να λύσει κάποια δύσκολα προβλήματα ή να αναπτύξει ένα πρόχειρο πρότυπο, αλλά σημαίνει ότι οι πληροφορίες που παράγονται δεν μπορούν έτσι απλά να προστεθούν στον παραγόμενο κώδικα αν δεν περάσουν από μια συλλογική συνεδρία. Ο συνεργατικός προγραμματισμός(Pair Programming) είναι απαιτητικός διότι πρέπει συνεχώς να είσαι συγκεντρωμένος στην εργασία που έχεις αναλάβει και να σκέφτεσαι σκληρά ώστε να επικυρώνεις τις απόψεις σου στον συνεργάτη σου. Είναι αναμενόμενο η κούραση να είναι πιο έντονη όταν δουλεύεις σε ζευγάρι παρά όταν δουλεύεις μόνος και τα οικονομικά οφέλη δεν μπορούν να φανούν από τις γραμμές κώδικα που γράφονται με τον συνεργατικό προγραμματισμό. Η πραγματική οικονομία προέρχεται από ολόκληρο τον κύκλο της ανάπτυξη του λογισμικού, από τις γραμμές κώδικα που δεν γράφηκαν, τα λάθη που δεν εισήχθηκαν και την επίπονη συντήρηση που αποφεύχθηκε. Είναι σημαντικό να

καταλάβουμε ότι η Extreme Programming μέθοδος δεν είναι μόνο μια λίστα από πρακτικές που απλά διαλέγεις. Η δύναμη της μεθόδου αυτής προέρχεται από όλες τις πρακτικές τις οι οποίες συνδυάζονται μαζί αρμονικά, καθώς όλες οι πρακτικές είναι κατά κάποιον τρόπο αλληλοεξαρτώμενες, όχι μόνο η μια με την άλλη αλλά επίσης υποστηρίζουν τις αξίες της επικοινωνίας, την ανατροφοδότηση, την τόλμη, την απλότητα και τον σεβασμό.

5.2Επικοινωνία[20]

Στις ομάδες ευέλικτης ανάπτυξης έχει αξία η επικοινωνία διότι είναι πιο προσηλωμένοι στο να προλάβουν και να πιάσουν λάθη πριν αυτά συμβούν παρά να καταλογίσουν ευθύνες ο ένας στον άλλο αφότου αυτά γίνουν. Είναι πολύ καλύτερο οι υπεύθυνοι ανάπτυξης να απασχολούνται με έναν προοδευτικό διάλογο με τους πελάτες ώστε αυτό που παραδίδεται να είναι αυτό που πραγματικά χρειάζεται παρά να καταναλώνουν μήνες στο να διαπραγματεύονται ένα συμβόλαιο που ο μοναδικός στόχος του είναι να υποστηρίξει κάποια μελλοντική διαμάχη. Η επιτυχία μιας τέτοιας ομάδας βασίζεται όχι μόνο να κερδίσει σε μια δικαστική υπόθεση αλλά στο να ικανοποιήσει τον πελάτη με το να παραδώσει ένα αξιόλογο λογισμικό νωρίτερα και συνεχόμενα. Η ομάδα αναγνωρίζει ότι αυτοί οι στόχοι πραγματοποιούνται καλύτερα μέσο των ατόμων και των αλληλεπιδράσεων τους παρά από την εφαρμογή των τεχνικών και την χρήση των εργαλείων, έτσι δίνουν ιδιαίτερη προσοχή στην σαφή, έγκαιρη και σωστή μεταφορά των πληροφοριών.

5.3Ανατροφοδότηση[20]

Η ανάπτυξη λογισμικών συνήθως συμβαίνει σε έναν ραγδαία αναπτυσσόμενο κόσμο. Τα χαρακτηριστικά που έδειχναν τόσο σημαντικά για την επιχείρηση στην αρχή του έργου μπορεί να είναι περιττά έξι μήνες μετά αλλά το περιθώριο της ευκαιρίας για αλλαγή έχει κλείσει. Η τεχνολογία που φαίνεται καλή σήμερα αναμφισβήτητα θα αντικατασταθεί από μια ακόμα καλύτερη τον επόμενο χρόνο. Οι ευέλικτες ομάδες δέχονται τέτοιες αλλαγές εφαρμόζοντας ανατροφοδότηση, περιοδικές αναπροσαρμογές στην κατεύθυνση που γίνονται βάση των έγκαιρων και ακριβών πληροφοριών σχετικά με την πρόοδο σε ένα δυναμικό περιβάλλον. Μια ευέλικτη ομάδα εκτιμάει την ανατροφοδότηση διότι η ανατροφοδότηση επιτρέπει στην ομάδα να ανταποκριθεί στις αλλαγές παρά να ακολουθήσει κάποιο συγκεκριμένο σχέδιο. Αυτό βοηθάει την ομάδα να κρατήσει την υπόσχεση της να παραδώσει ένα αξιόλογο λογισμικό όταν αυτό απαιτηθεί. Οι ευέλικτες ομάδες μπορούν να χρησιμοποιήσουν την ανατροφοδότηση αποτελεσματικά επειδή έχουν λειτουργικό λογισμικό και διαθέσιμα κριτήρια από τα αρχικά στάδια του έργου, τα αποτελέσματα των αλλαγών μεταδίδονται γρήγορα και ακριβή σε αυτούς. Οι άλλες αξίες και οι πρακτικές μιας ευέλικτης ομάδας συγκρατούν αυτήν την ανατροφοδότηση έτσι ώστε το έργο να μην βυθιστεί στο χάος σαν αποτέλεσμα της προσπάθειας της ομάδας να εφαρμόσει τόσες πολλές αλλαγές τόσο γρήγορα. Η

ανατροφοδότηση δουλεύει στις ευέλικτες ομάδες όχι μόνο σαν ένα μακροσκοπικό επίπεδο των όρων για τον έλεγχο του έργου αλλά και σαν ένα μακροσκοπικό επίπεδο σε σχέση με τις καθημερινές δραστηριότητες. Επομένως είναι τόσο σημαντικό για την ομάδα να αποκτήσει ανατροφοδότηση για μια έκδοση όσο είναι για ένα ζευγάρι υπεύθυνων ανάπτυξης να δώσει ο ένας στον άλλο ανατροφοδότηση σχετικά με μια ιδιαίτερη γραμμή κώδικα.

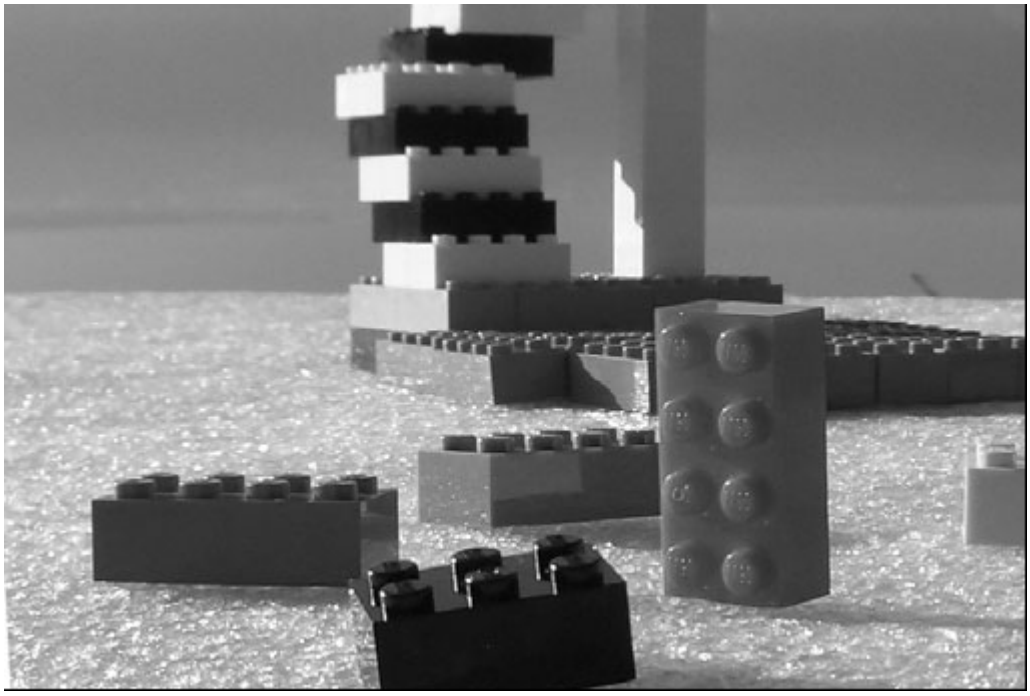
5.4Τόλμη[20]

Μια ευέλικτη ομάδα εκτιμάει την τόλμη επειδή ή τόλμη δείχνει ότι η ομάδα παίρνει τα τεχνικά και επαγγελματικά ρίσκα τα οποία οδηγούν σε ένα αξιόλογο λογισμικό. Ωστόσο, τέτοιες ομάδες δεν παίρνουν απερίσκεπτα τέτοια ρίσκα. Η ομάδα θα προσπαθήσει να μειώσει τα ρίσκα αυτά όσο το δυνατόν περισσότερο. Μπορεί να μειώσει το ρίσκο της να δοκιμάσει μια νέα τεχνολογία, για παράδειγμα με το να δημιουργήσει ένα μικρό έργο συγκεκριμένης διάρκειας που να εξετάζει κατά πόσο είναι κατορθωτό. Άλλες ευέλικτες αξίες και πρακτικές επίσης παρέχουν υποστήριξη όταν παίρνονται αυτά τα απαραίτητα ρίσκα, ώστε να μοιράσουν το βάρος και την επιτυχία σαν ομάδα και όχι μεμονωμένα. Οι ομάδες που στερούνται το θάρρος τείνουν να διστάζουν και να χάνουν την πρωτοβουλία, συχνά καθυστερούν να δράσουν μέχρι ότου η ευκαιρία περάσει. Οι τολμηρές ομάδες ισορροπούν τα ρίσκα σε σχέση με τη δυνατή επιτυχία, περιμένουν μέχρι την κατάλληλη στιγμή, και τότε προχωρούν σε ένα συγκεκριμένο σχέδιο δράσης με πλήρη εμπιστοσύνη στις δυνατότητες τους ώστε να ολοκληρώσουν αυτό το οποίο έχουν καθορίσει να κάνουν.

5.5Απλότητα[20]

Υπάρχει ένας κίνδυνος για τις ομάδες λογισμικού που χρησιμοποιούν τα προηγμένα εργαλεία και τεχνικές τους ώστε να επιδιώξουν να βρουν ακόμα πιο σύνθετες λύσεις για τα προβλήματα τους. Εξαρχής σκάφτονται για μια πολύπλοκη λύση οπότε όχι και τόσο απρόσμενα αυτό είναι που παράγουν. Εκτιμώντας την απλότητα δεν σημαίνει ότι δεν μπορείς να πάρεις τα πλεονεκτήματα των νέων εργαλείων και τεχνικών, αλλά σημαίνει ότι στοχεύεις να προχωρήσεις προς μια απλή λύση και όχι μακριά από αυτή. Αυτό σημαίνει συνεχώς να βελτιώνεις την λύση κάνοντας την πιο απλή μέχρι να φτάσεις στο σημείο στο οποίο οποιαδήποτε περεταίρω απλοποίηση θα αλλοίωση την λειτουργικότητα της. Η απλότητα συχνά είναι και ένα χαρακτηριστικό πολύ καλών σχεδίων από αυτά που είναι διαρκεί και δύσκολο να βελτιωθούν περισσότερο. Για παράδειγμα ένα τουβλάκι Lego έχει παραμείνει ουσιαστικά απαράλλακτο εδώ και πενήντα χρόνια γιατί κανένας δεν κατάφερε να φτιάξει ένα πιο απλό σχέδιο ώστε να φτιάχνεις κατασκευές που τα παιδιά μπορούν να συνδυάζουν μαζί αλλά και να χωρίζουν. Πολλά άλλα προϊόντα μοιράζονται αυτή την απλότητα. Οι άνθρωποι απέτυχαν να βελτιώσουν αυτές τις κατασκευές επειδή δεν μπορούν να γίνουν πιο απλά και κάνοντας τες πιο πολύπλοκες δεν θα κατάφερναν να τις έκαναν καλύτερα. Οι ευέλικτες ομάδες

αξιοποιούν την απλότητα διότι τους οδηγεί στο να αφαιρέσουν την μη απαραίτητη πολυπλοκότητα και τους αποτρέπει από το να κάνουν συμβιβασμούς για κάποιες μελλοντικές ανάγκες που μπορεί να μην υλοποιηθούν. Συγκεντρώνονται παρόλα αυτά στο να κατασκευάσουν μόνο ότι είναι απαραίτητο για το παρόν, το πιο απλό πράγμα το οποίο μπορεί να δουλέψει. Σκεπτόμενοι με απλότητα βοηθάει μια ευέλικτη ομάδα στο να δημιουργήσει λογισμικό το οποίο είναι αμέσως περιζήτητο, χωρίς να σπαταλείται προσπάθεια ή υλικά. Αυτός είναι ο τρόπος που μια ομάδα δίνει στον πελάτη της το καλύτερο και το πιο άμεσο αποτέλεσμα στην επένδυση του.



Σχήμα 6: Μοντέλο Lego

5.6Σεβασμός[20]

Σεβασμός είναι να αναγνωρίζεις την συνεισφορά που ένα άτομο μπορεί προσφέρει μέσα από τις ικανότητες και το ταλέντο του. Χωρίς σεβασμό είναι δύσκολο να συνεργαστούμε. Γιατί να θέλουμε να δουλέψουμε με κάποιον που δεν προσφέρει πραγματικές αξίες. Ωστόσο πρέπει να είμαστε προσεκτικοί να μην επιτρέψουμε στην προκατάληψη να επηρεάσει αυτές τις αξίες. Σεβασμός σημαίνει να αναγνωρίζετε η διαφορά μεταξύ ανθρώπων και να επενδύεται πάνω σε αυτή την ποικιλομορφία με το να δουλεύει αντίστοιχα με τις δυνάμεις της ομάδας και να αποζημιώνεται για τις διάφορες αδυναμίες της ομάδας. Δεν είναι μόνο ένα ηθικό θέμα σχετικά με την ποικιλομορφία αλλά προσφέρει προσαρμοστικότητα και άλλα οφέλη στην ομάδα. Μια ευέλικτη ομάδα εκτιμάει τον σεβασμό διότι βοηθάει την ομάδα να δουλέψει μαζί και πιο αποτελεσματικά σε ένα συνεχώς μεταβαλλόμενο περιβάλλον. Ο σεβασμός απαιτεί ένα βαθμό εμπιστοσύνης και ειλικρίνειας μεταξύ των ατόμων της ομάδας καθώς θα πρέπει να είναι σε θέση να πουν αυτό που

σκέφτονται όταν είναι απαραίτητο, χωρίς να φοβούνται για αντίποινα. Ωστόσο σεβασμός σημαίνει να δείχνεις συναισθηματική ωριμότητα σε ορισμένες καταστάσεις όπως να γίνεσαι εύθικτός στα συναισθήματα των άλλων μελών και να παίρνεις την κριτική δημιουργικά. Σε κάποιες περιπτώσεις ο σεβασμός είναι ο κάδος για τα ευερέθιστα συναισθήματα που προκύπτουν όταν άνθρωποι προσπαθούν να δουλέψουν μαζί σαν ομάδα. Ωστόσο μπορεί να γίνει ντροπιαστικό να συζητάς τέτοια θέματα διότι περιλαμβάνουν έντονα προσωπικά πράγματα όπως οι τρόποι και οι κοινωνικές ικανότητες, αυτό δεν σημαίνει ότι μπορούμε να τα αγνοούμε. Η έλλειψη σεβασμού και ιδιαίτερα ο αυτοσεβασμός είναι συχνά πρωταρχικό αίτιο για την δυσλειτουργία της ομάδας.

6. Θεωρητικό πλαίσιο: 5-Α μοντέλο(5-A Model)[10]

6.1 Ανάλυση

Οι σύγχρονοι οργανισμοί λογισμικού λειτουργούν σε μια υπερβολικά δυναμική αγορά κάτω από στενά χρονικά και χρηματικά περιθώρια. Η ανάπτυξη των προϊόντων είναι πολύ ευμετάβλητη σε σχέση με τις αλλαγές των απαιτήσεων, και τις ανάγκες της επιχείρησης και της αγοράς. Οι ευέλικτες μέθοδοι ανάπτυξης λογισμικών σχεδιάστηκαν για να μπορέσουν να ανταπεξέλθουν σε αυτά τα προβλήματα. Από την παρουσίαση της και μετά η XP(Extreme Programming) έχει γίνει η πιο τεκμηριωμένη ευέλικτη μέθοδος και έχει πυροδοτήσει έναν αριθμό από νέες έρευνες, άρθρα και αναφορές εμπειριών.

Ενώ η κατασκευή λογισμικού(Software Engineering, SE) είναι μια πρακτική πειθαρχίας, η θεωρητική ανάπτυξη συχνά είναι λιγότερο προσεγμένη. Μια κοινή άποψη είναι ότι οι θεωρίες απευθύνονται σε 2 στόχους της επιστήμης: την ερμηνεία(ή κατανόηση) και την πρόβλεψη. Η έλλειψη των θεωριών, επομένως παρεμποδίζει την ικανότητα να ερμηνεύσει και να προβλέψει απευθείας ή να εξετάσει την συμπεριφορά όπως κάνει μια επιτυχής εφαρμογή της XP σε συγκεκριμένο περιβάλλον. Για να μπορέσει να ολοκληρωθεί μια θεωρεία πρέπει να περιέχει 4 ουσιώδη στοιχεία:

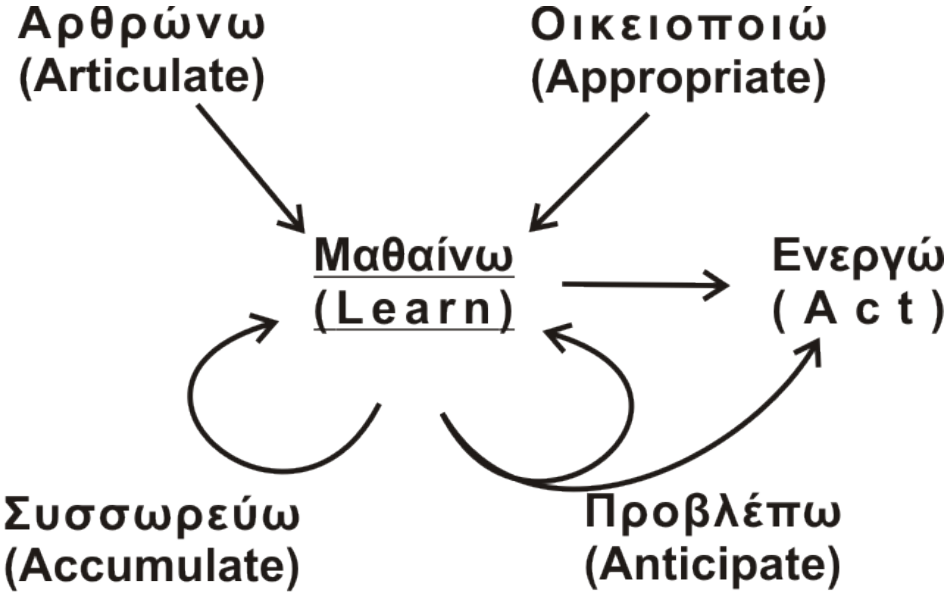
- **ΤΙ**. Ποιοι παράγοντες πρέπει λογικά να ληφθούν υπόψη σαν μέρος της εξήγησης των φαινομένων που μας ενδιαφέρουν.
- **ΠΩΣ**. Πως οι παράγοντες συσχετίζονται.
- **ΓΙΑΤΙ**. Ποια είναι η αιτιολογία που δικαιολογεί την επιλογή των παραγόντων και οι προτεινόμενες αιτιώδης σχέσεις. Μια επεξήγηση είναι απαραίτητη.
- **ΠΟΙΟΣ, ΠΟΥ, ΠΟΤΕ**. Τα όρια της δυνατότητας γενίκευσης και κατά συνέπια τα όρια της θεωρίας πρέπει να ορισθούν.

Για αυτούς τους λόγους ένα ευέλικτο μανιφέστο, μια συμφωνία για το τι είναι σημαντικό στην ανάπτυξη λογισμικού, παρουσιάζει αξίες και αρχές που δεν πληρούν τις απαιτήσεις της θεωρίας. Η XP και άλλες ευέλικτες μέθοδοι προέρχονται από προσωπική πρακτική εμπειρία. Κατά συνέπεια ο τρόπος με τον οποίο έχουν αναπτυχθεί δεν βασίζεται σε αξιόπιστη και συστηματική έρευνα. Η XP ωστόσο είναι μια από τις λίγες με πειστικά εμπειρικά στοιχεία. Οι αιτίες για την επιτυχία ή αποτυχία δεν μπορούν προς το παρόν να εξηγηθούν λόγο έλλειψης ενός θεωρητικού προτύπου/μοντέλου.

Ο εταιρίες λογισμικού μπορούν να θεωρούνται σαν εταιρίες εντατικής εκμάθησης στις οποίες υψηλής ποιότητας επαγγελματίες εκτελούν εργασίες οποίες κατά κύριο λόγο είναι διανοητικής φύσης. Η έρευνα έχει δείξει επιπλέον ότι η ανάπτυξη λογισμικού είναι μια πολύ έντονη δουλειά εντατικής εκμάθησης. Είναι επομένως εύλογο να υποστηριχθεί ότι η XP μπορεί να θεωρείται σαν μια διαδικασία όπου η δημιουργία γνώσης κατέχει μια κεντρική θέση. Ένα μοντέρνο μοντέλο δημιουργίας γνώσης κατά συνέπεια θα μπορούσε να χρησιμοποιηθεί για να παρουσιάσει τα χαρακτηριστικά της XP.

Υπάρχουν αρκετά θεωρητικά πλαίσια διαθέσιμα για την διαχείριση της γνώσης γενικά και αρκετά για την δημιουργία της γνώσης ειδικά. Το SEMI-μοντέλο(Socialization-Externalization-Combination-Internalization) είναι το πιο γνωστό μοντέλο για την δημιουργία γνώσης. Ωστόσο το 5-Α μοντέλο με την ικανότητα του να εξηγήσει την δημιουργία γνώσης σε περιπτώσεις όπου πολλαπλά σύνολα πρακτικών εμπλέκονται θεωρείται καλύτερο. Στην περίπτωση της ΧΡ, πάντα εμπλέκονται το λιγότερο 2 σύνολα πρακτικών: ο πελάτης που αντιπροσωπεύει τον ειδικό τομέα του συνόλου των πρακτικών και οι κατασκευαστές που αντιπροσωπεύουν το σύνολο των πρακτικών της ανάπτυξης λογισμικού. Το 5-Α μοντέλο για την δημιουργία γνώσης είναι πανομοιότυπο με το SEMI μοντέλο υπό την έννοια ότι και τα δυο προσπαθούν να περιγράψουν τι συμβαίνει όταν δημιουργείται η γνώση. Ωστόσο με το 5-Α μοντέλο είναι δυνατό να περιγράψουμε τον κύκλο της δημιουργίας της γνώσης με περισσότερες λεπτομέρειες.

Καθορίζονται 5 τρόποι για την δημιουργία της γνώσης: **άρθρωση (articulation), οικειοποίηση (appropriation), πρόβλεψη (anticipation)**. Επιπλέον υπάρχουν και οι διαδικασίες της **συσσώρευσης (accumulating*)*** γνώσης και της **δράσης (acting)** σύμφωνα με της γνώση. Αυτές οι διαδικασίες αποτελούν το θεμέλιο λίθο για τις αλλαγές στη δομή της γνώσης.



Σχήμα 7: Μοντέλο 5-Α

Άρθρωση(articulation)[10] σημαίνει ότι ένα άτομο εξωτερικεύει την σημασία των σχέσεων που έχει μέσα στο κεφάλι του. Κάποιες έννοιες παρακολουθούνται στενά ενώ την ίδια στιγμή κάποιες άλλες έννοιες σχέσεων αφήνονται σαν δευτερεύουσες και επουσιώδης(για παράδειγμα κάποιες έννοιες σχέσεων γίνονται λιγότερο σημαντικές και αφήνονται στο περιθώριο) Σαν αποτέλεσμα αυτής της διαδικασίας κάποιες πληροφορίες που θεωρούνται δεδομένες γίνονται σημαντικές και αρθρώνονται ρητά. Η γλώσσα δεν είναι ο μόνος τρόπος για την άρθρωση της γνώσης, μπορεί να εκφραστεί στις πράξεις και στα προϊόντα. Συνήθως οι αξιοπρεπείς διαδικασίες άρθρωσης περιλαμβάνουν την αφαίρεση, την γενίκευση,

την ταξινόμηση, τον συνδυασμό, την ανάλυση, τον επαναπροσδιορισμό, την οπτική αναπαράσταση και την απεικόνιση. Οι δομές της παραγόμενης γνώσης αρθρώνονται σαν αφηρημένες έννοιες, μεταφορές, εικόνες, εργαλεία, μοντέλα, ιστορίες, (άγραφα)σχέδια, ανέκδοτα και συνήθειες που καθοδηγούν τις πράξεις. Η άρθρωση δημιουργεί γνώση η οποία είναι νέα για τον μαθητευόμενο και μπορεί να δημιουργήσει γνώση η οποία να είναι νέα και για τον οργανισμό. Η άρθρωση είναι κύρια διαδικασία στην δημιουργία της γνώσης διότι η οικειοποίηση(appropriation) και η πρόβλεψη(anticipation) βασίζονται σε αυτή.

Οικειοποίηση(appropriation)[10] είναι μια διαδικασία όπου ο μαθητευόμενος αποκτάει γνώση που ήδη υπάρχει στον οργανισμό. Η οικειοποίηση παράγει γνώση που είναι διαθέσιμη στο εσωτερικό του οργανισμού αλλά είναι νέα για τον μαθητευόμενο. Η οικειοποίηση χρησιμοποιεί με το ίδιο τρόπο τις ικανότητες επεξεργασίας του αρχαρίου όπως και στην άρθρωση. Η Οικειοποίηση και ή άρθρωση μπορούν να χρησιμοποιούν το προσωπικό σαν εργαλεία γνώσης και μέσο αυτού να κατανέμουν την διαδικασία. Αυτό σημαίνει ότι κάποιο άλλο άτομο μπορεί να εισάγει μια νέα προοπτική που βοηθάει τον μαθητευόμενο στην διαδικασία της μάθησης.

Πρόβλεψη(anticipation)[10] κατάσταση δημιουργίας γνώσης όπου ένα άτομο δημιουργεί ένα μοντέλου του κόσμου. Η ένταση μεταξύ του κόσμου που προβλέπεται και του κόσμου που παρατηρείται μπορεί να παράγει νέα γνώση. Μπορεί να επιβεβαιώσει το μοντέλο ή να το διαλύσει και να οδηγήσει σε ένα νέο καλύτερο μοντέλο. Η οικειοποίηση αποτελεί θεμέλιο για την πρόβλεψη, επειδή προ-υπάρχον σχέδια, μοντέλα και προσδοκίες χρειάζονται για να αναγνωρίσουν την απόκλιση από αυτές.

Συσσώρευση(accumulation)[10] και το ιστορικό είναι τα θεμέλια για όλα όσα σημαίνουν επεξεργασία. Οι διαδικασίες για την δυναμική δημιουργία μάθησης αποκαλύπτονται μέσα από το πλαίσιο των συσσωρευμένων εννοιών των δομών και της γνώσης. Η συσσώρευση μπορεί να συμβαίνει εσωτερικά, καθώς η γνώση συσσωρεύεται στις προσωπικές δομές εννοιών.

Στην συσσώρευση, η αρθρωμένη γνώση είναι συσσωρευμένη πχ όπως οι πρακτικές, οι γνώσεις, τα σχέδια, τα ολοκληρωμένα πλάνα και η κουλτούρα του οργανισμού. Όπου υπάρχει μια εγγεγραμμένη γλώσσα (πχ φυσική γλώσσα, γλώσσα προγραμματισμού ή ένα σχεδιάγραμμα) διαθέσιμη, είναι δυνατόν να παρουσιαστούν κάποιες από αυτές τις δομές συσσωρευμένων γνώσεων σαν τεκμήρια. Για αυτό το λόγο τα τεκμήρια μπορούν να φαίνονται σαν προσπάθειες να αρθρωθούν κάποιες πλευρές των θεμελίων της ήδη υπάρχουσας συσσωρευμένης γνώσης μέσα σε γλωσσικές φόρμες. Η δημιουργία τεκμηρίου είναι γενικά μια πολύ απαιτητική δραστηριότητα που προϋποθέτει και μεγάλα τμήματα συσσωρευμένης γνώσης αλλά και επιδέξια χρήση αυτών που ήδη υπάρχουν. Στις περισσότερες περιπτώσεις η τεκμηριωμένη παρουσίαση αντιπροσωπεύει μόνο ένα κομμάτι των θεμελίων της δομής της γνώσης, η ερμηνεία της απαιτεί γνώση για την κουλτούρα, τις πρακτικές και την γλώσσα του οργανισμού και του συνόλου των πρακτικών όπου η τεκμηρίωση γράφηκε.

Ενέργεια(Action)[10] μπορεί να είναι και εσωτερική και εξωτερική. Ενώ ή εσωτερική ενέργεια καθοδηγεί τις ενέργειες των ατόμων, η εξωτερική ενέργεια κατευθύνεται αλλού. Η εξωτερική ενέργεια συνδυάζει δυο διαφορετικά είδη συμπεριφοράς: την επικοινωνία και την πράξη. Στην τεχνολογία λογισμικού, η πράξη μπορεί να κατευθύνεται απευθείας στον κώδικα που είναι υπό υλοποίηση(κατασκευή) πχ με το να τον συμπληρώνει ή να τον δοκιμάζει. Η ενέργεια είναι ο ανώτερος στόχος της δημιουργίας γνώσης. Ο στόχος της γνώσης είναι να οδηγήσει και να δημιουργήσει αποτελεσματικές ενέργειες. Εάν δεν υπάρχει επαρκής γνώση, οι κατάλληλες ενέργειες δεν επιλέγονται.

7. Ανάλυση των πρακτικών της XP

7.1Πρακτικές διαχείρισης έργου[10]

Σχεδιασμός τακτικής(Planning game)[10]. Κατά τον σχεδιασμό της τακτικής ο πελάτης αρθρώνει τις προσδοκίες του στα έγγραφα πλάνα των χρηστών τα οποία είναι φτιαγμένα κατάλληλα από την ομάδα σχεδίασης. Υπάρχει μια συζήτηση μεταξύ του πελάτη και της ομάδας ανάπτυξης για να αποσαφηνιστούν τα πλάνα των χρηστών μέχρι ότου ο πελάτης και η ομάδα ανάπτυξης να μπορέσουν να έχουν μια κοινή κατανόηση του πλάνου. Η αναμενόμενη λειτουργία είναι συσσωρευμένη όπως η γνώση που εξυπακούεται στη δομή των εννοιών των ατόμων της ανάπτυξης και, βασιζόμενοι σε αυτή την πληροφορία, η ομάδα μπορεί να προβλέψει πόσο χρόνο σε ιδανικές προγραμματιστικές εβδομάδες θα χρειαστεί για να εφαρμόσει κάθε πλάνο.

Τα διάφορα σημεία του πλάνου προσδιορίζονται από το πλάνο του χρήστη και προστίθενται στην συζήτηση. Η ταχύτητα του έργου απεικονίζει την συσσωρευμένη γνώση της ικανότητας που έχει η ομάδας. Εάν η ομάδα διαπιστώσει ότι δεν μπορεί να προβλέψει με αξιοπιστία πόσο θα διαρκέσει η εφαρμογή του πλάνου, μπορεί να ρωτήσει για περισσότερες πληροφορίες τον πελάτη ή με το να φτιάξει μια ακίδα.

Ο πελάτης οικειοποιεί τα εκτιμημένα πλάνα του χρήστη και προβλέπει μια συλλογή που θα του παρέχει την μέγιστη αξία. Δεδομένου ότι υπάρχουν τεχνικοί παράγοντες μεταξύ των πλάνων των χρηστών, γίνεται μια συζήτηση στην οποία και οι δυο και οι κατασκευαστές και ο πελάτης αποσαφηνίζουν τις απόψεις τους και κάνουν κατάλληλες τις απόψεις των άλλων. Σαν αποτέλεσμα της συζήτησης πρέπει να υπάρξει ένα σύνολο από πλάνα τα οποία να είναι συλλογικά συμφωνημένα και οι πληροφορίες να είναι συσσωρευμένες σε ένα σχέδιο.

Η επανάληψη του σχεδιασμού του πλάνου είναι παρόμοια με την απελευθέρωση του σχεδιασμού του πλάνου. Οι κατασκευαστές συγκεντρώνουν τη δουλειά που πρέπει να γίνει στην επανάληψη σαν εργασίες. Οι κατασκευαστές υπογράφουν τις εργασίες και προβλέπουν πόση θα είναι η διάρκεια ώστε να τελειώσουν αυτή την εργασίας. Οι κάρτες εργασίας είναι απτά σύμβολα τα οποία αντιπροσωπεύουν τις εργασίες, αλλά το νόημα των εργασιών επεξεργάζεται στην συζήτηση. Καθώς τα άτομα καθορίζουν, υπολογίζουν και αναθέτουν εργασίες, άλλα μέλη της ομάδας μορφοποιούν κατάλληλα την τεκμηρίωση και την εγκυρότητα της τακτικής που είναι υπό κατασκευή.

Όταν υπάρξει συμφωνία σχετικά με τις εργασίες και η προσπάθειες πρέπει να εφαρμοστούν, η γνώση συσσωρεύεται πχ. σε χαρτί, σε πίνακα ή στο Excel. Μόνο τα ονόματα των εργασιών συσσωρεύονται σαν ρητή γνώση. Τα λεπτομερή περιεχόμενα των εργασιών μοιράζονται σαν δεδομένη γνώση μεταξύ των μελών της ομάδας.

Ο σχεδιασμός της τακτικής μεταχειρίζεται όλα τα στοιχεία του 5-Α μοντέλου. Ενισχύει την διαδικασία της δημιουργίας της γνώσης χτίζοντας έναν συμμετοχικό διάλογο όπου οι πληροφορίες συνεχώς αρθρώνονται και σχηματίζονται όπως πρέπει. Η πρόβλεψη με την μορφή ενός απελευθερωμένου σχεδίου έχει γίνει μόνο για ένα μικρό σχετικό χρονικό διάστημα, μετά το οποίο η εγκυρότητα του σχεδίου ελέγχεται ξανά. Η γνώση που δημιουργήθηκε στο σχεδιασμό της τακτικής συσσωρεύεται τις περισσότερες φορές σαν μια συνήθης συλλογική κατανόηση του τι πρέπει να γίνει. Ο σχεδιασμός της τακτικής είναι επίσης έτσι σχεδιασμένος όπου τα άτομα που συμμετέχουν στην τακτική είναι δεσμευμένα στο να ξεκινήσουν ενέργειες για να εφαρμοστεί το σχέδιο.

Συχνές μικρές απελευθερώσεις(Frequent small releases)[10] δίνει στον πελάτη συχνές ευκαιρίες να ελέγξει τα επιχειρήματα έως εκείνη την στιγμή. Ο πελάτης από την μια ελέγχει ότι η ομάδα ανάπτυξης έχει εκπληρώσει τις δεσμεύσεις της, και από την άλλη εκφράζει την πρόβλεψη του για την λειτουργικότητα. Το παραδοτέο λογισμικό μπορεί να διαλύσει την προϋπάρχουσα αντίληψη του πελάτη σχετικά με το τι θέλει και συμπληρώνει μια νέα άρθρωση με τις ανάγκες του. Αυτή η άρθρωση δημιουργεί μια νέα γνώση σχετικά με την επιθυμητή λειτουργικότητα του συστήματος και μπορεί να οδηγήσει σε ενέργειες με την μορφή της αλλαγής των σχεδίων. Με τη συχνή απελευθέρωση, η ομάδα δίνει στον πελάτη περισσότερες ευκαιρίες να τροποποιήσει την μέχρι στιγμής πορεία και κατά συνέπεια να δημιουργήσει νέα γνώση.

7.2Εφαρμογή των πρακτικών[10]

Απλός Σχεδιασμός(Simple Design)[10]. Στις CRC(Class-Responsibilities-Collaboration) συνεδριάσεις ο κατασκευαστής αρθρώνει το πώς καταλαβαίνει την συμπεριφορά του συστήματος. Άλλοι προσαρμόζουν το σενάριο και προσπαθούν να προβλέψουν πιθανό λάθη και αδυναμίες. Το σχέδιο και το σκεπτικό πίσω από τις αποφάσεις του σχεδίου συσσωρεύονται σαν συλλογική δεδομένη γνώση της ομάδας. Η γνώση για το σχέδιο μένει με τα μέλη της ομάδας ακόμα και όταν οι κάρτες έχουν καταστραφεί και δεν υπάρχει γραμμένη τεκμηρίωση για το σχέδιο.

Η δημιουργία γνώσης στη διαδικασία του σχεδιασμού προάγεται παραπάνω από τον ίδιο μηχανισμό όπως στον σχεδιασμό της τακτικής: οι CRC συνεδριάσεις δημιουργούν έναν διάλογο που ενισχύει την άρθρωση και προσαρμοστικότητα, το καλύτερο σχέδιο είναι προβλεπόμενο μόνο για ένα μικρό χρονικό διάστημα, και οι πληροφορίες συσσωρεύονται σε μια συλλογική δεδομένη γνώση παρά σε έγγραφα.

Ακίδα(Spike) [10], ένα απορριπτέο πρότυπο, είναι μια πιθανότητα εάν η ομάδα δεν μπορεί επαρκώς να προβλέψει τις απαιτούμενες εργασίες. Αυτό είναι τυπικά η περίπτωση όπου η ομάδα δεν έχει κύρια συσσωρευμένη γνώση της τεχνολογίας που χρησιμοποιείται. Με την ιδιοποίηση του πρωτοτύπου, νέα γνώση δημιουργείται για να παίρνονται πιο σωστές αποφάσεις. Η ομάδα προβλέπει μόνο στις ουσιαστικές προοπτικές του συστήματος και γρήγορα τις αρθρώνει σε ένα πρότυπο.

Pair Programming[10]. Ο προγραμματισμός σε ζευγάρια οδηγεί σε πιο αποτελεσματική δημιουργία γνώσης. Διότι ωθεί τους κατασκευαστές να αρθρώσουν ρητά τις σκέψεις τους. Επίσης θέτει σε εφαρμογή μια αμοιβαία διαδικασία οικειοποίησης. Εκτός από αυτό οι κατασκευαστές προβλέπουν τι οι άλλοι θέλουν να κάνουν. Εάν η πρόβλεψη χαλάσει, μπορεί να οδηγήσει σε μια κερδοφόρα συζήτηση. Καθώς το ζευγάρι αλλάζει, η γνώση που συσσωρεύεται στο μυαλό των ανθρώπων διαδίδεται αποτελεσματικά. Μπορεί να οδηγήσει επίσης και σε συσσώρευση γνώσης υπό την μορφή της κοινών πρακτικών.

Επαναπαραγωγή(Refactoring) [10] μπορεί να φανεί σαν μια συνεχείς διαδικασία οικειοποίησης. Όταν ένα ζευγάρι παρατηρεί ότι δεν υπάρχει ταύτιση μεταξύ των απόψεων το σε σχέση με την κατανόηση του καλύτερου σχεδίου ή αυτού που ήδη υπάρχει, μπορεί αμέσως να αρθρώσει και να συσσωρεύσει το καλύτερο σχέδιο.

Αλληγορία(Metaphor) [10] είναι ένα εργαλείο που βοηθάει στο να εκφραστούν οι δύσκολες έννοιες των δομών. Οι άνθρωποι συχνά και ενστικτωδώς χρησιμοποιούν διαφορετικού είδους μεταφορές. Ωστόσο, αυτή η πρακτική συχνά έχει βρεθεί ότι είναι δύσκολο να χρησιμοποιηθεί σε έργα που χρησιμοποιούν την μέθοδο XP. Από την μεριά της δημιουργίας της γνώσης, και οι τυπικές συμφωνημένες μεταφορές και οι ενστικτωδώς χρησιμοποιούνται για συγκεντρωμένου σκοπού μεταφορές και μπορούμε να τις δούμε σαν πρακτικές που βοηθούν στη διαδικασία της άρθρωσης.

Πρότυπα κωδικοποίησης(Coding standard) [10] αντιπροσωπεύουν την συσσωρευμένη γνώση που η ομάδα έχει από τις σωστές πρακτικές κωδικοποίησης.

7.3Πρακτικές επαλήθευσης και επικύρωσης(Verification & validation practices)[10]

Δοκιμή(Testing) [10] συχνά ξεκινάει σαν μια απλή πρακτική, αλλά περιλαμβάνει πολλά τυπικά χαρακτηριστικά της XP, τα τεστ γράφονται πριν από τον πηγαίο κώδικα, είναι αυτοματοποιημένα και τρέχουν συχνά, και ο πελάτης γράφει αποδεκτά τεστ.

Η εφαρμογή αρχίζει με την πρόβλεψη και την άρθρωση της λειτουργικότητας με την μορφή των τεστ. Βασιζόμενο στα τεστ που γράφονται, το ζευγάρι των προγραμματιστών μπορεί αμοιβαία να οικειοποιηθεί ότι έχει μια κοινή αντίληψη για την επιδιωκόμενη λειτουργικότητα. λόγω του ότι τα τεστ φτιάχνονται πριν από τον κώδικα, χρειάζεται περισσότερη προσοχή στην κατασκευή τους. Τα τεστ προβλέπουν σε ένα πολύ μικρό κομμάτι της λειτουργικότητας σε κάποιο χρονικό διάστημα και η οπισθοδρόμηση έρχεται πολύ γρήγορα. Όταν ένα λάθος βρεθεί από το τεστ, θα πρέπει να οδηγήσει σε άμεση αντίδραση για την επίλυση του προβλήματος διότι όλα τα τεστ πρέπει να τρέχουν χωρίς προβλήματα συνέχεια. Όταν το ζευγάρι που προγραμματίζει τρέχει τα τεστ, μπορεί να χρησιμοποιεί τα αποτελέσματα για να εφαρμόσει τις κατάλληλες προσαρμογές ώστε να επιτύχει τις καθορισμένες προσδοκίες. Επιπλέον με το να τρέχει κανείς τα προϋπάρχον

τεστ μπορεί να εφαρμόσει τις κατάλληλες προσαρμογές στην υπάρχουσα βάση κώδικα.

Όταν καταγράφονται αποδεκτά τεστ, ο πελάτης αρθρώνει τις προσδοκίες του με λεπτομέρεια. Τα τεστ συσσωρεύουν με σαφήνεια γνώση σχετικά με την επιθυμητή λειτουργικότητα. Τρέχοντας τα αποδεκτά τεστ, τα μέλη της ομάδας προσαρμόζουν όχι μόνο τις προσδοκίες του πελάτη, αλλά προσαρμόζουν την εφαρμογή και στις δικές τους προσδοκίες.

Όταν ένα σφάλμα βρεθεί, ένα τεστ γράφεται για να αποτρέψει το λάθος αυτό να ξαναγίνει. Το τεστ αυτό προσφέρει αρθρωμένη και συσσωρευμένη γνώση του τι μπορεί να πάει λάθος.

Εκ όψεως επαφή με τον πελάτη(On Site Costumer) [10]. Όταν ένα μέλος της ομάδας ανάπτυξης είναι σε αμφιβολία μπορεί να ρωτήσει τον πελάτη για να του δώσει περισσότερες λεπτομέρειες η να προσαρμόσει την προτεινόμενη λύση του. Επειδή ο πελάτης είναι πρόθυμα διαθέσιμος, η άρθρωση και η προσαρμογή μπορεί να ξεκινήσει αμέσως.

7.4Διαμόρφωση των πρακτικών διαχείρισης(Configuration management practices)[10]

Συλλογικός Κώδικας(Collective Code Ownership) [10]. Η προσέγγιση αυτή είναι καλά προσαρμοσμένη με το μοντέλο δημιουργίας γνώσης διότι υποτίθεται ότι η ομάδα εργάζεται συλλογικά στις μοιραζόμενες βάσεις εννοιών. Ο κώδικα αναπαριστά ρητά την συσσώρευση των δομών των εννοιών της επιθυμητής λειτουργικότητας και σύμφωνα με τις πρακτικές του συλλογικού κώδικα, είναι επίσης μοιρασμένα συλλογικά.

Συνεχείς Ενσωματώσεις(Continuous Integration) [10]. Καθώς ο κώδικας είναι ενσωματωμένος και δεσμευμένος στην αποθήκη, συσσωρεύει την κοινή γνώση της ομάδας. Στην ενσωμάτωση η προσαρμογή γίνεται με 2 τρόπους. Πρώτον, πριν γραφεί ο κώδικας, το ζευγάρι ενσωμάτωσης υποστηρίζει ότι ο κώδικας του είναι συμβατός με την προϋπάρχουσα εργασία. Δεύτερον όταν ο κώδικας γραφεί είναι δυνατόν άλλοι προγραμματιστές να προσαρμόσουν την δουλεία τους σε σχέση με αυτές τις αλλαγές. Στις συνεχείς ενσωματώσεις, η αρθρωμένες πληροφορίες είναι συσσωρευμένες συχνά έτσι ώστε να είναι δυνατόν σε άλλους να προσαρμοστούν χωρίς ιδιαίτερες λεπτομέρειες. Κατά συνέπεια η συνεχής διαδικασία ενσωμάτωσης βοηθάει τα μέλη της ομάδας ώστε να κρατήσουν την δουλεία τους σε μια ευθεία.

Πίνακας 5: Πρακτικές και πιθανές βελτιώσεις

Επίπεδο κάλυψης (Level of Coverage)	**Επεξηγούμενο από το 5-Α μοντέλο (Explained by 5-A model)**	**Περιοχές Βελτίωσης (Areas of improvement)**
Επάρκεια (Sufficient)	• Συνεχείς Ενσωματώσεις (Continuous Integration) • Ακίδα (Spike) • Συλλογικός Κώδικας (Collective Code Ownership) • Αλληγορία (Metaphor) • Ανοικτός χώρος εργασίας (Open work space) • Πρότυπα κωδικοποίησης (Coding standard) • Επαναπαραγωγή(Refactoring)	Το 5-Α μοντέλο προσφέρει επαρκή κάλυψη για τις πρακτικές
Επεξηγείται αλλά χρειάζεται και άλλη υποστήριξη από άλλες θεωρίες (Explained, but need support from other theories)	Σχεδιασμός τακτικής (Planning game)	Δέσμευση διαχείρισης
	Μικρές απελευθερώσεις (Small releases)	Χρησιμότητα στη επιχείρηση
	Προγραμματισμός με ζευγάρι (Pair Programming)	Διαδικασία ενισχυμένης πειθαρχίας
	Εκ όψεως επαφή με τον πελάτη (On Site Costumer)	Λήψη αποφάσεων
	Δοκιμές(Testing)	Αποτελέσματα από την έγκαιρη ανίχνευση λαθών
	Απλός Σχεδιασμός (Simple Design)	Κόστος από την αλλαγή γραμμής
Εκτός στόχου (Outside of scope)	40 ώρες την Εβδομάδα (40 hour Week)	Βιώσιμος ρυθμός

8. Η μέθοδο Scrum

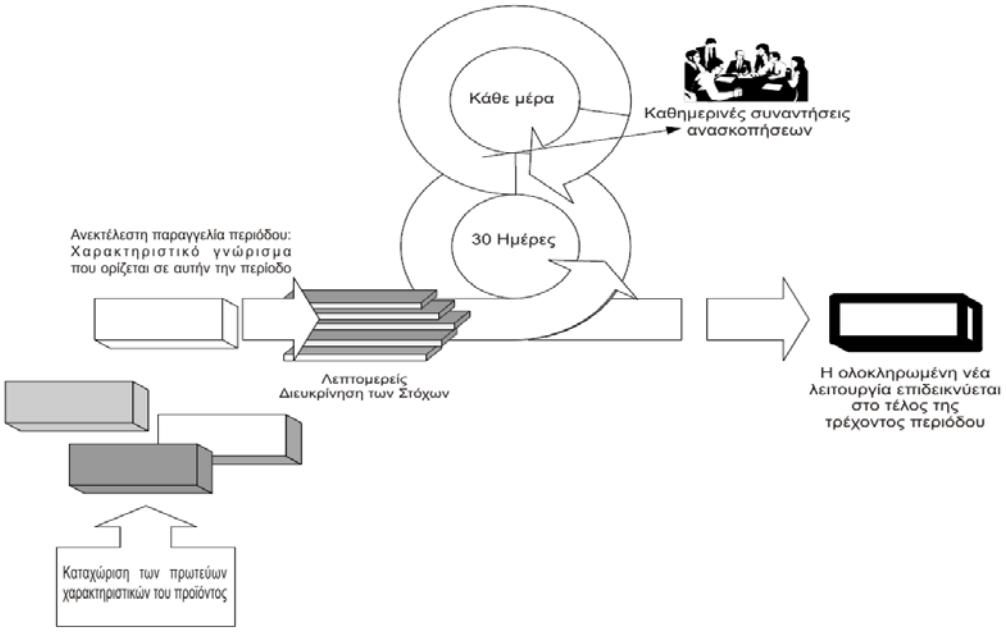
8.1 Εισαγωγή[17]

Η μέθοδος Scrum στοχεύει να διαχειριστεί και να ελέγξει την παραγωγή του λογισμικού χρησιμοποιώντας επαναληπτικές, επαυξητικές και ελαφριές διαδικασίες(οι οποίες είναι λιγότερο επιθετικές διαδικασίες). Αυτό το κάνει με το να περικλείει υπάρχουσες μεθόδους(όπως την RUP) και ευέλικτες μεθόδους(όπως η XP) μαζί ώστε να παρέχει μια εκμεταλλεύσιμη μεθοδολογία ευέλικτης ανάπτυξης.

Μια από τις ενδιαφέρουσες πλευρές της Scrum είναι ότι πραγματικά στοχεύει στο να βοηθήσει με την παραγωγή ενός προγράμματος του οποίου το λογισμικό είναι μόνο ένα κομμάτι. Τα πλεονεκτήματα της χρήσης του Scrum είναι:

1. Η διαχείριση και ο έλεγχος της ανάπτυξης λειτουργούν με ένα ευέλικτο τρόπο.
2. Αναγνωρίζει ότι οι απαιτήσεις μπορούν να αλλάξουν ραγδαία κατά την διάρκεια της επαναληπτικής και της επαυξητικής προσέγγισης στην ανάπτυξη του προϊόντος.
3. Είναι δυνατόν ακόμα να χρησιμοποιούνται υπάρχουσες πρακτικές εφαρμοσμένης μηχανικής μέσα στη Scrum(που μπορεί να βοηθήσουν στην παρουσίαση ευέλικτων μεθόδων σε έναν οργανισμό)
4. Είναι μια μέθοδος που βασίζεται στην ομαδικότητα και βοηθάει στο να βελτιωθούν οι επικοινωνίες και η συνεργατικότητα.
5. Διαβαθμίζετε από μικρά σε πολύ μεγάλα έργα.
6. Βοηθάει στο να αναγνωριστούν και να αφαιρεθούν οποιαδήποτε εμπόδια στην ομαλή λειτουργία της ανάπτυξης του τελικού προϊόντος.

Στον πυρήνα της η Scrum είναι μια συλλογή από κανόνες, διεργασίες και πρακτικές που όλες σχετίζονται μεταξύ τους και αυτό λειτουργεί έτσι ώστε να βελτιώσουμε το αναπτυσσόμενο περιβάλλον, μειώνει τα λειτουργικά έξοδα του οργανισμού και διασφαλίζει ότι οι επαναληπτικές παραδοτέες εκδόσεις ταιριάζουμε με τις τελικές απαιτήσεις του πελάτη.



Σχήμα 8:Κύκλος ζωής της Scrum

Η Scrum βασίζεται στις τρέχουσες διαδικασίες ελέγχου και συγκεκριμένα στοχεύει στο να παράγει το καλύτερο τελικό αποτέλεσμα, από τους δοθέντες τρέχοντες πόρους και τον διαθέσιμο χρόνο. Πρέπει να σημειωθεί ότι δεν στοχεύει να παράγει το καλύτερο δυνατό κομμάτι λογισμικού χρησιμοποιώντας απεριόριστους πόρους, αλλά μάλλον βασίζεται στην πραγματικότητα του σύγχρονου κόσμου και θα βοηθήσει να παραχθεί το καλύτερο που μπορεί να γίνει από τις δοθείσες συνθήκες που ισχύουν. Αυτό μπορεί να σημαίνει ότι μερικές λειτουργίες, για παράδειγμα, θα θυσιαστούν(κυρίως όταν αυτές οι λειτουργίες είναι χαμηλής προτεραιότητας λειτουργίες). Πρέπει να σημειώσουμε ότι καθώς θα έχουμε ένα επαναληπτικό κύκλο των 30 ημερών μεταξύ των επαναληπτικών παραδόσεων, αυτό απαιτεί και καθημερινές ανασκοπήσεις. Αυτές οι ανασκοπήσεις θα πρέπει να είναι σύντομες(για παράδειγμα των 15 λεπτών διάρκειας) και πρέπει να ωθεί τα μέλη της ομάδας να δρομολογήσουν τα βασικά στοιχεία των διαδικασιών, για να τα αναφέρουμε και ονομαστικά πρέπει να λάβουν υπόψη:

1. Τι έχει γίνει από τα μέλη της ομάδας από την τελευταία συνάντηση τους.
2. Υπάρχει κάτι που δημιουργεί πρόβλημα; Υπάρχουν τίποτα εμπόδια που ενοχλούν στην ολοκλήρωση της εργασίας;
3. Τι πρέπει να κάνει το κάθε μέλος της ομάδας πριν από την επόμενη συνάντηση;

Μια ενδιαφέρουσα παρατήρηση που σχετίζεται με την Scrum είναι ότι μπορεί να παρουσιαστεί σαν μια διαδικασία που βοηθάει στην ολοκλήρωση των υπάρχων πρακτικών εφαρμοσμένης μηχανικής μέσα από μια ελεγχόμενη επαναληπτική διαδικασία. Κάνοντας αυτό, παρέχονται αξίες, διαδικασίες και κανόνες που μπορούν να συστήσουν μια πιο δυναμική και υπεύθυνη διαδικασία ανάπτυξης(με άλλα λόγια, βοηθάει άλλες διαδικασίες να γίνουν ευέλικτες).

Η Scrum μπορεί να εφαρμοστεί από την αρχή του έργου ή μπορεί να παρουσιαστεί κατά την διάρκεια του κύκλου ζωής ενός έργου, ιδιαίτερα αν ένα έργο αντιμετωπίζει δυσκολίες στην ολοκλήρωση όλων των εργασιών και στον κατά κάποιας μορφής καθορισμό των προτεραιοτήτων των στόχων που απαιτούνται.

8.2Τι είναι Scrum[17]

Η πρώτη ερώτηση που πιθανόν κάποιος θέλει να απαντηθεί είναι: Είναι η Scrum(στρίμωγμα) ακρώνυμο από κάτι; Δεν θα είναι ο μόνος που θα σκέφτεστε κάτι τέτοιο διότι οι περισσότερες διαδικασίες διαχείρισης έργων και τακτικές είναι ακρώνυμα. Η Scrum ωστόσο είναι απλά ένα όνομα. Ομοίως δεν έχει σχέση με το Ράγκμπυ. Ωστόσο υπάρχει μια καθημερινή συνάντηση που μπορεί να συγκριθεί με ένα παραδοσιακό στρίμωγμα στο Ράγκμπυ.

Η Scrum δανείζεται από τον ευέλικτο κόσμο μέσο της προώθησης των επαναληπτικών και επαυξητικών πρακτικών της. Αυτό έχει μια απλή εφαρμογή, ότι έχει σχεδιαστεί να αυξήσει την παραγωγικότητα και να το χρόνο που χρειάζεται για να επωφεληθεί από την ανάπτυξη ενός λογισμικού/προϊόντος. Το πιο σημαντικό είναι ότι περικλείει την προσαρμοστική και εμπειρική ανάπτυξη συστημάτων.

Η πλειοψηφία των μεθόδων και τεχνικών διαχείρισης έργων είναι πολύ συγκεκριμένες, μας δεσμεύουν σε μια αμετάβλητη ακολουθία γεγονότων και προσφέρουν πολύ λίγα σε σχέση με την ευελιξία. Ομοίως οι νεώτερες, λιγότερο υφιστάμενες μέθοδοι συχνά υποστηρίζουν ότι είναι μια πανάκεια, και ακόμα στερούνται από οποιαδήποτε οριστική απόδειξη ή ιστορική διαδρομή. Ευτυχώς η Scrum είναι ώριμη, έχει μια καταγεγραμμένη διαδρομή που εκτείνεται από το 1995 και πιο πριν. Ομοίως είναι διαβαθμισμένη, Η Scrum μπορεί να χρησιμοποιηθεί σε έργα οποιουδήποτε μεγέθους.

Η Scrum επίσης προωθείτε και δανείζεται για την διαχείριση των έργων που βασίζονται στην XP. Η XP χρησιμοποιεί την εκ όψεως επαφή με τον πελάτη(costumer on site) σαν ένα μέσο για να διασφαλίσει ότι οι ερωτήσεις της ομάδας ανάπτυξης έχουν απαντηθεί αλλά επίσης και για να διασφαλίσει ότι αυτό που παράγει η ομάδα ανάπτυξης είναι αυτό που ο πελάτης πραγματικά θέλει.

Η Scrum[17] παρόλο που είναι μια από τις πολλές ευέλικτες μεθόδους για την ανάπτυξη και την παραγωγή λογισμικού, μπορεί επίσης να χρησιμοποιηθεί και για μη λογισμικού εργασίες όπου χρειάζονται πολλά άτομα να εργαστούν μαζί για να επιτύχουν ένα στόχο. Ο κύριος σκοπός μια ευέλικτης μεθόδου είναι να παραδώσει τις αξίες νωρίς στο κύκλο ζωής του έργου σύμφωνα πάντα με τις απαιτήσεις του πελάτη ή της αγοράς. Η ικανότητα να παραδώσει τις αξίες νωρίς και σύντομα θεωρείται ότι είναι ο κυριότερος συντελεστής στο να κάνει τις ευέλικτες μεθόδους μια από τις πιο ραγδαία αναπτυσσόμενες τάσεις στην τεχνολογία.

Παρακάτω[17] παρουσιάζονται τα χαρακτηριστικά της Scrum σε αντίθεση με τα παραδοσιακά χαρακτηριστικά καταρράκτη.

- **Μια δυναμική Λίστα του Προϊόντος(Product Backlog)με κατανεμημένες ιεραρχικά εργασίες που πρέπει να γίνουν.** Αυτή η Λίστα μπορεί ελάχιστα να συγκριθεί με τα παραδοσιακά έργα ή με μικρότερες προσπάθειες παραγωγής λογισμικού που περιμένουν να γίνουν ενεργές. Μαζί με την Έκδοση του Σχεδιασμού, η Λίστα του Προϊόντος θα συνταχθεί από κοινού από τον Ιδιοκτήτη της επιχείρησης του έργου και την ομάδα Ανάπτυξης.
- **Η έκδοση του σχεδίου για την παράδοση μεγάλων προσπαθειών δια μέσου πολλαπλών ροών, με πρώτη αυτή με την υψηλότερη προτεραιότητα.** Η έκδοση του σχεδίου(Release Plan) είναι παρόμοια με το παραδοσιακό Πρόγραμμα του Έργου(Project Schedule) στο οποίο διευκρινίζονται τα χαρακτηριστικά του προϊόντος και τα αντίστοιχα χρονικά πλαίσια(πιθανόν φάσεις) στα οποία τα χαρακτηριστικά θα παραδοθούν αν και σε υψηλότερο επίπεδο από το παραδοσιακό σχέδιο. Η ποιότητα που σχετίζεται με τα χαρακτηριστικά, τα ρίσκα, οι εξαρτήσεις, οι περιορισμοί, οι υποθέσεις και τα διάφορα προβλήματα μπορεί επίσης να αναγνωριστούν και να τεκμηριωθούν σαν μέρος της έκδοσης του σχεδίου, το οποίο δημιουργείται από τον Ιδιοκτήτη του Προϊόντος και της Ομάδας Ανάπτυξης.
- **Μια συνάντηση για τον προγραμματισμό των ροών στην οποία η Λίστα των αντικείμενων επιλέγονται για επανάληψη ονομάζεται Ροές(Sprints).**(οι οποίες συνήθως διαρκούν 30 ημέρες). Ο Ιδιοκτήτης του Προϊόντος και η Ομάδα Ανάπτυξης οριστικοποιούν τα χαρακτηριστικά και προσδιορίζουν τις σχετικές εργασίες οι οποίες πρέπει να ολοκληρωθούν μέσα στη Ροή, και θέτει εργασίες υπολογίζοντας τες σαν μέρος του σχεδιασμού. Όπου μπορεί να εφαρμοστεί, η έκδοση του Σχεδίου θα αναφερθεί κατά την διάρκεια της Συνάντησης για τον Προγραμματισμό των Ροών. Τα ρίσκα και τα διάφορα προβλήματα επίσης συζητούνται κατά την διάρκεια της Συνάντησης. Η Ροή που προκύπτει αποτελεί μια ένωση από το Σχέδιο(Planning), την Ανάπτυξη(Development), τις Δοκιμές(Testing) και την Έκδοση των εργασιών και των δραστηριοτήτων του κύκλου ζωής του έργου(Release Project Lifecycle tasks and activities)
- **Μια Λίστα Ροών(Sprint Backlog) ή ένα Σχέδιο Ροών(Sprint Plan) από εργασίες που πρέπει να γίνουν κατά τη διάρκεια της Ροή.** Όταν οι εργασίες προσδιορίζονται στη Συνάντηση ή κατά την διάρκεια της Ροής, μπαίνουν μέσα στη Λίστα Ροών. Η Λίστα αυτή σχετίζεται με την Προοπτική του Έργου(Project Scope) στη παραδοσιακή ανάπτυξη των έργων διότι περιλαμβάνει δραστηριότητες και προϊόντα τα οποία πρέπει να ολοκληρωθούν μέσα στη Ροή. Όταν μια συγκεκριμένη Ροή είναι μέρος της Έκδοσης του Σχεδίου που περιλαμβάνει πολλαπλές Ροές, η Λίστα της μπορεί να συγκριθεί με τα προϊόντα κάποιας φάσης

της παραδοσιακή ανάπτυξης ενός έργου. Αυτοί που κατασκευάζουν την Λίστα Ροών είναι ο Ιδιοκτήτης του Προϊόντος και η Ομάδα Ανάπτυξης.

- **Μια σύντομη καθημερινή Συνάντηση για την Ροή, στην οποία η πρόοδος κάθε μέλους την ομάδας φανερώνεται, η επερχόμενη εργασία περιγράφεται και αποδίδεται, και τα εμπόδια ξεπερνιούνται.** Η Λίστα των Ροών ενημερώνεται στη συνάντηση, και οι επιχειρηματικοί συνεργάτες είναι τακτικά μέλη την Ομάδας Scrum. Οι καθημερινές συναντήσεις είναι ο κυριότερος επίσημος τρόπος επικοινωνίας μεταξύ των μελών της ομάδας, παρόλα αυτά οι μη-επίσημες συναντήσεις και άλλου είδους επικοινωνίες κατά την διάρκεια της μέρας ενθαρρύνονται.
- **Ένα Δείγμα(Demo) στο οποίο στον Ιδιοκτήτης του Προϊόντος(και περιστασιακά οι κατασκευαστές) επιδεικνύονται τα μέχρι τώρα επιτεύγματα κατά την διάρκεια της Ροής.** Κάθε Ροή πρέπει να παραδώσει ένα λειτουργικό επαυξημένο προϊόν. Το Δείγμα δεν έχει καμιά ταύτιση στη παραδοσιακή ανάπτυξη έργου.
- **Μια ανασκόπηση, στην οποία τα μέλη της ομάδας αναλογίζονται τις περασμένες ροές και κάνουν προτάσεις για μελλοντικές βελτιώσεις και αλλαγές.** Η ανασκόπηση μπορεί ελάχιστα να συγκριθεί με τις Μετά-Εφαρμογής Αναθεωρήσεις(Post-Implementation Reviews) στην παραδοσιακή ανάπτυξη έργου.

Η Scrum[17] διευκολύνεται από ένα Ειδικό της Scrum(Scrum Master) του οποίου η κύρια υπευθυνότητα είναι να απομακρύνει εμπόδια τα οποία καθυστερούν την ομάδα από το να παραδώσει τους στόχους της Ροής. Ο Ειδικός της Scrum δεν είναι ο αρχηγός της ομάδας διότι η ομάδα αυτό-οργανώνεται. Παρόλα αυτά ενεργεί σαν βοηθός για την επίλυση των διαφόρων θεμάτων και την επικοινωνίας παρά σαν διαχειριστής για τον έλεγχο της ομάδας. Ο Ειδικός σημειώνει και αφαιρεί εμπόδια, διασφαλίζει την διαδικασία της Scrum, διευκολύνει την συνεργασία, και ενεργεί σαν φρουρός για την ομάδα.

Σχετικά[17] λίγα έχουν γραφεί για τον έλεγχο του κόστους και του προϋπολογισμού όσων αφορά την Scrum. Κάποιες επιχειρήσεις θεωρούν το κόστος έναν παράγοντα όταν εξετάζουν ποια χαρακτηριστικά θα περιληφθούν σε μια Ροή. Γενικά η ίδια η οργάνωση αφήνεται για να αποφασίσει ποιος θα είναι υπεύθυνος για τον έλεγχο του προϋπολογισμού σε μια Scrum προσπάθεια.

Η Scrum[17] χρησιμοποιεί μια εμπειρική προσέγγιση. Αντίθετα από την μέθοδο Καταρράκτη, η Scrum δέχεται ότι μια προσπάθεια για κατασκευή λογισμικού δεν μπορεί να κατανοηθεί ολοκληρωτικά ή να οριστεί ακριβώς και ότι οι απαιτήσεις θα αλλάξουν με την πάροδο του χρόνου. Ο αντικειμενικό σκοπός της Scrum είναι να μεγιστοποιήσει την ικανότητα της ομάδας να ανταποκριθούν σε μια απαιτητική αλλαγή, και να παράγει ένα λειτουργικό προϊόν το οποίο παρουσιάζετε και γίνετε δεκτό από τον χρήστη σε κάθε Ροή.

8.3Το αντίθετο της μεθόδου Καταρράκτη(Waterfall)[17]

Παραδοσιακά ακολουθούμε έναν κύκλο που περιλαμβάνει της συγκομιδή των απαιτήσεων, την ανάλυση τους, τον σχεδιασμό τους, την ανάπτυξη, τις δοκιμές, επεκτείνοντας τες με την ολοκλήρωση κάθε σταδίου πριν προχωρήσουμε στα επόμενα. Αυτό ήταν γνωστό σαν την προσέγγιση Καταρράκτη. Ωστόσο υπάρχει ένα σημείο στην προσέγγιση του Καταρράκτη, το οποίο έχει γίνει το αντικείμενο ενός χείμαρρου αντιδράσεων τα πιο πρόσφατα χρόνια. Το πιο αξιόλογο είναι ότι η μέθοδος Καταρράκτη προωθεί την δημιουργία της άμεσης τεκμηρίωσης πριν δημιουργηθεί οποιαδήποτε πραγματική εργασιακή αξία. Αυτό μπερδεύεται από το γεγονός ότι η ανάπτυξη του προϊόντος αρχίζει πολύ αργότερα από το αναμενόμενο χρονικό πλαίσιο του έργου.

Αλλά μπορεί τα πράγματα να γίνουν και πολύ χειρότερα, ο πελάτης/χρήστης προσπαθεί επίμονα να διασφαλίσει ότι όλες οι απαιτήσεις του έχουν τεκμηριωθεί κατά την διάρκεια των πρωίμων σταδίων, κατά συνέπεια αυτά τα χαρακτηριστικά γίνονται υψηλής προτεραιότητας. Η αποτυχία στον καθορισμό προτεραιοτήτων των χαρακτηριστικών συχνά έχει σαν αποτέλεσμα ένα χαμηλής ποιότητας σύστημα το οποίο είναι γεμάτο με χαρακτηριστικά τα οποία ο πελάτης/χρήστης δεν τα χρειάζεται στην πρώτη έκδοση.

Με αυτό στο μυαλό μπορεί να χτιστεί ένα προϊόν(ένα λογισμικό προϊόν) που να παρέχει το 20% του συνόλου των χαρακτηριστικών. Το οποίο θα μπορεί και θα γίνει επαναληπτικά. Μπορεί να παραδοθεί πρώτα η πρώτη έκδοση με το 20% των χαρακτηριστικών και μετά από μικρό χρονικό διάστημα η $2^{η}$ έκδοση με την επί πλέον συλλογή των χαρακτηριστικών. Η ομορφιά αυτής της προσέγγισης είναι ότι η ανάπτυξη του 20% των χαρακτηριστικών δεν θα χρειαστεί το 100% του αναμενόμενου χρόνου και πόρων του έργου.

Η Scrum ακολουθεί την αντίθετη προσέγγιση από αυτή του Καταρράκτη κατά την οποία ξεκινάει η ανάλυση αμέσως αφότου υπάρξουν κάποιες απαιτήσεις, όσο πιο γρήγορα υπάρξει ανάλυση τόσο πιο γρήγορα θα ξεκινήσει ο σχεδιασμός και κάπως έτσι συνεχίζεται. Με άλλα λόγια η εργασία γίνεται σε μικρά κομμάτια κάθε φορά. Αυτή η προσέγγιση μπορεί να αναφερθεί σαν επαναληπτική. Κάθε επανάληψη απαρτίζεται από κάποιες απαιτήσεις που συγκεντρώθηκαν, κάποια ανάλυση, κάποιο σχεδιασμό, κάποια ανάπτυξη και κάποιες δοκιμές.

8.4Βασικά στοιχεία της Scrum(Scrum Basics)[17]

Η Scrum περιστρέφεται γύρω από τις αξίες της απλότητας, με συνέπεια την παράδοση οτιδήποτε το οποίο οδηγεί το έργο μπροστά. Αυτό πραγματοποιείται με την πρόταση των επόμενων ερωτήσεων(οι οποίες είναι γνωστές σαν “Οι τρεις ερωτήσεις της Scrum”(Scrum’s three questions))

Πίνακας 6: Οι 3 βασικές Ερωτήσεις της Scrum

Η Scrum ρωτάει…	**Ζητήματα της θεμελιώδης Διαχείρισης του Έργου**
Τι έχει συμβεί κατά την διάρκεια των τελευταίων 24^{ων} ωρών;	Αυτή είναι πρόοδος, είναι η δουλειά που έχει ολοκληρωθεί μέχρι σήμερα
Τι έχει σχεδιαστεί να γίνει στις επόμενες 24 ώρες;	Αυτός είναι ο επόμενος προγραμματισμός, είναι η εργασία που πρόκειται να συμβεί
Τι αποτρέπει την εργασία των επόμενων 24^{ων} ωρών;	Αυτά είναι τα εμπόδια, είναι πράγματα που μπορεί να χρειάζονται για να συνεχιστεί η εργασία. Είναι επίσης η αναγνώριση των άμεσων κινδύνων.

8.5Διαχείριση Έργου Σύγκριση Παραδοσιακής Μεθόδου με Scrum[17]

Ο παρακάτω πίνακας παρουσιάζει μια σύγκριση των πρακτικών και των προϊόντων για την Διαχείριση του Έργου με την παραδοσιακή μεθοδολογία και την Scrum.

Πίνακας 7: Σύγκριση πρακτικών και προϊόντων της Μεθόδου Καταρράκτη με την Scrum

Αντικείμενα (ITEM)	**Παραδοσιακή μέθοδο Καταρράκτη (Traditional Waterfall)**	**Scrum**
Διαδικασίες (Processes)		
Σχεδιασμός Έργου(Στόχος, Πρόγραμμα, Επικοινωνία και Ανθρώπινοι πόροι)	– Δεν υπάρχει αντιστοιχία της Διορατικότητας(Vision) στα έργα της μεθόδου του Καταρράκτη, παρόλο που μια εταιρική Στρατηγική Οδηγία (Strategic Directive) μπορεί να παραχθεί για να διευκρινίσει την κατεύθυνση και τελικά τα έργα που το υποστηρίζουν. – Δεν υπάρχει αντιστοιχία ενός Χάρτη(Roadmap) στα έργα της μεθόδου του Καταρράκτη, παρόλο που οι συνεργάτες μπορούν να	5 Επίπεδα Σχεδιασμού – Διορατικότητα(Vision) (μια σύντομη δήλωση που διευκρινίζει την κατεύθυνση) παράγεται από τον Ιδιοκτήτη του Προϊόντος. – Χάρτης(Roadmap) (ένα σύντομο έγγραφο που αποτελείται από αξία 1^{ος} έτους υψηλού επιπέδου χαρακτηριστικά)

	παράγουν τα δικά τους εσωτερικά Στρατηγικά Σχέδια(Strategic Plans) για υποστήριξη των Στρατηγικών Οδηγιών (Strategic Directives). – Μόλις ένα έργο έχει ορισθεί και εγκριθεί, ο PM οδηγεί την συγκέντρωση των απαιτήσεων και την προσπάθεια εκτίμηση του χρόνου με το να γίνονται περιεκτικές συναντήσεις με τον Αναλυτή της Επιχείρησης, του Σχεδιαστές , του Κατασκευαστές, τον Ιδιοκτήτη του Προϊόντος και τους βασικούς μετόχους. Ο PM επιβλέπει την κατασκευή της τεκμηρίωσης που σχετίζεται με τα προϊόντα όπως την Επικοινωνία, τον Σχεδιασμό, τις Απαιτήσεις και άλλα. – Ο PM κατασκευάζει το Σχέδιο του Έργου για να παραγάγει ένα πρώτο πρόγραμμα για το έργο και στη συνέχεια αφού το αξιολογήσει με τα μέλη της ομάδας και τον διαχειριστή το ακολουθεί. – Ο PM συναντάται μαζί με την ομάδα του έργου περιοδικά(όπως έχει ορισθεί μέσα στο Επικοινωνιακό Σχέδιο) για να αναθεωρήσουν το Σχέδιο του Έργου με πραγματικές ώρες και να αναθεωρήσουν τις εκτιμήσεις και να συζητήσουν τα ρίσκα και/ή τα διάφορα θέματα. Ο PM είναι επίσης υπεύθυνος με την τεκμηρίωση των κινδύνων και των θεμάτων και επιβλέπει την επιτυχή κατάληψη τους. –Η Διαχείριση της Πορείας(Line Management) αποφασίζει για την κατανομή των πόρων στην ομάδα του έργου. Ο PM μπορεί να διαπραγματευτεί με την Διαχείριση ανά πάσα στιγμή για πόρους αν οι διαθέσιμοι πόροι είναι ανεπαρκείς να ολοκληρώσουν τους στόχους του έργου μέσα στο προβλεπόμενο πρόγραμμα. – Το έργο συνήθως τελειώνει στο τέλος του προγραμματισμού και γενικά ενσωματώνει όσο το δυνατών περισσότερες απαιτήσεις μπορεί.	κατασκευασμένο από τον Ιδιοκτήτη του Προϊόντος και Ανώτερα Στελέχη – Έκδοση Σχεδίου, για να παραδώσει τις μεγαλύτερες προσπάθειες στις πολλαπλές ροές, συνταγμένο από τον Ιδιοκτήτη του Προϊόντος και την Ομάδα Ανάπτυξης. Η προσπάθεια διευκολύνεται συνήθως από τον Ειδικό της Scrum. – Το Σχέδιο Ροών προσδιορίζει όλες τις εργασίες και τις εκτιμώμενες ώρες εργασίας, φτιάχνεται από τον Ιδιοκτήτη του Προϊόντος και την Ομάδα Ανάπτυξης όχι όμως από τον Ειδικό της Scrum. Αναβαθμίζεται καθημερινά από την Ομάδα στις συναντήσεις μόλις οι ροές αρχίσουν. Μόνο οι εκτιμώμενες σημαντικές ώρες ακολουθούνται, οι πραγματικές ώρες ούτε ζητούνται ούτε καταγράφονται. – Καθημερινές συναντήσεις Ροών οι οποίες επιβλέπονται από τον Ειδικό της Scrum. Αυτός είναι ο κύριος τρόπος επικοινωνίας της Ομάδας. Ο Ειδικός της Scrum ζητάει από κάθε μέλος της Ομάδας ήταν να γίνει εχθές, τι είναι να γίνει σήμερα και αν υπήρχε οποιοδήποτε εμπόδιο. Ο Ειδικός της Scrum καταγράφει τα εμπόδια και επιβλέπει την επίλυση τους. Το Σχέδιο Ροών αναβαθμίζεται με τις εκτιμώμενες σημαντικές ώρες. – Η Διαχείριση της Πορείας διαθέτει τους πόρους των Ροών. Η διαπραγμάτευση της προτεραιότητας στην αρχή της Ροής μπορεί ή όχι να είναι δυνατή, εξαρτάται από την εταιρική προσαρμογή στην Scrum. Η σύνθεση της Ομάδας δεν αλλάζει κατά την διάρκεια της Ροής. – Το πρόγραμμα του Έργου προέρχεται από την Έκδοση του Σχεδίου αν είναι απαραίτητο. Κάθε Ροή είναι σταθερής

		διάρκειας και τα υψηλότερα επιπέδου αντικείμενα παραδίδονται στην αρχική Ροή.
Επιτήρηση των αλλαγών στον έλεγχο και τα ρίσκα του έργου (Στόχος και Ρίσκο)	– ΟΙ αλλαγές στις απαιτήσεις είναι τυπικά ελεγχόμενες από την Διαδικασία Ελέγχου Αλλαγών που επιβλέπετε από τον Διαχειριστή του Έργου. Οι δραστηριότητες περιλαμβάνουν ταξινομήσεις και λαμβάνουν την έγκριση της διαχείρισης. – Ο διαχειριστής του Έργου είναι υπεύθυνος για την Ανάλυση των Ρίσκων και τον Απρόοπτα Σχεδιασμό, και συνήθως διατηρεί και δημοσιεύει μια Τεκμηρίωση Ρίσκου.	– Μόνο η Ομάδα μπορεί να αλλάξει χαρακτηριστικά και εργασίες μέσα στη Ροή, και μόνο αν αυτό συμφωνηθεί από όλα τα μέλη της Ομάδας. Σε διαφορετική περίπτωση τα απαιτούμενα αντικείμενα προσθέτονται στην Λίστα Προϊόντος. – Η ομάδα εκτελεί δραστηριότητες διαχείρισης ρίσκων πριν από την αρχίσει την δουλεία της ανάπτυξης. Τα ρίσκα μπορεί να συζητηθούν στην Έκδοση, Στο Σχέδιο Ροής και/ή στις καθημερινές Scrum συναντήσεις.
Κυριότητα Στόχου, Πραγματικός/Εκτιμώμενος (Προγραμματισμός και Ανθρώπινοι Πόροι)	– Ο PM δημιουργεί ένα Σχέδιο Έργου με εργασίες, αναθέσεις και εκτιμώμενες ώρες. Το Σχέδιο αυτό επανεξετάζεται με την Ομάδα Ανάπτυξης και μπαίνει σε εφαρμογή μόλις όλα τα μέλη συμφωνήσουν. Παρεκκλίσεις από την βασική γραμμή πρέπει να εξηγηθούν και να εξεταστούν από τον PM. – Ο PM περιοδικά ανανεώνει το Σχέδιο του Έργου με τις πραγματικές ώες και με νέες εκτιμήσεις, καθώς επίσης και με πρόσθετες εργασίες που βασίζονται στις εισαγωγές δεδομένων της Ομάδας. – Ο Pm λειτουργεί σαν προπονητής και αρχηγός για την Ομάδα του Έργου και βοηθάει τα μέλη της Ομάδας στα επερχόμενα εμπόδια.	– Τα μέλη της Ομάδας δημιουργούν, υπογράφουν και εκτιμούν εργασίες αντί να τους ανατεθούν εργασίες από τον Ειδικό της Scrum. Δεν υπάρχει μια βασική γραμμή στη Scrum – ΤΑ μέλη της Ομάδας μπορεί να προσαρμόσουν τις εκτιμώμενες σημαντικές ώρες και να προσθέσουν, μεταφέρουν ή να ακυρώσουν εργασίες κατά την διάρκεια των καθημερινών Συναντήσεων. Οι πραγματικές ώρες δεν παρακολουθούνται, μόνο οι ώρες που είναι σημαντικές. – Ο Ειδικός της Scrum διευκολύνει την Ομάδα αλλά τυπικά δεν καθοδηγεί ούτε διευθύνει τα μέλη της Ομάδας τα οποία είναι συνεχώς αυτό-διαχειριζόμενα.
Υπερβάσεις Έργου (|Πρόγραμμα)	– Εάν η ημερομηνία ολοκλήρωσης του Έργου ή της φάσης φαίνεται να είναι σε κίνδυνο, Ο Pm είναι υπεύθυνος να διαπραγματευτεί ένα ή και περισσότερα από τα 5 ακόλουθα αντικείμενα: • Στόχος(μείωση) • Πρόγραμμα • Κόστος • Πόροι • Ποιότητα	– Η Scrum προωθεί ένα διορθωτικό χρονικό περιθώριο των 30 ημερών για παράδοση των προϊόντων. Αν παρουσιαστεί περίπτωση να καθυστερήσει η παράδοση κάποιου από τα προϊόντα, το προϊόν αυτό θα προστεθεί στη Λίστα των Προϊόντων και το πιο πιθανών είναι να προχωρήσει την επόμενη Ροή. Η ίδια η Ροή δεν επεκτείνεται ούτε ο Ειδικός της Scrum διαπραγματεύεται για επιπρόσθετους πόρους ή χρόνο.

Προϋπολογισμός του Έργου (Κόστος)	– Τυπικά ένα έργο πρέπει να παρέχει ROI(return on investment επιστροφή επενδύσεων) ώστε να μπορέσει να τεθεί σε εφαρμογή, έτσι σε αυτό το σημείο παραδίδεται ένα Προϋπολογισμός Έργου. – Ο PM συνήθως διαχειρίζεται τον προϋπολογισμό του έργου όποιος περιλαμβάνει τις δαπάνες των πόρων, τεχνικές δαπάνες, γνωμάτευση, υποστήριξη υπηρεσιακών δαπανών(αν είναι απαραίτητο και άλλες δαπάνες. Πιθανές υπερβάσεις πρέπει να δικαιολογούνται και να εγκρίνονται, και/ή ο PM πρέπει να διαπραγματευτεί προσαρμογές στα προαναφερθέντα 5 αντικείμενα ώστε να φτάσει τον προϋπολογισμό.	– Στην αρχή της κάθε Ροής ο Ιδιοκτήτης του Προϊόντος μπορεί να ερωτηθεί πόσα θέλει η ομάδα να ξοδέψει για την Ροή. Βασιζόμενοι στην απάντηση, η Ομάδα μπορεί να χρησιμοποιήσει το κόστος σαν οδηγό για την επιλογή ενός κατάλληλου ποσού για την εργασία που πρέπει να γίνει από την Λίστα του Προϊόντος στη Ροή. – Καμιά αναφορά δεν μπορεί να βρεθεί στις πληροφορίες της Scrum σχετικά με την συσχέτιση Ροής και προϋπολογισμού. Υποθέτουμε ότι κάθε οργανισμός υιοθετεί τις δικές του πρακτικές υπολογίζοντας το κόστος ανάπτυξης του λογισμικού.
Έλεγχος Έργου (Ποιότητα)	– Από την προοπτική του πελάτη ποιότητα σημαίνει παράδοση του συστήματος στην ώρα του, με τις προδιαγραφές και στα πλαίσια του προϋπολογισμού και του προγράμματος. – Τα Σχέδια Διαχείρισης της Ποιότητας(Quality Management Plans) και οι Λίστες Ελέγχου(Checklists) χρησιμοποιούνται συχνά για να τεκμηριώσουν και να διπλό τσεκάρουν την ποιότητα των απαιτήσεων. Παρόλο που ο PM δεν είναι προσωπικά υπεύθυνος για την σύνταξη τους, αυτός/αυτή είναι υπεύθυνος/νη για να διασφαλίσει ότι η ποιότητα των απαιτήσεων και των μετρικών πραγματοποιούνται κατά την διάρκεια του έργου. – Σε έργα που χρησιμοποιούν την μέθοδο καταρράκτη, το προσωπικό της διασφάλισης της ποιότητας (QA personnel) τυπικά εμπλέκεται στην φάση των δοκιμών.	– Στην Scrum, η ποιότητα συνεπάγεται την παράδοση ενός λειτουργικού επαυξητικού προϊόντος μέχρι το τέλος της Ροής σύμφωνα με τα καθορισμένα χαρακτηριστικά. Στις περιπτώσεις που η δαπάνη είναι ένας παράγοντας, περιορισμοί για τις δαπάνες πρέπει να ληφθούν επίσης υπόψη. – Όπου μπορούν να εφαρμοστούν τα ποιοτικά συσχετιζόμενα χαρακτηριστικά μπορεί να προσδιοριστούν μέσα στην Έκδοση και/ή στις Συναντήσεις Σχεδιασμού Ροής, και χτίζονται κατά την διάρκεια της Ροής. Όπως και κάθε άλλο χαρακτηριστικό, ολόκληρη η Ομάδα είναι υπεύθυνη για να παρέχει ποιοτικά χαρακτηριστικά. – Στη Scrum, το προσωπικό της διασφάλισης της ποιότητας (QA personnel) μπορεί να χρειαστεί για δοκιμές σε κάθε Ροη, το οποίο μπορεί να δημιουργήσει ένα επιπλέον βάρος κατά των πόρων σε αυτή την Ομάδα.

Έγγραφα (Documents)		
Ανάλυση Πλεονεκτημάτων Κόστους (Cost Benefit Analysis)	– Μια Ανάλυση των Πλεονεκτημάτων του Κόστους(CBA) μπορεί να κατατεθεί και να εγκριθεί ώστε να προχωρήσει και να τελειώσει ένα έργο. Ο PM μπορεί και δεν μπορεί να χρησιμοποιηθεί για να συντάξει το έγγραφο αυτό. Η ROI(return on investment επιστροφή επενδύσεων) είναι τυπικά ένα βασικός παράγοντας για το αν μια ενέργεια εγκριθεί για την ανάπτυξή της.	– Ο δημιουργός δεν μπορεί να εντοπίσει μια παρόμοια Ανάλυση Πλεονεκτημάτων Κόστους στην Scrum.
Απαιτήσεις (Requirements)	– Τα σχετικά έγγραφα περιλαμβάνουν αλλά δεν περιορίζονται σε: ➢ Μελέτη Σκοπιμότητας(Feasibility Study) ➢ Δήλωση της Εργασίας (Statement of Work) ➢ Σκοπός Έργου (Project Scope) ➢ Έγγραφο Απαιτήσεων (Requirements Document) ➢ Λειτουργικό Σχέδιο (Functional Design) ➢ Λεπτομερές Σχέδιο (Detailed Design) ➢ Σχέδια Δοκιμής και Περιπτώσεις Δοκιμής, που κάποιες φορές γίνονται στη Φάση Καθορισμού των Απαιτήσεων (Test Plans and Test Cases) – Ο PM είναι γενικά υπεύθυνος για την επιτήρηση ότι τα απαραίτητα σχετικά με τις απαιτήσεις έγγραφα είναι ολοκληρωμένα και εγκριμένα.	– Στην Scrum υπάρχουν 3 σχετικά με τις απαιτήσεις αντικείμενα: ➢ Η Λίστα του Προϊόντος (Product Backlog) ➢ Το Πρόγραμμα για την Έκδοση (Release Plan) ➢ Λίστα/Σχέδιο Ροής (Sprint Backlog/Plan) – Και τα 3 αντικείμενα ολοκληρώνονται από τον Ιδιοκτήτη του προϊόντος και την Ομάδα Ανάπτυξης, αν και ο Ειδικός της Scrum μπορεί να διευκολύνει τις συναντήσεις.
Έλεγχος Αλλαγής (Change Control)	– Ο PM είναι αρμόδιος για την επιτήρηση ότι οι αλλαγές στον αρχικό στόχο του έργου έχουν τεκμηριωθεί, κατατεθεί και εγκριθεί.	– Μόλις μια Ροή είναι σε εξέλιξη ολόκληρη η ομάδα πρέπει να συμφωνήσει για μια αλλαγή πριν αυτή γίνει δεκτή. Άλλες αλλαγές μπορούν να προστεθούν στη Λίστα του Προϊόντος ανά πάσα στιγμή.
Σχέδιο Επικοινωνίας (Communication Plan)	– Ο PM συντάσσει και διαδίδει ένα Σχέδιο Επικοινωνίας αποτυπώνοντας τα σχετικά με την επικοινωνία παραδοτέα προϊόντα του έργου(όπως οι Συναντήσεις της Ομάδας και οι Εκθέσεις Κατάστασης της Οργανωτικής Επιτροπής) που καθορίζει ποιος θα εμπλακεί σε αυτές και πόσο συχνά.	– Δεν υπάρχει κάτι αντίστοιχο του Σχεδίου Επικοινωνίας στην Scrum.
Υπολογισμός Προϋπολογισμού με λογιστικό φύλλο (Budget Spreadsheet)	– Ο Pm είναι υπεύθυνος για να επιβλέπει και να δίνει αναφορές για τον Προϋπολογισμό του Έργου	– Ο δημιουργός δεν μπορεί να βρει καμία αναφορά της διατήρησης του προϋπολογισμού του έργου

		από την πλευρά της Scrum
Αρχείο καταγραφής ζητημάτων (Issus Log)	– Ο PM είναι υπεύθυνος για την σύνταξη και ενημέρωση του Αρχείου καταγραφής Ζητημάτων όπως επίσης και για την επίβλεψη της λύσης των ζητημάτων αυτών.	– Ζητήματα και εμπόδια μπορούν να εμφανιστούν ανά πάσα στιγμή, και ο Ειδικός της Scrum είναι υπεύθυνος για την εγγραφεί αυτών, απομάκρυνση των εμποδίων και επίβλεψη της περάτωσης τους.
Ανάλυση του Ρίσκου (Risk Analysis)	– Ο PM είναι υπεύθυνος για την σύνταξη και ενημέρωση της Ανάλυσης του Ρίσκου και του Σχεδίου Έκτακτης Ανάγκης, και για την ασφαλή καθοδήγηση της Ομάδας μέσα από τα ρίσκα ώστε να παραδοθεί το έργο επιτυχώς.	– Τα ρίσκα καταγράφονται από τον Ιδιοκτήτη του Προϊόντος και την Ομάδα Ανάπτυξης στο Προγραμματισμό της Έκδοσης. Ολόκληρη η Ομάδα είναι υπεύθυνη για την επίλυση των ρίσκων σε όλη την Ροή.
Σχέδιο Έργου (Project Plan)	– Ο PM χρησιμοποιεί το Σχέδιο του Έργου για να παράγει τον αρχικό προγραμματισμό του έργου. Ο PM μετά βασίζεται στον προγραμματισμό και συναντιέται περιοδικά με την ομάδα για να ανανεώσει το Σχέδιο του έργου με πραγματικές ώρες και αναθεωρημένες εκτιμήσεις/στόχους.	– Το Σχέδιο Ροής(εργασίες και ώρες ανά Ροή) δημιουργείται από τον Ιδιοκτήτη του Προϊόντος και την Ομάδα, όχι από τον Ειδικό της Scrum. Αναβαθμίζεται καθημερινά από την ομάδα μόλις η Ροή ξεκινήσει. Μόνον οι εκτιμώμενες πιο σημαντικές ώρες παρακολουθούνται. Το Σχέδιο Ροής δεν είναι βασική γραμμή
Αναφορά Κατάστασης (Status Report)	– Ο PM παράγει και παραδίδει την Αναφορά Κατάστασης του Έργου όπως είναι εξουσιοδοτημένος.	– Η Scrum δεν παραδίδει συγκεκριμένα αναφορά κατάστασης του έργου. Η Scrum είναι γενικώς πιο ελεύθερη σχετικά με την τεκμηρίωση, δίνει έμφαση στην προσωπική αλληλεπίδραση.
Σχέδιο Διαχείρισης της Ποιότητας, Λίστες Ελέγχου (Quality Management Plan, Checklists)	– Το Σχέδιο Διαχείρισης της Ποιότητας και Λίστες Ελέγχου συχνά χρησιμοποιούνται για να τεκμηριώσουν και διπλό τσεκάρουν την ποιότητα των απαιτήσεων. Ωστόσο ο PM μπορεί να μην είναι προσωπικά υπεύθυνος για την σύνταξη τους, αυτός/τη είναι υπεύθυνος/νη για την διασφάλιση της ποιότητας των απαιτήσεων και των μετρικών που πραγματοποιούνται στο έργο.	– Όποτε μπορούν να εφαρμοστούν, τα σχετικά με την ποιότητα χαρακτηριστικά μπορούν να καθοριστούν στην Έκδοση και/ή στις προγραμματισμένες συναντήσεις και να κατασκευαστούν κατά την διάρκεια της Ροής. Όπως και με κάθε άλλο χαρακτηριστικό ολόκληρη η Ομάδα είναι υπεύθυνη για την προσφορά στις εργασίες και τις δραστηριότητες ώστε να υποστηρίξουν τα σχετικά με την ποιότητα αντικείμενα.
Σχέδια Δοκιμής (Test Plans)	– Παρόλο που ο PM δεν γράφει ο ίδιος τα σχέδια δοκιμής είναι υπεύθυνος για την διασφάλιση ότι τα σχέδια δοκιμής είναι τεκμηριωμένα και πλήρως ανεπτυγμένα.	– Η Scrum γνωρίζει ότι μια βιώσιμη επαύξηση προϊόντος πρέπει να παραδοθεί και να γίνει δεκτή σε μια Ροή. Εξαρτάται από τον Ιδιοκτήτη του Προϊόντος να καθορίσει και εν τέλει να δοκιμάσει την βιωσιμότητα μέσα στη Ροή.

Μετά την εφαρμογή Υποστήριξη (Post-Implementation Support)	– Ο PM είναι υπεύθυνος για να διασφάλιση ότι η μετέπειτα υποστήριξη είναι διαθέσιμη για το προϊόν.	– Μια σταθεροποιητική ροή μπορεί να έχει προγραμματιστεί για το Σχέδιο Έκδοσης και να εκτελεστεί όποτε είναι απαραίτητο.
Συναντήσεις Ομάδας(Team Meetings)	–	–
Συναντήσεις για την κατάσταση από την Οργανωτική επιτροπή(Steering Committee Status Meetings)	– Για Υψηλή διαφάνεια και υψηλού κινδύνου προσπάθειες ο PM θα φτιάξει και θα συμμετέχει σε τακτικές προγραμματισμένες συναντήσεις της οργανωτικής επιτροπής(σαν μέρος του Σχεδίου Επικοινωνίας) στις οποίες η κατάσταση του έργου θα συζητείται.	– Ο δημιουργός δεν μπορεί να βρει καμία αναφορά για τις Συναντήσεις τις οργανωτικές επιτροπής που να έχουν σχέση με την Scrum.
Συναντήσεις την Ομάδας(Team Meetings)	– Σαν μέρος του Σχεδίου της Επικοινωνίας ο PM συνήθως θα προγραμματίζει τακτικές συναντήσεις της ομάδας γα να συζητήσουν την πρόοδο, τα πραγματικά/εκτιμώμενα ζητήματα και κινδύνους. Τέτοιες συναντήσεις μπορεί να διαρκέσουν 1 ώρα και πάνω.	– Στην Scum οι 15λεπτες καθημερινές συναντήσεις είναι ο βασικός φορέας για την συζήτηση της προόδου, των εμποδίων και των επερχόμενων υποχρεώσεων.
Μετά την εφαρμογή Αξιολόγηση (Post-Implementation Review)	– Δεν υπάρχει τίποτα αντίστοιχο την Επίδειξης της Ροής στην παραδοσιακή ανάπτυξη λογισμικού με την μέθοδο καταρράκτη. – Είναι κοινό για τον PM να κρατάει μια μετά την εφαρμογή αξιολόγηση με την Ομάδα του Έργου στο τέλος του έργου για να καταγράψει τι πήγε καλά και τι μπορεί να βελτιωθεί.	– Στο τέλος κάθε Ροής, μένα πρότυπο κρατιέται το οποίο επιδεικνύει το προϊόν στον παραλήπτη με τις προσθέσεις που έχουν γίνει κατά την διάρκεια της Ροής. – Μια ανασκόπηση κρατιέται, η οποία τα μέλη της Ομάδας συλλογίζονται σχετικά με την περασμένη Ροή και κάνουν προτάσεις σχετικά με τις μελλοντικές βελτιώσεις ή αλλαγές. Ο Ειδικός της Scrum τυπικά κρατάει την ανασκόπηση.

8.6Οι ρόλοι της Scrum(Scrum Roles)[21]

Η Scrum χρησιμοποιεί 3 ρόλους, ο Ιδιοκτήτης του Έργου(Product Owner), Ο Ειδικός της Scrum(Scrum Master) και η Ομάδα του Έργου(Project Team).

Ο Ιδιοκτήτης του Έργου(Product Owner) είναι πιθανόν ο Διαχειριστής του έργου ή ο Σπόνσορας του έργου, ένα μέλος του μάρκετινγκ ή ένας Εσωτερικός Πελάτης.

Ο Ειδικός της Scrum(Scrum Master) είναι κλειδί, αυτός ή αυτή αντιπροσωπεύει την διαχείριση του έργου. Ένας τέτοιος ρόλος συνήθως καταλαμβάνεται από έναν Διαχειριστή του Έργου ή τον Αρχηγό της Ομάδας. Είναι υπεύθυνοι για την θέσπιση των αξιών και των πρακτικών της Scrum. Η κύρια τους δουλεία είναι να απομακρύνουν τα εμπόδια πχ ζητήματα του έργου τα οποία μπορεί να καθυστερήσουν ή να σταματήσουν μια δραστηριότητα που προωθεί μπροστά το έργο.

Η Ομάδα του Έργου(Project Team) πρέπει να αποτελείται από 5-10 μέλη. Η ομάδα από μόνη της πρέπει να είναι λειτουργική, περικλείοντας τα άτομα από ένα σύνολο κανόνων.

8.7Η Διαδικασία[21]

Οι περισσότερες διαδικασίες έχουν μια λίστα απαιτήσεων(τύπος του συστήματος, σχεδιασμός αντικειμένων, τύπος λειτουργιών, περιβάλλον ανάπτυξης, εξέταση χρηστών κα.) Η Scrum καταγράφει τις απαιτήσεις σε μια Λίστα Προϊόντων(Product Backlog). Οι απαιτήσεις δεν χρειάζεται να είναι ακριβείς ούτε να περιγράφονται πλήρως. Όπως και στα περισσότερα έργα, οι απαιτήσεις προέρχονται από τους μελλοντικούς χρήστες ή την επιχείρηση. Ο κάτοχος του προϊόντος δίνει προτεραιότητα στη Λίστα Προϊόντων: στοιχεία τα οποία είναι σημαντικά για την επιχείρηση/έργο πχ. αυτά τα στοιχεία τα οποία προσθέτουν άμεσες και σημαντικές αξίες για την επιχείρησης είναι καταχωρημένες στην αρχή.

Η ομάδα του έργου είναι υπεύθυνη για να κάνει την πραγματική δουλεία και μετά δημιουργεί μια Ταχεία Λίστα(Sprint Backlog), αυτή περιλαμβάνει στοιχεία από την Λίστα Προϊόντων τα οποία η ομάδα πιστεύει ότι μπορούν να ολοκληρωθούν μέσα σε ένα χρονικό περιθώριο 30 ημερών. Η ομάδα του έργου μπορεί να επικοινωνήσει με τον Κάτοχο του Προϊόντος και με άλλους ώστε να μπορέσει να επεκτείνει τα στοιχεία στην Ταχεία Λίστα. Αφότου περάσουν οι 30 ημέρες, η ομάδα πρέπει να έχει “ένα εκτελέσιμο παραδοτέο προϊόν”.

Ο κάτοχος του Προϊόντος, ο Ειδικός της Scrum και η ομάδα του Έργου θα κάνουν έναν αρχικό έλεγχο των στοιχείων της Λίστας του Προϊόντος ώστε να διαπιστώσουν πόσο καιρό το κάθε στοιχείο θα χρειαστεί για να ολοκληρωθεί. Για αρχή αυτές είναι εκτιμήσεις ή καλύτερα εικασίες. Καθώς ο χρόνος προχωράει, για ένα διάστημα 30 ημερών, η ομάδα θα ξέρει αν αυτές οι εικασίες είναι σωστές και έπεσαν αρκετά κοντά με την πραγματικότητα.

Η Scrum μας επιτρέπει να τελειοποιήσουμε τις εικασίες μας κατά την διάρκεια, εάν πιστεύουμε ότι μια εργασία θα διαρκέσει παραπάνω από ότι προβλέφθηκε, έχουμε την δυνατότητα να το αναγγείλουμε πριν η εργασία αυτή ξεκινήσει. Δουλεύοντας με μικρότερα πακέτα εργασίας(με προκαθορισμένο χρονικό διάστημα των 30 ημερών) οποιοδήποτε θέμα υπάρχει με το πρόγραμμα ή τις απαιτήσεις διευθετείται αμέσως μόλις αναγνωριστεί, όχι πολύ πιο πέρα στη διαδικασία ανάπτυξης με αποτέλεσμα το κόστος για την διόρθωση του να μην είναι υψηλό.

Τι πραγματικά σημαίνει όμως “ένα εκτελέσιμο παραδοτέο προϊόν”(potentially shippable product increment); Αν το θέσουμε απλά κάθε 30 ημέρες η ομάδα πρέπει να προσφέρει κάτι που να έχει αξία στην επιχείρηση, κάτι που να μπορεί η επιχείρηση να το χρησιμοποιήσει ή κάτι που να δείχνει την εξέλιξη που υπάρχει.

Η ομορφιά της προσέγγισης των 30 ημερών είναι: εάν ο Κάτοχος του Προϊόντος, ο πελάτης ή η επιχείρηση τους αρέσει αυτό που βλέπει στο διάστημα των 30 ημερών, μπορούν μετά να επαναπροσδιορίσουν την Λίστα του Προϊόντος. Αυτό είναι σημαντικό διότι σημαίνει ότι η ομάδα ανάπτυξης δουλεύει καλά και παράγει αυτό που χρειάζεται, μην ξεχνάμε ότι μόνο το 20% των χαρακτηριστικών του προϊόντος χρησιμοποιείται συχνά. Με άλλα λόγια το 80% των αξιών ενός προϊόντος προέρχεται από το 20% των χαρακτηριστικών του.

8.8Τεχνάσματα της Scrum(Scrum Artifacts)[21]

Η Scrum έχει αξιοσημείωτες τεχνικές. Υπάρχουν 3 τεχνικές που κάθε μια από αυτές μπορεί να διαχειριστεί χρησιμοποιώντας τίποτα παραπάνω από ένα φύλλο εργασίας Excel. Υπάρχουν περισσότερο προηγμένα/περίπλοκα εργαλεία, από αυτά πολλά είναι ακριβά, άλλα βασισμένα στο δίκτυο ή είναι ακόμα υπό κατασκευή. Τα δικτυακά εργαλεία είναι πολύ καλά, αλλά δεν είναι το ίδιο χρήσιμα όταν είναι σε εκτός δικτύου λειτουργία.

Η γραφική ανατροφοδότηση η οποία μπορεί να εκτυπωθεί ή να παρουσιαστεί σε δημόσιο μέρος κάνει την αναφορά της προόδου πολύ φανερή. Η Scrum προωθεί αυτού του είδους την αναφορά και προσφέρει καταγραφεί διαγραμμάτων σαν ένα μέσο γραφικής αναπαράστασης της προόδου του έργου.

Με την φράση που αναφέρθηκε προηγουμένως ότι οι τεχνικές της Scrum μπορούν να διαχειριστούν χρησιμοποιώντας τίποτα παραπάνω από ένα φύλλο εργασίας Excel εννοούσαμε ότι η Λίστα του Προϊόντος μπορεί να παρουσιαστεί χρησιμοποιώντας ένα φύλλο εργασία του Excel. Και επιπλέον κάθε Ταχεία Λίστα χρησιμοποιεί ένα άλλα φύλλο αλλά μέσα στο ίδιο αρχείο.

Ένα παράδειγμα μιας τέτοιας Λίστας Προϊόντος είναι η ακόλουθη.

Product Backlog
Estimating System Upgrade

Sprint	ID		Backlog Item	Owner	Estimate (days)	Remaining (days)
1	1	Minor	Remove user kludge in .dpr file	BC	1	1
1	2	Minor	Remove cMap/cMenu/cMenuSize from disciplines.pas	BC	1	1
1	3	Minor	Create "Legacy" discipline node with old civils and E&I content	BC	1	1
1	4	Major	Augment each tbl operation to support network operation	BC	10	10
1	5	Major	Extend Engineering Design estimate items to include summaries	BC	2	2
1	6	Super	Supervision/Guidance	CAM	4	4
	7	Minor	Remove Custodian property from AppConfig class in globals.pas	BC	1	
	8	Minor	Remove LOC_ constants in globals.pas and main.pas	BC	1	
	9	Minor	New E&I section doesn't have lblCaption set	BC	1	
	10	Minor	Delay in main.releaseform doesn't appear to be required	BC	1	
	11	Minor	Undo modifications to Other Major Equipment in formExcel.pas	BC	1	
	12	Minor	AJACS form to be centred on the screen	BC	1	
	13	Major	Extend DUnit tests to all 40 disciplines	BC	6	

Σχήμα 9 – Δείγμα Λίστα Προϊόντος

Χρησιμοποιώντας τις πληροφορίες από το Σχήμα 9 μπορούμε να φτιάξουμε ένα άλλο φύλλο εργασίας με την Ταχεία Λίστα, που περιέχει μια λίστα από πράγματα τα οποία θα γίνουν. Κάθε αντικείμενο από την Ταχεία Λίστα έχει μια χρονική πρόβλεψη για το πόσο θα χρειαστεί ώστε να ολοκληρωθεί, συνήθως μετρημένη σε ώρες. Και βλέποντας λίγο το παράδειγμα θα διαπιστώσουμε ότι καμία από τις 6 αυτές εργασίες δεν έχει ξεκινήσει.

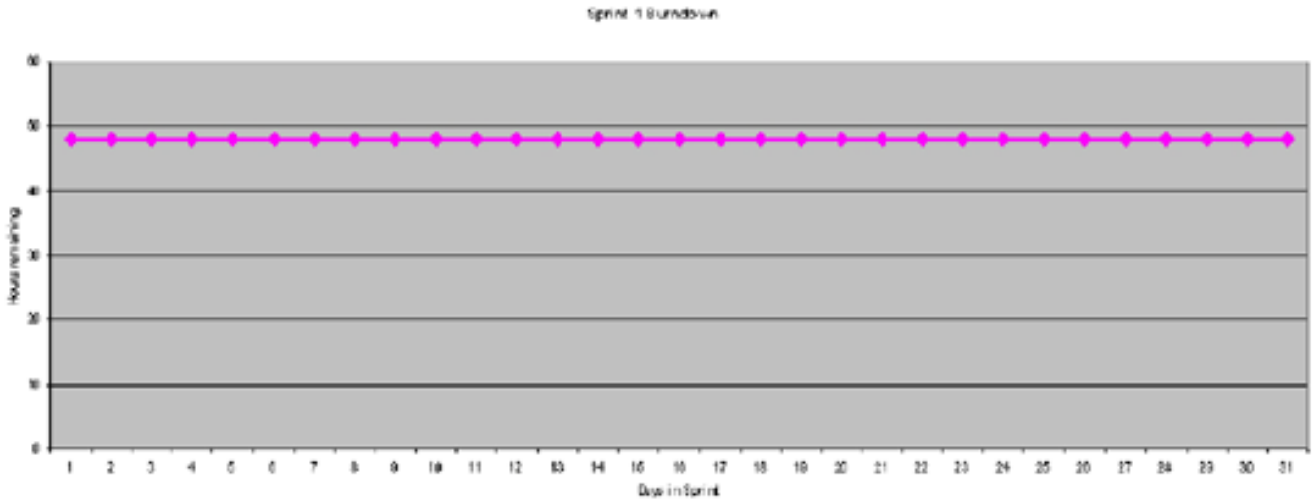
Sprint 1

01/11/2004				Sprint Day	1 Mo	2 Tu	3 We	4 Th	5 Fr	6 Sa	7 Su
	19 days work in this sprint			Hours remaining	152	152	152	152	152	152	152
	Backlog Item	Backlog Item	Owner	Estimate							
1	Minor	Remove user kludge in .dpr file	BC	8	8	8	8	8	8	8	8
2	Minor	Remove cMap/cMenu/cMenuSize from disciplines.pas	BC	8	8	8	8	8	8	8	8
3	Minor	Create "Legacy" discipline node with old civils and E&I content	BC	8	8	8	8	8	8	8	8
4	Major	Augment each tbl operation to support network operation	BC	80	80	80	80	80	80	80	80
5	Major	Extend Engineering Design estimate items to include summaries	BC	16	16	16	16	16	16	16	16
6	Super	Supervision/Guidance	CAM	32	32	32	32	32	32	32	32

Σχήμα 10 – Δείγμα Ταχείας Λίστας

Το κλειδί είναι να κρατιέται η Ταχεία Λίστα ενημερωμένη, και αυτό όχι μόνο μας επιτρέπει να διαπιστώσουμε πόσο γρήγορα μια ομάδα μπορεί να δουλέψει αλλά είναι και ένας δείκτης πρόγνωσης προβλημάτων.

Αυτό που είναι πολύ χρήσιμο στην Ταχεία Λίστα είναι η ικανότητα της να παρουσιάζετε γραφικά. Η Scrum χρησιμοποιεί την καταγραφή διαγραμμάτων για να παρουσιάσει την δουλεία που έχει γίνει. Στο σχήμα 11 παρουσιάζετε ένα διάγραμμα όπου καμία δουλειά δεν έχει γίνει από την Ταχεία Λίστα. Η ιδέα πίσω από την καταγραφή των διαγραμμάτων είναι ότι πρέπει να παρουσιάσει μια σταθερή κίνηση όταν ο χρόνος τελειώσει, παρουσιάζει ένα βήμα της εργασία το οποίο είναι λειτουργικό. Στην πραγματικότητα ωστόσο κάποιες εργασίες διαρκούν περισσότερο από άλλες και κάποιες είναι ακόμα πιο σύντομες έτσι το καταγραμμένο διάγραμμα μπορεί να μην είναι μια ευθεία γραμμή.



Σχήμα 11 –Ταχεία Λίστα, Καμία εργασία δεν πραγματοποιήθηκε

Το σχήμα 12 παρουσιάζει μια Ταχεία Λίστα η οποία έχει ενημερωθεί μετά από εργασίες οι οποίες έχουν πραγματοποιηθεί. Ακλουθώντας την "Remove user kludge" στο αρχείο βλέπουμε ότι πραγματοποιήθηκαν 4 ώρες εργασίας την Τετάρτη στις 3 και συνεχίστηκε με 2 ώρες εργασίας την Πέμπτη στις 4 και η συνολική εργασία ολοκληρώθηκε την Παρασκευή σπαταλώντας 2 ώρες.

Θα παρατηρήσετε επίσης ότι την ίδια ημέρα(Τετάρτη), το άτομο που ασχολήθηκε αφιέρωσε 4 ώρες στην "Remove cMap/cMenu…". Άρα υποθέτουμε ότι σε μια ημέρα 8ωρης εργασίας το άτομο χώρισε το χρόνο για να ασχοληθεί με 2 εργασίες.

Sprint 1

01/11/2004

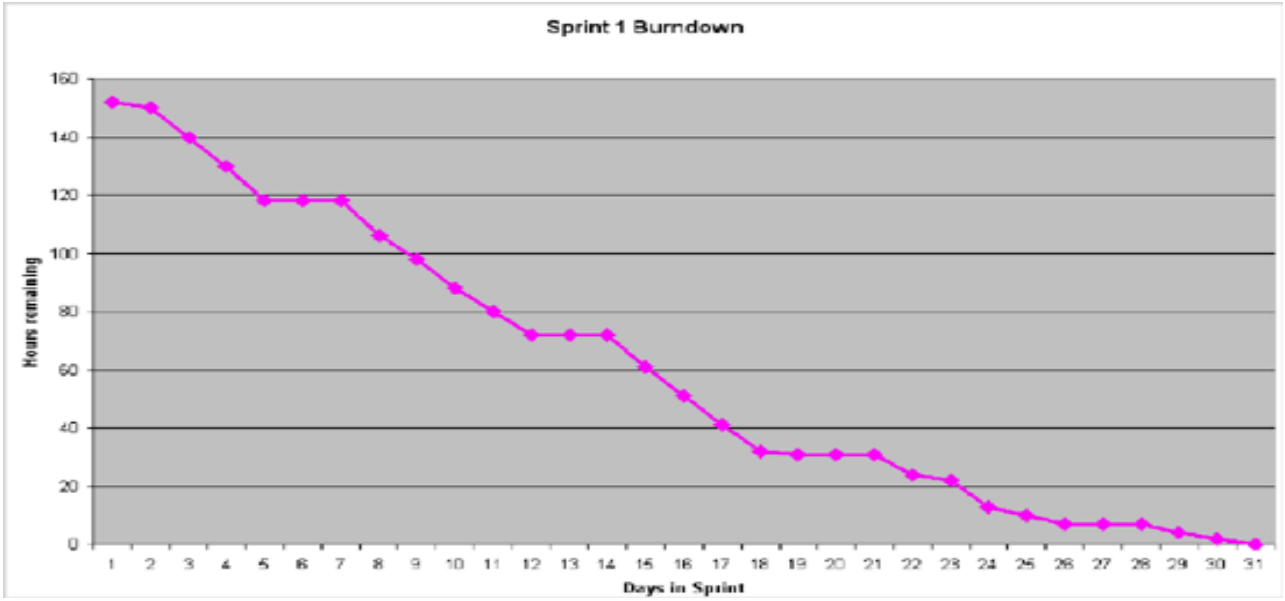
19 days work in this sprint

	Backlog Item	Backlog Item	Owner	Estimate	Sprint Day	1 Mo	2 Tu	3 We	4 Th	5 Fr	6 Sa	7 Su
					Hours remaining	152	160	140	130	118	118	118
1	Minor	Remove user kludge in .dpr file	BC	8		8	8	4	2	0		
2	Minor	Remove cMap/cMenu/cMenuSize from disciplines pas	BC	8		8	8	4	0			
3	Minor	Create "Legacy" discipline node with old civils and E&I content	BC	8		8	8	8	6	0		
4	Major	Augment each tol operation to support network operation	BC	80		80	80	80	80	78	78	78
5	Major	Extend Engineering Design estimate items to include summaries	BC	16		16	16	16	16	16	16	16
6	Super	Supervision/Guidance	CAM	32		32	30	28	26	24	24	24

Σχήμα 12 – Ταχεία Λίστα μετά από εργασία που έχει γίνει

Η ομορφιά της Ταχείας Λίστας είναι η απλότητα της. Με την καταγραφή των ωρών που πραγματικά σπαταλήθηκαν για να ολοκληρωθεί μια εργασία σε σχέση με τις υπολογιζόμενες ώρες για την ολοκλήρωση της, είμαστε ικανοί να σχεδιάσουμε εξίσου απλά αλλά δυναμικά διαγράμματα τα οποία παρέχουν με εμπιστοσύνη την πρόοδο η οποία μέχρι στιγμής έχει γίνει. Εάν τα διαγράμματα δεν δείχνουν την πρόοδο που έχει γίνει, τότε μας παρέχει μια άλλη υπηρεσία, μας δίνει μια έγκυρη προειδοποίηση ότι αυτή η ροή περιέχει ή πάρα πολύ δουλεία ή πολύ λίγη.

Στο σχήμα 13 παρουσιάζετε ένα διάγραμμα που βασίζεται στην Ταχεία Λίστα του γραφικού 9. Το ιδανικό είναι το διάγραμμα να δείχνει την ταχύτητα των επιπέδων πχ η εργασία πραγματοποιείται με σταθερό ρυθμό. Στην πραγματικότητα χρειάζεται να υπολογίσουμε τα Σαββατοκύριακα και άλλες μη εργάσιμες ημέρες της εβδομάδας. Όπως θα δούμε εάν η ομάδα του έργου ή τα άτομα απασχοληθούν από κάτι άλλο για κάποιες ώρες, η καταγραφή των διαγραμμάτων είναι ένας καλός τρόπος να ανιχνευτούν αυτές οι αποσπάσεις της προσοχής πολύ νωρίς μέσα στο έργο.

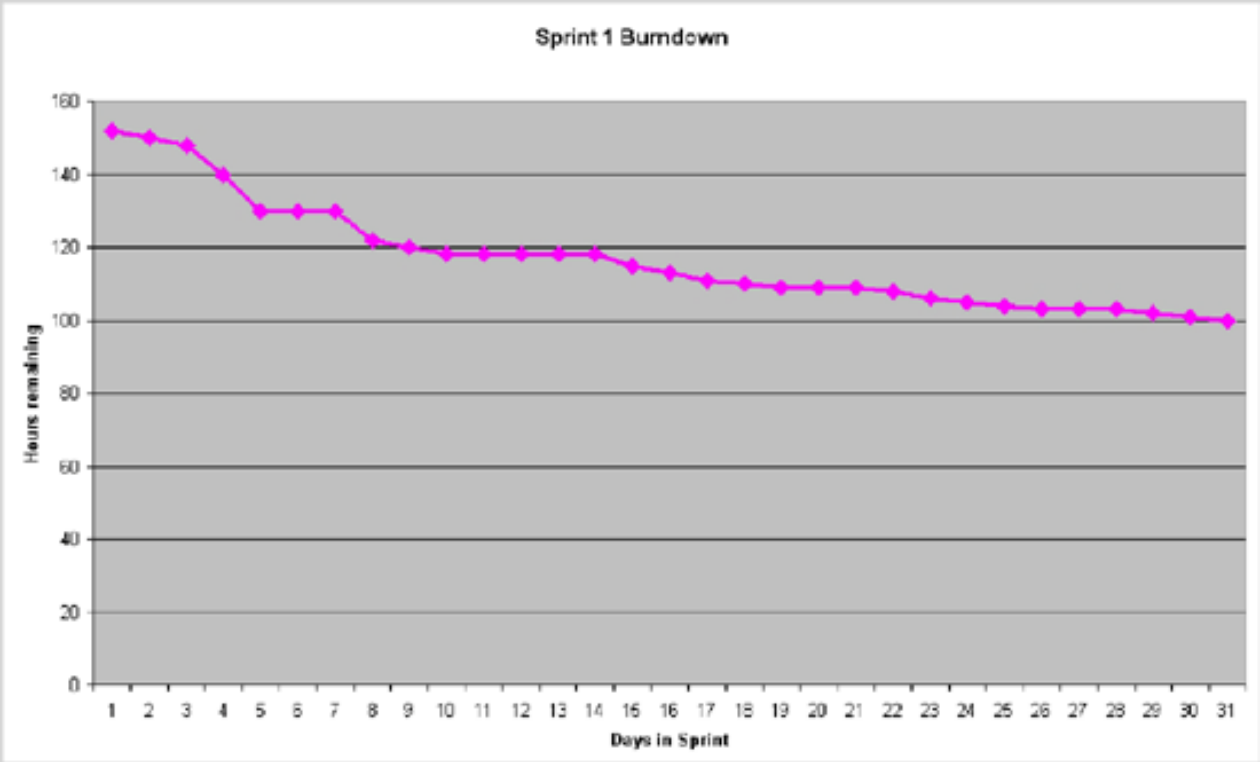


Σχήμα 13-Καταγραφή διαγράμματος μετά από εργασία που έχει πραγματοποιηθεί

Το σχήμα 14 παρουσιάζει ένα διάγραμμα το οποίο επιδεικνύει ότι η πρόοδος έχει γίνει, ωστόσο δεν έχει γίνει τόσο γρήγορα όσο θα έπρεπε. Το διάγραμμα αυτό μας δίνει ένα στοιχείο ότι κάτι δεν γίνεται σωστά στην $2^{η}$ προς $3^{η}$ ημέρα από την αρχή και επιβεβαιώνει ότι η εργασία δεν προχωράει σε καλό ρυθμό από την 7η έως την $14^{η}$ ημέρα.

Υπάρχουν κάποιοι λόγοι γιατί το διάγραμμα αυτό εμφανίζετε έτσι.

1. Η ομάδα ανάπτυξης ή κάποιο άτομο αποσπάται από την εργασία του.
2. Η Ταχεία Λίστα δεν ανανεώθηκε.
3. Τα στοιχεία στην Ταχεία Λίστα είναι πολύ πολύπλοκα.

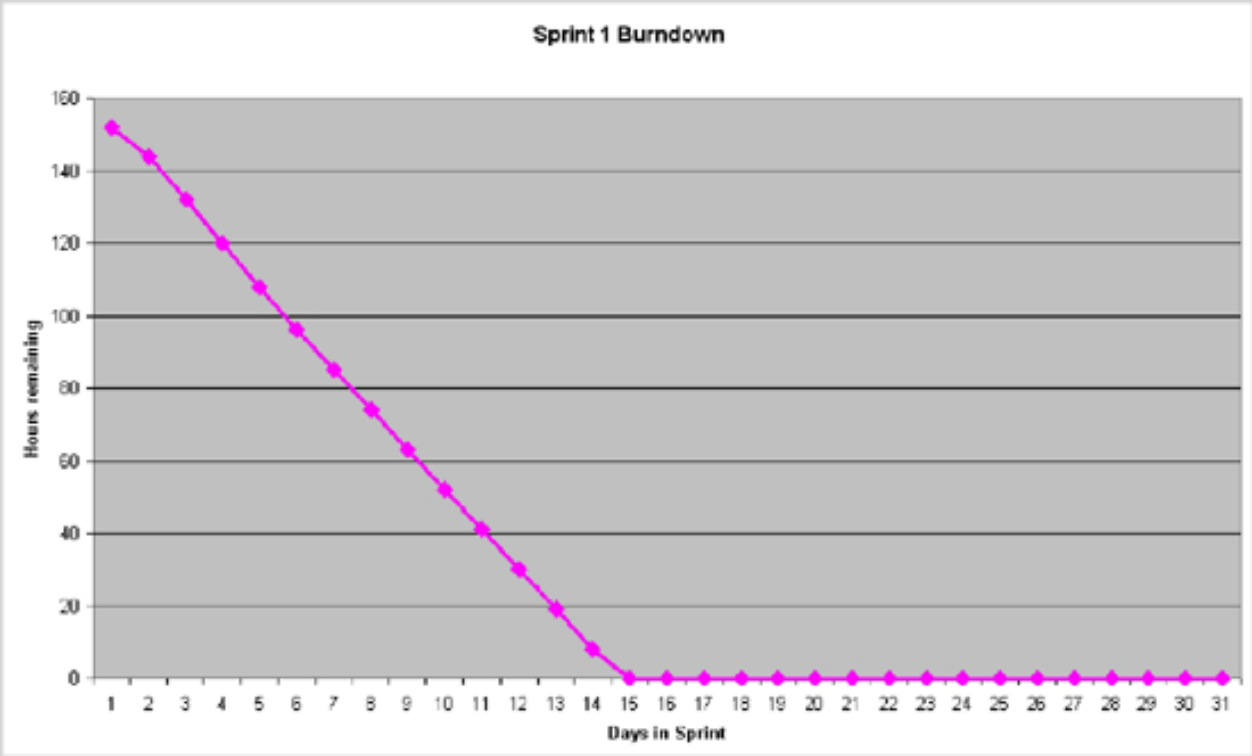


Σχήμα 14 – Διάγραμμα μετά από εργασία που έχει πραγματοποιηθεί αλλά όχι αρκετά γρήγορα

Εννοείται ότι τα διαγράμματα μπορούν να αποκαλύψουν ότι η Ταχεία Λίστα έχει τελειώσει νωρίτερα. Γα παράδειγμα, εάν μια ροή δεν περιέχει πολύ δουλειά μπορεί να τελειώσει με ένα διάγραμμα σαν αυτό του σχήματος 15. Υπάρχουν όμως και άλλοι λόγοι για τους οποίους μπορεί μια Ταχεία Λίστα να τελειώσει νωρίτερα:

Υπέρμετρη εργασία: εάν η ομάδα ανάπτυξης δουλέψει περισσότερο από 8 ώρες την ημέρα(σε αυτή την περίπτωση) η εργασία θα τελειώσει πριν από τον καθορισμένο προγραμματισμό. Στην πραγματικότητα η υπέρμετρη εργασία παρουσιάζεται μόνο για να κερδηθεί μικρό χρονικό διάστημα διότι διαφορετικά το έργο θα χάσει από ποιότητα ή/και από την κούραση θα μειωθεί η παραγωγικότητα της ομάδας.

Στην Ταχεία Λίστα τα αντικείμενα μπορεί να εκτιμηθούν λάθος: μπορεί να χρειαστεί να επιστρέψουμε στην αρχική μας εκτίμηση καθώς η εργασία ολοκληρώνεται σύμφωνα με τις υπάρχουσες εκτιμήσεις. Αυτή του είδους η εξαίρεση μπορεί να γίνει εάν η ομάδα ανάπτυξης ξέρει ότι πρέπει να ανακοινώσει τα γεγονότα που οδήγησαν στην ολοκλήρωση μιας εργασίας πριν από το προβλεπόμενο χρόνο, στην Ταχεία Λίστα απλά χρειάζεται να σημειωθεί μια εργασία ότι έχει 0 ώρες υπόλοιπο για να δειχθεί ότι έχει τελειώσει.



Σχήμα 15 – Διάγραμμα μετά από εργασία που έχει πραγματοποιηθεί, αλλά πολύ γρήγορα

Στην πραγματικότητα παρατηρούμε ότι τα διαγράμματα έχουν τις διακυμάνσεις τους αλλά παρόλα αυτά εκπληρώνουν τους στόχους τους. Εάν οι διακυμάνσεις τους είναι δραματικές, τότε κίνδυνος θα σήμαινε ότι κάτι είναι λάθος είτε με τους στόχους της ροής είτε με την ομάδα ανάπτυξης. Όπως και να έχει θα έχουμε μια έγκαιρη προειδοποίηση ότι αυτή η ροή(διάρκειας λιγότερο από 30 ημέρες) πρόκειται να αντιμετωπίσει πρόβλημα. Είναι καλύτερα να αντιμετωπίζουμε και να λύνουμε θέματα του έργου όταν αυτά εμφανίζονται, Τα διαγράμματα της Scrum βάζουν τα προβλήματα του έργου σε ένα πλαίσιο, είναι καλύτερα να χάσει κάποιος 30 μέρες νωρίτερα στο έργο παρά να ψάχνει να βρει χρόνο αργότερα ώστε να διορθώσει τα λάθη που προέκυψαν κατά την διάρκεια.

Οι Ταχείες Λίστες και τα Διαγράμματα είναι έγκαιροι δείκτες προειδοποίησης, δίνουν έμφαση στην έλλειψη προόδου. Ομοίως δίνουν έμφαση σε σενάρια τα οποία δεν έχουν πολύ δουλειά ή η εργασία είναι εύκολη. Ωστόσο η χρησιμότητα τους υποθέτει ότι όλοι είναι δεσμευμένοι στην τήρηση της ενημέρωσης της Ταχείας Λίστας.

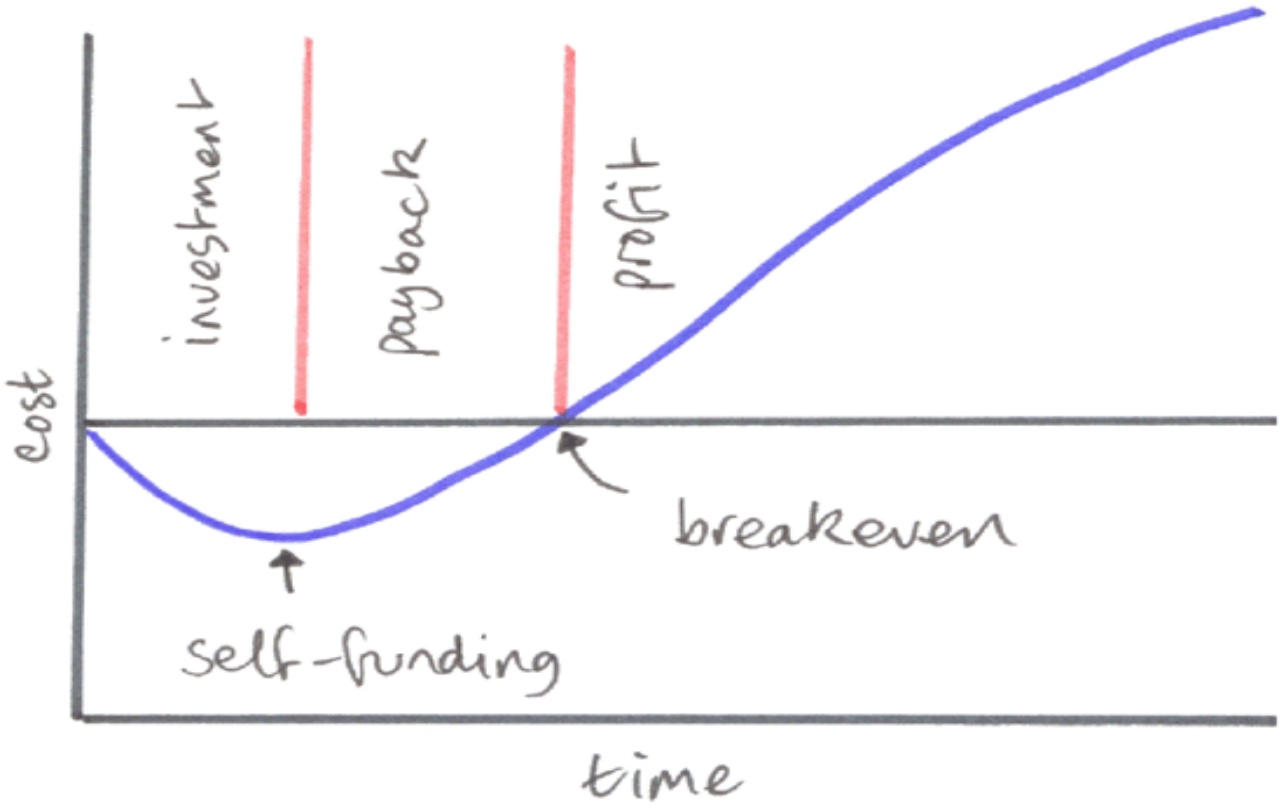
Με προσεκτική χρήση ακόμα και αυτά τα απλά λογιστικά φύλλα μπορούν να βοηθήσουν σε ένα βαθμό σε άλλους τομείς. Για παράδειγμα είναι δυνατό να εξάγουμε μεμονωμένα διαγράμματα για κάθε μέλος της ομάδας ανάπτυξης. Ενώ ένα γενικό διάγραμμα μπορεί να επισημάνει ένα γενικό πρόβλημα με το έργο, χρησιμοποιώντας τα μεμονωμένα διαγράμματα των μελών της ομάδας μπορούμε να εντοπίσουμε αυτό το ένα μέλος που δημιουργεί το πρόβλημα.

Ομοίως αναθέτοντας τα αντικείμενα της Ταχείας Λίστας στα μέλη της ομάδας μπορούμε να υπολογίσουμε την ώρα εργασίας τους, ιδανικά κανένας υπάλληλος δεν πρέπει να δουλεύει περισσότερο από έναν δεδομένο αριθμό ωρών την ημέρα(πχ 8 ώρες).

8.9Γιατί η επαναληπτική ανάπτυξη δουλεύει

Το σχήμα 16 παρουσιάζει ένα ιδανικό κέρδος και την καμπύλη απώλειας βάση του κόστους και του χρόνου. Στην αρχή των περισσότερων έργων, υπάρχει ένα αρχικό χτύπημα, ένα αρχικό κόστος και αυτή είναι περίοδος κάτω από την γραμμή.

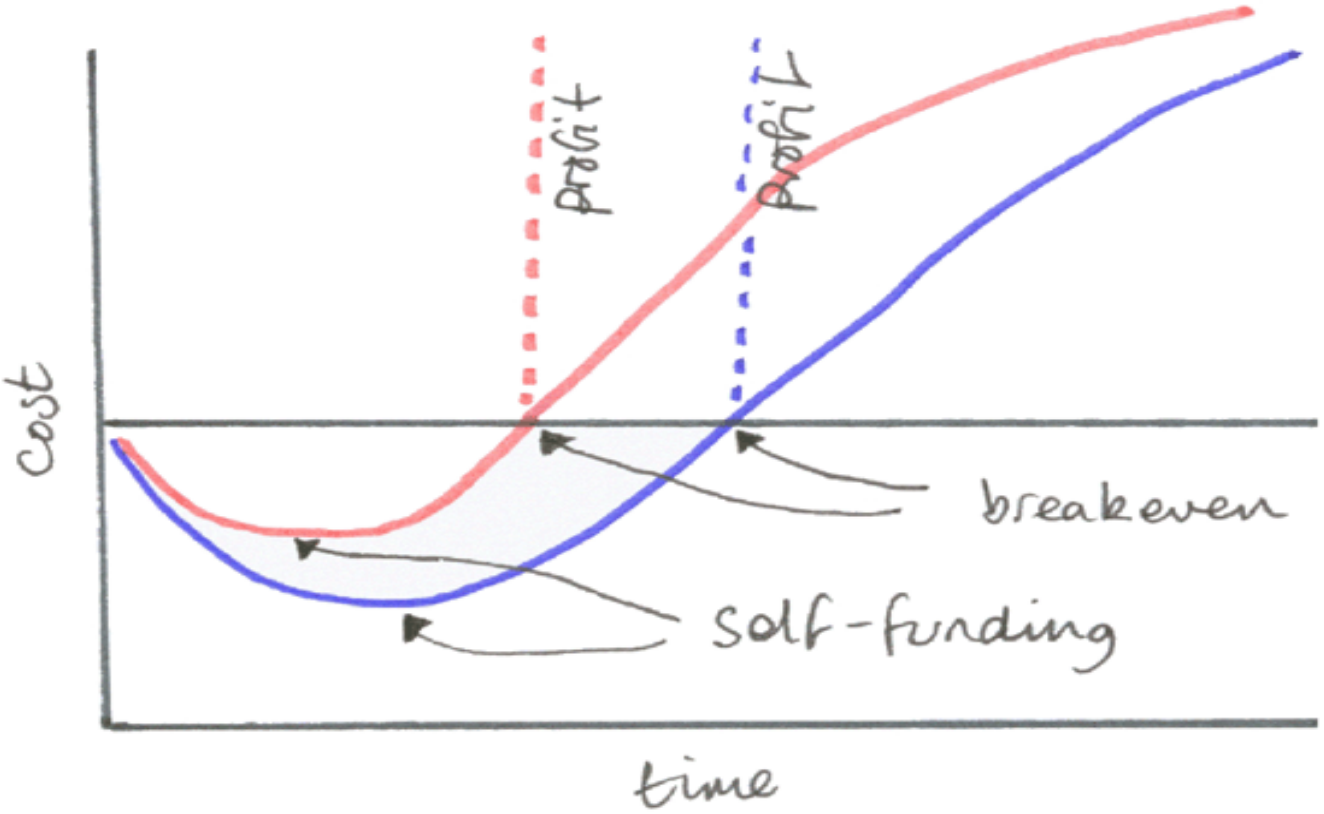
Αφού περάσει κάποιο χρονικό διάστημα το έργο αρχίζει να εμφανίζει κάποια κέρδη και αυτή είναι η περίοδος πάνω από την γραμμή



Σχήμα 16 – Κέρδος και απώλειες χρησιμοποιώντας τον παραδοσιακό τρόπο ανάπτυξης

Το κύριο θέμα είναι να φτάσεις το έργο στο κέρδος το συντομότερο δυνατό. Προφανώς υπάρχει ένας αριθμός από κόλπα που μπορούμε να χρησιμοποιήσουμε για να το πετύχουμε αυτό, κάποια από τα οποία συμπεριλαμβάνουν την μείωση της ποιότητας, μείωση του πεδίου κα. Προσοχή στη χρήση τέτοιων τεχνικών, η πρόωρη προσπάθεια για την επίτευξη του κέρδους μπορεί να αποτελέσει προσωρινή λύση που μπορεί να επηρεάσει το μελλοντικό κέρδος. Η φτωχή ποιότητα(του προϊόντος, της υποστήριξης κα.) μπορεί να οδηγήσει στην μείωση των μελλοντικών πωλήσεων.

Η επαναληπτική ανάπτυξη είναι μια τεχνική που επιτρέπει την ελεγχόμενη προσπάθεια για την επίτευξη του σημείου του κέρδους, εάν μπορούμε να δώσουμε μια έκδοση ενός προϊόντος σε επαναληπτική και επαυξητική βάση, μπορούμε να καταλάβουμε τα πλεονεκτήματα της πιο γρήγορης παράδοσης μιας έκδοσης πιο νωρίς από το προϊόν. Το σχήμα 17 μας παρουσιάζει ένα τυπικό κέρδος και την καμπύλη απώλειας στην οποία η επαναληπτική ανάπτυξη έχει εφαρμοσθεί.



Σχήμα 17 – Κέρδος και απώλειες χρησιμοποιώντας την επαναληπτική ανάπτυξη

Η επαναληπτική ανάπτυξη σημαίνει ότι είμαστε ικανοί να αναπτύξουμε την εφαρμογή νωρίτερα απ' ότι πριν. Όσο πιο νωρίς μπορέσουμε να αναπτύξουμε την εφαρμογή τόσο πιο σύντομα αποκομίσουμε τα κέρδη.

Η Scrum προάγει την επαναληπτική ανάπτυξη: κάθε 30 ημέρες έχουμε ένα πιθανό παραδοτέο επαυξητικό προϊόν (ή εκτελέσιμο). Εάν μπορέσουμε να μεταφέρουμε τις περισσότερες λειτουργίες σε μια πρόωρη έκδοση όχι μόνο οι πελάτες μας θα αποκτήσουν νωρίτερα πρόσβαση αλλά θα είναι και σε καλύτερη θέση ώστε να επιβεβαιώσουν ότι πήραν αυτό που ζητήθηκε.

9. Διασφάλιση ποιότητας λογισμικού

9.1 Εισαγωγή

Από τότε που ο Kent Beck(Αμερικάνος Μηχανικός Λογισμικού και κατασκευαστής των μεθοδολογιών της Extreme Programming και της Test Driven Ανάπτυξης Λογισμικού)[10] μας σύστησε την μέθοδο Extreme Programming, η ευέλικτη ανάπτυξη λογισμικού έχει γίνει το επίμαχο θέμα της μηχανικής λογισμικού. Κάποιοι επαγγελματίες και ερευνητές διαφωνούν έντονα για τα πλεονεκτήματα της, άλλοι είναι κάθετα αντίθετοι με τις ευέλικτες μεθόδους ενώ άλλοι προτείνουν μια μίξη των ευέλικτων και καθοδηγούμενων από σχέδιο πρακτικών. Ωστόσο, η πραγματικότητα είναι ότι οι ευέλικτες μέθοδοι έχουν κερδίσει τρομερή αποδοχή στην περιοχή του εμπορίου από τη δεκαετία του ’90 διότι αυτές μπορούσαν να προσαρμόσουν ευμετάβλητες απαιτήσεις, επικεντρώθηκαν στην συνεργασία κατασκευαστών και πελατών και υποστηρίζουν την πρώιμη παράδοση του προϊόντος.

Δυο από τα πιο σημαντικά χαρακτηριστικά της ευέλικτης προσέγγισης είναι 1) μπορούν να χειριστούν ασταθές απαιτήσεις κατά την διάρκεια του κύκλου ζωής της ανάπτυξης και 2) μπορούν να παραδώσουν προϊόντα σε μικρότερο χρονικό περιθώριο και κάτω από το προϋπολογισμό όταν συγκρίνονται με τις παραδοσιακές μεθόδους. Πολλές δημοσιευμένες αναφορές υποστηρίζουν τα παραπάνω πλεονεκτήματα των ευέλικτων μεθοδολογιών. Ωστόσο, οι υποστηρικτές των ευέλικτων μεθοδολογιών δεν έχουν παρουσιάσει ακόμα μια πιστική απάντηση στην ερώτηση “Ποια είναι η ποιότητα των προϊόντων που παράγονται”. Παρέχει αρκετή αυστηρότητα ώστε να διασφαλίσει την ποιότητα, όπως κάνουν οι παραδοσιακές μέθοδοι ανάπτυξης πχ το μοντέλο καταρράκτη, και αν αυτές οι μέθοδοι προσφέρουν το ίδιο επίπεδο ποιότητας πως αυτό κατορθώνεται;

Για να μπορέσουμε να καταλάβουμε και να δώσουμε πιθανές απαντήσεις σε αυτά τα ερωτήματα θα συγκρίνουμε τις τεχνικές διασφάλισης της ποιότητας στις ευέλικτες και παραδοσιακές διαδικασίες ανάπτυξης λογισμικού. Η προσέγγιση μας αποτελείται από 3 βήματα: 1) Κατασκευή μιας πλήρης περίληψης του παραδοσιακού μοντέλου καταρράκτη συμπεριλαμβανομένου και των διαδικασιών υποστήριξης, 2) Προσδιορισμός αυτών των πρακτικών μέσα στις ευέλικτες μεθοδολογίες που έχουν σαν νόημα την διασφάλιση της ποιότητας του λογισμικού σε σύγκριση με τις τεχνικές διασφάλισης λογισμικού που βρίσκονται στις παραδοσιακές μεθόδους, 3) καθορισμός των ομοιοτήτων και των διαφορών μεταξύ ευέλικτων και παραδοσιακών τεχνικών διασφάλισης ποιότητας λογισμικού. Με την εφαρμογή μιας τέτοιας προσέγγισης, πιστεύουμε ότι θα μπορέσουμε συστηματικά να ερευνήσουμε πώς οι ευκίνητες μέθοδοι ενσωματώνουν την υποστήριξη της ποιότητας λογισμικού μέσα στον κύκλο ζωής.

9.2 Μοντέλο Καταρράκτη εναντίων Ευέλικτων μεθόδων

Από την αρχή της δεκαετίας του 60, διαφορετικές μέθοδοι ανάπτυξης λογισμικού(όπως το μοντέλο καταρράκτη, η εξελικτική μέθοδο ανάπτυξης, το ελλειπτικό μοντέλο ανάπτυξης κ.α.) έχουν αναπτυχθεί και ευρέως χρησιμοποιηθεί από την κοινότητα της μηχανικής λογισμικού. Μέσα στα χρόνια οι κατασκευαστές και οι χρήστες αυτών των μεθόδων έχουν επενδύσει σημαντικά ποσά χρόνου και ενέργειας για να τις αναβαθμίσουν και να τις βελτιώσουν. Αποτέλεσμα της συνεχής αυτής προσπάθειας βελτίωσης και χρήσης τους για τόσο μεγάλο χρονικό διάστημα, οι περισσότερες από τις προαναφερθείσες μέθοδοι έφτασαν σε ένα πολύ καλό ώριμο και σταθερό επίπεδο. Για αυτό το λόγο συνήθως αναφέρονται σαν παραδοσιακές μέθοδοι ανάπτυξης λογισμικού.

Κάθε μια από τις παραδοσιακές μεθόδους ανάπτυξης προσπαθούν να διευθετήσουν διαφορετικά θέματα ανάπτυξης και να πραγματοποιήσουν διαφορετικές συνθήκες. Ανάμεσα στις παραδοσιακές προσεγγίσεις ανάπτυξης, το μοντέλο καταρράκτη είναι το πιο παλιό μοντέλο διαδικασίας ανάπτυξης λογισμικού. Το μοντέλο του καταρράκτη έχει ευρέως χρησιμοποιηθεί και σε μεγάλα αλλά και σε μικρά εντατικά έργα λογισμικού. Έχει αναφερθεί ότι είναι μια επιτυχείς προσέγγιση ανάπτυξης ιδίως για μεγάλα και πολύπλοκα έργα μηχανικής. Το μοντέλο του καταρράκτη διαιρεί τον κύκλο ζωής της ανάπτυξης λογισμικού σε πέντε ευδιάκριτα και γραμμικά στάδια. Επειδή το μοντέλο καταρράκτη είναι το πιο παλιό και ώριμο μοντέλο ανάπτυξης λογισμικού θα εξετάσουμε την διαδικασία διασφάλισης λογισμικού του.

Παρά την επιτυχία του μοντέλου καταρράκτη στα μεγάλα και πολύπλοκα συστήματα, έχει έναν αριθμό από μειονεκτήματα, όπως την γραμμικότητα(linearity), έλλειψη ευελιξίας(inflexibility) από την άποψη της αλλαγής απαιτήσεων, ιδιαίτερες επίσημες διαδικασίες άσχετα με την φύση και το μέγεθος του έργου κ.α. Τέτοια μειονεκτήματα μπορούν επίσης να βρεθούν και σε άλλες παραδοσιακές προσεγγίσεις. Ωστόσο, οι ευέλικτες μέθοδοι αναπτύχθηκαν για να διευθετήσουν έναν αριθμό από τα μειονεκτήματα που εμπεριέχονται στο μοντέλο του καταρράκτη.

Οι ευέλικτες μέθοδοι χειρίζονται τέτοιες ασταθείς και ευμετάβλητες απαιτήσεις χρησιμοποιώντας έναν αριθμό από τεχνικές από τις οποίες οι πιο σημαντικές είναι 1) απλός σχεδιασμός(simply planning), 2) σύντομη επανάληψη(short iteration), 3)πρώιμες εκδόσεις(earlier release) και 4) συχνές ανατροφοδοτήσεις από τον πελάτη. Αυτά τα χαρακτηριστικά επέτρεψαν στις ευέλικτες μεθόδους να παραδίδουν εκδόσεις προϊόντων σε μικρότερη χρονική περίοδο συγκρινόμενη με την προσέγγιση του καταρράκτη.

Αυτή η σύντομη σύγκριση της μεθόδου του καταρράκτη με αυτή των ευέλικτων μεθόδων μας έφεραν πάλι στην αρχική μας ερώτηση, πως μπορούν οι ευέλικτες μέθοδοι να διασφαλίσουν την ποιότητα του προϊόντος σε τόσο σύντομο χρονικό διάστημα. **Η ερευνητική υπόθεση μας είναι ότι στις ευέλικτες μεθόδους, μέχρι κάποιο βαθμό, μερικές από τις πρακτικές τους περιλαμβάνουν παραδοσιακές διαδικασίες υποστήριξης της διασφάλισης της ποιότητας μέσα στον κύκλο ζωής τους.**

Πριν προχωρήσουμε καλό θα ήταν να αναλύσουμε τις διάφορες τεχνικές διασφάλισης της ποιότητας, μια γενική περιγραφή αυτών των τεχνικών και τις σχετιζόμενες με αυτές διαδικασίες υποστήριξης.

9.3 Τεχνικές διασφάλισης ποιότητας

Από τότε που μας απασχόλησε η ποιότητα του παραγόμενου λογισμικού με το μοντέλο του καταρράκτη και με τις ευέλικτες προσεγγίσεις, εξετάζουμε τις υποστηρικτικές διαδικασίες στην ανάπτυξη του λογισμικού που σχετίζονται με την ποιότητα. Επικεντρωνόμαστε σε 2 από τις πιο συχνά χρησιμοποιούμενες γενικές διαδικασίες που εστιάζουν στην Ποιότητα, την Διασφάλιση της Ποιότητας του Λογισμικού(Software Quality Assurance SQA) και την Επαλήθευση και Επικύρωση (Verification and Validation V&V) για να εξετάσουμε την ποιότητα του λογισμικού.

Η SQA καθοδηγεί τις διαδικασίες που πρόκειται να χτίσουν την επιθυμητή ποιότητα στο προϊόν και η V&V στοχεύει περισσότερο απευθείας στην ποιότητα που περιλαμβάνεται στα ενδιάμεσα προϊόντα. Αυτές οι δυο υποστηρικτικές διαδικασίες χρησιμοποιούνται κανονικά για να υποστηρίξουν το μοντέλο του καταρράκτη ώστε να προσφέρει ένα πλήρως τελειοποιημένο μοντέλο.

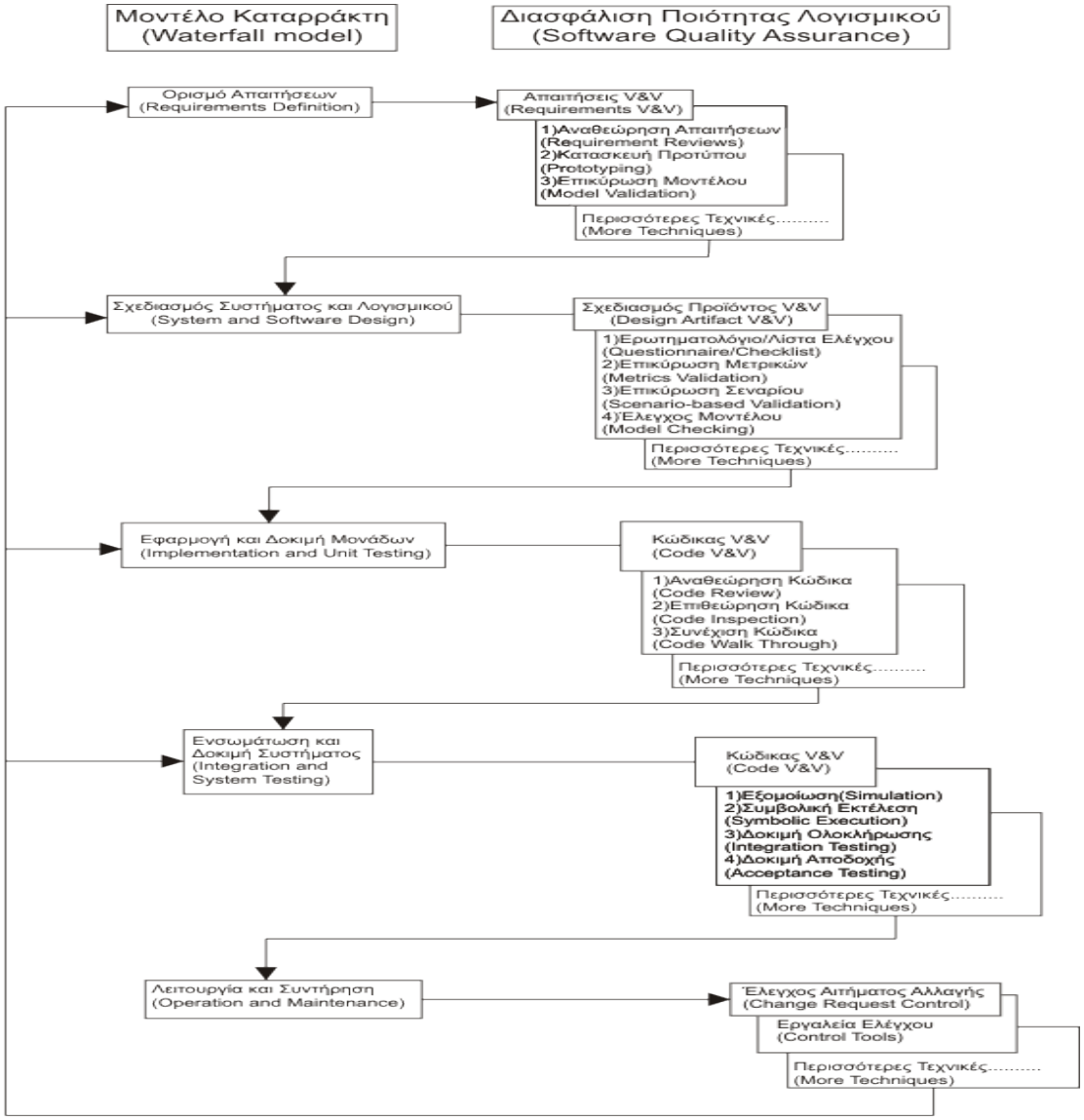
Οι τεχνικές διασφάλισης της ποιότητας μπορούν να κατηγοριοποιηθούν σε 2 τύπους, στατικές και δυναμικές. Και οι δυο μπορούν να χρησιμοποιηθούν στην SQA διαδικασία. Η επιλογή, οι στόχοι και η οργάνωση μιας συγκεκριμένης τεχνικής εξαρτάται από τις απαιτήσεις και την φύση του έργου. Η μέθοδος ανάπτυξης του καταρράκτη επιλέγει αυτές τις τεχνικές σύμφωνα με πολύ διαφορετικά κριτήρια.

Σε αντίθεση με τις δυναμικές τεχνικές, οι στατικές τεχνικές δεν εμπλέκονται με τον εκτελέσιμο κώδικα. Οι στατικές τεχνικές εμπλέκονται με την εξέταση των εγγράφων από μεμονωμένα άτομα ή από ομάδες, αυτή η εξέταση μπορεί να βοηθηθεί από λογισμικά-εργαλεία, όπως για παράδειγμα η εξέταση της προδιαγραφής των απαιτήσεων και η τεχνική επιθεώρηση του κώδικα. Οι δοκιμές και οι προσομοιώσεις είναι δυναμικές τεχνικές. Μερικές φορές οι στατικές τεχνικές χρησιμοποιούνται για να υποστηρίξουν τις δυναμικές τεχνικές και το αντίθετο.

Το μοντέλο του καταρράκτη χρησιμοποιεί και τις δυο τεχνικές. Ωστόσο οι ευέλικτες μέθοδοι περισσότερο χρησιμοποιούν τις δυναμικές τεχνικές.

9.4 Το μοντέλο του καταρράκτη με SQA και V&V[12]

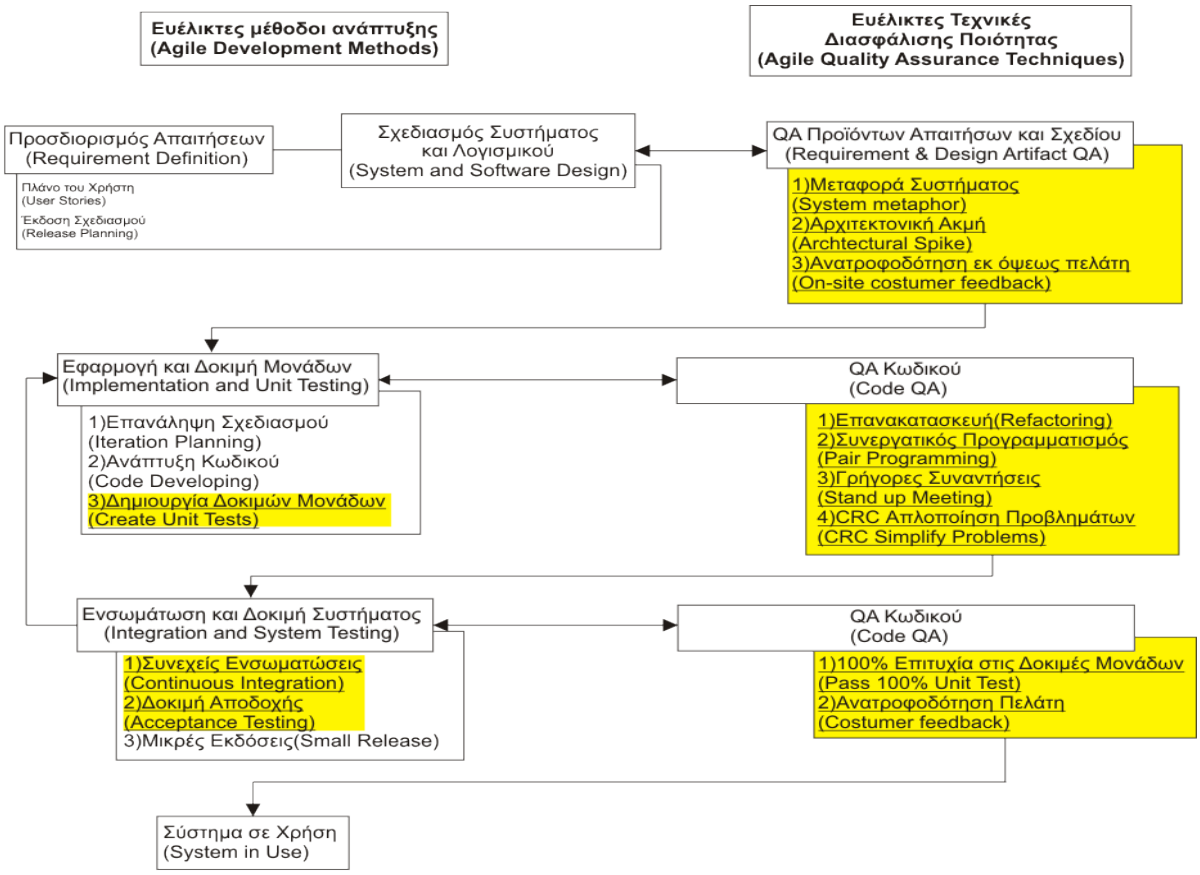
Οι θεμελιώδης δραστηριότητες ανάπτυξης στο μοντέλο του καταρράκτη[11] περιλαμβάνουν: 1) ορισμό απαιτήσεων(requirements definition), 2) σχεδιασμός του συστήματος και του λογισμικού(system and software design), 3) ενσωμάτωση και δοκιμές του συστήματος(integration and system testing), 4) λειτουργία και συντήρηση(operation and maintenance). Κάθε δραστηριότητα υποστηρίζεται από τη V&V τεχνική που υποστηρίζει την παραγωγή καλά ορισμένων προϊόντων. Από τη στιγμή που τα προϊόντα από την μια δραστηριότητα γίνονται είσοδοι στην επόμενη δραστηριότητα, καμία επόμενη φάση δεν μπορεί να ξεκινήσει μέχρι ότου η προηγούμενη φάση τελειώσει και όλα τα προϊόντα της οριστούν σαν ολοκληρωμένα. Για να ολοκληρωθεί η παραγωγή πρέπει να εγκριθεί από αυτές τις QA δραστηριότητες. Το σχήμα 18 μας δείχνει την διαδικασία ανάπτυξης και θα χρησιμοποιήσουμε αυτό το ολοκληρωμένο μοντέλο για να το συγκρίνουμε με τις τεχνικές διασφάλισης της ποιότητας των ευέλικτων μεθόδων.



Σχήμα 18: Ολοκληρωμένο μοντέλο ποιότητας

9.5 Ευέλικτοι μέθοδοι με QA[12]

Στο Σχήμα 19 παρουσιάζεται ένας γενικευμένος κύκλος ζωής της ανάπτυξης με ευέλικτη μέθοδο. Σε αυτό το διάγραμμα κάποια ευέλικτα στάδια κανονικά επικαλύπτουν το ένα το άλλο. Αυτό κάνει δύσκολο το να φανούν με ευκρίνεια οι φάσεις. Η γενική διαδικασία ανάπτυξης είναι η ίδια με του μοντέλου του καταρράκτη, ωστόσο στις ευέλικτες μεθόδους ο επαναλαμβανόμενος μοναδιαίος κύκλος είναι μια σύντομη έκδοση, ο οποίος δεν υπάρχει στο κανονικό μοντέλο του καταρράκτη. Στο σχήμα 19, στην αριστερή πλευρά του διαγράμματος φαίνονται οι ευέλικτες διαδικασίες της κύριας ακολουθίας και στην δεξιά πλευρά συμπεριλαμβάνονται οι ευέλικτες πρακτικές που έχουν δυνατότητα διασφάλισης ποιότητας. Υπάρχουν 2 κύριες διαφορές μεταξύ των δυο μοντέλων, 1) στις ευέλικτες μεθόδους υπάρχουν μερικές πρακτικές που έχουν και λειτουργία ανάπτυξης και δυνατότητα διασφάλισης ποιότητας. Αυτό σημαίνει ότι οι ευέλικτες μέθοδοι μετακινούν κάποιες ευθύνες διασφάλισης ποιότητας στους κατασκευαστές. Αυτές οι πρακτικές στο σχήμα 19 είναι υπογραμμισμένες και τονισμένες, 2) στις ευέλικτες φάσεις ένα μικρό μέρος των αποτελεσμάτων στέλνονται συχνά στις πρακτικές διασφάλισης ποιότητας και παρέχεται γρήγορη ανατροφοδότηση, πχ. οι πρακτικές ανάπτυξης και οι πρακτικές διασφάλισης ποιότητας συνεργάζονται στενά μεταξύ τους και ανταλλάσουν τα αποτελέσματα γρήγορα ώστε να κρατούν την ταχύτητα της διαδικασίας. Αυτό σημαίνει ότι οι επικοινωνίας στις ευέλικτες μεθόδους είναι ταχύτερη από ότι στο μοντέλο του καταρράκτη.



Σχήμα 19:Ευέλικτες μέθοδοι και Διασφάλιση Ποιότητας(QA)

9.6 Ευέλικτες μέθοδοι, Τεχνικές ποιότητας[12]

Οι ευέλικτες μέθοδοι συμπεριλαμβάνουν πολλές πρακτικές που έχουν την πιθανή δυνατότητα για διασφάλιση της ποιότητας. Με την αναγνώριση αυτών των πρακτικών και την σύγκριση τους με τις τεχνικές ποιότητας που χρησιμοποιούνται στο μοντέλο καταρράκτη, μπορούμε να εξετάσουμε το επίπεδο της διασφάλισης της ποιότητας στις ευέλικτες μεθόδους. Μερικές από τις ευέλικτες πρακτικές που έχουν αναγνωριστεί σαν τεχνικές ποιότητας περιγράφονται παρακάτω, και σίγουρα υπάρχουν και άλλες τεχνικές που δεν έχουν οριστεί με σαφήνεια.

Έχοντας μια **εκ όψεως επαφή με τον πελάτη(on-site costumer)** είναι μια γενική πρακτική στις περισσότερες ευέλικτες μεθόδους. Ο πελάτης βοηθάει τον κατασκευαστεί να τελειοποιήσει και να διορθώσει τις απαιτήσεις. Ο πελάτης πρέπει να υποστηρίζει την ομάδα ανάπτυξης κατά την διάρκεια ολόκληρης της διαδικασίας ανάπτυξης. Δεν υπάρχει τέτοια δραστηριότητα στις παραδοσιακές μεθόδους. Στην μέθοδο καταρράκτη, οι πελάτες συνήθως εμπλέκονται στον ορισμό των απαιτήσεων και πιθανώς στο σχεδιασμό του συστήματος και του λογισμικού αλλά δεν εμπλέκονται τόσο πολύ και δεν συνεισφέρουν τόσα πολλά όπως αναμένεται στις ευέλικτες μεθόδους. Συνεπώς η εμπλοκή του πελάτης στις ευέλικτες μεθόδους είναι πιο έντονη απ’ ότι στην ανάπτυξη καταρράκτη.

Συνεργατικός προγραμματισμός(pair programming) που σημαίνει ότι δυο προγραμματιστές δουλεύουν συνεχώς στον ίδιο κώδικα. Ο Cockburn και ο Williams βρήκαν ότι ο συνεργατικός προγραμματισμός μπορεί να βελτιώσει την ποιότητα και να μειώσει τα ελαττώματα. Τα αποτελέσματα του δείχνουν ότι ο συνεργατικός προγραμματισμός περιλαμβάνει τεχνικές κώδικα V&V. Η τεχνική ώμο με ώμο(shoulder-to-shoulder) εξυπηρετεί σαν μια συνεχή διαδικασία αξιολόγησης του σχεδίου και του κώδικα και τα αποτελέσματα είναι η μείωση των ποσοστών λάθους. Αυτή η ενέργεια έχει αναγνωριστεί σαν συνεχείς εξέταση του κώδικα.

Συνεχείς ενσωματώσεις(continuous integration) είναι επίσης μια δημοφιλής πρακτική ανάμεσα στις ευέλικτες μεθόδους. Συνεχείς ενσωματώσεις σημαίνει ότι η ομάδα δεν ενσωματώνει στον υπάρχοντα κώδικα μια ή δυο φορές. Η ομάδα έχει την ανάγκη να κρατάει το σύστημα πλήρως ολοκληρωμένο κάθε στιγμή. Η ενσωμάτωση μπορεί να γίνει αρκετές φορές την ημέρα. Ο Martin κατέδειξε ότι: Το σημείο κλειδί είναι ότι η συνεχείς ενσωμάτωσης διορθώνουν πολλά ελαττώματα και έτσι αξίζει το κόστος. Η συνεχείς ενσωματώσεις επίσης μειώνουν τον χρόνο που οι άνθρωποι σπαταλάνε για την αναζήτηση ελαττωμάτων και επιτρέπουν την ανίχνευση προβλημάτων συμβατότητας νωρίς. Αυτή η πρακτική μπορεί να συμπεριφερθεί σαν τεχνική κώδικα V&V και είναι παράδειγμα δυναμικής τεχνικής. Η ανάπτυξη με το μοντέλο καταρράκτη επίσης απαιτεί ενσωμάτωση, αλλά είναι πολύ αργότερα και η συχνότητα της είναι πολύ μικρότερη απ‘ ότι στις ευέλικτες μεθόδους.

Αποδεκτές δοκιμές(Acceptable Testing) πραγματοποιούνται αφότου έχουν τελειώσει όλες οι περιπτώσεις ελέγχου μονάδων. Αυτή η δραστηριότητα είναι μια δυναμική τεχνική διασφάλισης της ποιότητας. Και η προσέγγιση του καταρράκτη έχει αποδεκτές δοκιμές αλλά η διαφορά μεταξύ ευέλικτων αποδεκτών δοκιμών και παραδοσιακών είναι η εξής: οι αποδεκτές δοκιμές γίνονται πολύ πιο νωρίς και πολύ πιο συχνά και όχι μόνο μια φορά.

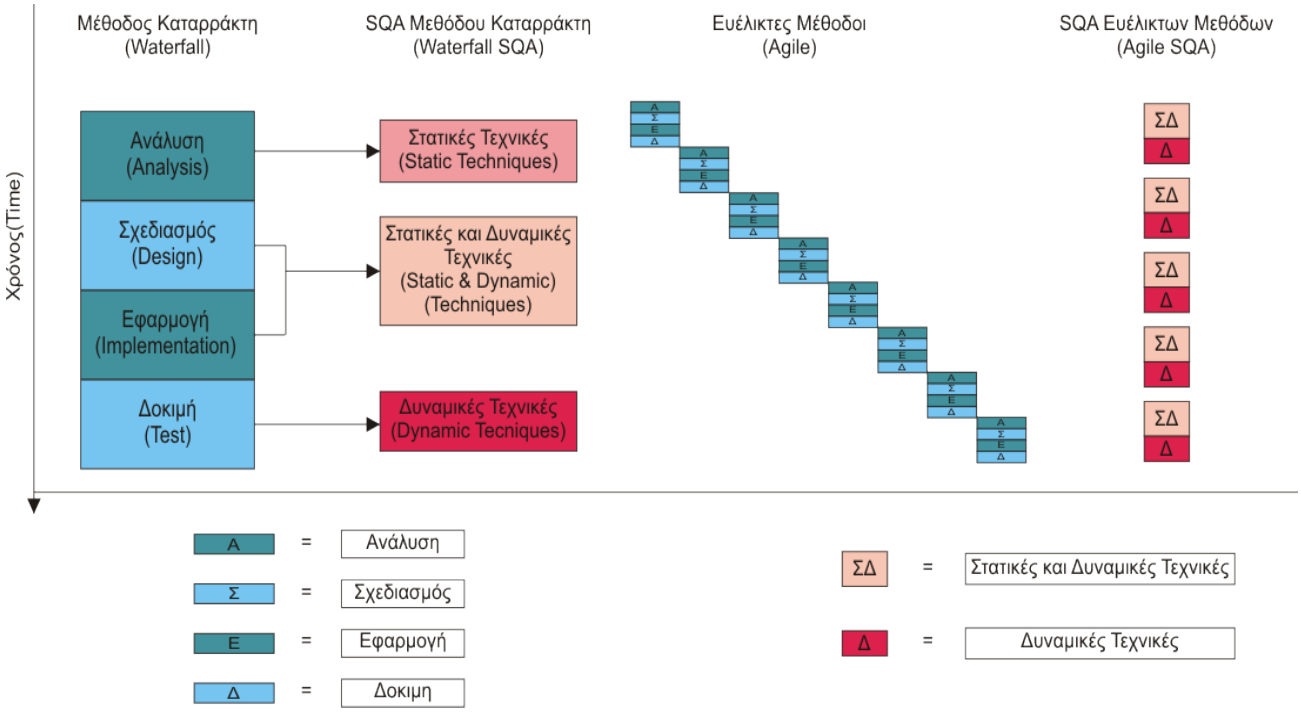
Επανακατασκευή(Refactoring)[11] είναι μια τεχνική με μεγάλη πειθαρχία που ξανακατασκευάζει ένα υπάρχον σώμα κώδικα, αλλάζοντας την εσωτερική δομή χωρίς να αλλάζει την εξωτερική συμπεριφορά του. Ο πυρήνας της είναι μια σειρά από μικρές συμπεριφορές που συντηρούν τους μετασχηματισμούς. Κάθε μετασχηματισμός κάνει μια μικρή αλλαγή, αλλά μια σειρά από μετασχηματισμούς μπορεί να παράγει μια σημαντική επανακατασκευή. Επειδή κάθε μετασχηματισμός είναι μικρός, η πιθανότητα να πάει κάτι στραβά είναι επίσης μικρή και το σύστημα είναι απόλυτα λειτουργικό μετά από κάθε μικτό μετασχηματισμό. Ο μετασχηματισμός μπορεί να μειώσει τις πιθανότητες ώστε ένα σύστημα να μπορεί να πληγεί σοβαρά κατά την διάρκεια της επανακατασκευής. Κατά την διάρκεια της επανακατασκευής οι κατασκευαστές ξαναφτιάχνουν τον κώδικα και αυτή η ενέργεια παρέχει την λειτουργία εξέτασης του κώδικα. Αυτή η ενέργεια μειώνει την πιθανότητα για γενικευμένα λάθη κατά την διάρκεια της ανάπτυξης.

Εάν λοιπόν συγκρίνουμε τα διαφορές μεταξύ ευέλικτων δραστηριοτήτων διασφάλισης ποιότητας και SQA του μοντέλου καταρράκτη από 3 οπτικές γωνίες θα δούμε ότι: 1)πολλές από τις ευέλικτες δραστηριότητες συμβαίνουν πολύ πιο νωρίς απ’ ότι στην ανάπτυξη με την μέθοδο του καταρράκτη, 2)η συχνότητα αυτών των δραστηριοτήτων είναι πολύ πιο μεγάλη απ’ ότι στο μοντέλο του καταρράκτη, 3)οι ευέλικτες μέθοδοι έχουν λιγότερες τεχνικές διασφάλισης της ποιότητας απ’ ότι οι παραδοσιακές μέθοδοι ανάπτυξης.[10]

Οι ευέλικτες μέθοδοι προχωρούν στην φάση ανάπτυξης πολύ γρήγορα. Επίσης οι κατασκευαστές είναι περισσότερο υπεύθυνοι για την διασφάλιση της ποιότητας αν συγκριθούν με το να έχουν μια ξεχωριστή QA ομάδα και διαδικασία. Αυτό επιτρέπει περισσότερη ένταξη της διασφάλισης της ποιότητας στην φάση ανάπτυξης. Μικρές εκδόσεις επίσης φέρνουν ανατροφοδότηση από τον πελάτη για την επικύρωση του προϊόντος και την επικύρωση των απαιτήσεων. Οι QA τεχνικές τις για τις ευέλικτες μεθόδους είναι βασισμένες στις:

- Εφαρμογή των δυναμικών τεχνικών όσο πιο νωρίς γίνεται(πχ. αποδεκτές δοκιμές)
- Μεταφορά περισσοτέρων ευθυνών για την διασφάλιση της ποιότητας στους κατασκευαστές(εξέταση του κώδικα στον συνεργατικό προγραμματισμό, επανακατασκευή, πρότυπα κωδικοποίησης)
- Έγκαιρη επικύρωση του προϊόντος(εκ όψεως επαφή με τον πελάτη, μικρές εκδόσεις, συνεχείς ενσωματώσεις)

Στο σχήμα 20 παρουσιάζεται ο κύκλος ζωής του μοντέλου του καταρράκτη και των ευέλικτων μεθόδων ανάπτυξης βασισμένος στο χρόνο και τις διαθέσιμες τεχνικές τους για διασφάλιση ποιότητας. Μπορούμε να διακρίνουμε ότι οι δυναμικές τεχνικές εφαρμόζονται αργά στην ανάπτυξη του καταρράκτη όταν συγκριθούν με την ευέλικτη ανάπτυξη. Στον ευέλικτο κύκλο ανάπτυξης οι στατικές και δυναμικές τεχνικές μπορούν και οι δυο να εφαρμοστούν από τα αρχικά στάδια.



Σχήμα 20: SQA Χρονοδιάγραμμα

10. Σχεδιασμός και εργαλεία Ευέλικτων έργων

10.1Εισαγωγή

[13]Υπάρχει ένας μεγάλος αριθμός από εργαλεία για την διαχείριση έργων που σχεδιάστηκαν για να υποστηρίζουν τις ευέλικτες μεθόδους όπως είναι η Scrum,η Extreme Programming(XP) και άλλες, συμπεριλαμβανομένων και διαφόρων ανοικτού κώδικα επιλογών.[12] Τα πιο κατάλληλα κριτήρια για την τελική επιλογή του πιο χρήσιμου εργαλείου για την διαχείριση του κάθε έργου είναι

- Το Σύνολο των χαρακτηριστικών του(Feature Set)
- Χρησιμότητα(Usability)
- Βιωσιμότητα(Viability)
- Καταλληλότητα για μεγάλα έργα και προϊόντα(Suitability for large projects and products)

Οι ευέλικτες μέθοδοι κέρδισαν ευρύτατη διάδοση και αποδοχή τα τελευταία χρόνια. Η Λογική Ενοποιημένη Διαδικασία(Rational Unified Process RUP) της IBM έχει προσαρμοστεί για να δημιουργήσει την Ευέλικτη Ενοποιημένη Διαδικασία(Agile Unified Process AUP). Η ευέλικτη προσέγγιση έχει κερδίσει έκταση ακόμα και σε μεγάλους οργανισμούς με πολύπλοκα έργα και γεωγραφικά χωρισμένες ομάδες. Ενώ η Ευέλικτη Διακήρυξη ευνοεί τα άτομα και τις αλληλεπιδράσεις πέρα από τις διαδικασίες και τα εργαλεία, ένα πολύ καλά φτιαγμένο εργαλείο για την διαχείριση ευέλικτων έργων έχει καθαρό προβάδισμα και είναι βασικό στοιχείο καθώς οι ευέλικτες ομάδες και οι οργανισμοί μεγαλώνουν τις γεωγραφικές τους θέσεις και την επιχειρησιακή κλίμακα τους. Τα ευέλικτα λογισμικά του ανοικτού κώδικα έχουν αποκτήσει μια στενή σχέση κατά το πέρασμα του χρόνου, με αποτέλεσμα πολλά έργα ανοικτού κώδικα να εμπνέονται και να εξαρτώνται από τις ευέλικτες κινήσεις. Η αγορά για τα ευέλικτα εργαλεία διαχείρισης έργων είναι τώρα ώριμη και εμποτισμένη με δεκάδες προσφορές από μικρούς και μεγάλους πωλητές και περισσότερα προϊόντα χτυπούν την αγορά καθημερινά. Οι εμπορικές προσφορές που οδηγούν την αγορά περιλαμβάνουν τα Rally, VersionOne, Thoughtworks Mingle, and Danube ScrumWorks. Μερικά ανοικτού κώδικα εργαλεία βρίσκονται στην αγορά ήδη εδώ και μερικό καιρό Ενώ περισσότερα έχουν αναδυθεί πιο πρόσφατα. Συγκρίνοντας τις ανοικτού κώδικα προσφορές, τα κυρίαρχα εμπορικά εργαλεία προσφέρουν καλύτερα χαρακτηριστικά για μεγάλους οργανισμούς και μεγάλης κλίμακας προϊόντα και έργα. Τείνουν επίσης να προσφέρουν περισσότερες ενσωματώσεις με εφαρμογές τρίτων. Αυτές οι επιπλέον επιλογές έρχονται και με επιπλέον κόστος ασφαλώς.

10.2Υποψήφια Εργαλεία[13]

Σε αυτή την παράγραφο συγκρίνονται μόνο ανοικτού κώδικα εργαλεία σχεδιασμού ευέλικτων έργων και (με μια μόνο εξαίρεση) λαμβάνοντας υπόψη μόνο αυτά τα οποία είναι υπό εξέλιξη ακόμα και έχει παρουσιαστεί η χρησιμότητα τους. Αυτά είναι τα εξής:

- Agilefant
- IceScrum
- Agilo
- EXPlainPMT
- XPlanner (Η αλήθεια είναι ότι αυτό το εργαλείο δεν είναι πλέον ενεργό σαν ανάπτυξη αλλά απέκτησε ευρεία χρησιμότητα κάποια χρόνια πριν, με πάνω από 45000 downloads της τελευταίας του έκδοσης μόνο. Περικλείεται διότι παρέχει μια καλή βάση για να καθορίσει πως τα εργαλεία εξελίχθηκαν στο πέρασμα του χρόνου.)

10.3Συνολική Σύγκριση[13]

Υπόμνημα:
√ Το χαρακτηριστικό περιέχετε
X Το χαρακτηριστικό δεν περιέχετε
√[1] Βλέπε σημείωση
* 1 άστρο βαθμολογίας: φτωχό
** 2 άστρα βαθμολογίας: αποδεκτό
*** 3 άστρα βαθμολογίας: καλό
**** 4 άστρα βαθμολογίας: άριστο

Πίνακας 8: Συνολική Σύγκριση Ευέλικτων Εργαλείων

	Agilefant	IceScrum	Agilo	EXPlainPM	XPlanner
Έκδοση (Version reviewed)	1.6.2	2#13	1.0.2 Pro	? (online demo)	0.7 beta
Άδεια (License)	MIT	GPL	Apache License 2.0	GPL	LGPL
Πλατφόρμα (Platform)	Java 6, Tomcat 5.5, MySQL	Java 1.5, Servlet μηχανή, HSQLDB (ή άλλη RDBMS)	Python,Trac, RDBMS (SQLite, MySQL, PostgreSQL)	Ruby, RDMBS (MySQL, PostgreSQL, ή SQLite)	Java 1.5, Tomcat 5.0 (όχι 5.5), Servlet 2.3, MySQL (ή άλλη RDMBS)
Απόλυτη ταξινόμηση λίστα έναντι προτεραιότητας (Backlog absolute ranking vs. prioritization)	Προτεραιότητα 1-5 κλίμακα (Priority, 1-5 scale)	Ταξινόμηση, drag and drop (Ranking, drag and drop)	Ταξινόμηση, drag and drop (Ranking, drag and drop)	Ταξινόμηση, όχι drag and drop (Ranking, not drag and drop)	Ταξινόμηση, όχι drag and drop (Ranking, not drag and drop)
Σημεία Πλάνου (Story points)	**X**	√	√[1]	√	**X**

Ώρες εργασίας (Task hours)	√	√	√	**X**	√
Εικόνα των εργασιών στον πίνακα (Task board view)	**X**	√	√	**X**	**X**
Χαρτογράφηση ολοκληρωτικής επανάληψης (Iteration burn down chart)	√	√[2]	√[3]	**X**[4]	√
Ιεραρχία των στοιχείων της Λίστας (Hierarchy of backlog items)	**X**	√	√[5]	√[6]	**X**
Εκδόσεις (Releases)	√	√	√[7]	**X**	**X**
Σχέδιο (πολλαπλές εκδόσεις (Roadmap (multiple releases))	√	√	√[8]	**X**	**X**
Πολλαπλά προϊόντα/έργα (Multiple products/ projects)	√	√	**X**	√	√
Σχεδιασμός χαρτοφύλακα (Portfolio planning)	√	**X**	**X**	**X**	**X**
Αποδεκτές δοκιμές (Acceptance tests)	**X**	√	**X**	√	**X**
Εντοπισμός ελαττώματος (Impediment tracking)	**X**	√	√	**X**	**X**
Ελαττώματα τύπου Λίστας αντικειμένων (Defects as backlog item type)	**X**	√	√[9]	**X**	√
Θέματα Πλάνου (Story Themes)	√	**X**	**X**	**X**	**X**
Ομάδες Χρηστών (Teams of users)	√	**X**	√	√	**X**
Ρόλοι Χρηστών (User roles)	Κανένας(None)	PO, SM, Μέλος	SM, PO,Μέλος	Κανένας(None)	Επόπτη, Συντάκτη,

		ομάδας και προσαρμο-σμένους ρόλους	Ομάδας		Διαχειριστή, Κύριο Διαχειριστή (Viewer, Editor, Admin, Super Admin)
Αναφορές (Reports)	* Χρονοδια-γράμματα μόνο (Timesheets only)	**X**	*** Μπορεί να αποθηκεύσει προσαρμοσμένες αναφορές (Can save customized reports)	**X**	*** Εκτεταμένη ενσωμάτωση αναφορών αλλά όχι προσαρμογή αναφορών (Extensive built-in reports, but no custom reporting)
Ενσωμάτωση και API (Intergration and API(s))	Κανένα(None)	Κανένα(None)	SVN ενσωμάτωση μόνο (SVN integration only)	Κανένα(None)	SOAP, συμβολισμός για URL συνδέσεις (SOAP, notation for URL linking)
Υποστήριξη (Support)	Email, forums	Email	Εμπορική, 8 € ανά μήνα στο χρήστη (Commercial 8 € per month per user)	Καμία(None)	Forums, αλλά αναποτελεσματικά (but inactive)
Forums	**	***	**** 2 θέματα την μέρα (2 topics per day)	Κανένα(None)	* 2 θέματα το μήνα με μερικές απαντήσεις
Οδηγός εγκατάστασης (Installation guide)	***	* Γαλλικά μόνο	***	Κανένας(None)	**
Έγγραφα Χρήστη (User docs)	**	**	**	Κανένα(None)	*
Χρησιμότητα (Usability)	***	** Όχι πάντα διαθέσιμη (Not always intuitive)	* Όχι διαθέσιμη πολλά κλικ (Not intuitive; lots of clicks)	*** Διαθέσιμη (Intuitive)	***
Καταλληλότητα για μεγάλα έργα (Suitability for large projects)	***	*	Κανένα αστέρι (No stars)	*	*

Σημειώσεις:

1. Η ευελιξία επιτρέπει μόνο τους ψευτό- Fibonacci αριθμούς για σημεία πλάνου
2. Η χαρτογράφηση της ολοκληρωμένης επανάληψης της IceScrum είναι ατελής, ο οριζόντιος άξονας δείχνει μόνο τις μέρες που πέρασαν στη Ροή παρά τον πλήρη χρόνο της Ροής
3. Η χαρτογράφηση της ολοκληρωμένης επανάληψης της Agilo είναι ατελής στην online demo έκδοση, ο κάθετος άξονας δεν υπάρχει
4. Το eXplainPMT έχει μια ολοκληρωτική χαρτογράφηση για όλο το έργο και όχι για την κάθε επανάληψη
5. Η λίστα της ιεράρχησης του Agilo δεν είναι εύκολο να χρησιμοποιηθεί, απαιτεί τα στοιχεία από τους αριθμούς των στοιχείων
6. Το eXplainPMT έχει "αρμοδιότητες" που μοιάζουν με τα έπη, αλλά είναι μόνο χαρακτηριστικά του πλάνου
7. Η ευελιξία χρησιμοποιεί τον όρο "ορόσημο"(milestones) στη θέση της έκδοσης(releases)
8. Τα χαρακτηριστικά του σχεδίου του Agilo δεν είναι χρήσιμα, δείχνει όλες τις Ροές και χρειάζονται πολλές κυλίσεις για να δεις όλες τις εκδόσεις
9. Το Agilo έχει ξεχωριστή Λίστα για τα λάθη

11.Agilefant [13]

Τα παρακάτω αναφέρονται για την έκδοση 1.6.2 του προγράμματος.

11.1Η Ιδέα(Concepts)

Τα προϊόντα είναι τα πιο υψηλής ποιότητας δημιουργήματα, και κάθε επέκταση μπορεί να έχει πολλαπλά προϊόντα. Κάθε προϊόν μπορεί να έχει ένα ή περισσότερα έργα, τα οποία είναι ουσιαστικά εκδόσεις. Κάθε έργο μπορεί να έχει μια ή περισσότερες επαναλήψεις. Κάθε προϊόν, έργο(έκδοση) και επανάληψη έχει την δική του Λίστα που περιέχει τα πλάνα. Τα πλάνα μπορούν να μεταφέρονται σε οποιαδήποτε άλλη Λίστα, δηλαδή από την Λίστα του προϊόντος στη Λίστα της επανάληψης. Τα πλάνα μπορούν να αποτελούνται από μηδενικές ή περισσότερες εργασίες. Τα έργα μπορούν να ιεραρχούνται στο τομέα “Χαρτοφύλακα”(Portfolio view).

Το εργαλείο μπορεί να υποστηρίζει πολλές ταυτόχρονες επαναλήψεις, που επιτρέπουν μεγαλύτερη οργάνωση στην αποτελεσματική χρήση του προϊόντος.

Το Agilefant υποστηρίζει πολλούς χρήστες οι οποίοι μπορούν να ομαδοποιηθούν σε ομάδες. Ωστόσο δεν υποστηρίζει καθόλου ρόλους για τους χρήστες, ένας χρήστης είναι ένας χρήστης χωρίς καμία διαφορά στα δικαιώματα ή την πρόσβαση σε χαρακτηριστικά. Στις ομάδες μπορεί να ανατεθεί συγκεκριμένη επανάληψη.

Η έκδοση 1.6.2 δεν περιλαμβάνει την ιδέα της υψηλής ποιότητας χαρακτηριστικών, αν και αυτό το χαρακτηριστικό έχει αναγγελθεί για την έκδοση 2.0 που υποτίθεται ότι θα εκδοθεί σύντομα. Υποστηρίζει την ιδέα των θεμάτων, που είναι μια ιδιότητα του πλάνου, και αυτό παρέχει έναν πιο απλό τρόπο για να συνδεθεί ένα γκρουπ από πλάνα.

11.2Βιωσιμότητα, υποστήριξη & τεκμηρίωση(Viability, Support & Documentation)

Τα βοηθητικά έγγραφα του προϊόντος είναι επαρκείς αλλά όχι άριστα. Το προϊόν είναι αρκετά διαισθητικό ώστε να μην χρειάζεται τόσο πολύ τεκμηρίωση. Η βοήθεια για την εγκατάσταση είναι επαρκής. Τα forum και τα email υποστήριξης είναι αρκετά καλά. Η τεκμηρίωση είναι ενεργή με την 2,0 έκδοση.

11.3Ευχρηστία(Usability)

Το Agilefant είναι διαισθητικό. Ο χρήστης είναι σε θέση να καταλάβει πως θα χρησιμοποιήσει όλα τα χαρακτηριστικά χωρίς να βασίζεται στη βοήθεια. Υστερεί στα drag and drop χαρακτηριστικά, αλλά παρακινεί την καλή χρήση των δευτερεύων χαρακτηριστικών του πελάτη η οποία κάνει την ενασχόληση του χρήστη πιο εύκολη.

11.4Πλεονεκτήματα(Strengths)

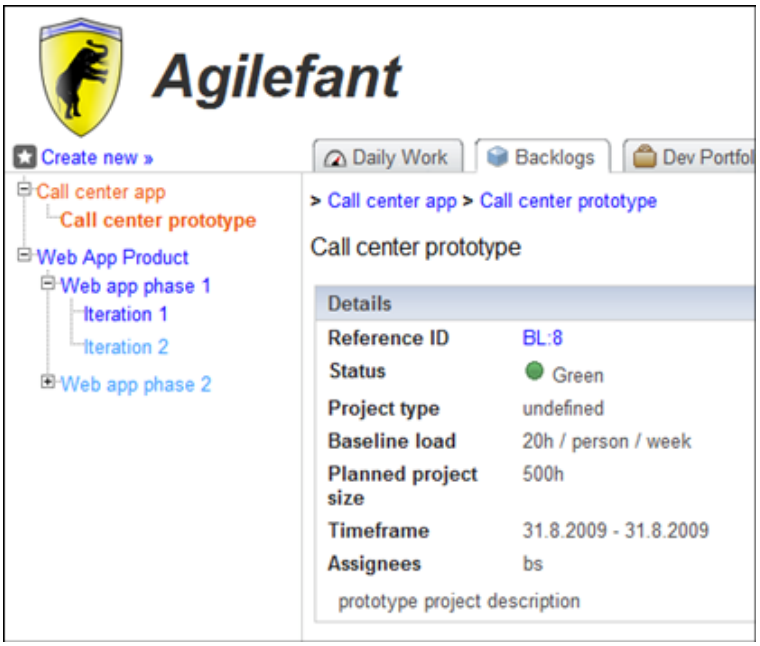
- Πλούσιο σύνολο χαρακτηριστικών
- Κατάλληλο για μεγάλους οργανισμούς και έργα, εκτός από την έλλειψη των “επών”(epics) ή την ιεράρχηση των πλάνων κάτι που στη 2.0 έκδοση διορθώνεται.
- Λογικά διαισθητικό και εύκολο στη χρήση.
- Χαρακτηριστικά προγραμματισμού “Χαρτοφύλακα”.
- Χαρακτηριστικά για φύλλα παρουσιών.

11.5Αδυναμίες(Weaknesses)

- Τα πλάνα μπορούν να υπολογιστούν μόνο σε ώρες και όχι σε σημεία.
- Τα πλάνα δεν μπορούν να βαθμολογηθούν παρά μόνο να ιεραρχηθούν σε μια κλίμακα από το 1-5.
- Καμία drag and drop επαναταξινόμηση των πλάνων.
- Έλλειψη των “επών”(epics) ή της ιεράρχησης των πλάνων.
- Καμία εμφάνιση πίνακα εργασιών
- Καμία διαφοροποίηση μεταξύ των ρόλων των χρηστών.

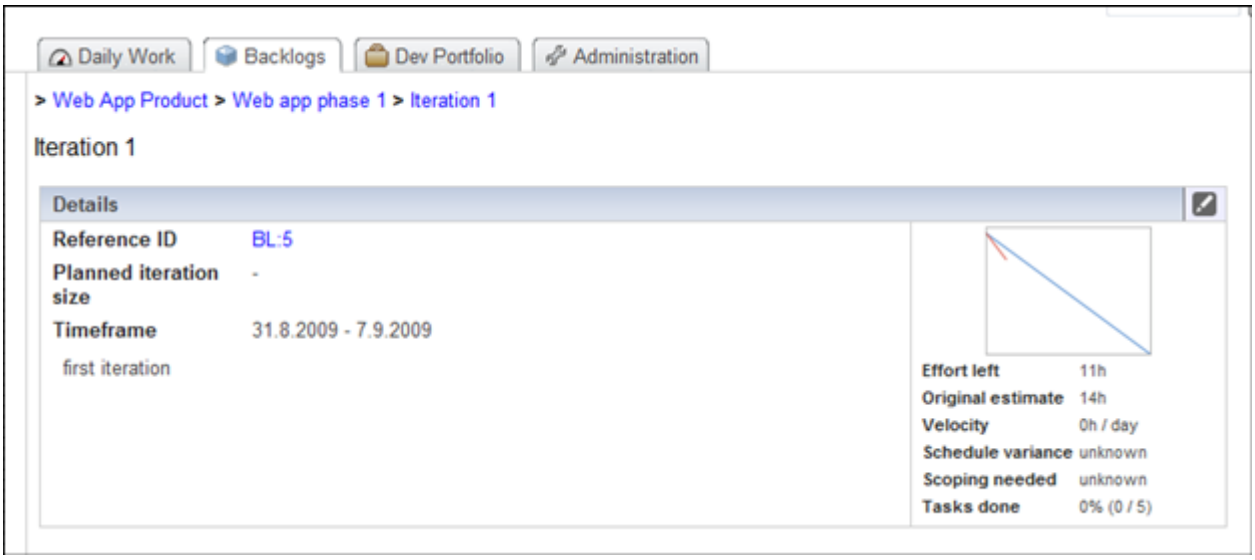
11.6Γενική εκτίμηση(Overall Rating)

Το Agilefant είναι ένα πολύ ικανό εργαλείο με ένα πλούσιο σύνολο χαρακτηριστικών αλλά και μερικές αδυναμίες. Είναι καλύτερα ταιριαστό σε μεγάλα έργα και μεγάλους οργανισμούς αλλά η έλλειψη ιεράρχησης των πλάνων, το επίπεδο απαιτήσεων είναι ένα σημαντικό μειονέκτημα για μεγάλα έργα.

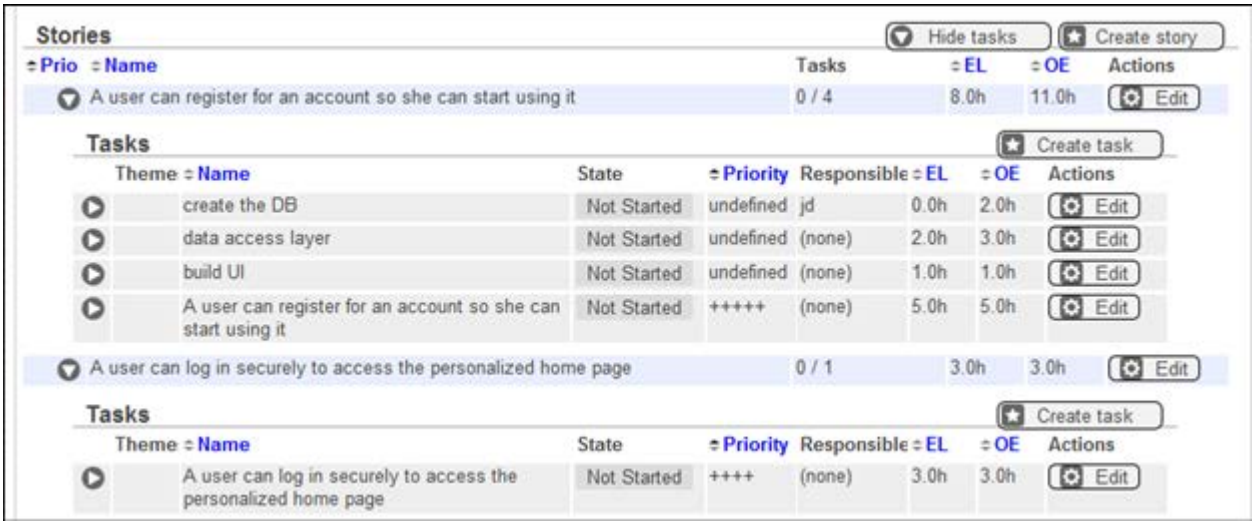


Εικόνα 1: Πλοήγηση με μορφή τύπου Δέντρου

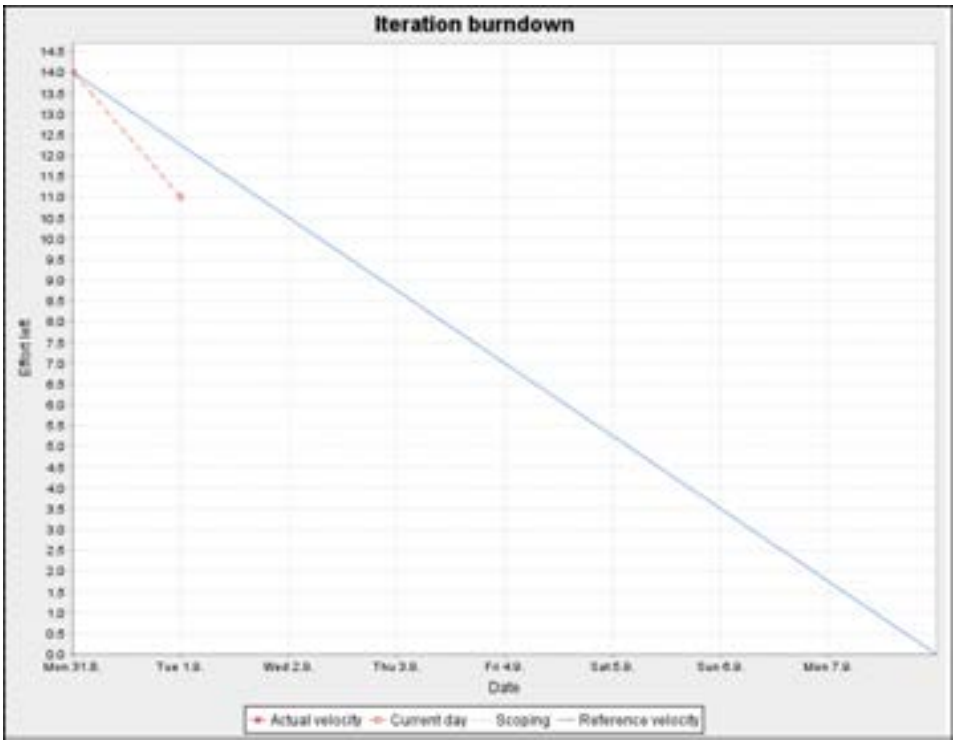
Εικόνα 2: Αντικείμενα που μπορεί να φτιάξει ο χρήστης



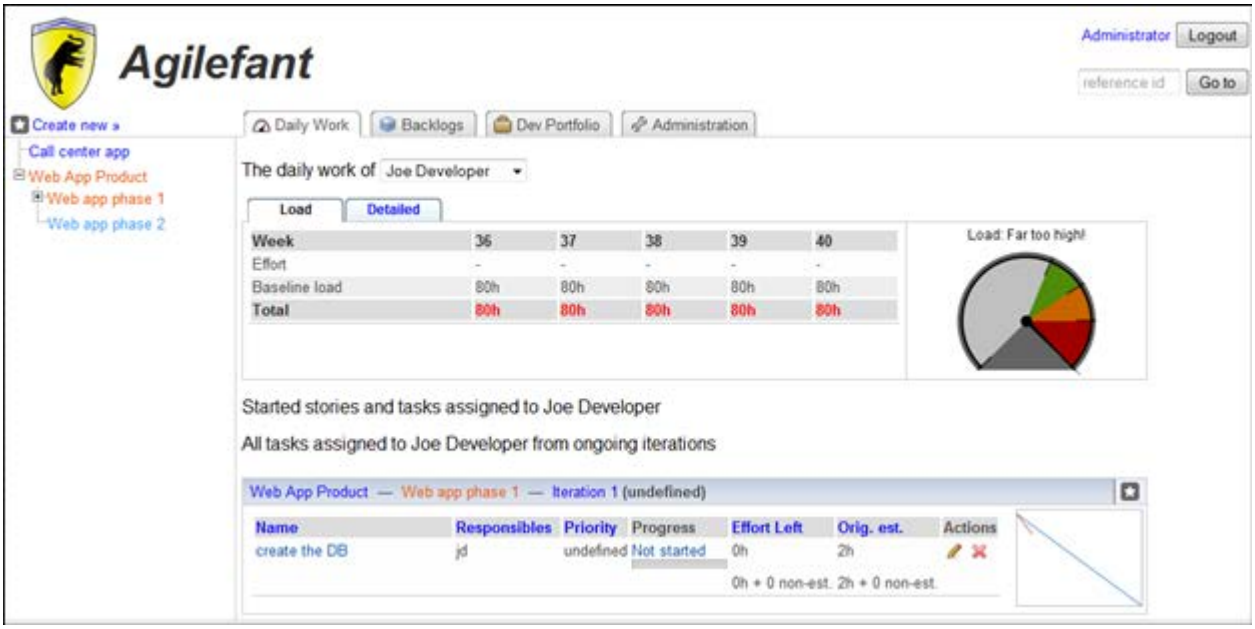
Εικόνα 3: Η εικόνα με τις Λεπτομέρειες Επανάληψης



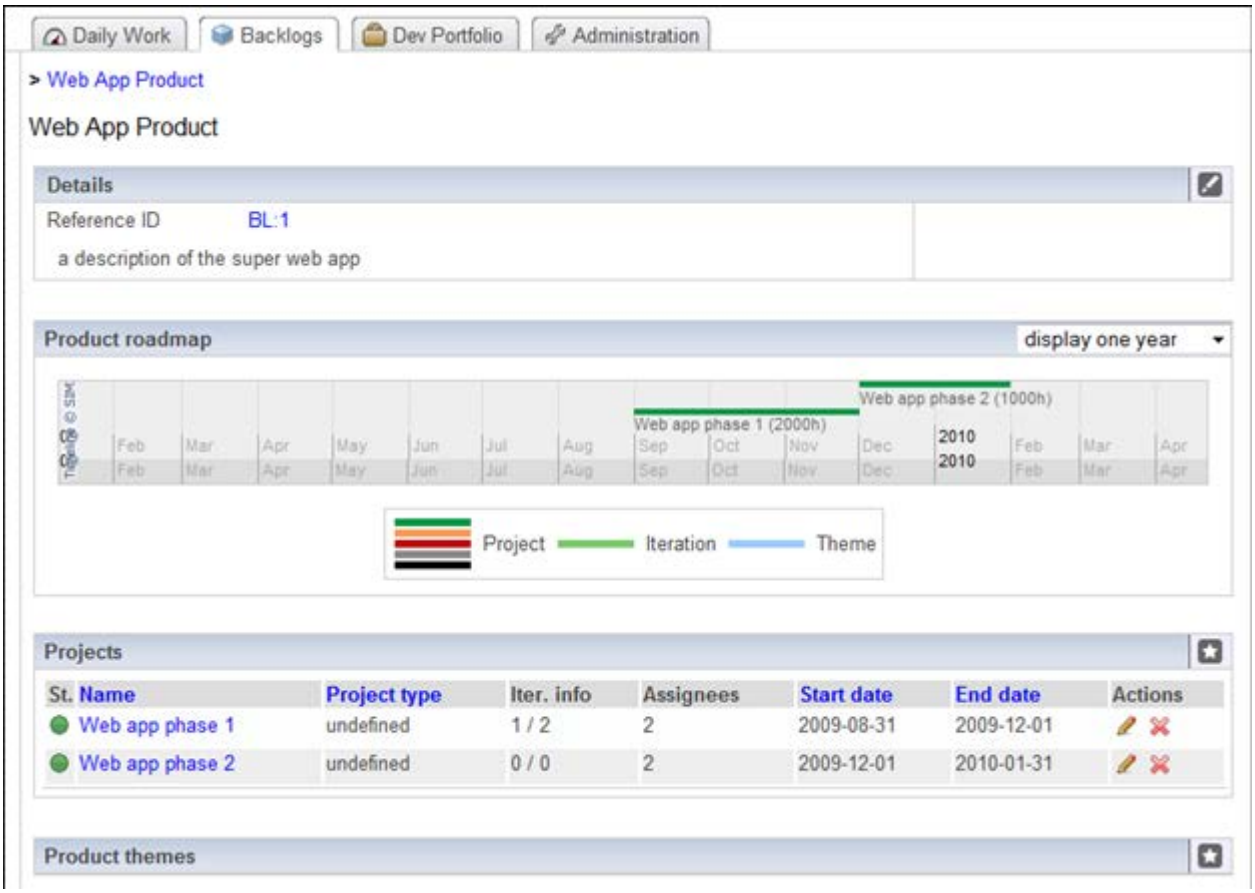
Εικόνα 4: Η Λίστα των πλάνων και των εργασιών μέσα σε μια επανάληψη



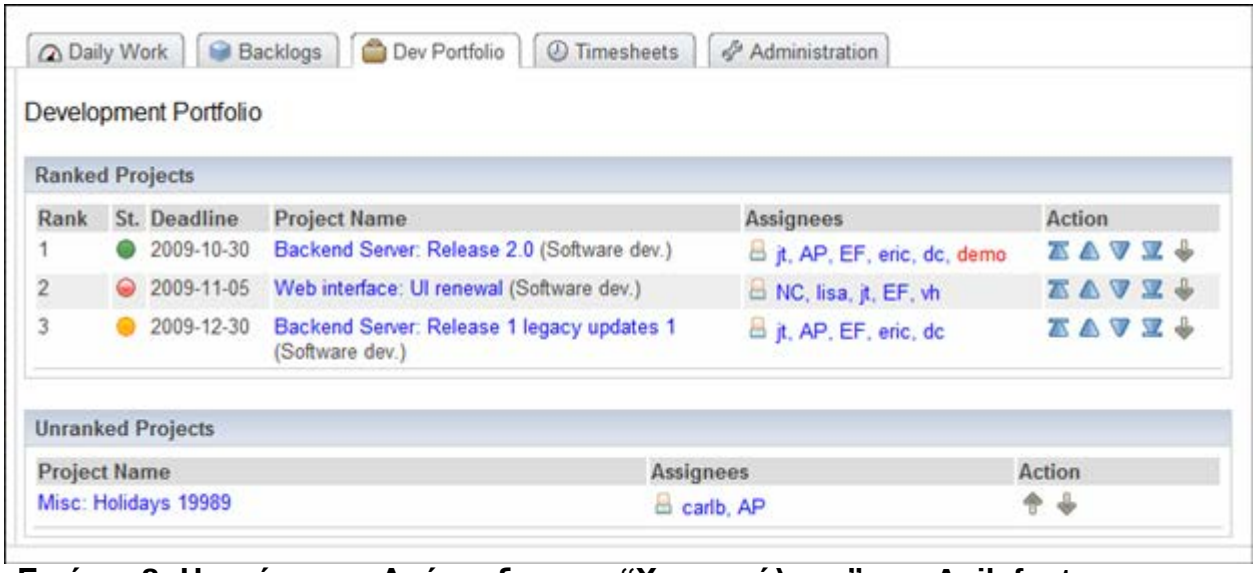
Εικόνα 5: Το διάγραμμα Προόδου Επανάληψης



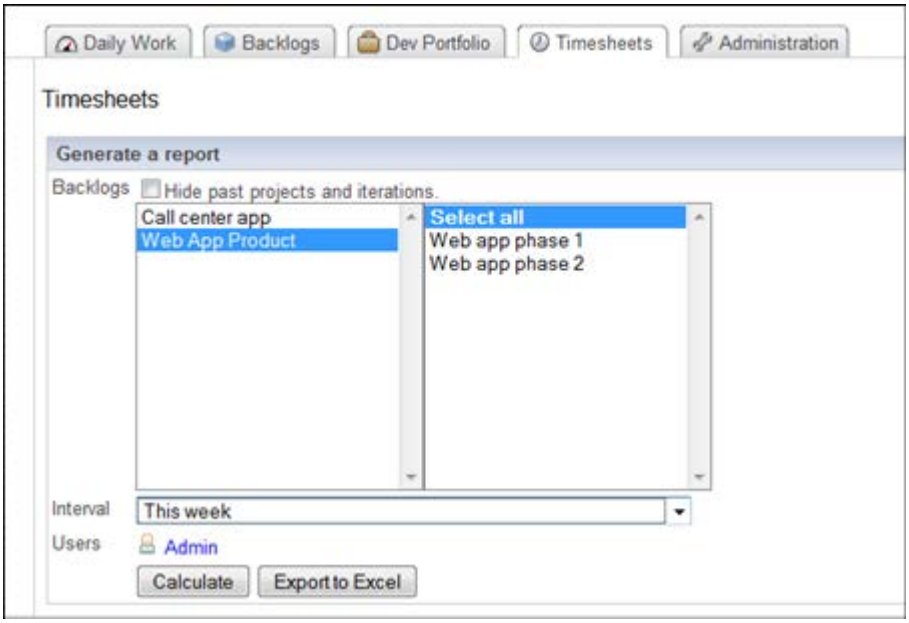
Εικόνα 6: Η εικόνα της Καθημερινής Εργασίας



Εικόνα 7: Η εικόνα του Προϊόντος



Εικόνα 8: Η εικόνα της Ανάπτυξης του "Χαρτοφύλακα" του Agilefant που επιτρέπει την ιεράρχηση των έργων.



Εικόνα 9: Το χαρακτηριστικό παρουσιών του Agilefant παρέχει μια αναφορά για το πώς οι χρήστες διαχειρίστηκαν το χρόνο τους.

12.IceScrum[13]

Τα παρακάτω αναφέρονται για την έκδοση 2#13 του προγράμματος.

12.1Η Ιδέα(Concepts)

Τα προϊόντα(που σε ορισμένες περιπτώσεις αναφέρονται και σαν έργα) είναι τα πιο υψηλής ποιότητας δημιουργήματα και κάθε επέκταση μπορεί να έχει πολλαπλά προϊόντα. Κάθε προϊόν έχει μια απλή Λίστα(Backlog) και μια πορεία που ακολουθεί(roadmap). Μια Λίστα περιέχει χαρακτηριστικά(παρόμοια με τα έπη), πλάνα χρηστών, ελαττώματα και τεχνικά πλάνα. Μια πορεία περιέχει πολλαπλές εκδόσεις κάθε μια από τις οποίες έχει μια απλή έκδοση σχεδίου. Μια έκδοση σχεδίου αποτελείται από πολλαπλές Ροές. Κάθε Ροή αποτελείται από πλάνα που με την σειρά τους αποτελούνται από εργασίες και αποδεκτές δοκιμές. Τα ελαττώματα μπορούν να ανιχνευθούν για κάθε προϊόν.

Αντίθετα με το Agilefant, το IceScrum εμπεριέχει ένα πίνακα εργασιών για τις επαναλήψεις, επιτρέποντας τις εργασίες να χρησιμοποιούν την τεχνική drag and drop. Επίσης επιτρέπει την χρήση drag and drop για την ιεράρχηση των πλάνων των χρηστών στην Λίστα. Πολλές επιλογές είναι διαθέσιμες με δεξί κλικ που εμφανίζει ένα μενού επιλογών, το οποίο αν και δεν είναι εμφανές με την πρώτη ματιά μόλις παρατηρηθεί είναι πάρα πολύ εύκολο στην χρήση.

Το IceScrum είναι το μόνο προϊόν το οποίο έχει ένα χαρακτηριστικό τον Poker σχεδιασμό. Αυτό το χαρακτηριστικό επιτρέπει σε μια κατανεμημένη ομάδα να παίξει τον Poker σχεδιασμό για τον υπολογισμό των χαρακτηριστικών και των πλάνων των χρηστών.

Το IceScrum επιτρέπει μόνο μια έκδοση και μια μόνο Ροή να είναι ενεργεί σε κάποια χρονική περίοδο(για ένα συγκεκριμένο προϊόν) κάνοντας το όχι κατάλληλο για μεγάλους οργανισμούς που χρειάζονται πολλαπλές σύγχρονες Ροές με πολλαπλές ομάδες που εργάζονται παράλληλα για ένα μοναδικό προϊόν.

Οι χρήστες του IceScrum μπορούν να έχουν όποιον από τους ρόλους της Scrum θέλουν(ιδιοκτήτη Προϊόντος, Scrum Master, Μέλος Ομάδας) και επιπλέον μπορεί να δημιουργηθεί και ο ρόλος του πελάτη. Δεν επιτρέπει όμως τους χρήστες να ομαδοποιούνται σε ομάδες.

Πέραν του διαγράμματος προόδου, το IceScrum δεν έχει καμία άλλη αναφορά ή API(Application Programming Interface) αναφορά.

12.2Βιωσιμότητα, υποστήριξη & τεκμηρίωση(Viability, Support & Documentation)

Τα βοηθητικά έγγραφα του προϊόντος είναι επαρκείς αλλά όχι άριστα. Ο οδηγός εγκατάστασης έχει γραφεί μόνο στα Γαλλικά. Τα forum και τα email υποστήριξης φαίνονται να είναι αρκετά καλά. Η ανάπτυξη του προϊόντος είναι ενεργεί.

12.3Ευχρηστία(Usability)

Μερικά χαρακτηριστικά είναι διαθέσιμα μόνο με ένα δεξί κλικ για την εμφάνιση ενός μενού. Είναι πολύ καλό χαρακτηριστικό μόλις το βρει κάποιος αλλά δεν είναι εμφανή σε νέους χρήστες. Έχει σχετικά ένα πλούσιο interface για τον χρήστη με την δυνατότητα drag and drop σε πολλά σημεία.

12.4Πλεονεκτήματα(Strengths)

- Ένα σύνολο πλούσιων χαρακτηριστικών.
- Η εικόνα της Λίστας της Ροής μοιάζει με έναν φυσικό πίνακα εργασιών αρκετά αποτελεσματικό
- Η εικόνα της Πορείας, της Έκδοσης Σχεδίου και του Σχεδίου Ροής υποστηρίζει πολλά επίπεδα σχεδιασμού.
- Οι αποδεκτές δοκιμές μπορεί να καταγραφούν για κάθε πλάνο.
- Εμπεριέχει το χαρακτηριστικό του Poker σχεδιασμού.
- Υποστηρίζει απόλυτα την ιεράρχηση των πλάνων με την μέθοδο drag and drop.

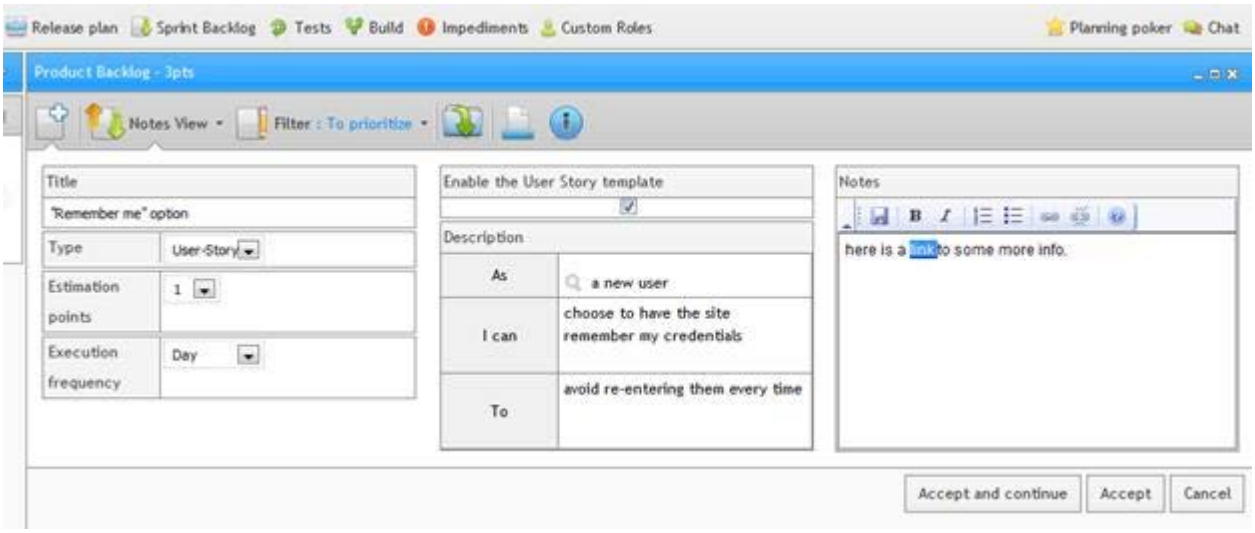
12.5Αδυναμίες(Weaknesses)

- Η διάταξη των καρτών του πλάνου κάνει δύσκολη την ιεράρχηση μια μεγάλης Λίστας προϊόντος.
- Κάποια χαρακτηριστικά δεν είναι διαθέσιμα. Η μέθοδος drag and drop λειτουργεί σε άλλα σημεία και σε άλλα όχι, αν και ο κέρσορας του ποντικιού κάνει να φαίνεται ότι η μέθοδος drag and drop λειτουργεί. Το μενού με δεξί κλικ δεν είναι προφανή αλλά είναι πολύ εύκολα να χρησιμοποιηθεί μόλις το βρει κάποιος.
- Το IceScrum δεν είναι κατάλληλο για μεγάλα έργα με πολλαπλές ομάδες εργασίας σε ένα μοναδικό προϊόν, μόνο μια έκδοση και μια μόνο ροή είναι ενεργεί κάθε φορά.

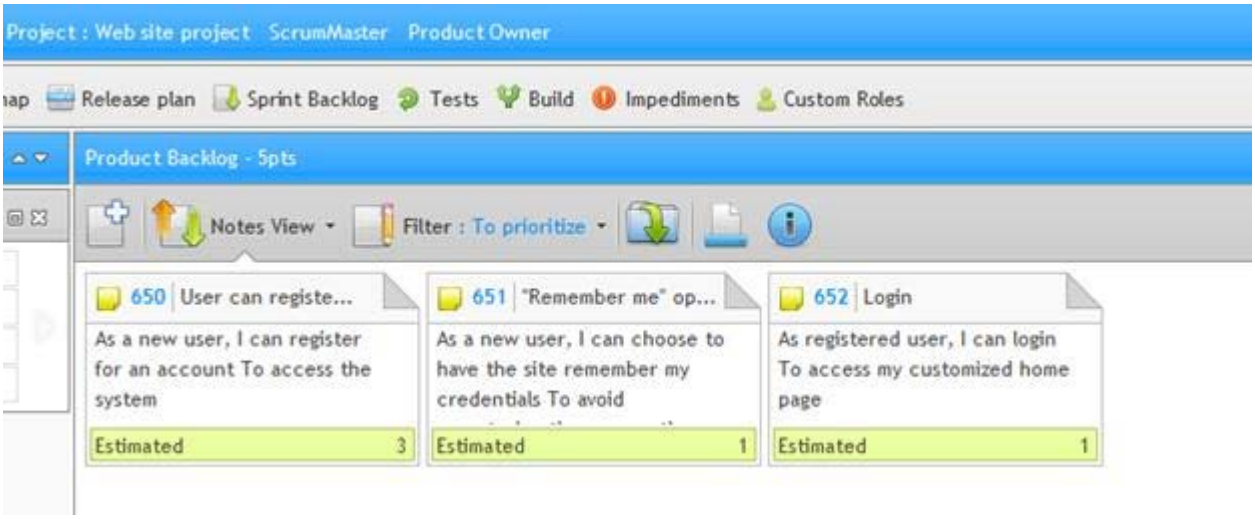
12.6Γενική εκτίμηση(Overall Rating)

Το IceScrum είναι ένα πολύ ικανό εργαλείο με ένα πλούσιο σύνολο χαρακτηριστικών και κάποια μειονεκτήματα. Αν και υποστηρίζει πολλαπλά προϊόντα(έργα), είναι κατάλληλο μόνο για μικρά έργα με μια μόνο ομάδα εργασίας σε μια μόνο ροή την φορά ανά προϊόν.

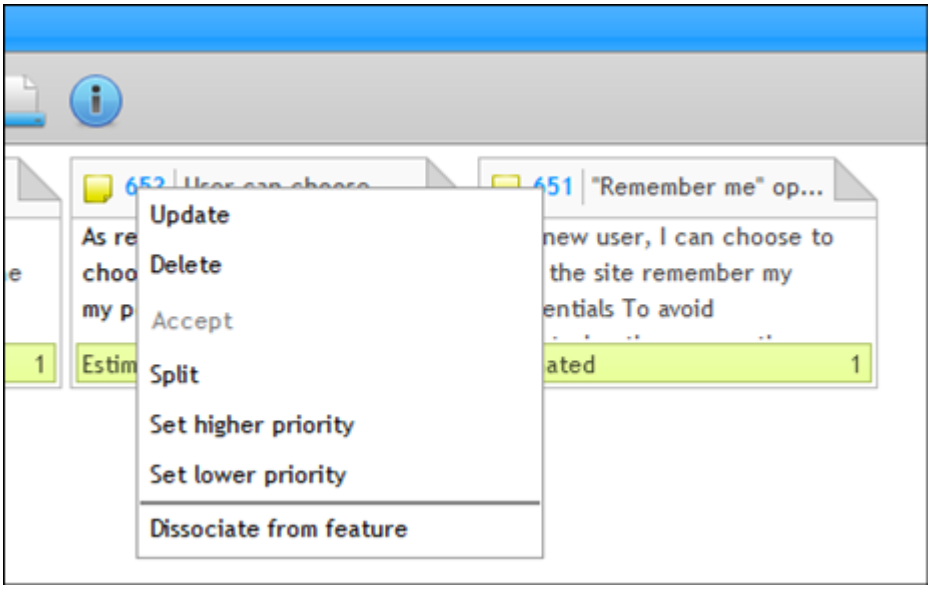
Εικόνα 10: Η εικόνα των χαρακτηριστικών της IceScrum.



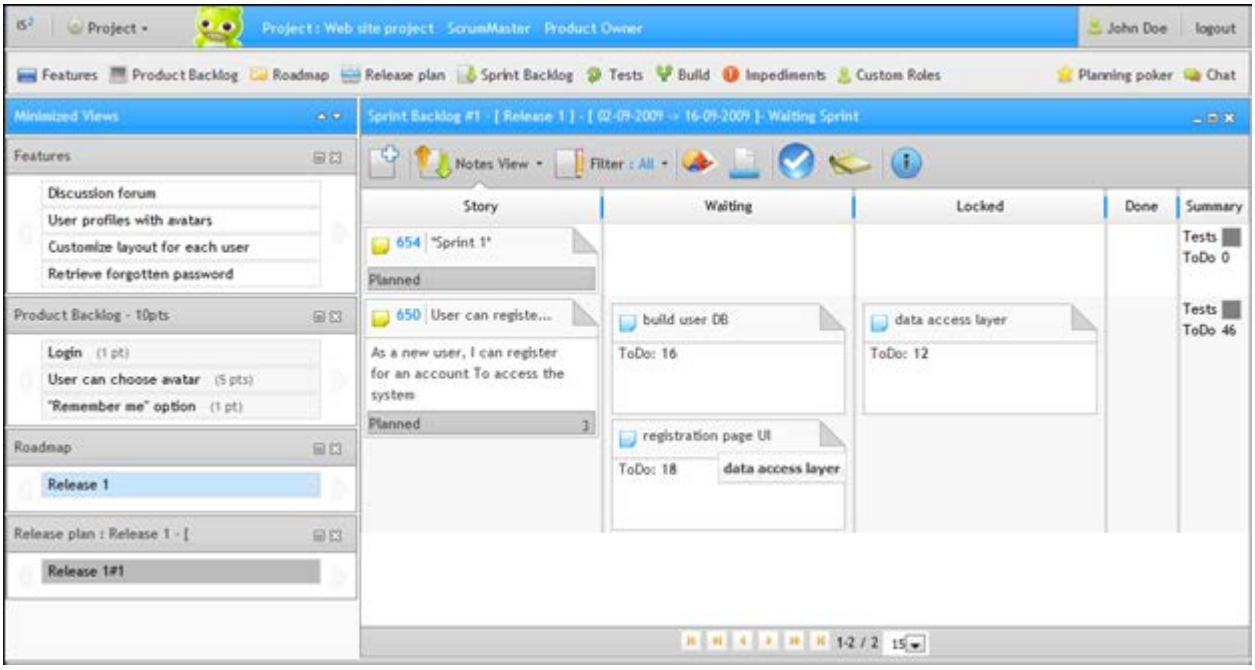
Εικόνα 11: Η κατασκευή ενός πλάνου χρήστη στην Λίστα του προϊόντος.



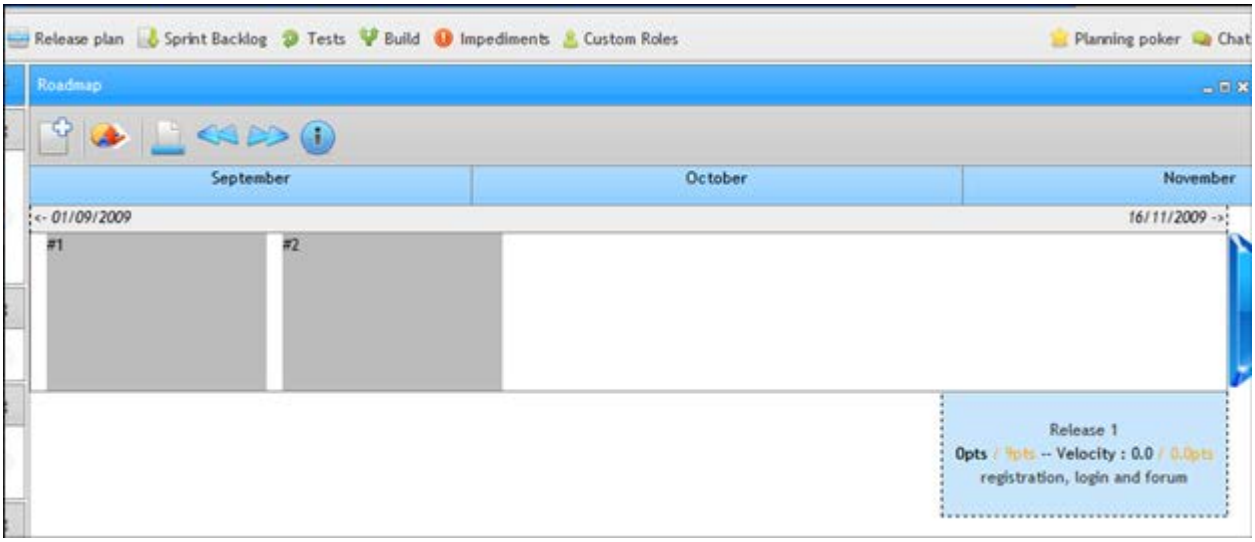
Εικόνα 12: Η εικόνα της Λίστας του προϊόντος. Κάθε κάρτα πλάνου μπορεί με την μέθοδο drag and drop να αλλάξει ιεραρχία. Τα χαρακτηριστικά μπορούν να συσχετιστούν με τα πλάνα σέρνοντας το χαρακτηριστικό από την αριστερή πλευρά πάνω στο επιθυμητό πλάνο.



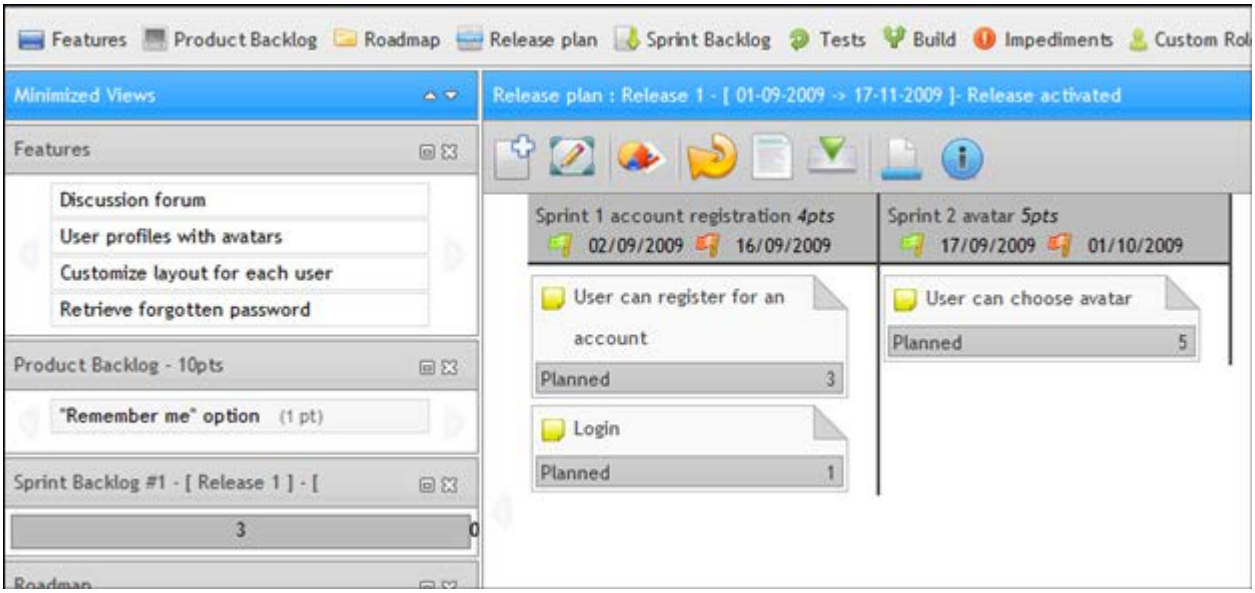
Εικόνα 13: Δεξί κλικ πάνω σε ένα χαρακτηριστικό μας εμφανίζει το μενού επιλογής.



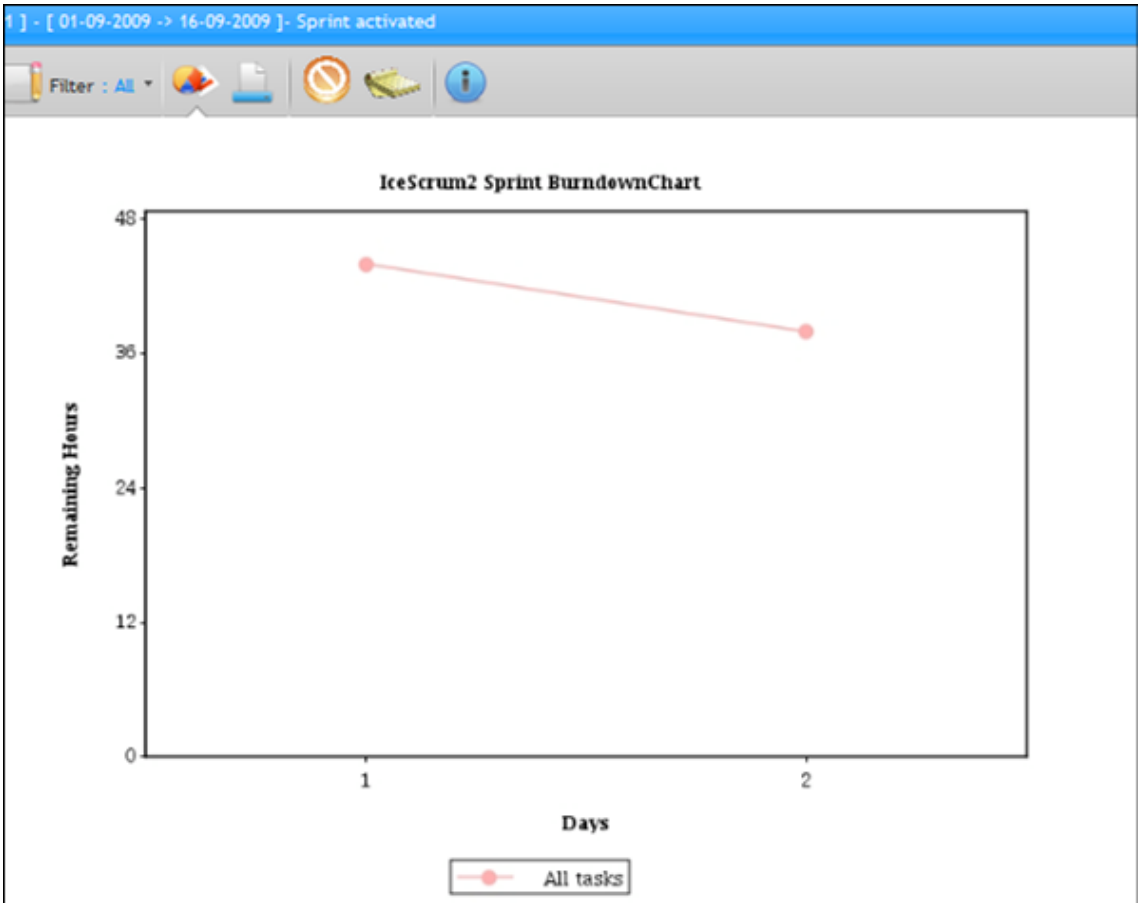
Εικόνα 14: Η εικόνα της Λίστας της Ροής. Για την πρόσθεση ενός πλάνου από την Λίστα του προϊόντος στη Ροή σέρνουμε το πλάνο από την αριστερή πλευρά και το αφήνουμε στην περιοχή των πλάνων της Ροής.



Εικόνα 15: Στην εικόνα της πορείας φαίνονται τα χρονικά όρια των εκδόσεων και των Ροών μέσα στην έκδοση.



Εικόνα 16: Η εικόνα του Σχεδίου της Έκδοσης. Τα αντικείμενα από την Λίστα του Προϊόντος μπορούν να συρθούν σε οποιαδήποτε Ροή.



Εικόνα 17: Η εικόνα του διαγράμματος προόδου. Το διάγραμμα είναι ελαττωματικό, δείχνει μόνο τις τωρινές μέρες που έχουν περάσει στον οριζόντιο άξονα πάρα τον πλήρη χρόνο που διήρκησε η Ροή. Σε αυτή την εικόνα η Ροή είναι 14 μέρες αλλά το διάγραμμα εμφανίζει μόνο τις 2 πρώτες.

13.Agilo[13]

Τα παρακάτω αναφέρονται για την έκδοση 1.0.2 Pro του προγράμματος.

13.1Η Ιδέα(Concepts)

Κάθε ανάπτυξη του Agilo υποστηρίζει μόνο ένα μοναδικό προϊόν\έργο – μια Λίστα. Κάθε επέκταση μπορεί να έχει πολλαπλά κύρια σημεία. Η μοναδική Λίστα περιέχει απαιτήσεις, πλάνα χρηστών και εργασίες. Αυτοί οι 3 τύποι στοιχείων υποστηρίζουν ιεραρχία με βάση την αναφορά: οι απαιτήσεις μπορούν να παραπέμψουν στα πλάνα, και τα πλάνα μπορούν να παραπέμψουν στις εργασίες. Η ιεραρχία είναι δύσκολο να καθιερωθεί και να αξιοποιηθεί ωστόσο το Agilo έχει μια ξεχωριστή Λίστα για τα μειονεκτήματα η οποία κάνει την δυσκολία του καθορισμού των μειονεκτημάτων ανάλογη των πλάνων των χρηστών. Επίσης έχει και μια ξεχωριστή Λίστα για τα ελαττώματα.

Στους χρήστες μπορούν να δίνονται οι ρόλοι του Master Scrum, του ιδιοκτήτη του προϊόντος ή του μέλους της ομάδας. Οι χρήστες επιπλέον μπορούν να ομαδοποιούνται και σε ομάδες.

Το Agilo επίσης έχει και ένα πολύ καλό και διαισθητικό πίνακα εργασιών(whiteboard) για τις επαναλήψεις, επιτρέποντας τις εργασίες να χρησιμοποιούν την μέθοδο drag and drop. Αυτό είναι κατά κύριο λόγο το καλύτερο χαρακτηριστικό του εργαλείου. Όλα τα άλλα χαρακτηριστικά είναι μη-διαισθητικά σύμφωνα με την σύγκριση. Πιθανόν οι Trac χρήστες θα το βρουν περισσότερο φιλικό.

Επειδή το Agilo επιτρέπει μόνο ένα προϊόν/έργο ανά εγκατάσταση, είναι κατάλληλο μόνο για μικρές περιόδους εργασίας σε μοναδικά προϊόντα, ή οργανισμούς που είναι πρόθυμοι να κάνουν διαφορετικές εγκαταστάσεις του ίδιου εργαλείου για κάθε προϊόν/έργο.

Εκτός του διαγράμματος πορείας το Agilo δεν έχει καμία άλλη αναφορά ή Api.

13.2Βιωσιμότητα, υποστήριξη & τεκμηρίωση(Viability, Support & Documentation)

Τα βοηθητικά έγγραφα του προϊόντος είναι επαρκείς αλλά όχι άριστα. Τα forum είναι πολύ ενεργά και δείχνουν να είναι ένα πολύ καλό μέρος για υποστήριξη. Η Pro έκδοση περιλαμβάνει επαγγελματική υποστήριξη για περίπου 8,50 euro το μήνα.

13.3Ευχρηστία(Usability)

Πολλά από τα χαρακτηριστικά του Agilo βρέθηκαν να είναι μη-διαισθητικά και δύσκολα στη χρήση, απαιτούνται πολλά κλικ και πολλές διαφορετικές σελίδες για να επιτύχεις μια κοινή εργασία. Το χαρακτηριστικό του πίνακα εργασιών είναι άριστο και πολύ εύκολο στη χρήση.

13.4Πλεονεκτήματα(Strengths)

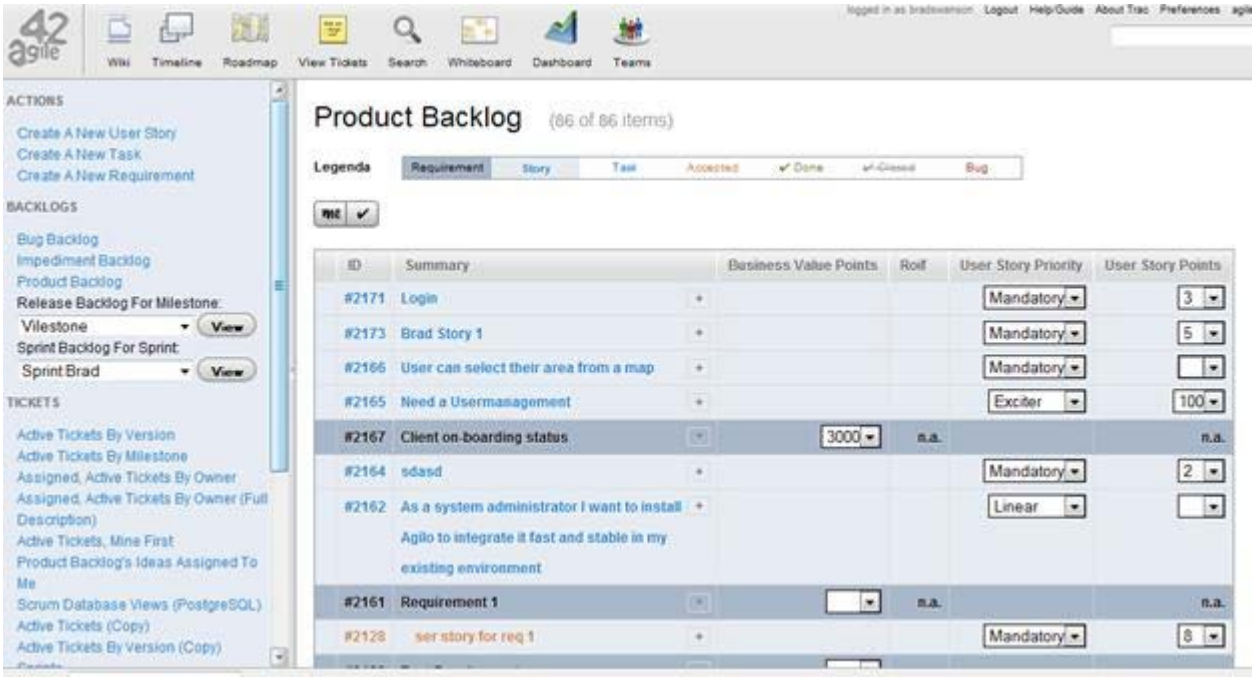
- Άριστος πίνακας εργασιών για επαναλήψεις με την μέθοδο drag and drop.
- Υποστηρίζει απόλυτη ταξινόμηση των πλάνων με drag and drop.
- Καλά χαρακτηριστικά αναφοράς, συμπεριλαμβανομένου και μετατροποιημένες αναφορές από τον χρήστη με δυνατότητα αποθήκευσης.

13.5Αδυναμίες(Weaknesses)

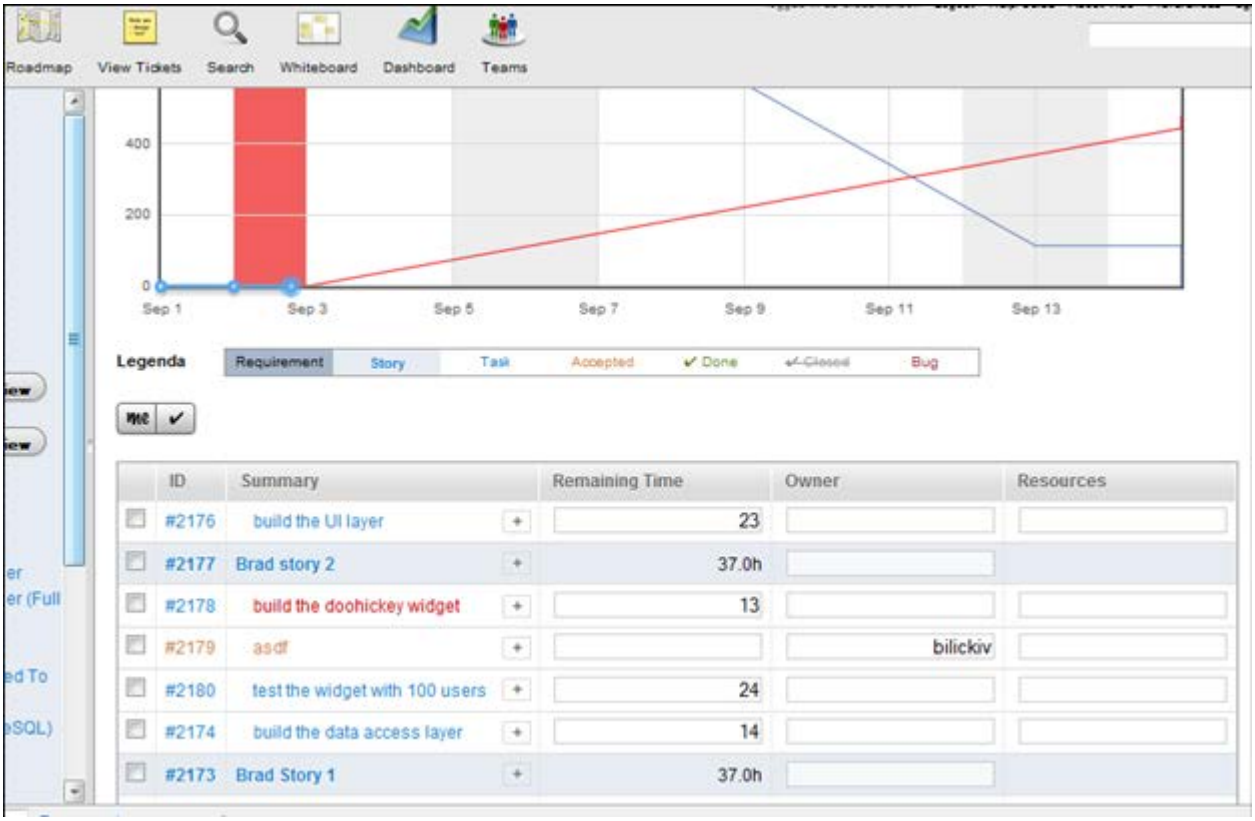
- Πολλά χαρακτηριστικά είναι μη-διαισθητικά. Φαίνεται σαν Trac(είναι ένα ανοικτού κώδικα, βασισμένο στο δίκτυο διαχείρισης έργων και ανίχνευση λαθών εργαλείο[14]) δεν υποστηρίζει την ιδέα της ευελιξίας τόσο καλά οπότε έπρεπε να τα ενσωματώσουν σε μοντέλο Trac.
- Όταν δημιουργηθεί μια εργασία, μπορεί να συσχετιστεί με μια Ροή αλλά όχι με ένα πλάνο.
- Πολλές λειτουργίες χρειάζονται πολλά κλικ για να ολοκληρωθούν.
- Υποστηρίζει μόνο μια μοναδική Λίστα προϊόντος ανά εγκατάσταση.

13.6Γενική εκτίμηση(Overall Rating)

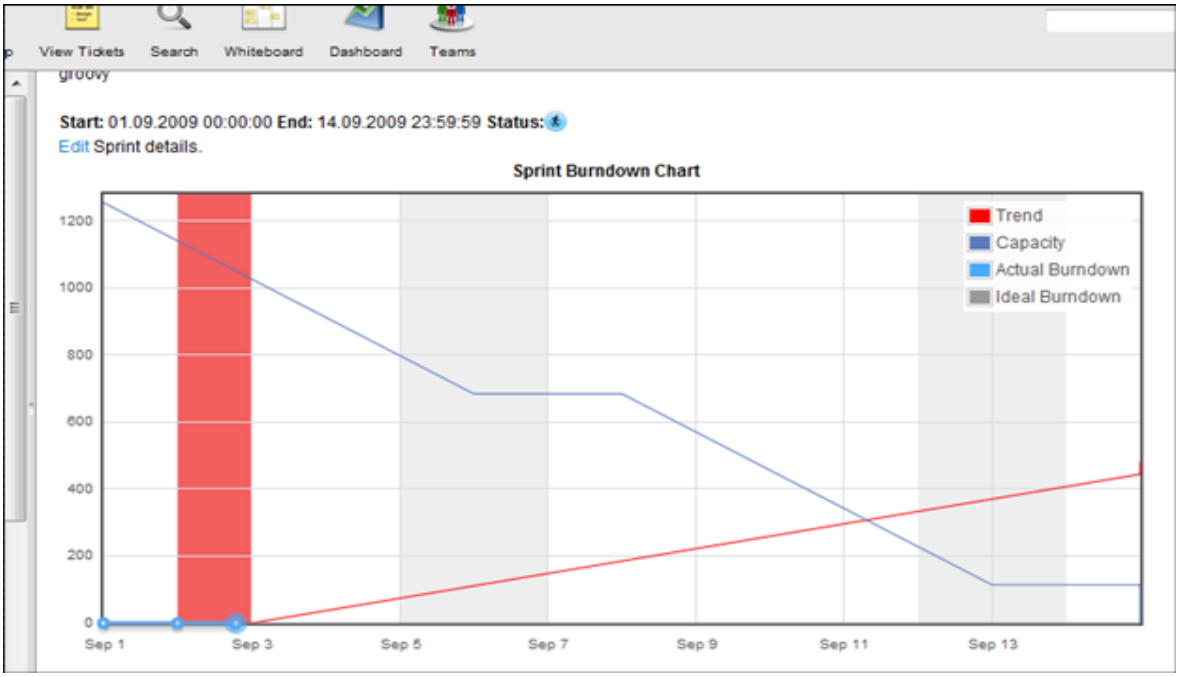
Παρόλο που ο πίνακας των εργασιών είναι διαισθητικός και άριστος, τα άλλα χαρακτηριστικά είναι λιγότερο διαισθητικά και συχνά όχι πολύ εύκολα στη χρήση. Το όριο της δημιουργίας ενός προϊόντος ανά εγκατάσταση είναι πολύ μεγάλο μειονέκτημα. Στο διάγραμμα Ροής στην δοκιμαστική έκδοση είναι ατελές, δείχνει 1200 ώρες όταν μόλις 37 ώρες εργασιών έχουν προγραμματιστεί.



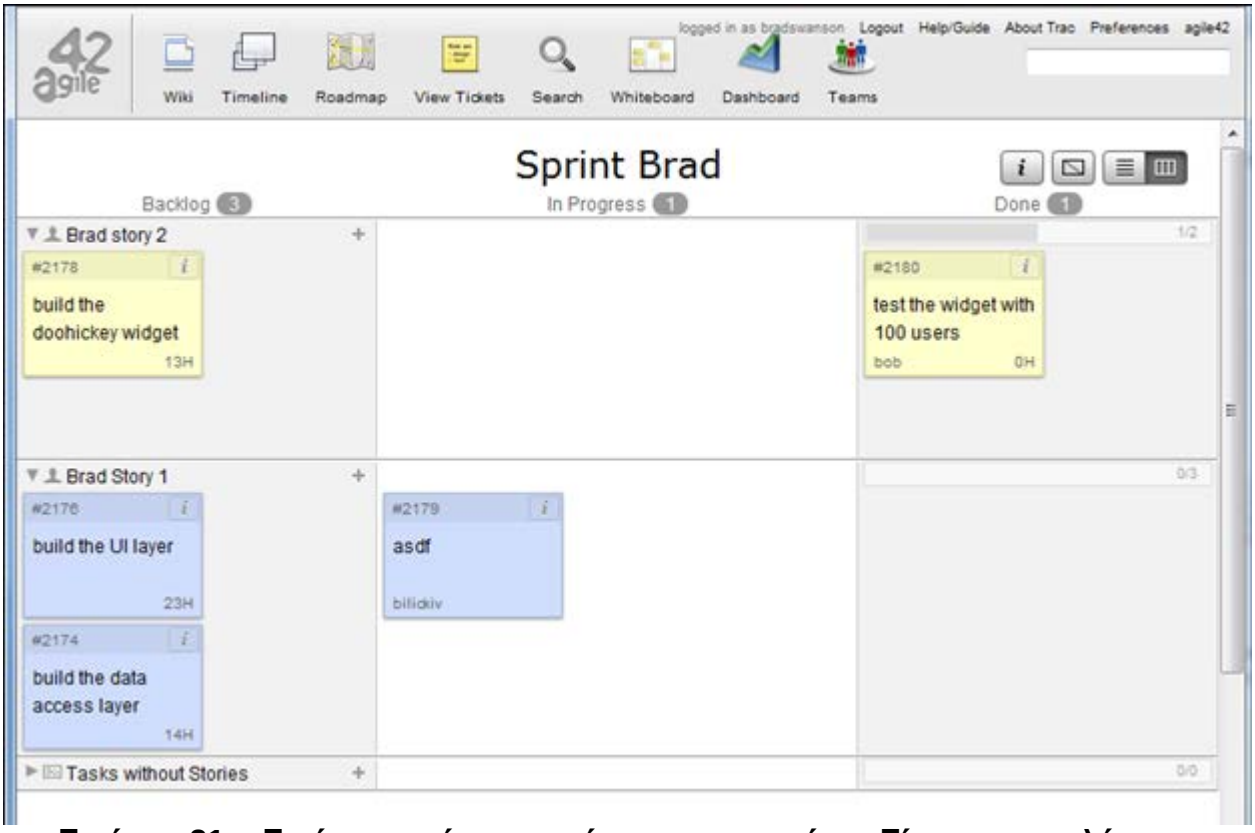
Εικόνα 18: Εικόνα από την Λίστα προϊόντος του Agilo



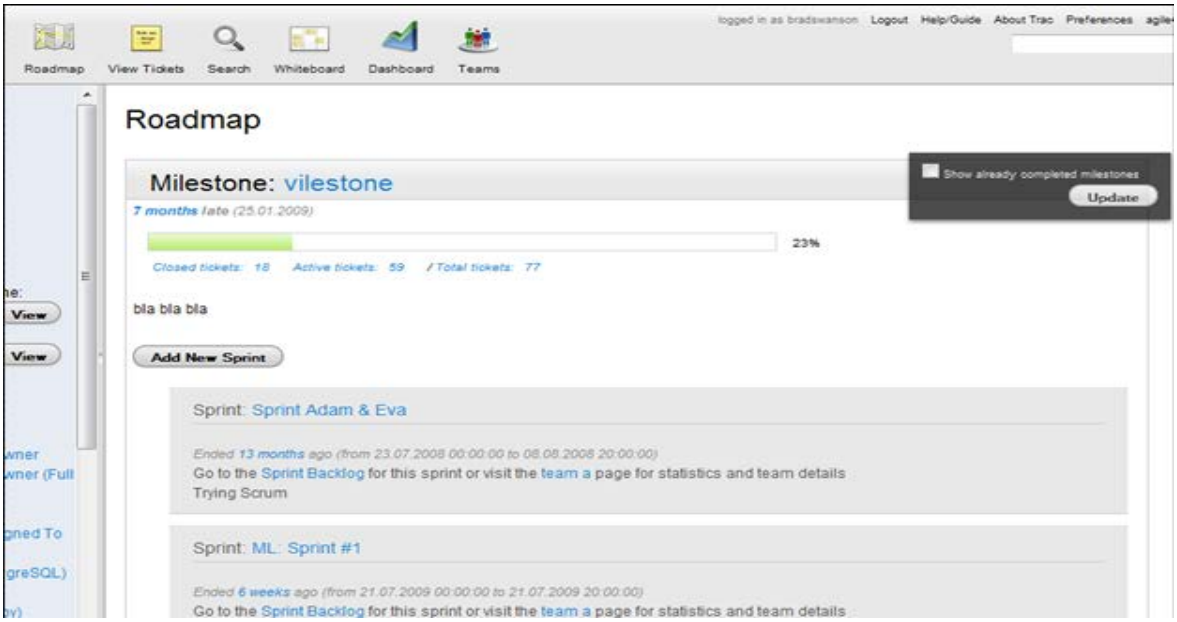
Εικόνα 19: Εικόνα από τη Λίστα Ροής. Αυτή η εικόνα μπερδεύει γιατί οι εργασίες δεν εμφανίζονται κάτω από τα πλάνα τους. Για παράδειγμα η 2174 και 2176 είναι εργασίες του πλάνου 2173.



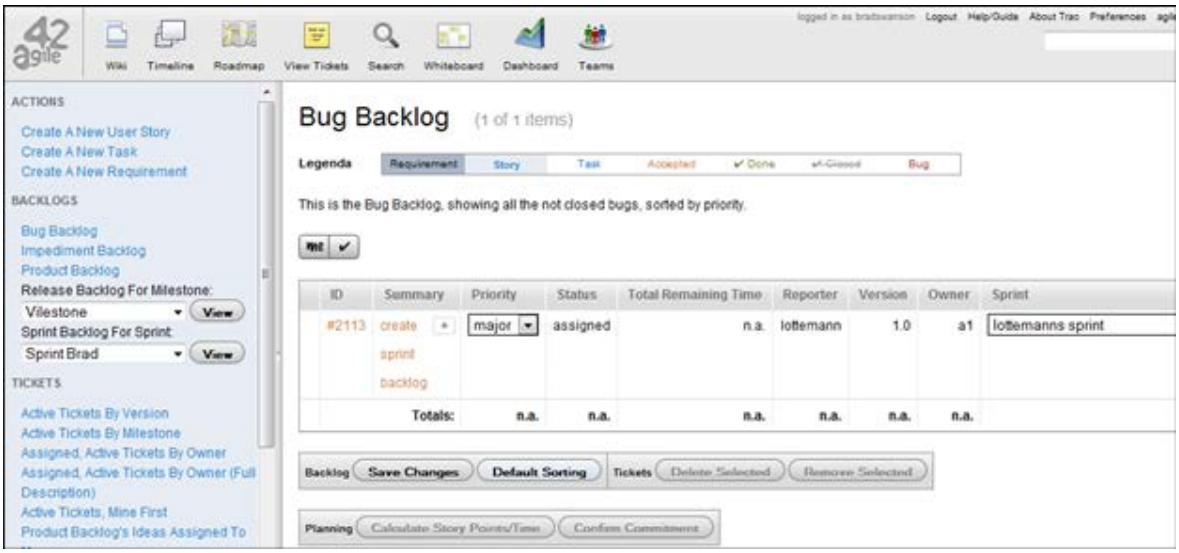
Εικόνα 20: Διάγραμμα Ροής. Αυτό το διάγραμμα είναι λάθος διότι στον οριζόντιο άξονα δείχνει 1200 ώρες όταν μόλις 37 ώρες εργασίας έχουν προγραμματιστεί.



Εικόνα 21: Εικόνα από τον πίνακα εργασιών. Είναι το καλύτερο χαρακτηριστικό αυτού του εργαλείου. Έχει δυνατότητα drag and drop. Για να προσθέσουμε μια εργασίας σε ένα πλάνο με ένα κλικ στο "+"

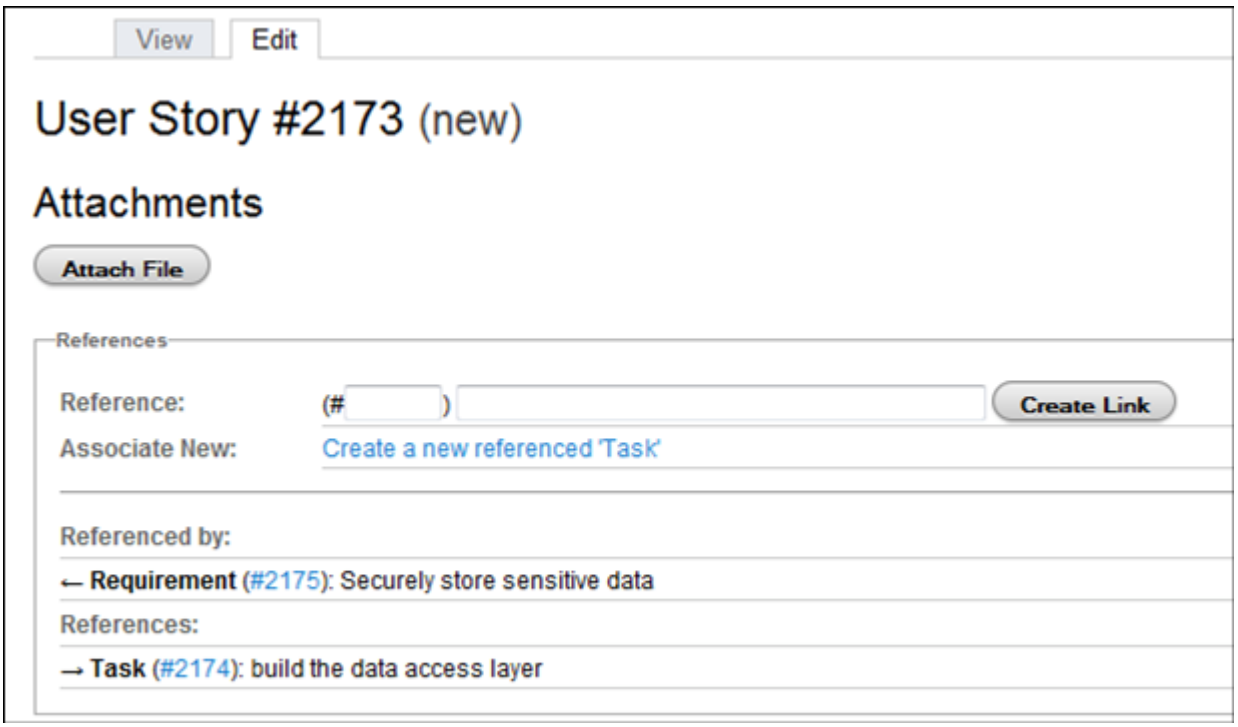


Εικόνα 22: Εικόνα της πορεία. Σε αυτή την εικόνα φαίνονται όλα τα κύρια σημεία και μια σύνοψη της κάθε Ροής του κάθε κύριου σημείου. Με πολλαπλές Ροές ανά κύριο σημείο πάρα πολλές κυλίσεις χρειάζονται για να φανεί η μεγάλη εικόνα πράγμα το οποίο κάνει αυτό το χαρακτηριστικό αναποτελεσματικό.

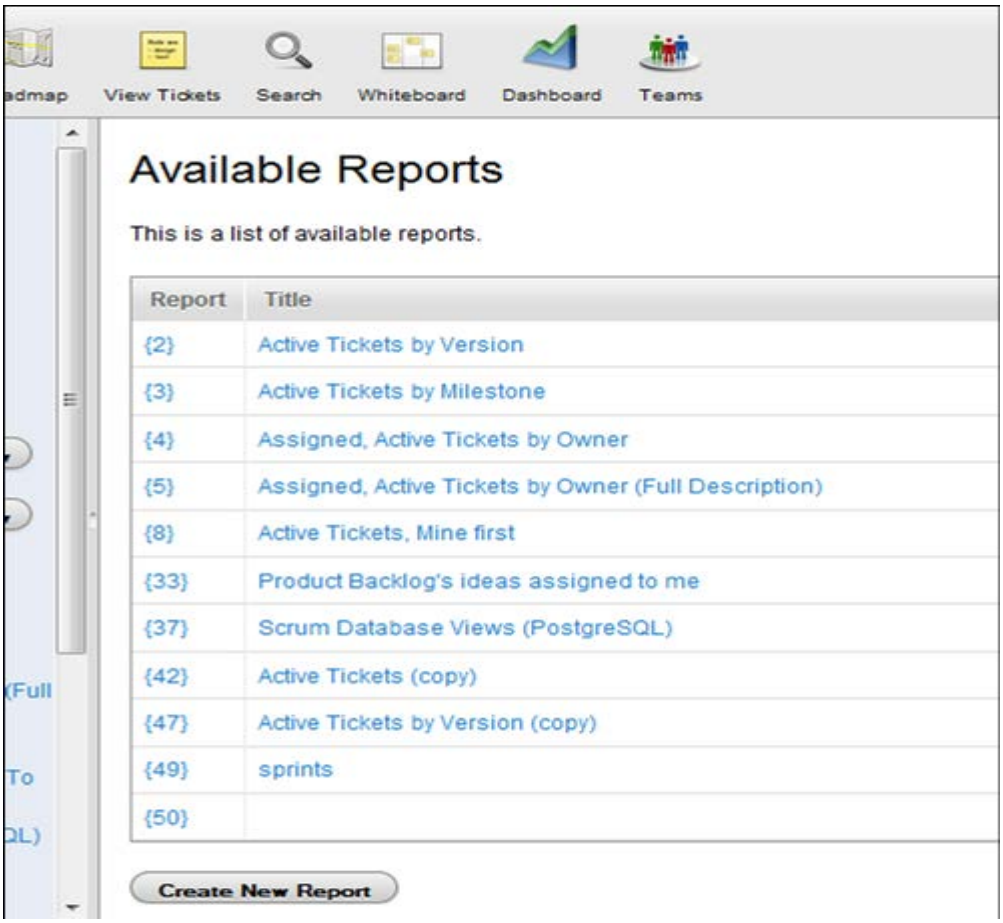


Εικόνα 23: Εικόνα της Λίστας λαθών. Τα λάθη ανιχνεύονται και εμφανίζονται σε μια δική τους ξεχωριστή Λίστα, πράγμα το οποίο είναι μειονέκτημα εφόσον δεν μπορεί να τους δοθεί προτεραιότητα συσχετιζόμενα με τα αντικείμενα άλλης Λίστας του προϊόντος όπως τα πλάνα των χρηστών.

Εικόνα 24: Η εικόνα αυτή μας δείχνει συσχετίσεις απαιτήσεων με πλάνα χρηστών και πλάνα με εργασίες. Αυτό το χαρακτηριστικό δεν είναι εύκολο στη χρήση αφού τα πλάνα πρέπει ξεχωριστά να ψαχτούν με νούμερα, και έπειτα να γίνουν ξεχωριστές καταχωρίσεις σε αυτή την σελίδα.



Εικόνα 25: Αυτή η εικόνα δείχνει ένα πλάνο χρήστη που συσχετίζεται με μια απαίτηση και μια εργασία.



Εικόνα 26: Εικόνα σελίδας αναγραφής στοιχείων, η οποία έχει σε λίστα φτιαγμένες και ορισμένες από το χρήστη αναφορές.

14.eXplainPMT[13]

Στο site δεν αναφέρεται η έκδοση του εργαλείου.

14.1Η Ιδέα(Concepts)

Η μεγαλύτερη δημιουργία είναι το έργο, και το eXplainPMT υποστηρίζει πολλαπλά έργα. Κάθε έργο έχει ένα ή περισσότερες εκδόσεις και επαναλήψεις. Πρέπει να σημειώσουμε ότι οι επαναλήψεις σχετίζονται με τα έργα και όχι με τις εκδόσεις. Ένα έργο έχει μια μόνο Λίστα και η Λίστα αυτή περιλαμβάνει πλάνα. Τα πλάνα έχουν εργασίες και αποδεκτές δοκιμές, και μπορούν να αντιστοιχιστούν σε μια έκδοση. Τα πλάνα μπορούν επίσης να συνδεθούν με μια πρωτοβουλία, η οποία είναι σε κάποιο βαθμό σαν ένα έπος, με την εξαίρεση ότι οι πρωτοβουλίες δεν μπορούν να βαθμολογηθούν ή να ταξινομηθούν. Αιφνιδιαστικά, οι εργασίες δεν έχουν υπολογιζόμενες ή πραγματικές ώρες, έχουν μόνο 2 καταστάσεις: ατελείς ή ολοκληρωμένες.

Οι χρήστες στο eXplainPMT δεν έχουν συγκεκριμένους ρόλους, όλοι οι χρήστες είναι το ίδιο. Οι χρήστες μπορούν να ομαδοποιούνται σε ομάδες.

14.2Βιωσιμότητα, υποστήριξη & τεκμηρίωση(Viability, Support & Documentation)

Το eXplainPMT δεν έχει οδηγίες χρηστών και έχει μόνο ένα σύντομο ενημερωτικό README αρχείο για οδηγό εγκατάστασης. Δεν έχει forum ούτε υποστήριξη μέσο email.

14.3Ευχρηστία(Usability)

Το interface με τον χρήστη είναι απλό και διαισθητικό, αλλά μερικά πάρα πολύ βασικά χαρακτηριστικά λείπουν. Για παράδειγμα μετά την δημιουργία ενός πλάνου μέσα στη Λίστα, αυτό δεν μπορεί να μετακινηθεί σε μια επανάληψη. Επιπλέον το eXplainPMT δεν υποστηρίζει την μέθοδο drag and drop ούτε την εμπειρία των πιο μοντέρνων εφαρμογών περιήγησης στο διαδίκτυο.

14.4Πλεονεκτήματα(Strengths)

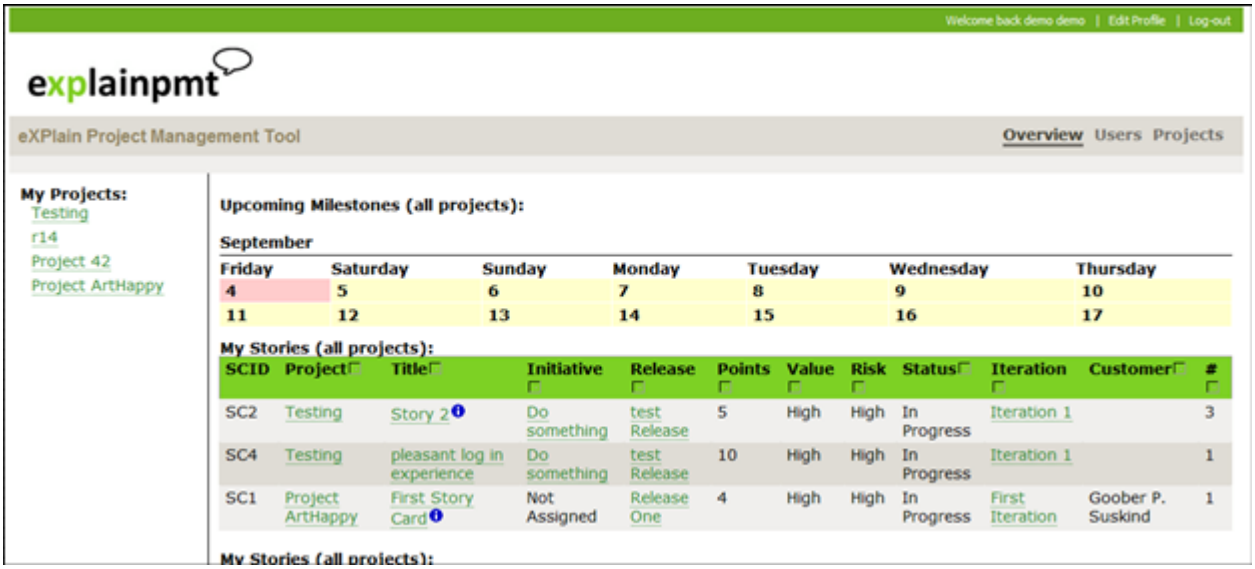
- Διαισθητικό interface
- Υποστηρίζει τον ορισμό της έννοιας αποδεκτές δοκιμές για τα πλάνα.

14.5Αδυναμίες(Weaknesses)

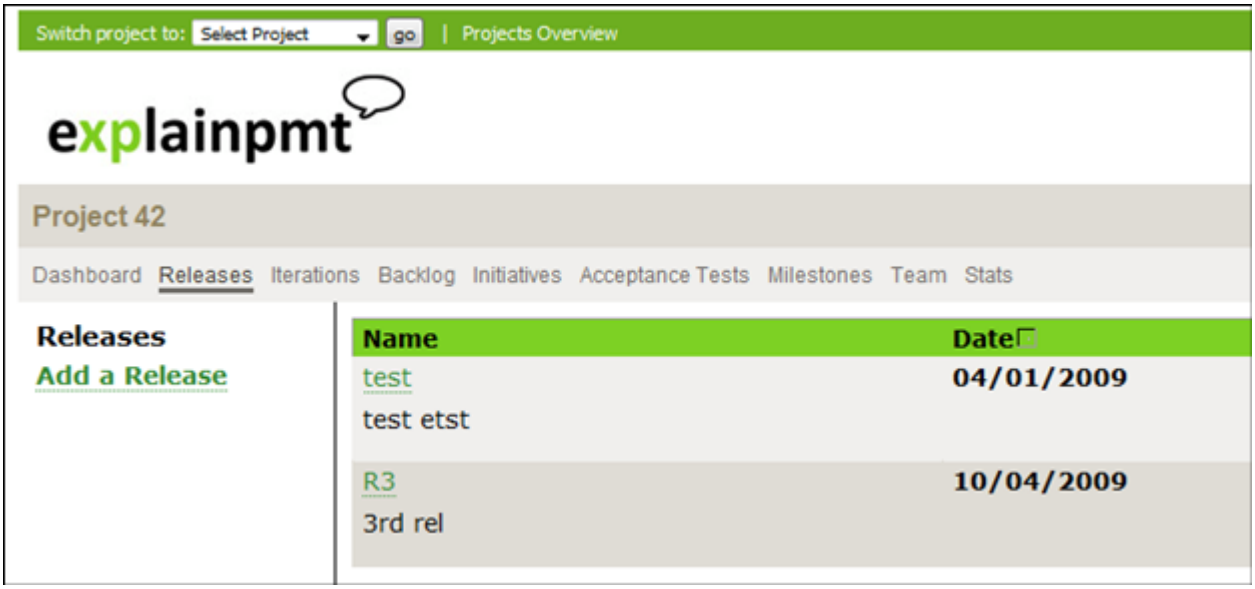
- Δεν μπορεί να υπολογίσει ή να ανιχνεύσει τις ώρες της εργασίας.
- Δεν υπάρχει πίνακας εργασιών(whiteboard)
- Δεν υπάρχει διάγραμμα αναφοράς επαναλήψεων.
- Καμία έκδοση ή πορεία.
- Η υποστήριξη και τα έγγραφα είναι σχεδόν ανύπαρκτα.
- Όχι βολικό για μεγάλα έργα διότι επιτρέπει μόνο μια μοναδική επανάληψη να είναι ενεργή την φορά.

14.6Γενική εκτίμηση(Overall Rating)

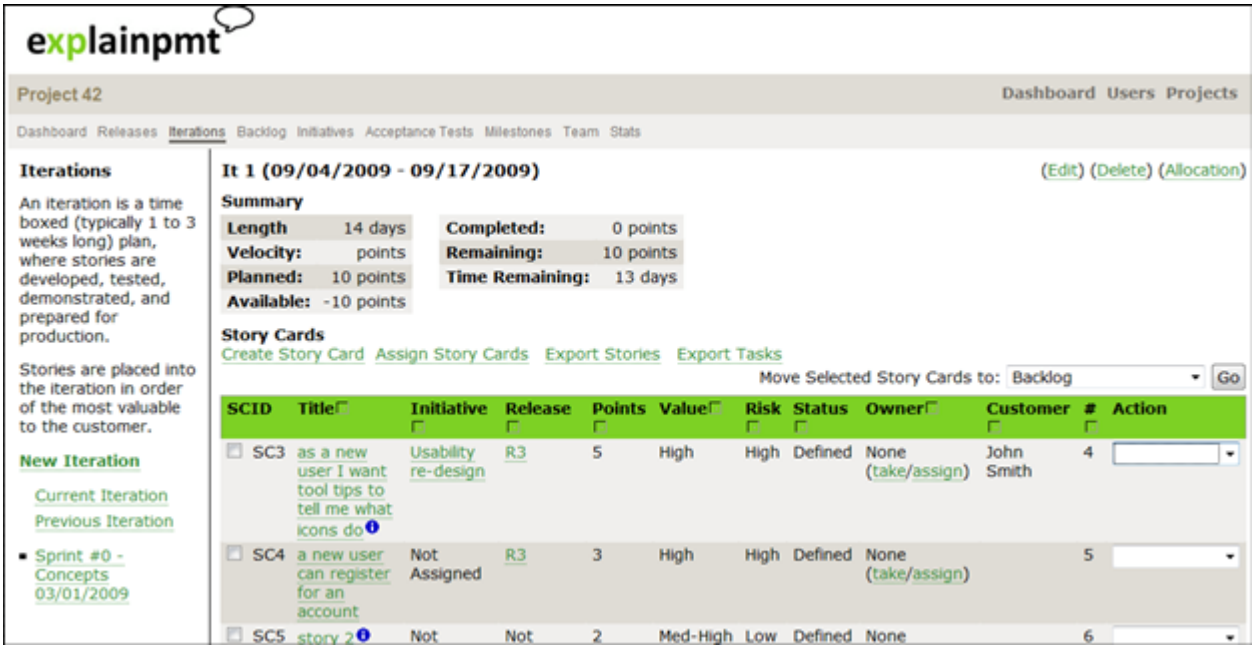
Στο eXplainPMT λείπουν πολλά σημαντικά χαρακτηριστικά, η κατάσταση ανάπτυξης είναι αβέβαιη και η υποστήριξη και η τεκμηρίωση είναι σχεδόν ανύπαρκτη. Δεν υποστηρίζει μεγάλες ομάδες. Το Agilefant, το IceScrum, το Agilo και το XPlanner είναι όλα πιθανών τα καλύτερα εργαλεία για τις περισσότερες ευέλικτες ομάδες.



Εικόνα 27: Εικόνα των έργων του eXplainPMT

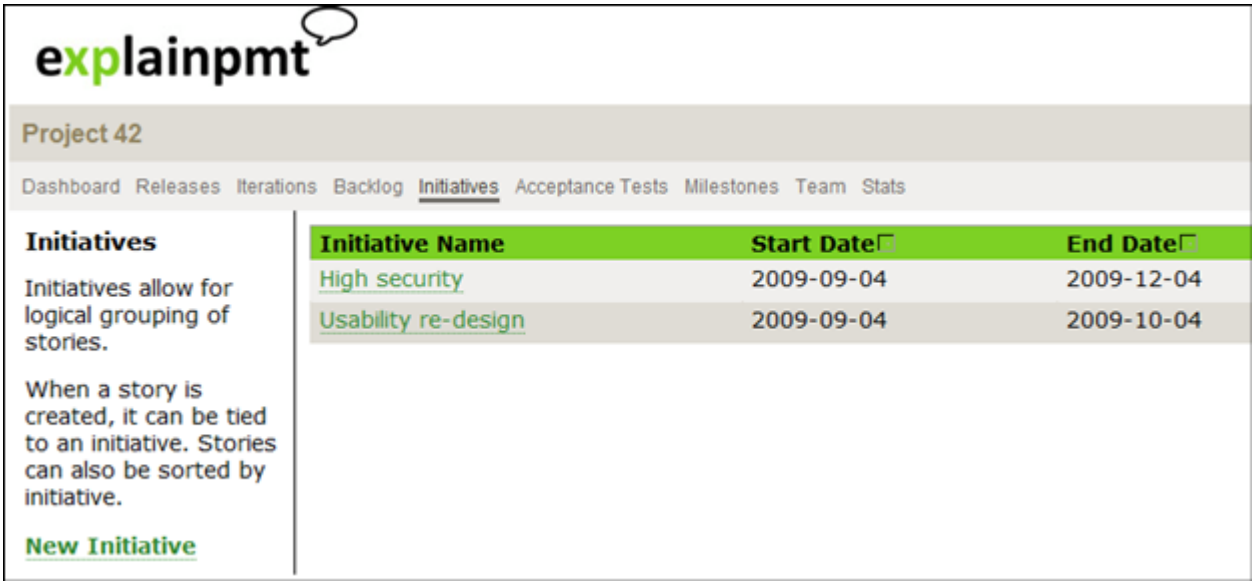


Εικόνα 28: Εικόνα των Εκδόσεων του eXplainPMT

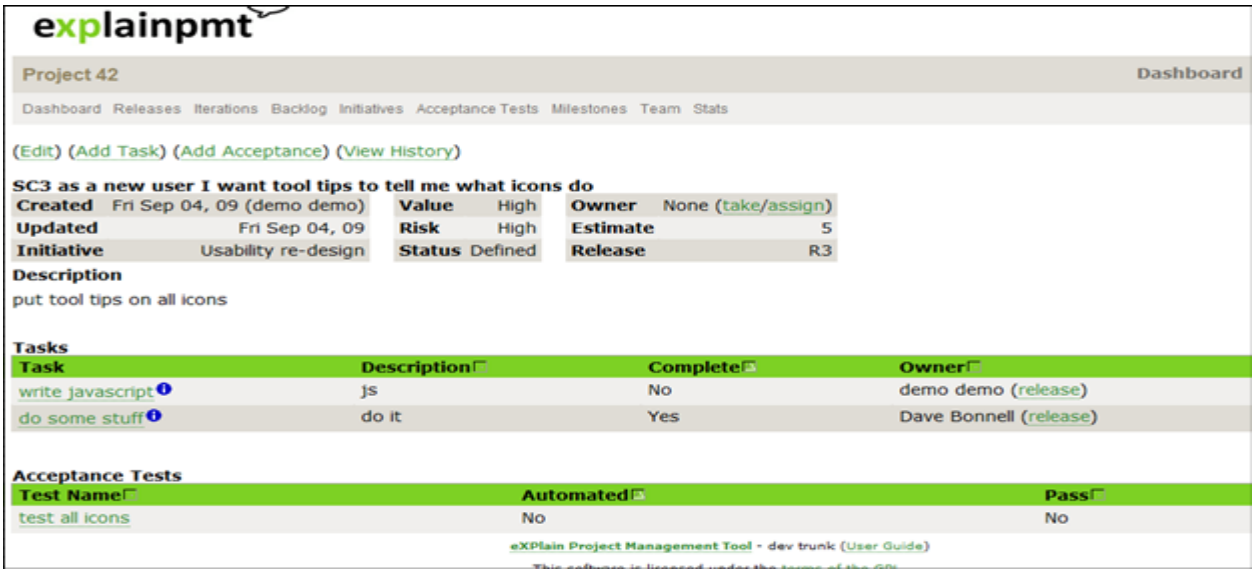


Εικόνα 29: Εικόνα της Επανάληψης

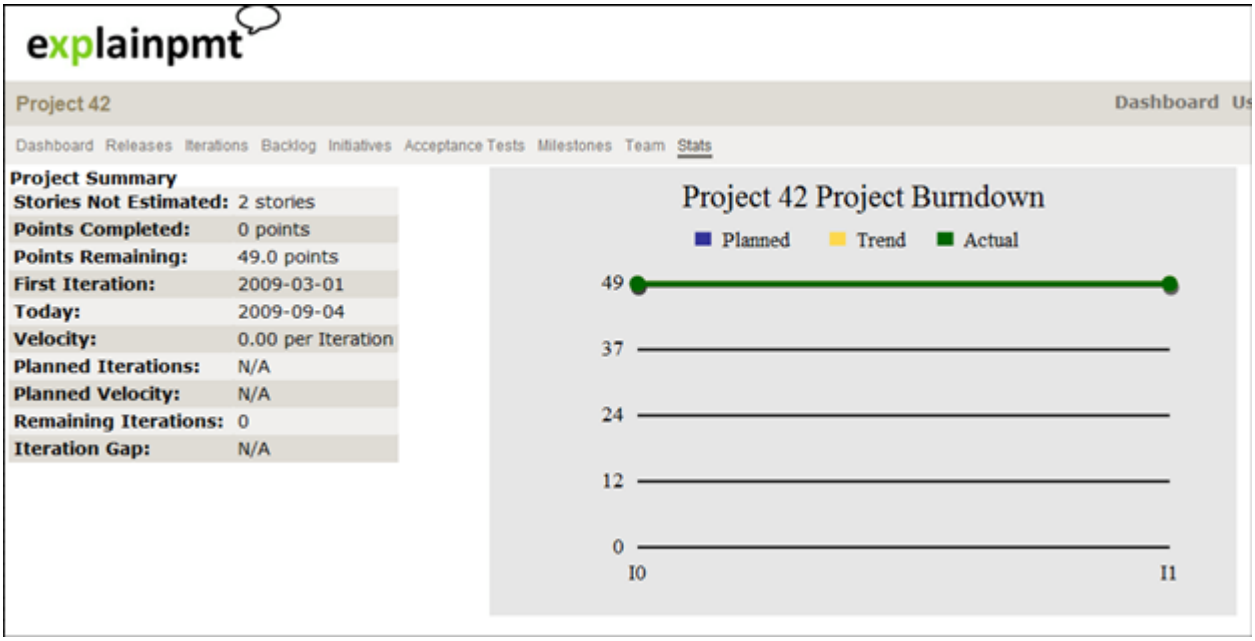
Εικόνα 30: Εικόνα της Λίστας



Εικόνα 31: Εικόνα της πρωτοβουλίας. Οι πρωτοβουλίες είναι λογικά ομαδοποιημένες σαν πλάνα.



Εικόνα 32: Εικόνα του Πλάνου



Εικόνα 33: Η εικόνα αυτή μας δείχνει το διάγραμμα του έργου. Το διάγραμμα, μετριέται σε βαθμούς, είναι για όλα το έργο μόνο και όχι για την επανάληψη.

15.XPlanner[13]

Το εργαλείο αυτό παρόλο που είναι πλέον αδρανή σαν ανάπτυξη συμπεριλαμβάνεται σε αυτή την σύγκριση γιατί κάποια στιγμή είχε μια μεγάλη βάση χρηστών και ήταν το κύριο εργαλείο σε αυτό τον τομέα. Επίσης προσφέρει μια καλή βάση για να δούμε πως εξελίχθηκαν τα διάφορα εργαλεία από τότε.

Τα παρακάτω αναφέρονται για την έκδοση 0,7b (beta) του προγράμματος.

15.1Η Ιδέα(Concepts)

Η μεγαλύτερη δημιουργία είναι το έργο, και το eXplainPMT υποστηρίζει πολλαπλά έργα. Δεν περιλαμβάνει εκδόσεις. Κάθε έργο έχει μια ή περισσότερες επαναλήψεις. Δεν περιέχει Λίστα προϊόντος, αλλά μια επιλογή μας βοηθάει να φτιάξουμε μια ειδική επανάληψη για να χρησιμοποιηθεί σαν Λίστα προϊόντος. Οι επαναλήψεις περιέχουν πλάνα, και τα πλάνα εργασίες. Οι εργασίες μπορεί να είναι τύπου: χαρακτηριστικό, υποχρέωση, ελάττωμα, ftest(functional test, λειτουργική δοκιμή), attest(acceptance test, αποδεκτή δοκιμή), επιβάρυνση. Τα πλάνα υπολογίζονται σε ώρες, και οι εργασίες υπολογίζονται και παρακολουθούνται σε ώρες.

Οι χρήστες στο XPlanner μπορούν να καταχωρηθούν σε 4 επίπεδα δικαιωμάτων: Παρατηρητών(Viewer, μόνο ανάγνωση), Συντάκτη, Διαχειριστή και Super Διαχειριστή. Το XPlanner δεν έχει ένα κατασκεύασμα σαν την ομάδα για να ομαδοποιήσει τους χρήστες.

15.2Βιωσιμότητα, υποστήριξη & τεκμηρίωση(Viability, Support & Documentation)

Το τελευταίο update του εργαλείου ήταν το Μάιο του 2006 οπότε μπορούμε να πούμε με ασφάλεια ότι είναι πλέον αδρανές. Το εργαλείο αυτό έχει forum για υποστήριξη που περιέχουν πολλές χρήσιμες βοήθειες αλλά είναι ανενεργά τώρα πια, σε αυτά τα forum υπήρχαν 2 νέα θέματα ανά τον μήνα με αρκετές απαντήσεις το καθένα. Ο οδηγός εγκατάστασης είναι επαρκείς αν και στην τελευταία έκδοση ήταν μόνο ένα README αρχείο. Τα έγγραφα για τον χρήστη είναι λίγα αλλά το εργαλείο είναι αρκετά διαισθητικό οπότε δεν χρειάζεται και πάρα πολλά έγγραφα.

15.3Ευχρηστία(Usability)

Το interface του XPlanner είναι ξεκάθαρο και διαισθητικό. Ωστόσο υπάρχει έλλειψη από τα δευτερεύοντα λειτουργικά χαρακτηριστικά των μοντέρνων εφαρμογών περιήγησης στο διαδίκτυο.

15.4Πλεονεκτήματα(Strengths)

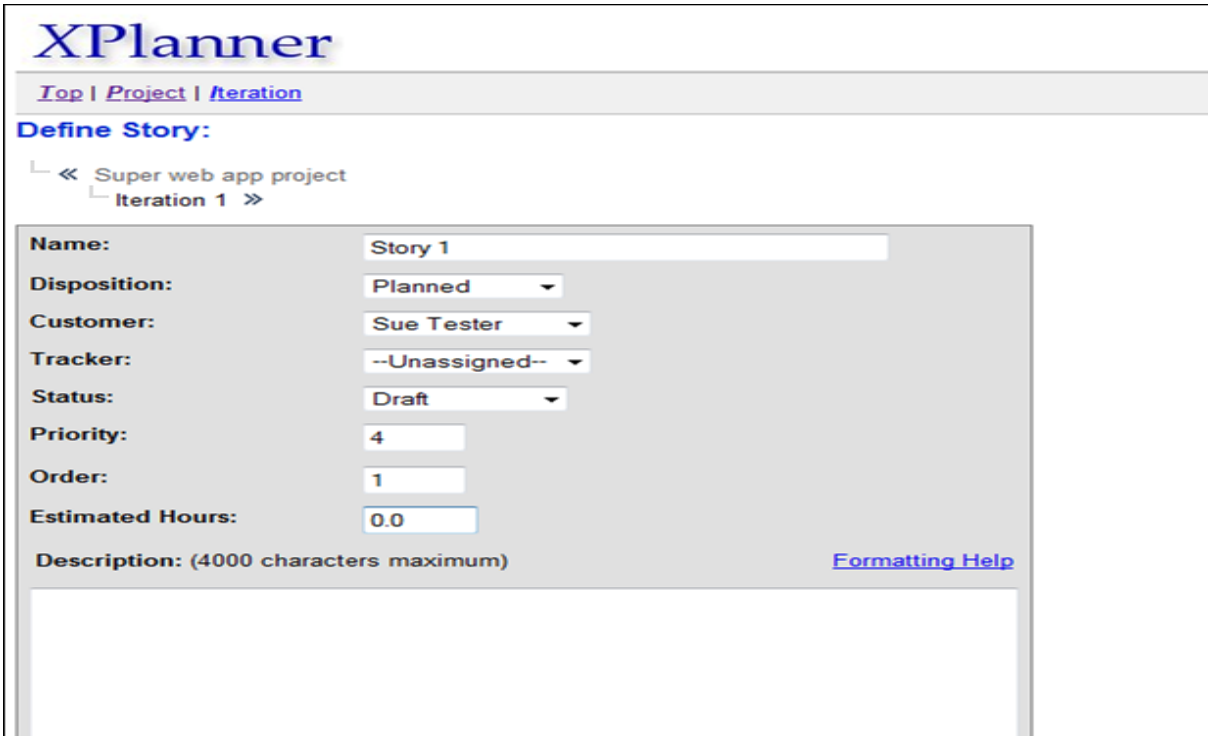
- Διαισθητικό interface
- Πλούσια κατασκευή αναφορών και διαγραμμάτων.
- Προσαρμογή συμβολισμού για απλή σύνδεση URL με εξωτερικά συστήματα όπως ο εντοπισμός ελαττωμάτων.

15.5Αδυναμίες(Weaknesses)

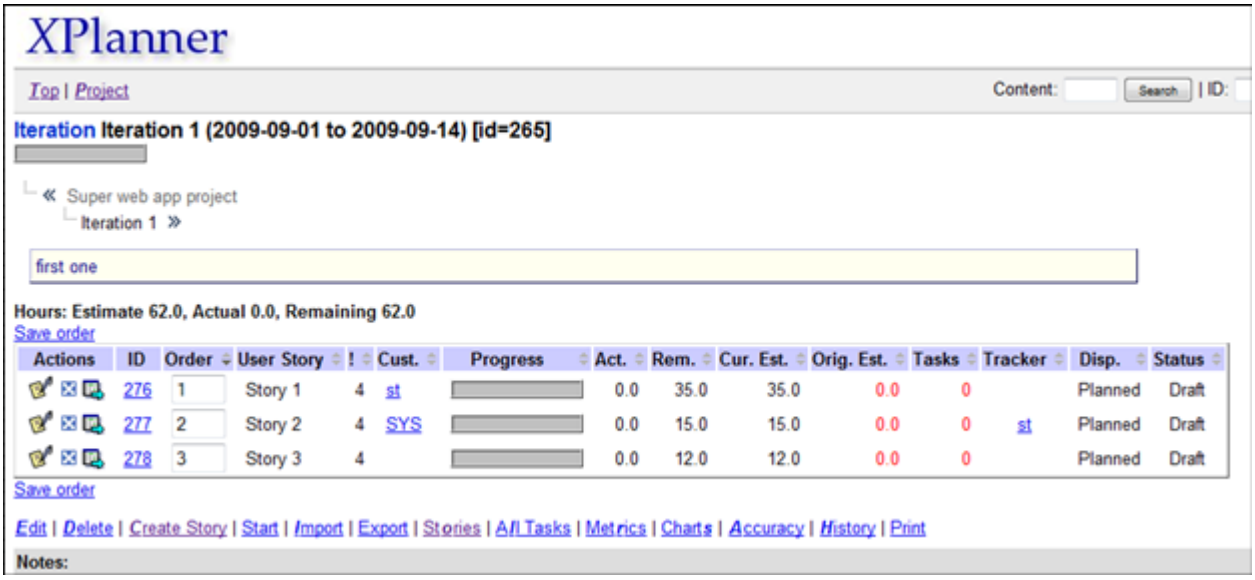
- Καμία έκδοση ή πορεία.
- Κανένας πίνακας εργασιών (whiteboard)
- Κανένα στάδιο του πλάνου.
- Καμία ιεράρχηση των πλάνων, των χαρακτηριστικών ή των απαιτήσεων.
- Το εργαλείο είναι πλέον αδρανές με τελευταίο update τον Μάιο του 2006
- Πολύ περιορισμένη υποστήριξη αφότου το εργαλείο είναι αδρανές.

15.6Γενική εκτίμηση(Overall Rating)

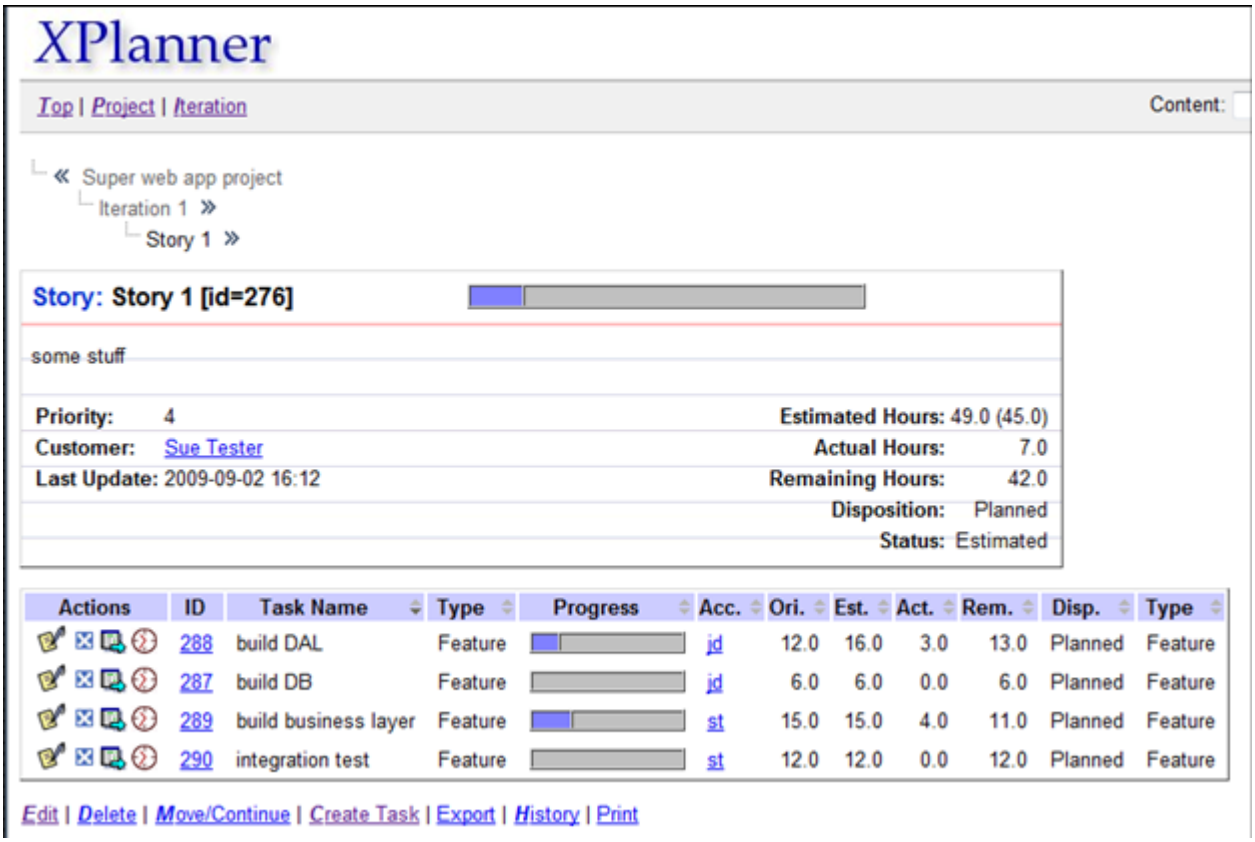
Παρά τις ανεπάρκειες του, το XPlanner είναι ένα καλό εργαλείο για μικρές ομάδες. Παρόλο που η ανάπτυξη αυτού του εργαλείου είναι αδρανείς και άλλα εργαλεία έχουν καλύτερα χαρακτηριστικά τώρα προτιμότερο είναι να χρησιμοποιηθεί κάποιο από τα Agilefant, IceScrum ή Agilo παρά το XPlanner, αλλά προτείνεται να χρησιμοποιηθεί το XPlanner αντί που eXplainPMT για τις περισσότερες ομάδες.



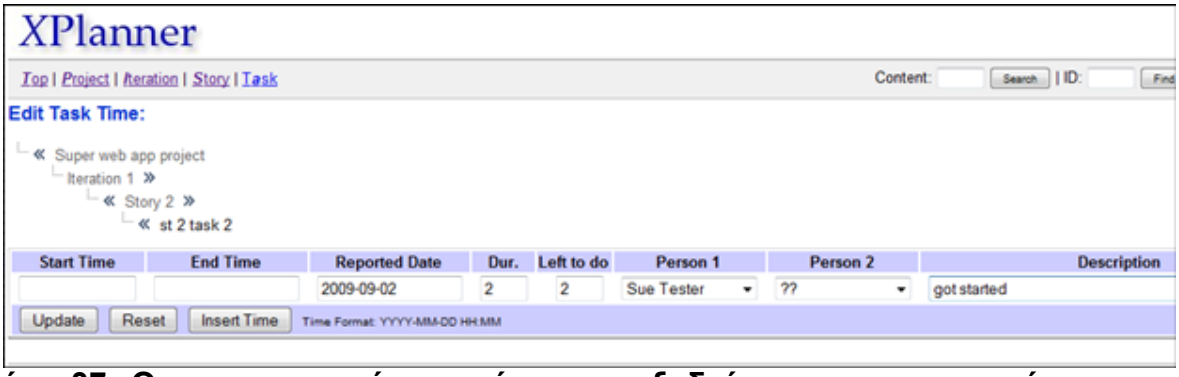
Εικόνα 34: Ορισμός ενός πλάνου στο XPlanner



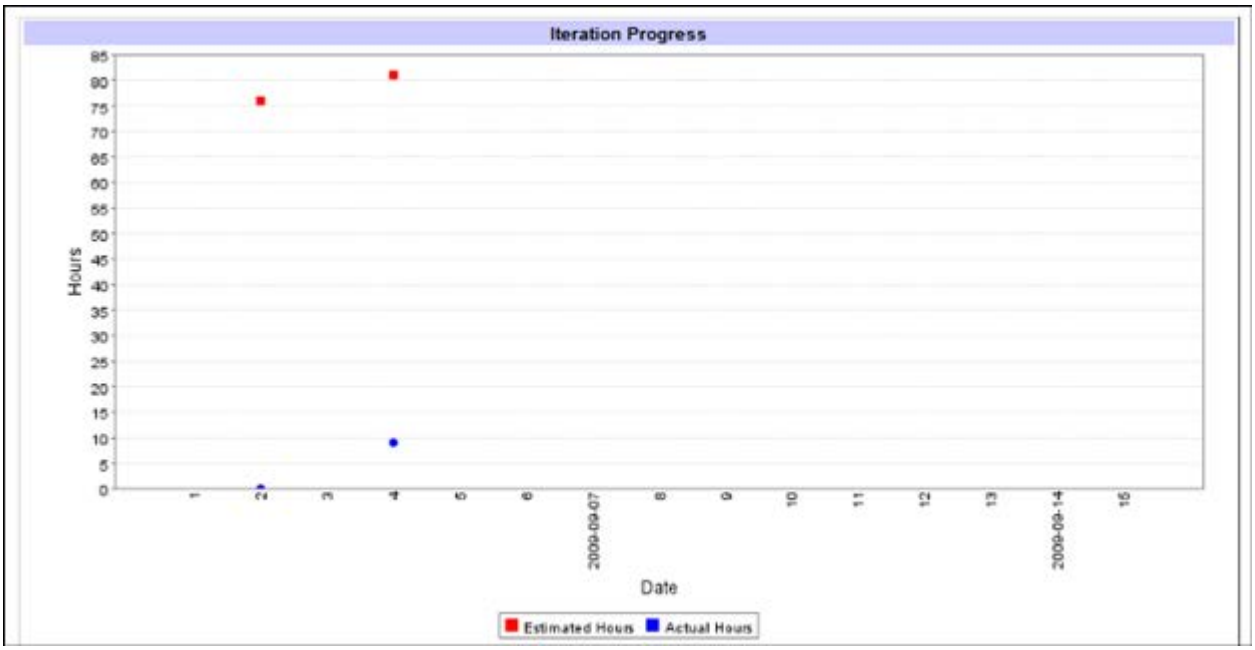
Εικόνα 35: Η εικόνα της επανάληψης στο XPlanner. Τα πλάνα είναι ταξινομημένα χειροκίνητα. Μπορούμε να μετακινήσουμε ένα πλάνο σε μια διαφορετική επανάληψη με το 3^{ο} ενεργό εικονίδιο.



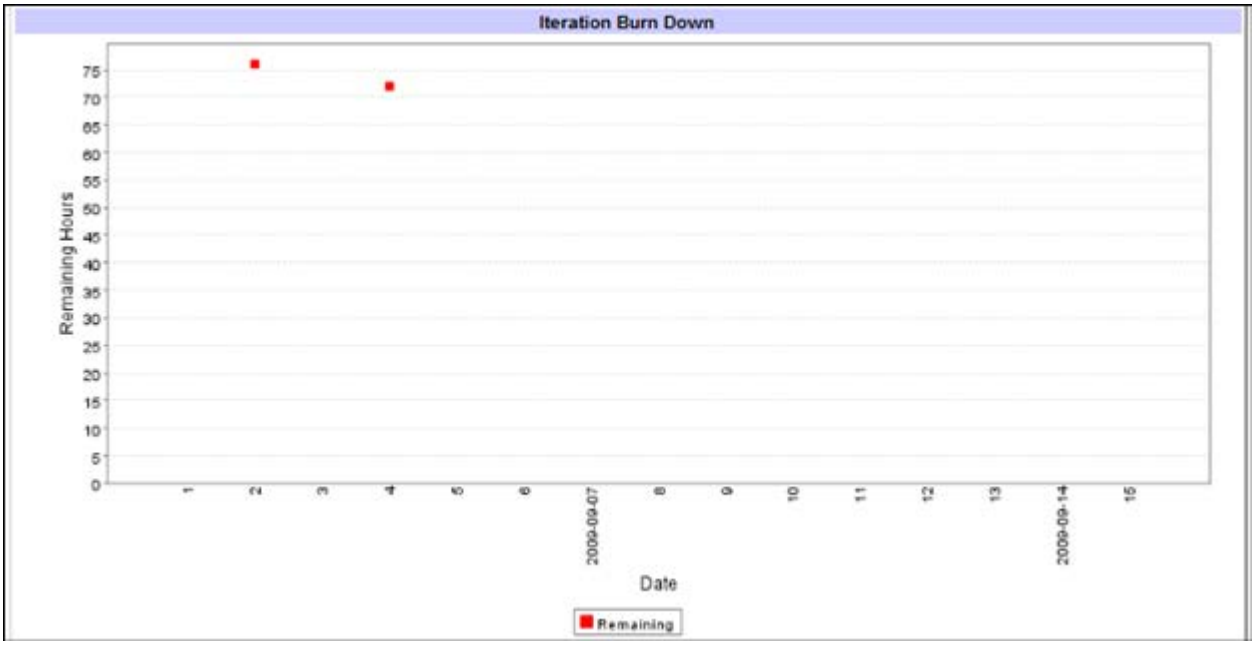
Εικόνα 36: Η εικόνα του Πλάνου



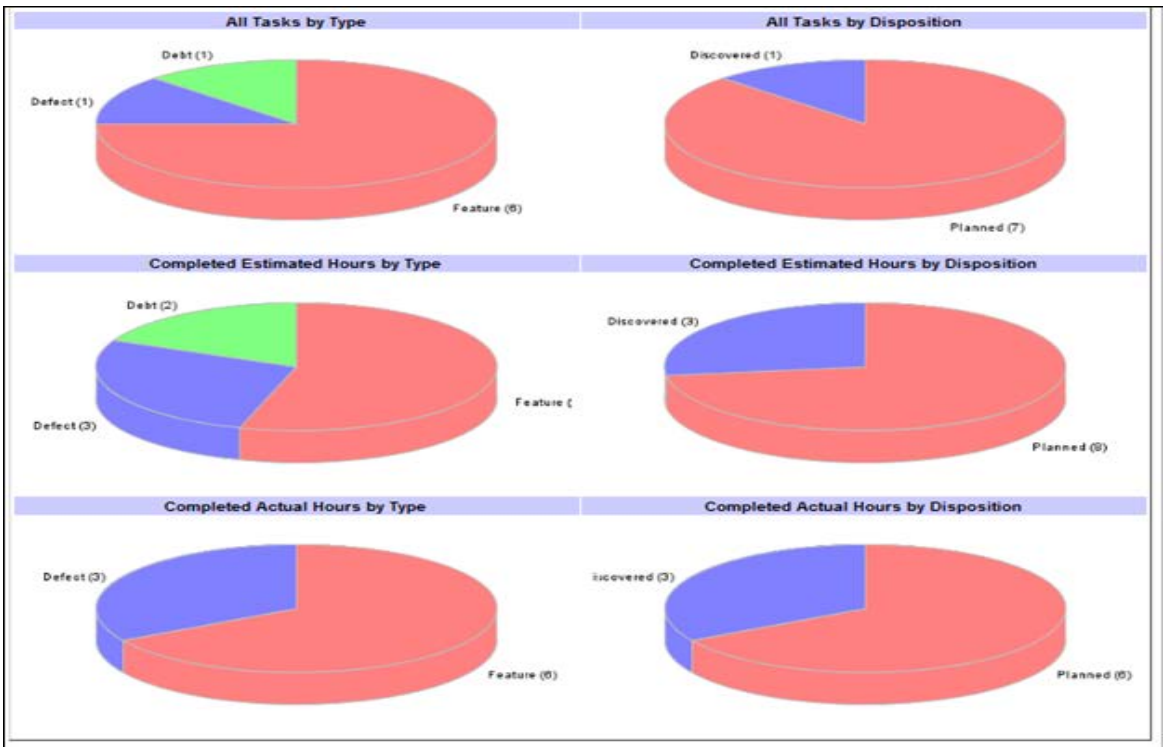
Εικόνα 37: Ο καταγεγραμμένος χρόνος που ξοδεύτηκε σε μια εργασία στο XPlanner



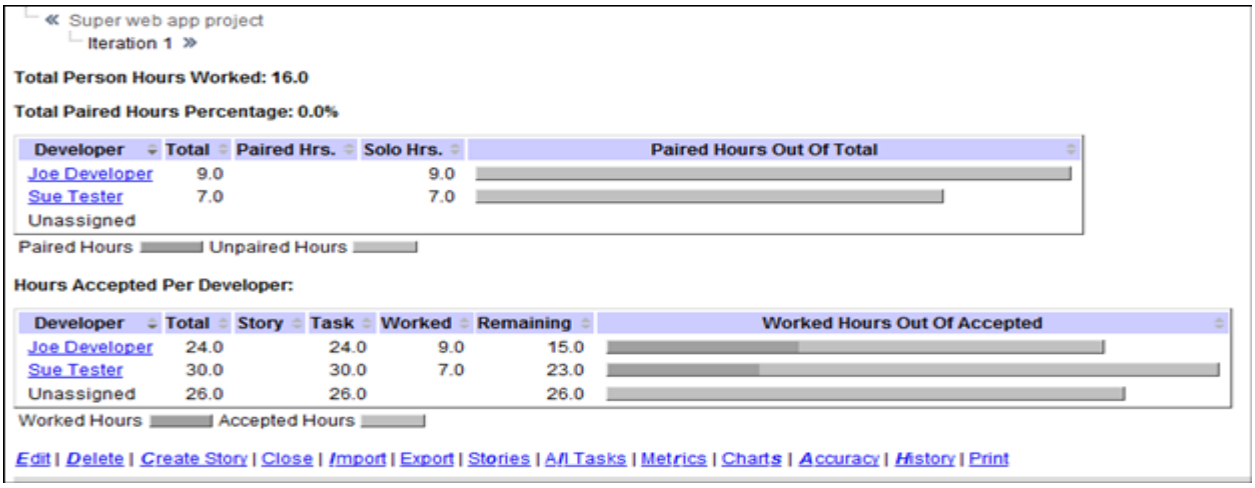
Εικόνα 38: Το διάγραμμα προόδου της επανάληψης δείχνει τον υπολογιζόμενο και πραγματικό χρόνο.



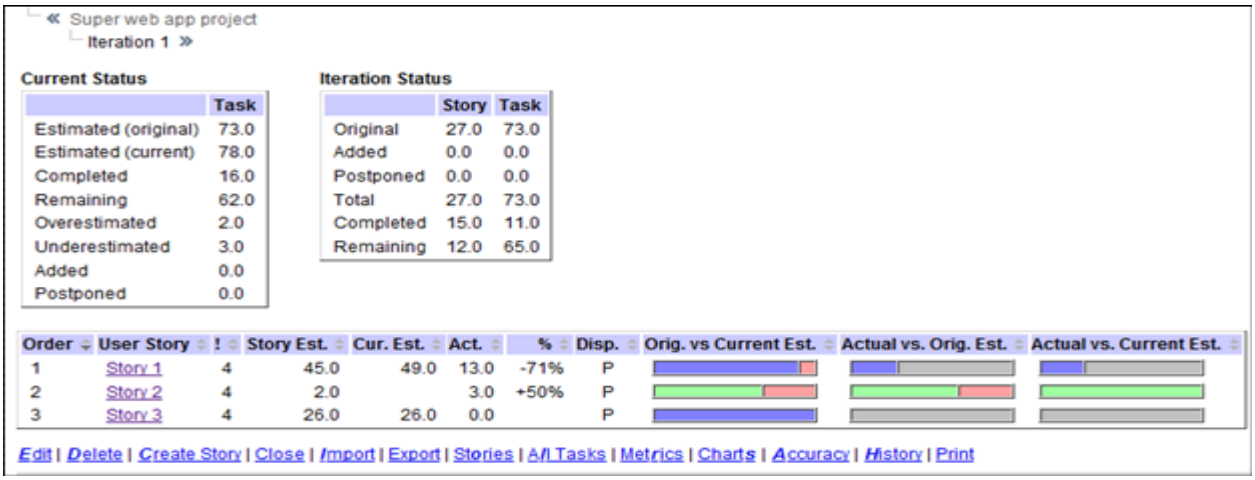
Εικόνα 39: Το διάγραμμα της επανάληψης δείχνει πόσος χρόνος έχει μείνει στην επανάληψη



Εικόνα 40: Μια ποικιλία της κατασκευής διαγραμμάτων στο XPlanner



Εικόνα 41: Η εικόνα των μετρικών του XPlanner



Εικόνα 42: Οι ακριβείς μετρικές του XPlanner, Δείχνουν την ακρίβεια των υπολογισμών.

Τελικό Συμπέρασμα(Conclusion)

Συνολικά βαθμολογήθηκε το Agilefant και το IceScrum με το μεγαλύτερο βαθμό ανάμεσα στα άλλα εργαλεία που αναφέρθηκαν. Το Agilo και το XPlanner απέτυχαν διότι και τα δυο έχουν σημαντικά ελαττώματα αλλά είναι πολύ ικανά εργαλεία όταν το σύνολο των χαρακτηριστικών τους ταιριάζουν με τις ανάγκες σας. Στο eXplainPMT λείπουν πάρα πολλά χαρακτηριστικά κλειδιά έτσι δεν προτείνεται. Παρόλα αυτά οι διάφοροι οργανισμοί πρέπει να σκεφτούν πρώτα την κατάσταση τους και τα χαρακτηριστικά που χρειάζονται πριν επιλέξουν ένα εργαλείο γιατί κάποια χαρακτηριστικά που είναι σημαντικά για έναν οργανισμό μπορεί να μην είναι για κάποιον άλλο.

Επίλογος

Η διασφάλιση της ποιότητας είναι τελικά ένα αναπόσπαστο κομμάτι της ευέλικτης ανάπτυξης το οποίο έχει προέλθει από τις εσωτερικές πρακτικές των διαδικασιών καθώς επίσης και από τις πρακτικές για τον καθορισμό συγκεκριμένων στόχων. Αυτό κάνει την εργασία της διασφάλισης της ποιότητας τόσο διαφορετική με τα ευέλικτα έργα, είναι ο τρόπος με τον οποίο οι στόχοι είναι ορισμένοι και αλλάζουν κατά την διάρκεια του έργου. Διάφοροι στόχοι μπορούν να επιτευχθούν με διαφορετικές πρακτικές, μπορεί όμως μια πρακτική από μόνη της να μην μπορεί να επιτύχει τον δοθέντα στόχο. Από τις πιο σημαντικές πρακτικές για την βελτίωση της διασφάλισης της ποιότητας είναι η ενεργή προσπάθεια για να μπορέσει να επιτευχθεί η επικοινωνία. Οι διάφορες έρευνες μας έχουν δείξει πόσο σημαντική είναι η πληροφορία και ας είναι σε μορφή ηλεκτρονικής αλληλογραφίας, συζητήσεις ακόμα και ο κώδικας από μόνος του. Το σημαντικότερο ρόλο βέβαια τον παίζουν οι απαιτήσεις των χρηστών/συμμετεχόντων που οι ευέλικτες μέθοδοι τους έδωσαν το δικαίωμα να συμμετέχουν ενεργά πλέον στην ανάπτυξη του λογισμικού. Η εργασία του μετασχηματισμού των απαιτήσεων των συμμετεχόντων σε επαρκεί ανθρωποκεντρικά συστήματα πληροφοριών υποστηρίζει τον συνδυασμό δημιουργικής διαίσθησης και αρχών εφαρμοσμένης μηχανικής. Οι ανάγκες των συμμετεχόντων θα αντικατοπτριστούν με φυσικό ομαλό και σαφή τρόπο, εάν ο μετασχηματισμός των προϊόντων βασιστεί στις ευέλικτες μεθόδους που χρησιμεύουν σαν μια επικοινωνιακή πλατφόρμα για κατανόηση και προσφέρουν απόλυτη διασφάλιση ποιότητας.

REFERENCES

[1] **Empirical Studies on Quality in Agile Practices: A Systematic Literature Review** των Π. Σφέτσου και Ι. Στέμελου.

[2] **ISO/IEC 9126 in practice: what do we need to know?** των P. Botella, X. Burgués, J.P. Carvallo, X. Franch, G. Grau, J. Marco, C. Quer

[3] **Applying the ISO 9126 Quality Model to Test Specifications** των Benjamin Zeiss, Diana Vega, Ina Schieferdecker, Helmut Neukirchen, Jens Grabowski

[4] **Extending the ISO/IEC 9126-1 Quality Model with Non-Technical Factors for COTS Components Selection** των Juan Pablo Carvallo, Xavier Franch

[5] **INTERNATIONAL STANDARD ISO/IEC 12207 SOFTWARE LIFE CYCLE PROCESSES** του Raghu Singh

[6] **ISO 12207 and Related Software Life-Cycle Standards** του Jim Moore

[7] **Agile Software Construction** του John Hunt

[8] **Agile Software Development Quality Assurance** του Παναγιώτη Σφέτσου και Ιωάννη Στάμελου

[9] **Extreme Programming Modified: Embrace Requirements Engineering Practices** των Jerzy Nawrocki, Michal Jasinski, Batrosz Walter, Adam Wojciechowski

[10] **Digging into the Fundamentals of Extreme Programming** των Tuomo Kahkonen και Pekka Abrahamsson

[11] **How does agility ensure quality?** των Ming Huo, June Verner, Muhammad Ali Babar, Liming Zhu

[12] **Software Quality and Agile Methods** των Ming Huo, June Verner, Muhammad Ali Babar, Liming Zhu

[13] **http://www.linux.com/news/software/applications**

[14] **http://olex.openlogic.com/wazi/2009/comparing-open-source-agile-project-management-tools/ , Comparing Open Source Agile Project Management Tools** του Brad Swanson

[15] **http://en.wikipedia.org/wiki/Trac**

[16] **Adapting Extreme Programming For A Core Software Engineering Course** των Anuja Shukla και Dr. Laurie Williams

[17] **Project Management and Scrum – A Side by Side Comparison** της Anne Loeser

[18] **Distributed Extreme Programming Development** της Monica Yap

[19] **Value based Extreme Programming** της Monica Yap

[20] **The Social Dynamic of Pair Programming** των Jan Chong και Tom Hurlbutt

[21] **Agile Software Development** του Alan S. Koch