ΣΧΟΛΗ ΟΙΚΟΝΟΜΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ
«ΔΙΟΙΚΗΣΗ, ΤΕΧΝΟΛΟΓΙΑ ΚΑΙ ΠΟΙΟΤΗΤΑ»

ΔΙΑΤΡΙΒΗ ΕΠΙΠΕΔΟΥ ΜΑΣΤΕΡ

ΜΕΘΟΔΟΙ ΑΝΑΠΤΥΞΗΣ ΛΟΓΙΣΜΙΚΟΥ & ΔΙΑΣΦΑΛΙΣΗ ΤΗΣ ΠΟΙΟΤΗΤΑΣ:

Η ΠΕΡΙΠΤΩΣΗ ΤΩΝ ΕΛΛΗΝΙΚΩΝ & ΚΥΠΡΙΑΚΩΝ ΕΤΑΙΡΕΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΙΧΑΗΛ ΞΑΝΘΑΚΗΣ

ΕΠΙΒΛΕΠΟΥΣΑ ΚΑΘΗΓΗΤΡΙΑ

ΙΦΙΓΕΝΕΙΑ ΓΕΩΡΓΙΟΥ

ΛΕΥΚΩΣΙΑ, ΙΑΝΟΥΑΡΙΟΣ, 2018

Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Οικονομικών Επιστημών και Διοίκησης

Μεταπτυχιακό Πρόγραμμα Σπουδών
Διοίκηση, Τεχνολογία και Ποιότητα

Μεταπτυχιακή Διατριβή

Μέθοδοι ανάπτυξης λογισμικού και διασφάλιση της ποιότητας: Η περίπτωση των ελληνικών και κυπριακών εταιρειών πληροφορικής

Μιχαήλ Ξανθάκης

Επιβλέπουσα Καθηγήτρια
Ιφιγένεια Γεωργίου

Ιανουάριος 2018

Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Οικονομικών Επιστημών και Διοίκησης

Μεταπτυχιακό Πρόγραμμα Σπουδών
Διοίκηση, Τεχνολογία και Ποιότητα

Μεταπτυχιακή Διατριβή

Μέθοδοι ανάπτυξης λογισμικού και διασφάλιση της ποιότητας: Η περίπτωση των ελληνικών και κυπριακών εταιρειών πληροφορικής

Μιχαήλ Ξανθάκης

Επιβλέπουσα Καθηγήτρια
Ιφιγένεια Γεωργίου

Η παρούσα μεταπτυχιακή διατριβή υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων για απόκτηση μεταπτυχιακού τίτλου σπουδών
στην Διοίκηση, Τεχνολογία και Ποιότητα
από τη Σχολή Οικονομικών Επιστημών και Διοίκησης
του Ανοικτού Πανεπιστημίου Κύπρου.

Ιανουάριος 2018

Περίληψη

Η ορθή εφαρμογή των μεθόδων ανάπτυξης λογισμικού από τις εταιρείες πληροφορικής μπορεί να οδηγήσει στην πρόοδο, στην ανάπτυξη, καθώς και στην απόκτηση ανταγωνιστικού πλεονεκτήματος. Η χρήση παραδοσιακών μεθόδων ανάπτυξης λογισμικού (π.χ. καταρράκτης, προτυποποίηση, κ.λπ.), προϋποθέτουν την σειριακή ακολουθία φάσεων ανάπτυξης, τον αρχικό καθορισμό των απαιτήσεων και την εκτενή τεκμηρίωση. Τις τελευταίες δεκαετίες, όμως, έχουν αναπτυχθεί μια σειρά από νέα ευέλικτα μοντέλα κύκλου ζωής λογισμικού (π.χ. ακραίος προγραμματισμός, scrum κ.λπ.), τα οποία δίνουν έμφαση στην ομαδική εργασία, στις αλλαγές των απαιτήσεων και στην απλότητα του προγραμματισμού. Από την άλλη, η χρήση προτύπων ποιότητας και η αξιολόγηση του λογισμικού, από τις εταιρείες πληροφορικής, εξασφαλίζει συμμόρφωση προς τις απαιτήσεις, μειώνει τον κίνδυνο σφαλμάτων και συμβάλει στην ικανοποίηση του τελικού χρήστη. Ο σκοπός της παρούσας μεταπτυχιακής διατριβής είναι η διερεύνηση των μεθόδων ανάπτυξης λογισμικού και των πρακτικών διασφάλισης της ποιότητας που χρησιμοποιούνται από εταιρείες πληροφορικής της Ελλάδας και της Κύπρου. Εμπειρικά δεδομένα συλλέχθηκαν από 22 εταιρείες, μέσω της συμπλήρωσης ερωτηματολογίου, που αποτελείται από τρία βασικά σκέλη ερωτήσεων, που αφορούν: τη χρήση των μοντέλων κύκλου ζωής λογισμικού, την επίπτωση τους σε διάφορες πτυχές της εταιρείας, τους τρόπους διασφάλισης της ποιότητας και την μελλοντική κατάσταση του κλάδου ανάπτυξης λογισμικού. Τα αποτελέσματα της έρευνας έδειξαν ότι οι εταιρείες χρησιμοποιούν τόσο παραδοσιακές, όσο και ευέλικτες μεθόδους ανάπτυξης λογισμικού, αναζητώντας την ορθή πρακτική, την οποία ενσωματώνουν στην δομή τους. Η πλειοψηφία των εταιρειών διαθέτει ξεχωριστό τμήμα ποιότητας και εφαρμόζει πρότυπα. Η χρήση ευέλικτων μεθόδων ανάπτυξης λογισμικού βελτιώνει την ποιότητα του λογισμικού και την παραγωγικότητα των εταιρειών, ενώ αυξάνει την ικανοποίηση των τελικών χρηστών και υπό προϋποθέσεις μειώνει το κόστος υλοποίησης. Ο κλάδος ανάπτυξης λογισμικού εμφανίζει θετικές προοπτικές, οι οποίες εν μέρει οφείλονται στην καινοτομία του κλάδου και στην εύκολη διείσδυση σε νέες αγορές.

Λέξεις Κλειδιά: Μέθοδοι Ανάπτυξης Λογισμικού, Διασφάλιση Ποιότητας, Ευέλικτες Μέθοδοι, Κλάδος Εταιρειών Πληροφορικής, Ελλάδα, Κύπρος.

Summary

Software development methodologies used by Greek and Cypriot companies can lead to progress and growth as well as profitability. The use of traditional software development models (waterfall, prototyping, incremental, fountain) requires the sequence of software development steps: planning, programming, resource allocation, workflows, activities, roles, training. In the last few decades, however, many new software lifecycle models (Extreme Programming, Scrum, etc.) have been developed that emphasize teamwork, changes in software development requirements and simple programming. Software quality, on the other hand, is the result of the combination of project management and software technology.

By using software quality assurance, it is probable to produce an infrastructure to support software development methodologies. The aim of the thesis was to examine methods of software development and quality assurance practices used by the software industry sector of Greece and Cyprus.

Empirical data was collected by 22 software development companies in Greece and Cyprus by conducting a questionnaire survey that focused on three main parts: software life cycle models, software quality assurance, including quality standards, current and future state of the development industry software in Greece and Cyprus.

The outcomes of the survey have revealed that the use of agile methods is progressively beginning to gain the confidence of Greek and Cypriot software developers in relation to the use of traditional software development methods. The survey also discovered that many IT companies in Greece and Cyprus use a quality assurance plan in their software development projects. For the future situation of the Greek and Cypriot software industry, most of the respondents believe that despite its weaknesses, the software development industry of Greece and Cyprus will progress in the future.

Key-words: Software Development Methods, Quality Assurance, Agile Software Development Methodologies, IT Companies, Greece, Cyprus.

Ευχαριστίες

Η παρούσα έρευνα δεν θα είχε ολοκληρωθεί χωρίς την συνεχή ενασχόληση, το γνήσιο και ειλικρινές ενδιαφέρον της κ. Ιφιγένειας Γεωργίου, επιβλέπουσας Καθηγήτριας της παρούσας διατριβής. Την ευχαριστώ θερμά, όχι μόνο για την ανάθεση του θέματος, αλλά για την αδιάκοπη επιστημονική της παρουσία, την πολύτιμη και καίρια καθοδήγηση και τη συμπαράσταση της.

Ιδιαίτερες ευχαριστίες θα πρέπει να δοθούν στην κ. Στέλλα Βαρέλη, για την ευγενική προσφορά της να παραχωρήσει τηλεφωνική συνέντευξη, τις εύστοχες παρατηρήσεις της επί του ερωτηματολογίου, καθώς και την προώθησή του σε εταιρείες του κλάδου.

Τελειώνοντας, θα ήθελα να ευχαριστήσω την οικογένειά μου για την αμέριστη κατανόηση, συμπαράσταση και αγάπη τους για τους λόγους αυτούς, η παρούσα μεταπτυχιακή διατριβή αφιερώνεται στην γυναίκα μου Αναστασία-Ελένη και στα παιδιά μου Γιώργο και Γεράσιμο.

Περιεχόμενα

Περίληψη ..... **iv**

Summary ..... **v**

Ευχαριστίες ..... **vi**

Περιεχόμενα ..... **vii**

Πίνακας Εικόνων ..... **xii**

Κεφάλαιο 1 ..... **2**

Εισαγωγή **2**

1.1 Γενικά **2**

1.2 Σκοπός της έρευνας **2**

1.3 Διάρθρωση της μεταπτυχιακής διατριβής **3**

Κεφάλαιο 2 ..... **4**

Μεθοδολογίες Ανάπτυξης Λογισμικού **4**

2.1 Γενικά **4**

2.2 Οι παραδοσιακές μέθοδοι ανάπτυξης λογισμικού **5**

2.2.1 Το μοντέλο του καταρράκτη **6**

2.2.2 Το σπειροειδές μοντέλο **9**

2.2.3 Το μοντέλο της προτυποποίησης **11**

2.2.4 Το μοντέλο της λειτουργικής επαύξησης **13**

2.2.5 Το μοντέλο του πίδακα **14**

2.3 Οι Ευέλικτες Μέθοδοι Ανάπτυξης Λογισμικού 16

2.3.1 Η μέθοδος Scrum 16

2.3.2 Η μέθοδος Ακραίου Προγραμματισμού (Extreme programming - XP) 20

2.3.3 Η μέθοδος ανάπτυξης με βάση τα χαρακτηριστικά (Feature Driven Development) 24

2.3.4 Η μέθοδος ανάπτυξης δυναμικών συστημάτων (Dynamic Systems Development) 26

2.3.5 Η λιτή ανάπτυξη λογισμικού (Lean Software Development) 28

2.3.6 Η μέθοδος του καθαρού κρυστάλλου (Crystal clear) 30

2.3.7 Η ανοικτή ενοποιημένη διεργασία ανάπτυξης λογισμικού (Open Unified Process) 33

2.4 Διαφορές μεταξύ παραδοσιακών και ευέλικτων μεθόδων ανάπτυξης λογισμικού 35

2.5 Η Διασφάλιση της Ποιότητας του Λογισμικού 38

2.5.1 ISO 9001:2015 - Quality Management Systems 38

2.5.2 ISO 27001:2013 - Information Security Management Systems 39

2.5.3 ISO 20000-1:2011 - IT Service Management 41

2.5.4 ISO 22301:2012 - Business Continuity Management Systems 43

2.5.5 ISO 9126 - Software Engineering 44

2.5.6 Οι δοκιμές του λογισμικού 45

Κεφάλαιο 3 48

Μεθοδολογία και Έρευνα 48

3.1 Ορισμός του προβλήματος – Ερευνητικοί Στόχοι 48

3.2 Επιλογή του Ερευνητικού Μοντέλου 49

3.3 Μέθοδος διεξαγωγής της έρευνας 49

3.4 Μέθοδος δειγματοληψίας 50

3.5 Η αξιοπιστία και η εγκυρότητα της έρευνας. 51

3.6 Η δομή του ερωτηματολογίου 51

3.6.1 Η ανάλυση των ερωτήσεων πολλαπλής επιλογής 51

Κεφάλαιο 4 52

Ανάλυση δεδομένων 52

4.1 Ο κλάδος των Ελληνικών και Κυπριακών εταιρειών ανάπτυξης λογισμικού 52

4.2 Η διεξαγωγή της έρευνας 53

4.2.1 Το δείγμα της έρευνας 54

4.2.2 Η μέθοδος συλλογής των δεδομένων 55

4.2.3 Ο χρόνος συλλογής των δεδομένων 56

4.3 Το μοντέλο του κύκλου ζωής των ελληνικών και κυπριακών εταιρειών ανάπτυξης λογισμικού 56

4.3.1 Η χρήση των παραδοσιακών μεθόδων ανάπτυξης λογισμικού 56

4.3.2 Η χρήση των ευέλικτων μεθόδων ανάπτυξης λογισμικού 58

4.3.3 Η συμμετοχή των πελατών στην διαμόρφωση του λογισμικού 59

4.3.4 Το τμήμα διασφάλισης ποιότητας λογισμικού των εταιρειών 60

4.4 Οι ευέλικτες μέθοδοι ανάπτυξης λογισμικού στις ελληνικές και κυπριακές επιχειρήσεις..... 61

4.4.1 Εμπειρία χρήσης των ευέλικτων μεθόδων ανάπτυξης λογισμικού στην Ελλάδα και την Κύπρου..... 61

4.4.2 Επιπτώσεις από την χρήση ευέλικτων μεθόδων ανάπτυξης λογισμικού 63

4.4.3 Χαρακτηριστικά των ευέλικτων μεθόδων ανάπτυξης λογισμικού..... 66

4.5 Η διασφάλιση της ποιότητας λογισμικού στις ελληνικές και κυπριακές εταιρείες πληροφορικής..... 67

4.5.1 Πρότυπα διασφάλισης της ποιότητας λογισμικού που εφαρμόζονται στις ελληνικές και κυπριακές επιχειρήσεις 67

4.6 Μελλοντική κατάσταση των ελληνικών και κυπριακών εταιρειών ανάπτυξης λογισμικού..... 73

Κεφάλαιο 5 75

Συμπεράσματα - Συζήτηση 75

5.1 Γενικά..... 75

5.2 Χρήση μοντέλων κύκλου ζωής λογισμικού και χρησιμοποιούμενες μέθοδοι στη Ελλάδα και την Κύπρο..... 76

5.3 Επίδραση των ευέλικτων μεθόδων στον κύκλο ζωής του λογισμικού .. 78

5.4 Μελλοντική κατάσταση του κλάδου εταιρειών ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου..... 79

Κεφάλαιο 6 81

Επίλογος..... 81

Βιβλιογραφία 91

Πίνακας Εικόνων

Εικόνα 1. Η φιλοσοφία του μοντέλου του Καταρράκτη (Petersen, et al., 2009).....6

Εικόνα 2. Η φιλοσοφία του σπειροειδούς μοντέλου (Boehm, 1986).....9

Εικόνα 3. Η φιλοσοφία του μοντέλου της προτυποποίησης (Βεσκούκης, 2015). ..12

Εικόνα 4. Η φιλοσοφία του μοντέλου της λειτουργικής επαύξησης (Βεσκούκης, 2015).....13

Εικόνα 5. Η φιλοσοφία του μοντέλου του πίδακα (Βεσκούκης, 2015)**.**.....15

Εικόνα 6. Η φιλοσοφία της ευέλικτης μεθόδου Scrum (Πηγή:Aequor technology). 16

Εικόνα 7. Η φιλοσοφία της ευέλικτης μεθόδου του ακραίου προγραμματισμού (Beck, 2004).....23

Εικόνα 8. Η μέθοδος ανάπτυξης με βάση τα χαρακτηριστικά (Palmer & Felsing, 2002).....25

Εικόνα 9. Η μέθοδος ανάπτυξης δυναμικών συστημάτων (Stapleton, 1997).....27

Εικόνα 10. Διαφορές μεταξύ παραδοσιακών και ευέλικτων μεθόδων ανάπτυξης λογισμικού (Πηγή: Helsinki University of Technology). 37

Εικόνα 11. Το μοντέλο V, που χρησιμοποιείται για δοκιμές λογισμικού (Πηγή: softwareengineering.gr).....46

Εικόνα 12. Ποσοστά των εταιρειών του δείγματος όσον αφορά τον αριθμό των εργαζομένων σε αυτές. 54

Εικόνα 13. Η χρήση των παραδοσιακών μεθόδων ανάπτυξη λογισμικού στις εταιρείες του δείγματος. 57

Εικόνα 14. Η χρήση των παραδοσιακών μεθόδων ανάπτυξης λογισμικού σε Ελλάδα και Κύπρο.....57

Εικόνα 15. Η χρήση των ευέλικτων μεθόδων ανάπτυξης λογισμικού στις εταιρείες της Ελλάδας και της Κύπρου.....58

Εικόνα 16. Η χρήση των ευέλικτων μεθόδων ανάπτυξη λογισμικού σε Ελλάδα και Κύπρο. 59

Εικόνα 17. Συμμετοχή των πελατών στην διαδικασία ανάπτυξης λογισμικού. 60

Εικόνα 18. Ποσοστό απαντήσεων στο αν υπάρχει ξεχωριστό τμήμα ποιότητας στην εταιρεία 61

Εικόνα 19. Χρόνος υιοθέτησης ευέλικτων μεθόδων ανάπτυξης λογισμικού από τις εταιρείες. 62

Εικόνα 20. Εμπειρία στην εφαρμογή ευέλικτων μεθόδων ανάπτυξης λογισμικού των ερωτηθέντων. 62

Εικόνα 21. Επιπτώσεις της υιοθέτησης ευέλικτων μεθόδων ανάπτυξης λογισμικού στην παραγωγικότητα της εταιρείας 63

Εικόνα 22. Επιπτώσεις της υιοθέτησης ευέλικτων μεθόδων ανάπτυξης λογισμικού στην ποιότητα λογισμικού. 64

Εικόνα 23. Επιπτώσεις της υιοθέτησης ευέλικτων μεθόδων ανάπτυξης λογισμικού στο κόστος λογισμικού. 65

Εικόνα 24. Επιπτώσεις της υιοθέτησης ευέλικτων μεθόδων ανάπτυξης λογισμικού στην ικανοποίηση των πελατών σχετικά με το τελικό προϊόν. 66

Εικόνα 25. Αναγκαία χαρακτηριστικά των ευέλικτων μεθόδων ανάπτυξης λογισμικού. 67

Εικόνα 26. Λόγοι υιοθέτησης κάποιου πρότυπου διασφάλισης της ποιότητας από ελληνικές και κυπριακές εταιρείες ανάπτυξης λογισμικού. 68

Εικόνα 27. Πρότυπα διασφάλισης ποιότητας που εφαρμόζονται από ελληνικές και κυπριακές εταιρείες ανάπτυξης λογισμικού. 69

Εικόνα 28. Πρότυπα διασφάλισης ποιότητας που εφαρμόζονται στην Ελλάδα και την Κύπρο. 69

Εικόνα 29. Τα σημαντικότερα χαρακτηριστικά ποιότητας λογισμικού από ελληνικές και κυπριακές εταιρείες πληροφορικής 70

Εικόνα 30. Η επιρροή που έχει η υιοθέτηση σύγχρονων μεθόδων ανάπτυξης λογισμικού στην ποιότητα του. 71

Εικόνα 31. Ποσοστά εταιρειών που κάνουν χρήση διαφόρων μετρητών ποιότητας λογισμικού. 72

Εικόνα 32. Ποσοστά εταιρειών που κάνουν χρήση είτε αυτόματων, είτε χειροκίνητων μεθόδων μέτρησης της ποιότητας λογισμικού. 72

Εικόνα 33. Ποσοστά απαντήσεων σχετικά με την μελλοντική κατάσταση του κλάδου εταιρειών ανάπτυξης λογισμικού. 73

Εικόνα 34. Ποσοστά απαντήσεων σχετικά με τον λόγο που οι εταιρείες ανάπτυξης λογισμικού έχουν πλεονέκτημα στην αγορά εργασίας. 74

Κεφάλαιο 1
Εισαγωγή

1.1 Γενικά

«*Όλα τα προϊόντα λογισμικού θα πρέπει να παράγονται κάνοντας χρήση κάποιου είδους μεθοδολογίας*». Όταν μικρά και μεγάλα κομμάτια λογισμικού μιας επιχείρησης αναπτύσσονται με κάποια μεθοδολογία, μπορεί να οδηγήσουν στην πρόοδο και στην ανάπτυξη αυτής (Munassar & Govardhan, 2011). Η ποιότητα λογισμικού είναι το αποτέλεσμα του συνδυασμού της διαχείρισης έργων και της τεχνολογίας λογισμικού. Με τη χρήση της διασφάλισης ποιότητας λογισμικού είναι δυνατόν να δημιουργηθεί μία υποδομή για την υποστήριξη μεθόδων ανάπτυξης λογισμικού, διαχείρισης έργων πληροφορικής και ενεργειών ελέγχου ποιότητας (Pressman, 2005).

1.2 Σκοπός της έρευνας

Ο σκοπός της παρούσας μεταπτυχιακής διατριβής είναι η μελέτη μεθόδων ανάπτυξης λογισμικού και η διασφάλιση της ποιότητας του, σε εταιρείες ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου. Η έρευνα επικεντρώνεται στα μοντέλα κύκλου ζωής λογισμικού και στις μεθοδολογίες ανάπτυξής τους, που χρησιμοποιούνται στις δύο χώρες, καθώς και πώς επηρεάζει η χρήση ευέλικτων μεθόδων (agile methods) τα έργα λογισμικού. Όσον αφορά την διασφάλιση της ποιότητας λογισμικού, ιδιαίτερο βάρος δίνεται στην εφαρμογή προτύπων ποιότητας και στην αξιολόγηση του λογισμικού. Μέσα από την παρούσα διατριβή περιγράφεται συνοπτικά, η κατάσταση του κλάδου πληροφορικής της Ελλάδας και της Κύπρου, με επίκεντρο την ανάπτυξη λογισμικού, δίνοντας παράλληλα έμφαση στα πλεονεκτήματα, μειονεκτήματα και τις αδυναμίες του κλάδου. Η μελέτη αυτή διεξήχθη στη Ελλάδα και την Κύπρο και όλα τα εμπειρικά δεδομένα

συγκεντρώθηκαν από εταιρείες που βρίσκονται σε αυτές τις χώρες. Τα δεδομένα προέκυψαν κατόπιν έρευνας που διεξήχθη με την συμπλήρωση ερωτηματολογίου από εταιρείες ανάπτυξης λογισμικού.

Προκύπτουν τρία ερευνητικά ερωτήματα που αυτή η μεταπτυχιακή διατριβή θα προσπαθήσει να απαντήσει:

1. Ποιες μέθοδοι ανάπτυξης λογισμικού και πρότυπα διασφάλισης της ποιότητας χρησιμοποιούνται από εταιρείες ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου κατά τη διάρκεια του κύκλου ζωής του λογισμικού;
2. Πώς η χρήση των ευέλικτων μεθόδων ανάπτυξης λογισμικού επηρεάζει τον κύκλο ζωής του στην Ελλάδα και την Κύπρο;
3. Πώς αναμένεται να διαμορφωθεί η μελλοντική κατάσταση του κλάδου εταιρειών ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου;

1.3 Διάρθρωση της μεταπτυχιακής διατριβής

Η παρούσα μεταπτυχιακή διατριβή περιλαμβάνει έξι κεφάλαια και ένα παράρτημα. Το πρώτο κεφάλαιο αναφέρεται στην εισαγωγή του προβλήματος, τα ερευνητικά ερωτήματα και το σκοπό της έρευνας. Στο δεύτερο κεφάλαιο, αναφέρεται στο θεωρητικό μέρος της μεταπτυχιακής διατριβής, καθώς γίνεται αναφορά στις κυριότερες παραδοσιακές μεθοδολογίες ανάπτυξης λογισμικού, στις κυριότερες νέες ευέλικτες μεθόδους καθώς και στα κυριότερα πρότυπα διασφάλισης λογισμικού, που χρησιμοποιούνται από τις εταιρείες του κλάδου. Το τρίτο κεφάλαιο παρουσιάζει τη μεθοδολογία της έρευνας, την ανάλυση του δείγματος, την μέθοδο δειγματοληψίας που χρησιμοποιήθηκε στην παρούσα μεταπτυχιακή διατριβή. Το τέταρτο κεφάλαιο περιλαμβάνει την ανάλυση των εμπειρικών αποτελεσμάτων της έρευνας. Το πέμπτο κεφάλαιο αναλύει τα συμπεράσματα της μελέτης και στο έκτο κεφάλαιο δίνεται ο επίλογος της μεταπτυχιακής διατριβής. Στο παράρτημα δίνεται το ερωτηματολόγιο που χρησιμοποιήθηκε για την συλλογή των εμπειρικών δεδομένων.

Κεφάλαιο 2
Μεθοδολογίες Ανάπτυξης Λογισμικού

2.1 Γενικά

Σύμφωνα με τους Munassar & Govardhan, (2011) όλα τα προϊόντα λογισμικού πρέπει να αναπτύσσονται χρησιμοποιώντας κάποια μεθοδολογία. Με τον όρο μεθοδολογία εννοείται ένας συστηματικός τρόπος προσέγγισης των διαδικασιών ανάπτυξης από τα πρώτα στάδια της ανάπτυξης λογισμικού μέχρι την προώθηση του τελικού προϊόντος στην αγορά. Η μεθοδολογία αυτή θα πρέπει να καθορίσει την διαδικασία ανάπτυξης και παραγωγής του λογισμικού που θα ακολουθηθεί. Επίσης, θα πρέπει να περιλαμβάνει τις τεχνικές για την διαχείριση των πόρων, το σχεδιασμό, τον προγραμματισμό του λογισμικού καθώς και άλλες εργασίες που σχετίζονται με τη διαχείριση έργων πληροφορικής. Οι Munassar & Govardhan, (2011) αναφέρουν, επίσης, ότι η ευρεία χρήση των κατάλληλων μεθοδολογιών ανάπτυξης του λογισμικού είναι απαραίτητες για την ωρίμανση του κλάδου.

Ο O'Docherty, (2005) εξηγεί ότι η χρήση της κατάλληλης μεθοδολογίας θα αντιμετωπίσει τουλάχιστον τα ακόλουθα ζητήματα: σχεδιασμού, προγραμματισμού, κατανομής πόρων, ροών εργασίας, δραστηριοτήτων, ρόλων, εκπαίδευσης. Οι Guimarães & Vilela, (2005) υποστηρίζουν ότι υπάρχουν κάποιες όμοιες φάσεις σε κάθε μεθοδολογία ανάπτυξης λογισμικού, ξεκινώντας με την καταγραφή των απαιτήσεων και τελειώνοντας με τη συντήρηση και την αξιολόγηση του προϊόντος. Τις τελευταίες δεκαετίες έχουν αναπτυχθεί μια σειρά από νέα μοντέλα κύκλου ζωής λογισμικού στην επιστήμη της μηχανικής του λογισμικού. Ο O'Docherty, (2005) προσθέτει επίσης ότι η χρήση

παραδοσιακών μεθόδων, προϋποθέτει την σειριακή ακολουθία βημάτων ανάπτυξης του λογισμικού. Με τη χρήση των νέων μοντέρνων προσεγγίσεων, επιτρέπεται να εκτελεστούν περισσότερες από μία φάσεις ανάπτυξης, με όποια σειρά επιθυμεί ο δημιουργός του λογισμικού.

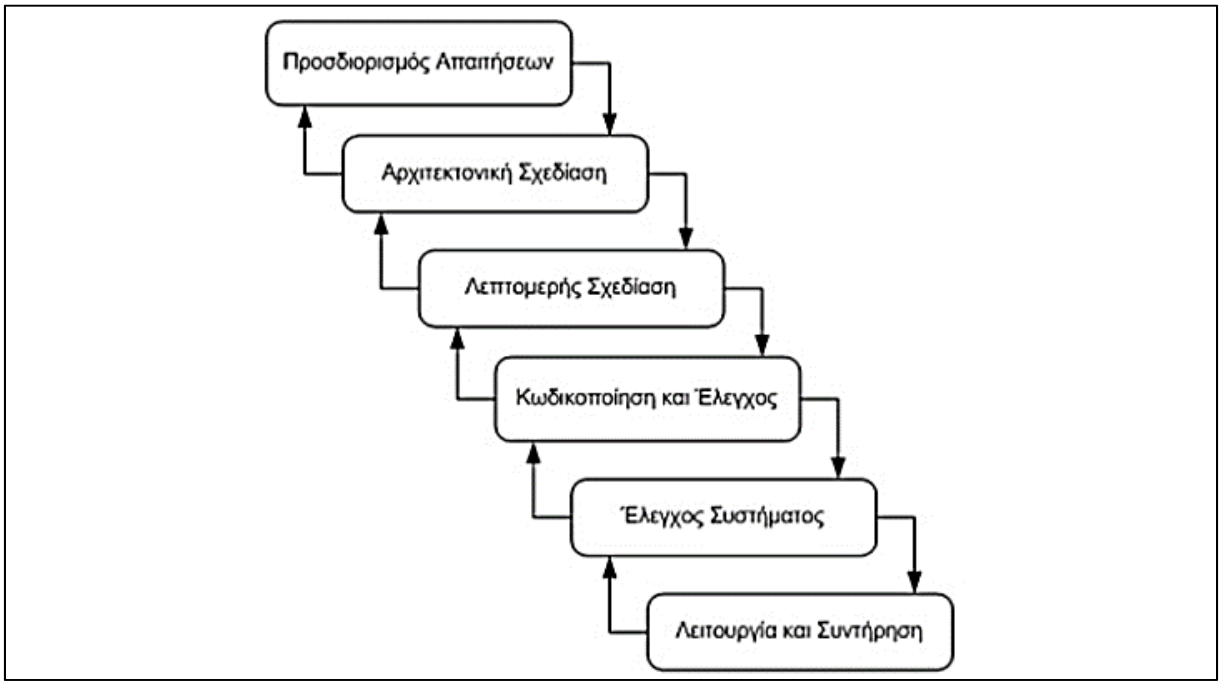
2.2 Οι παραδοσιακές μέθοδοι ανάπτυξης λογισμικού

Οι παραδοσιακές μέθοδοι ανάπτυξης λογισμικού βασίζονται σε μια διαδοχική σειρά από φάσεις ανάπτυξης, όπως απαιτήσεις, προγραμματισμός, ανάλυση, σχεδίαση και υλοποίηση (Yu, et al., 2012). Υπάρχουν πολλές διαφορετικές παραδοσιακές μέθοδοι ανάπτυξης λογισμικού. Στην παρούσα μεταπτυχιακή διατριβή θα αναφερθούν οι πέντε δημοφιλέστερες μέθοδοι: το μοντέλο του καταρράκτη, το σπειροειδές μοντέλο, το μοντέλο της προτυποποίησης, το μοντέλο της λειτουργικής επαύξησης και το μοντέλο του πίδακα. Σύμφωνα με τους Nikiforova, et al., (2009), οι παραδοσιακές μέθοδοι ανάπτυξης λογισμικού εξαρτώνται από ένα σύνολο από προκαθορισμένες διεργασίες και εν εξελίξει έγγραφα τα οποία συμπληρώνονται καθώς το έργο εξελίσσεται και καθοδηγούν την περαιτέρω ανάπτυξη του. Η επιτυχία ενός έργου που προσεγγίζεται με τις παραδοσιακές μεθόδους ανάπτυξης λογισμικού βασίζεται στον καθορισμό των απαιτήσεων του χρήστη πριν από τη φάση της ανάπτυξης του λογισμικού. Η εφαρμογή αλλαγών καθίσταται δύσκολη. Ωστόσο, με τις παραδοσιακές μεθόδους είναι ευκολότερο να καθοριστούν οι δαπάνες του έργου, να οριστεί ένα χρονοδιάγραμμα και να εκχωρηθούν οι κατάλληλοι πόροι (Gornik, 2001).

Οι Yu, et al., (2012) ορίζουν τέσσερις φάσεις που είναι χαρακτηριστικές μιας παραδοσιακής μεθόδου ανάπτυξης λογισμικού. Το πρώτο βήμα είναι να καθοριστούν οι απαιτήσεις για το έργο και να σχεδιαστεί το χρονοδιάγραμμα που θα χρησιμοποιηθεί στις διάφορες φάσεις. Μόλις καθοριστούν οι απαιτήσεις, το επόμενο βήμα είναι η φάση σχεδιασμού και αρχιτεκτονικού σχεδιασμού. Στη φάση αυτή παράγεται μια τεχνική υποδομή με τη μορφή διαγραμμάτων και μοντέλων. Αυτά φέρνουν στην επιφάνεια πιθανά ζητήματα που μπορεί να αντιμετωπίσει το έργο, καθώς προχωράει και παρέχουν έναν εφαρμόσιμο οδικό χάρτη για την υλοποίηση του από τους προγραμματιστές.

Αφού η ομάδα ικανοποιηθεί με το αρχιτεκτονικό σχέδιο, το έργο κινείται στην επόμενη φάση, όπου ο κώδικας παράγεται μέχρι να φτάσει σε καθορισμένους στόχους. Συνήθως, η ανάπτυξη διαχωρίζεται σε μικρότερα καθήκοντα, που κατανέμονται μεταξύ διαφορετικών ομάδων με βάση τις δεξιότητες και τις γνώσεις τους. Μετά την ανάπτυξη υπάρχουν συνήθως δοκιμές, αλλά η φάση της δοκιμής συχνά επικαλύπτεται με τη φάση ανάπτυξης, για να διασφαλιστεί ότι τα ζητήματα εντοπίζονται νωρίτερα. Όταν το έργο έρχεται στο τέλος και οι προγραμματιστές πλησιάζουν την τήρηση των απαιτήσεων του, ο πελάτης συμμετέχει στον κύκλο δοκιμών και ανατροφοδότησης και το λογισμικό παίρνει την τελική μορφή του.

2.2.1 Το μοντέλο του καταρράκτη



Εικόνα 1. Η φιλοσοφία του μοντέλου του Καταρράκτη **(Petersen, et al., 2009)**.

Το μοντέλο καταρράκτη (Waterfall model) είναι το παραδοσιακό μοντέλο της μηχανικής λογισμικού (Εικόνα 1). Το μοντέλο καταρράκτη είναι μία από τις παλαιότερες και πιο συχνά χρησιμοποιούμενες μεθόδους και έχει χρησιμοποιηθεί ευρέως σε δημόσια έργα και από πολλές μεγάλες ιδιωτικές εταιρείες. Το χαρακτηριστικό αυτού του μοντέλου είναι ότι σχεδιάζεται σε πρώιμα στάδια για να αποκαλύψει τα ελαττώματα του σχεδιασμού πριν αναπτυχθούν. Επίσης, το μοντέλο ενθαρρύνει την εντατική τεκμηρίωση και τον

προγραμματισμό, γεγονός που το κάνει να λειτουργεί καλά για έργα, όπου ο ποιοτικός έλεγχος είναι σημαντικός παράγοντας (Munassar & Govardhan, 2011). Ο Punkka, (2005) εξηγεί ότι ο καταρράκτης παίρνει τη διαδικασία και τη χωρίζει σε ξεχωριστές φάσεις που ακολουθούν η μία την άλλη. Κανονικά οι φάσεις περιλαμβάνουν τις απαιτήσεις, την ανάλυση, τον σχεδιασμό, την υλοποίηση και την ενσωμάτωση, αλλά μπορεί επίσης να υπάρχουν διαφορετικές παραλλαγές. Η συνέχιση της επόμενης φάσης είναι διαδοχική και απαιτεί την ολοκλήρωση ορισμένων κριτηρίων. Το πρόβλημα με ένα διαδοχικό μοντέλο είναι ότι είναι ένας συνδυασμός διαφορετικών σταδίων. Αυτό οδηγεί σε μια κατάσταση όπου ο μοναδικός χαρακτήρας του λογισμικού δεν λαμβάνεται υπόψη.

Σύμφωνα με τους Petersen, et al., (2009), το μοντέλο καταρράκτη που χρησιμοποιείται από εταιρείες διατρέχει τις φάσεις: προσδιορισμός απαιτήσεων, σχεδιασμός και υλοποίηση, κωδικοποίηση και έλεγχος, λειτουργία και συντήρηση. Ο έλεγχος ποιότητας πραγματοποιείται μεταξύ όλων των φάσεων και για τη μετάβαση στην επόμενη φάση πρέπει να ολοκληρωθεί ο ποιοτικός έλεγχος της προηγούμενης, αυτή η προσέγγιση αναφέρεται ως μοντέλο πύλης.

Οι Petersen, et al., (2009) εξηγούν τα βήματα υλοποίησης του μοντέλου του καταρράκτη ως εξής:

Συλλογή Απαιτήσεων: Σε αυτή τη φάση, οι ανάγκες των πελατών εντοπίζονται και τεκμηριώνονται με υψηλό επίπεδο αφαιρετικότητας. Μετά από αυτήν, οι απαιτήσεις είναι ακριβείς, ώστε να μπορούν να χρησιμοποιηθούν ως εισροές για την επόμενη φάση. Όλες οι απαιτήσεις αποθηκεύονται σε ένα έγγραφο συλλογής απαιτήσεων. Στην πρώτη πύλη ποιότητας (και στις άλλες πύλες) ελέγχεται κατά πόσο όλες οι απαιτήσεις γίνονται κατανοητές και τεκμηριώνονται.

Σχεδιασμός και Υλοποίηση: Μετά τον καθορισμό των απαιτήσεων, η επόμενη φάση είναι ο σχεδιασμός και η υλοποίηση. Στη φάση σχεδιασμού δημιουργείται και τεκμηριώνεται η αρχιτεκτονική του συστήματος. Μετά την ολοκλήρωση του σχεδιασμού της αρχιτεκτονικής, ξεκινά η πραγματική ανάπτυξη και κωδικοποίηση. Οι προγραμματιστές πραγματοποιούν επίσης βασικές δοκιμές μονάδας, πριν περάσουν τον κώδικα στην

επόμενη φάση. Η λίστα ελέγχου πύλης ποιότητας επαληθεύει εάν η αρχιτεκτονική έχει αξιολογηθεί και διαφοροποιείται από την προηγούμενη απόφαση ελέγχου ποιότητας.

Έλεγχος Συστήματος: Σε αυτή τη φάση, η ολοκλήρωση του συστήματος δοκιμάζεται σχετικά με τις ιδιότητες ποιότητας και λειτουργικότητας. Σκοπός αυτής της φάσης είναι να αποφασιστεί εάν μπορεί να αναπτυχθεί το λογισμικό. Μέτρα ποιότητας και λειτουργικά χαρακτηριστικά συλλέγονται στο σύστημα ποιότητας. Σε μια περίπτωση, όπου η εταιρεία παρέχει ολοκληρωμένες λύσεις (συμπεριλαμβανομένου του υλικού και του λογισμικού), οι δοκιμές πρέπει να διεξάγονται με ποικίλες διαμορφώσεις υλικού και λογισμικού, επειδή αυτές μπορεί να διαφέρουν μεταξύ των πελατών. Η λίστα ελέγχου πύλης ποιότητας εξετάζει αν το αναπτυγμένο σύστημα έχει επαληθευτεί και διαφέρει από τις προηγούμενες ποιοτικές πύλες. Η πύλη ποιότητας ελέγχει εάν το αποτέλεσμα του έργου ικανοποιεί τις απαιτήσεις των πελατών.

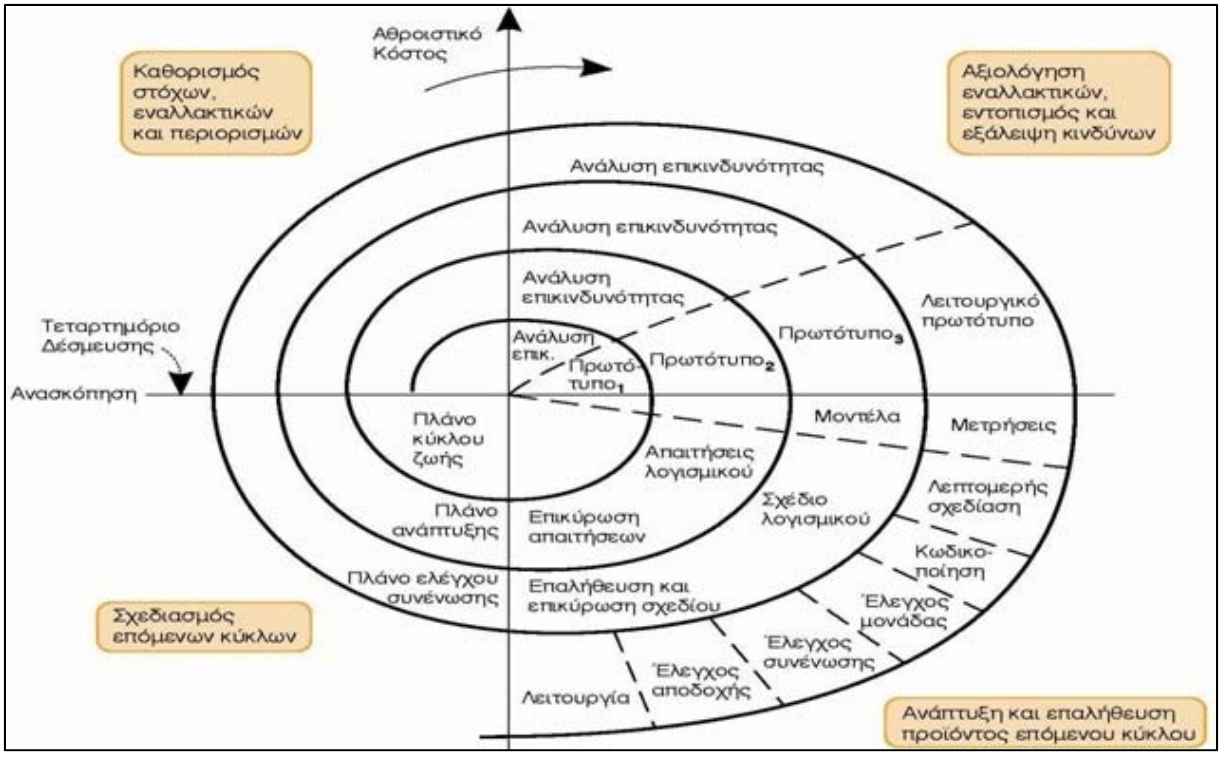
Λειτουργία: Πριν από την τελική φάση, το προϊόν είναι τώρα σε κατάσταση τελικής διαμόρφωσης. Η τεκμηρίωση έχει πλέον οριστικοποιηθεί (π.χ. οδηγίες εγκατάστασης του συστήματος για πελάτες και οδηγούς χρηστών). Στην ποιοτική πύλη ελέγχεται εάν το αποτέλεσμα ικανοποιεί τις απαιτήσεις των πελατών. Μια τελική ανάλυση διεξάγεται επίσης στο τέλος αυτής της φάσης.

Συντήρηση: Κατά την τελευταία φάση το προϊόν αποδίδεται στον πελάτη και πρέπει να συντηρηθεί. Σε περίπτωση που ο πελάτης ανακαλύψει προβλήματα σχετικά με το προϊόν, τα αναφέρει στην εταιρεία και λαμβάνει υποστήριξη. Εάν τα προβλήματα οφείλονται σε λάθη στο προϊόν, οι ενημερώσεις του συστήματος παραδίδονται στους πελάτες.

Υπάρχουν επίσης ορισμένα προβλήματα σχετικά με την εφαρμογή του μοντέλου του καταρράκτη. Ο Brooks, (1987) δηλώνει ότι οι ιδιότητες του λογισμικού είναι η πολυπλοκότητα, η συμμόρφωση και η μεταβλητότητα. Όταν εξετάζουμε τις αρχές των παραδοσιακών μεθόδων ανάπτυξης λογισμικού, αυτές δεν ικανοποιούν πλήρως τις ιδιότητες του λογισμικού. Επίσης, οι Munassar & Govardhan, (2011) υποστηρίζουν ότι πολλοί πιστεύουν ότι αυτό το μοντέλο δεν μπορεί να εφαρμοστεί σε όλες τις καταστάσεις. Για παράδειγμα, όταν χρησιμοποιείται το μοντέλο καταρράκτη, οι

απαιτήσεις πρέπει να καθοριστούν πριν από το σχεδιασμό και ο πλήρης σχεδιασμός πρέπει να δηλωθεί πριν από την ανάπτυξη. Αυτό έχει ως αποτέλεσμα να μην υπάρχει επικάλυψη μεταξύ των φάσεων. Ενώ, δεν απαγορεύεται η επιστροφή σε προγενέστερη φάση της μεθόδου, για παράδειγμα, επιστροφή από τη φάση σχεδιασμού στη φάση απαιτήσεων, το πρόβλημα είναι ότι η επιστροφή συνεπάγεται δαπανηρή επαναδιατύπωση. Επειδή, η ουσιαστική εξέλιξη του λογισμικού έρχεται αργά στη διαδικασία, κανείς δεν βλέπει αποτελέσματα για μεγάλο χρονικό διάστημα. Οι Munassar & Govardhan, (2010) προσθέτουν επίσης ότι αν και το μοντέλο του καταρράκτη έχει τις αδυναμίες του, είναι διδακτικό, γιατί ακολουθεί σημαντικές και βασικές φάσεις του κύκλου ζωής του λογισμικού.

2.2.2 Το σπειροειδές μοντέλο



Εικόνα 2. Η φιλοσοφία του σπειροειδούς μοντέλου **(Boehm, 1986)**.

Σύμφωνα με τον Boehm, (1986), το σπειροειδές μοντέλο (Spiral model) του κύκλου ζωής λογισμικού (Εικόνα 2) έχει προκύψει από διάφορες βελτιώσεις του μοντέλου καταρράκτη. Ο Munassar & Govardhan, (2010) εξηγούν ότι το σπειροειδές μοντέλο είναι παρόμοιο με το μοντέλο του καταρράκτη, αλλά τονίζει περισσότερο την ανάλυση του κινδύνου. Το μοντέλο των σπειρών χωρίζεται σε τέσσερις φάσεις: Σχεδιασμός, Ανάλυση

Κινδύνου, Μηχανική και Αξιολόγηση. Ένα έργο λογισμικού περνά αυτές τις φάσεις σε επαναλήψεις, που συνεχίζουν επανειλημμένα. Αυτές οι επαναλήψεις ονομάζονται σπείρες σε αυτό το μοντέλο.

Ο Boehm, (1986) εξηγεί τις φάσεις του σπειροειδούς μοντέλου. Στον τυπικό κύκλο της σπείρας κάθε κύκλος αρχίζει με την ταυτοποίηση τριών διαφορετικών τμημάτων:

- Οι στόχοι του διαφορετικού τμήματος του προϊόντος που επεξεργάζεται (απόδοση, λειτουργικότητα, ικανότητα προσαρμογής της αλλαγής κ.λπ.).
- Οι εναλλακτικές απαιτήσεις διαφόρων τμημάτων του προϊόντος.
- Οι περιορισμοί που επιβάλλονται στην εφαρμογή των εναλλακτικών λύσεων.

Η επόμενη φάση είναι η αξιολόγηση των εναλλακτικών λύσεων σε σύγκριση με τους στόχους και τους περιορισμούς. Σκοπός αυτής της φάσης είναι να εντοπιστούν τομείς, που αποτελούν σημαντικές πηγές κινδύνου για το έργο. Εάν εντοπιστεί κάποιος κίνδυνος, το επόμενο βήμα θα πρέπει να είναι η αποτελεσματική από πλευράς κόστους στρατηγική για την επίλυση των πηγών κινδύνου. Η στρατηγική μπορεί να περιλαμβάνει δημιουργία πρωτότυπων, προσομοίωση, συγκριτική αξιολόγηση, έλεγχο αναφοράς, διαχείριση ερωτηματολογίων χρήστη, αναλυτική μοντελοποίηση ή συνδυασμούς αυτών και άλλων τεχνικών επίλυσης κινδύνου (Boehm, 1986).

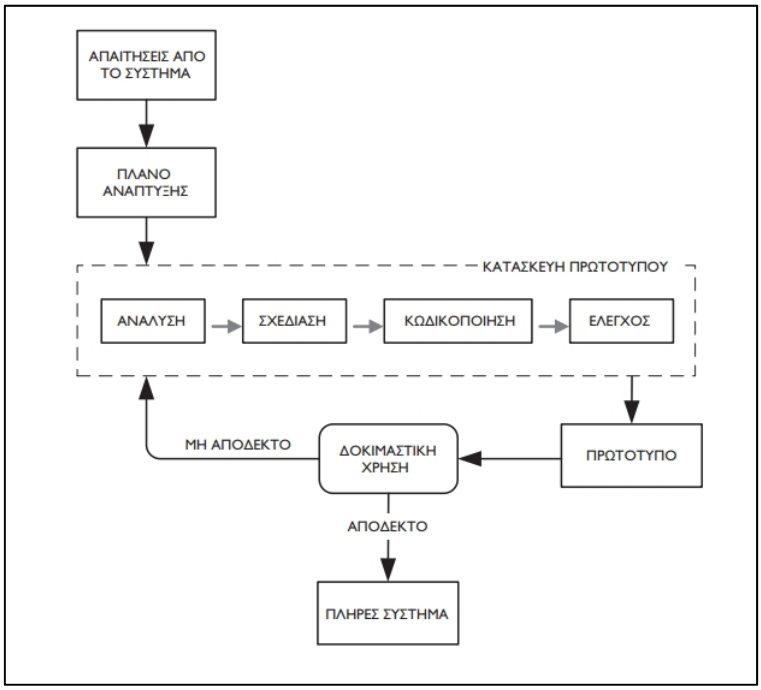
Μετά την αξιολόγηση των κινδύνων, η επόμενη φάση καθορίζεται από τους υπόλοιπους κινδύνους. Εάν οι επιδόσεις ή οι κίνδυνοι διεπαφής χρήστη κυριαρχούν από τους κινδύνους του προγραμματισμού ή των εσωτερικών διασυνδέσεων, το επόμενο βήμα μπορεί να είναι, για παράδειγμα: μια ελάχιστη προσπάθεια προσδιορισμού της συνολικής φύσης του προϊόντος, ενός σχεδίου για το επόμενο επίπεδο πρωτοτύπων ή η ανάπτυξη ενός πιο λεπτομερούς πρωτοτύπου για τη συνέχιση της επίλυσης των κυριότερων ζητημάτων κινδύνου. Αν αυτό το πρωτότυπο είναι λειτουργικά χρήσιμο και αρκετά ισχυρό, ώστε να χρησιμεύσει ως βάση χαμηλού κινδύνου για τη μελλοντική εξέλιξη του προϊόντος, τα επόμενα βήματα με γνώμονα τον κίνδυνο θα είναι μια σειρά εξελικτικών πρωτοτύπων, σύμφωνα με την Εικόνα 2 (Boehm, 1986).

Στην περίπτωση όπου τα προηγούμενα πρωτότυπα έχουν ήδη επιλύσει όλους τους κινδύνους επιδόσεων ή διεπαφής χρήστη και περάσει τον έλεγχο του προγραμματισμού ή του ελέγχου των διεπαφών, το επόμενο βήμα ακολουθεί τη βασική προσέγγιση του καταρράκτη. Σε αυτή την περίπτωση, κάθε επίπεδο προδιαγραφών λογισμικού στην Εικόνα 2 ακολουθείται από ένα βήμα επικύρωσης και την προετοιμασία σχεδίων για τον επόμενο κύκλο. Σε αυτή την περίπτωση, οι επιλογές για πρωτότυπο, προσομοίωση, μοντέλο κ.λπ. αντιμετωπίζονται, οδηγώντας στη χρήση ενός διαφορετικού υποσυνόλου βημάτων (Boehm, 1986).

Οι Munassar & Govardhan, (2010) περιέγραψαν τα πλεονεκτήματα και τα μειονεκτήματα του σπειροειδούς μοντέλου. Τα πλεονεκτήματα περιλαμβάνουν υψηλό ποσοστό ανάλυσης κινδύνου, καλή προσαρμογή για μεγάλα έργα και έργα ζωτικής σημασίας, και λογισμικό, που παράγεται στις αρχές του κύκλου ζωής του. Μειονεκτήματα του μοντέλου είναι ότι γίνεται δαπανηρό για χρήση, απαιτείται ειδική εμπειρογνωμοσύνη για την ανάλυση κινδύνου και η επιτυχία του έργου εξαρτάται από τη φάση ανάλυσης κινδύνου.

2.2.3 Το μοντέλο της προτυποποίησης

Το μοντέλο της προτυποποίησης (Prototyping model) βασίζεται στην τμηματική ανάπτυξη του λογισμικού, τα τμήματα του οποίου ονομάζονται «πρωτότυπα». Η μεθοδολογία ανάπτυξης επαναλαμβάνεται διαδοχικά για κάθε διεργασία του λογισμικού και έτσι το μοντέλο αυτό έχει χαρακτηριστεί, από την επιχειρηματική και επιστημονική κοινότητα, ως επαναληπτικό. Κάθε πρωτότυπο του λογισμικού διαθέτει τις πιο χαρακτηριστικές από τις διεργασίες, που είναι προορισμένο να εκπληρώσει και στην συνέχεια δίδεται στον πελάτη για αξιολόγηση. Ο πελάτης δίνει στο προγραμματιστή και το σχεδιαστή του συστήματος τις παρατηρήσεις του και η εργασία ανάπτυξης αναθεωρημένου πρωτοτύπου επαναλαμβάνεται, έως ότου, το τελικό πρωτότυπο να εκπληρώνει τις προσδοκίες του πελάτη, δηλαδή να επιτελεί τις αναγκαίες απαιτήσεις του λογισμικού, όπως τις έχει σκεφθεί ο πελάτης, ο οποίος στην τελική αξιολόγηση θα πρέπει να αποδεχθεί το λογισμικό (Εικόνα 3). Μετέπειτα, προστίθενται και οι δευτερεύουσες διεργασίες και εργαλεία και το λογισμικό αποκτά την τελική του μορφή και ολοκληρώνεται η εργασία.



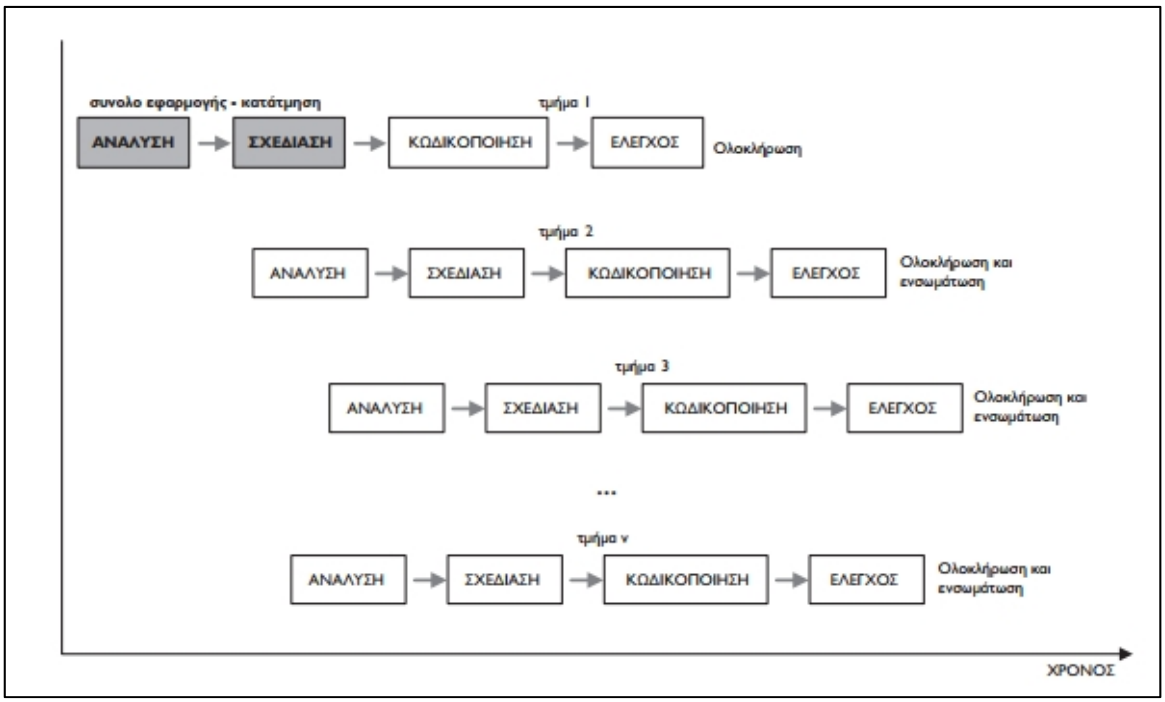
Εικόνα 3. Η φιλοσοφία του μοντέλου της προτυποποίησης **(Βεσκούκης, 2015)**.

Το πιο σημαντικό προτέρημα του μοντέλου της προτυποποίησης είναι ότι δίδει την δυνατότητα στο πελάτη να διαμορφώσει άποψη για το λογισμικό σε αρχικές φάσεις ανάπτυξής του και σαφώς νωρίτερα από άλλα μοντέλα ανάπτυξης λογισμικού, όπως π.χ. του καταρράκτη. Το γεγονός αυτό προσδίδει στο έργο ευελιξία και γλιτώνει από καθυστερήσεις και συνεπαγόμενη αύξηση του κόστους ανάπτυξης του λογισμικού, καθώς και από, στην χειρότερη των περιπτώσεων, απόρριψη του λογισμικού από την πλευρά του πελάτη, στην περίπτωση που δεν τον ικανοποιεί το τελικό αποτέλεσμα. Παράλληλα, θα πρέπει η διοίκηση του έργου να δείξει την απαιτούμενη αφοσίωση και προσήλωση στην υλοποίηση των πρωτοτύπων του λογισμικού και στις διαδικασίες αναθεώρησης τους. Όμως, θα πρέπει να τονισθεί ότι η υλοποίηση του πρωτοτύπου θεωρείται ένα ξεχωριστό έργο και μπορεί να πραγματοποιηθεί ακολουθώντας μεθοδολογία άλλων μοντέλων ανάπτυξης λογισμικού, όπως αυτού του καταρράκτη.

Με βάση τα παραπάνω θεωρείται βέβαιο ότι το μοντέλο της προτυποποίησης χρησιμοποιείται σε έργα ανάπτυξης λογισμικού για τα οποία οι απαιτήσεις τους δεν έχουν διαμορφωθεί στα αρχικά στάδια του έργου και ενδεχόμενα θα υπάρξουν αλλαγές σε μεταγενέστερες φάσεις της ανάπτυξης. Τέτοια έργα θεωρούνται τα έργα, τα οποία ο πελάτης δεν είναι σίγουρος για την τελική μορφή του λογισμικού, αλλά θέλει να

δοκιμάσει διαφορετικές μορφές υλοποίησης του τελικού προϊόντος. Επομένως, το έργο εξαρτάται απόλυτα από τις διαθέσεις και απαιτήσεις του. Ωστόσο, το μέγεθος του λογισμικού δεν μπορεί να είναι μεγάλο, διότι η υλοποίηση του πρωτοτύπου θέλει σημαντικό χρόνο και η ευελιξία του έργου μειώνεται.

2.2.4 Το μοντέλο της λειτουργικής επαύξησης



Εικόνα 4. Η φιλοσοφία του μοντέλου της λειτουργικής επαύξησης **(Βεσκούκης, 2015)**.

Το μοντέλο της λειτουργικής επαύξησης (Incremental model) αποτελεί ένα συνδυασμό των δύο μοντέλων ανάπτυξης, του καταρράκτη, από το οποίο υιοθετεί την σειριακή ανάπτυξη του λογισμικού και της προτυποποίησης από το οποίο ακολουθεί την δημιουργία πρωτοτύπων (Εικόνα 4).

Το κύριο βάρος υλοποίησης που ακολουθείται είναι η τμηματοποίηση του έργου σε μικρά κομμάτια τα οποία υλοποιούνται ανεξάρτητα το ένα από το άλλο, υιοθετώντας όπως προαναφέρθηκε την σειριακή φιλοσοφία του μοντέλου του καταρράκτη. Το σημαντικότερο βήμα υλοποίησης αποτελεί η κατάτμηση του έργου στα κατάλληλα τμήματα, η ανάπτυξη των οποίων είναι ανεξάρτητη και παράλληλη. Όταν η ανάπτυξη ενός τμήματος ολοκληρωθεί, τότε αυτό ενσωματώνεται στην συνολική εφαρμογή του

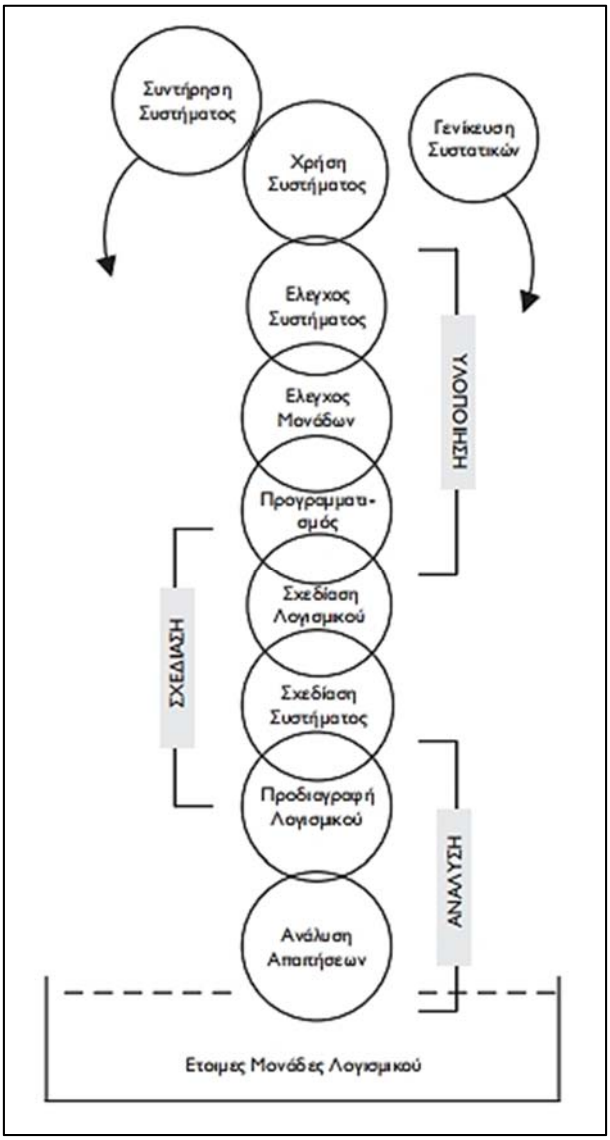
λογισμικού και θεωρείται ένα είδος επαύξησης του έργου, οπότε προέκυψε και η ονομασία του μοντέλου ως «μοντέλο λειτουργικής επαύξησης».

Στα πλεονεκτήματα του μοντέλου καταλογίζεται η ανάπτυξη των τμημάτων του λογισμικού, παράλληλα, η οποία έχει ως αποτέλεσμα την μείωση του συνολικού χρόνου υλοποίησης του έργου και η σειριακή προσθήκη των λειτουργικών χαρακτηριστικών του λογισμικού. Τα κυριότερα μειονεκτήματά του είναι η σημασία που δίνεται στην κατάτμηση του έργου του λογισμικού, αφού τυχόν λάθη σε αυτή την φάση του συστήματος μπορεί πιθανότατα να οδηγήσει ακόμη και στην αποτυχία ολόκληρου του έργου. Στην περίπτωση που σε κάποιο από τα τμήματα υλοποίησης αλλάξουν οι απαιτήσεις σε τέτοιο βαθμό που να αλλάζει ολόκληρη η αρχιτεκτονική του συστήματος, τότε μπορεί να επηρεαστεί και η ανάπτυξη ολόκληρων των υπόλοιπων τμημάτων του έργου.

Γενικότερα, το μοντέλο της λειτουργικής επαύξησης χρησιμοποιείται για την ανάπτυξη μεγάλων έργων λογισμικού, στα οποία μπορεί να εφαρμοστεί η σειριακή πορεία ανάπτυξης του καταρράκτη και υπάρχει μικρή έως μηδαμινή μεταβλητότητα των απαιτήσεών του λογισμικού, καθώς και σαφής γνώση της μορφή του.

2.2.5 Το μοντέλο του πίδακα

Πολλά από τα μοντέλα κύκλου ζωής λογισμικού που αναφέρονται στην παρούσα μεταπτυχιακή διατριβή συγκροτούν τροποποιήσεις αυτών, τα γνωρίσματα των οποίων υποκινούνται από τις μεθόδους υλοποίησης λογισμικού. Οι πρώτες υλοποιήσεις λογισμικού, με βάση την αντικειμενοστραφή (object-oriented) τεχνολογία διαμόρφωσαν το μοντέλο του πίδακα (Fountain model), βασιζόμενες σε δύο ιδιαίτερα χαρακτηριστικά της τεχνικής: πρώτον, ότι οι φάσεις «ανάλυση», «σχεδίαση», «κωδικοποίηση» πλησιάζουν στο αντικειμενοστραφές προγραμματισμό και δεύτερον ότι το επακόλουθο κάθε φάσης υλοποίησης λογισμικού αποτελείται όχι μόνο από ένα σύστημα, αλλά και από επαναχρησιμοποιήσιμες μονάδες, οι οποίες μπορούν να χρησιμοποιηθούν από τις πρώτες φάσεις της ανάπτυξης μελλοντικών λογισμικών. Με τον τρόπο αυτό περιγράφεται το μοντέλο του πίδακα που απεικονίζεται στην Εικόνα 5.



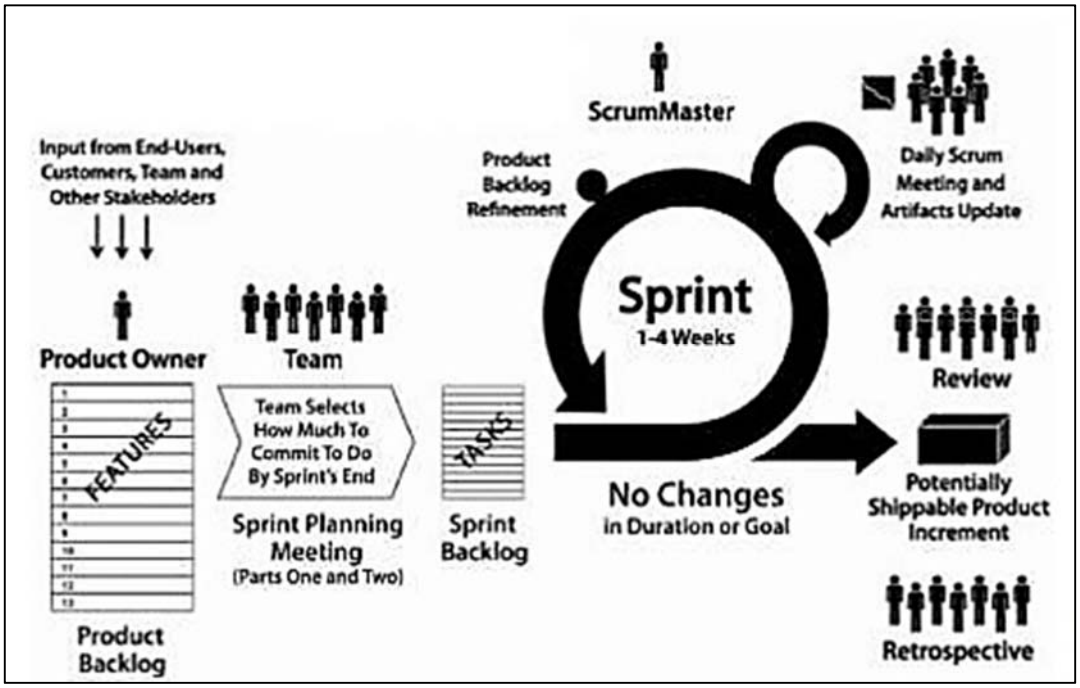
Εικόνα 5. Η φιλοσοφία του μοντέλου του πίδακα (Βεσκούκης, 2015)**.**

Κατά την υλοποίηση του λογισμικού, γίνονται καλύψεις των φάσεων «ανάλυση», «σχεδίαση», «κωδικοποίηση», οι οποίες διακρίνονται στα κοινά σημεία των κύκλων της Εικόνα 5. Με το πέρας της υλοποίησης κάποια από τα συστήματα του λογισμικού, που έχουν αναπτυχθεί εισχωρούν σε μια «δεξαμενή» στοιχείων και χρησιμοποιούνται στην υλοποίηση μελλοντικών λογισμικών. Η βασική αρχή του μοντέλου του πίδακα υπογραμμίζει κυρίως τα επιθυμητά γνωρίσματα της μεθόδου υλοποίησης του λογισμικού με βάση την αντικειμενοστραφή προγραμματιστική τεχνική. Η μέθοδος του πίδακα ήταν αρκετά δημοφιλής κατά την περίοδο που μεσουρανούσε η αντικειμενοστραφή τεχνολογία, κυρίως στα τέλη της δεκαετίας του ’80 έως τις αρχές της δεκαετίας του ’90.

2.3 Οι Ευέλικτες Μέθοδοι Ανάπτυξης Λογισμικού

Στην παρούσα μεταπτυχιακή διατριβή αναφέρονται επτά δημοφιλείς μέθοδοι ευέλικτης ανάπτυξης λογισμικού, όπως: Scrum, ακραίος προγραμματισμός (XP), FDD, DSDM, Lean Software Development, Crystal Clear και Open Unified Process.

2.3.1 Η μέθοδος Scrum



Εικόνα 6. Η φιλοσοφία της ευέλικτης μεθόδου Scrum (Πηγή:Aequor technology).

Η μέθοδος Scrum ανήκει στην οικογένεια των ευέλικτων μεθόδων ανάπτυξης λογισμικού (agile) που έχουν σημειώσει σημαντική αποδοχή μεταξύ των επαγγελματιών λογισμικού (Mahnič & Drnovščeк, 2005). Οι Kane & Schwaber, (2002) περιγράφουν την Scrum ως μεθοδολογία ανάπτυξης προϊόντων, που αποτελείται από πρακτικές και κανόνες που πρέπει να χρησιμοποιούνται από τη διοίκηση, τους πελάτες και το τμήμα διαχείρισης έργων για τη μεγιστοποίηση της παραγωγικότητας και της αξίας της ανάπτυξης. Ο Sutherland, (2010), ορίζει την Scrum, ως επαναληπτικό και διαδοχικό πλαίσιο διεργασιών για την ανάπτυξη λογισμικού. Η ιδέα της Scrum είναι να αναπτύξει λογισμικό σε κύκλους που ονομάζονται Sprints. Αυτές οι επαναλήψεις διαρκούν λιγότερο από ένα μήνα και συνήθως μετρούνται σε εβδομάδες.

Ο Schwaber, (1995) χωρίζει την μέθοδο Scrum σε τρεις χωριστές φάσεις (Εικόνα 6):

Φάση Σχεδιασμού: Σε αυτή τη φάση ορίζονται οι προδιαγραφές του προϊόντος, με εκτίμηση του χρονοδιαγράμματος και του κόστους. Εάν αναπτύσσεται ένα νέο λογισμικό, αυτή η φάση περιλαμβάνει κυρίως την ανάλυση της ιδέας. Εάν επικαιροποιείται υπάρχον λογισμικό, αυτή η φάση αποτελείται μόνο από την ανάλυση (Schwaber, 1995). Μετά τη φάση σχεδιασμού, πραγματοποιείται η υλοποίηση των αντικειμένων του λογισμικού.

Φάση ανάπτυξης: Ανάπτυξη λογισμικού με συνεχή σεβασμό στις μεταβλητές του χρόνου, των απαιτήσεων, της ποιότητας, του κόστους και του ανταγωνισμού. Η φάση ανάπτυξης περιλαμβάνει πολλά Sprints, που χρησιμοποιούνται για την ανάπτυξη του συστήματος.

Φάση Προώθησης: Στην τελευταία φάση το προϊόν είναι έτοιμο για προώθηση και γίνονται η τελική τεκμηρίωση και οι δοκιμές αποδέσμευσης.

Ο Sutherland, (2010) ορίζει πέντε στάδια μέσα σε ένα Sprint: Κατά πρώτον, ο Product Owner καθορίζει τα ιεραρχικά βήματα με τα γνωρίσματα που θέλει να συμπεριλαμβάνονται στο προϊόν τα οποία λέγονται **Product Backlog** (κατάλογος των ανεκτέλεστων εργασιών του προϊόντος). Η λίστα ανανεώνεται κατά τη διάρκεια της ανάπτυξης του λογισμικού, ώστε να συμπεριλαμβάνει όλες τις οριστικές απαιτήσεις. Οι απαιτήσεις μπορεί να πηγάζουν από τον ίδιο τον πελάτη, το τμήμα πωλήσεων και μάρκετινγκ ή τους ίδιους τους σχεδιαστές του έργου. Οι απαιτήσεις διαχωρίζονται ανάλογα με την σειρά ανάπτυξής τους και ελέγχεται η προσπάθεια που έγινε για την εκπλήρωσή τους. Η λίστα των απαιτήσεων αλλάζει και ανανεώνεται σε κάθε επανάληψη με καινούρια και πιο διεξοδικά δεδομένα, καθώς επίσης και με λεπτομερέστερες αξιολογήσεις για την απαιτούμενη προσπάθεια και τις νέες προτεραιότητες που ανακύπτουν. Το Product Backlog εμπεριέχει κυρίως τα γνωρίσματα του προϊόντος, όπως αυτά καθορίζονται από τον πελάτη, αλλά μπορεί να εμπεριέχει και νέες ιδιότητες ή μειονεκτήματα που θα πρέπει να εξετασθούν από τους σχεδιαστές. Τα χαρακτηριστικά που συνυπολογίζονται στη λίστα, μπορούν να σχεδιαστούν με ορθό τρόπο, όπως για παράδειγμα, μέσω σεναρίων Use cases ή User stories.

Τα **Sprints** είναι επαναλήψεις του έργου. Διαρκούν συνήθως 1-4 εβδομάδες. Τα Sprints τελειώνουν σε συγκεκριμένη ημερομηνία. Στην αρχή κάθε Sprint, πραγματοποιείται η συνάντηση σχεδιασμού Sprint. Ο ιδιοκτήτης του προϊόντος και η ομάδα Scrum εξετάζουν το Product Backlog και ορίζουν τους στόχους για το επερχόμενο Sprint. Κάθε στοιχείο που έχει ληφθεί από το Product Backlog αναλύεται σε ένα σύνολο ατομικών εργασιών για την ομάδα. Το σύνολο των γνωρίσματων του Product Backlog που διεκπεραιώνονται σε μια επανάληψη (**Sprint**), καλείται **Release Backlog**. Η διάταξη με την οποία πραγματοποιείται η υλοποίηση τους συναρτάται από την βαρύτητα, που τους έχει αποδοθεί. Μία ορθή διεκπεραίωση της διαδικασίας προσδιορισμού της βαρύτητας των διαδικασιών, είναι να διεκπεραιώνονται πρώτα οι διεργασίες με υψηλή βαρύτητα και λιγότερο κόστος προς πραγματοποίηση. Όσον αφορά την εκτιμώμενη προσπάθεια που απαιτείται για την πραγματοποίηση κάθε διεργασίας, η ομάδα Scrum ενημερώνει και τον Product Owner (Sutherland, 2010).

Κατά τη φάση ενός Sprint η ομάδα υλοποιεί μια ακόμη από τις πρωταρχικές πρακτικές της Scrum, την τακτική καθημερινή συνεδρίαση Scrum **(Daily Scrum Meeting)**. Είναι μια διεξοδική συνεδρία, η οποία διαρκεί 15 λεπτά και συντελείται καθημερινά. Στη συνεδρίαση αυτή παίρνει μέρος όλη η ομάδα, ώστε τα μέλη της να συντονιστούν μεταξύ τους και να συζητηθούν τα προβλήματα και τα ζητήματα που ανακύπτουν κατά την διάρκεια της υλοποίησης. Κατά την συνεδρίαση αυτή κάθε μέλος της ομάδας πρέπει να συζητήσει με τα άλλα μέλη της ομάδας, τρία βασικά θέματα (Sutherland, 2010):

1. Τι υλοποιήθηκε από την τελευταία συνεδρίαση.
2. Τι περιορισμοί και εμπόδια επέλυσε.
3. Τι στοχεύει να περατώσει μέχρι την μετέπειτα συνεδρία.

Αφού ολοκληρωθεί ένα Sprint, ο Product Owner μαζί με την ομάδα Scrum αναθεωρούν την επανάληψη που μόλις ολοκληρώθηκε. Η αναθεώρηση αυτή λέγεται **Sprint Review**. Η ομάδα προβάλλει τα επακόλουθα του Sprint και διερευνά, κατά πόσο, οι επιδιώξεις που έχουν τεθεί στο Sprint Planning έχουν περατωθεί. Ένας ουσιώδης νόμος στην Scrum είναι η δοκιμή και η προσαρμοστικότητα (inspect and adapt). Αυτό δηλώνει πώς αφού δοκιμαστεί και ελεγχθεί το αποτέλεσμα, τότε συγκεντρώνονται όλες οι αλλαγές ή προσαρμογές, που πρέπει να γίνουν στο λογισμικό κατά τις μετέπειτα επαναλήψεις.

Γίνεται ένας διεξοδικός διάλογος μεταξύ του Product Owner και της ομάδας, ώστε να καθοριστούν τα επόμενα βήματα της ανάπτυξης του λογισμικού. Το **Sprint Retrospective** είναι μια διαδικασία που συνοδεύει τον επανέλεγχο του Sprint. Αφορά τον έλεγχο και επανεξέταση του σχεδιασμού που πραγματοποιήθηκε κατά το τελευταίο Sprint. Η διαδικασία γίνεται για τον έλεγχο των καίριων παραγόντων που θα οδηγήσουν στην επιτυχία υλοποίησης του λογισμικού και στην διερεύνηση πιθανών πτυχών της υλοποίησης που μπορούν να εξελιχθούν. Στην φάση αυτή η ομάδα έχει την ευελιξία να κουβεντιάσει τί λειτουργεί και τί όχι και να οδηγηθεί, σε τυχόν αναθεωρήσεις, που μπορούν να πραγματοποιηθούν στο επόμενο Sprint (Sutherland, 2010).

Στην υλοποίηση του λογισμικού με την μέθοδο Scrum δεν ορίζεται Project Manager, γιατί προκύπτουν άλλοι ρόλοι οι οποίοι αναφέρονται παρακάτω και οι οποίοι μοιράζονται τον ρόλο του:

Product Owner: Πρόκειται για τον υπεύθυνο της οικονομικής παραγωγικότητας της υλοποίησης του λογισμικού (Hundermark, 2009). Επισημαίνει τα γνωρίσματα του λογισμικού, και τα καταχωρεί σε μια λίστα απαιτήσεων ανάλογα με την βαρύτητά τους. Ο Product Owner έχει δραστικό ρόλο στην υλοποίηση των συστημάτων και σε συγκεκριμένες περιστάσεις είναι και ο πελάτης. Στην περίπτωση της Scrum, τον product owner μπορούν να τον αναλάβουν περισσότερα του ενός άτομα.

The **Team** (Η ομάδα): Η ομάδα του Scrum κατά κανόνα συντίθεται από εφτά άτομα. Η ομάδα μπορεί να περιέχει άτομα με ποικίλες δεξιότητες και προτερήματα στην ανάλυση, υλοποίηση, έλεγχο, σχεδιασμό διεπαφής, βάσεις δεδομένων κ.ά. Η ομάδα έχει το χαρακτηριστικό της αυτό-οργάνωσης με μεγάλο επίπεδο αυτοδιοίκησης και υπευθυνότητας. Η ομάδα ορίζει το πώς θα υλοποιηθεί το λογισμικό και δίνει λύσεις και ιδέες στον Product Owner για το πώς θα βελτιώσει την υλοποίηση. Η ομάδα είναι υπεύθυνη, ώστε το λογισμικό να αναπτυχθεί, συμπεριλαμβανομένων όλων των φάσεων υλοποίησης (σχεδιασμός, ανάλυση, προγραμματισμός και δοκιμές). Η ομάδα γίνεται πιο αποδοτική και εποικοδομητική, εάν όλα τα μέλη της είναι πιστά στην υλοποίηση του λογισμικού κατά την φάση ενός Sprint και δεν γίνονται αλλαγές στην σύνθεση της. Οι μεγάλες ομάδες συνήθως χωρίζονται σε μικρότερες και κάθε μια επιλαμβάνεται την

υλοποίηση διαφορετικών γνωρίσματων του λογισμικού, ενώ συγχρόνως διαφυλάσσουν την μεταξύ τους επικοινωνία (Deemer , et al., 2012).

Scrum Master: Είναι υπόλογος απέναντι στην διασφάλιση της σωστής υλοποίησης του λογισμικού και ενισχύει τα μέλη της ομάδας να αποκτήσουν τις απαραίτητες γνώσεις και να υλοποιήσουν καλύτερα τις τεχνικές που ακολουθούνται στην μεθοδολογία του Scrum. Αναλαμβάνει όλη την δράση, ώστε να βοηθήσει την ομάδα και τον Product Owner. Επειδή οι περιορισμοί και τα εμπόδια είναι μέρος της υλοποίησης, είναι αναγκαίο να υπάρχει κάποιος που να μπορεί να δώσει ουσιώδεις λύσεις στα προβλήματα, έτσι ώστε η ομάδα να διαφυλάξει όσο το δυνατόν ανώτερο το επίπεδο αποτελεσματικότητας. Υπό προβλεπόμενες συνθήκες χρειάζεται να ορισθεί ένας Scrum Master που θα επιληφθεί μόνος του τον ρόλο, αλλά σε μικρές ομάδες, κάποιο μέλος της ομάδας θα πρέπει να ορισθεί να αναλάβει τον δύσκολο ρόλο αυτό (Deemer , et al., 2012).

2.3.2 Η μέθοδος Ακραίου Προγραμματισμού (Extreme programming - XP)

Ο ακραίος προγραμματισμός είναι μια μέθοδος από τις πιο δημοφιλείς των ευέλικτων μεθόδων ανάπτυξης λογισμικού. Έχει αποδειχθεί πολύ επιτυχημένη για πολλές εταιρείες και έργα παγκοσμίως. Το κλειδί πίσω από την επιτυχία της μεθόδου είναι ότι τονίζει την ικανοποίηση των πελατών και εξουσιοδοτεί τους προγραμματιστές να αλλάζουν τις απαιτήσεις των πελατών, ακόμα και αργότερα στον κύκλο ζωής. Ο ακραίος προγραμματισμός συγκεντρώνει διευθυντικά στελέχη, πελάτες και προγραμματιστές σε μια συνεργατική ομάδα (Wells, 2013).

Η μεθοδολογία του ακραίου προγραμματισμού βασίζεται σε αξίες και βασικές αρχές. Ο Misic, (2006) περιγράφει τέσσερις βασικές αρχές που περιλαμβάνονται στον ακραίο προγραμματισμό. Η ***επικοινωνία*** βοηθά όλους να κατανοήσουν τι είναι το έργο. Αυτή η αρχή δείχνει ότι η ανάπτυξη λογισμικού είναι μια εντατική διαδικασία μεταξύ των ομάδων και η επικοινωνία τους βοηθά να συνεργάζονται πιο αποτελεσματικά και αποδοτικά. Η ***ενσωμάτωση της αλλαγής*** αποτελεί σημαντικό μέρος της ανάπτυξης λογισμικού. Κατά τη διάρκεια των απαιτήσεων του κύκλου ζωής του λογισμικού, οι προτεραιότητες και ο σχεδιασμός μπορούν να αλλάξουν. Αυτός είναι και ο λόγος για τον οποίο η διαδικασία ανάπτυξης θα πρέπει να είναι σε θέση να συμπεριλάβει τις αλλαγές

και να διατηρήσει τα καλύτερα δυνατά αποτελέσματα. Η ***ανατροφοδότηση*** συμβάλλει στην πλήρη αξιοποίηση της επικοινωνίας. Τελευταία αρχή στον ακραίο προγραμματισμό είναι η ***απλότητα***. Ο στόχος της ανάπτυξης λογισμικού είναι η εύρεση της απλούστερης λύσης που μπορεί να λειτουργήσει. Η απλότητα μεταφράζεται σε αποδοτικότητα, αλλά και σε αποτελεσματικότητα. Αυτές οι τέσσερις βασικές αξίες ή αρχές είναι η βάση των πρακτικών που χρησιμοποιούνται στην μεθοδολογία.

Ο Beck, (1999) αναφέρει τα 11 βήματα που χρησιμοποιούνται στον ακραίο προγραμματισμό:

- *Σχεδιασμός (planning)*: Ο προγραμματιστής αξιολογεί τις εργατοώρες που απαιτούνται για την ανάπτυξη των ιστοριών του λογισμικού, σε συνεργία με τον πελάτη, ορίζει το πρόγραμμα ανάπτυξης των ιστοριών, βάσει υπολογισμών.
- *Μικρές εκδόσεις (small short releases):* Το λογισμικό υλοποιείται με την εφαρμογή μιας σειράς μικρών, λειτουργικών πρωτοτύπων. Οι εκδόσεις προγραμματίζονται κάθε μέρα και έχουν διάρκεια έως και έναν μήνα.
- *Αναπαράσταση του συστήματος με μεταφορές (metaphor):* Μια μεταφορά (metaphor) είναι μια κωδικοποίηση για μία διεργασία του λογισμικού. Το λογισμικό αναπτύσσεται από μία ενότητα μεταφορών μεταξύ πελατών και σχεδιαστών-προγραμματιστών και έτσι γίνεται ευκολότερη η συνεννόηση για το πώς θα αναπτυχθεί.
- *Απλή σχεδίαση (simple design):* Δίνει βαρύτητα στον προγραμματισμό και την αποτελεσματικότερη λύση για το λογισμικό, η οποία και προσαρμόζεται στο έργο, καθώς ανώφελη πολυπλοκότητα και αυξημένος κώδικας θεωρούνται σφάλματα για το λογισμικό.
- *Αναδιάρθρωση του λογισμικού (refactoring):* Η μεταβολή σε ένα τμήμα κώδικα μπορεί αναπόφευκτα να συνεπάγεται σημαντικές μετατροπές σε ένα άλλο τμήμα κώδικα. Η μείωση του κώδικα γίνεται με την αναδιάρθρωση του λογισμικού, την εξέλιξη της επικοινωνίας, την απλότητα και την προσθήκη ευελιξίας, χωρίς να μεταβάλλεται η λειτουργικότητα του λογισμικού.
- *Προγραμματισμός σε ζεύγη (pair programming):* Όλος ο κώδικας αναπτύσσεται από δύο αναλυτές, οι οποίοι εργάζονται συγχρόνως σε ένα υπολογιστή. Ο ένας γράφει και ο δεύτερος παρατηρεί τον κώδικα για τυχόν λάθη. Ο προγραμματιστής

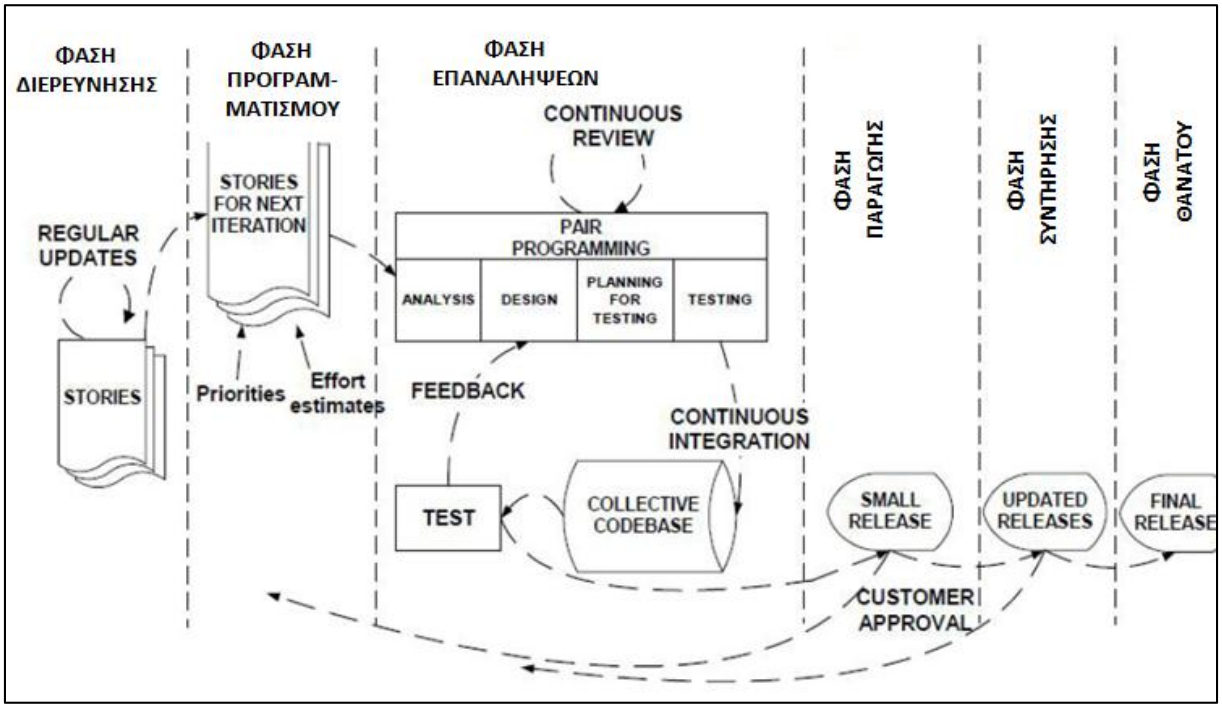
που γράφει τον κώδικα λέγεται «οδηγός», ενώ ο δεύτερος καλείται «παρατηρητής ή πλοηγός».

- *Συλλογική ιδιοκτησία (collective ownership):* Ο κώδικας του λογισμικού είναι συλλογική δουλειά ολόκληρης της ομάδας και όχι μόνο των δύο προγραμματιστών. Κάθε μέλος της ομάδας μπορεί να εισηγηθεί αλλαγή του κώδικα ανά πάσα στιγμή. Με τον τρόπο αυτό επιτυγχάνεται μία ώθηση στην υλοποίηση του λογισμικού, αφού αν κάποιο μέλος της ομάδας εντοπίσει κάποιο λάθος ή ασάφεια μπορεί να το διορθώσει.
- *Συνεχής ολοκλήρωση (continuous intergation):* Κάθε τμήμα του κώδικα που είναι έτοιμο, άμεσα συμπληρώνει το λογισμικό. Κάθε φορά που γίνεται αναθεώρηση του λογισμικού με τον εμπλουτισμό ενός νέου μέρος του κώδικα, τότε θα πρέπει να «ξαναπεράσει» την διαδικασία αξιολόγησης, ώστε οι μετατροπές να γίνουν αποδεκτές.
- *40 ώρες την εβδομάδα (40-hour week):* Οι εβδομαδιαίες ώρες εργασίας είναι 40. Αν αυξηθούν οι ώρες, τότε όλη η ομάδα, το βλέπει ως πρόβλημα υλοποίησης του λογισμικού. Αυτό που η ομάδα προσπαθεί να επιτύχει είναι όλα τα μέλη της να εργάζονται ξεκούραστα, ώστε να αποδίδουν τα μέγιστα.
- *Συμμετοχή του πελάτη (on site customer):* Ο πελάτης «παίζει» καίριο ρόλο στην υλοποίηση του λογισμικού. Δεν συμμετέχει ως πελάτης, αλλά ως μέρος της ομάδας που θα αξιολογήσει το τελικό προϊόν. Ο πελάτης πρέπει να είναι στην διάθεση της ομάδα ανάπτυξης, όποτε τον χρειαστεί, έτοιμος να ξεδιαλύνει κάθε απορία που θα ανακύψει.
- *Πρότυπα κώδικα (coding standards):* Υπάρχουν αρχές κωδικοποίησης που ακολουθούνται πιστά από τους προγραμματιστές, ώστε να υφίσταται η συνέπεια και η επικοινωνία μεταξύ των στελεχών της ομάδας υλοποίησης. Τα πρότυπα καθορίζονται από κοινού από όλα τα μέλη της ομάδας.

Σύμφωνα με τον Abrahamsson, et al., (2002) υπάρχουν συνολικά 6 φάσεις στην μεθοδολογία του ακραίου προγραμματισμού (Εικόνα 7): Διερεύνηση (Exploration), Προγραμματισμός (Planning), Επαναλήψεις (Iterations to release phase), Παραγωγή (Productionizing), Συντήρηση (Maintenance) και Θάνατο (Death).

Στη φάση της διερεύνησης πραγματοποιείται η καταμέτρηση των γνωρίσματων και των διεργασιών που θα περιέχει το λογισμικό. Οι πελάτες, στην φάση αυτή, είναι υποχρεωμένοι να συμμετάσχουν και να γράψουν σε «κάρτες ιστορίας» (story cards) τις απαιτήσεις που θα υλοποιηθούν στο λογισμικό, ενώ η ομάδα ανάπτυξης εξοικειώνεται με τα εργαλεία, τις δεξιότητες και τις τεχνικές, που θα γίνουν. Μετά τον ορισμό των τεχνολογιών που θα πραγματοποιηθούν, υλοποιείται ένα «πρωτότυπο» του λογισμικού.

Στη φάση του προγραμματισμού ορίζονται οι προδιαγραφές (ιστορίες) που θα υλοποιηθούν αρχικά και το τι θα περιλαμβάνει η πρώτη δημοσίευση του λογισμικού. Η ομάδα ανάπτυξης κρίνει τις ιστορίες, γίνεται υπολογισμός του φόρτου εργασίας για κάθε μια ιστορία και στη συνέχεια δημιουργείται ένα χρονοδιάγραμμα (πρόγραμμα) ενασχόλησης.



Εικόνα 7. Η φιλοσοφία της ευέλικτης μεθόδου του ακραίου προγραμματισμού **(Beck, 2004)**.

Στη φάση των επαναλήψεων πραγματοποιείται η διαίρεση του λογισμικού που έχει αποφασιστεί στην προγενέστερη φάση. Η χρονική διάρκεια της επανάληψης συνήθως διαρκεί από μια έως τέσσερις εβδομάδες. Η πρώτη επανάληψη περιέχει τον ορισμό της αρχιτεκτονικής του λογισμικού και μετέπειτα προηγείται η βελτίωση του συστήματος με

την εισαγωγή των απαιτήσεων (ιστοριών). Μόλις ολοκληρωθεί κάθε επανάληψη ελέγχεται το προϊόν από τον πελάτη. Κάθε επανάληψη περατώνεται με την παραγωγή της έκδοσης που υλοποιήθηκε κατά την επανάληψη.

Στη φάση της παραγωγής γίνονται οι αξιολογήσεις και οι έλεγχοι, ώστε να εξασφαλιστεί η αποδοτικότητα και αποτελεσματικότητα του λογισμικού, πριν αυτό δοθεί στον πελάτη. Στην φάση αυτή εντοπίζονται μεταβολές που πρέπει να εφαρμοστούν και κρίνεται, αν θα γίνουν στην επόμενη επανάληψη.

Στη φάση της συντήρησης καταχωρούνται νέες απαιτήσεις και προτάσεις, ώστε να μπορεί η ανάπτυξη τους να γίνει σε μεταγενέστερες εκδόσεις του λογισμικού.

Στη φάση του θανάτου του λογισμικού, έχουν ολοκληρωθεί όλες οι εργασίες ανάπτυξης του προϊόντος και οι πελάτες δεν έχουν άλλες απαιτήσεις (ιστορίες) για ανάλυση. Αυτό σημαίνει πώς το λογισμικό βρίσκεται στην τελική του μορφή και έχει συμπεριλάβει το σύνολο των απαιτήσεων. Οι λειτουργίες – διεργασίες καταγράφονται σε έγγραφο (τεκμηρίωση) του λογισμικού. Στη φάση αυτή το προϊόν δεν επιδέχεται άλλες αλλαγές, είτε στην αρχιτεκτονική του, είτε στην υλοποίηση, είτε στον κώδικα. Υπάρχει περίπτωση η φάση αυτή να ολοκληρωθεί νωρίτερα, αν το προϊόν δεν καλύπτει τις επιθυμητές απαιτήσεις του πελάτη.

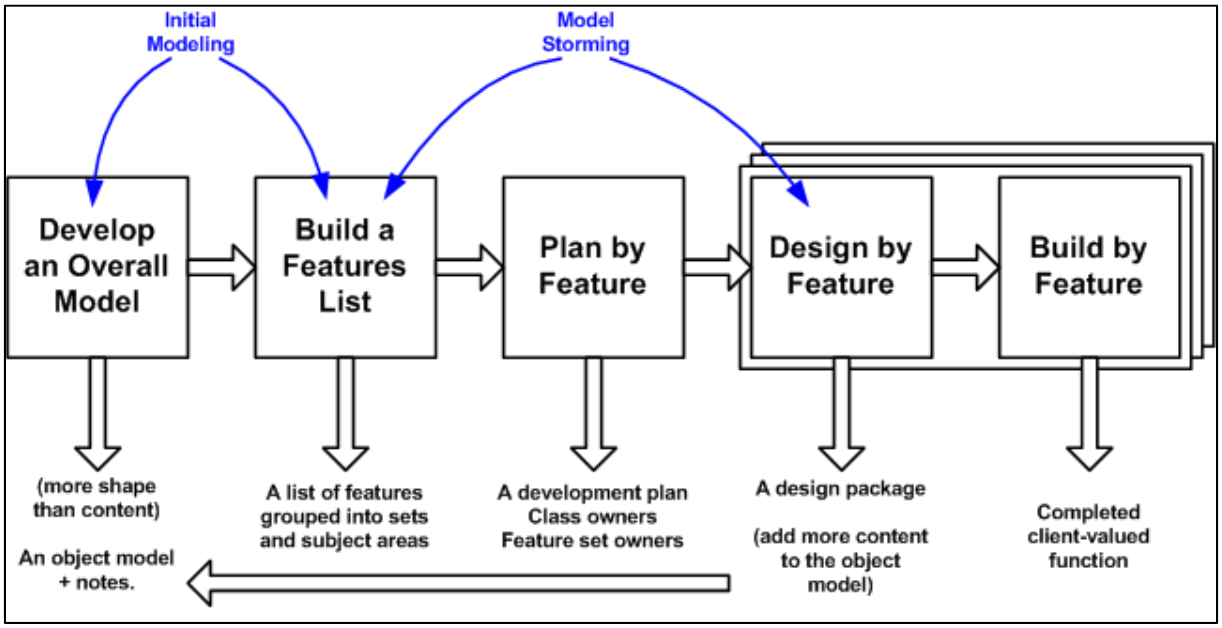
Ο ακραίος προγραμματισμός αποτελεί μία σύνθεση αντιλήψεων και πρακτικών που στηρίζονται σε ήδη καταξιωμένες μεθοδολογίες. Αποβλέπει στην υλοποίηση ενός επιτυχούς λογισμικού, παρά την αοριστία στην οριοθέτηση των προδιαγραφών και τις αβεβαιότητες στις αξιώσεις του πελάτη.

2.3.3 Η μέθοδος ανάπτυξης με βάση τα χαρακτηριστικά (Feature Driven Development)

Η ανάπτυξη με βάση τα χαρακτηριστικά (Feature Driven Development, FDD) εμπεριέχει πέντε φάσεις από τις οποίες διέρχεται η ανάπτυξη του λογισμικού (Εικόνα 8). Οι τρεις πρώτες φάσεις γίνονται στην έναρξη του έργου, ενώ οι δύο μεταγενέστερες φάσεις αποτελούν το επαναληπτικό τμήμα της υλοποίησης, όπου και εμφανίζονται οι αρχές της

ευέλικτης ανάπτυξης με ταχεία ενσωμάτωση τυχόν αλλαγών, όσον αφορά τις απαιτήσεις του λογισμικού. Η μεθοδολογία FDD περικλείει συχνές αξιολογήσεις του λογισμικού από τον πελάτη και διεξοδική έλεγχο του λογισμικού.

Η μεθοδολογία αυτή εμφανίστηκε για πρώτη φορά στην υλοποίηση ενός σύνθετου τραπεζικού λογισμικού στα τέλη της δεκαετίας του '90. Σε αντιπαράθεση με τις υπόλοιπες μεθοδολογίες δεν εφαρμόζει ακέραια την διαδικασία υλοποίησης, αλλά βασίζεται κυρίως στη φάση του σχεδιασμού και της υλοποίησης (Palmer & Felsing, 2002).



Εικόνα 8. Η μέθοδος ανάπτυξης με βάση τα χαρακτηριστικά (Palmer & Felsing, 2002).

Παρακάτω δίνονται οι φάσεις υλοποίησης της μεθόδου:

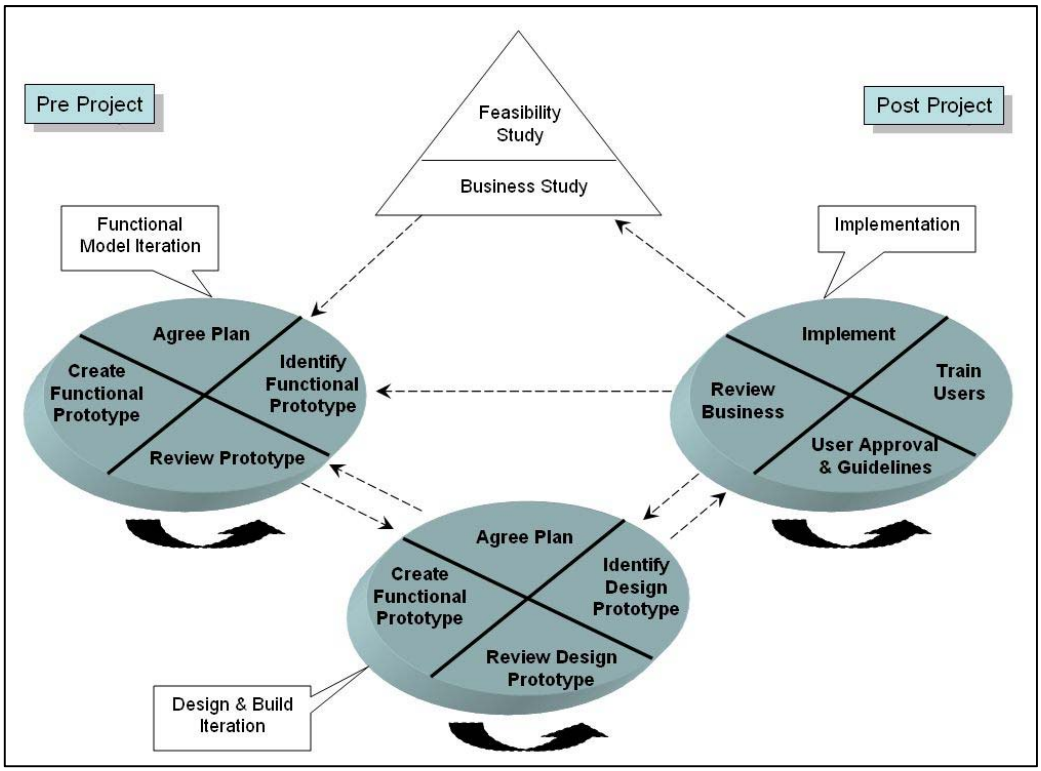
1. *Ανάπτυξη ενός γενικού μοντέλου (Develop an Overall Model):* Με την εκκίνηση της φάσης πραγματοποιείται μια αρχική εξοικείωση με τις πτυχές, το πλάνο και τις απαιτήσεις του λογισμικού. Είναι πιθανόν να συμπεριληφθούν επιπλέον πιστοποιημένες απαιτήσεις, όπως οι περιπτώσεις χρήσης και οι λειτουργικές προδιαγραφές. Διενεργείται ένα αναλυτικό «πέρασμα» (walkthrough) του λογισμικού και ο επικεφαλής σχεδιαστής (software architect) πληροφορείται

λεπτομερειακώς σχετικά με το λογισμικού. Το λογισμικό χωρίζεται σε επιπλέον υποσυστήματα και γίνονται και άλλα «περάσματα».

2. *Σχηματισμός μιας λίστας χαρακτηριστικών (Build a Features List):* Δημιουργείται ένας κατάλογος κατηγοριών με τα χαρακτηριστικά για να υποστηριχθούν από τις απαιτήσεις.
3. *Σχεδιασμός με βάση τα χαρακτηριστικά (Plan by Feature):* Η ομάδα υλοποίησης κατατάσσει τα γνωρίσματα του λογισμικού σύμφωνα με τις προτεραιότητες και τις αλληλοσυνδέσεις. Επιπροσθέτως, τα τμήματα λογισμικού που κατηγοριοποιήθηκαν στην πρώτη φάση δίδονται σε μεμονωμένους προγραμματιστές (class owners). Επίσης, το χρονοδιάγραμμα και οι παραδόσεις του λογισμικού προγραμματίζονται για τα χαρακτηριστικά γνωρίσματα.
4. *Σχεδιασμός και Υλοποίηση του λογισμικού οδηγούμενα από τα χαρακτηριστικά (Design & Build by Feauture):* Στην φάση αυτή επιλέγονται σαφώς τα χαρακτηριστικά γνωρίσματα για σχεδιασμό και υλοποίηση. Οι φάσεις αυτές έχουν επαναληπτικό χαρακτήρα, εξαγόμενο των οποίων είναι ο προγραμματισμός των χαρακτηριστικών, που προκρίνονται σε κάθε γύρο. Μια ομάδα προγραμματιστών αναλαμβάνει να τα υλοποιήσει μέσα από επαναλήψεις. Ενδεχομένως, πολλές ομάδες μπορούν να εργάζονται συγχρόνως στην υλοποίηση ποικίλων χαρακτηριστικών. Μια επανάληψη μπορεί να διαρκέσει από μερικές μέρες έως δύο βδομάδες, το μέγιστο. Όταν περατωθεί μια επανάληψη τότε το λογισμικό αναβαθμίζεται με τις νέες λειτουργίες και τότε αναφύεται μια νέα έκδοση.

2.3.4 Η μέθοδος ανάπτυξης δυναμικών συστημάτων (Dynamic Systems Development)

Η μεθοδολογία υλοποίησης δυναμικών συστημάτων (Dynamic Systems Development, DSD), παρουσιάστηκε πρώτα στο Ηνωμένο Βασίλειο, εντός της δεκαετίας του ‘90. Είναι ένας συνδυασμός, αλλά ταυτόχρονα και διεύρυνση της ταχείας προτυποποίησης και πρακτικών επαναληπτικής υλοποίησης (Stapleton, 1997). Πρωταρχικός σκοπός της DSD είναι να ορισθεί ο χρόνος, οι πόροι, και μετέπειτα να ταιριάξει η λειτουργικότητα αντίστοιχα και όχι να προσδιοριστεί το ποσό της λειτουργικότητας στο λογισμικό, και έπειτα να ταιριάξει ανάλογα ο χρόνος και οι πόροι για την εκπλήρωση αυτής της λειτουργικότητας (DSDM Consortium, 2012) (Εικόνα 9).



Εικόνα 9. Η μέθοδος ανάπτυξης δυναμικών συστημάτων **(Stapleton, 1997)**.

Ο Martin Fowler, ένας από τους πρωτεργάτες της διακήρυξης των ευέλικτων μεθόδων, αποδέχεται πώς «Η DSD μέθοδος είναι αξιοσημείωτη για το λόγο ότι ακολουθεί την υποδομή των παραδοσιακών μεθόδων ανάπτυξης λογισμικού, ενώ παράλληλα ακολουθεί τις αρχές του ευέλικτου τρόπου ανάπτυξης» (Fowler, 2005).

Οι αρχικές δύο φάσεις γίνονται σειριακά και μόνο μια φορά. Οι άλλες, όπου και υλοποιείται το λογισμικό, είναι επαναληπτικές και αυξητικές. Ένα αξιόλογο γνώρισμα της μεθόδου αυτής είναι ο χρόνος. Οι επαναλήψεις θεωρούνται ως χρονικά κατώφλια. Ένα χρονικό κατώφλι έχει ορισμένα όρια και η επανάληψη είναι αναγκαίο να περατωθεί μέσα στα χρονικά αυτά πλαίσια. Ένα χρονικό κατώφλι διαρκεί από μερικές ημέρες έως λίγες βδομάδες. Οι φάσεις υλοποίησης του λογισμικού με βάση την μεθοδολογία DSD αναφέρονται παρακάτω:

1. *Μελέτη σκοπιμότητας (Feasibility study):* Παίρνεται η απόφαση της χρησιμοποίησης ή όχι της μεθοδολογίας DSD. Αυτό καθορίζεται από την κατηγορία του λογισμικού και άλλες οργανωτικές πτυχές και ιδιαιτέρως τον

ανθρώπινο συντελεστή. Στη φάση αυτή προκύπτουν δύο αποτελέσματα εργασίας, μια έκθεση σκοπιμότητας και ένα γενικό σχέδιο για την υλοποίηση.

2. *Επιχειρησιακή μελέτη (Business study):* Η συνιστάμενη προσέγγιση σε αυτή τη φάση είναι να εξηγηθεί λεπτομερώς η επιχειρηματική πτυχή του λογισμικού. Τα πρωταρχικά επακόλουθα αυτής της φάσης είναι ο καθορισμός της αρχιτεκτονικής του λογισμικού και ένα πρωτότυπο διάγραμμα αυτού.
3. *Λειτουργική πρότυπη επανάληψη (Functional model iteration):* Είναι η αρχική επαναληπτική κατάσταση. Περιέχει την εξήγηση, την κωδικοποίηση και την προτυποποίηση. Τα επακόλουθα που συγκεντρώνονται χρησιμοποιούνται, ώστε να εξελιχθούν τα πρωτότυπα. Βασικό επακόλουθο της φάσης είναι ένα λειτουργικό πρωτότυπο.
4. *Σχεδιασμός και δοκιμή της επανάληψης (Design and build iteration):* Το λογισμικό υλοποιείται πραγματικά σε αυτή τη φάση. Ο σχεδιασμός και η λειτουργική εκτίμηση των πρωτοτύπων παράγονται από τους χρήστες και η περαιτέρω υλοποίηση συναρτάται στις κριτικές των χρηστών.
5. *Εφαρμογή (Implementation):* Το σύστημα αξιολογείται από τους χρήστες. Η κριτική των χρηστών, σύμφωνα με την διαχείριση του λογισμικού, λαμβάνεται υπόψη. Εγχειρίδια χρήστη και μια έκθεση ανασκόπησης του λογισμικού διατίθενται στους χρήστες. Πάντως, η επαναληπτική και αυξητική φάση της μεθοδολογίας, η οποία δηλώνει την συντήρηση μπορεί να λογιστεί ως συνεχιζόμενη υλοποίηση. Αντί να ολοκληρωθεί το λογισμικό με ένα κύκλο, μπορεί να επανέλθει σε κάποια από τις προγενέστερες φάσεις, έτσι ώστε να βελτιωθεί ως προς τις απαιτήσεις.

2.3.5 Η λιτή ανάπτυξη λογισμικού (Lean Software Development)

Η λιτή ανάπτυξη λογισμικού (LSD) είναι μια εφαρμογή των αρχών και πρακτικών της λιτής παραγωγής στον τομέα της ανάπτυξης λογισμικού. Η λιτή ανάπτυξη λογισμικού προσφέρει ένα σταθερό εννοιολογικό πλαίσιο, αξίες και αρχές, καθώς και καλές πρακτικές που προέρχονται από την εμπειρία, που υποστηρίζουν και οι ευέλικτες αρχές ανάπτυξης λογισμικού. Οι αρχές, στις οποίες βασίζεται η λιτή ανάπτυξη λογισμικού είναι:

Εξάλειψη της σπατάλης. Οι υποστηρικτές της λιτής σκέψης θεωρούν κάθε δραστηριότητα που δεν προσθέτει αξία άμεσα στο λογισμικό ως σπατάλη. Οι τρεις μεγαλύτερες πηγές σπατάλης στην ανάπτυξη λογισμικού είναι η προσθήκη λανθασμένων χαρακτηριστικών, η μείωση αξίας και το ξεπέρασμα οργανωτικών ορίων (ιδίως μεταξύ των ενδιαφερομένων και των αναπτυξιακών ομάδων). Για να μειωθούν τα λάθη είναι κρίσιμο να επιτρέπεται στις ομάδες υλοποίησης των έργων να αυτό-οργανώνονται και να λειτουργούν με τρόπο που να αντικατοπτρίζει το έργο που προσπαθούν να επιτύχουν.
Η ανάπτυξη με βάση την ποιότητα. Η διαδικασία ανάπτυξης λογισμικού δεν πρέπει να επιτρέπει την εμφάνιση ελαττωμάτων. Στην περίπτωση εμφάνισης ελαττωμάτων, θα πρέπει να διαμορφωθεί η εργασία με τέτοιο τρόπο, ώστε να διορθωθούν τα λάθη, να επικυρωθεί το αποτέλεσμα, να διορθωθούν τυχόν προβλήματα και η διαδικασία να επαναλαμβάνεται. Η επιθεώρηση και η αντιμετώπιση ελαττωμάτων σε κάποια στιγμή στο μέλλον γίνεται αποτελεσματική. Οι ευέλικτες μέθοδοι ανάπτυξης λογισμικού, που δημιουργούν ποιότητα στο λογισμικό περιλαμβάνουν εξελιγμένες δοκιμές και ομαδικές πρακτικές ανάπτυξης.
Η δημιουργία γνώσης. Ο σχεδιασμός είναι χρήσιμος, αλλά και η μάθηση είναι απαραίτητη. Η προώθηση στρατηγικών, όπως η επαναληπτική διαδικασία, βοηθά τις ομάδες να ανακαλύψουν τι ακριβώς θέλουν πραγματικά και να στηριχθούν πάνω σε αυτές τις γνώσεις. Είναι επίσης σημαντικό για μια ομάδα να σκέφτεται τακτικά τι κάνει και στη συνέχεια να ενεργεί για να βελτιώσει την προσέγγισή της.
Η αποτροπή της δέσμευσης. Δεν είναι απαραίτητο να ξεκινήσει η ανάπτυξη λογισμικού καθορίζοντας από την αρχή τις προδιαγραφές. Μπορεί να αναπτυχθεί το λογισμικό αποτελεσματικά μέσω ευέλικτων αρχιτεκτονικών, που είναι ανεκτικές στις αλλαγές να σχεδιαστούν μη αναστρέψιμες αποφάσεις μέχρι την τελευταία στιγμή. Συχνά, η αναβολή της αφοσίωσης στην τελευταία πιο υπεύθυνη στιγμή απαιτεί τη δυνατότητα να συνδυάσει ο σχεδιαστής του έργου επιχειρηματικά σενάρια με δυνατότητες ανάπτυξης πολλαπλών εφαρμογών από πολλαπλά προγράμματα.
Η ταχύτερη παράδοση. Είναι δυνατή η γρήγορη παράδοση λογισμικών υψηλής ποιότητας. Περιορίζοντας την εργασία μιας ομάδας στην ικανότητά της, η οποία αντανακλάται από την ταχύτητα της ομάδας, μπορεί να δημιουργηθεί μια αξιόπιστη και επαναλαμβανόμενη ροή εργασίας. Μια αποτελεσματική οργάνωση δεν απαιτεί από τις ομάδες να κάνουν περισσότερα από όσα είναι ικανές να κάνουν, αλλά να αυτό-

οργανώνονται και να καθορίζουν τι μπορούν να επιτύχουν. Ο περιορισμός αυτών των ομάδων στην παροχή τακτικών λύσεων, τους παρακινεί να παραμείνουν συγκεντρωμένοι στη συνεχή προστιθέμενη αξία στο προϊόν.

Ο σεβασμός στο πελάτη. Το ανταγωνιστικό πλεονέκτημα κερδίζεται από τους εμπλεκόμενους ανθρώπους. Η συνέπεια της διοίκησης των εταιρειών ανάπτυξης λογισμικού που επικεντρώνεται στην παροχή κινήτρων και δυνατοτήτων στις ομάδες έργων και όχι στον έλεγχο τους, οδηγεί στην κερδοφορία και στην μέγιστη αποτελεσματικότητα.

Η βελτιστοποίηση του προϊόντος. Σημαντική είναι η κατανόηση των επιχειρηματικών διεργασιών υψηλού επιπέδου στις οποίες τα μεμονωμένα έργα υποστηρίζουν διαδικασίες που συχνά εμπεριέχουν πολλαπλά συστήματα. Η ολοκληρωμένη διαχείριση έργων με αλληλένδετα συστήματα, μπορούν να παραδώσουν ένα ολοκληρωμένο προϊόν στους πελάτες. Οι μετρήσεις της ποιότητας του θα πρέπει να εξετάζουν το πόσο καλά προσφέρεται η επιχειρηματική αξία.

2.3.6 Η μέθοδος του καθαρού κρυστάλλου (Crystal clear)

Η οικογένεια των «Κρυστάλλινων» μεθοδολογιών περιλαμβάνει έναν αριθμό από διαφορετικές μεθοδολογίες, ώστε να επιλέγεται κάθε φορά η καταλληλότερη μεθοδολογία για κάθε έργο. Κάθε μέλος της οικογένειας των κρυστάλλινων μεθοδολογιών είναι μαρκαρισμένο με ένα χρώμα που δείχνει την «ένταση» της μεθοδολογίας, δηλαδή όσο σκοτεινότερο το χρώμα, τόσο πιο εντατικά εφαρμόζεται η μεθοδολογία. Η προσέγγιση του «Κρυστάλλου» προτείνει το κατάλληλο χρώμα για κάθε έργο βασισμένο στο μέγεθος και την κρισιμότητά του.

Υπάρχουν ορισμένοι κανόνες, χαρακτηριστικά γνωρίσματα και αξίες που είναι κοινά σε όλες τις μεθόδους στην οικογένεια των κρυστάλλινων μεθοδολογιών. Καταρχήν, τα προγράμματα χρησιμοποιούν πάντα επαυξητικούς κύκλους ανάπτυξης με ένα μέγιστο χρόνο τεσσάρων μηνών, αλλά κατά προτίμηση μεταξύ ενός και τριών μηνών. Ιδιαίτερη έμφαση δίνεται στην επικοινωνία και τη συνεργασία των ανθρώπων. Οι μεθοδολογίες κρυστάλλου δεν περιορίζουν οποιεσδήποτε πρακτικές ανάπτυξης, εργαλεία ή προϊόντα εργασίας, επίσης επιτρέπουν την υιοθέτηση, παραδείγματος χάριν, του XP και του Scrum. Επιπλέον, η προσέγγιση κρυστάλλου ενσωματώνει στόχους για τα ενδιάμεσα προϊόντα

εργασίας και τις συμβάσεις μέσα σε μια μεθοδολογία για κάθε λογισμικό και καθώς αυτό εξελίσσεται. Υπάρχουν αυτήν την περίοδο τρεις κύριες μεθοδολογίες κρυστάλλου που έχουν δομηθεί: Το Καθαρό Κρύσταλλο (Crystal Clear), το Πορτοκάλι Κρύσταλλο (Crystal Orange) και ο Πορτοκαλής Ιστός Κρυστάλλου (Crystal Orange Web) (Cockburn, 2007).

Το «Καθαρό Κρύσταλλο» σχεδιάστηκε για τα πολύ μικρά έργα ανάπτυξης λογισμικού, περιλαμβάνοντας μέχρι έξι υπεύθυνους για την ανάπτυξη. Εντούτοις, με κάποια επέκταση στην επικοινωνία και τη δοκιμή μπορεί να εφαρμοστεί επίσης και σε μεγαλύτερα έργα. Μια ομάδα που χρησιμοποιεί το «Καθαρό Κρύσταλλο» πρέπει να βρίσκεσαι σε ένα κοινό γραφείο λόγω των περιορισμών στην επικοινωνία (Cockburn, 2007).

Στο «Καθαρό Κρύσταλλο», οι κύριοι ρόλοι που απαιτούνται να είναι χωριστά πρόσωπα είναι (Cockburn, 2007): ο χορηγός, ο ανώτερος σχεδιαστής-προγραμματιστής, ο σχεδιαστής-προγραμματιστής και ο χρήστης. Αυτοί οι ρόλοι ενσωματώνουν πολλαπλούς υπό-ρόλους. Παραδείγματος χάριν, ο σχεδιαστής-προγραμματιστής αποτελείται από το σχεδιαστή επιχειρησιακής κατηγορίας, το προγραμματιστή, τον υπεύθυνο τεκμηρίωση και τον ελεγκτή (Cockburn, 1998). Οι υπό-ρόλοι που μπορούν να οριστούν σε άλλους ρόλους είναι αυτοί του συντονιστή, του επιχειρησιακού εμπειρογνώμονα και του συντονιστή των απαιτήσεων (Cockburn, 2007). Ο επιχειρησιακός εμπειρογνώμονας αντιπροσωπεύει μια επιχειρησιακή άποψη στο πρόγραμμα, κατέχοντας την γνώση για το συγκεκριμένο επιχειρησιακό πλαίσιο. Πρέπει να είναι σε θέση να φροντίσει το επιχειρησιακό σχέδιο (Cockburn, 1998).

Κάθε ομάδα αποτελείται από τουλάχιστον έναν επιχειρησιακό σχεδιαστή, σχεδιαστή UI, δύο έως τρεις σχεδιαστές-προγραμματιστές, έναν σχεδιαστή βάσεων δεδομένων και ενδεχομένως, έναν ελεγκτή. Επίσης, οι τεχνικοί βοηθοί μπορούν να απαιτηθούν από μια ομάδα. Ο ρόλος ύπαρξης των ομάδων είναι να είναι λειτουργικές, ώστε να μειωθούν τα παραδοτέα και να ενισχυθεί η επικοινωνία (Cockburn, 2007).

Όλες οι μεθοδολογίες κρυστάλλου περιλαμβάνουν διάφορες πρακτικές, όπως η επαυξητική ανάπτυξη. Στην περιγραφή του Πορτοκαλί Κρυστάλλου (Cockburn, 1998), η

αύξηση περιλαμβάνει τις δραστηριότητες, όπως: η τμηματοποίηση (staging), ο έλεγχος (monitoring), η αναθεώρηση (reviewing), μαζί με τον παραλληλισμό (parallelism) και τη ροή (flux). Επίσης άλλες δραστηριότητες και πρακτικές μπορούν να προσδιοριστούν. Σε αυτές συμπεριλαμβάνονται οι ακόλουθες:

Τμηματοποίηση

Η τμηματοποίηση περιλαμβάνει τον προγραμματισμό της επόμενης προσθήκης του λογισμικού. Πρέπει να σχεδιαστεί, έτσι ώστε να παράγει μια έκδοση κάθε τρεις ή τέσσερις μήνες στο μέγιστο. Ένα πρόγραμμα ενός έως τριών μηνών προτιμάται. Η ομάδα επιλέγει τις απαιτήσεις, που εφαρμόζονται στην αύξηση και σχεδιάζει τι αισθάνεται ότι είναι σε θέση να παραδώσει (Cockburn, 1998).

Έκδοση και αναθεώρηση

Κάθε αύξηση περιλαμβάνει διάφορες επαναλήψεις. Κάθε επανάληψη περιλαμβάνει τις ακόλουθες δραστηριότητες: κατασκευή, επίδειξη και αναθεώρηση των στόχων της αύξησης (Cockburn, 1998).

Έλεγχος

Η πρόοδος ελέγχεται σχετικά με τα προϊόντα των ομάδων κατά τη διάρκεια της διαδικασίας ανάπτυξης, όσον αφορά την πρόοδο και τη σταθερότητά τους. Η πρόοδος μετριέται από τα κύρια σημεία (η έναρξη, αναθεώρηση Α, αναθεώρηση Β, δοκιμή, παράδοση) και τα στάδια σταθερότητας (με μεγάλη διακύμανση, αρκετή διακύμανση και αρκετά σταθερό για να αναθεωρηθεί). Ο έλεγχος απαιτείται και στον Καθαρό κρύσταλλο και στον Πορτοκάλι κρύσταλλο (Cockburn, 1998).

Παραλληλισμός και ροή

Μόλις ο έλεγχος της σταθερότητας δώσει το αποτέλεσμα «αρκετά σταθερό για να αναθεωρηθεί» για τα προϊόντα, ο επόμενος στόχος μπορεί να ξεκινήσει. Στο πορτοκάλι κρύσταλλο, αυτό σημαίνει ότι οι πολλαπλές ομάδες μπορούν να συνεχίσουν με το μέγιστο παραλληλισμό. Για να το εξασφαλίσουν αυτό, οι ομάδες ελέγχου και αρχιτεκτονικής αναθεωρούν τα σχέδια εργασίας, τη σταθερότητα και το συγχρονισμό τους (Cockburn, 1998).

Ολιστική στρατηγική ποικιλομορφίας

Στον Πορτοκάλι κρύσταλλο περιλαμβάνεται μια μέθοδος που καλείται ολιστική στρατηγική ποικιλομορφίας, η οποία διαχωρίσει τις μεγάλες λειτουργικές ομάδες σε μικρότερες. Η κεντρική ιδέα αυτής είναι να περιληφθούν οι πολλαπλές ειδικότητες σε μια ενιαία ομάδα. Η ολιστική στρατηγική ποικιλομορφίας επιτρέπει επίσης τις μικρές ομάδες με την απαραίτητη ειδική τεχνογνωσία, και εξετάζει επίσης ζητήματα, όπως: τον εντοπισμό των ομάδων, την επικοινωνία, την τεκμηρίωση και τον συντονισμό των πολλαπλών ομάδων (Cockburn, 1998).

Τεχνική συντονισμού της μεθοδολογία

Η τεχνική συντονισμού της μεθοδολογίας είναι μια από τις βασικές τεχνικές του καθαρού και του πορτοκαλί κρυστάλλου. Χρησιμοποιεί τις συνεντεύξεις προγράμματος και τα εργαστήρια ομάδων για να προσαρμόζει μια συγκεκριμένη μεθοδολογίας κρυστάλλου σε κάθε μεμονωμένο πρόγραμμα (Cockburn, 2007). Μια από τις κεντρικές ιδέες της επαυξητικής ανάπτυξης είναι να επιτρέψει την διαμόρφωση ή βελτίωση της διαδικασία ανάπτυξης (Cockburn, 1998). Σε κάθε αύξηση, το πρόγραμμα μπορεί να μάθει και να χρησιμοποιήσει τη γνώση που έχει αποκτήσει στην επόμενη αύξηση.

Επιθεωρήσεις των χρηστών

Δύο εξετάσεις χρηστών προτείνονται για το καθαρό κρύσταλλο ανά απελευθέρωση (Cockburn, 2007). Στο πορτοκάλι κρύσταλλο, οι αναθεωρήσεις χρηστών πρέπει να οργανωθούν τρεις φορές για κάθε αύξηση (Cockburn, 1998).

Εργαστήρια ανασκόπησης

Και το καθαρό και το πορτοκάλι κρύσταλλο περιλαμβάνουν τον κανόνα ότι μια ομάδα πρέπει να συμμετάσχει σε εργαστήρια ανασκόπησης (Cockburn, 2007).

2.3.7 Η ανοικτή ενοποιημένη διεργασία ανάπτυξης λογισμικού (Open Unified Process)

Η ενοποιημένη διαδικασία (Unified Process) αποτελείται και αυτή από μία σειρά επαναλήψεων εργασιών που βαθμιαία υλοποιούν το λογισμικό. Κάθε επαναληπτική σειρά εργασιών περατώνεται με την δημιουργία ενός πρωτοτύπου και την αξιολόγησή του από τον πελάτη. Κάθε επανάληψη περιέχει τέσσερις φάσεις: σύλληψη (inception),

επεξεργασία (elaboration), κατασκευή (construction) και μετάβαση (transition). Κάθε φάση διαιρείται σε επαναλήψεις.

Κάθε επανάληψη περατώνεται με ένα νέο πρωτότυπο του λογισμικού και κάθε πρωτότυπο είναι ένα προϊόν έτοιμο για αξιολόγηση από τον πελάτη. Απαρτίζεται από το τμήμα του κώδικα μαζί με άλλα στοιχεία, τα οποία μπορούν να μεταγλωττιστούν και να εκτελεστούν μαζί με τα εγχειρίδια χρήσης. Πέρα από τα προηγούμενα, το τελικό προϊόν πρέπει να εκπληρώνει τις απαιτήσεις όχι μόνο των πελατών, άλλα και των χρηστών και όλων όσων εμπλέκονται με το λογισμικό. Το λογισμικό πρέπει να συγκροτεί κάτι παραπάνω από απλό κώδικα ο οποίος εκτελείται. Η τελική έκδοση του λογισμικού περιέχει τις απαιτήσεις, τις περιπτώσεις χρήσης, τις μη λειτουργικές απαιτήσεις και τις περιπτώσεις δοκιμής. Επίσης περιέχει την αρχιτεκτονική και τα οπτικά μοντέλα, δηλαδή τα διαγράμματα που κατασκευάστηκαν με την χρήση της ενοποιημένης γλώσσας μοντελοποίησης (UML). Αναλυτικά, περιέχει όλα τα συστατικά στοιχεία κώδικα που προαναφέρθηκαν, γιατί αυτά θα δώσουν την ικανότητα σε όλους τους εμπλεκομένους, όπως πελάτες, χρήστες, αναλυτές, σχεδιαστές, προγραμματιστές και αυτούς που θα κάνουν τις δοκιμές, να ορίσουν, να σχεδιάσουν, να ολοκληρώσουν και να δοκιμάσουν το λογισμικό, αντίστοιχα. Επιπλέον, θα αφήσουν τους εμπλεκομένους να μεταχειριστούν και να βελτιώσουν το σύστημα περαιτέρω. Οι εκτελέσιμες διεργασίες του λογισμικού είναι τα πιο βασικά συστατικά, από την πλευρά των χρηστών. Αυτό συμβαίνει, γιατί το περιβάλλον μεταβάλλεται. Λειτουργικά συστήματα και συστήματα βάσεων δεδομένων εξελίσσονται. Καθώς, η υλοποίηση ενός λογισμικού βελτιώνεται και το πεδίο γίνεται πιο κατανοητό, οι απαιτήσεις μπορεί να αλλάξουν. Αναλυτικότερα, αποτελεί κάτι βασικό στην υλοποίηση του λογισμικού, δηλαδή πιστοποιεί ότι οι απαιτήσεις αλλάζουν. Κάποια στιγμή, οι προγραμματιστές θα πρέπει να αρχίσουν ένα νέο κύκλο εργασιών και οι πελάτες πρέπει να τον αξιολογήσουν. Για να εκτελέσουν εύστοχα τον επόμενο κύκλο επανάληψης, οι προγραμματιστές χρειάζονται όλα τα συστατικά του λογισμικού που είναι οι εξής:

- Ένα μοντέλο περιπτώσεων χρήσης με όλες τις περιπτώσεις χρήσης και τις αλληλεξαρτήσεις τους με τους χρήστες.

- Ένα μοντέλο ανάλυσης, που επιτελεί διπλό σκοπό: αναβαθμίσει τις περιπτώσεις χρήσης με μεγαλύτερες λεπτομέρειες και μία αρχική έκδοση της συμπεριφοράς του λογισμικού σε ένα άθροισμα αντικειμένων που θα διοχετεύουν αυτήν τη συμπεριφορά.
- Ένα μοντέλο σχεδίασης που θα ορίζει τη δομή του λογισμικού με τα συστήματά του, κλάσεις και διασυνδέσεις, και τις περιπτώσεις χρήσης, πλήρεις από τις συνεργασίες μεταξύ των υποσυστημάτων, των κλάσεων και των διασυνδέσεων.
- Ένα μοντέλο υλοποίησης, το οποίο θα περιέχει τα συστατικά και τον καταμερισμό των κλάσεων σε συστατικά.
- Ένα μοντέλο διάταξης που θα ορίζει τους φυσικούς κόμβους υπολογιστών και την κατανομή των συστατικών στους κόμβους αυτούς.
- Ένα μοντέλο δοκιμής που θα αναπαριστά τις περιπτώσεις δοκιμής που θα βεβαιώνουν την ικανοποίηση των περιπτώσεων χρήσης.
- Την αναπαράσταση της αρχιτεκτονικής.

Το λογισμικό επιπλέον πρέπει να έχει ένα μοντέλο πεδίου που περιγράφει το επιχειρηματικό περιβάλλον του συστήματος. Όλα αυτά τα μοντέλα αλληλεξαρτώνται. Μαζί δημιουργούν το λογισμικό ως σύνολο. Τα συστατικά σε ένα μοντέλο αφήνουν «αποτυπώματα», τα οποία αποκαλύπτουν την βελτίωση του στοιχείου στα μετέπειτα ή προγενέστερα μοντέλα.

2.4 Διαφορές μεταξύ παραδοσιακών και ευέλικτων μεθόδων ανάπτυξης λογισμικού

Οι ευέλικτες μέθοδοι ανάπτυξης λογισμικού δίνουν έμφαση στις ομάδες, το λογισμικό εργασίας, τη συνεργασία με τους πελάτες και την ανταπόκριση στις αλλαγές, ενώ οι παραδοσιακές μέθοδοι επικεντρώνονται σε συμβάσεις, σχέδια, διαδικασίες, έγγραφα και εργαλεία (Fowler & Highsmith, 2001). Οι Nerur, et al., (2005) εξηγούν τις κυριότερες διαφορές μεταξύ παραδοσιακών και ευέλικτων μεθόδων ανάπτυξης λογισμικού (Πίνακας 1).

Πίνακας 1. Οι διαφορές των παραδοσιακών και ευέλικτων τρόπων ανάπτυξης λογισμικού.

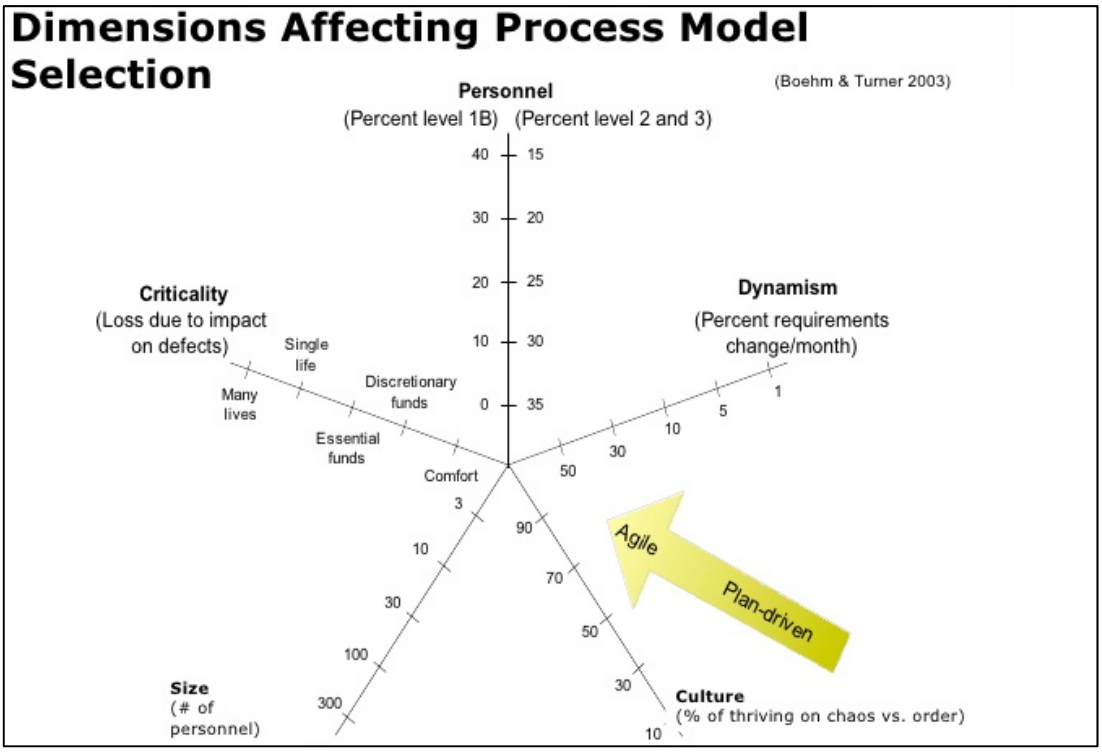
	Παραδοσιακές	Ευέλικτες
Θεμελιώδεις υποθέσεις	Λογισμικό πλήρως προσδιορισμένο, προβλέψιμο που μπορεί να κατασκευαστεί με σχολαστικό και εκτεταμένο σχεδιασμό	Υψηλής ποιότητας, λογισμικό μπορεί να αναπτυχθεί από μικρές ομάδες χρησιμοποιώντας τις αρχές της συνεχούς βελτίωσης του σχεδιασμού και των δοκιμών που βασίζονται στην ταχεία ανάδραση και αλλαγή
Έλεγχος	Διεργασιοκεντρικός	Πελατοκεντρικός
Τρόπος διοίκησης	Εντολή & Έλεγχος	Ηγεσία & συνεργασία
Διαχείριση της γνώσης	Σαφής	Σιωπηρή
Αναθέσεις ρόλων	Ατομικός - ευνοεί την εξειδίκευση	Αυτό-οργανωτικές ομάδες - ενθαρρύνει την εναλλαγή ρόλων
Επικοινωνία	Επίσημη	Ανεπίσημη
Ρόλος του πελάτη	Σημαντικός	Κρίσιμος
Κύκλος του έργου	Καθοδηγούμενη από εργασίες ή δραστηριότητες	Καθοδηγούμενη από τις λειτουργίες του προϊόντος
Μοντέλο ανάπτυξης	Μοντέλο κύκλου ζωής (καταρράκτη, σπειροειδές ή συνδυασμός)	Μοντέλο εξελικτικής παράδοσης
Επιθυμητή οργανωτική μορφή / δομή	Μηχανιστική (γραφειοκρατική με υψηλή τυποποίηση)	Οργανική (ευέλικτη και συμμετοχική, ενθάρρυνση της συνεργατικής κοινωνικής δράσης)
Τεχνολογία	Χωρίς περιορισμούς	Προτίμηση στην αντικειμενοστραφή τεχνολογία

Οι Boehm & Turner, (2003) περιγράφουν τους πέντε άξονες, που χρησιμοποιούνται για τη διάκριση των διαφορών μεταξύ ευέλικτων και παραδοσιακών μεθόδων ανάπτυξης λογισμικού. Με τη χρήση των πέντε αξόνων είναι δυνατόν να αναγνωριστεί ποια μέθοδος ανάπτυξης λογισμικού μπορεί να είναι κατάλληλη για διάφορα έργα (Εικόνα 10). Παρακάτω στον Πίνακας 2 δίνεται μία πιο λεπτομερής εξήγηση των διαφορών μεταξύ ευέλικτων και παραδοσιακών μεθόδων σε πέντε άξονες.

Πίνακας 2. Οι πέντε κρίσιμοι παράγοντες ευελιξίας και σχεδιασμού (Boehm & Turner, 2003).

Παράγοντες	Παραδοσιακές	Ευέλικτες
Μέγεθος	Οι μέθοδοι εξελίχθηκαν για να χειρίζονται μεγάλα προϊόντα και ομάδες. δύσκολο να προσαρμοστούν σε μικρά έργα	Συνδυάζεται καλά με μικρά προϊόντα και ομάδες. Εμπιστοσύνη στην σιωπηρή δυνατότητα κλιμάκωσης των ορίων γνώσης
Κρίση	Οι μέθοδοι εξελίχθηκαν για να χειριστούν προϊόντα που απαιτούν κριτική ικανότητα.	Δεν έχει δοκιμαστεί σε προϊόντα κρίσιμα για την ασφάλεια. πιθανές δυσκολίες

	Δύσκολο να προσαρμοστεί αποτελεσματικά σε προϊόντα χαμηλής κριτικής ικανότητας	με απλό σχεδιασμό και έλλειψη τεκμηρίωσης
Δυναμισμός	Τα λεπτομερή σχέδια και ο σειριακός σχεδιασμός πλεονεκτούν σε σταθερό περιβάλλον, αλλά αποτελούν πηγή δαπανηρής επεξεργασίας σε δυναμικά περιβάλλοντα	Ο απλός σχεδιασμός και η συνεχής ανατροφοδότηση πλεονεκτούν σε δυναμικά περιβάλλοντα, αλλά αποτελούν πηγή δυνητικά δαπανηρών επανασχεδιασμών σε σταθερά περιβάλλοντα
Προσωπικό	Απαιτείται μια κρίσιμη μάζα σπάνιων εμπειρογνωμόνων κατά τη διάρκεια του ορισμού του έργου, αλλά μπορεί να λειτουργήσει με λιγότερους, αργότερα στο έργο-εκτός αν το περιβάλλον είναι ιδιαίτερα δυναμικό	Απαιτείται συνεχής παρουσία μίας κρίσιμης μάζας εμπειρογνωμόνων
Κουλτούρα	Ευδοκιμούν σε μια κουλτούρα όπου οι άνθρωποι αισθάνονται άνετα και εξουσιοδοτημένοι έχοντας τους ρόλους τους καθορισμένους από σαφείς πολιτικές	Ευδοκιμούν σε μια κουλτούρα όπου οι άνθρωποι αισθάνονται άνετα και εξουσιοδοτημένοι έχοντας πολλούς βαθμούς ελευθερίας. Ευδοκιμούν στο χάος



Εικόνα 10. Διαφορές μεταξύ παραδοσιακών και ευέλικτων μεθόδων ανάπτυξης λογισμικού (Πηγή: Helsinki University of Technology).

2.5 Η Διασφάλιση της Ποιότητας του Λογισμικού

Σύμφωνα με τον Pressman, (2005), η ποιότητα του λογισμικού είναι αποτέλεσμα της καλής διαχείρισης ενός έργου και της μηχανικής λογισμικού. Με τη χρήση της διασφάλισης ποιότητας είναι δυνατόν να δημιουργηθεί η υποδομή για την υποστήριξη μεθόδων ανάπτυξης λογισμικού, διαχείρισης έργων και ενεργειών ελέγχου ποιότητας. Σκοπός της διασφάλισης της ποιότητας είναι η παροχή στη διοίκηση και στην τεχνική ομάδα, δεδομένων, που είναι απαραίτητα για την επίτευξη της ποιότητας του προϊόντος. Οι Owens & Khazanchi, (2009) ορίζουν τη διασφάλιση της ποιότητας λογισμικού (SQA), ως μια καθορισμένη και επαναλαμβανόμενη διαδικασία που αποτελεί μέρος της διαχείρισης του έργου και του κύκλου ζωής ανάπτυξης λογισμικού. Ο στόχος της SQA είναι να εξασφαλίσει τη συμμόρφωση προς τις απαιτήσεις, να μειώσει τον κίνδυνο και να βελτιώσει την ποιότητα.

Ο ISO (Διεθνής Οργανισμός Τυποποίησης) και η IEC (Διεθνής Ηλεκτροτεχνική Επιτροπή) αποτελούν οργανισμούς δημιουργίας και έκδοσης παγκόσμιων προτύπων. Ο Διεθνής Οργανισμός Τυποποίησης είναι ο ευρύτερα αναγνωρισμένος οργανισμός τυποποίησης στον κόσμο (International Organization for Standardization, 1999). Τα πρότυπα που εξετάζονται στην παρούσα μεταπτυχιακή διατριβή είναι το ISO 9001:2015, το ISO 27001:2013, το ISO 20000-1:2011, το ISO 22301:2012 και το ISO / IEC 9126 για την αξιολόγηση της ποιότητας του λογισμικού.

2.5.1 ISO 9001:2015 - Quality Management Systems

Το πρότυπο ISO 9001 είναι το πλέον διαδεδομένο πρότυπο διαχείρισης της ποιότητας, που θέτει τις απαιτήσεις με τις οποίες πρέπει να λειτουργεί μια επιχείρηση, ώστε το τελικό προϊόν (ή υπηρεσία) να ικανοποιεί τις προσδοκίες των πελατών της και των ενδιαφερόμενων μερών.

Η εφαρμογή του ISO 9001:2015 βασίζεται στις 8 αρχές της ποιότητας όπως αυτές προσδιορίζονται και στο πρότυπο ISO 9000:2005 και είναι: Εστίαση στο πελάτη, ηγεσία, συμμετοχή του ανθρώπινου δυναμικού, διεργασιοκεντρική προσέγγιση, συστημική

προσέγγιση διαχείρισης, συνεχής βελτίωση, λήψη αποφάσεων βασισμένη σε δεδομένα, αμοιβαία ωφέλιμες σχέσεις με προμηθευτές.

Το ISO 9001 περιέχει απαιτήσεις συστήματος διαχείρισης βασισμένες στην κυκλική δυναμική διεργασία σχεδιάζω, εφαρμόζω, ελέγχω και βελτιώνω. Αναλυτικότερα, στο πρώτο «σταθμό», στον σχεδιασμό, αναλύεται το επιχειρηματικό περιβάλλον και οι ανάγκες των πελατών και αναγνωρίζονται οι επιπτώσεις τους στον οργανισμό. Επίσης, προσδιορίζονται οι στρατηγικοί στόχοι για την βελτίωση της ικανοποίησης του πελάτη. Στη δεύτερη φάση εφαρμογής, πραγματοποιούνται οι διεργασίες από τις επιχειρήσεις. Στη φάση του ελέγχου παρακολουθούνται και μετριούνται οι διεργασίες με βάση τους επιχειρηματικούς στόχους, ενώ αναλύονται τα αποτελέσματα. Στην τελευταία φάση της βελτίωσης λαμβάνονται ενέργειες για την βελτίωση της ικανοποίησης του πελάτη σε διαρκή βάση.

Το ISO 9001 μπορεί να εφαρμοστεί από οποιονδήποτε οργανισμό ενδιαφέρεται να βελτιώσει τον τρόπο λειτουργίας του και τις σχέσεις με τους πελάτες και προμηθευτές ανεξάρτητα από το μέγεθος ή τον τομέα στον οποίο δραστηριοποιείται. Για να πραγματοποιηθεί αυτό θα πρέπει να αναγνωριστούν, διαχειριστούν, ελεγχθούν και βελτιωθούν όλες οι επιχειρησιακές διεργασίες που έχουν επίπτωση στην ικανοποίηση του πελάτη.

2.5.2 ISO 27001:2013 - Information Security Management Systems

Η πληροφορία είναι κρίσιμης σπουδαιότητας για τη λειτουργία και την επιβίωση μίας εταιρείας. Η πιστοποίηση κατά ISO/IEC 27001 ενισχύει μια εταιρεία να βελτιώσει και να προφυλάξει τα ανεκτίμητα δεδομένα της. Το ISO/IEC 27001 είναι το αποκλειστικώς οικουμενικό πρότυπο, που μπορεί να εξετάσει και να προσδιορίζει τις προϋποθέσεις για ένα σύστημα διαχείρισης ασφάλειας πληροφοριών. Το πρότυπο είναι ικανό να κατοχυρώνει την διαλογή ικανών και σταθερών αξιολογήσεων ασφάλειας. Αυτή η διαλογή ενισχύει μία εταιρεία να προφυλάξει τις πληροφορίες που διαθέτει και να ανακτήσει την εμπιστοσύνη των πελατών, των μετόχων και των ενδιαφερομένων μερών της.

Το πρότυπο βασίζεται στη διεργασιακή γεφύρωση για την σταθεροποίηση, εκτέλεση, χειρισμό, έλεγχο, επισκόπηση, διατήρηση και αναβάθμιση ενός συστήματος διαχείρισης ασφάλειας πληροφοριών. Το ISO/IEC 27001 είναι κατάλληλο για όλες τις εταιρείες του κλάδου πληροφορικής, μικρομεσαίες ή πολυεθνικές και σε κάθε εργασιακό περιβάλλον. Είναι, ιδιαιτέρως, κατάλληλο για εταιρείες που η προφύλαξη των δεδομένων είναι αποφασιστικής σημασίας, όπως: τράπεζες, ασφαλιστικές εταιρείες, εταιρείες τηλεπικοινωνιών, νοσοκομεία, δημόσιες υπηρεσίες και εταιρείες πληροφορικής. Το ISO/IEC 27001 είναι επίσης κατάλληλο για εταιρείες που διαθέτουν, διαφυλάσσουν και διαχειρίζονται δεδομένα για λογαριασμό άλλων, όπως εταιρείες παροχής υπηρεσιών πληροφορικής και μπορεί να διασφαλίσει ότι τα προσωπικά δεδομένα των πελατών τους προστατεύονται.

Η εφαρμογή του προτύπου ISO/IEC 27001, σε μία εταιρεία, μπορεί να παρέχει τα παρακάτω πλεονεκτήματα:

- Τεκμηριώνει μέσω ενός ανεξάρτητου πιστοποιητή-φορέα ότι οι εσωτερικοί έλεγχοι της εταιρείας διεξάγονται αποτελεσματικά και διασφαλίζουν τους εταιρικούς σκοπούς και στρατηγικές.
- Επαληθεύει ότι οι προϋποθέσεις για σωστή διακυβέρνηση και επιχειρησιακή συνέχεια ικανοποιούνται.
- Βεβαιώνει ότι η εθνική και ευρωπαϊκή νομοθεσία και οι κανονισμοί τους εφαρμόζονται.
- Προσφέρει ανταγωνιστικό πλεονέκτημα στην διασφάλιση τυποποιημένων υποχρεώσεων και διαβεβαιώνει στους πελάτες της εταιρείας ότι η ασφάλεια των δεδομένων τους είναι στοιχειώδους βαρύτητας για την εταιρεία.
- Τεκμηριώνει μέσω ενός πιστοποιητή-φορέα ότι οι οργανωτικοί κίνδυνοι έχουν αντιμετωπιστεί και αξιολογηθεί θετικά και ορθά.
- Προωθεί την οντότητα ενός σοβαρού και λειτουργικού συστήματος διαχείρισης ασφάλειας πληροφοριών.
- Αναδεικνύει τη δέσμευση της διοίκησης της εταιρείας στην προστασία των δεδομένων της.
- Βεβαιώνει ότι μέσω συνεχών ελέγχων ενισχύει την εταιρεία να εποπτεύει την αποδοτικότητά της και να εξελίσσεται.

- Επιβεβαιώνει ότι όλα τα δεδομένα που διατηρούνται, διαχειρίζονται ή προωθούνται μέσω πληροφοριακών συστημάτων είναι σημαντικά για την εταιρεία.

Το ISO/IEC 27001πιστοποιεί επιχειρήσεις που αξιολογούν το επιχειρηματικό ρίσκο, ώστε να χρησιμοποιούν ένα πρότυπο-σύστημα διαχείρισης που προσδίδει:

- Μέγιστη αποδοτικότητα των συστημάτων.
- Ισχυροποίηση της επεξεργασίας της πληροφορίας και της συντηρισιμότητας των συστημάτων της εταιρείας.
- Σιγουριά ότι η πληροφορία διατηρείται σε μέγιστο βαθμό.

2.5.3 ISO 20000-1:2011 - IT Service Management

Το ISO/IEC 20000 είναι το πρώτο διεθνές πρότυπο για τα Συστήματα Διαχείρισης Υπηρεσιών Πληροφορικής (IT Service Management Systems). Αναπτύχθηκε το 2005 από τον ISO/IEC/TC1/SC7 βασιζόμενο (και με σκοπό να το αντικαταστήσει) στο BS 15000. Το πρότυπο αναθεωρήθηκε το 2011, όπου έλαβε και τη σημερινή του ισχύουσα έκδοση. Ειδικότερα, το ISO/IEC 20000 αναπτύχθηκε λαμβάνοντας υπόψη τις αρχές του ISO 9001, καθώς επίσης και του ITIL. Είναι μια «οικογένεια προτύπων», που σήμερα αποτελείται από 8 πρότυπα. Τα τρία κύρια πρότυπα της οικογένειας έχουν τον γενικό τίτλο: Σύστημα Υπηρεσιών Πληροφορικής (Information technology – Service management).

- *Μέρος 1 – Απαιτήσεις* (ISO/IEC 20000-1:2011, Information technology -- Service management -- Part 1: Service management system requirements): Περιλαμβάνει τις απαιτήσεις για τον πάροχο υπηρεσιών, για το σχεδιασμό, καθορισμό, υλοποίηση, λειτουργία, ανασκόπηση, διατήρηση και βελτίωση ενός Συστήματος Διαχείρισης Υπηρεσιών Πληροφορικής. Οι απαιτήσεις περιλαμβάνουν τον σχεδιασμό, μετάβαση, παράδοση και βελτίωση των υπηρεσιών, ώστε να εκπληρώνονται οι συμφωνημένες απαιτήσεις των πελατών.
- *Μέρος 2ο – Οδηγός Εφαρμογής Απαιτήσεων* (ISO/IEC 20000-2:2012, Information technology -- Service management -- Part 2: Guidance on the application of service management systems): Παρέχει ένα οδηγό εφαρμογής ενός Συστήματος Διαχείρισης Υπηρεσιών Πληροφορικής βασισμένο στις απαιτήσεις του ISO/IEC

20000-1:2011. Επίσης, βοηθάει στη καλύτερη ερμηνεία του ISO/IEC 20000-1:2011 και συνεπώς στην αποτελεσματικότερη εφαρμογή του.

- *Μέρος 3ο – Οδηγός Εφαρμογής Πεδίου* (ISO/IEC 20000-3:2012 Information technology -- Service management -- Part 3: Guidance on scope definition and applicability of ISO/IEC 20000-1): Αποτελεί ένα χρήσιμο βοήθημα σε πάροχους υπηρεσιών πληροφορικής, συμβούλους και επιθεωρητές σε θέματα που αφορούν τον ορισμό του πεδίου, την εφαρμογή και την απόδειξη της συμμόρφωσης με τις απαιτήσεις του ISO 20000-1:2011.

Αναμφίβολα, η πληροφορική καταλαμβάνει όλο και μεγαλύτερο μέρος της επιχειρησιακής και ιδιωτικής ζωής. Συνεπώς, οι υπηρεσίες και οι πάροχοι υπηρεσιών πληροφορικής πληθαίνουν συνεχώς. Μαζί τους μεγαλώνουν η αξία και η σημαντικότητα της ποιότητας των σχετικών παρεχόμενων υπηρεσιών. Το ISO/IEC 20000 παρέχει το πλαίσιο εκείνο που διασφαλίζει τη ποιότητα, συνέχεια και βελτίωση των παρεχόμενων υπηρεσιών πληροφορικής. Γι' αυτό και η εφαρμογή του γίνεται όλο και πιο σημαντική.

Μία επιχείρηση / πάροχος υπηρεσιών πληροφορικής που έχει πιστοποιηθεί από ανεξάρτητο φορέα πιστοποίησης αποδεικνύει ότι αντιμετωπίζει με ευθύνη, καθώς και ότι εφαρμόζει και ελέγχει τη ποιότητα των παρεχόμενων υπηρεσιών πληροφορικής. Τα πλεονεκτήματα όμως δεν σταματούν εκεί. Η πιστοποίηση προσφέρει και τα ακόλουθα πλεονεκτήματα:

- Προσθέτει σημαντική αξία στο Σύστημα Διαχείρισης μέσα από τη διαδικασία ανεξάρτητης αξιολόγησης / επιθεώρησης.
- Ενδυναμώνει την εμπιστοσύνη των πελατών, εργαζομένων, συνεργατών, φορέων και γενικά όλων των ενδιαφερομένων, ως προς τη ποιότητα, συνέχεια και βελτίωση των περιεχόμενων υπηρεσιών πληροφορικής.
- Προβάλει σημαντική αξιοπιστία και εμπιστοσύνη.
- Μπορεί να οδηγήσει σε σημαντική μείωση κόστους.
- Διασφαλίζει ότι υπάρχει δέσμευση ως προς την ποιότητα παρεχόμενων υπηρεσιών πληροφορικής από όλους και σε όλα τα επίπεδα του οργανισμού.

2.5.4 ISO 22301:2012 - Business Continuity Management Systems

Το πρότυπο ISO 22301 εφαρμόζεται, ώστε να μην σταματήσει η λειτουργία των εταιρειών σε απαιτητικές και έκτακτες ανάγκες. Σε πολλές εταιρείες απαιτείται, για ποικίλους λόγους, η διαρκής λειτουργία τους κατόπιν κάποιου γεγονότος το οποίο οδηγεί σε διακοπή, σε συγκεκριμένα χρονικά διαστήματα. Το πρότυπο εφαρμόζεται στις προηγούμενες περιπτώσεις, με σκοπό να συνεχιστεί η λειτουργία της εταιρείας. Το πρότυπο παρέχει στην εταιρεία τις βασικές αξίες και εφαρμογές ενός Συστήματος Διαχείρισης Επιχειρησιακής Συνέχειας, το οποίο προφυλάσσει το προσωπικό, διαφυλάσσει τη φήμη της εταιρείας και προσφέρει την δεξιότητα στην εταιρεία να μπορεί να υπηρετεί τον πελάτη και να εμπορεύεται τα προϊόντα της.

Η αδιάκοπη εργασία μίας εταιρείας, στην περίπτωση κάποιου απρόβλεπτου γεγονότος π.χ. διακοπής ηλεκτρικού ρεύματος, φυσικών καταστροφών, είναι στοιχειώδης αξίωση. Το ISO 22301 αποτελεί την συνέχεια του Βρετανικού Πρότυπου BS 25999, που ήταν το πρώτο στον κόσμο πρότυπο Συστήματος Διαχείρισης Επιχειρησιακής Συνέχειας, που λειτούργησε και υλοποιήθηκε, ώστε να μηδενίζει τα ρίσκα κινδύνων, που ενδεχομένως επηρεάζουν την λειτουργία των εταιρειών.

Παρέχει τη δομή για την υλοποίηση και εκτέλεση της επιχειρηματικής συνέχειας σε μία εταιρεία και βελτιώνει την αξιοπιστία σε δοσοληψίες μεταξύ εταιρειών (B2B) και μεταξύ εταιρειών και πελάτη (B2C). Επίσης, περιλαμβάνει μια ολόκληρη σειρά ελέγχων βασισμένων σε ήδη εφαρμοσμένες πρακτικές άλλων εταιρειών σχετικά με το Συστήμα Διαχείρισης Επιχειρησιακής Συνέχειας και συμπεριλαμβάνει όλο τον κύκλο εργασιών τους.

Το ISO 22301 εφαρμόζεται για το σύνολο των εταιρειών, είτε μικρές, είτε μικρομεσαίες είτε πολυεθνικές και σε κάθε εργασιακό περιβάλλον. Το πρότυπο θεωρείται ιδιαίτερα εφαρμόσιμο για εταιρείες που εργάζονται σε περιβάλλοντα υψηλού κινδύνου, όπως τράπεζες, χρηματιστηριακοί οργανισμοί, εταιρείες τηλεπικοινωνιών, εταιρείες logistics και δημόσιες υπηρεσίες, όπου παρέχονται συνεχείς υπηρεσίες και η εταιρεία είναι απαραίτητο να λειτουργεί 24 ώρες το 24ωρο, ενώ τυχόν απρόβλεπτες συνθήκες θα

επηρεάσουν τόσο τους εργαζομένους, όσο τα ενδιαφερόμενα μέλη, τους μετόχους και φυσικά τους πελάτες.

Τα πλεονεκτήματα του ISO 22301 είναι ποικίλα και αφορούν ποικίλες εταιρικές διεργασίες:

- Διαθέτει συγκεκριμένους κανόνες, που συναρτώνται από διεθνείς εταιρικές πρακτικές, για την δραστική εφαρμογή της επιχειρηματικής συνέχειας.
- Αναβαθμίζει, χρησιμοποιώντας την πρόληψη, τις υπηρεσίες μιας εταιρείας, ώστε να μπορεί να επιτύχει τους πρωταρχικούς σκοπούς της σε περίπτωση παύσεων εργασίας.
- Εφαρμόζει μια εξασφαλισμένη και δραστική μεθοδολογία αντιμετώπισης ακραίων συνθηκών και θεμελιώνει την δεξιότητα μίας εταιρείας, να παρέχει θεμελιώδη προϊόντα ή/και υπηρεσίες, χωρίς να επηρεαστεί η γραμμή παραγωγής της στην περίπτωση που συμβεί μία διακοπή λειτουργίας.
- Μεταχειρίζεται ήδη τσεκαρισμένες πρακτικές για να διευθύνει αποτελεσματικά οποιαδήποτε παύση εργασίας.
- Ενισχύει την προάσπιση και την αναβάθμιση της φήμης της εταιρείας.
- Προσφέρει ανταγωνιστικό πλεονέκτημα με την διείσδυση σε νέες αγορές και ενισχύει την εταιρεία, ώστε να κερδίσει μεγαλύτερο μερίδιο της αγοράς.
- Βοηθά στην κατανόηση από τους πελάτες ή τους εργαζομένους του τρόπου που λειτουργεί μία εταιρεία και βοηθά στην εφαρμογή κανόνων βελτίωσης της αποδοτικότητας και αποτελεσματικότητας της εργασίας.
- Εισάγει την σχετική νομοθεσία και τους τυποποιημένους ελέγχους, που υλοποιούνται.
- Μειώνει το κόστος παραγωγής και προώθησης των προϊόντων από εσωτερικούς ή/ και εξωτερικούς ελέγχους του Συστήματος Διαχείρισης Επιχειρησιακής Συνέχειας και μειώνει το κόστος χρηματικής ασφάλειας στην περίπτωση παύσεων εργασίας.

2.5.5 ISO 9126 - Software Engineering

Το πρότυπο ISO 9126 που αφορά την μέτρηση της ποιότητα λογισμικού (Kitchenham & Pfleeger, 1996), έχει ως βάση το μοντέλο του McCall (McCall, et al., 1977). Σύμφωνα με το

πρότυπο αυτό, η ποιότητα ορίζεται ως «το σύνολο των στοιχείων και των χαρακτηριστικών ενός προϊόντος λογισμικού που επιδρούν στην ικανότητα του να ανταπεξέρχεται και να ικανοποιεί είτε δεδηλωμένες είτε υπονοούμενες ανάγκες» και απαρτίζεται από έξι ποιοτικούς παράγοντες:

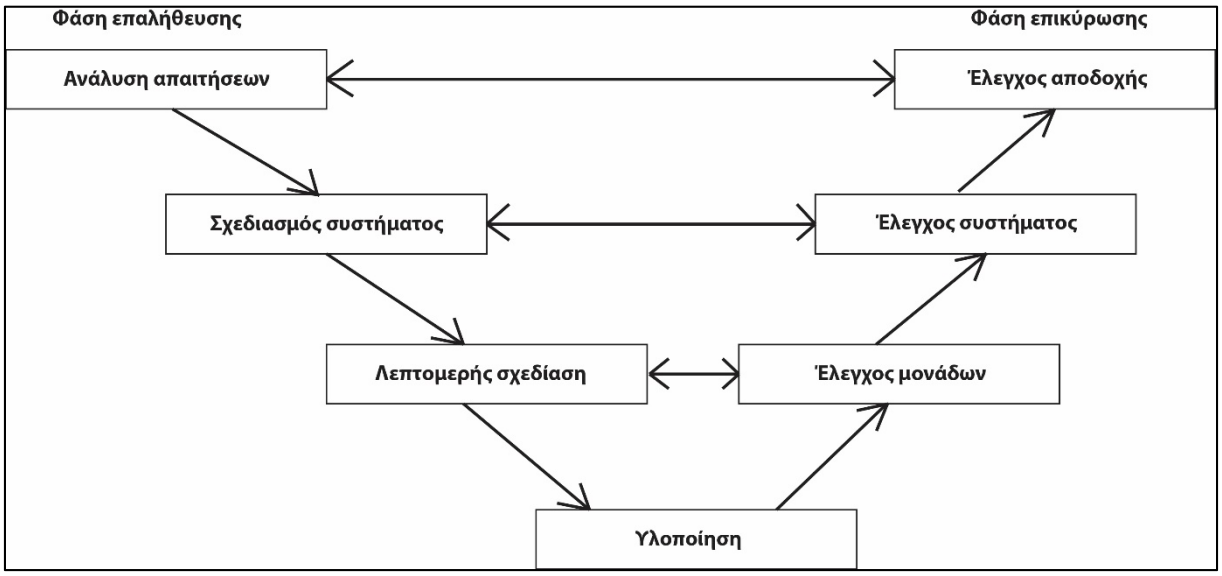
- Functionality (λειτουργικότητα). Το λογισμικό αποτελείται από διεργασίες, που ικανοποιούν τις αξιώσεις του πελάτη.
- Usability (ευχρηστία). Το λογισμικό παρέχει τις λειτουργίες του με τρόπο αποδοτικό και αποτελεσματικό, διευκολύνοντας τους τελικούς χρήστες.
- Reliability (αξιοπιστία). Το λογισμικό παράγει αξιόπιστα αποτελέσματα.
- Efficiency (αποδοτικότητα). Το λογισμικό διαθέτει ανώτερα επίπεδα επεξεργασίας και αποτελεσματικότητας, υπό καθορισμένες συνθήκες.
- Maintainability (συντηρησιμότητα). Το λογισμικό έχει την δυνατότητα να διορθώνει, τροποποιεί ή προσθέτει διεργασίες και χαρακτηριστικά.
- Portability (μεταφερσιμότητα). Το λογισμικό έχει την δυνατότητα να μεταφερθεί από ένα λειτουργικό σύστημα σε άλλο.

Το πρότυπο περιέχει μια πρόταση διαχωρισμού των ποιοτικών παραγόντων σε αναλυτικότερες κατηγορίες, η οποία όμως δεν είναι δεσμευτική. Η διαφορά του προτύπου ISO 9126, από το μοντέλο του McCall, είναι ότι το πρότυπο ISO 9126 αντιμετωπίζει την ποιότητα του λογισμικού από την πλευρά του χρήστη, ενώ το μοντέλο του McCall από την πλευρά του προϊόντος.

2.5.6 Οι δοκιμές του λογισμικού

Η δοκιμή του λογισμικού καταλαμβάνει ένα σημαντικό κομμάτι του κύκλου ζωής του λογισμικού. Οι Shao, et al., (2007) ορίζουν τη δοκιμή λογισμικού ως μια τεχνική που χρησιμοποιείται για την επικύρωση της ποιότητας του λογισμικού. Σύμφωνα με τον Bentley, (2005), η δοκιμή του λογισμικού έχει τρεις κύριους σκοπούς: επαλήθευση, επικύρωση και εύρεση ελαττωμάτων. Η διαδικασία επαλήθευσης επιβεβαιώνει τις τεχνικές προδιαγραφές του λογισμικού. Η διαδικασία επικύρωσης επιβεβαιώνει ότι το λογισμικό πληροί τις επιχειρηματικές απαιτήσεις. Ένα ελάττωμα είναι μια διαφορά μεταξύ του αναμενόμενου και του πραγματικού αποτελέσματος.

Ένα μεγάλο μέρος των δοκιμών λογισμικού βασίζεται στο μοντέλο V, που αναπτύχθηκε αρχικά από το μοντέλο του καταρράκτη. Σύμφωνα με τον Sommerville, (2004), το μοντέλο V έχει τέσσερις κύριες φάσεις: απαιτήσεις, προδιαγραφές, σχεδιασμό και υλοποίηση και κάθε φάση έχει αντίστοιχες φάσεις ελέγχου και επικύρωσης. Ο Bentley, (2005) αναφέρει ότι η χρήση του μοντέλου V προσθέτει δοκιμές σε ολόκληρο τον κύκλο ζωής του λογισμικού. Η διαδικασία του V-μοντέλου περιγράφεται στην Εικόνα 11.



Εικόνα 11. Το μοντέλο V, που χρησιμοποιείται για δοκιμές λογισμικού (Πηγή: softwareengineering.gr).

Σύμφωνα με τον Bentley, (2005), το μοντέλο δοκιμών V προσδιορίζει πέντε φάσεις δοκιμών λογισμικού: δοκιμές μονάδων, δοκιμές συστημάτων, δοκιμές ολοκλήρωσης, δοκιμές αποδοχής από τον χρήστη και δοκιμές επαλήθευσης παραγωγής. Ο έλεγχος μονάδας (ονομάζεται επίσης δοκιμή μονάδας) εκτελείται κατά τη φάση ανάπτυξης του λογισμικού. Η εξέταση μονάδων είναι πολύ απλούστερη διαδικασία από τη δοκιμή ενός προγράμματος / συστήματος στο σύνολό του. Η δοκιμή μονάδας καθιστά ευκολότερες τις εργασίες εντοπισμού σφαλμάτων, διότι σε περίπτωση που εντοπιστεί σφάλμα, βρίσκεται μέσα στη ελεγχόμενη μονάδα αντί για τις άλλες μονάδες. Ο ελεγκτής πρέπει να είναι εξοικειωμένος με τις εσωτερικές λεπτομέρειες αυτής της μονάδας (Eldon, 1990).

Η φάση δοκιμής ολοκλήρωσης γίνεται για να δούμε, αν οι ενότητες λειτουργούν σωστά, χωρίς να αντιβαίνουν στις εσωτερικές και εξωτερικές προδιαγραφές του συστήματος (Eldon, 1990). Σύμφωνα με τον Bentley, (2005) εξετάζονται όλα τα στοιχεία που είναι απαραίτητα για να σχηματιστεί ένα πλήρες σύστημα. Οι δοκιμές ενοποίησης απαιτούν συμμετοχή με άλλα συστήματα, συμπεριλαμβανομένων συστημάτων με εξωτερικούς προμηθευτές. Ο Bentley, (2005) καθορίζει ως σύστημα δοκιμής, τον έλεγχο όλων των συστατικών, που χρειάζονται για να σχηματίσουν την πλήρη εφαρμογή. Η δοκιμή του συστήματος προσπαθεί να ελαχιστοποιήσει την αλληλεπίδραση με άλλα συστήματα. Η Δοκιμή συστήματος απαιτεί πολλούς ελέγχους, καθώς περιλαμβάνει δυνατότητα αναγνώρισης χαρακτηριστικών. Ο Eldon, (1990) προσθέτει ότι οι δοκιμές συστήματος απαιτούν από τον ελεγκτή να γνωρίζει τις λειτουργίες πολύ καλά.

Ο έλεγχος αποδοχής συστήματος από τον χρήστη (UAT) ονομάζεται επίσης δοκιμή beta ή δοκιμή τελικού χρήστη (Bentley, 2005). Σύμφωνα με τους Miller & Collins, (2001), οι δοκιμές UAT αντιπροσωπεύουν τις παραμέτρους του πελάτη, τα ενδιαφέροντα κ.α. Η ιδέα του UAT είναι να πείσει τον πελάτη ότι τα χαρακτηριστικά της εφαρμογής συμπεριφέρονται σωστά. Ο Eldon, (1990) αναφέρει ότι ο UAT πρέπει να εκτελείται στο λογισμικό με την παρουσία τελικών χρηστών και της ομάδας ανάπτυξης.

Η τελευταία φάση της δοκιμής του λογισμικού ονομάζεται δοκιμή επαλήθευσης προϊόντος. Ο έλεγχος επαλήθευσης είναι η τελευταία ευκαιρία να δοκιμαστεί, αν το λογισμικό είναι έτοιμο για είσοδο στην αγορά (Bentley, 2005).

Κεφάλαιο 3
Μεθοδολογία και Έρευνα

3.1 Ορισμός του προβλήματος – Ερευνητικοί Στόχοι

Η μέθοδος που επελέγη στην παρούσα μεταπτυχιακή διατριβή είναι η ποσοτική έρευνα μέσω ερωτηματολογίων. Σύμφωνα με τον Rajasekar, et al., (2006) είναι μια λογική και συστηματική αναζήτηση για νέες και χρήσιμες πληροφορίες σχετικά με ένα συγκεκριμένο θέμα. Πρόκειται για μια έρευνα, για εξεύρεση λύσεων, σε επιστημονικά και κοινωνικά προβλήματα, μέσα από αντικειμενική και συστηματική ανάλυση. Είναι μια αναζήτηση για γνώση, δηλαδή, μια ανακάλυψη κρυμμένων αληθειών. Οι βασικές και εφαρμοσμένες έρευνες μπορεί να είναι ποσοτικές ή ποιοτικές ή ακόμη και συνδυασμός των δύο. Ο Sellers, (1998) περιγράφει την ποιοτική μέθοδο ως μια εις βάθος εξερεύνηση του τι κάνει τους ανθρώπους να προσκολλώνται σε ένα συγκεκριμένο θέμα: τα συναισθήματα, οι αντιλήψεις, οι διαδικασίες λήψης αποφάσεων, κ.ά. και την ποσοτική ως μία μεθοδολογία που ερευνά ένα δείγμα το οποίο είναι αντιπροσωπευτικό ολόκληρου του πληθυσμού που θα ερευνηθεί.

Σύμφωνα με τους Copper & Schindler (2013) μια ερευνητική διαδικασία έχει τις ακόλουθες φάσεις:

- Αναγνώριση του προβλήματος.
- Καθορισμός του ερευνητικού ερωτήματος.
- Διεξαγωγή βιβλιογραφικής μελέτης για αποσαφήνιση του προβλήματος και κατανόηση του ερευνητικού ερωτήματος.
- Ανάπτυξη ενός σχεδίου για διεξαγωγή της έρευνας.
- Ανάπτυξη ενός σχεδίου για τη συλλογή δεδομένων.

- Διεξαγωγή της συλλογής δεδομένων με επιλεγμένη μέθοδο.
- Ανάλυση των δεδομένων που συλλέχθηκαν και παρουσίαση των αποτελεσμάτων.

3.2 Επιλογή του Ερευνητικού Μοντέλου

Η ποσοτική έρευνα είναι εμπειρική διερεύνηση των κοινωνικών φαινομένων μέσω στατιστικής, μαθηματικών ή υπολογιστικών τεχνικών. Ο στόχος της ποσοτικής έρευνας είναι η ανάπτυξη θεωριών βάσει φαινόμενων. Τα ποσοτικά δεδομένα είναι σε αριθμητική μορφή, όπως για παράδειγμα, στατιστικές ή ποσοστά (Given, 2008). Ο Creswell (1994) περιγράφει την ποσοτική έρευνα ως μια διαδικασία επεξήγησης φαινομένων με τη συλλογή στατιστικών, αφού αυτά αναλυθούν με μαθηματικές μεθόδους. Σύμφωνα με τον Sukamolson (2007), υπάρχουν τέσσερις διαφορετικοί τύποι έρευνας: έρευνα, δημοσκόπηση, έρευνα συσχέτισης, πειραματική έρευνα και συγκριτική έρευνα.

3.3 Μέθοδος διεξαγωγής της έρευνας

Η χρήση ερωτηματολογίων είναι, πιθανώς, η πιο συχνά χρησιμοποιούμενη μέθοδος έρευνας παγκοσμίως. Οι Pfleeger & Kitchenham (2002) περιγράφουν την δημοσκόπηση ως ένα ολοκληρωμένο σύστημα για τη συλλογή πληροφοριών που περιγράφει τη στάση και τη συμπεριφορά. Επίσης, διαχωρίζουν την διαδικασία εκπόνησης δημοσκοπήσεων σε 10 διαφορετικές φάσεις:

1. Καταγραφή ειδικών, μετρήσιμων στόχων.
2. Σχεδιασμός και προγραμματισμός της δημοσκόπησης.
3. Διασφάλιση ότι οι κατάλληλοι πόροι είναι διαθέσιμοι.
4. Σχεδιασμός της δημοσκόπησης.
5. Προετοιμασία του ερωτηματολογίου συλλογής δεδομένων.
6. Επικύρωση του ερωτηματολογίου.
7. Επιλογή των συμμετεχόντων.
8. Συμπλήρωση του ερωτηματολογίου.
9. Ανάλυση δεδομένων.
10. Αναφορά αποτελεσμάτων.

Σύμφωνα με τους Taylor-Powell & Hermann (2002), υπάρχουν πέντε κύρια μέσα δημοσκοπήσεων: το ταχυδρομείο, το τηλέφωνο, η επικοινωνία πρόσωπο με πρόσωπο, η χρήση έντυπου ερωτηματολογίου και η ηλεκτρονική χρήση ερωτηματολογίου. Τα πλεονεκτήματα και μειονεκτήματα τους παρουσιάζονται στον Πίνακας 3.

Πίνακας 3. Πλεονεκτήματα και μειονεκτήματα των διαφόρων μεθόδων έρευνας δημοσκοπήσεων.

Μέθοδος δημοσκόπησης	Πλεονεκτήματα	Μειονεκτήματα
Αποστολή ερωτηματολογίου ταχυδρομικά	Αποστολή ερωτηματολογίου σε μεγάλο μέγεθος του πληθυσμού. Η αποστολή είναι φθηνή.	Χρειάζεται ένας κατάλογος από ταχυδρομικές διευθύνσεις των συμμετεχόντων. Κίνδυνος για μικρό ποσοστό απαντήσεων.
Ερωτήσεις μέσω τηλεφώνου	Λαμβάνουμε αποτελέσματα γρήγορα, επειδή η πλειοψηφία από το κοινό-στόχο διαθέτει τηλέφωνα. Οι ερωτήσεις της έρευνας γίνονται επίσης, πιο σαφείς στο κοινό από το τηλέφωνο.	Νοικοκυριά χωρίς τηλεφωνική σύνδεση αποκλείονται από την έρευνα.
Επικοινωνία πρόσωπο με πρόσωπο	Η εγκυρότητα και αξιοπιστία των αποτελεσμάτων είναι καλύτερη.	Χρονοβόρα έρευνα και με μεγάλο κόστος.
Αποστολή έντυπου ερωτηματολογίου	Οι ερωτηθέντες είναι εύκολα διαθέσιμοι και δίνουν άμεσα την ανατροφοδότηση τους στις ερωτήσεις. Τα έντυπα αποτελούν επίσης ένα φθηνό τρόπο διεξαγωγής της έρευνας.	Δεν είναι εύκολο να βρεθούν οι ερωτώμενοι σε περίπτωση διευκρινήσεων.
Αποστολή ερωτηματολογίου μέσω διαδικτύου	Μεγάλο μέγεθος δείγματος που είναι εύκολο να αναζητηθεί.	Χαμηλό ποσοστό απαντήσεων.

3.4 Μέθοδος δειγματοληψίας

Η τυχαία δειγματοληψία εξασφαλίζει ότι κάθε μέλος του πληθυσμού έχει την ίδια πιθανότητα να συμπεριληφθεί στο δείγμα. Οι Pfleeger & Kitchenham (2002) περιγράφουν τέσσερις μεθόδους δειγματοληψίας διαφόρων πιθανοτήτων: απλού τυχαίου δείγματος στρωματοποιημένης τυχαίας δειγματοληψίας, συστηματικής δειγματοληψίας και δειγματοληψίας κλάσεων. Η μέθοδος του απλού τυχαίου δείγματος προϋποθέτει ότι το κάθε μέλος του πληθυσμού-στόχου έχει την ίδια πιθανότητα να συμπεριληφθεί στο δείγμα. Η στρωματοποιημένη τυχαία δειγματοληψία διαιρεί τον

πληθυσμό-στόχο σε υποομάδες και η κάθε υποομάδα αναμένεται να απαντήσει με διαφορετικό τρόπο στα ερωτήματα. Η συστηματική δειγματοληψία επιλέγει κάθε νιοστό μέλος από μία λίστα πληθυσμού. Η δειγματοληψία κλάσεων βασίζεται στην ομαδοποίηση ατόμων π.χ. με γεωγραφικά κριτήρια που ανήκουν σε καθορισμένες κλάσεις.

3.5 Η αξιοπιστία και η εγκυρότητα της έρευνας.

Η αξιοπιστία και η εγκυρότητα των δεδομένων είναι σημαντικοί παράγοντες κατά τη διεξαγωγή μιας έρευνας. Η αξιοπιστία ορίζεται ως ο βαθμός στον οποίο ένα ερωτηματολόγιο, δοκιμή, παρατήρηση ή οποιαδήποτε διαδικασία μέτρησης παράγει τα ίδια αποτελέσματα σε επαναλαμβανόμενες δοκιμές. Η εγκυρότητα ορίζεται ως η έκταση στην οποία το ερωτηματολόγιο μετρά το σκοπό για τον οποίο έχει συνταχθεί (Miller, 2013). Οι Pfleeger & Kitchenham (2002) ορίζουν διάφορες πτυχές που απαιτούνται και πρέπει να ληφθούν υπόψη κατά τη μέτρηση της εγκυρότητας μιας έρευνας. Αυτές οι πτυχές είναι η εγκυρότητα προσώπου, η εγκυρότητα περιεχομένου, τα κριτήρια εγκυρότητας και η εγκυρότητα σχεδιασμού. Εγκυρότητα προσώπου είναι μια βιαστική επισκόπηση των στοιχείων από αμερόληπτους κριτές. Η εγκυρότητα περιεχομένου ελέγχει πόσο κατάλληλο φαίνεται το ερωτηματολόγιο σε μια ομάδα ερωτηθέντων. Οι εγκυρότητα των κριτηρίων δείχνει, πώς το ερωτηματολόγιο συγκρίνεται με άλλα παρόμοια ερωτηματολόγια. Η εγκυρότητα σχεδιασμού εξετάζει πώς επηρεάζει την έρευνα το μέσο που χρησιμοποιείται (Pfleeger & Kitchenham, 2002).

3.6 Η δομή του ερωτηματολογίου

Το ερωτηματολόγιο περιλαμβάνει ερωτήσεις πολλαπλών επιλογών, που πρέπει να απαντηθούν και να αναλυθούν με διαφορετικούς τρόπους.

3.6.1 Η ανάλυση των ερωτήσεων πολλαπλής επιλογής

Τα αποτελέσματα από ερωτήσεις πολλαπλής επιλογής παρουσιάζονται σε γραφήματα πίτας ή ράβδων ανάλογα με το ερώτημα που περιέχουν. Τα γραφήματα ράβδων και πίττας αποτελούν μια γραφική τεχνική ανάλυση των δεδομένων για τη σύνοψη πληροφοριών μιας μεταβλητής (Callaghan, 1996).

Κεφάλαιο 4
Ανάλυση δεδομένων

4.1 Ο κλάδος των Ελληνικών και Κυπριακών εταιρειών ανάπτυξης λογισμικού

Στην ελληνική και κυπριακή αγορά ανάπτυξης λογισμικού και υπηρεσιών πληροφορικής συμπεριλαμβάνεται μεγάλος αριθμός εταιρειών. Ο τομέας της ανάπτυξης λογισμικού είναι από τους πιο προσοδοφόρους και σημαντικούς τομείς της ελληνικής και κυπριακής οικονομίας, λόγω της αυξημένη ζήτησης για αξιοποίηση της τεχνολογίας και για αυτοματοποίηση και ψηφιοποίηση στον ιδιωτικό και δημόσιο τομέα των δύο χωρών. Η κρίση που επήλθε στις δύο χώρες από το 2009 και μετά, σαφώς και επηρέασε και τον κλάδο της ανάπτυξης λογισμικού, όχι όμως σε τόσο μεγάλο βαθμό, όσο άλλους τομείς της οικονομίας.

Οι περισσότερες εταιρείες ασχολούνται με την αντιπροσώπευση μεγάλων εταιρειών ανάπτυξης λογισμικού του εξωτερικού και προωθούν τα προϊόντα τους, δίνοντας περισσότερο φόρτο στην αλλαγή, προγραμματισμό, ολοκλήρωση και παραμετροποίηση των ήδη υπαρχόντων λογισμικών, ώστε να προσαρμοστούν στις ελληνικές συνθήκες και όχι στην ανάπτυξη νέου λογισμικού, από το μηδέν. Υπάρχουν όμως και οι μεγάλες εταιρείες του κλάδου, οι οποίες αναπτύσσουν δικό τους λογισμικό και παρέχουν μεγάλο εύρος υπηρεσιών πληροφορικής. Η ζήτηση των προϊόντων λογισμικού στις μεγάλες εταιρείες του κλάδου εξαρτάται από το κόστος, την φήμη, την ποιότητα και το χρόνο ολοκλήρωσης των παρεχόμενων υπηρεσιών. Επίσης, τυχόν πολιτικές εξελίξεις, αποφάσεις, θεσμικές αλλαγές επηρεάζουν τον κλάδο και επιδρούν στην ζήτηση των προϊόντων ανάπτυξης λογισμικού.

Το λογισμικό εφαρμογών αποτελεί την κυρίαρχη κατηγορία στον κλάδο εταιρειών ανάπτυξης λογισμικού με ποσοστό 64%, το 2015, όπως προκύπτει από τα αποτελέσματα μελέτης που διεξήχθη από την εταιρεία ICAP Group, σε μεγάλο μέρος επιχειρήσεων που δραστηριοποιούνται στο κλάδο ανάπτυξης λογισμικού. Το υπόλοιπο 36% των εταιρειών ασχολείται κυρίως με την κατηγορία του λογισμικού συστημάτων. Η ζήτηση του λογισμικού στην αγορά το 2015 έδειξε μικρή αύξηση, της τάξης των 0,7% σε σύγκριση με το 2014. Στο διάστημα 2010-2013, ο κλάδος εταιρειών του λογισμικού εμφάνισε μέσο ετήσιο ρυθμό μείωσης της τάξης του -3,2%. Για το 2016, ο κλάδος παρουσίασε αύξηση κατά 1,4% σε σύγκριση με το 2015. Από την έρευνα προέκυψε επίσης ότι η ύφεση στο κλάδο εταιρειών ανάπτυξης λογισμικού ήταν σαφώς μικρότερη σε σχέση με άλλους κλάδους της ελληνικής οικονομίας. Η έλλειψη ρευστότητας για επενδύσεις στον κλάδο εταιρειών πληροφορικής και η στασιμότητα που παρατηρείται από πλευράς εταιρειών του κλάδου να διεξάγουν επενδύσεις στην εγχώρια οικονομία αποτελεί το πιο βασικό αίτιο ύφεσης που επιδεικνύει ο κλάδος τα τελευταία χρόνια.

4.2 Η διεξαγωγή της έρευνας

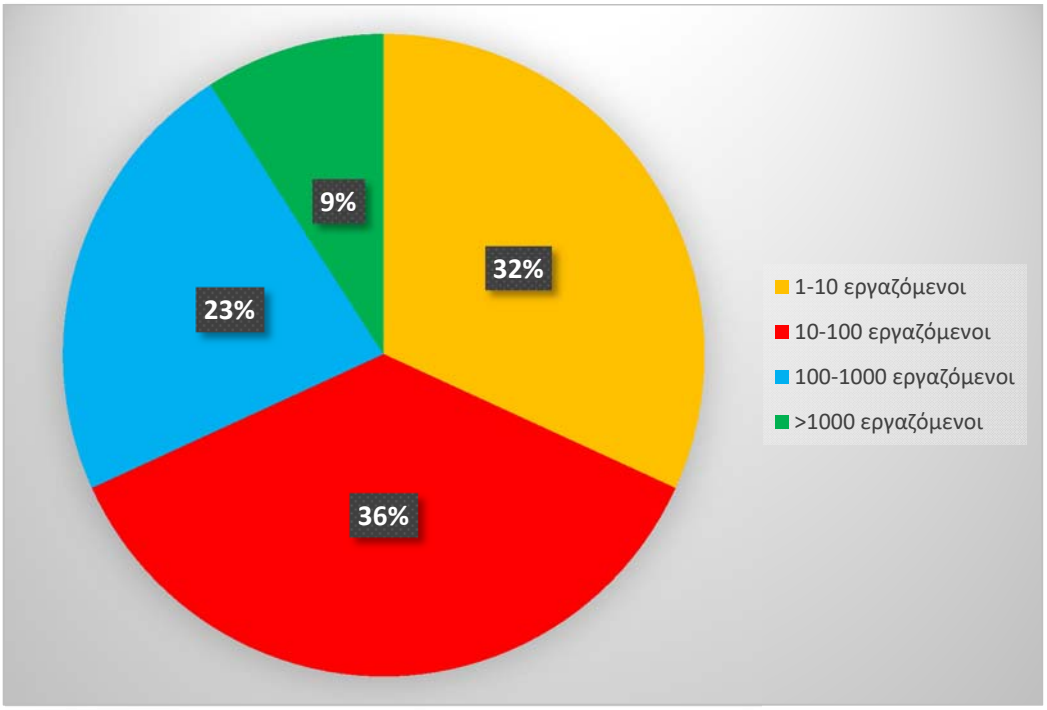
Η μελέτη ξεκίνησε τον Ιούνιο του 2017 με ανασκόπηση του ερευνητικού θέματος. Η ανασκόπηση της βιβλιογραφίας περιελάβανε την μελέτη των μεθόδων κύκλου ζωής λογισμικού, την διασφάλιση ποιότητας του λογισμικού και υλικό σχετικά με το κλάδο εταιρειών ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου. Έρευνες συγκεντρώθηκαν κυρίως μέσω διαδικτύου, από το scholar.google.com, από επιστημονικά περιοδικά, σχετικές ιστοσελίδες και ακαδημαϊκά βιβλία. Η ανασκόπηση της βιβλιογραφίας ολοκληρώθηκε τον Σεπτέμβριο του 2017 και με βάση τις πληροφορίες που συλλέχθηκαν προσδιορίστηκαν τα ερευνητικά ερωτήματα. Ο Οκτώβριος χρησιμοποιήθηκε για την προετοιμασία του ερωτηματολογίου και από τον Νοέμβριο έως τον Δεκέμβριο του 2017 πραγματοποιήθηκε η συλλογή των δεδομένων. Η ανάλυση των δεδομένων έγινε από τον Δεκέμβριο του 2017 έως τον Ιανουάριο του 2018, παράλληλα με την σύνταξη των αποτελεσμάτων και την συγγραφή της μεταπτυχιακής διατριβής.

Ένας από τους σκοπούς της παρούσας μεταπτυχιακής διατριβής ήταν η διερεύνηση του κλάδου εταιρειών ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου. Ο στόχος της

ανασκόπησης της βιβλιογραφίας ήταν να αυξηθεί η γνώση στον τομέα της έρευνας και να δημιουργηθούν κατάλληλα ερευνητικά ερωτήματα.

4.2.1 Το δείγμα της έρευνας

Το δείγμα της μελέτης αποτελείται από 22 εταιρείες της Ελλάδας και της Κύπρου, που αναπτύσσουν λογισμικό. Οι ερωτηθέντες ζητήθηκε να συμπληρώσουν τον αριθμό των εργαζομένων στην εταιρεία τους και τα αποτελέσματα χωρίστηκαν σε τέσσερις διαφορετικές κλάσεις. Τα αποτελέσματα δείχνουν τα μεγέθη των εταιρειών στη μελέτη (Εικόνα 12). Το μεγαλύτερο δείγμα ερωτηθέντων μοιράστηκε μεταξύ εταιρειών που αποτελούνται από 10-100 άτομα με ποσοστό 36% και από εταιρείες με 1-10 άτομα με ποσοστό 32%. Η τρίτη μεγαλύτερη κλάση του δείγματος αντιπροσωπεύει επιχειρήσεις μεταξύ 100 έως 1000 ατόμων (23%). Η τελευταία κατηγορία αποτελείται από εταιρείες άνω των 1000 εργαζομένων με ποσοστό 9%.



Εικόνα 12. Ποσοστά των εταιρειών του δείγματος όσον αφορά τον αριθμό των εργαζομένων σε αυτές.

Από τις εταιρείες του δείγματος, οι δεκαέξι (16) είχαν την έδρα της εταιρείας τους στην Ελλάδα (73%) και έξι (6) είχαν την έδρα της εταιρείας τους στην Κύπρο (27%).

4.2.2 Η μέθοδος συλλογής των δεδομένων

Η χρησιμοποιούμενη μέθοδος συλλογής δεδομένων στην παρούσα μελέτη ήταν η έρευνα μέσω διαδικτύου. Άλλες δυνατότητες περιλάμβαναν τηλεφωνικές συνεντεύξεις. Συγκεκριμένα πραγματοποιήθηκαν δύο τηλεφωνικές συνεντεύξεις. Η πρώτη με την εταιρεία TÜV Hellas (TÜV Nord Group), ώστε να συμπληρωθεί ορθά η ερώτηση του ερωτηματολογίου που αφορούσε την χρήση προτύπων από τις ελληνικές και κυπριακές εταιρείες ανάπτυξης λογισμικού και από την κ. Στέλλα Βαρέλη, εκπρόσωπο της εταιρείας Project Beagle, ώστε να διαπιστωθεί η απήχηση του ερωτηματολογίου και να διασαφηνιστούν περαιτέρω οι απαντήσεις που δόθηκαν σε αυτό.

Το ερωτηματολόγιο (βλ. Παράρτημα Α) δημιουργήθηκε με τη βοήθεια ηλεκτρονικού λογισμικού που ονομάζεται Google Forms, που είναι μια εφαρμογή της Google που βασίζεται στην χρήση του διαδικτύου για την συλλογή απαντήσεων. Η εφαρμογή δημιουργεί μεταξύ άλλων κυρίως ερωτηματολόγια, συλλέγει τις απαντήσεις και δημιουργεί αναφορές αποτελεσμάτων. Το ερωτηματολόγιο της παρούσας μεταπτυχιακής διατριβής περιελάβανε συνολικά 26 ερωτήσεις (Βλ. Παράρτημα Α). Οι ερωτήσεις χωρίστηκαν σε: γενικές (5), ερωτήσεις που αφορούσαν τις μεθόδους ανάπτυξης λογισμικού (12), ερωτήσεις που αφορούσαν την διασφάλιση της ποιότητας του λογισμικού (7) και ερωτήσεις διασαφήνισης της μελλοντικής κατάστασης του κλάδου εταιρειών ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου (2).

Η συλλογή πιθανών και κατάλληλων ερωτηθέντων ξεκίνησε με τη χρήση του Google, ως μηχανή αναζήτησης, σε ιστότοπους εταιρειών της Ελλάδας και της Κύπρου. Οι υποψήφιες εταιρείες επιλέχθηκαν με βάση το εύρος δραστηριοτήτων τους που αποκτήθηκε από τις ιστοσελίδες τους. Οι ιστότοποι των εταιρειών ήταν κυρίως στην ελληνική γλώσσα, ενώ σημαντικός αριθμός κυπριακών εταιρειών διέθετε κυρίως αγγλική ιστοσελίδα. Μετά την απόκτηση διευθύνσεων ηλεκτρονικού ταχυδρομείου από ιστοσελίδες εταιρειών, το Google Forms χρησιμοποιήθηκε για να αποσταλεί η σύνδεση (hyperlink) της έρευνας στα ηλεκτρονικά ταχυδρομεία των εταιρειών. Περίπου 300 τυχαία επιλεγμένα μηνύματα ηλεκτρονικού ταχυδρομείου στάλθηκαν σε εταιρείες της Ελλάδας και της Κύπρου, με αποτέλεσμα να παρθούν 22 απαντήσεις.

4.2.3 Ο χρόνος συλλογής των δεδομένων

Η συλλογή απαντήσεων από τα ερωτηματολόγια πραγματοποιήθηκε από τις 15 Νοεμβρίου 2017 έως τις 10 Ιανουαρίου 2018.

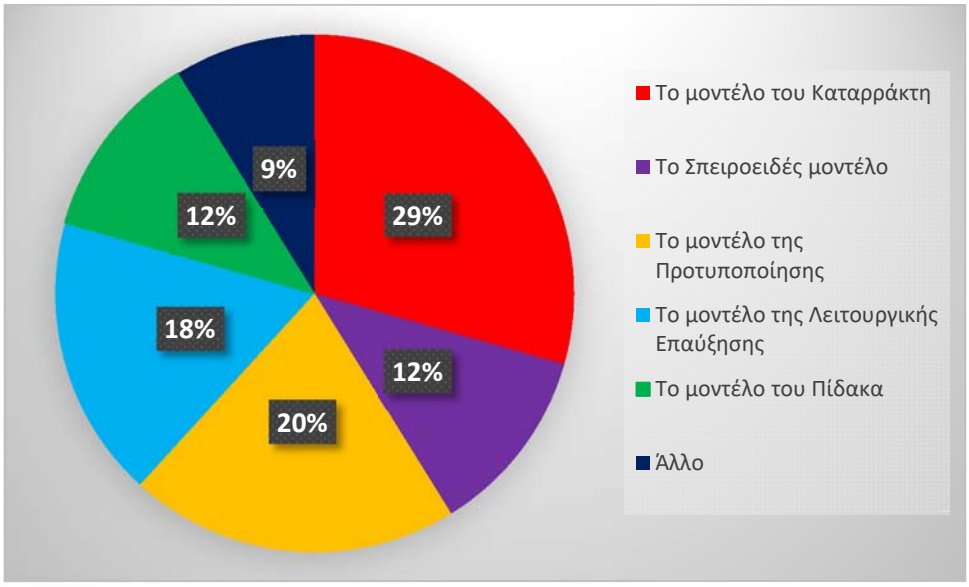
4.3 Το μοντέλο του κύκλου ζωής των ελληνικών και κυπριακών εταιρειών ανάπτυξης λογισμικού

Το πρώτο μέρος της έρευνας περιλαμβάνει πληροφορίες σχετικά με τη χρήση μοντέλων κύκλου ζωής λογισμικού που χρησιμοποιούνται από εταιρείες ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου. Οι συμμετέχοντες στην έρευνα απαντησαν σε ερωτήσεις σχετικά με τις μεθόδους κύκλου ζωής λογισμικού που χρησιμοποιούν σε έργα ανάπτυξης λογισμικού, για την συμμετοχή του πελάτη στον κύκλο ζωής του λογισμικού και για το αν διαθέτει η εταιρείας τους ξεχωριστό τμήμα διασφάλισης ποιότητας.

4.3.1 Η χρήση των παραδοσιακών μεθόδων ανάπτυξης λογισμικού

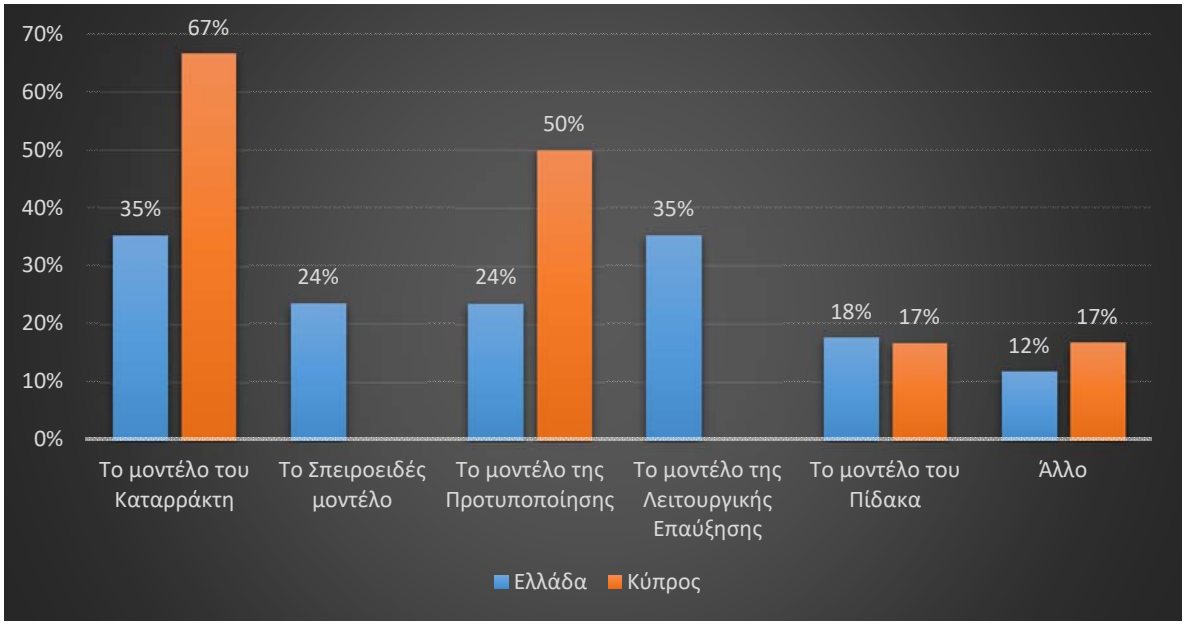
Κοιτώντας βαθύτερα τον κύκλο ζωής του λογισμικού, η έρευνα αποκαλύπτει ποια συγκεκριμένα μοντέλα ή μέθοδοι είναι πιο δημοφιλή μεταξύ των εταιρειών λογισμικού της Ελλάδας και της Κύπρου. Στην περίπτωση όπου η εταιρεία του ερωτηθέντος χρησιμοποιούσε δύο ή περισσότερες διαφορετικές μεθόδους ταυτόχρονα, ήταν δυνατό να δοθούν πολλαπλές απαντήσεις.

Οι παραδοσιακές μέθοδοι που χρησιμοποιούνται από τις εταιρείες του δείγματος αποτελούνται από διαφορετικές γνωστές μεθόδους. Η Εικόνα 13 δείχνει τα πιο δημοφιλή παραδοσιακά μοντέλα ανάπτυξης λογισμικού που χρησιμοποιούνται από τις εταιρείες του δείγματος. Το πιο δημοφιλές παραδοσιακό μοντέλο κύκλου ζωής λογισμικού είναι ο καταρράκτης με 29% ακολουθεί το μοντέλο της προτυποποίησης με 20%, το μοντέλο της λειτουργικής επαύξησης με 18% και ακολούθως το μοντέλο του πίδακα και το σπειροειδές μοντέλο από 12% το καθένα. Άλλες χρησιμοποιούμενες παραδοσιακές μέθοδοι είναι το V-model με 9%.



Εικόνα 13. Η χρήση των παραδοσιακών μεθόδων ανάπτυξη λογισμικού στις εταιρείες του δείγματος.

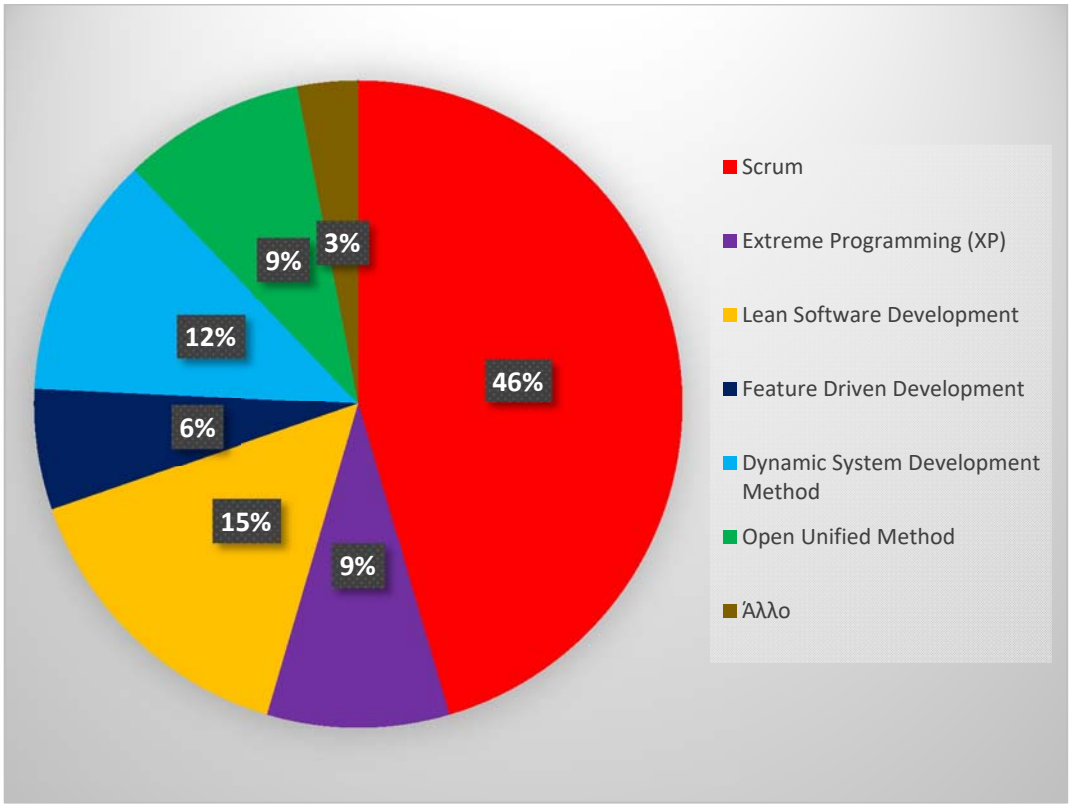
Όσον αφορά, σε επίπεδο χώρας, δίνεται η παρακάτω Εικόνα 14. Από το γράφημα παρατηρούμε ότι οι εταιρείες της Ελλάδας προτιμούν το μοντέλο του καταρράκτη και το μοντέλο της λειτουργικής επαύξησης (35%). Ενώ οι εταιρείες της Κύπρου προτιμούν το μοντέλο του καταρράκτη (67%) και το μοντέλο της προτυποποίησης (50%) και ακολουθεί το μοντέλο του Πίδακα (17%).



Εικόνα 14. Η χρήση των παραδοσιακών μεθόδων ανάπτυξης λογισμικού σε Ελλάδα και Κύπρο.

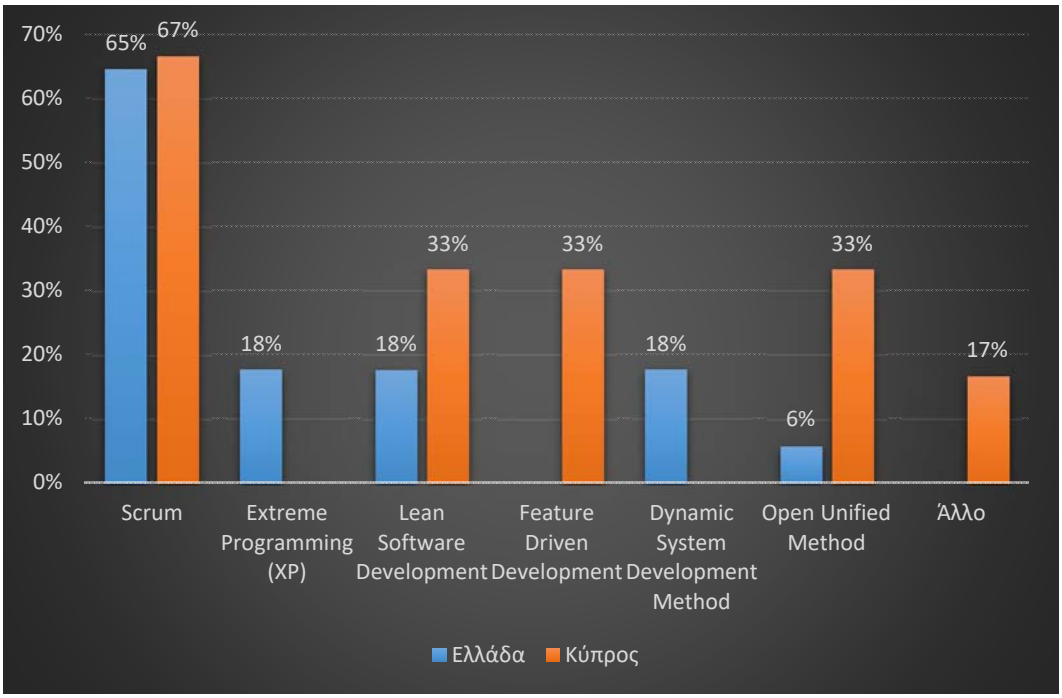
4.3.2 Η χρήση των ευέλικτων μεθόδων ανάπτυξης λογισμικού

Οι ευέλικτες μέθοδοι που χρησιμοποιούνται από τις εταιρείες αποτελούνται από διαφορετικές γνωστές μεθόδους. Η Εικόνα 15, δείχνει τα πιο δημοφιλή μοντέλα που χρησιμοποιούνται από τις εταιρείες του δείγματος. Το πιο δημοφιλές ευέλικτο μοντέλο κύκλου ζωής λογισμικού είναι το Scrum (46%), ακολουθεί το μοντέλο της λιτής ανάπτυξης (15%) και στην συνέχεια το μοντέλο DSDM με 12%. Μικρότερο ποσοστό καταλαμβάνουν τα μοντέλα της ενοποιημένης μεθόδου και του ακραίου προγραμματισμού (9% το καθένα). Ακολουθεί το μοντέλο FDD με 6%, ενώ άλλες απαντήσεις έδωσε το 3%.



Εικόνα 15. Η χρήση των ευέλικτων μεθόδων ανάπτυξης λογισμικού στις εταιρείες της Ελλάδας και της Κύπρου.

Από την Εικόνα 16 παρατηρούμε ότι οι εταιρείες της Ελλάδας και της Κύπρου προτιμούν το μοντέλο του Scrum το οποίο δείχνει ιδιαίτερα δημοφιλές και στις δύο χώρες με ελαφρύ προβάδισμα στην Κύπρο (67%). Έπειτα, δημοφιλές είναι και το μοντέλο της λιτής παραγωγής και στις δύο χώρες με προβάδισμα στην Κύπρο (33%). Η μέθοδος Crystal clear δεν χρησιμοποιείται σε καμία από την δύο χώρες.



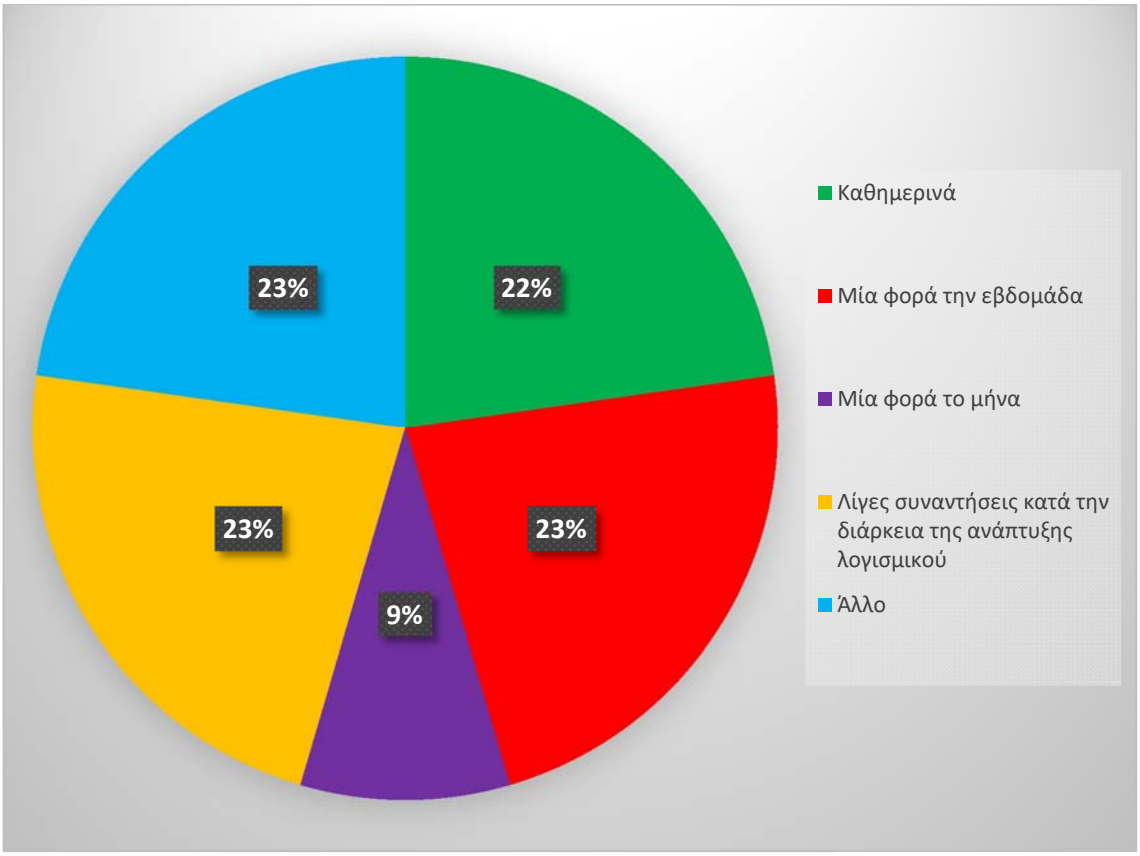
Εικόνα 16. Η χρήση των ευέλικτων μεθόδων ανάπτυξη λογισμικού σε Ελλάδα και Κύπρο.

Τα αποτελέσματα των μεθόδων ανάπτυξης λογισμικού δείχνουν ότι ο κλάδος εταιρειών ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου αρχίζει να προσαρμόζεται στις ευέλικτες μεθόδους. Σχεδόν όλες οι εταιρείες που ερωτήθηκαν χρησιμοποιούν ευέλικτες μεθόδους και αυτό δείχνει το μεγάλο ενδιαφέρον που δείχνουν στις νέες μεθοδολογίες. Ένα άλλο ενδιαφέρον αποτέλεσμα είναι ότι η χρήση των παραδοσιακών μεθόδων εξακολουθεί να είναι αξιοσημείωτη. Αυτό δείχνει ο κλάδος εξακολουθεί να εφαρμόζει και τις παραδοσιακές μεθόδους του κύκλου ζωής λογισμικού και επεκτείνεται και στις νέες.

4.3.3 Η συμμετοχή των πελατών στην διαμόρφωση του λογισμικού

Η συμμετοχή του πελάτη αποτελεί σημαντικό μέρος της ανάπτυξης λογισμικού. Η ερώτηση είχε σκοπό να διερευνήσει την συμμετοχή του πελάτη στον κύκλο ζωής του λογισμικού. Τα αποτελέσματα (Εικόνα 17) δείχνουν ότι οι ερωτηθέντες έδωσαν ποικίλες απαντήσεις. Ποσοστό 23% ερωτηθέντων δήλωσαν ότι πραγματοποιούνται λίγες συναντήσεις κατά την διάρκεια ανάπτυξης του προϊόντος. Ίδιο ποσοστό (23%) δήλωσε ότι πραγματοποιούνται συναντήσεις μία φορά την εβδομάδα. Επίσης 23% δήλωσε ότι πραγματοποιεί συναντήσεις ανάλογα με το project (Άλλο), ενώ 22% πραγματοποιεί

συναντήσεις καθημερινά. Ποσοστό 9% των εταιρειών διεξάγουν συναντήσεις μία φορά το μήνα με τον πελάτη κατά την διάρκεια της ανάπτυξης του λογισμικού.



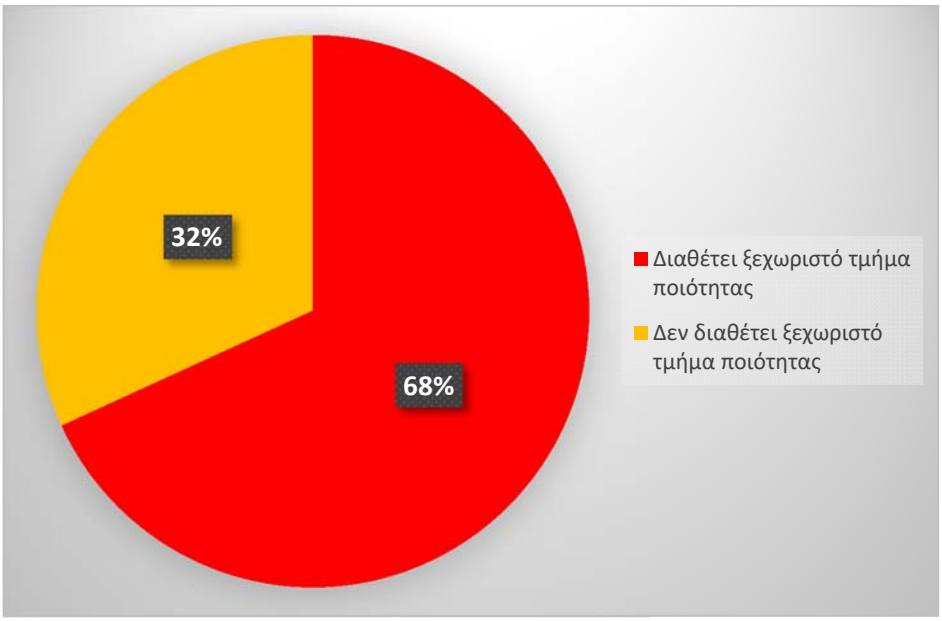
Εικόνα 17. Συμμετοχή των πελατών στην διαδικασία ανάπτυξης λογισμικού.

Οι παραδοσιακές μέθοδοι, συνήθως βασίζονται περισσότερο στην τεκμηρίωση και δεν χρειάζονται πολλές συναντήσεις με τους πελάτες κατά τη διάρκεια του κύκλου ζωής, ενώ οι ευέλικτες μέθοδοι ενθαρρύνουν τον χρήστη να έχει συναντήσεις σχεδόν σε εβδομαδιαία βάση. Φαίνεται ότι οι εταιρείες του δείγματος προσαρμόζουν τις συναντήσεις τους με το πελάτη ανεξάρτητα με την μέθοδο ανάπτυξης λογισμικού που χρησιμοποιούν. Ορισμένες εταιρείες του δείγματος συναντούν πολύ συχνά τον πελάτη, ενώ άλλες εξακολουθούν να ακολουθούν τον παραδοσιακό τρόπο να έχουν λίγες μόνο συναντήσεις με τον πελάτη κατά τη διάρκεια του κύκλου ζωής του λογισμικού.

4.3.4 Το τμήμα διασφάλισης ποιότητας λογισμικού των εταιρειών

Όσον αφορά τη διασφάλιση της ποιότητας του λογισμικού, οι συμμετέχοντες ρωτήθηκαν εάν η εταιρεία τους διαθέτει ξεχωριστό τμήμα διασφάλισης ποιότητας. Τα αποτελέσματα

δείχνουν (Εικόνα 18) ότι το 68% των εταιρειών έχουν ξεχωριστό τμήμα ποιότητας, ενώ το 32% των εταιρειών δεν διαθέτουν το δικό τους τμήμα διασφάλισης ποιότητας.



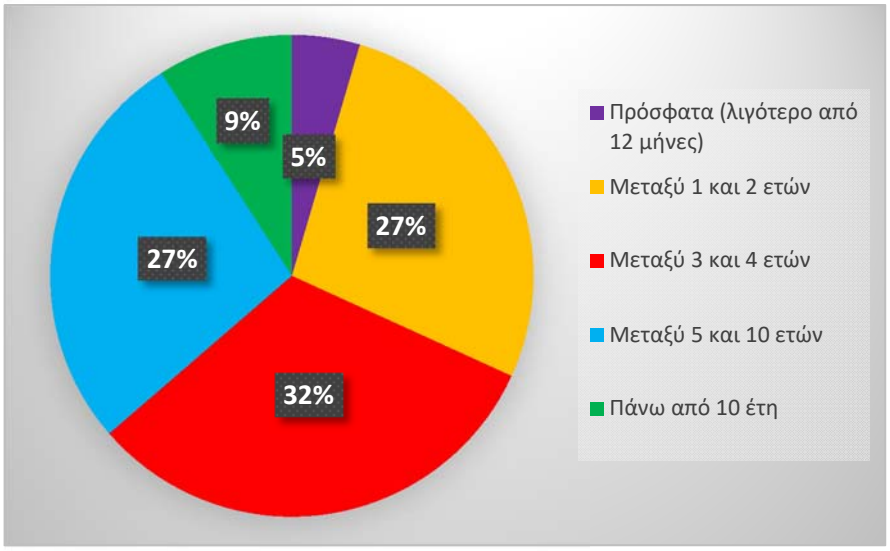
Εικόνα 18. Ποσοστό απαντήσεων στο αν υπάρχει ξεχωριστό τμήμα ποιότητας στην εταιρεία.

4.4 Οι ευέλικτες μέθοδοι ανάπτυξης λογισμικού στις ελληνικές και κυπριακές επιχειρήσεις

Οι ερωτήσεις επικεντρώθηκαν επίσης στο πώς η χρήση ευέλικτων μεθόδων επηρεάζει τον κύκλο ζωής του λογισμικού. Οι ερωτήσεις αφορούσαν τέσσερις πτυχές του κύκλου ζωής του λογισμικού: παραγωγικότητα, ποιότητα, κόστος και ικανοποίηση του πελάτη.

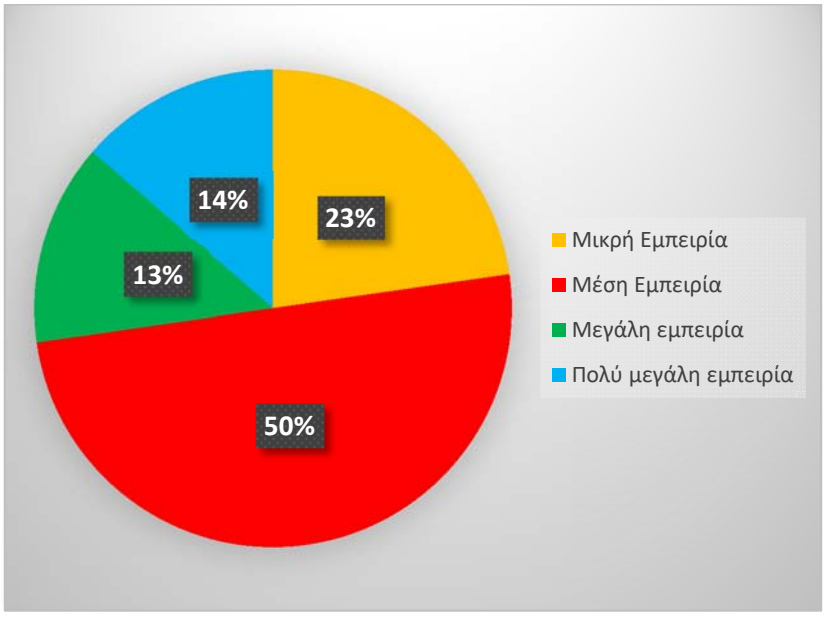
4.4.1 Εμπειρία χρήσης των ευέλικτων μεθόδων ανάπτυξης λογισμικού στην Ελλάδα και την Κύπρου

Οι συμμετέχοντες στην έρευνα ρωτήθηκαν σχετικά με τον χρόνο υιοθέτησης των ευέλικτων μεθόδων ανάπτυξης λογισμικού (Εικόνα 19). Το 32% του δείγματος υιοθέτησε τις ευέλικτες μεθόδους πριν από 3 με 4 έτη. Ποσοστό 27% υιοθέτησε τις ευέλικτες μεθόδους πριν από 1 με 2 έτη. Ίδιο ποσοστό τις υιοθέτησε πριν από 5 με 10 έτη. Ποσοστό 9% έχει υιοθετήσει τις μεθόδους πάνω από 10 έτη, ενώ ένα ποσοστό 5% τις έχει υιοθετήσει πρόσφατα (λιγότερο από 1 έτος).



Εικόνα 19. Χρόνος υιοθέτησης ευέλικτων μεθόδων ανάπτυξης λογισμικού από τις εταιρείες.

Οι συμμετέχοντες ρωτήθηκαν, επίσης, σχετικά με την εμπειρία εφαρμογής των ευέλικτων μεθόδων ανάπτυξης λογισμικού. Οι ερωτηθέντες απάντησαν (Εικόνα 20) ότι διαθέτουν μέση εμπειρία σε ποσοστό 50%, ενώ το 23% την χαρακτήρισαν ως μικρή. Οι υπόλοιπες απαντήσεις που δόθηκαν κυμάνθηκαν σε ποσοστό 14% με πολύ μεγάλη εμπειρία και 13% με μεγάλη. Κανείς δεν απάντησε ότι διαθέτει πολύ μικρή εμπειρία.

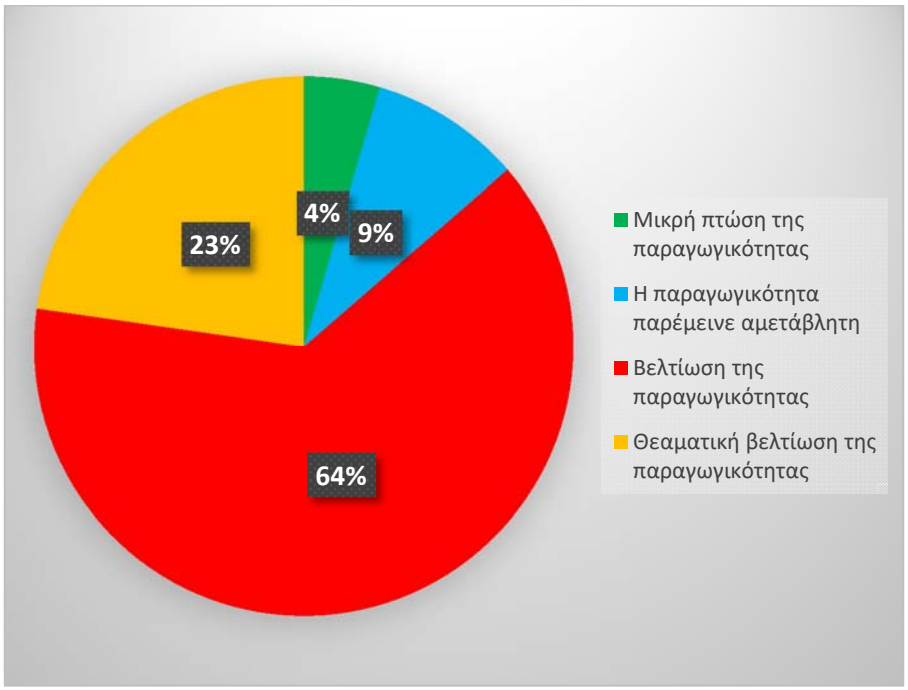


Εικόνα 20. Εμπειρία στην εφαρμογή ευέλικτων μεθόδων ανάπτυξης λογισμικού των ερωτηθέντων.

Τα αποτελέσματα δείχνουν ότι ο κλάδος εταιρειών ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου διαθέτει αρκετή εμπειρία στην χρήση ευέλικτων μεθόδων ανάπτυξης λογισμικού. Πάνω από το 60% των ερωτηθέντων χρησιμοποιούν ευέλικτες μεθόδους για περισσότερο από 3 χρόνια και αυτό δείχνει πολύ ενδιαφέρον για την εφαρμογή τους. Από την άλλη πλευρά, το ήμισυ (50%) των ερωτηθέντων περιέγραψε την εμπειρία σχετικά με τις ευέλικτες μεθόδους ως μέση και το 23% την χαρακτήρισε ως μικρή. Αυτό δείχνει ότι ακόμα και αν οι ερωτηθέντες έχουν χρησιμοποιήσει ευέλικτες μεθόδους για αρκετά μεγάλο χρονικό διάστημα, δεν αισθάνονται ότι γνωρίζουν πολλά γι' αυτές.

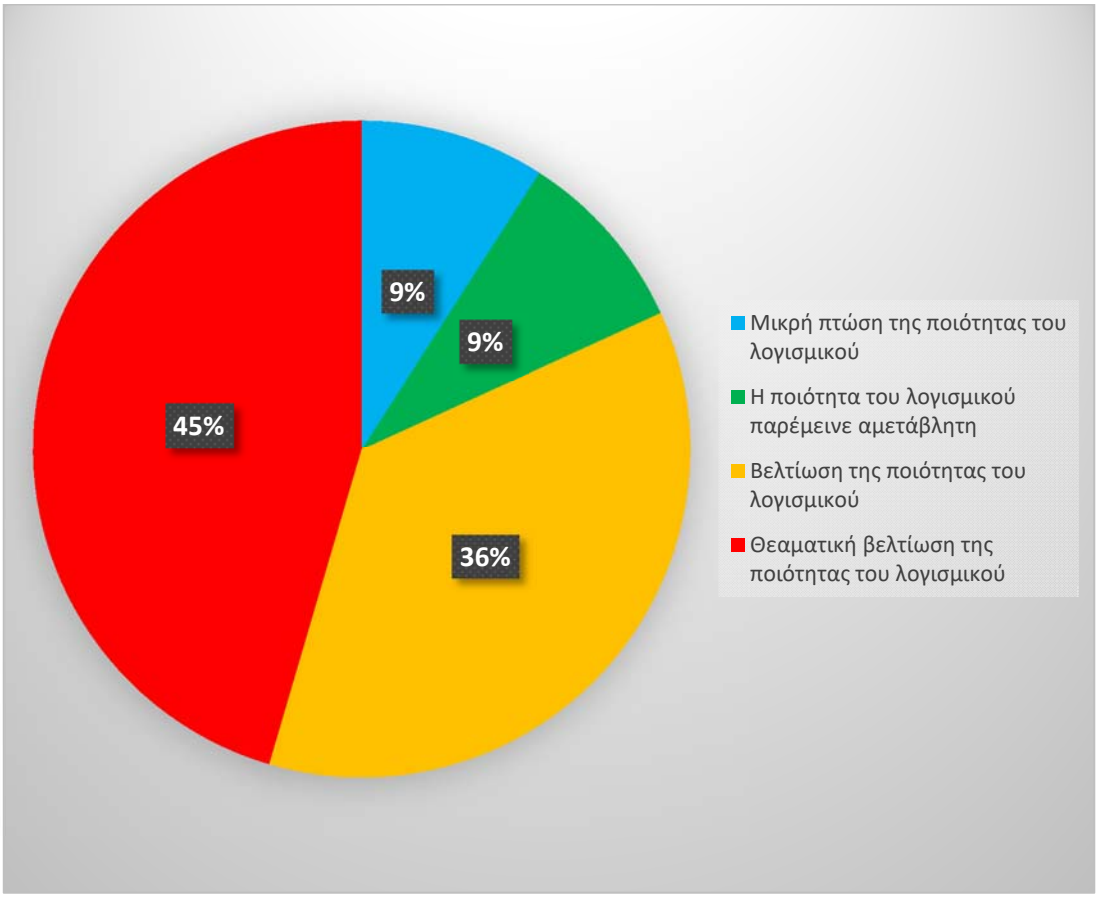
4.4.2 Επιπτώσεις από την χρήση ευέλικτων μεθόδων ανάπτυξης λογισμικού

Τα αποτελέσματα των απαντήσεων σχετικά με την αύξηση της παραγωγικότητας της εταιρείας δείχνουν (Εικόνα 21) ότι το 64% των ερωτηθέντων απάντησε ότι η παραγωγικότητα βελτιώθηκε από την στιγμή που υιοθετήθηκαν οι ευέλικτες μέθοδοι ανάπτυξης λογισμικού, το 23% απάντησε ότι η παραγωγικότητα είχε θεαματική βελτίωση, ενώ μόνο το 9% εκτιμά ότι η παραγωγικότητα παρέμεινε αμετάβλητη και ποσοστό 4% ότι εμφάνισε μικρή πτώση.



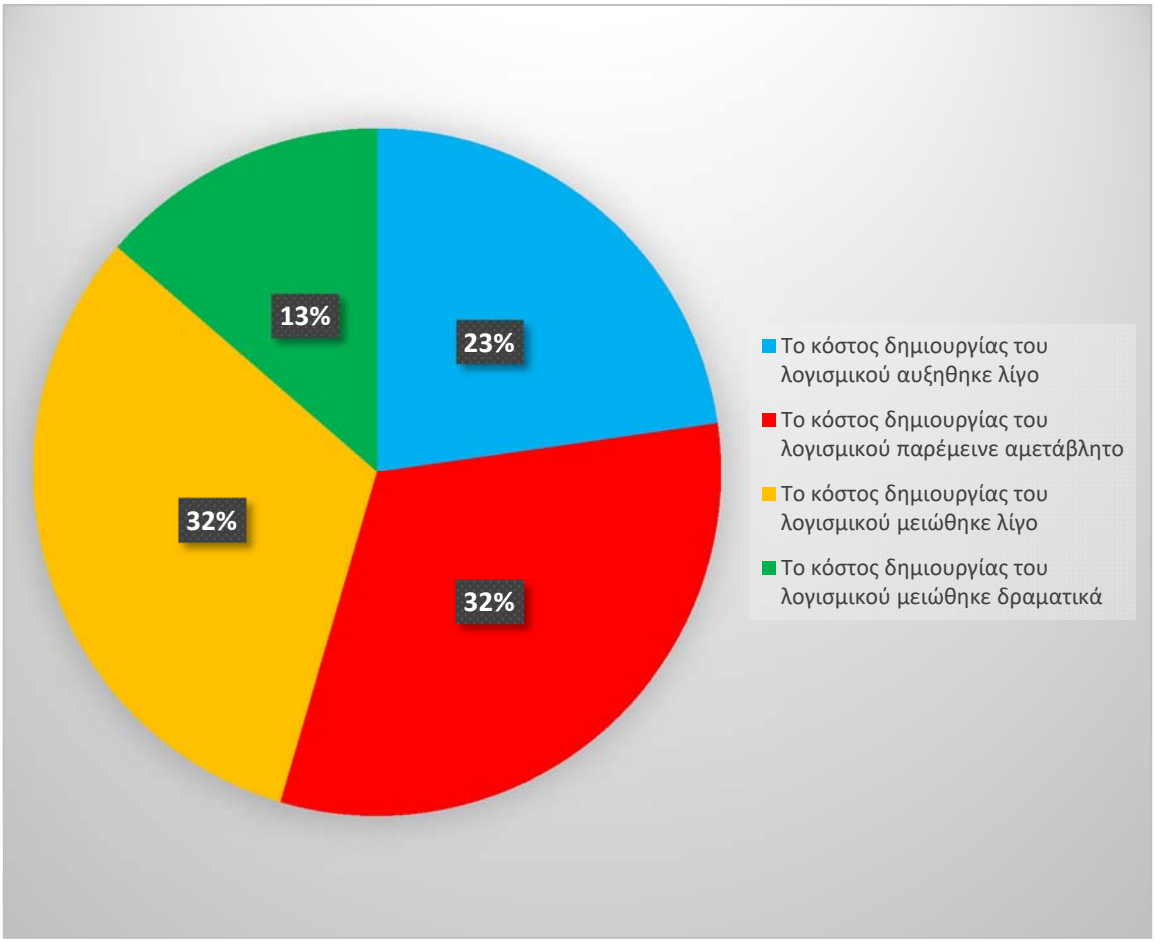
Εικόνα 21. Επιπτώσεις της υιοθέτησης ευέλικτων μεθόδων ανάπτυξης λογισμικού στην παραγωγικότητα της εταιρείας.

Οι επιπτώσεις της υιοθέτησης ευέλικτων μεθόδων στην ποιότητα λογισμικού ακολούθησε παρόμοια τάση με αυτήν της παραγωγικότητας (Εικόνα 22). Η πλειοψηφία (45%) των ερωτηθέντων απάντησε ότι η ποιότητα βελτιώθηκε θεαματικά και το 36% ότι η ποιότητα παρουσίασε μικρή βελτίωση, ενώ ποσοστό 9% των ερωτηθέντων απάντησε ότι η ποιότητα παρέμεινε αμετάβλητη και ίδιο ποσοστό απάντησε ότι παρουσίασε μικρή πτώση. Αξιοσημείωτο είναι ότι κανένας από τους ερωτηθέντες δεν απάντησε ότι η ποιότητα μειώθηκε δραματικά μετά τη υιοθέτηση ευέλικτων μεθόδων.



Εικόνα 22. Επιπτώσεις της υιοθέτησης ευέλικτων μεθόδων ανάπτυξης λογισμικού στην ποιότητα λογισμικού.

Σχετικά με το ερώτημα (Εικόνα 23) μεταβολής του κόστους ανάπτυξης λογισμικού με τη χρήση ευέλικτων μεθόδων, το 32% των ερωτηθέντων απάντησε ότι το κόστος μειώθηκε λίγο, ενώ ίδιο ποσοστό 32% ότι το κόστος παρέμεινε αμετάβλητο, το 23% ότι το κόστος αυξήθηκε λίγο και το 13% ότι μειώθηκε δραματικά.

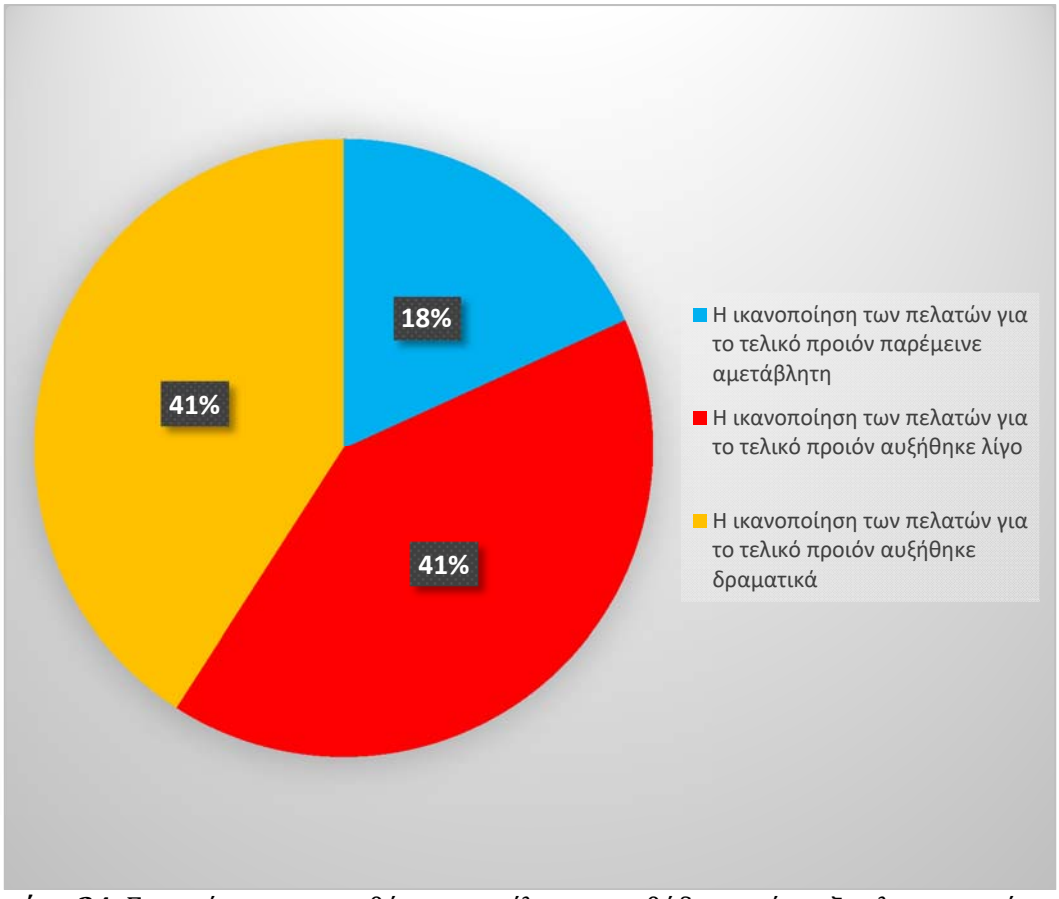


Εικόνα 23. Επιπτώσεις της υιοθέτησης ευέλικτων μεθόδων ανάπτυξης λογισμικού στο κόστος λογισμικού.

Η ικανοποίηση των πελατών από το προϊόν μετά την υιοθέτηση ευέλικτων μεθόδων περιγράφεται στην Εικόνα 24. Ποσοστό 41% των ερωτηθέντων απάντησε ότι η ικανοποίηση αυξήθηκε λίγο, το 41% ότι η ικανοποίηση αυξήθηκε δραματικά και το 18% ότι η ικανοποίηση παρέμεινε αμετάβλητη. Αξιοσημείωτο είναι ότι κανένας από τους ερωτηθέντες δεν απάντησε ότι η ικανοποίηση των πελατών μειώθηκε με την χρήση των ευέλικτων μεθόδων ανάπτυξης λογισμικού.

Συνολικά, η χρήση ευέλικτων μεθόδων από τις εταιρείες ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου έχει αξιοσημείωτα θετικά αποτελέσματα. Η πλειοψηφία των ερωτηθέντων πιστεύει ότι η παραγωγικότητα, η ποιότητα και η ικανοποίηση του πελάτη ήταν καλύτερες από πριν. Αυτό δείχνει ότι οι ευέλικτες μέθοδοι μπορούν να βελτιώσουν

τον κύκλο ζωής του λογισμικού, όταν χρησιμοποιούνται ορθά και στα κατάλληλα έργα. Μόνο το κόστος ανάπτυξης λογισμικού μεθόδους δεν μειώθηκε μετά την χρήση των νέων μεθόδων. Το 32% από τους ερωτηθέντες απάντησαν ότι το κόστος ανάπτυξης λογισμικού δεν μειώθηκε μετά την προσαρμογή των ευέλικτων μεθόδων, και άλλο ένα 32% πιστεύει ότι το κόστος μειώθηκε ελάχιστα. Αυτό δείχνει ότι το κόστος ανάπτυξης λογισμικού μπορεί να μειωθεί υπό προϋποθέσεις με την χρήση ευέλικτων μεθόδων.

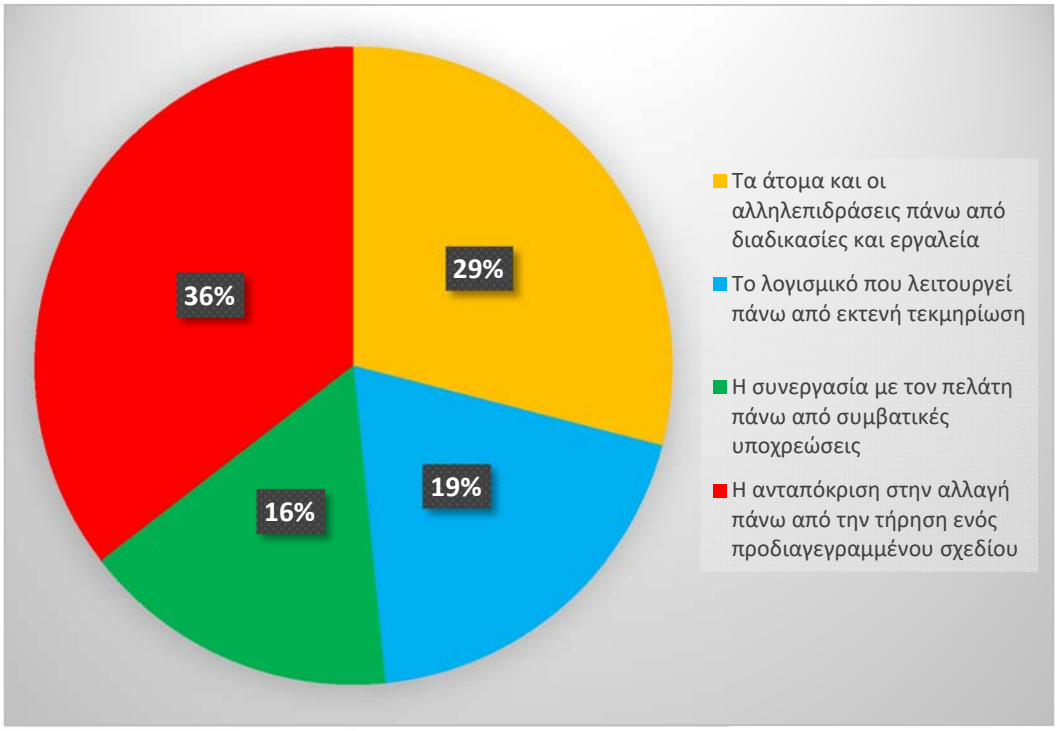


Εικόνα 24. Επιπτώσεις της υιοθέτησης ευέλικτων μεθόδων ανάπτυξης λογισμικού στην ικανοποίηση των πελατών σχετικά με το τελικό προϊόν.

4.4.3 Χαρακτηριστικά των ευέλικτων μεθόδων ανάπτυξης λογισμικού

Η επόμενη ερώτηση, διερεύνησε τα αναγκαία χαρακτηριστικά των ευέλικτων μεθόδων ανάπτυξης λογισμικού. Η πλειοψηφία των ερωτηθέντων (Εικόνα 25) απάντησε ότι η ανταπόκριση στην αλλαγή πάνω από την τήρηση ενός προδιαγεγραμμένου σχεδίου (36%) είναι το πιο αναγκαίο χαρακτηριστικό των ευέλικτων μεθόδων. Το δεύτερο πιο αναγκαίο χαρακτηριστικό είναι τα άτομα και οι αλληλεπιδράσεις πάνω από τις διαδικασίες και τα εργαλεία (29%), ενώ ποσοστό 19% απάντησε το λογισμικό που

λειτουργεί πάνω από την εκτενή τεκμηρίωση, τέλος το 16% την συνεργασία με τον πελάτη πάνω από συμβατικές διαπραγματεύσεις.



Εικόνα 25. Αναγκαία χαρακτηριστικά των ευέλικτων μεθόδων ανάπτυξης λογισμικού.

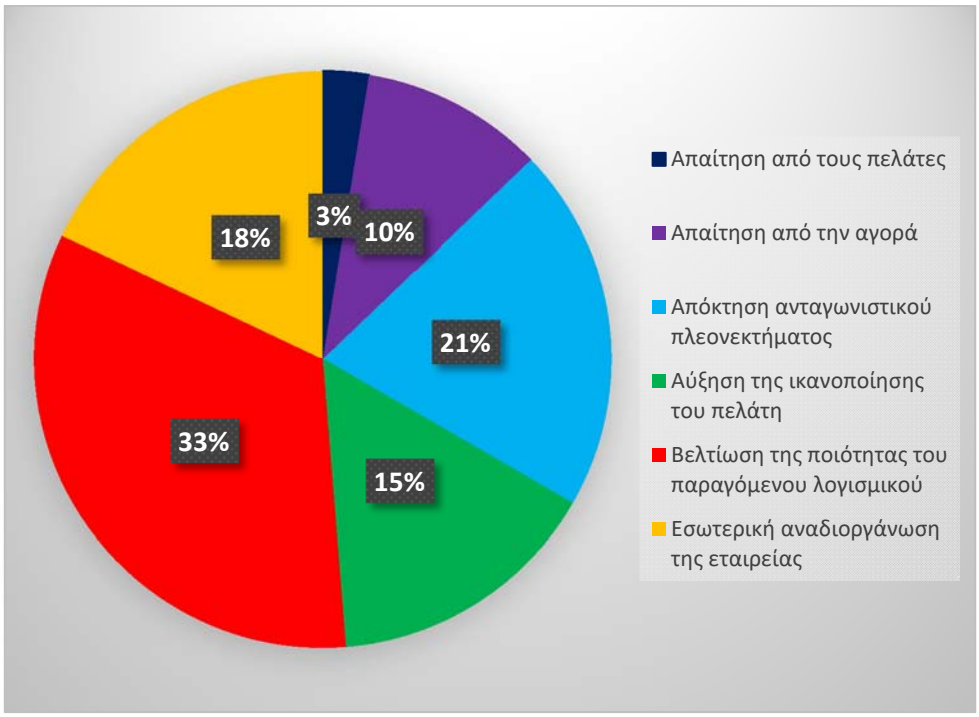
4.5 Η διασφάλιση της ποιότητας λογισμικού στις ελληνικές και κυπριακές εταιρείες πληροφορικής

Αυτό το μέρος της ανάλυσης των αποτελεσμάτων περιλαμβάνει τη μελέτη της διασφάλισης ποιότητας λογισμικού στη Ελλάδα και την Κύπρο. Οι συμμετέχοντες ρωτήθηκαν σχετικά με το σχεδιασμό διασφάλισης ποιότητας λογισμικού και τα πρότυπα διασφάλισης της ποιότητας λογισμικού που εφαρμόζουν κατά τη διάρκεια των έργων ανάπτυξης λογισμικού.

4.5.1 Πρότυπα διασφάλισης της ποιότητας λογισμικού που εφαρμόζονται στις ελληνικές και κυπριακές επιχειρήσεις

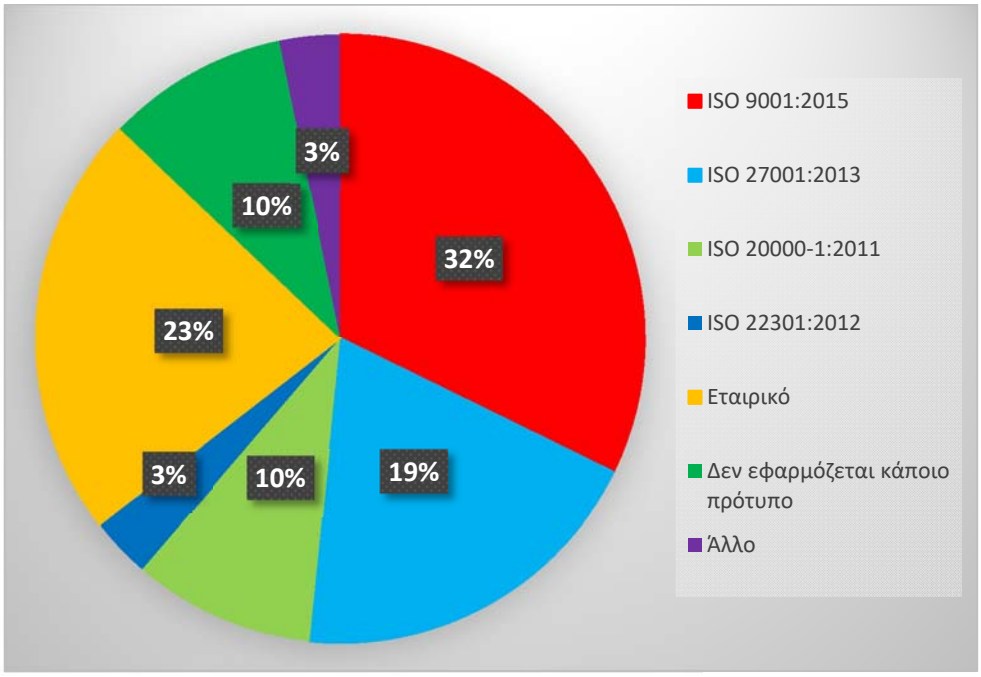
Μία ερώτηση διερεύνησε τον λόγο που ώθησε τις εταιρείες να εφαρμόσουν ένα πρότυπο διασφάλισης της ποιότητας λογισμικού. Από τις απαντήσεις (Εικόνα 26) που δόθηκαν προκύπτει ότι ο λόγος που συγκέντρωσε το μεγαλύτερο ποσοστό ήταν η βελτίωση της

ποιότητας του παραγόμενου λογισμικού με 33%. Ακολούθησε με 21% η απόκτηση ανταγωνιστικού πλεονεκτήματος. Ενώ το 18% των ερωτηθέντων θεωρεί ότι πρότυπα εφαρμόστηκαν ως αποτέλεσμα εσωτερικής αναδιοργάνωσης και το 15% για την αύξηση της ικανοποίησης του πελάτη. Ένα ποσοστό 10% απάντησε για λόγους απαίτησης της αγοράς, ενώ το μικρότερο ποσοστό 3% απάντησε για λόγους απαίτησης από τους πελάτες.



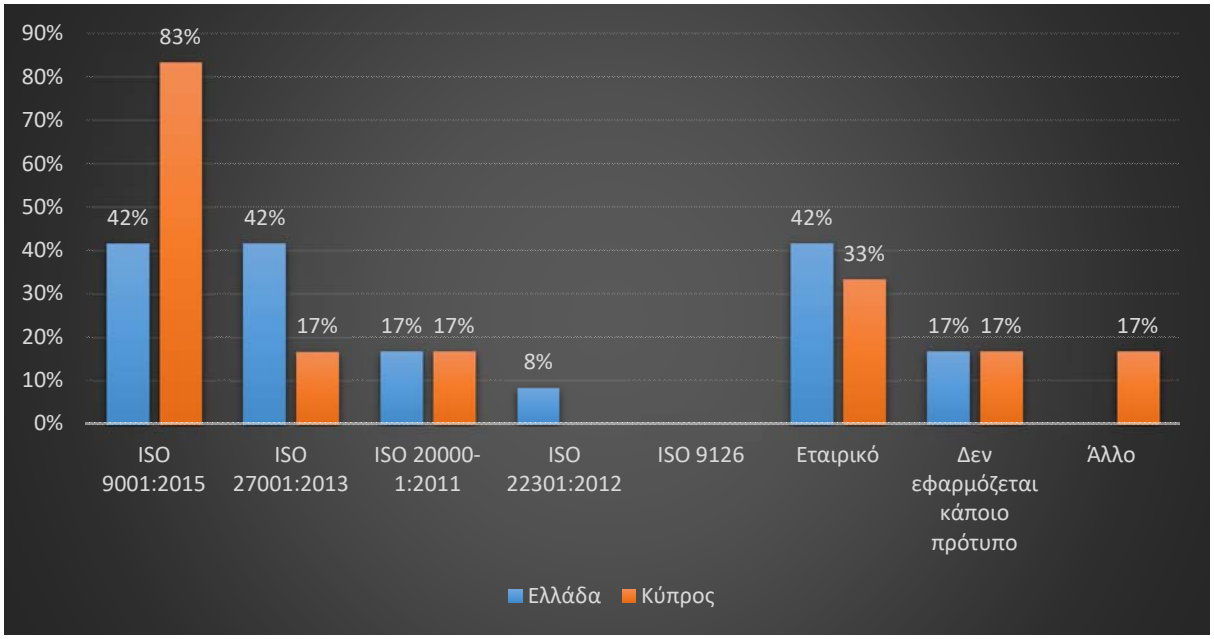
Εικόνα 26. Λόγοι υιοθέτησης κάποιου πρότυπου διασφάλισης της ποιότητας από ελληνικές και κυπριακές εταιρείες ανάπτυξης λογισμικού.

Η Εικόνα 27 δείχνει ποια πρότυπα διασφάλισης ποιότητας λογισμικού χρησιμοποιούνται από τις εταιρείες κατά τη διάρκεια του κύκλου ζωής του λογισμικού. Οι περισσότερες εταιρείες εφαρμόζουν το πρότυπο ISO 9001:2015 – Quality Management Systems (32%) το οποίο αποδείχθηκε το πιο διαδεδομένο. Επίσης, ποσοστό 23% ακολουθούν κάποιο εταιρικό πρότυπο και το 19% εφαρμόζουν το πρότυπο ISO 27001:2013. Το 10% εφαρμόζουν το πρότυπο ISO 20000-1:2011, ενώ το πρότυπο ISO 22301:2012 συγκέντρωσε το 3% και το ISO 9126 το 0%. Το 3% των ερωτηθέντων χρησιμοποίει άλλα πρότυπα, όπως τα πρότυπα CMMI & Tick IT. Επίσης, το 10% των εταιρειών δεν χρησιμοποίει κάποιο πρότυπο διασφάλισης της ποιότητας. Όσον αφορά σε επίπεδο χώρας δίνεται η παρακάτω Εικόνα 28.



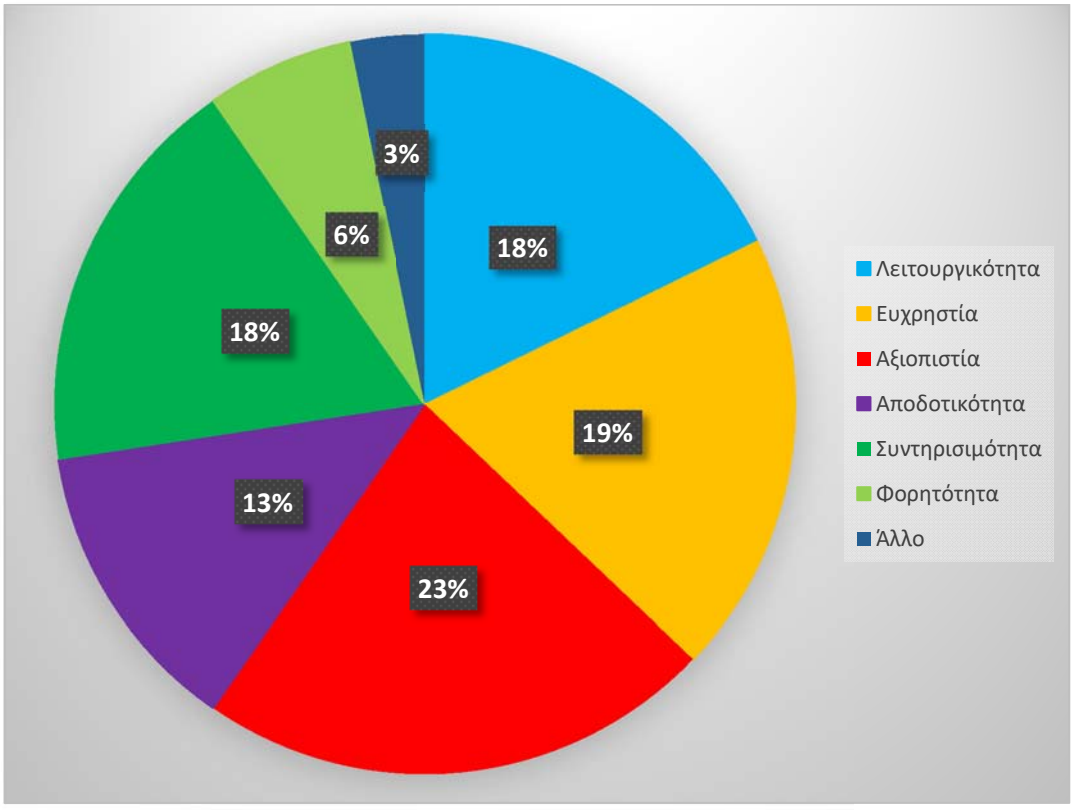
Εικόνα 27. Πρότυπα διασφάλισης ποιότητας που εφαρμόζονται από ελληνικές και κυπριακές εταιρείες ανάπτυξης λογισμικού.

Από την Εικόνα 28 παρατηρούμε ότι οι εταιρείες της Ελλάδας, είτε εφαρμόζουν το πρότυπο ISO 9001:2015, είτε το ISO 27001:2013, είτε κάποιο εταιρικό πρότυπο σε ποσοστό 42% το καθένα. Οι κυπριακές εταιρείες χρησιμοποιούν περισσότερο το πρότυπο ISO 9001:2015 σε ποσοστό 83% και εταιρικά πρότυπα σε ποσοστό 33%.



Εικόνα 28. Πρότυπα διασφάλισης ποιότητας που εφαρμόζονται στην Ελλάδα και την Κύπρο.

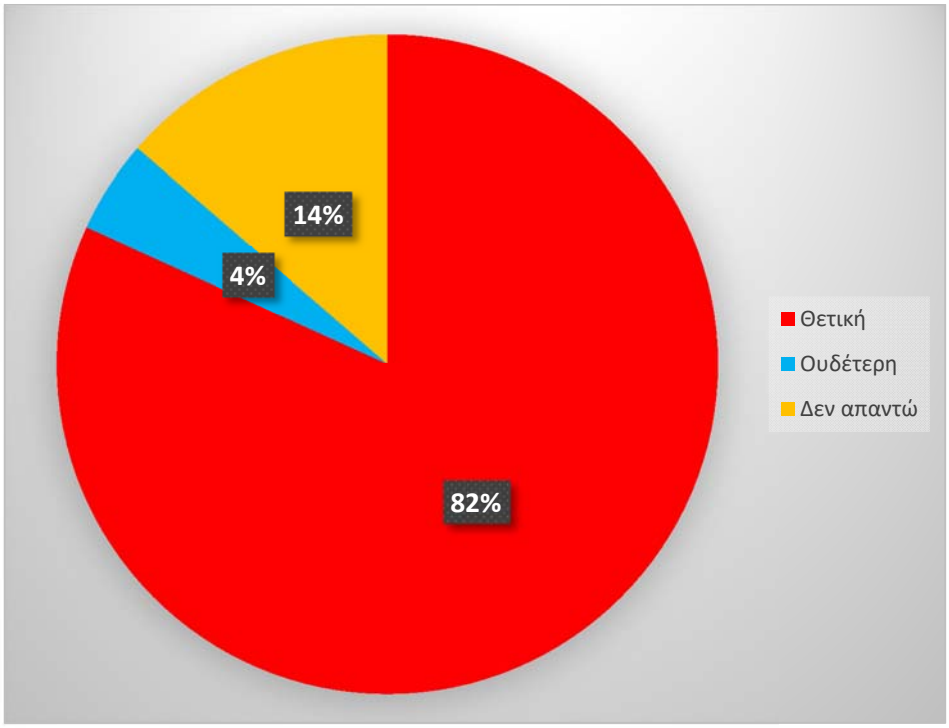
Οι συμμετέχοντες απαντησαν επίσης σχετικά με τα σημαντικότερα χαρακτηριστικά της ποιότητας του λογισμικού. Οι πιθανές επιλογές απαντήσεων λήφθηκαν από τα χαρακτηριστικά του προτύπου ISO 9126, τα οποία περιλαμβάνουν την λειτουργικότητα, την ευχρηστία, την αξιοπιστία, την αποδοτικότητα, την συντηρησιμότητα και την φορητότητα (Εικόνα 29). Τα αποτελέσματα δείχνουν ότι οι πιο δημοφιλείς παράγοντες σε φθίνουσα σειρά ήταν η αξιοπιστία με ποσοστό 23%, ακολουθούμενη από την ευχρηστία με 19%, την λειτουργικότητα και την συντηρισιμότητα με 18%, την φορητότητα με 6% και τέλος την απάντηση Άλλο (διασυνδεσιμότητα-επεκτασιμότητα) με ποσοστό 3%.



Εικόνα 29. Τα σημαντικότερα χαρακτηριστικά ποιότητας λογισμικού από ελληνικές και κυπριακές εταιρείες πληροφορικής.

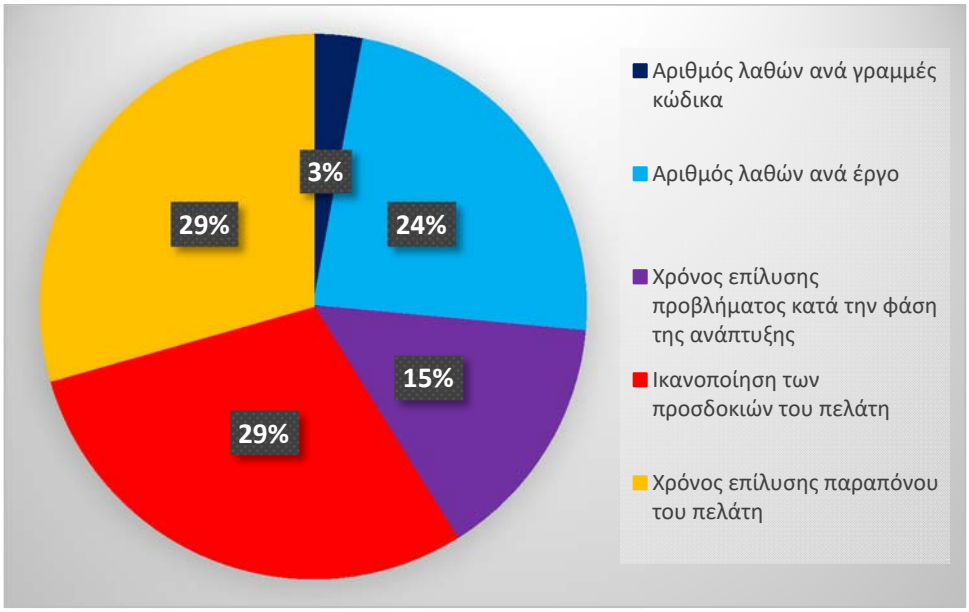
Η επόμενη ερώτηση (Εικόνα 30) διερεύνησε την επιρροή που έχει η υιοθέτηση σύγχρονων μεθόδων ανάπτυξης λογισμικού στην ποιότητα του. Η πλειοψηφία των ερωτηθέντων (82%) συμφωνεί ότι η υιοθέτηση ευέλικτων μεθόδων στην ανάπτυξη

λογισμικού έχει θετική επιρροή στην ποιότητα λογισμικού. Ποσοστό 4% πιστεύει ότι έχει ουδέτερη επιρροή, ενώ ποσοστό 14% των ερωτηθέντων δεν απάντησε.



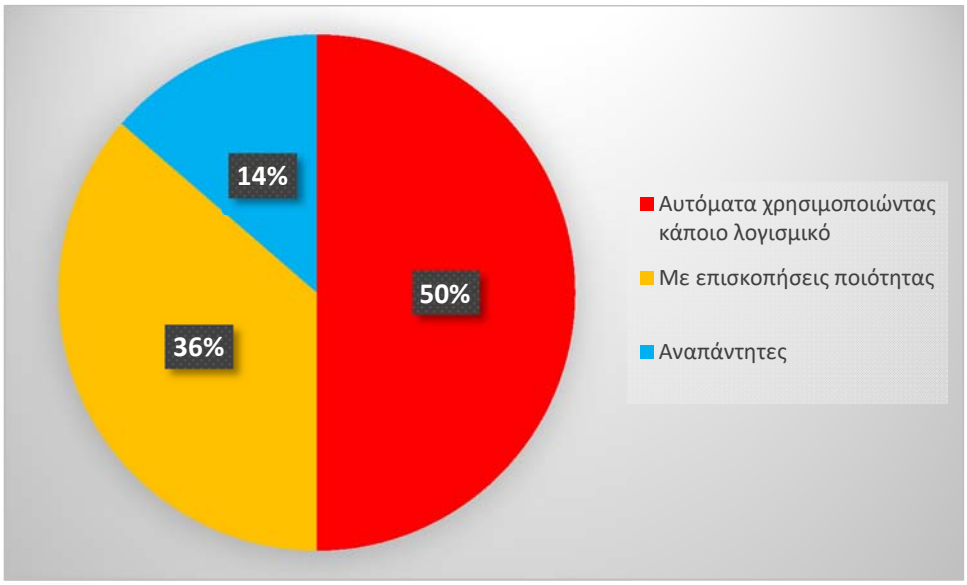
Εικόνα 30. Η επιρροή που έχει η υιοθέτηση σύγχρονων μεθόδων ανάπτυξης λογισμικού στην ποιότητα του.

Οι μετρητές ποιότητας είναι ένας πολύ σημαντικός παράγοντας που επιδρά στην συνολική ποιότητα του προϊόντος. Στην ανάπτυξη του λογισμικού χρησιμοποιούνται ποικίλες μέθοδοι ως μετρητές ποιότητας από τον χρόνο επίλυσης κάποιου προβλήματος στο έργο, μέχρι την ικανοποίηση των προσδοκιών του πελάτη και των αριθμό των λαθών στις γραμμές του κώδικα του προϊόντος. Οι ερωτηθέντες απάντησαν σχετικά με τον τρόπο αξιολόγησης της ποιότητα του παραγόμενου λογισμικού (Εικόνα 31). Το μεγαλύτερο ποσοστό των εταιρειών χρησιμοποιεί είτε τον χρόνο επίλυσης του παραπόνου του πελάτη, είτε την ικανοποίηση των προσδοκιών του πελάτη σε ποσοστό 29% το καθένα. Ακολουθεί ποσοστό 24% που κάνει χρήση του αριθμό λαθών ανά έργο, ενώ ποσοστό 15% απάντησε ότι χρησιμοποιεί τον χρόνο επίλυσης προβλήματος κατά την φάση της ανάπτυξης και 3% τον αριθμό λαθών ανά γραμμές κώδικα. Αξιοσημείωτο επίσης είναι ότι το κόστος ποιότητας δεν λαμβάνεται υπόψη στο δείγμα ως μετρητής ποιότητας.



Εικόνα 31. Ποσοστά εταιρειών που κάνουν χρήση διαφόρων μετρητών ποιότητας λογισμικού.

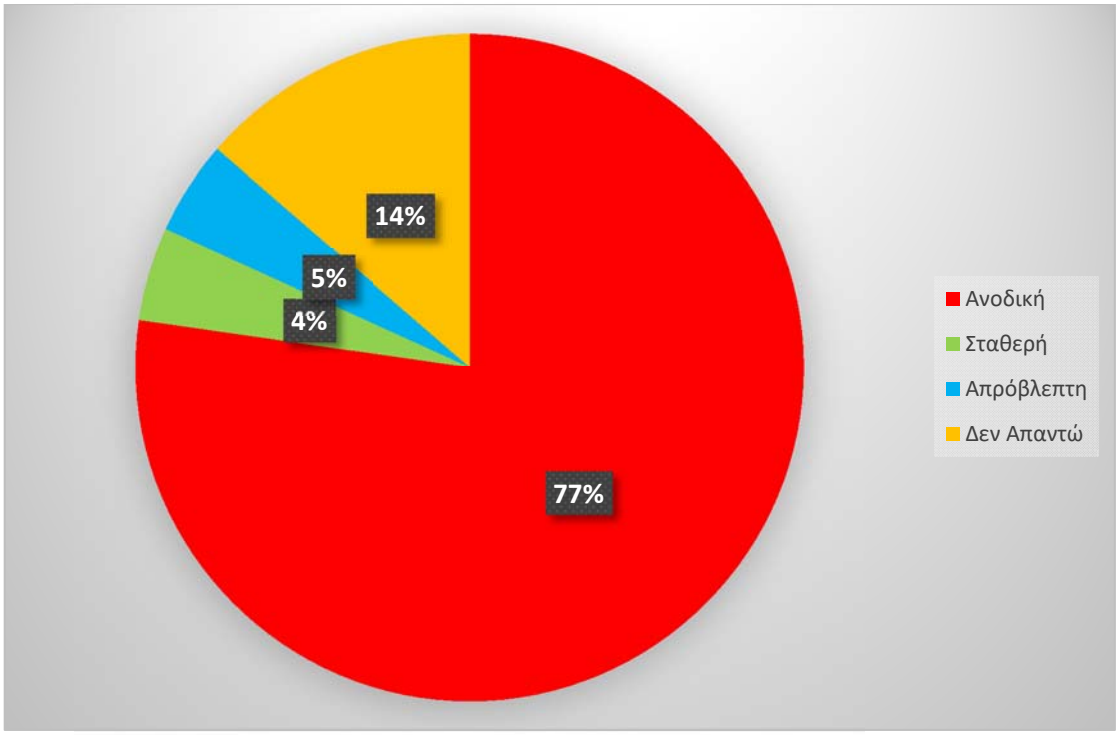
Η Εικόνα 32 δείχνει τα ποσοστά των εταιρειών που διεξάγουν δοκιμές λογισμικού είτε αυτόματα με την βοήθεια κάποιου άλλου λογισμικού, είτε χειροκίνητα κυρίως με επισκοπήσεις ποιότητας. Τα αποτελέσματα δείχνουν ότι το 50% των εταιρειών του δείγματος χρησιμοποιούν αυτόματες δοκιμές με χρήση λογισμικού, ενώ το 36% χρησιμοποιούν επισκοπήσεις ποιότητας. Υπήρξε και ένα ποσοστό 14% που δεν απάντησε.



Εικόνα 32. Ποσοστά εταιρειών που κάνουν χρήση είτε αυτόματων, είτε χειροκίνητων μεθόδων μέτρησης της ποιότητας λογισμικού.

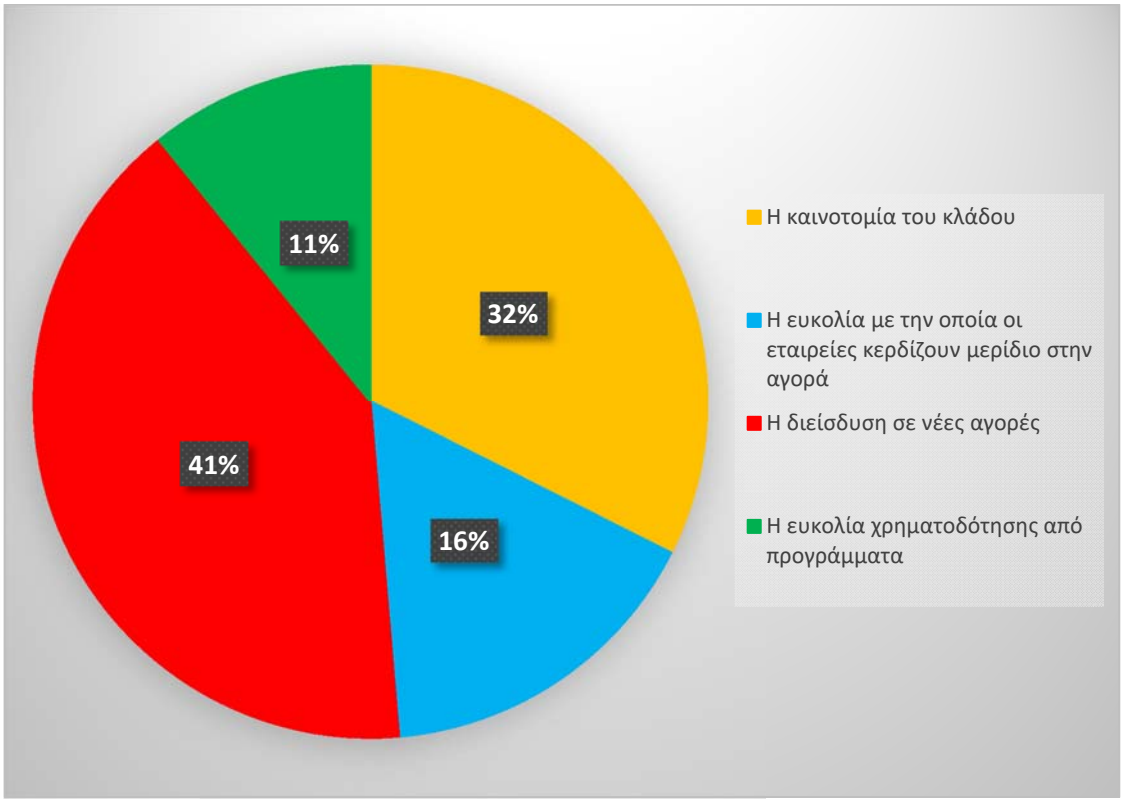
4.6 Μελλοντική κατάσταση των ελληνικών και κυπριακών εταιρειών ανάπτυξης λογισμικού

Οι ερωτηθέντες ρωτήθηκαν σχετικά με την μελλοντική κατάσταση του κλάδου εταιρειών ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου. Το 77% των ερωτηθέντων πιστεύουν ότι ο κλάδος εταιρειών ανάπτυξης λογισμικού μελλοντικά θα έχει ανοδική πορεία (Εικόνα 33).



Εικόνα 33. Ποσοστά απαντήσεων σχετικά με την μελλοντική κατάσταση του κλάδου εταιρειών ανάπτυξης λογισμικού.

Η Εικόνα 34 δείχνει τον λόγο που ο κλάδος εταιρειών ανάπτυξης λογισμικού εμφανίζει πλεονεκτήματα στην αγορά εργασίας. Τα αποτελέσματα δείχνουν ότι το 41% των ερωτηθέντων πιστεύει ότι αυτό οφείλεται στην ευκολία διείσδυσης των εταιρειών σε νέες αγορές, όπως π.χ. η αγορά των έξυπνων συσκευών, ενώ το 32% πιστεύει ότι αυτό οφείλεται στην καινοτομία του κλάδου και ποσοστό 16% πιστεύει ότι οφείλεται στην ευκολία αύξησης του μεριδίου της αγοράς. Τέλος, ένα 11% πιστεύει ότι οφείλεται στην ευκολία χρηματοδότησης από ευρωπαϊκούς και εθνικούς πόρους.



Εικόνα 34. Ποσοστά απαντήσεων σχετικά με τον λόγο που οι εταιρείες ανάπτυξης λογισμικού έχουν πλεονέκτημα στην αγορά εργασίας.

Κεφάλαιο 5
Συμπεράσματα - Συζήτηση

5.1 Γενικά

Αυτό το κεφάλαιο συζητά τα αποτελέσματα της έρευνας. Η έρευνα περιορίστηκε σε εταιρείες που βρίσκονται στη Ελλάδα και την Κύπρο και έχουν ως κύρια δραστηριότητα την ανάπτυξη λογισμικού. Στόχος ήταν να απαντηθούν τρία ερευνητικά ερωτήματα που καθορίστηκαν στην αρχή της παρούσας μεταπτυχιακής διατριβής. Τα αποτελέσματα συγκρίνονται με παρόμοιες μελέτες. Σκοπός είναι να συγκρίνει κανείς πώς διαφοροποιείται η ανάπτυξη λογισμικού σε ελληνικές και κυπριακές εταιρείες σε σχέση και με άλλες χώρες και ποιες ομοιότητες υπάρχουν. Οι άλλες μελέτες που τα αποτελέσματά τους συγκρίθηκαν με τα αποτελέσματα της παρούσας έρευνας παρουσιάζονται στον Πίνακας 4.

Πίνακας 4. Μελέτες που τα αποτελέσματά τους συγκρίθηκαν με τα αποτελέσματα της παρούσας έρευνας.

Έρευνα	Κλίμακα/Χώρα	Έτος	Αριθμός Απαντήσεων	Περιοχή εστίασης
Stelzer, et al., (1996)	Ευρώπη	1996	36	Ευέλικτες μέθοδοι ανάπτυξης λογισμικού
Johnson, (2003)	Παγκόσμια	2003	131	Ευέλικτες μέθοδοι ανάπτυξης λογισμικού
Awad, (2005)	-	2005	15	Ευέλικτες μέθοδοι ανάπτυξης λογισμικού

				& Πρότυπα Ποιότητας
Schindler, (2008)	Αυστρία	2008	61	Ευέλικτες μέθοδοι ανάπτυξης λογισμικού
French Scrum User Group, (2009)	Γαλλία	2009	230	Ευέλικτες μέθοδοι ανάπτυξης λογισμικού
Laanti, et al., (2010)	Παγκόσμια	2010	1000	Ευέλικτες μέθοδοι ανάπτυξης λογισμικού
Rodríguez, et al., (2011)	Φιλανδία	2011	408	Ευέλικτες & λιτές μέθοδοι ανάπτυξης λογισμικού
Garousi & Zhi, (2012)	Καναδάς	2012	246	Πρακτικές δοκιμής λογισμικού
Yli-Huumo, (2013)	Νότια Κορέα	2013	32	Ευέλικτες μέθοδοι ανάπτυξης λογισμικού & Πρότυπα Ποιότητας

5.2 Χρήση μοντέλων κύκλου ζωής λογισμικού και χρησιμοποιούμενες μέθοδοι στη Ελλάδα και την Κύπρο

Ποιες μέθοδοι ανάπτυξης λογισμικού και πρότυπα διασφάλισης της ποιότητας χρησιμοποιούνται από εταιρείες ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου κατά τη διάρκεια του κύκλου ζωής του λογισμικού;

Τα αποτελέσματα της παρούσας έρευνας δείχνουν ότι οι ευέλικτες μέθοδοι χρησιμοποιούνται μαζί με τις παραδοσιακές στις εταιρείες του κλάδου ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου. Οι πιο δημοφιλείς ευέλικτες μέθοδοι περιλαμβάνουν την Scrum και την Λιτή παραγωγή, ενώ η πιο δημοφιλής παραδοσιακή

μέθοδος είναι ο καταρράκτης και η μέθοδος της λειτουργικής επαύξησης. Η παρούσα έρευνα έδειξε ότι σταδιακά οι ευέλικτες μέθοδοι ανάπτυξης λογισμικού αντικαθιστούν τις παραδοσιακές. Άλλες παρόμοιες έρευνες επίσης δείχνουν ότι η χρήση ευέλικτων μεθόδων σταδιακά αντικαθιστούν τις παραδοσιακές μεθόδους (Schindler, 2008), (Rodríguez, et al., 2011), (Garousi & Zhi, 2012), (Yli-Huumo, 2013). Αυτό δείχνει ότι ο κλάδος εταιρειών ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου ακολουθεί την τάση που επικρατεί και σε άλλες χώρες και είναι διατεθειμένος να μάθει και να δοκιμάσει νέες ευέλικτες μεθόδους ανάπτυξης λογισμικού.

Στην έρευνα υπήρξε ερώτηση αν οι εταιρείες της Ελλάδας και της Κύπρου διαθέτουν ξεχωριστό τμήμα διασφάλισης ποιότητας και ποια πρότυπα διασφάλισης ποιότητας λογισμικού εφαρμόζονται. Τα αποτελέσματα ήταν αξιοσημείωτα, καθώς το 68% των εταιρειών διαθέτουν τμήμα διασφάλισης ποιότητας λογισμικού και μόνο το 32% από τις εταιρείες που ρωτήθηκαν δεν διαθέτουν. Αυτά τα αποτελέσματα είναι σημαντικά, επειδή η διασφάλιση της ποιότητας του λογισμικού είναι σημαντικό μέρος του κύκλου ζωής του λογισμικού και δείχνουν ότι οι ελληνικές και κυπριακές επιχειρήσεις ανάπτυξης λογισμικού έχουν αντιληφθεί τα πλεονεκτήματα που εμφανίζει η χρήση προτύπων στην διασφάλιση της ποιότητας των προϊόντων τους.

Ένα άλλο ενδιαφέρον αποτέλεσμα ήταν η χρήση προτύπων διασφάλισης της ποιότητας στον κύκλο ζωής του λογισμικού. Σημαντικό ποσοστό των εταιρειών της Ελλάδας και της Κύπρου χρησιμοποιούν το γενικό πρότυπο ISO 9001:2015. Σημαντικό ποσοστό του δείγματος, αντί των γνωστών προτύπων χρησιμοποιούν εταιρικά πρότυπα ποιότητας σε ποσοστό 23%. Αξιοσημείωτο όμως είναι και το γεγονός ότι ποσοστό 17% των εταιρειών ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου και 10% του συνολικού δείγματος δεν χρησιμοποιεί κάποιο πρότυπο. Επίσης, δεν χρησιμοποιούνται αρκετά ή καθόλου εξειδικευμένα πρότυπα, που αφορούν την ανάπτυξη του λογισμικού, όπως το ISO 9126 ή το ISO 20001:2011. Πιθανόν αυτό να συμβαίνει κυρίως λόγο της έλλειψης πληροφόρησης για τα συγκεκριμένα πρότυπα. Στην περίπτωση όπου δεν χρησιμοποιούνται πρότυπα λόγω έλλειψης γνώσεων, είναι πραγματικά σημαντικό να ξεκινήσει η εκπαίδευση για αυτά. Μελέτη από τους Stelzer, et al., (1996) δείχνει ότι με τα πρότυπα διασφάλισης ποιότητας λογισμικού, είναι δυνατόν να βελτιωθεί σημαντικά η ποιότητα των έργων.

Η έρευνα αποκάλυψε επίσης ότι οι εταιρείες του δείγματος της Ελλάδας και της Κύπρου χρησιμοποιούν μετρικές για την διασφάλιση της ποιότητας του λογισμικού. Επίσης, χρησιμοποιούν περισσότερο αυτοματοποιημένους ελέγχους σε εταιρικό επίπεδο παρά επισκοπήσεις ποιότητας. Σε σύγκριση με την έρευνα Garousi & Zhi, (2012), παρατηρείται παρόμοια τάση, οι αυτοματοποιημένοι έλεγχοι να υπερτερούν από τις επισκοπήσεις ποιότητας.

5.3 Επίδραση των ευέλικτων μεθόδων στον κύκλο ζωής του λογισμικού

Πώς επηρεάζει η χρήση ευέλικτων μεθόδων τον κύκλο ζωής του λογισμικού;

Οι ευέλικτες μέθοδοι αρχίζουν και γίνονται δημοφιλείς μεταξύ εταιρειών του κλάδου ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου. Είναι σημαντικό να συζητηθούν οι επιπτώσεις των ευέλικτων μεθόδων σε έργα ανάπτυξης λογισμικού. Τα στοιχεία που συγκεντρώθηκαν από εταιρείες ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου, δείχνουν ότι οι ευέλικτες μέθοδοι αυξάνουν την ποιότητα, την παραγωγικότητα και την ικανοποίηση των πελατών και, υπό προϋποθέσεις δύναται να μειώσουν το κόστος, όταν χρησιμοποιούνται σε έργα λογισμικού. Οι Johnson, (2003), Awad, (2005), Laanti, et al., (2010), Garousi & Zhi, (2012) και Yli-Huumo, (2013), συγκέντρωσαν παρόμοια αποτελέσματα στην έρευνά τους, όπου η παραγωγικότητα, η ποιότητα και η ικανοποίηση των πελατών ήταν σημαντικά καλύτερες, ενώ το κόστος παρέμεινε αμετάβλητο.

Μπορούμε να συμπεράνουμε ότι η χρήση ευέλικτων μεθόδων επηρεάζει θετικά τα έργα ανάπτυξης λογισμικού. Αυτό σημαίνει ότι οι εργαζόμενοι, πίσω από την ευέλικτη μέθοδο πρέπει να διαθέτουν ένα ορισμένο επίπεδο δεξιοτήτων και γνώσης κατά τη χρήση της. Η εμπειρία και η γνώση των ευέλικτων μεθόδων από τις εταιρείες ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου παρουσιάζει ενδιαφέροντα αποτελέσματα. Οι εταιρείες φαίνεται να έχουν μακρόχρονη (3-10 έτη) εμπειρία ως προς την χρήση ευέλικτων μεθόδων, αλλά εξακολουθούν να θεωρούν ότι το επίπεδο εμπειρίας τους σε αυτές βρίσκεται σε μέσο επίπεδο. Αυτό δείχνει ότι δεν χρειάζεται απαραίτητα υψηλό επίπεδο

γνώσης στις ευέλικτες μεθόδους για να αρχίσουν να διαφαίνονται θετικά αποτελέσματα σε έργα ανάπτυξης λογισμικού. Αυτά τα αποτελέσματα δείχνουν επίσης ότι εταιρείες ανάπτυξης λογισμικού χρειάζονται προγράμματα εκπαίδευσης των υπαλλήλων τους σε διάφορες πρακτικές πτυχές των ευέλικτων μεθόδων.

Η έρευνα έδειξε ότι η επικοινωνία και οι μεταβαλλόμενες απαιτήσεις είναι τα πιο αναγκαία χαρακτηριστικά των ευέλικτων μεθόδων κατά τη διαδικασία ανάπτυξης του λογισμικού. Τα αποτελέσματα της έρευνας δείχνουν ότι οι εταιρείες του δείγματος πιστεύουν ότι οι δύο παραπάνω λόγοι είναι οι πιο σημαντικοί, που αποδεικνύουν το λόγο που οι ευέλικτες μέθοδοι είναι τόσο καλές να χρησιμοποιηθούν σε έργα ανάπτυξης λογισμικού.

Με βάση τις απαντήσεις που συλλέχθηκαν, είναι ασφαλές να πούμε ότι η υιοθέτηση των ευέλικτων μεθόδων έχει θετική επίδραση στο κλάδο εταιρειών ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου. Με υψηλότερο επίπεδο εμπειρίας και γνώσης θα μπορούσε να έχει ακόμα μεγαλύτερη επίδραση. Είναι γεγονός ότι όταν οι εταιρείες αποκτήσουν καλύτερη γνώση των βασικών αρχών των ευέλικτων μεθόδων, τότε τα αποτελέσματα θα γίνουν θεαματικά.

5.4 Μελλοντική κατάσταση του κλάδου εταιρειών ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου

Πώς αναμένεται να διαμορφωθεί η μελλοντική κατάσταση του κλάδου εταιρειών ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου;

Η πλειοψηφία των ερωτηθέντων πιστεύει ότι ο κλάδος ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου καταγράφει ανοδική πορεία. Αυτή η ανοδική πορεία συμβαίνει, σύμφωνα με τους συμμετέχοντες στην έρευνα, κυρίως λόγω της εύκολης διείσδυσης σε νέες αγορές και της καινοτομίας που εμφανίζει ο κλάδος.

Η χρήση τόσο παραδοσιακών όσο και ευέλικτων μεθόδων κύκλου ζωής λογισμικού φαίνεται να αποτελεί θετικό παράγοντα στον κλάδο εταιρειών ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου. Η χρήση ποικίλων μοντέλων και μεθόδων δείχνει ότι οι εταιρείες αναζητούν πραγματικά την ορθή πρακτική για να την ταιριάξουν στη δομή της εταιρείας, αντί π.χ. να χρησιμοποιούν απλώς το βασικό μοντέλο καταρράκτη σε κάθε έργο. Κάθε έργο έχει διάφορους παράγοντες που αποφασίζουν ποιο μοντέλο ή μέθοδος ταιριάζει καλύτερα. Οι εταιρείες είναι σε θέση να εφαρμόσουν διαφορετικά μοντέλα και μεθόδους, που οδηγεί σε καλύτερα αποτελέσματα σε έργα λογισμικού.

Τα αποτελέσματα αποκάλυψαν επίσης ότι η χρήση ευέλικτων μεθόδων αυξάνει την παραγωγικότητα, την ποιότητα, την ικανοποίηση του πελάτη και σε ορισμένες περιπτώσεις το κόστος του έργου. Επίσης, το επίπεδο γνώσης αναφέρθηκε ότι ήταν κατά βάση μέσο. Αυτό δείχνει ότι ο κλάδος ανάπτυξης λογισμικού θα πρέπει να καταβάλει περισσότερες προσπάθειες για την εκμάθηση και τη χρήση των ευέλικτων μεθόδων. Αυτό το βήμα μπορεί να οδηγήσει σε ακόμα καλύτερα αποτελέσματα σε έργα, γεγονός που αυξάνει αυτόματα τη συνολική εικόνα του κλάδου ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου.

Η διασφάλιση ποιότητας λογισμικού στο κλάδο εταιρειών ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου είναι ένας παράγοντας που απαιτεί περαιτέρω έρευνα και στο μέλλον. Τα αποτελέσματα έδειξαν ότι η διασφάλιση της ποιότητας αποτελεί υψηλή προτεραιότητα, επειδή πάνω από τις μισές εταιρείες διαθέτουν σχεδιασμό διασφάλισης ποιότητας. Επίσης, η έλλειψη εξειδικευμένων προτύπων διασφάλισης ποιότητας κατά την ανάπτυξη λογισμικού είναι πραγματικά ένα γεγονός που αποδεικνύει ότι η ενημέρωση για τα πρότυπα αυτά είναι μηδαμινή. Η ενημέρωση για το πρότυπο ποιότητας, όπως το ISO 9126, δίνει τη δυνατότητα να βελτιωθεί ακόμη περισσότερο η ποιότητα του λογισμικού.

Κεφάλαιο 6
Επίλογος

Σκοπός αυτής της παρούσας μελέτης ήταν η εύρεση πληροφοριών σχετικά με την κατάσταση του κλάδου εταιρειών ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου. Η μελέτη διεξήχθη με τη μέθοδο της συμπλήρωσης ερωτηματολογίων και τα δεδομένα που συλλέχθηκαν χρησιμοποιήθηκαν για να απαντήσουν στις ερευνητικές ερωτήσεις. Ο συνολικός αριθμός του δείγματος ήταν 22 εταιρείες. Οι τρεις ερευνητικές ερωτήσεις που απαντήθηκαν είναι:

1. Ποιες μέθοδοι ανάπτυξης λογισμικού και πρότυπα διασφάλισης της ποιότητας χρησιμοποιούνται από εταιρείες ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου κατά τη διάρκεια του κύκλου ζωής του λογισμικού;
2. Πώς η χρήση των ευέλικτων μεθόδων ανάπτυξης λογισμικού επηρεάζει τον κύκλο ζωής του στην Ελλάδα και την Κύπρο;
3. Πώς αναμένεται να διαμορφωθεί η μελλοντική κατάσταση του κλάδου εταιρειών ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου;

Όσον αφορά την πρώτη ερευνητική ερώτηση, η έρευνα διερεύνησε το είδος των μοντέλων κύκλου ζωής του λογισμικού και ποιες μέθοδοι χρησιμοποιούνται από τις εταιρείες του δείγματος. Τα αποτελέσματα της έρευνας έδειξαν ότι η χρήση ευέλικτων μεθόδων σταδιακά υποσκελίζουν τη χρήση παραδοσιακών μεθόδων ανάπτυξης λογισμικού. Αυτό δείχνει ότι οι εταιρείες της Ελλάδας και της Κύπρου αρχίζουν σταδιακά, να υιοθετούν πλέον και τη χρήση ευέλικτων μεθόδων ανάπτυξης λογισμικού στα έργα τους. Μία άλλη ενότητα του ερωτηματολογίου αφορούσε ερωτήσεις σχετικά με την διασφάλιση της ποιότητας του λογισμικού. Η έρευνα αποκάλυψε ότι γενικά οι εταιρείες

της Ελλάδας και της Κύπρου διαθέτουν ξεχωριστό τμήμα διασφάλισης ποιότητας λογισμικού. Οι περισσότερες εταιρείες χρησιμοποιούν το γενικό πρότυπο ISO 9001 και όχι εξειδικευμένα πρότυπα σχετικά με τις μεθόδους ανάπτυξης λογισμικού, όπως π.χ. το ISO 9126.

Για το δεύτερο ερευνητικό ερώτημα, η έρευνα αξιολόγησε το πως η χρήση ευέλικτων μεθόδων επηρεάζει την παραγωγικότητα, την ποιότητα, την ικανοποίηση του πελάτη και το κόστος ενός έργου λογισμικού. Τα αποτελέσματα αποκάλυψαν ότι με τη χρήση ευέλικτων μεθόδων, η παραγωγικότητα της εταιρείας, η ποιότητα του λογισμικού και η ικανοποίηση των πελατών για το τελικό προϊόν βελτιώθηκαν. Αυτά είναι ενδιαφέροντα αποτελέσματα, δεδομένου ότι οι ερωτηθέντες δεν περιέγραψαν την εμπειρία τους, όσον αφορά τις ευέλικτες μεθόδους ως υψηλή. Αυτό δείχνει ότι με τη μέση εμπειρία, οι ευέλικτες μέθοδοι μπορούν να κάνουν τα έργα ανάπτυξης λογισμικού αποτελεσματικότερα. Οι ερωτηθέντες πιστεύουν ότι τα μεγαλύτερα πλεονεκτήματα των ευέλικτων μεθόδων είναι η ικανότητα να ανταποκρίνονται στις μεταβαλλόμενες απαιτήσεις και στην καλύτερη επικοινωνία.

Για την τελευταία ερευνητική ερώτηση, οι ερωτηθέντες ρωτήθηκαν τι σκέφτονται σχετικά με την μελλοντική κατάσταση του κλάδου εταιρειών ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου και ποια είναι τα μεγαλύτερα πλεονεκτήματα του κλάδου. Η συντριπτική πλειοψηφία των ερωτηθέντων απάντησε ότι η μελλοντική κατάσταση κλάδο εταιρειών ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου αναμένεται να είναι ανοδική. Οι ερωτηθέντες πιστεύουν ότι ο μεγαλύτερος λόγος για αυτό είναι η εύκολη διείσδυση που έχουν σε νέες αγορές, καθώς και η καινοτομία που εμφανίζει ο κλάδος των εταιρειών. Ο κλάδος εταιρειών ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου διαθέτει αξιόλογο εργατικό δυναμικό που είναι και ταλαντούχο και διαθέτει την κατάλληλη υποδομή.

Το γενικό συμπέρασμα είναι ότι ο κλάδος εταιρειών ανάπτυξης λογισμικού της Ελλάδας και της Κύπρου έχει σαφή πλεονεκτήματα, που θα επηρεάσουν την μελλοντική κατάσταση του. Σημαντικά πλεονεκτήματα, πέρα από αυτά της παρούσας έρευνας περιλαμβάνουν το εργατικό δυναμικό, την υποδομή και την ποικιλομορφία στα μοντέλα

και τις μεθόδους κύκλου ζωής λογισμικού που χρησιμοποιούν. Οι εταιρείες είναι πρόθυμες να δοκιμάσουν διαφορετικά στυλ ανάπτυξης για να βρουν το καλύτερο για τα έργα τους. Κάποιες αδυναμίες που προέκυψαν από τις συνεντεύξεις περιλαμβάνουν τη μεταχείριση του εργατικού δυναμικού, την έλλειψη ρευστότητας και επενδύσεων, την έλλειψη κυβερνητικής πολιτικής και υποστήριξης και την οικονομική κρίση.

Παράρτημα Α
Ερωτηματολόγιο

Έρευνα σε εταιρείες Πληροφορικής, που βρίσκονται στην Ελλάδα και την Κύπρο σχετικά με τις Μεθοδολογίες Ανάπτυξης Λογισμικού και την Διασφάλιση της Ποιότητας.

1. **Όνομα Εταιρείας** (προαιρετικό)

2. **E-mail** (προαιρετικό)

3. **Θέση στην Εταιρεία**

4. **Τμήμα εταιρείας**

5. **Παρακαλώ σημειώστε τον αριθμό εργαζομένων της εταιρείας:**

( ) 1-10
( ) 11-100
( ) 100-1000
( ) >1000

6. **Χώρα εταιρείας:**

( ) Ελλάδα
( ) Κύπρο
( ) Άλλο:

Μεθοδολογίες Ανάπτυξης Λογισμικού στην Ελλάδα και την Κύπρο.

7. Ποια από τα παρακάτω Παραδοσιακά μοντέλα κύκλου ζωής λογισμικού έχουν χρησιμοποιηθεί από την εταιρεία;

| | Το μοντέλο του Καταρράκτη - Αλληλουχία διαδοχικών διακριτών βημάτων ανάπτυξης

| | Το Σπειροειδές μοντέλο - Εξελικτική διαδικασία διαδοχικών βελτιώσεων ενός πρωτοτύπου

| | Το μοντέλο της Προτυποποίησης - Ανάπτυξη του λογισμικού σε τμήματα / πρότυπα

| | Το μοντέλο της Λειτουργικής Επαύξησης - Αυξητική ανάπτυξη των λειτουργιών του λογισμικού

| | Το μοντέλο του Πίδακα - Αντικειμενοστραφή φιλοσοφία και χρησιμοποίηση ετοίμων συστατικών

| | Άλλο:

8. Ποια από τα παρακάτω Μοντέρνα (Ευέλικτα - Agile) μοντέλα κύκλου ζωής λογισμικού έχουν χρησιμοποιηθεί από την εταιρεία;

| | Scrum

| | Extreme programming (XP)

| | Lean Software Development (LSD)

| | Feature Driven Development (FDD)

| | Dynamic System Development Method (DSDM)

| | Crystal Clear

| | Open Unified Process (Open UP)

| | Άλλο:

9. Πόσο συχνά συμμετέχουν οι πελάτες στην ανάπτυξη λογισμικού;

( ) Καθημερινά

( ) Μία φορά την εβδομάδα

( ) Μία φορά το μήνα

( ) Λίγες συναντήσεις κατά την διάρκεια της ανάπτυξης λογισμικού

( ) Άλλο:

10. Διαθέτει η εταιρεία ειδικό τμήμα διασφάλισης της ποιότητας λογισμικού;

( ) Ναι

( ) Όχι

Η υιοθέτηση ευέλικτων μεθόδων Λογισμικού από τις εταιρείες.

11. Πότε υιοθετήθηκαν οι ευέλικτες μέθοδοι λογισμικού για πρώτη φορά από την εταιρεία

( ) Πρόσφατα (π.χ. λιγότερο από 12 μήνες)

( ) Μεταξύ 1 και 2 ετών

( ) Μεταξύ 3 και 4 ετών

( ) Μεταξύ 5 και 10 ετών

( ) Πάνω από 10 έτη

12. Πως θα χαρακτηρίζατε την εμπειρία της εταιρείας σας στην υιοθέτηση ευέλικτων μεθόδων ανάπτυξης λογισμικού;

( ) Πολύ μικρή εμπειρία

( ) Μικρή εμπειρία

( ) Μέση εμπειρία

( ) Μεγάλη εμπειρία

( ) Πολύ μεγάλη εμπειρία

13. Τι επίπτωση έχει η υιοθέτηση ευέλικτων μεθόδων ανάπτυξης λογισμικού στην παραγωγικότητα της εταιρείας;

( ) Μεγάλη πτώση παραγωγικότητας

( ) Μικρή πτώση παραγωγικότητας

( ) Η παραγωγικότητα έμεινε αμετάβλητη

( ) Βελτίωση της παραγωγικότητας

( ) Θεαματική βελτίωση της παραγωγικότητας

14. Τι επίπτωση έχει η υιοθέτηση ευέλικτων μεθόδων ανάπτυξης λογισμικού στην ποιότητα του λογισμικού;

( ) Μεγάλη πτώση της ποιότητας του λογισμικού

( ) Μικρή πτώση της ποιότητας του λογισμικού

( ) Η ποιότητα του λογισμικού παρέμεινε αμετάβλητη

( ) Μικρή βελτίωση της ποιότητας του λογισμικού

( ) Θεαματική βελτίωση της ποιότητας του λογισμικού

15. Τι επίπτωση έχει η υιοθέτηση ευέλικτων μεθόδων ανάπτυξης λογισμικού στο κόστος ανάπτυξης λογισμικού;

( ) Το κόστος δημιουργίας του λογισμικού αυξήθηκε δραματικά

( )Το κόστος δημιουργίας του λογισμικού αυξήθηκε λίγο

( ) Το κόστος δημιουργίας του λογισμικού παρέμεινε αμετάβλητο

( ) Το κόστος δημιουργίας του λογισμικού μειώθηκε λίγο

( ) Το κόστος δημιουργίας του λογισμικού μειώθηκε δραματικά

16. Τι επίπτωση έχει η υιοθέτηση ευέλικτων μεθόδων ανάπτυξης λογισμικού στην ικανοποίηση των πελατών από το τελικό προϊόν;

( ) Η ικανοποίηση των πελατών από το τελικό προϊόν μειώθηκε δραματικά

( ) Η ικανοποίηση των πελατών από το τελικό προϊόν μειώθηκε λίγο

( ) Η ικανοποίηση των πελατών από το τελικό προϊόν παρέμεινε αμετάβλητη

( ) Η ικανοποίηση των πελατών από το τελικό προϊόν αυξήθηκε λίγο

( ) Η ικανοποίηση των πελατών από το τελικό προϊόν αυξήθηκε δραματικά

17. Ποιο χαρακτηριστικό των ευέλικτων μεθόδων ανάπτυξης λογισμικού θεωρείτε περισσότερο αναγκαίο;

| | Τα άτομα και οι αλληλεπιδράσεις πάνω από τις διαδικασίες και τα εργαλεία

| | Το λογισμικό που λειτουργεί πάνω από την εκτενή τεκμηρίωση

| | Την συνεργασία με τον πελάτη πάνω από τις συμβατικές διαπραγματεύσεις

| | Η ανταπόκριση στην αλλαγή πάνω από την τήρηση ενός προδιαγεγραμμένου σχεδίου

Διασφάλιση της Ποιότητας Λογισμικού.

18. Ποιοι λόγοι ώθησαν την εταιρεία στην εφαρμογή ενός συστήματος διασφάλισης της Ποιότητας Λογισμικού;

| | Απαίτηση από τους πελάτες
| | Απαίτηση από την αγορά
| | Απόκτηση ανταγωνιστικού πλεονεκτήματος
| | Αύξηση της ικανοποίησης του πελάτη
| | Βελτίωση της ποιότητας του παραγόμενου λογισμικού
| | Εσωτερική αναδιοργάνωση της εταιρίας
| | Άλλο:

19. Ποιο από τα παρακάτω πρότυπα χρησιμοποιούνται από την εταιρεία κατά την διαδικασία διασφάλισης της ποιότητας λογισμικού;

Επιλέξτε όλα όσα ισχύουν.

| | ISO 9001:2015 - Quality Management Systems
| | ISO 27001:2013 - Information Security Management Systems
| | ISO 20000-1:2011 - IT Service Management
| | ISO 22301:2012 - Business Continuity Management Systems
| | ISO 9126 - Software Engineering
| | Εταιρικό
| | Δεν εφαρμόζεται κάποιο πρότυπο
| | Άλλο:

20. Ποιους από τους παρακάτω παράγοντες θεωρείτε πιο σημαντικούς για την διασφάλιση της ποιότητας λογισμικού;

| | Λειτουργικότητα - Το Λογισμικό παρέχει λειτουργίες, που ικανοποιούν τις απαιτήσεις του χρήστη
| | Ευχρηστία - Το Λογισμικό παρέχει τις λειτουργίες του με τρόπο αποδοτικό και αποτελεσματικό

| | Αξιοπιστία - Το Λογισμικό παράγει αξιόπιστα αποτελέσματα

| | Αποδοτικότητα - Το Λογισμικό διατηρεί υψηλά επίπεδα επεξεργασίας και απόδοσης

| | Συντηρησιμότητα - Το Λογισμικό δέχεται διόρθωση & προσθήκη λειτουργικών χαρακτηριστικών

| | Φορητότητα - Το Λογισμικό είναι δυνατό να μεταφερθεί από ένα περιβάλλον σε άλλο

| | Άλλο:

21. Τι επιρροή πιστεύετε ότι έχει η εφαρμογή σύγχρονων μεθόδων ανάπτυξης λογισμικού στην διασφάλιση της ποιότητάς του;

| | Θετική

| | Αρνητική

| | Ουδέτερη

22. Ποιες μετρήσεις ποιότητας εφαρμόζονται από την εταιρεία κατά την διάρκεια της ανάπτυξης λογισμικού;

| | Αριθμός λαθών ανά γραμμές κώδικα

| | Αριθμός λαθών ανά έργο

| | Χρόνος επίλυσης προβλήματος κατά τη φάση της ανάπτυξης

| | Ικανοποίηση των προσδοκιών του πελάτη

| | Χρόνος επίλυσης παραπόνου του πελάτη

| | Κόστος ποιότητας

| | Άλλο:

23. Πως εφαρμόζεται ο έλεγχος ποιότητας στο τελικό προϊόν;

| | Με αυτοματοποιημένη αξιολόγηση και πραγματοποίηση μετρήσεων λογισμικού

| | Με επισκοπήσεις ποιότητας

Μελλοντική κατάσταση του κλάδου εταιρειών ανάπτυξης λογισμικού.

24. Πως θα χαρακτηρίζατε την μελλοντική πορεία του κλάδου των εταιρειών ανάπτυξης λογισμικού;

Να επισημαίνεται μόνο μία έλλειψη.

( ) Ανοδική

( ) Καθοδική

( ) Στάσιμη

( ) Απρόβλεπτη

25. Ποια πιστεύετε ότι είναι τα μεγαλύτερα πλεονεκτήματα του που εμφανίζουν οι εταιρείες ανάπτυξης λογισμικού;

| | Η καινοτομία των εταιρειών, η οποία περιορίζει τις επιπτώσεις τις οικονομικής κρίσης

| | Η ευκολία με την οποία οι εταιρείες καταφέρνουν να κερδίσουν μερίδιο στην αγορά

| | Οι προσπάθειες διείσδυσης των εταιρειών ανάπτυξης λογισμικού σε νέες αγορές

| | Η ευκολία χρηματοδότησης των εταιρειών μέσω ευρωπαϊκών και εθνικών προγραμμάτων

| | Άλλο:

26. Έχετε κάτι άλλο να συμπληρώσετε που θα βοηθούσε την έρευνα;

27. Στην περίπτωση που θα επιθυμούσατε να σας αποσταλεί μία μικρή περίληψη των αποτελεσμάτων της έρευνας παρακαλώ σημειώστε παρακάτω:

( ) Επιθυμώ να μου αποσταλεί μία μικρή περίληψη των αποτελεσμάτων της διατριβής

( ) Δεν επιθυμώ να μου αποσταλεί μία μικρή περίληψη των αποτελεσμάτων της διατριβής

Βιβλιογραφία

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J., 2002. *Agile software development methods: Review and Analysis,* Espoo, Finland: VTT Publications.

Awad, M., 2005. *A comparison between agile and traditional software development methodologies,* Perth: University of Western Australia.

Beck, K., 1999. Embracing change with Extreme Programming. *IEEE Computer*, October.

Beck, K., 2004. *Extreme Programming explained.* Boston: Addison-Wesley.

Bentley, J. E., 2005. *Software Testing Fundamentals – Concepts, Roles, and Terminology.* Philadelphia, SAS, pp. 1-12.

Boehm, B., 1986. A Spiral Model of Software Development and Enhancement. *ACM SIGSOFT Software Engineering Notes*, pp. 61-67.

Boehm, B. & Turner, R., 2003. Using Risk to Balance Agile and Plan-Driven Methods. *IEEE Computer*, 36(6), pp. 57-66.

Brooks, F., 1987. No Silver Bullet: Essence and Accidents of Software Engineering. *Computer*, 20(4), pp. 10-19.

Callaghan, K., 1996. The Correlation Coefficient. Στο: L. Kime & J. Clark, επιμ. *Explorations in College Algebra: Discovering Databased Application.* New York: John Wiley, pp. 553-557.

Cockburn, A., 1998. *Surviving Object-Oriented Projects: A Manager's Guide.* 1rst Edition επιμ. Boston: Addison Wesley Longman.

Cockburn, A., 2007. *Agile Software Development: the Cooperative Game.* First Edition επιμ. Boston: Addison-Wesley.

Copper, D. R. & Schindler, P. S., 2013. *Business Research Methods.* 12nd Edition επιμ. New York: McGraw-Hill Education.

Creswell, J. W., 1994. *Research Design: Qualitative and Quantitative Approaches.* London: SAGE.

Deemer , P., Benefield, G., Larman , C. & Vodde, B., 2012. *The Scrum Primer (Version 2.0),* s.l.: InfoQ Enterprize Software Development Series.

DSDM Consortium, 2012. *The DSDM Agile Project Framework for Scrum,* USA: DSDM.

Eldon, L., 1990. Software testing in system development process: A life cycle perspective. *Journal of Systems Management*, August, 41(8), pp. 23-31.

Fowler, M., 2005. *The New Methodology.* [Ηλεκτρονικό] Available at: https://www.martinfowler.com/articles/newMethodology.html [Πρόσβαση 15 Σεπτέμβριος 2017].

Fowler, M. & Highsmith, J., 2001. The Agile Manifesto. *Software Development*, 9(8), pp. 28-32.

French Scrum User Group, 2009. *A National Survey on Agile Methods in France,* Paris: French Scrum User Group.

Garousi, V. & Zhi, J., 2012. A survey of software testing practices in Canada. *The Journal of Systems and Software*, Τόμος 86, pp. 1354-1376.

Given, L. M., 2008. *The Sage Encyclopedia of Qualitative Research Methods.* Los Angeles, California: Sage Publications.

Gornik, D., 2001. *IBM Rational Unified Process: Best Practices for Software Development Teams,* New York: IBM Corporation Software Group.

Guimarães, L. R. & Vilela, P. R. S., 2005. *Comparing Software Development models using CDM.* Newark NJ, ACM, pp. 339-347.

Hundermark, P., 2009. *Measuring for Results,* Cape Town: ScrumSense.

International Organization for Standardization, 1999. *ISO Standard 9126: Information Technology - Software product quality, parts 1,* Geneve: International Organization for Standarization.

Johnson, M., 2003. *Agile methodologies: Survey results.* Victoria, Australia: Shine Technologies.

Kane , M. & Schwaber, K., 2002. *Scrum with XP.* 1η Έκδοση επιμ. New Jersey: Prentice Hall.

Kitchenham, B. & Pfleeger, S., 1996. Software Quality: The Elusive Target. *IEEE Software*, pp. 12-21.

Laanti, M., Salo, O. & Abrahamsson, P., 2010. Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation. *Information and Software Technology*, Τόμος 53, pp. 276-290.

Mahnič, V. & Drnovšček, S., 2005. *Agile Software Project Management with Scrum.* Manchester, EUNIS.

McCall, J. A., Richards, P. K. & Walters , G. F., 1977. *Factors in Software Quality: Concept and Definition of Software Quality,* New York: General Electric Company.

Miller, R. W. & Collins, C. T., 2001. *Acceptance Testing.* Raleigh, XPUniverse.

Misic, V., 2006. Perceptions of Extreme Programming: An Exploratory Study. *AACM SIGSOFT Software Engineering Notes*, 6th March, 31(2).

Munassar, N. & Govardhan, A., 2010. A Comparison Between Five Models Of Software Engineering. *IJCSI International Journal of Computer Science Issues*, 7(5), pp. 94-101.

Munassar, N. & Govardhan, A., 2011. Comparison between Traditional Approach and Object-Oriented Approach in Software Engineering Development. *International Journal of Advanced Computer Science and Applications*, 2(6), pp. 70-76.

Nerur, S., Mahapatra, R. & Mangalaraj, G., 2005. Challenges of Migrating to Agile Methodologies. *Communications of the ACM*, May, 48(5), pp. 72-78.

Nikiforova, O., Nikulsins, V. & Sukovskis, U., 2009. Integration of MDA framework into the model of traditional software development. Στο: H. M. Haav & A. Kalja, επιμ. *Databases and Information Systems V : Selected Papers from the Eighth International Baltic Conference, DB&IS. Frontiers in Artificial Intelligence and Applications.* Tallinn: IOS Press, pp. 229-242.

O'Docherty, M., 2005. *Object-Oriented Analysis & design – understanding system development with UML 2.0.* First Edition επιμ. New Jersey: John Wiley.

Owens, D. & Khazanchi, D., 2009. Software Quality Assurance. Στο: T. T. Kidd, επιμ. *Handbook of Research on Technology Project Management, Planning and Operations.* s.l.:Hershey, PA: Information Science Reference (an imprint of IGI Global), pp. 242-260.

Palmer, S. R. & Felsing, J. M., 2002. *A Practical Guide to Feature-Driven Development.* First Edition επιμ. New York: Prentice Hall.

Petersen, K., Wohlin, C. & Baca, D., 2009. *The Waterfall Model in Large-Scale Development.* Oulu, Springer LNBI.

Pfleeger, S. L. & Kitchenham, B., 2002. Principles of Survey Research (parts 4, 5). *Software Engineering Notes*, 27(5).

Pressman, R. S., 2005. *Software Engineering: A Practitioner's Approach.* 6th Edition επιμ. New York: McGraw-Hill.

Punkka, T., 2005. Agile Methods and Firmware Development. *HUT*, pp. 1-21.

Rajasekar, S., Philominathan, P. & Chinnathambi, V., 2006. *Research methodology,* Tiruchirapalli: Bharathidasan University.

Rodríguez, P., Markkula, J., Oivo, M. & Turula, K., 2011. *Survey on agile and lean usage in Finnish software industry.* New York, USA, ACM-IEEE.

Schindler, C., 2008. Agile Software Development Methods and Practices in Austrian IT-Industry : Results of an Empirical Study. *Technology*, pp. 321-326.

Schwaber, K., 1995. *Scrum Development Process.* Ostin, TX, Springer-Verlag.

Sellers, R., 1998. Qualitative versus quantitative research - choosing the right approach. *The NonProfit Times,* March 15.

Shao, D., Khurshid, S. & Perry, D. E., 2007. *Case for White-box Testing Using Declarative Specifications.* Washington, IEEE, p. 137.

Sommerville, I., 2004. *Software Engineering.* 7th Edition επιμ. Boston: Addison Wesley.

Stapleton, J., 1997. *DSDM, Dynamic Systems Development Method: The Method in Practice.* 1η Έκδοση επιμ. Bosto: Addison-Wesley.

Stelzer, D., Mellis, W. & Herzwurm, G., 1996. *Software Process Improvement via IS0 9000? Results of two surveys among European software houses.* Hawaii, Interantional Society for the Systems Sciences.

Sukamolson, S., 2007. *Fundamentals of quantitative research.* [Ηλεκτρονικό] Available at: http://www.culi.chula.ac.th/e-journal/bod/suphat%20sukamolson.pdf [Πρόσβαση 27 Οκτωβρίου 2017].

Sutherland, J., 2010. *Scrum Handbook.* Somerville: Scrum Training Institute Press.

Taylor-Powell, E. & Hermann, C., 2002. *Collecting Evaluation Data: Surveys.* [Ηλεκτρονικό] Available at: http://learningstore.uwex.edu/assets/pdfs/G3658-10.PDF [Πρόσβαση 28 Οκτώβριος 2017].

Wells, D., 2013. *Extreme Programming: A gentle introduction.* [Ηλεκτρονικό] Available at: http://www.extremeprogramming.org/ [Πρόσβαση 2017 Δεκεμβρίου 15].

Yli-Huumo, J., 2013. *Software development methods and quality assurance: Special focus on South Korea,* Lappeenranta, Finland: Lappeenranta University of Technology.

Yu, B., WooiKhong, L., Wai, W. T. & Soo, F. T., 2012. *Software Development Life Cycle AGILE vs. Traditional Approaches.* Singapore, IACSIT Press.

Βεσκούκης, Β., 2015. *Στοιχεία Τεχνολογίας Λογισμικού.* 1η Έκδοση επιμ. Αθήνα: Σύνδεσμος Ελληνικων Ακαδημαϊκων Βιβλιοθηκών.