Πανεπιστήμιο Θεσσαλίας

Σχολή Διοίκησης & Οικονομίας

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Διοίκηση Έργων & Προγραμμάτων»

Διπλωματική Εργασία

Διερεύνηση του βαθμού χρήσης «ευέλικτων» μεθόδων management από στελέχη εταιρειών πληροφορικής και επαγγελματίες πληροφορικής στην Ελλάδα

Ονοματεπώνυμο Συγγραφέα: **Αθανάσιος Γκουρομπίνος**

Επιβλέπων Καθηγητής: **Βασίλειος Γερογιάννης**

Λάρισα 2021

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πανεπιστήμιο Θεσσαλίας

Σχολή Διοίκησης & Οικονομίας

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Διοίκηση Έργων & Προγραμμάτων»

Διπλωματική Εργασία

Διερεύνηση του βαθμού χρήσης «ευέλικτων» μεθόδων management από στελέχη εταιρειών πληροφορικής και επαγγελματίες πληροφορικής στην Ελλάδα

Ονοματεπώνυμο Συγγραφέα: **Αθανάσιος Γκουρομπίνος**

Επιβλέπων Καθηγητής: **Βασίλειος Γερογιάννης**

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 10η Μαΐου 2021

(Υπογραφή)	(Υπογραφή)	(Υπογραφή)
…….	…….	…….
Βασίλειος Γερογιάννης	Λεωνίδας Ανθόπουλος	Παναγιώτης Φιτσιλής
Καθηγητής Πανεπιστημίου Θεσσαλίας	Καθηγητής Πανεπιστημίου Θεσσαλίας	Καθηγητής Πανεπιστημίου Θεσσαλίας

Λάρισα 2021

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΑΘΑΝΑΣΙΟΣ ΓΚΟΥΡΟΜΠΙΝΟΣ

Διπλωματούχος Μηχανικός Έργων Υποδομής, ΑΤΕΙ Θεσσαλονίκης

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ

Αυτή η διπλωματική εργασία υποβάλλεται από τον κ. Αθανάσιο Γκουρομπίνο ως μερική εκπλήρωση των απαιτήσεων του Μεταπτυχιακού Προγράμματος Σπουδών στη «Διοίκηση Έργων & Προγραμμάτων» του Πανεπιστημίου Θεσσαλίας.
Υπεύθυνα δηλώνεται ότι η συγκεκριμένη διπλωματική εργασία έχει συγγραφεί από τον κ. Αθανάσιο Γκουρομπίνο και δεν έχει υποβληθεί ούτε έχει αξιολογηθεί στο πλαίσιο άλλου μεταπτυχιακού ή προπτυχιακού τίτλου σπουδών, στην Ελλάδα ή στο εξωτερικό.

ΟΝΟΜΑΤΕΠΩΝΥΜΟ:

Αθανάσιος Γκουρομπίνος

ΥΠΟΓΡΑΦΗ:

Η σελίδα αυτή είναι σκόπιμα λευκή.

Στον αδερφό μου Ηλία, που μας άφησε νωρίς.

Η σελίδα αυτή είναι σκόπιμα λευκή.

Ευχαριστίες

Στον ανιψιούς μου, Θωμά και Δημήτρη, που είναι η πηγή της ζωής μου.

Στον επιβλέποντα καθηγητή μου, κύριο Βασίλειο Γερογιάννη, για την άψογη συνεργασία και όλους τους ορίζοντες που ανοίχτηκαν από τη διδασκαλία και τη συνεργασία μαζί του.

Στην οικογένεια και τους φίλους μου, που είναι πάντα δίπλα μου.

Στην κυρία Αθανασία Νταλαβίκα για την επιμέλεια του κειμένου.

Η σελίδα αυτή είναι σκόπιμα λευκή.

Περίληψη

Από το 2001 και μετά, όπου και ορίστηκε το Agile Μανιφέστο, υπήρξαν πολλοί οι υποστηρικτές που υιοθέτησαν τη φιλοσοφία του από την πρώτη στιγμή, βελτιώνοντας την καθημερινότητά τους σχεδιάζοντας - κατασκευάζοντας λογισμικό. Τα πεδία εφαρμογής και τα παραδείγματα πολλά, όπως Πανεπιστήμια, Δημόσιοι Φορείς, κλπ., αλλά κυρίως οι επαγγελματίες της Agile. Όλοι αυτοί με την μελέτη και την εργασία τους συνέβαλαν ώστε να καθορίσουν τους παράγοντες επιτυχίας και τους περιορισμούς των έργων.

Στην παρούσα εργασία θεωρούμε σκόπιμο να παρουσιάσουμε τις βασικές έννοιες προκειμένου κάποιος που δεν έχει γνώση πάνω στις βασικές έννοιες του έργου, της διοίκησης έργων, του Λογισμικού & της Τεχνολογίας Λογισμικού, του Κύκλου Ζωής των Έργων και των Μοντέλων του (Παραδοσιακά & Ευέλικτα), και ειδικότερα στην διαχείριση έργων πληροφορικής, ώστε να είναι σε θέση να κατανοήσει όλες αυτές τις έννοιες που θα αποτελέσουν το υπόβαθρο της εργασίας.

Συγκεκριμένα, θα επιχειρηθεί να γίνει μια ποιοτική έρευνα, στην οποία μέσω ανάλυσης θα καταγραφεί αν γίνεται εφαρμογή των «ευέλικτων» μεθοδολογιών από στελέχη εταιρειών πληροφορικής και επαγγελματίες πληροφορικής στην Ελλάδα. Παράλληλα, θα εντοπίσουμε εκείνους τους παράγοντες που οδηγούν στην επιτυχία ολοκλήρωσης των έργων μέσω της διεθνούς βιβλιογραφίας και συγκεκριμένα μέσω 14 μελετών περίπτωσης εφαρμογής των «ευέλικτων» μεθοδολογιών, ενώ στο τέλος θα δοθεί το έναυσμα για περαιτέρω διενέργεια νέων μελετών περίπτωσης στην ελληνική αγορά.

Λέξεις κλειδιά: Agile Management, παράγοντες επιτυχίας, περιορισμοί, ελληνικές εταιρείες λογισμικού, εφαρμογή

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

Since 2001, when the Agile Manifesto was established, there were many supporters who adopted its philosophy immediately, improving their daily lives by designing - constructing software. The fields of application are numerous and include Universities, Public Bodies, etc., however the main users are "Agile" professionals. All of them, through their study and work, contributed towards determining success factors and constrains for their respective projects.

Within the present dissertation it is attempted to present the basic concepts such as ‘project’, ‘project management’, ‘Software & Software Technology’, ‘Project Lifecycle and Models (Traditional & Agile)’, and especially ‘IT project management’, in a manner that aims to familiarize and educate readers without any pre-existing knowledge of said concepts.

Specifically, a qualitative study will be attempted, in which, via means of analysis, the level of application of "agile" methodologies by Greek software companies will be recorded and reported on. We aim to identify those factors that lead to the successful completion of projects using international literature and specifically 14 case studies of application of agile methodologies. Finally, this may provide the impetus for further conduct of case studies within the Greek market.

Key Words: Agile Management, success factors, constraints, Greek Software Companies, application

Η σελίδα αυτή είναι σκόπιμα λευκή.

Περιεχόμενα

Η σελίδα αυτή είναι σκόπιμα λευκή.

Λίστα Εικόνων

Η σελίδα αυτή είναι σκόπιμα λευκή.

Λίστα Πινάκων

Η σελίδα αυτή είναι σκόπιμα λευκή.

Κεφάλαιο 1: Εισαγωγή

Στο πέρασμα των αιώνων, από τότε που ο άνθρωπος ένιωσε την ανάγκη να προστατεύσει τον εαυτό του και τους γύρω του, άρχισε να κατασκευάζει πρόχειρα έργα για την προστασία του. Ξεκίνησε με την κατασκευή πρόχειρων κατοικιών, ενώ όταν άρχισε να οργανώνεται σε κοινωνίες προσπάθησε να ανοίξει δρόμους επικοινωνίας κατασκευάζοντας μεγαλύτερα έργα.

Για να πραγματοποιηθούν όλα αυτά τα κατασκευαστικά επιτεύγματα, σίγουρα θα υπήρχαν φωτεινά μυαλά, με λαμπρές ιδέες και μεθόδους. Μπορεί όλα αυτά να μην έφτασαν στις μέρες μας (μέθοδοι, ιδέες, κλπ.), τα μεγάλα, όμως, αυτά έργα μαρτυρούν το σημαντικό ρόλο των μεθόδων διοίκησης αλλά και την ικανότητα των ηγετών τους να τα κατασκευάσουν.

Τα πρώτα εμφανή δείγματα διοίκησης έργων καταγράφονται στη Γαλλία τον Μεσαίωνα. Εκεί οι πρωτομάστορες ηγούνταν των εργασιών. Επίσημα πια μιλάμε για διοίκηση έργων στις αρχές του 1900 με τον Gantt, και λίγο αργότερα στα μεγάλα σύγχρονα έργα φτάνοντας έως τις μέρες μας. Η επιστήμη της Διοίκησης έργων είναι μια επιστήμη η οποία εξελίσσεται διαρκώς, ακόμα και στις μέρες μας, μιας και άρχισε να διδάσκεται επίσημα στα Πανεπιστήμια το 2000!

Και ενώ η εξέλιξη του ανθρώπου άρχισε να κορυφώνεται, παράλληλα άρχισαν να αναπτύσσονται τεχνολογίες που θα έκαναν καλύτερη την καθημερινότητα του. Ο Τιούρινγκ. την δεκαετία του 30 αρχίζει να εισαγάγει την επιστήμη της πληροφορικής στη ζωή μας, ανοίγοντας το δρόμο σε μια νέα μορφή καθημερινότητας. Η είσοδος της πληροφορικής, των λογισμικών και των νέων τεχνολογιών στις ζωές μας δεν έχει παρά να μας δώσει θετικά πράγματα. Παρουσίασε ταχύτατη εξέλιξη, εξαιτίας της ραγδαίας εξέλιξης της επιστήμης της ηλεκτρονικής αλλά και της ζήτησης της από τον πληθυσμό.

Τέλος, η είσοδος του διαδικτύου συμπλήρωσε αυτήν την εξέλιξη, οδηγώντας σε ακόμη μεγαλύτερα τεχνολογικά επιτεύγματα. Στην εποχή της 4ης Βιομηχανικής Επανάστασης, την οποία και διανύουμε, διαπιστώνουμε ότι βρισκόμαστε μπροστά σε οικονομικές και κοινωνικές εξελίξεις, που μόνο σκοπό έχουν τη βελτίωση του προσδόκιμου ζωής των έργων και των ανθρώπων.

1.1. Το πρόβλημα

Τα πρώτα χρόνια ανάπτυξης της επιστήμης της ηλεκτρονικής και ειδικότερα της πληροφορικής και του λογισμικού, δεν υπήρχε καμία δομημένη διαδικασία ανάπτυξης των τελικών προϊόντων.

Ειδικότερα τη δεκαετία του 70, με τη ραγδαία εξέλιξη της ηλεκτρονικής, και συγκεκριμένα, των έργων ανάπτυξης λογισμικού, τα έργα αρχίζουν να γίνονται όλο και πιο πολύπλοκα,

εμπλέκοντας μεγάλο αριθμό εξειδικευμένου προσωπικού, κάνοντας ορατή την ανάγκη ανάπτυξης μεθοδολογίας για τη διαχείριση των έργων.

Από τις πρώτες δομημένες μεθοδολογίες, που περιγράφουν με συγκεκριμένα βήματα το σύνολο της ανάπτυξης ενός έργου ανάπτυξης λογισμικού, είναι αυτό του **Σειριακού Μοντέλου Καταρράκτη** (Waterfall Model), όπως δόθηκε από τον Royce το 1970. Η μεθοδολογία αυτή ήδη εφαρμοζόταν από τον ίδιο και την ομάδα του. Ακολούθησαν αρκετά χρόνια αργότερα και άλλες μεθοδολογίες ανάπτυξης έργων. Μερικές από αυτές ήταν: το **Σπειροειδές Μοντέλο** (Spiral Model) του Bohem το 1988, το μοντέλο **Rapid – Prototyping** του Connell του 1989, το μοντέλο **Incremental (staged) Delivery** του Wong του 1984, το **Evolutionary Delivery** μοντέλο του Gilb του 1988, η **μέθοδος Rapid Application Development (RAD)** του Μάρτιν το 1991, η μεθοδολογία **Rational Unified Process (RUP)** του Kruchten του 1996 και άλλες.

Από όλες τις παραπάνω μεθοδολογίες φαίνεται ξεκάθαρα ότι έχουν άριστη εφαρμογή σε έργα τα οποία ακολουθούν μια σειρά ανάπτυξης, σε αντίθεση με έργα πληροφορικής και λογισμικού, οπού γενικότερα παρουσιάζεται μια σειρά από προβλήματα και δυσκολίες, που αφορούν κυρίως υπερβάσεις σε χρονοδιαγράμματα, προϋπολογισμό, ποιότητα και απαιτήσεις του πελάτη.

Δουλεύοντας και βλέποντας όλα τα προαναφερθέντα προβλήματα, μια ομάδα 17 επιστημόνων μαζεύτηκε το Φεβρουάριο του 2001 στο Lodge, στο χιονοδρομικό κέντρο Snowbird στα βουνά Wasatch της Γιούτα (ΗΠΑ), για φαγητό και σκι, αναζητώντας κοινό έδαφος για την ανάπτυξη λογισμικού. Από αυτή τη συνάντηση γεννήθηκε το «Ευέλικτο Μανιφέστο» (Agile Management). Σύμφωνα με τις αξίες και τις αρχές του, υπόσχεται πια μεγαλύτερη προσαρμοστικότητα και ανταπόκριση στις αλλαγές, παραγωγικότερες πρακτικές και λιγότερη γραφειοκρατία. Οι «ευέλικτες» μεθοδολογίες προτείνουν μια «επαναστατική» προσέγγιση σε σχέση με τη συνηθισμένη πρακτική, όπου ο ανθρώπινος παράγοντας (τόσο ο χρήστης - πελάτης όσο και τα μέλη της ομάδας ανάπτυξης) τοποθετείται σε πρώτο επίπεδο.

Η χρήση των «ευέλικτων» μεθοδολογιών είχε σταδιακά ευρεία εφαρμογή παγκόσμια. Δηλαδή, εφαρμόστηκε σε πολλούς τομείς της καθημερινότητας, όπως: πανεπιστήμια, δημόσιες υπηρεσίες, εταιρείες κα. Και κυρίως σε έργα πληροφορικής και λογισμικού. Μελέτες που πραγματοποιήθηκαν έδειξαν τους παράγοντες επιτυχίας των μεθοδολογιών - πρακτικών αυτών παγκόσμια.

Λαμβάνοντας υπόψη όλα τα παραπάνω, και το γεγονός ότι στην ελληνική πραγματικότητα την τελευταία 20ετία υλοποιήθηκαν πολλά σημαντικά έργα και πραγματοποιήθηκαν, επίσης, μεγάλες εφαρμογές για την διευκόλυνση της καθημερινότητας, η παρούσα μεταπτυχιακή

διατριβή έχει ως σκοπό να διερευνήσει κατά πόσο ακολουθήθηκε από τις εταιρείες κατασκευής λογισμικού και από το ελληνικό δημόσιο η εφαρμογή των «Ευέλικτων» μεθοδολογίων.

1.2. Αντικείμενο διπλωματικής

Βασικός στόχος της παρούσας διπλωματικής είναι αρχικά να εντοπίσει και να διερευνήσει κατά πόσο οι εταιρείες και τα στελέχη εταιρειών στην Ελλάδα εφαρμόζουν – δουλεύουν πάνω στις «ευέλικτες» μεθοδολογίες. Επίσης, διερευνάται ποιοι παράγοντες θεωρούνται ως επιτυχείς και ποιοι ως εμπόδια για την ολοκλήρωση των έργων λογισμικού, εφαρμόζοντας «ευελιξία». Αρχικά, θα θέσουμε τους υποθετικούς παράγοντες επιτυχίας των «ευέλικτων» πρακτικών, όπως αυτοί αναφέρονται στη διεθνή βιβλιογραφία, και θα τους προσαρμόσουμε στην ελληνική πραγματικότητα. Στη συνέχεια, με βάση όλα τα παραπάνω, θα καταγραφεί όλη η επιστημονική ορολογία, ενώ, παράλληλα, θα δημιουργηθεί ένα ερωτηματολόγιο και θα διανεμηθεί στις ελληνικές εταιρείες κατασκευής λογισμικού. Αφού γίνει η παραπάνω προεργασία, θα συλλεχθεί όλο το υλικό που διανεμήθηκε και θα ακολουθηθεί η επεξεργασία και η ανάλυση των δεδομένων. Σύμφωνα με όλα τα αποτελέσματα, θα εντοπίσουμε αν οι «ευέλικτες» μεθοδολογίες ακολουθούνται από στελέχη εταιρειών πληροφορικής και επαγγελματίες πληροφορικής στην Ελλάδα, και ποιοι είναι εκείνοι οι παράγοντες που βοηθούν στην επιτυχή ολοκλήρωση του έργου.

1.2.1. Συνεισφορά

Επομένως, η συνεισφορά της παρούσας διπλωματικής έρευνας αναλύεται στα παρακάτω στάδια:

1.Αρχικά, θα ορίσουμε τις βασικές έννοιες προκειμένου κάποιος που δεν έχει γνώση πάνω στις βασικές έννοιες του έργου, της διοίκησης έργων, του Λογισμικού & της Τεχνολογίας Λογισμικού, του Κύκλου Ζωής των Έργων και των Μοντέλων του (Παραδοσιακά), και ειδικότερα στην διαχείριση έργων πληροφορικής, να είναι σε θέση να κατανοήσει όλες αυτές τις έννοιες που θα αποτελέσουν το υπόβαθρο της εργασίας.

2.Στη συνέχεια, θα επικεντρωθούμε σε όλους τους κανόνες, τους τύπους και τα μοντέλα των «ευέλικτων» μεθοδολογιών, έτσι ώστε να είναι ξεκάθαρο το νέο αυτό πεδίο της επιστήμης της Διοίκησης.

3.Θα καθορίσουμε τους παράγοντες επιτυχίας ενός λογισμικού, αλλά και τους περιορισμούς στην άσκηση «ευελιξία», σύμφωνα με τη διεθνή βιβλιογραφία, ενώ παράλληλα θα τους έχουμε προσαρμόσει στα ελληνικά δεδομένα. Παράλληλα, θα δημιουργήσουμε έναν πίνακα με τους παράγοντες επιτυχίας και τους περιορισμούς.

4.Θα διατεθεί ένα ερωτηματολόγιο, το οποίο θα λειτουργεί ως εργαλείο - πλαίσιο καθορισμού των παραγόντων επιτυχίας, αλλά και αναγνώρισης των περιορισμών, εφαρμόζοντας «ευέλικτες» μεθόδους στην ελληνική αγορά.

5.Θα πραγματοποιηθεί ανάλυση των αποτελεσμάτων, με βάση τα ποιοτικά χαρακτηριστικά που οριοθετήθηκαν παραπάνω, για να ελέγξουμε αν οι «ευέλικτες» μεθοδολογίες εφαρμόζονται στην ελληνική αγορά και ποιοι από τους παράγοντες οδηγούν στην επιτυχία ολοκλήρωσης των έργων.

6.Θα καταγραφεί η υπάρχουσα κατάσταση στην ελληνική αγορά, έτσι ώστε η παρούσα διπλωματική διατριβή να αποτελεί εργαλείο άμεσης εφαρμογής και αποκόμισης γνώσης των «ευέλικτων» μεθόδων από τον κάθε αναγνώστη, υποψήφιο για την εφαρμογή των μεθοδολογιών.

1.2.2. Οργάνωση κειμένου

Συνολικά η παρούσα διπλωματική εργασία αποτελείται από τέσσερα διακριτά κεφάλαια, συμπεριλαμβανομένου και του πρώτου εισαγωγικού κεφαλαίου, όπου παρατίθεται μια γενική εισαγωγή, το πρόβλημα, το αντικείμενο και, τέλος, η συνεισφορά.

Στο δεύτερο κεφάλαιο παρουσιάζεται μια βιβλιογραφική επισκόπηση, πάνω στην οποία στηρίζεται το θεωρητικό υπόβαθρο της εργασίας. Δηλαδή, θα ορισθούν το εννοιολογικό πλαίσιο και οι θεωρίες, θα τεθούν τα ερευνητικά ερωτήματα και τι προκύπτει από τη βιβλιογραφία, ενώ στο τέλος θα βγουν τα βασικά συμπεράσματα.

Στο τρίτο κεφάλαιο θα αναλυθεί η μεθοδολογία της εργασίας και περιγραφή των μεθόδων έρευνας ανταποκρινόμενη στα ερευνητικά ερωτήματα, με αναλυτικές περιγραφές των μεθόδων ξεχωριστά. Επίσης, θα γίνει πλήρης ανάλυση της έρευνας και της διαδικασίας που ακολουθήθηκε.

Στο τέταρτο κεφάλαιο, μετά τη συλλογή των στοιχείων της έρευνας, θα ακολουθηθεί η ανάλυση των αποτελεσμάτων. Αυτό θα γίνει με χρήση κατάλληλών τεχνικών σε συνδυασμό με τα ερευνητικά ερωτήματα. Η ανάλυση θα είναι διεξοδική, που θα βασίζεται στην περιγραφή μέσω της επαγωγικής ανάλυσης. Θα ακολουθήσει η συζήτηση των αποτελεσμάτων, σχολιάζοντας κριτικά κα ερμηνεύοντάς τα σε συνάρτηση με το θεωρητικό πλαίσιο που αναπτύχθηκε στα προηγούμενα κεφάλαια και τα πορίσματα άλλων ερευνών. Κλείνοντας, η παρούσα διπλωματική ολοκληρώνεται με προτάσεις χρήσης της έρευνας στο ευρύτερο επιστημονικό πεδίο. Δηλαδή, πόσο μπορεί να αναπτυχθεί η παρούσα έρευνα μελλοντικά και να χρησιμοποιηθεί γενικά ως εργαλείο σε ακαδημαϊκό επίπεδο λειτουργώντας ως εργαλείο εφαρμογής των μεθόδων από αυτούς που εφαρμόζουν τις μεθοδολογίες.

Η σελίδα αυτή είναι σκόπιμα λευκή.

Κεφάλαιο 2: Βιβλιογραφική Επισκόπηση

Στο κεφάλαιο που ακολουθεί θεωρούμε σκόπιμο να παρουσιάσουμε τις βασικές έννοιες, προκειμένου ο αναγνώστης που δεν έχει γνώση σχετικά με αυτές και, ειδικότερα, με την διαχείριση έργων πληροφορικής, να είναι σε θέση να τις κατανοήσει, αφού αυτές θα αποτελέσουν το υπόβαθρο της εργασίας. Συγκεκριμένα, στο κεφάλαιο αυτό θα επιχειρηθεί η καταγραφή των όρων του Έργου, της Διοίκησης Έργων, του Λογισμικού & της Τεχνολογίας Λογισμικού, των Απαιτήσεων Λογισμικού, του Κύκλου Ζωής των Έργων & των Μοντέλων του (Παραδοσιακά & Ευέλικτα).

2.1. Έργο

Όπως αναφέρθηκε και στην εισαγωγή της έρευνας, ο άνθρωπος από τη στιγμή που ένιωσε την ανάγκη - απειλή άρχισε να κατασκευάζει έργα, αρχικά για την προστασία του, και αργότερα, όταν άρχισε να οργανώνεται σε κοινωνίες, μέσω των έργων αυτών ερχόταν σε επικοινωνία με τις υπόλοιπες κοινωνίες. Τι είναι άραγε ένα έργο και πώς άραγε δημιουργείται;

Κάθε οργανισμός επιτελεί μια βασική λειτουργία: **μετασχηματίζει πόρους σε ενέργειες**. Την ίδια διαδικασία την πράττουν όλοι οι οργανισμοί. Από το ανθρώπινο σώμα μέχρι πχ τις επιχειρήσεις, τους φορείς του δημοσίου, τους μη κερδοσκοπικούς – κυβερνητικούς οργανισμούς, κλπ.

Για να πραγματοποιηθούν όλα τα κατασκευαστικά επιτεύγματα, από την γέννηση της ανθρώπινης ζωής, σίγουρα θα υπήρχαν ξεχωριστοί άνθρωποι, οι οποίοι χωρίς γνώση βοήθησαν στην δημιουργία και την αλματώδη ανάπτυξη, αργότερα, του Project Management, αν και πολλές πληροφορίες δεν καταγράφηκαν από την αρχή. Τα μεγάλα έργα του παρελθόντος, όπως ο πύργος της Βαβέλ, οι πυραμίδες της Αιγύπτου, το Σινικό Τοίχος, κα., έργα πριν τους κλασσικούς ιστορικούς χρόνους, μας δίδαξαν ότι οι άνθρωποι με αυτή τους την προσπάθεια επιδίωκαν να πετύχουν κάτι. Τα πρώτα δείγματα καταγραφής διοίκησης έργων βρίσκουμε στη Γαλλία κατά το Μεσαίωνα, με τους Αρχιμάστορες. Η πρώτη συστηματική προσπάθεια επίτευξης καλύτερου αποτελέσματος μέσω της παραγωγικής διαδικασίας έγινε το 19ο αιώνα σε ένα εργοστάσιο χυτοσιδήρου στις ΗΠΑ. Υπεύθυνος της προσπάθειας αυτής ήταν ο Αρχιμηχανικός του εργοστασίου, F. W. Taylor (1856 – 1915), ο οποίος σήμερα θεωρείται ο πατέρας της Διοίκησης – Διαχείρισης.

Από την εμφάνιση του Taylor και έπειτα, η διοίκηση έργων αρχίζει να βρίσκει ευρεία αποδοχή και κυρίως στον τομέα της βιομηχανίας, των τεχνικών έργων, ενώ ιδιαίτερη έμφαση δόθηκε στα έργα ανάπτυξης στρατιωτικών σκοπών, όπως στο πρόγραμμα Polaris, στο διαστημικό πρόγραμμα της Nasa, Apollo, στην ανάπτυξη των έξυπνων βομβών κα. Τα τελευταία πενήντα

χρόνια, η διοίκηση των έργων αντιμετωπίστηκε ως ξεχωριστό πεδίο επιστημονικής και επαγγελματικής μελέτης – ενασχόλησης.

Τι είναι έργο λοιπόν;

«***Μια προσωρινή προσπάθεια που αναλαμβάνεται για να δημιουργηθεί ένα μοναδικό προϊόν ή υπηρεσία***» σύμφωνα με το PMBOK, Project Management Institute.

«***Μια προσπάθεια στην οποία ανθρώπινοι, υλικοί και χρηματοοικονομικοί πόροι οργανώνονται με ένα νέο τρόπο προκειμένου να χρησιμοποιηθούν στην εκτέλεση εργασιών με συγκεκριμένες προδιαγραφές, σε χρονικούς και οικονομικούς περιορισμούς, έτσι ώστε να επιτευχθεί κάποιο ωφέλιμο αποτέλεσμα που καθορίζεται από ποσοτικά και ποιοτικά κριτήρια***» κατά Turner J.R., το 1990

«***Έργο είναι μία προσωρινή διοικητική δομή, στην οποία εκχωρούνται πόροι για να αναληφθεί μία μοναδική, νέα και παροδική προσπάθεια για τη διαχείριση της εγγενούς αβεβαιότητας και ανάγκης συντονισμού και ολοκλήρωσης, ώστε να προκύψουν ωφέλιμες αλλαγές***», όπως είπε ο Turner J.R., το 2002.

Συγκρίνοντας τα παραπάνω, μπορούμε να πούμε ότι: **έργο** είναι ένα προσωρινό εγχείρημα που στοχεύει στη δημιουργία ενός μοναδικού προϊόντος ή υπηρεσίας. Προσωρινό, γιατί σημαίνει ότι κάθε έργο έχει ένα προκαθορισμένο τέλος, ενώ με τη μοναδικότητα εννοούμε ότι το προϊόν ή η υπηρεσία διαφέρει κατά διακριτό τρόπο από όλα τα παρόμοια προϊόντα ή τις υπηρεσίες.

Εναλλακτικά θα μπορούσαμε να το αναλύσουμε και ως εξής: έργο είναι ένα εγχείρημα, κατά το οποίο όλοι οι πόροι (ανθρώπινοι – υλικοί – οικονομικοί) οργανώνονται με ένα πρωτότυπο τρόπο, με στόχο την ανάληψη συγκεκριμένου αντικειμένου εργασιών που έχουν τις δικές τους προδιαγραφές και υπόκεινται σε περιορισμούς (κοστολογικούς και χρονικούς), έτσι ώστε να παραχθεί μια ωφέλιμη αλλαγή, η οποία θα έχει καθοριστεί από κριτήρια ποσοτικά και ποιοτικά.

Αναλυτικά, ένα έργο θεωρείται, επίσης, μια σειρά από αλληλεξαρτώμενες δραστηριότητες με συγκεκριμένα χαρακτηριστικά, όπως:

- Συγκεκριμένες ημερομηνίες έναρξης και περάτωσης
- Καλώς ορισμένους στόχους
- Παραγωγή συγκεκριμένου αποτελέσματος
- Μη επανάληψη της ίδιας σειράς δραστηριοτήτων
- Ανάλωση χρημάτων, χρόνου, ανθρώπινων και υλικών πόρων

Ένα από τα επιτυχημένα μοντέλα έργων είναι αυτό του 4-D model που αποτελείται από τα παρακάτω 4 στάδια:

- D1 – Καθορισμός του έργου (Define the project)
- D2 – Σχεδίαση των διαδικασιών του έργου (Design the project process)
- D3 – Εκτέλεση & Παράδοση του έργου (Deliver the project)
- D4 – Λειτουργία & Συντήρηση της διαδικασίας (Develop the process)

Ενώ οι 4 βασικές διαδικασίες ενός έργου είναι οι εξής:

- **Αρχικοποίηση**: εγκατάσταση
- **Προγραμματισμός:** εκτιμήσεις, εξαρτήσεις, περιορισμοί, αναθέσεις
- **Εκτέλεση:** βελτιστοποίηση, αναφορές, επικαιροποίηση
- **Ολοκλήρωση:** αξιολογήσεις

Η διαδικασία ανάπτυξης ενός έργου γίνεται όταν έχουμε την ικανοποίηση της ανάγκης, χρησιμοποιώντας μηχανισμούς, όπως: τους ανθρώπινους πόρους, τη γνώση, την εμπειρία, τα κεφάλαιο, τα εργαλεία – τεχνικές, την τεχνολογία, έχοντας λάβει υπόψιν όλους τους περιορισμούς, όπως: οικονομικούς, νομικούς, δεοντολογικούς, λογικούς, χρονικούς, κα.

Που **διαφέρει** ένα έργο από μια διαδικασία – σύνολο λειτουργιών;

Οι **ομοιότητες,** τόσο στα έργα όσο και στις διαδικασίες, είναι οι εξής:

- να καταναλώνουν πόρους
- περιορίζονται από τον προϋπολογισμό, το χρόνο, τους πόρους κ.λπ.
- προγραμματίζονται, εκτελούνται και ελέγχονται

Ενώ οι **διαφορές** είναι ξεχωριστές για κάθε περίπτωση. Στα έργα οι διαφορές έχουν να κάνουν με τα εξής:

- Είναι μοναδικές
- Είναι προσωρινές

Οι **διαδικασίες** έχουν να κάνουν με τα εξής:

- είναι επαναλαμβανόμενες
- είναι σε εξέλιξη

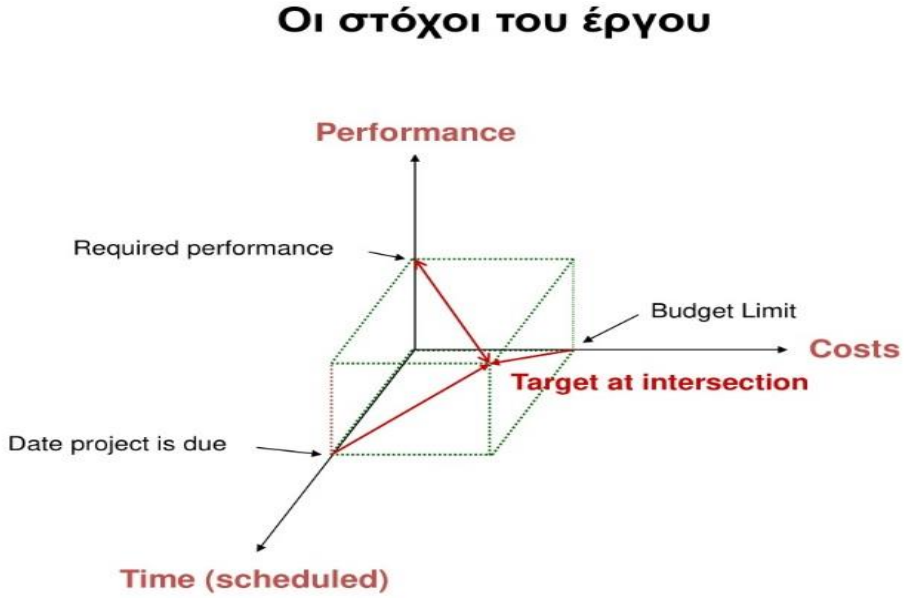
Άρα, η επιτυχία ενός έργου κρίνεται από την κάλυψη των απαιτήσεων - αναγκών του πελάτη, σύμφωνα με το διαθέσιμο χρόνο και με τον συμφωνημένο προϋπολογισμό, ενώ στο τέλος γίνεται αποδεκτό από τον πελάτη.

Η αποτυχία ενός έργου, από την άλλη πλευρά, κρίνεται από τις μη καλυπτόμενες απαιτήσεις – ανάγκες του πελάτη, που προφανώς προέκυψαν από την λανθασμένη καταγραφή τους, έτσι ώστε να μη γίνει ένας σωστός και ρεαλιστικός σχεδιασμός, καταναλώνοντας έξτρα πόρους και εξαντλώντας τους για άλλους σκοπούς.

Οπότε, και σύμφωνα με όλα τα παραπάνω, στόχος ενός έργου είναι η ικανοποίηση του τριγώνου των στόχων που αποτελείται από τους τρεις περιορισμούς: της ποιότητας, του κόστους και του χρόνου (Σχήμα 1). Αναλυτικά έχουμε το εξής (Σχήμα 2):

- Αυξημένο εύρος = αυξημένος χρόνος + αυξημένο κόστος
- Σφιχτός χρόνος = αυξημένο κόστος + μειωμένο εύρος
- Σφιχτός προϋπολογισμός = αυξημένος χρόνος + μειωμένο εύρος.

Εικόνα 1 - Το τρίγωνο των έργων - Πηγή: slideplayer.gr



Εικόνα 2 - Οι στόχοι του έργου - Πηγή: slideplayer.gr

2.2. Διοίκηση Έργων

Η επιστήμη, η οποία ασχολείται με την οργάνωση των έργων, είναι η επιστήμη της **Διοίκησης έργων**. Η «**Διοίκηση Έργων**» προσπαθεί να δώσει απαντήσεις στις ερωτήσεις που αφορούν στο: γιατί, τι, πως, ποιος, πόσο κοστίζει και πότε ένα έργο, από τη στιγμή της σύλληψης μέχρι τη στιγμή της υλοποίησης, καταστροφής, εγκατάλειψης, κλπ. Αντικειμενικοί στόχοι είναι η ισορροπία μεταξύ του χρόνου (δηλαδή της διάρκειας ζωής του έργου), του χρήματος (δηλαδή του χαμηλότερου κόστους) και της ποιότητας (δηλαδή της βελτιστοποίησης της ποιότητας).

Εναλλακτικά θα μπορούσαμε να πούμε ότι Διοίκηση έργων είναι η διαδικασία κατά την οποία εφαρμόζουμε γνώσεις, δεξιότητες, εργαλεία και τεχνικές κατά την εκτέλεση δραστηριοτήτων ενός έργου, με μοναδικό στόχο την κάλυψη των απαιτήσεων και των προσδοκιών όλων των συμμετεχόντων.

Επίσης, θα μπορούσαμε να πούμε ότι είναι μια διαδικασία ενσωμάτωσης όλων όσων πρέπει να πραγματοποιηθούν, καθώς το έργο διανύει τον κύκλο ζωής του ικανοποιώντας όλους τους στόχους.

2.3. Λογισμικό - Τεχνολογία Λογισμικού

Τα τελευταία 20 χρόνια, η ραγδαία εξέλιξη της πληροφορικής και λογισμικού έφερε προϊόντα και υπηρεσίες, οι οποίες απλούστευσαν διαδικασίες στην καθημερινότητά μας. Η υιοθέτηση αυτών των διαδικασιών και προϊόντων ήρθε πρώτα, κυρίως, από τις πιο αναπτυγμένες χώρες του πλανήτη. Και είχε άμεση εφαρμογή σε όλους τους τομείς της ζωής. Η ελληνική πραγματικότητα διαφέρει κατά πολύ. Και μπορεί τα τελευταία χρόνια να έχει αρχίσει να εισέρχεται δειλά - δειλά στις ζωές των Ελλήνων, η Ελλάδα εξακολουθεί να υστερεί κατά πολύ σε απλοποιημένες διαδικασίες και προϊόντα, ενώ μεγαλύτερος βραχνάς αποτελεί η ημιμάθεια της πλειοψηφίας του πληθυσμού και κυρίως του γερασμένου, ενώ μεγαλύτερο κομμάτι αποτελεί και η τεράστια γραφειοκρατία της.

Η νέα δεκαετία που ξεκίνησε σχεδόν ένα χρόνο πριν μας έφερε αντιμέτωπους με ένα μεγάλο κίνδυνο, αυτόν της πανδημίας του Κορονοϊού, ενώ ταυτόχρονα επέβαλε σε όλους τους τομείς της καθημερινότητας τη χρήση της τεχνολογίας. Οι Έλληνες ήρθαν αντιμέτωποι, σε μια βραδιά, με την ευρεία χρήση προγραμμάτων – εντολών ηλεκτρονικού υπολογιστή και άλλων έξυπνων συσκευών και υπηρεσιών, ενώ το κράτος έπρεπε ταχύτατα να μετασχηματιστεί ψηφιακά.

Άρα, όλοι έρχονται αντιμέτωποι με το λογισμικό. Τι είναι άραγε λογισμικό; **Λογισμικό** είναι όλα τα προγράμματα ή οι εντολές του ηλεκτρονικού υπολογιστή και όλων των ηλεκτρονικών συσκευών. **Λογισμικό** είναι όλες οι δομές δεδομένων, με μόνιμη ή προσωρινή αποθήκευση,

που επιτρέπουν στα προγράμματα να διαχειριστούν πληροφορίες, οι οποίες απαιτούν πάντα τεκμηρίωση.

Λογισμικό θα μπορούσαμε να πούμε ότι είναι ένα σύνθετο τεχνικό έργο. Και αυτό γιατί τα τεχνικά έργα έχουν απτή (υλική) υπόσταση και περιγράφονται – γίνονται αντιληπτά. Το λογισμικό έχει ιδεατή υπόσταση. Γίνεται αντιληπτό μόνο με τη χρήση του, ενώ δε περιγράφεται εύκολα, τυποποιημένα και μοναδικά. Η ανάπτυξή του γίνεται με καθορισμό όλων των απαιτήσεων εξαρχής. Γι' αυτό και στα πρώιμα χρόνια ανάπτυξης λογισμικών, αναφέρθηκε από τον Bauer (1968) η «κρίση λογισμικού», η οποία και σήμερα είναι επίκαιρη. Το 1968 επίσης, σε ένα συνέδριο του ΝΑΤΟ τονίστηκαν τα εξής:

- Με την κρίση λογισμικού έχουμε τον χαρακτηρισμό του προβλήματος.
- Με την «Τεχνολογία Λογισμικού» (software engineering) έχουμε μια ιδέα για λύση.

Άρα, για να είναι ένα λογισμικό σωστό θα πρέπει να αποτελείται από τα παρακάτω συστατικά στοιχεία: software components, elements, artifact, που περιλαμβάνουν τα εξής χαρακτηριστικά: Εκτελέσιμο και πηγαίο κώδικα, προδιαγραφές – εκθέσεις – αναφορές, γενικά κείμενα, σχέδια, διαγράμματα, κλπ.

Η «Τεχνολογία Λογισμικού» (software engineering), λόγος περί τέχνης, είναι η επιστήμη η οποία περιγράφει τον τρόπο εκείνο με τον οποίο θα πραγματοποιηθεί μια τεχνική κατασκευή συμπεριλαμβανομένων και των λογισμικών.

Τεχνολογία Λογισμικού ορίζεται «***η συστηματική προσέγγιση για την ανάλυση, τη σχεδίαση, την αξιολόγηση, την υλοποίηση, τον έλεγχο, τη συντήρηση, και τον επανασχεδιασμό λογισμικού, δηλαδή την εφαρμογή της μηχανικής στο λογισμικό***» (Laplante, 2007).

Η Τεχνολογία Λογισμικού περιγράφει – καθορίζει ένα καλής ποιότητας Λογισμικό, σύμφωνα με τα παρακάτω:

- Τις ενέργειες που υλοποιούνται
- Τα προϊόντα που εκδίδονται
- Την περιγραφή των προδιαγραφών – προτύπων των τελικών προϊόντων και ενεργειών
- Τις μεθόδους ελέγχων, επαλήθευσης και διασφάλισης της ποιότητας.

Γι' αυτό και η ανάγκη για «Καλό Λογισμικό» είναι αυτονόητη και επιτακτική, μιας και ο ρόλος των υπολογιστών στην οικονομία, την παραγωγή, την ενημέρωση, την ψυχαγωγία, την εκπαίδευση, κα. εκπληρώνεται μόνο με Λογισμικό που κάνει σωστά τη δουλειά του, παράγεται με χαμηλό κόστος, με ελάχιστή διάρκεια παραγωγής και πάντα σε άριστη ποιότητα.

Αντικείμενο της Τεχνολογίας Λογισμικού είναι η επιστημονική θεμελίωση του κύκλου ζωής, της παραγωγικής διαδικασίας, του τρόπου περιγραφής - συντήρησης και του πλαισίου χρήσης ενός Λογισμικού. Οι παραπάνω διαδικασίες είναι ιδιαίτερα απαιτητικές, μιας και τα Λογισμικά είναι δύσκολα από τη φύση τους και ως υπόσταση τελικού προϊόντος. Ένα άλλο αντικείμενο της Τεχνολογίας Λογισμικού είναι η αναζήτηση συστημικών τρόπων για να εμφανίσουμε ποια στοιχεία του πρέπει να κατασκευάσουμε, έτσι ώστε ένα λογισμικό να κάνει τη δουλειά του σωστά.

Τελικά, τι είναι «ποιοτικό – καλό» Λογισμικό και πώς το υλοποιώ; Σίγουρα είναι ένα δύσκολο ερώτημα, μιας και το κόστος, η ποιότητα και ο χρόνος ανάπτυξης είναι αμοιβαία συγκρουόμενα στοιχεία.

2.4. Απαιτήσεις Συστήματος Λογισμικού

Για την σωστή ανάπτυξη ενός λογισμικού μια από τις διαδικασίες, η οποία θα πρέπει να γίνεται πάντα πρώτη, είναι η προδιαγραφή, ανεξάρτητα από το μοντέλο κύκλου ζωής που ακολουθεί. Ταυτόχρονα, μια από τις σημαντικότερες εργασίες κατά τη διάρκεια ανάπτυξης ενός λογισμικού είναι ο προσδιορισμός των προδιαγραφών των απαιτήσεων. Και αυτό γιατί είναι μια δύσκολη και δημιουργική διαδικασία, απαιτώντας ιδιαίτερες γνώσεις επικοινωνίας, δομημένη – κριτική σκέψη και συστηματική προσέγγιση. Το μεθοδολογικό πλαίσιο για την πραγματοποίηση της εργασίας αυτής ορίζεται από την **Τεχνολογία Λογισμικού**.

Μια απαίτηση από το σύστημα είναι η περιγραφή μιας εργασίας που θα πρέπει να εκτελείται από κάποια εκ των συνιστωσών του συστήματος (άνθρωποι, μηχανές, λογισμικό) ή ενός χαρακτηριστικού το οποίο θα πρέπει να έχει ένα σύστημα (Β. Βεσκούκης, 2015).

Η διαδικασία προσδιορισμού των υπηρεσιών που απαιτεί ο πελάτης από ένα σύστημα και των περιορισμών κάτω από τους οποίους το σύστημα λειτουργεί και αναπτύσσεται ορίζεται ως **τεχνολογία απαιτήσεων**. Οι ίδιες οι απαιτήσεις είναι οι περιγραφές των υπηρεσιών – των περιορισμών του συστήματος, οι οποίες παράγονται κατά τη διαδικασία της τεχνολογίας των απαιτήσεων.

Τι είναι όμως απαίτηση; **Απαίτηση** θα μπορούσε να είναι το οτιδήποτε, από κάτι πολύ αφηρημένο, όπως μια υπηρεσία – ένας περιορισμός, μέχρι μια λεπτομέρεια, ένας μαθηματικός ορισμός του συστήματός μας. Άρα, οι απαιτήσεις μπορούν να έχουν διπλή λειτουργία. Μπορούν να αποτελούν μια βάση διαπραγμάτευσης μιας σύμβασης, επομένως θα πρέπει να αφήνουν περιθώρια διαφορετικών ερμηνειών. Μπορούν να αποτελούν τη βάση για την ίδια τη σύμβαση, επομένως θα πρέπει να καθορίζονται λεπτομερώς. Και οι δυο αυτές μορφές μπορούν να παίξουν το ρόλο των απαιτήσεων.

Μια απαίτηση από το λογισμικό είναι μια λειτουργία που αυτό θα πρέπει να επιτελεί, ή μια συνθήκη που θα πρέπει να ικανοποιεί, όταν θα έχει ολοκληρωθεί η κατασκευή του (Β. Βεσκούκης, 2015).

Οι τύποι των απαιτήσεων χωρίζονται στις **απαιτήσεις του χρήστη (user requirements)**, στις **απαιτήσεις του συστήματος (system requirements)** και στις **προδιαγραφές σχεδιασμού λογισμικού (software design specification).** Οι **απαιτήσεις του χρήστη** περιλαμβάνουν τις δηλώσεις σε φυσική γλώσσα και τα διαγράμματα των υπηρεσιών που παρέχει το σύστημα και των λειτουργικών περιορισμών του (γράφονται με προορισμό τους πελάτες). Απευθύνονται στο διοικητικό προσωπικό του πελάτη και του προμηθευτή, τους τελικούς χρήστες, τους μηχανικούς τον πελάτη και τους αρχιτέκτονες του συστήματος. Οι **απαιτήσεις του συστήματος** περιλαμβάνουν ένα δομημένο έγγραφο που περιγράφει με λεπτομέρειες τις λειτουργίες, τις υπηρεσίες και τους λειτουργικούς περιορισμούς του συστήματος. Ορίζεται με ακρίβεια πώς πρέπει να υλοποιηθεί ώστε να αποτελεί μέρος της σύμβασης μεταξύ του πελάτη και του αναδόχου. Απευθύνονται στους τελικούς χρήστες, τους μηχανικούς του πελάτη, τους αρχιτέκτονες του συστήματος και τους προγραμματιστές του προμηθευτή. Το έγγραφο **προδιαγραφών σχεδιασμό λογισμικού** περιέχει ένα δομημένο έγγραφο που περιγράφει με λεπτομέρειες τις λειτουργικές απαιτήσεις, τους περιορισμούς σχεδίασης, τα ιδιώματα, τις απαιτήσεις επίδοσης και τις άλλες απαιτήσεις λειτουργίας του συστήματος. Απευθύνεται στους μηχανικούς του πελάτη, τους αρχιτέκτονες του συστήματος και τους προγραμματιστές του προμηθευτή.

Οι προδιαγραφές οδηγούνται κατά κύριο λόγο από τους επιχειρηματικούς στόχους (υψηλού επιπέδου), το περιβάλλον λειτουργίας και το οργανωτικό περιβάλλον. Στη διαμόρφωση των προδιαγραφών συμμετέχουν οι παρακάτω εταίροι (stakeholders): χρήστες, πελάτες, ρυθμιστικές αρχές και μηχανικοί ανάπτυξης. Στοιχεία μπορούν ακόμα να προέλθουν και από άλλες πηγές, όπως μελέτες της αγοράς.

Μπορούμε να χωρίσουμε τις απαιτήσεις σε δυο μεγάλες κατηγορίες. Στις **λειτουργικές απαιτήσεις** (*functional requirements*) και στις **μη λειτουργικές απαιτήσεις** (*non-functional requirements*). Οι **λειτουργικές απαιτήσεις** θα πρέπει να είναι πλήρεις (Complete) και συνεπείς (Consistent), ενώ περιγράφουν τις εργασίες (λειτουργίες) που θα πρέπει να εκτελεί το λογισμικό, την αλληλεπίδραση ανάμεσα στο σύστημα και το περιβάλλον του και, τέλος, περιγράφουν τον τρόπο συμπεριφοράς του συστήματος, όταν δέχεται συγκεκριμένα ερεθίσματα. Οι **μη λειτουργικές απαιτήσεις** θα περιγράφουν χαρακτηριστικά που πρέπει να έχει το λογισμικό, τα οποία δεν αφορούν την εκτέλεση κάποιας λειτουργίας από αυτό, αλλά και τις προδιαγραφές που περιορίζουν τις επιλογές για την αναζήτηση της λύσης στο πρόβλημα. Παραδείγματα μη λειτουργικών απαιτήσεων αποτελούν οι απαιτήσεις χρήστη,

αξιοπιστίας, επιδόσεων, υποστήριξης, σχεδίασης, υλοποίησης, επικοινωνίας με άλλα συστήματα, βάσεων δεδομένων και οι φυσικές απαιτήσεις.

Ο κύκλος ζωής των απαιτήσεων περιλαμβάνει:

- **Τις τεχνικές εκμαίευσης (elicitation):** μέσω των συνεντεύξεων, σεναρίων, της κατασκευής αρχέτυπων, των δομημένων συναντήσεων, της παρατήρησης και της υπάρχουσας εσωτερικής και εξωτερικής τεκμηρίωσης.
- **Την ανάλυση και διαπραγμάτευση (analysis and negotiation):** οι οποίες γίνονται με την περιγραφή του ιδεατού μοντέλου (conceptual model) που παριστάνει τον κόσμο και του μοντέλου των απαιτήσεων (requirements model) που παριστάνει τις απαιτήσεις του λογισμικού. Για την περιγραφή του ιδεατού μοντέλου χρησιμοποιούνται διαγράμματα UML, όπως Διαγράμματα Κλάσεων (Class Diagram), Διαγράμματα Δραστηριότητας (Activity Diagram) και Διαγράμματα Κατάστασης (Statechart Diagram). Για την περιγραφή του μοντέλου των απαιτήσεων χρησιμοποιούνται διαγράμματα UML, όπως Διαγράμματα Κλάσεων (Class Diagram), Διαγράμματα Κατάστασης (Statechart Diagram), Διαγράμματα Περιπτώσεων Χρήσης (Use Case Diagram), Διαγράμματα συνεργασίας (Sequence Diagram) και Διαγράμματα Ακολουθίας (Collaboration Diagram).
- **Καθορισμός (specification):** Περιγράφει τις δυνατότητες (Capabilities) του συστήματος. Πρέπει να είναι πλήρης (Complete) και Συνεπής (Consistent).
- **Επικύρωση (Validation):** Αναφέρεται στον έλεγχο λαθών, συνέπειας, πληρότητας, τη ρεαλιστικότητα και την επαληθευσιμότητα.

Τέλος, η **διαχείριση των απαιτήσεων** θα πρέπει να γίνεται με διαχείριση των προδιαγραφών που περιλαμβάνει τον προσδιορισμό προδιαγραφών, τη διεργασία διαχείρισης αλλαγών, τις πολιτικές ιχνηλασιμότητας (πηγές απαιτήσεων, σύνδεση μεταξύ απαιτήσεων και σύνδεση απαιτήσεων με το σχέδιο λογισμικού) και τα εργαλεία υποστήριξης. Επίσης, συχνά η περιγραφή των απαιτήσεων γίνεται σε φυσική γλώσσα. Κατά την περιγραφή των απαιτήσεων θα πρέπει να αποφεύγονται τα παρακάτω συχνά λάθη: Μακροπερίοδος λόγος με παρενθετικές προτάσεις, Χρήση διφορούμενων όρων, Παρουσίαση πολλαπλών απαιτήσεων ως μια απαίτηση, Ασυνέπεια στη χρήση όρων με χρήση διαφορετικών όρων για την ίδια έννοια και Σύγχυση λειτουργικών - μη λειτουργικών απαιτήσεων. Συνεπώς, ο ρόλος των απαιτήσεων στην κατασκευή ενός λογισμικού είναι μια σύνθετη εργασία και αποτελεί ένα σημαντικό παράγοντα που θα πρέπει να λαμβάνεται υπόψιν εξαρχής.

2.5. Κύκλος Ζωής Έργου

Με την έννοια του **Κύκλου Ζωής ενός Έργου (Project Life Cycle)** ορίζουμε τη σειρά των ενεργειών που απαιτούνται από τη σύλληψη μια ιδέας ενός έργου μέχρι την υλοποίησή του. Αρχικά, θα πρέπει να γίνει ο προσδιορισμός της ιδέας, σειρά έχει η ανάπτυξη – έκφρασή της μέσω ενός σχεδίου εργασίας, το οποίο θα μπορεί να εφαρμοστεί στην πράξη, και στη συνέχεια, να αξιολογηθεί. Οι ιδέες προσδιορίζονται στα πλαίσια μιας στρατηγικής που συμφωνείται από τον κύριο του έργου και τον ανάδοχο. Τα σημαντικά ορόσημα μεταξύ των φάσεων του κύκλου ζωής αναπαριστούν τα σημεία λήψης των αποφάσεων υψηλής σημασίας, ενώ οι φάσεις, σε πολλές περιπτώσεις, μπορεί επικαλύπτονται.

Κάθε έργο, και κυρίως κάθε έργο Πληροφορικής, έχει διαφορετικά χαρακτηριστικά και συνεπώς διαφορετικό Κύκλο Ζωής **(Clarke).**

Ο κύκλος ζωής ενός έργου παρέχει τη δομή που εξασφαλίζει τη συμμετοχή των εμπλεκόμενων στο έργο, καθώς και τη διαθεσιμότητα των απαραίτητων πληροφοριών που πιθανώς να χρησιμεύουν κατά τη διάρκεια ζωής του έργου.

Η έννοια της διαχείρισης του Κύκλου Ζωής (PCM) ενός έργου εισήχθη από την Ευρωπαϊκή Επιτροπή στις αρχές της δεκαετίας του ’90, με σκοπό τη βελτίωση της ποιότητας σχεδιασμού, της διαχείρισης των έργων και της αποτελεσματικότητας των έργων που χρηματοδοτούσε η Ευρωπαϊκή Ένωση. Η μεθοδολογία αυτή αναπτύχθηκε στα τέλη της δεκαετίας του ’80, μετά από έρευνα, μιας και τα συμπεράσματα της αξιολόγησης έδειχναν έργα κακώς εκτελεσμένα από πολλές και διάφορες αιτίες. Θέτοντας τον τελικό στόχο, ένα έργο θα πρέπει να έχει μετρήσιμα SMART χαρακτηριστικά (Specific, Measurable, Achievable, Realistic, Time – Limited).

Τα **βασικά στοιχεία ενός συμβατικού κύκλου ζωής,** σύμφωνα με τον Clarke, είναι τα εξής:

- **Η Φάση (Phase),** είναι το σύνολο από τις αλληλοσχετιζόμενες δραστηριότητες
- **Η Εργασία (Task),** είναι μια συγκεκριμένη δραστηριότητα με ένα καθορισμένο σκοπό.
- **Το Ορόσημο (Milestone) ή Παραδοτέο (Deliverable),** είναι ένα ορισμένο αποτέλεσμα μια Φάσης ή Εργασίας.

Αναλυτικά, ένας **συμβατικός Κύκλος Ζωής περιέχει τις ακόλουθες φάσεις**:

- **Σχεδιασμός Έργου (Project Planning):** Αρχικά καθορίζονται ο σκοπός του έργου και οι εμπλεκόμενοι σε αυτό. Γίνεται ανάλυση κόστους και πόρων. Παραδοτέο αυτής της φάσης είναι το συμφωνημένο Terms of Reference και το Project Plan.
- **Καταγραφή Απαιτήσεων (Requirement Analysis):** Σε αυτή τη φάση καθορίζεται το «τι» θα κάνει το λογισμικό από όλους τους εμπλεκόμενους στο έργο και τους τελικούς

χρήστες. Απαραίτητη είναι η συμμετοχή όλων των παραπάνω στο σχεδιασμό του προϊόντος. Παραδοτέο αυτής της φάσης είναι το συμφωνημένο System Requirements Statement (SRS).

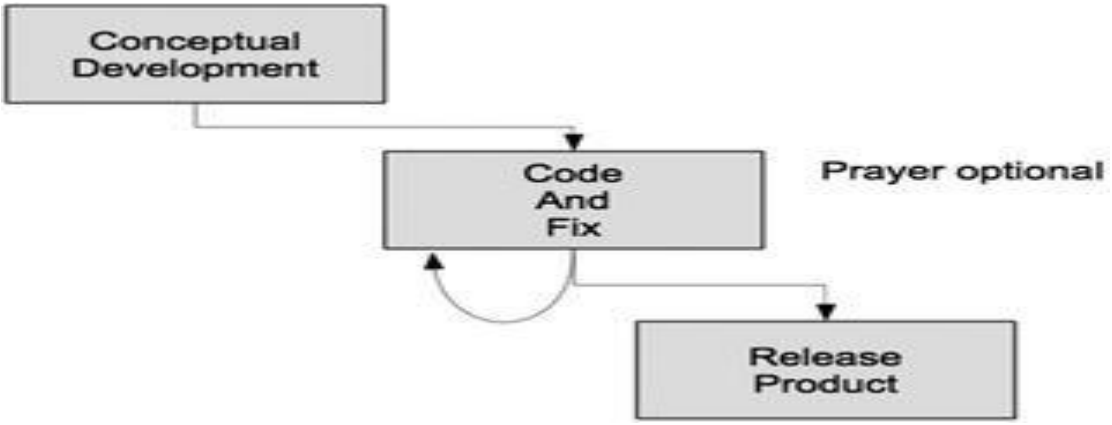
- **Σχεδιασμός Συστήματος (System Design):** Στην ενότητα αυτή αναπτύσσεται «πώς» το λογισμικό θα λειτουργεί σύμφωνα με το SRS. Παραδοτέο αυτής της φάσης είναι το System Design Specification.
- **Υλοποίηση (Development):** Στην υλοποίηση περιλαμβάνουμε την αποτίμηση και την συλλογή του υπάρχοντος λογισμικού. Τη συγγραφή νέου, τον καθορισμό των αναλυτικών προδιαγραφών – δραστηριοτήτων, την ενοποίηση όλων των στοιχείων στο τελικό προϊόν, καθώς και το επίπεδο δοκιμής του λογισμικού. Παραδοτέο αυτής της φάσης είναι η εφαρμογή, ολοκληρωμένη, που ικανοποιεί όλα τα παραπάνω.
- **Εφαρμογή (Implementation):** Λίγο πριν την τελική έκδοση του, το λογισμικό χρησιμοποιείται από τους τελικούς χρήστες. Παραδοτέο αυτής της φάσης είναι η αποδοχή λειτουργίας του συστήματος.
- **Λειτουργία (Operation):** Στην τελική φάση το λογισμικό χρησιμοποιείται ευρύτερα από τους χρήστες, οι οποίοι καλούνται να εντοπίσουν πιθανά προβλήματα, παραλείψεις ή τυχόν αλλαγές και προσθήκες.

Αν και οι φάσεις δείχνουν να ακολουθούν μια σειρά, ολοκληρώνοντας πρώτα η μία την άλλη, στην πραγματικότητα σπάνια ακολουθείται αυτή η σειρά.

2.6. Μοντέλα Κύκλου Ζωής Έργων

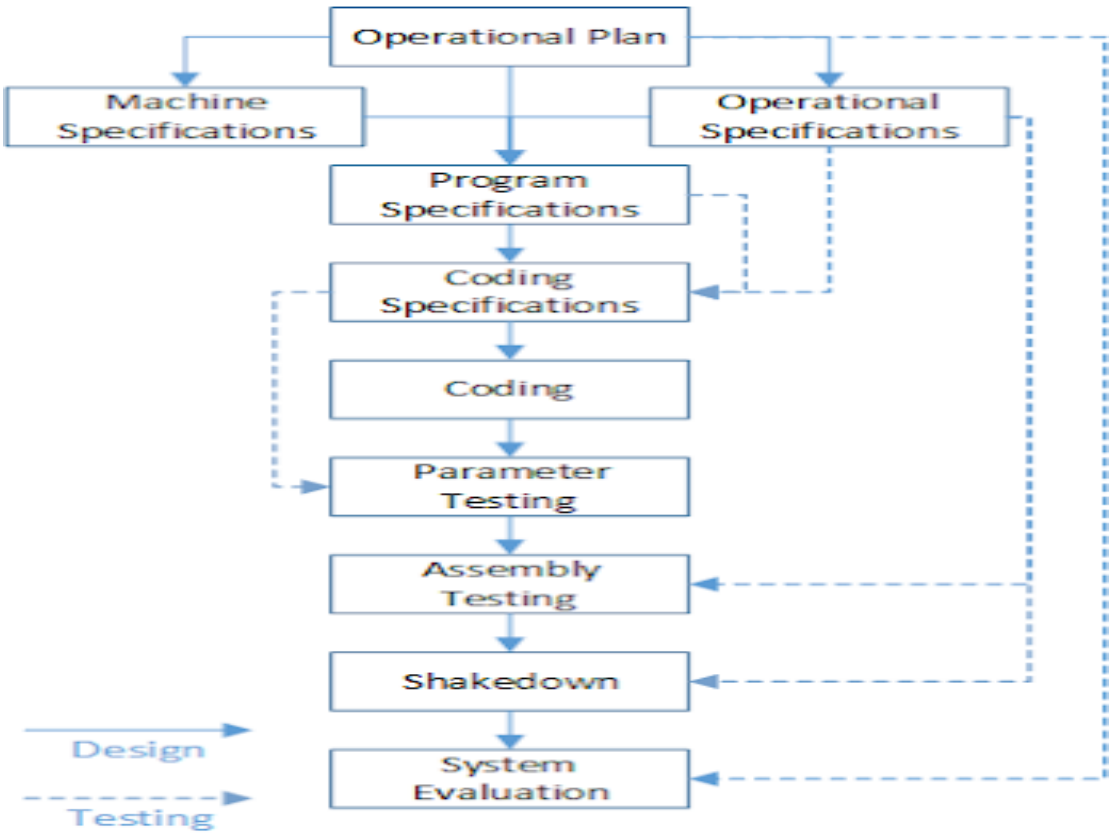
Η ανάπτυξη λογισμικού στα πρώτα χρόνια της πληροφορικής είχε κυρίαρχο τον προγραμματιστή, του οποίου οι ικανότητες επηρέαζαν το τελικό αποτέλεσμα. Εμφανής ήταν και η έλλειψη μιας δομημένης διαδικασίας έκδοσης προϊόντων. Ενώ η συνεισφορά του τελικού χρήστη ήταν σχεδόν μηδενική και δεν απασχολούσε ιδιαιτέρα τους κατασκευαστές.

Για να περιγραφεί η διαδικασία ανάπτυξης λογισμικού μεταγενέστερα, χρησιμοποιήθηκε ο όρος «**Code & Fix Model**», ή αλλιώς **Μοντέλο που στοχεύει στη δημιουργία κώδικα και στη διόρθωση λαθών στον κώδικα**, ο οποίος περιελάμβανε δύο στάδια ανάπτυξης, αυτό της συγγραφής του κώδικα και αυτό της διόρθωσης τυχών προβλημάτων. Αυτό το μοντέλο δημιούργησε μια σειρά προβλημάτων, που από νωρίς έκαναν αντιληπτή την ανάγκη για ενσωμάτωση και άλλων στοιχείων για τη σωστή ανάπτυξη.



Εικόνα 3 - Code & Fix Model - Πηγή: media.springernature.com

Παράλληλα, η ανάγκη ύπαρξης ενός πιο δομημένου τρόπου ανάπτυξης έργων λογισμικού εμφανίζεται ακόμα και από τον Bennington από το 1956, μεταφέροντας τις εμπειρίες του από την ανάπτυξη μεγάλων έργων. Το μοντέλο που περιέγραψε είναι εκείνο του «**Stagewise Model**» **(Μοντέλο Ακολουθιακών Σταδίων),** το οποίο αποτελείται από μια σειρά διαδοχικών σταδίων.

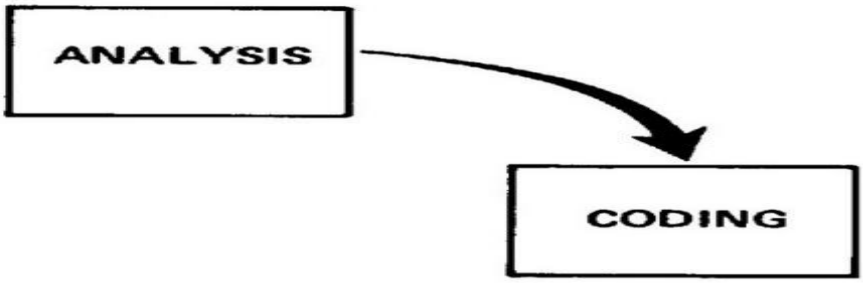


Εικόνα 4 - Stagewise Model - Πηγή: lh3.googleusercontent.com

2.6.1. Μοντέλο Καταρράκτη (Waterfall Model)

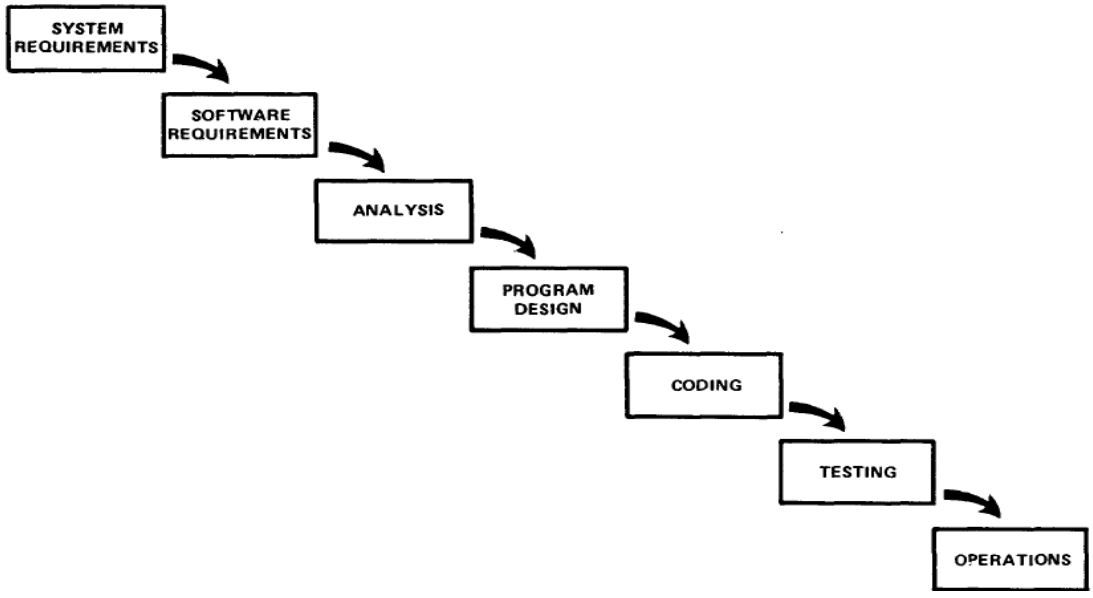
Και αφού πέρασαν αρκετά χρόνια δοκιμών, στις αρχές του 1970 παρουσιάζεται από τον Royce η πρώτη προσπάθεια περιγραφής ολοκληρωμένης μεθοδολογίας ανάπτυξης λογισμικού. Η αρχική του ονομασία ήταν «Διαχείριση Ανάπτυξης Μεγάλων Συστημάτων», ενώ αργότερα δόθηκε η ονομασία του «Καταρράκτη», η οποία αποτελεί την εξέλιξη του «Stagewise Model».

Σύμφωνα με τον συγγραφέα, δυο είναι τα βήματα ανάπτυξης λογισμικού, ανεξάρτητα από το μέγεθος και την πολυπλοκότητα. Το πρώτο στάδιο είναι εκείνο της **ανάλυσης (Analysis**), το οποίο ακολουθεί ένα στάδιο **συγγραφής κώδικα (Coding)**.



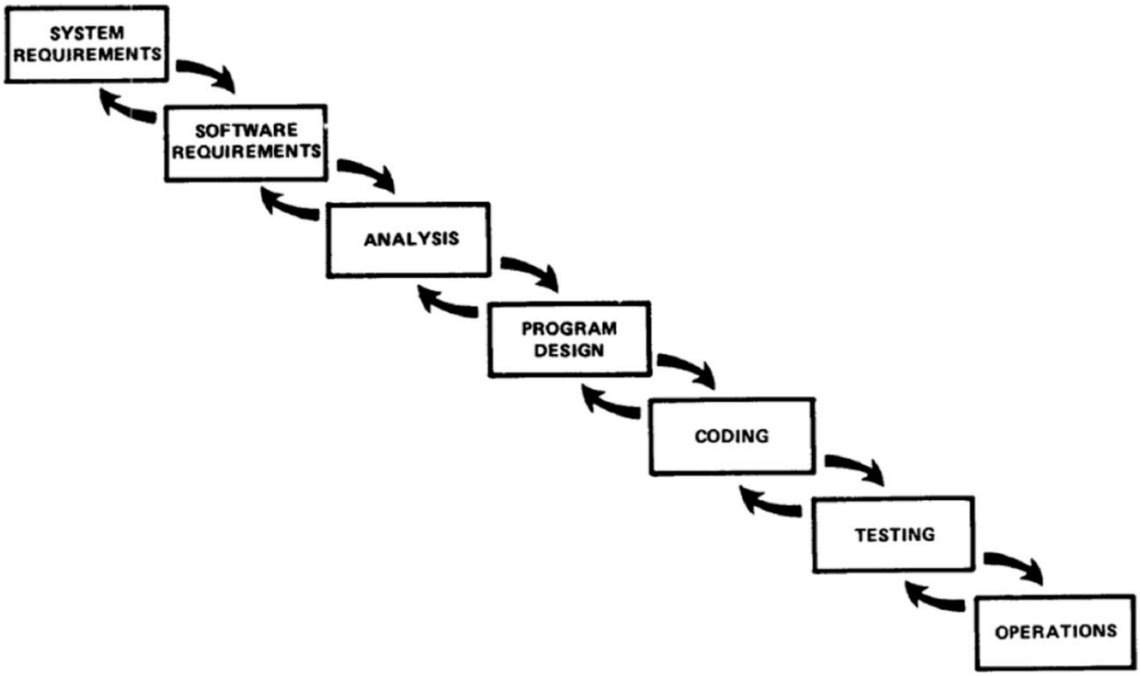
Εικόνα 5 - 1ο Στάδιο Ανάπτυξης του Waterfall Model - Πηγή: arialdomartini.wordpress.com

Στις περιπτώσεις μικρών συστημάτων λειτουργεί απόλυτα αυτή η ακολουθία, ενώ για μεγάλα συστήματα θα πρέπει να εισάγονται και επιπλέον βήματα ανάπτυξης που θα συνεισφέρουν (Contribute) στο τελικό προϊόν, αυξάνοντας το κόστος ανάπτυξης, γι' αυτό και θα πρέπει να παρακολουθούνται.



Εικόνα 6 - 2ο Στάδιο Ανάπτυξης του Waterfall Model - Πηγή: researchgate.net

Έτσι, εισάγονται οι δυο φάσεις ανάλυσης απαιτήσεων, του σχεδιασμού και των δοκιμών, οι οποίες βρίσκονται ανάμεσα στις απαιτήσεις. Εμβαθύνοντας, ο συγγραφέας έδωσε έμφαση στην επαναληπτική σχέση (iteration relationship) μεταξύ των φάσεων ανάπτυξης. Δηλαδή, καθώς τα βήματα προχωρούν, ο σχεδιασμός γίνεται πιο αναλυτικός, ενώ υπάρχει αλληλεπίδραση μεταξύ των προηγούμενων και των επόμενων βημάτων, ενώ πολύ σπάνια αλληλεπιδρούν με τα πιο απομακρυσμένα.



Εικόνα 7 - 3ο Στάδιο Ανάπτυξης του Waterfall Model - Πηγή: changearc.files.wordpress.com

Άρα, αναλυτικά οι φάσεις που διέρχεται το «Waterfall Model» έχουν ως εξής:

1.**Ανάλυση Απαιτήσεων (Requirements):** Ένα από τα μεγαλύτερα προβλήματα αποτυχίας έργων λογισμικού είναι η μη σωστή καταγραφή των απαιτήσεων – αναγκών του συστήματος. Θα πρέπει να γίνεται εξαρχής και σωστά από τους υπεύθυνους ανάπτυξης του συστήματος, μέσω συνεντεύξεων ή άλλων μεθόδων εκμαίευσης, και σε συνεργασία με τους μελλοντικούς χρήστες να προχωρούν στην καταγραφή των προδιαγραφών και των απαιτήσεων. Αυτό βοηθά έτσι ώστε οι πελάτες να κατανοήσουν πλήρως αυτό που θα αναπτυχθεί.

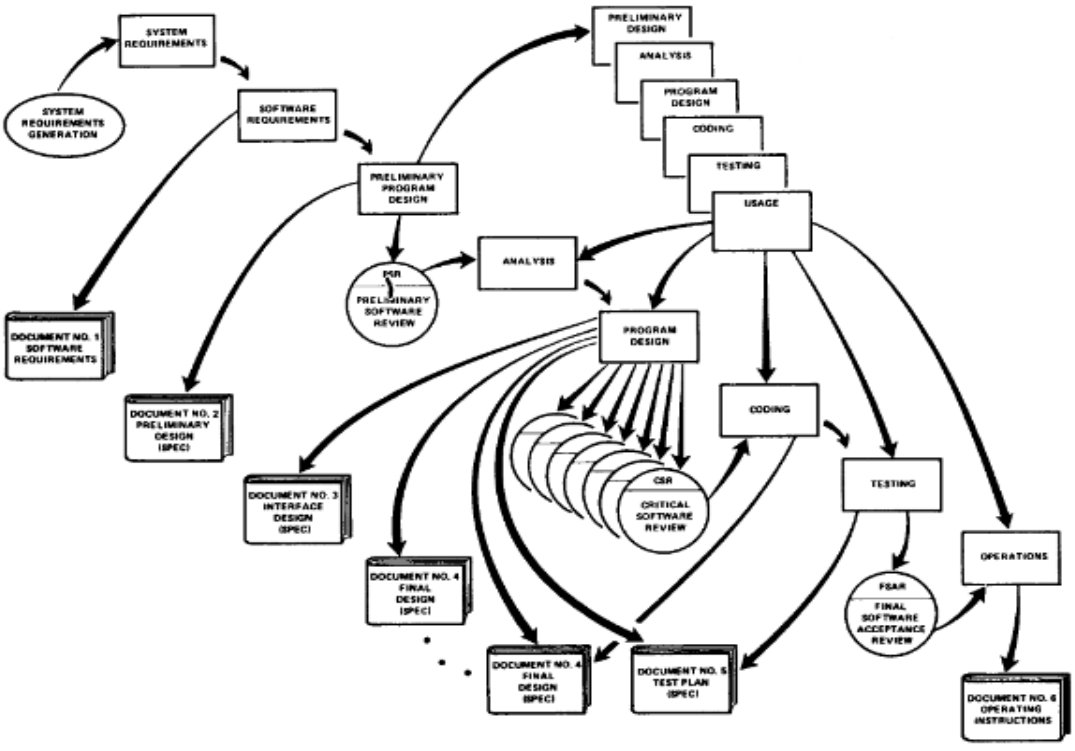
2.**Σχεδιασμός (Design):** Αρχικά γίνεται ένας αρχικός προκαταρκτικός σχεδιασμός (Preliminary Program Design) χωρίς ανάλυση, καταγράφοντας ότι δεν θα αποτύχει εξαιτίας τεχνικών αιτιών (μπορεί να προηγηθεί και της φάσης των απαιτήσεων και της ανάλυσης). Ακολουθεί ένας γενικός σχεδιασμός της αρχιτεκτονικής του συστήματος. Άρα, στη φάση αυτή απαιτείται μεγάλος βαθμός τεκμηρίωσης. Δηλαδή, απαιτούνται 6 διαφορετικά έγγραφα μέχρι

να ολοκληρωθεί η φάση του «Σχεδιασμού», τα οποία είναι τα εξής: Απαιτήσεις Λογισμικού, Προκαταρκτικός Σχεδιασμός, Σχεδιασμός Interface, Τελικός Σχεδιασμός, Σχέδια Ελέγχων – Δοκιμών και Οδηγίες Λειτουργίας.

3.**Υλοποίηση (Implementation):** Στη φάση αυτή το λογισμικού έχει ολοκληρωθεί, συμπεριλαμβανομένων όλων των επιμέρους τμημάτων του. Είναι σημαντικό το σύστημα να έχει αναπτυχθεί τουλάχιστον δυο φορές, έτσι ώστε η τελική έκδοση του λογισμικού που θα παραδοθεί στον πελάτη να μην παρουσιάσει τυχόν προβλήματα.

4.**Επικύρωση (Verification):** Στο στάδιο αυτό γίνεται ο τελικός έλεγχος του λογισμικού. Ελέγχεται αν καλύπτονται όλες οι προδιαγραφές και αν έχει παραχθεί ένα ολοκληρωμένο σύστημα.

5.**Συντήρηση (Maintenance):** Η φάση αυτή πραγματοποιείται αφού έχει ολοκληρωθεί το λογισμικό και έχει παραδοθεί στους χρήστες. Περιλαμβάνονται, επίσης, οι αλλαγές που πρέπει να γίνουν, τα λάθη, η βελτίωση ή και ακόμα νέες απαιτήσεις χρησιμοποιώντας το λογισμικό. Ακόμα, ο Royce προσέθεσε ότι ο Σχεδιασμός, ο Έλεγχος και οι Δοκιμές θεωρούνται από τις πιο σημαντικές φάσεις κατά τη διάρκεια κατασκευής ενός λογισμικού, διότι εκεί απαιτείται η μεγαλύτερη χρήση των διαθέσιμων πόρων. Ταυτόχρονα μίλησε για τη σημαντικότητα της εμπλοκής του πελάτη στο έργο. Με βάση όλα τα παραπάνω, το τελικό μοντέλο «Καταρράκτη», όπως πρότεινε ο Royce (1970), έχει την εξής δομή:



Εικόνα 8 - Αναπτυγμένη έκδοση του Waterfall Model - Πηγή: beza1e1.tuxen.de

Στα **πλεονεκτήματα** της μεθοδολογίας αυτής ανήκουν: (1) η ευρεία αποδοχή και χρήση, (2) ότι αποτέλεσε βάση για όλες τις νεότερες μεθοδολογίες, (3) ξεκάθαρες φάσεις ανάπτυξης με συγκεκριμένα παραδοτέα (ευκολία διαχείρισης – τρόπος μέτρησης και εξέλιξης του έργου) και (4) διευκόλυνση στη σύνταξη συμβολαίων με τις απαιτήσεις του χρήστη να αποτελούν στην ουσία το συμβόλαιο.

Στα **μειονεκτήματα** της μεθοδολογίας αυτής ανήκουν: (1) η γενικότερη εξάρτηση στην τεκμηρίωση (παραγωγή εντύπων), (2) η έμφαση στην ανάπτυξη της ανάλυσης των απαιτήσεων σε έγγραφα, (3) η έλλειψη ευελιξίας, μιας και για να ξεκινήσει κάτι θα πρέπει να έχει ολοκληρωθεί το προηγούμενο, (4) η φάση που βρίσκεται ένα έργο δεν αντικατοπτρίζει πάντα την πραγματική του εξέλιξή.

Συμπερασματικά, το «Μοντέλο Καταρράκτη» (Waterfall Model) αποτελεί, ακόμα και σήμερα, μια από τις πιο δεδομένες μεθοδολογίες ανάπτυξης και διαχείρισης έργων πληροφορικής, παρόλο που δέχτηκε και δέχεται έντονη κριτική. Πάνω στις αρχές του βασίστηκαν όλα τα μεταγενέστερα μοντέλα.

2.6.2. Μοντέλο Πρωτοτυποποίησης (Rapid - Prototyping Model)

Από τα μέσα της δεκαετίας του 1980, άρχισαν να αναπτύσσονται νέες μεθοδολογίες, οι οποίες προσπάθησαν να δώσουν λύσεις στους περιορισμούς του μοντέλου Καταρράκτη. Χαρακτηριστικά παραδείγματα είναι αυτά του μοντέλου Rapid - Prototyping (Connell 1989) και του μοντέλου Spiral (Bohem 1988), το Incremental (staged) Delivery μοντέλο (Wong 1984) και το Evolutionary Delivery μοντέλο (Gilb 1988), καθώς και η μέθοδος Rapid Application Development (RAD) (Martin 1991), η οποία στην ουσία αποτελεί βελτίωση του προηγούμενου. Τέλος, το 1996 εμφανίστηκε η μεθοδολογία Rational Unified Process (RUP) από τον Krutchten, η οποία εισήγαγε τη λογική της αντικειμενοστραφούς ανάλυσης, σχεδιασμού και ανάπτυξης λογισμικού (Object Oriented Software Analysis & Design).

Η μεθοδολογία του **μοντέλου Πρωτοτυποποίησης** αποτελεί ένα μοντέλο ανάπτυξης το οποίο κατασκευάζεται τμηματικά μη ολοκληρωμένες εκδόσεις του λογισμικού που πρόκειται να εκδοθεί. Κάθε μια από τις εκδόσεις αυτές ονομάζεται **πρωτότυπο**, που συμπεριλαμβάνει χαρακτηριστικά και λειτουργίες του τελικού λογισμικού, ενώ ενδέχεται να είναι διαφορετικό από το τελικό λογισμικό. Η μέθοδος αυτή εφαρμόζεται γιατί επιτρέπει στους χρήστες του συστήματος να αξιολογούν τις προτάσεις των μηχανικών λογισμικού δοκιμάζοντας στην πράξη ένα πρωτότυπο του προϊόντος, αντί να πρέπει να κατανοήσουν και να αξιολογήσουν προδιαγραφές και περιγραφές του υπό κατασκευήν λογισμικού, ενώ παράλληλα αυτό θα βελτιώνεται. Όταν υπάρξει η ικανοποίηση όλων των απαιτήσεων, τότε μπορεί να μετατραπεί σε ένα τελικό παραγωγικό λογισμικό.

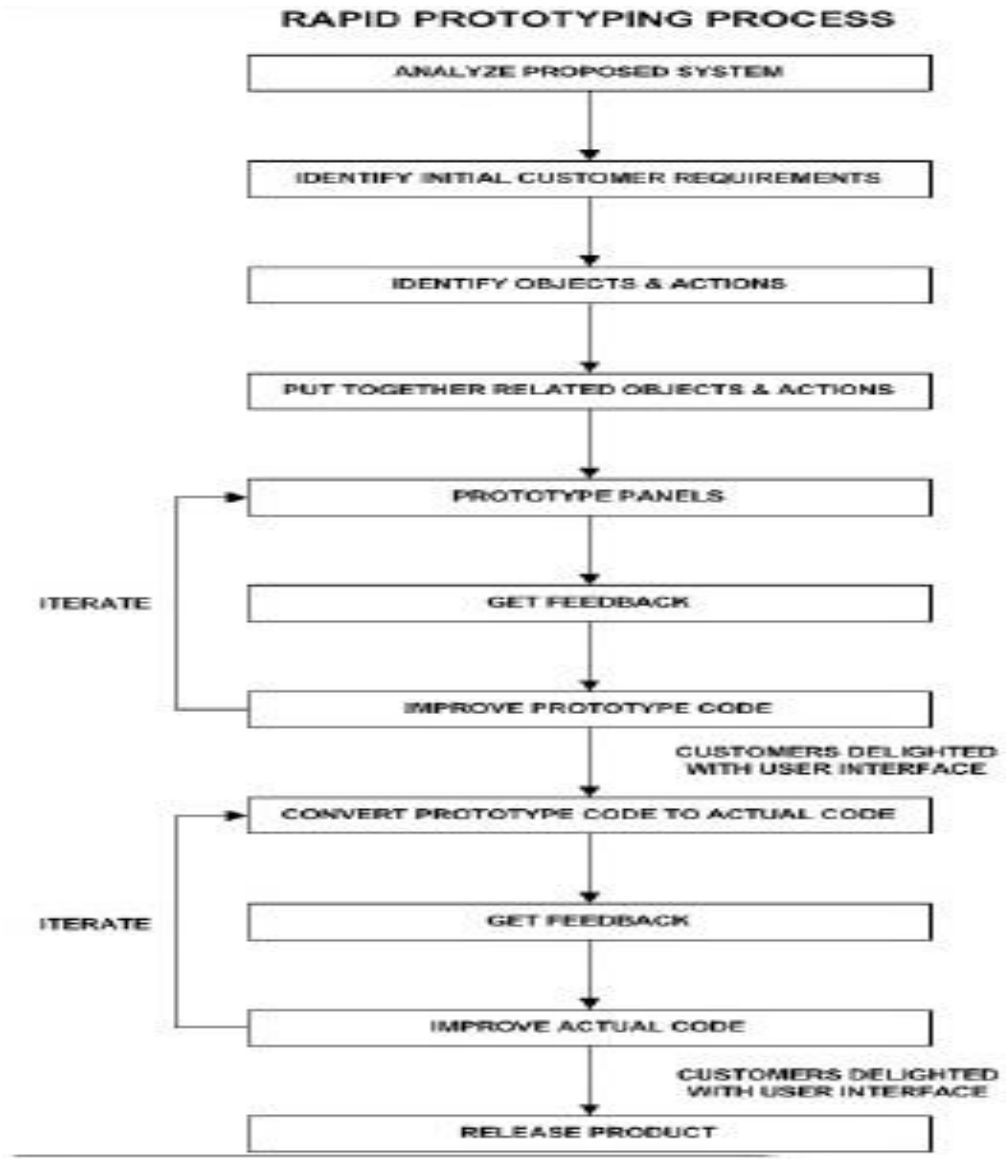
Η μέθοδος δημιουργίας πρωτότυπου είναι μια **επαναληπτική διαδικασία,** όπως όλα τα μοντέλα, και περιέχει βασικές φάσεις ανάπτυξης, όπου κάθε νέα έκδοση συναντά ακριβέστερα τις απαιτήσεις των χρηστών. Τα βήματα έχουν ως εξής:

1. Εντοπισμός και καταγραφή των αναγκών του χρήστη.
2. Γρήγορη ανάπτυξη του λειτουργικού πρωτοτύπου.
3. Δοκιμή του πρωτότυπου, όπου οι χρήστες ενθαρρύνονται να εκφράσουν την άποψή τους για ασάφειες – απορίες και να καθορίσουν ποιες ανάγκες δεν ικανοποιούνται απόλυτα.
4. Διόρθωση και βελτίωση πρωτότυπου, όπου οι σχεδιαστές με βάση τους χρήστες αλλάζουν και βελτιώνουν το πρωτότυπο.

Η διαδικασία που ακολουθείται κατόπιν είναι η συνεχόμενη επανάληψη του βήματος 3 και 4, μέχρι να έχουμε την τελική ικανοποίηση και το τελικό προϊόν με όλες τις προδιαγραφές. Με βάση αυτό υπάρχει η **δυνατότητα απόκτησης άποψης** για την εφαρμογή του λογισμικού **νωρίτερα**. Επίσης, η μεθοδολογία αυτή βοηθά όταν **ΔΕΝ υπάρχει βεβαιότητα για τις απαιτήσεις από την αρχή** από τους πελάτες ή τους τελικούς χρήστες.

Το **βασικό πλεονέκτημα** αυτού του συγκεκριμένου μοντέλου είναι ότι βοηθά τον χρήστη στην κατανόηση του συστήματος από την αρχή, ενώ, ταυτόχρονα, ενσωματώνεται στην παραγωγική διαδικασία. Από την άλλη πλευρά, σημαντικό **μειονέκτημα** είναι ότι το σύστημα σχεδιάζεται και αναπτύσσεται σύμφωνα με το μοντέλο καταρράκτη, με όποιες συνέπειες επιφέρει η χρήση αυτή.

Συμπερασματικά, το μοντέλο **Πρωτοτυποποίησης (Rapid - Prototyping Model)** που παρουσιάστηκε από τον Connell, το 1989, για να εφαρμοστεί σωστά, προϋποθέτει την κατανόηση του επιχειρησιακού προβλήματος της εκάστοτε επιχείρησης, για την οποία θα πρέπει να αναπτυχθεί μια λύση. Η σχηματική αναπαράσταση του μοντέλου έχει ως εξής:



Εικόνα 9 - Μοντέλο Πρωτοτυποποίησης (Rapid - Prototyping Model) - Πηγή: image.slidesharecdn.com

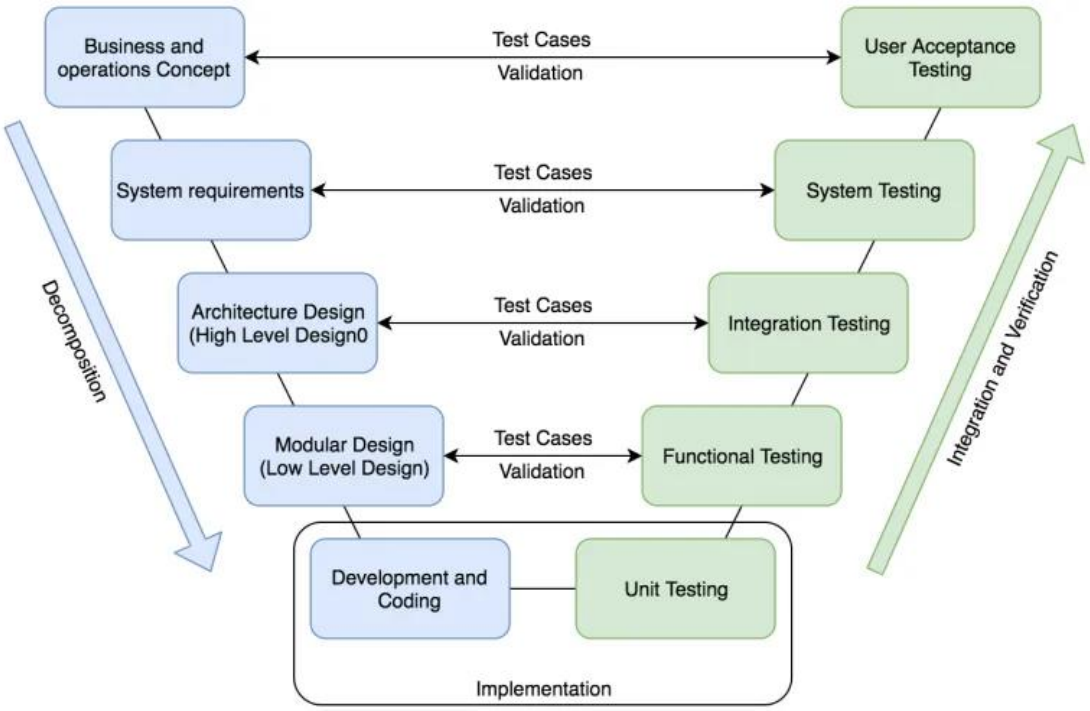
2.6.3. V - Μοντέλο

Το «V- μοντέλο» ανάπτυξης λογισμικού υιοθετεί αρκετά από τα χαρακτηριστικά του μοντέλου καταρράκτη ή θα μπορούσαμε να πούμε ότι είναι και ένα είδος καταρράκτη, καθώς η διαδικασία που ακολουθεί ξεκινά με μια καθοδική κλίση, ενώ μετά έρχεται η ανοδική κλίση.

Η βασική ιδέα του μοντέλου αυτού είναι ο προσδιορισμός των απαιτήσεων του σχεδιασμού και της υλοποίησης, από την αριστερή πλευρά του γράμματος "V" ακολουθώντας καθοδική πορεία, ενώ ο έλεγχος μονάδας και συστήματος έχει ανοδική πορεία και είναι από τη δεξιά πλευρά του "V". Τα λάθη αναγνωρίζονται αργά στον κύκλο ζωής τους έργου, όπως στη μεθοδολογία του καταρράκτη.

Το γράμμα «**V**» το πήρε από τη λέξη **Verification VS Validation** (επαλήθευση έναντι επικύρωσης). Η **επαλήθευση** απαντά στην ερώτηση αν κατασκευάσαμε σωστά το προϊόν και αφορά τη διαδικασία σύγκρισης ενός προϊόντος με τις προδιαγραφές ή τα πρότυπα για τον εντοπισμό των σφαλμάτων. Ο σχεδιασμός επαληθεύεται συγκρίνοντάς τον με τις απαιτήσεις, ενώ ο κώδικας επαληθεύεται συγκρίνοντάς τον με τον σχεδιασμό. Η **επικύρωση** απαντά στην ερώτηση αν κατασκευάσαμε το σωστό προϊόν και είναι η διαδικασία σύγκρισης ενός προϊόντος με τις απαιτήσεις υψηλού επιπέδου, χρησιμοποιώντας μια μέθοδο επικύρωσης για να προσδιορίσουμε εάν το σύστημα κάνει αυτό που έλεγαν οι απαιτήσεις για να είναι επικυρωμένο. Οι ίδιες μέθοδοι χρησιμοποιούνται, επίσης, και στην επαλήθευση.

Η σχηματική αναπαράσταση του μοντέλου έχει ως εξής:



Εικόνα 10 - V – Μοντέλο - Πηγή: i0.wp.com/melsatar.blog

2.6.4. Μοντέλο Λειτουργικής Επαύξησης (Incremental Staged Delivery Model)

Το **Μοντέλο Λειτουργικής Επαύξησης** συνδυάζει τη σειριακή ανάπτυξη του μοντέλου καταρράκτη, ενώ, παράλληλα, έχει και την επαναληπτική ανάπτυξη του μοντέλου παραγωγής πρωτοτύπων. Κεντρική ιδέα είναι η κατάτμηση του λογισμικού σε τμήματα που αναπτύσσονται το καθένα μόνο του, ενώ στο καθένα από αυτά ακολουθείται η μέθοδος του καταρράκτη, ενώ στο τέλος γίνεται η συνένωση. Από την αρχή καθορίζονται οι απαιτήσεις των χρηστών, κατασκευάζεται ένα μικρό λειτουργικό υπό-τμήμα του τελικού συστήματος, ενώ,

παράλληλα, σταδιακά προστίθενται πάνω σε αυτό οι υπόλοιπες λειτουργίες του συστήματος, μετά από μια σειρά επαναληπτικών εκδόσεων.

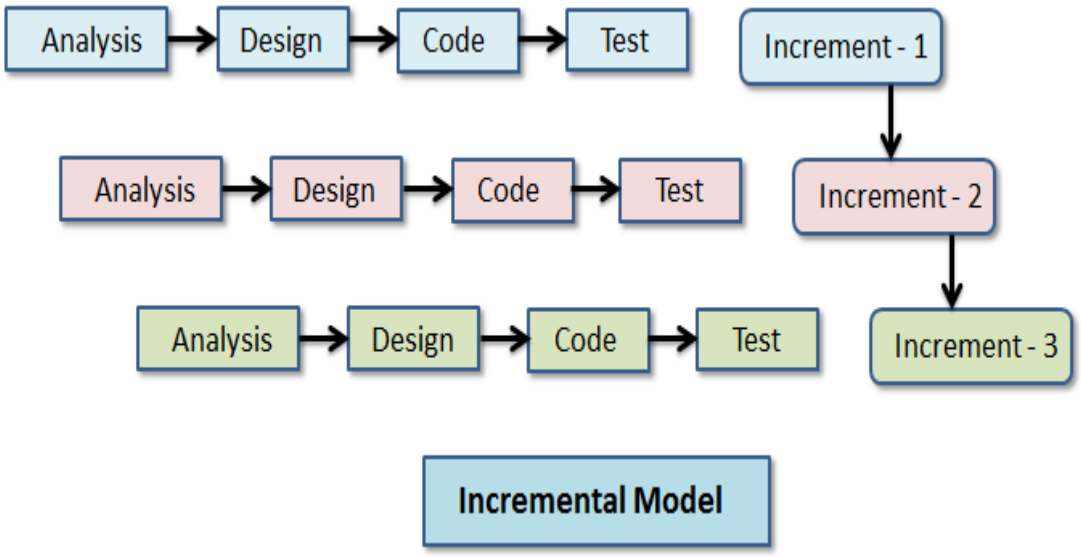
Τα **βασικά πλεονεκτήματα** της μεθοδολογίας είναι τα εξής:

- Ο χρήστης προτείνει βελτιώσεις μετά την παράδοση ενός υπό-τμήματος.
- Από τις πρώτες εκδόσεις ο πελάτης μπορεί να ελέγξει την απόδοση της επένδυσης.

Τα **μειονεκτήματα** της μεθοδολογίας αυτής είναι τα εξής:

- Με την ολοκλήρωση ενός υπό-τμήματος δεν είναι δυνατή η επιστροφή πίσω σε αυτό.
- Η χρήση του καταρράκτη στα υπό-τμήματα αυτά.
- Οποιαδήποτε σημαντική αλλαγή στον σχεδιασμό του επόμενου υπό-τμήματος, που προέρχεται από τις παρατηρήσεις του πελάτη, δεν είναι εφικτή και γίνεται μόνο στην εμφάνιση και όχι στην λειτουργικότητα και την αρχιτεκτονική.

Η σχηματική αναπαράσταση του μοντέλου έχει ως εξής:



Εικόνα 11 - Μοντέλο Λειτουργική Επαύξησης (Incremental Staged Delivery Model) - Πηγή: cdn.educba.com

2.6.5. Μοντέλο Εξελικτικής Ανάπτυξης – Παράδοσης (Evolutionary Delivery Model)

Στο **Μοντέλο Εξελικτικής Ανάπτυξης - Παράδοσης** ο κύκλος ανάπτυξης του λογισμικού χωρίζεται σε μικρότερους επαναληπτικούς κύκλους, όπου στο τέλος του καθενός καταγράφεται το feedback του χρήστη, το οποίο εμπλουτίζει τις γνώσεις και την εμπειρία των κατασκευαστών για τους επόμενους κύκλους. Οι κύκλοι επανάληψης διαρκούν από δύο έως τέσσερις βδομάδες, έως ότου ικανοποιηθούν οι απαιτήσεις τους πελάτη. Η διαφορά του

συγκεκριμένου μοντέλου με το προηγούμενο είναι η σειριακή υλοποίηση και όχι η παράλληλη ανάπτυξη του κύκλου ανάπτυξης ενός λογισμικού. Δηλαδή, με το τέλος του κάθε κύκλου και μετά τα σχόλια του χρήστη σχεδιάζεται ο επόμενος κύκλος.

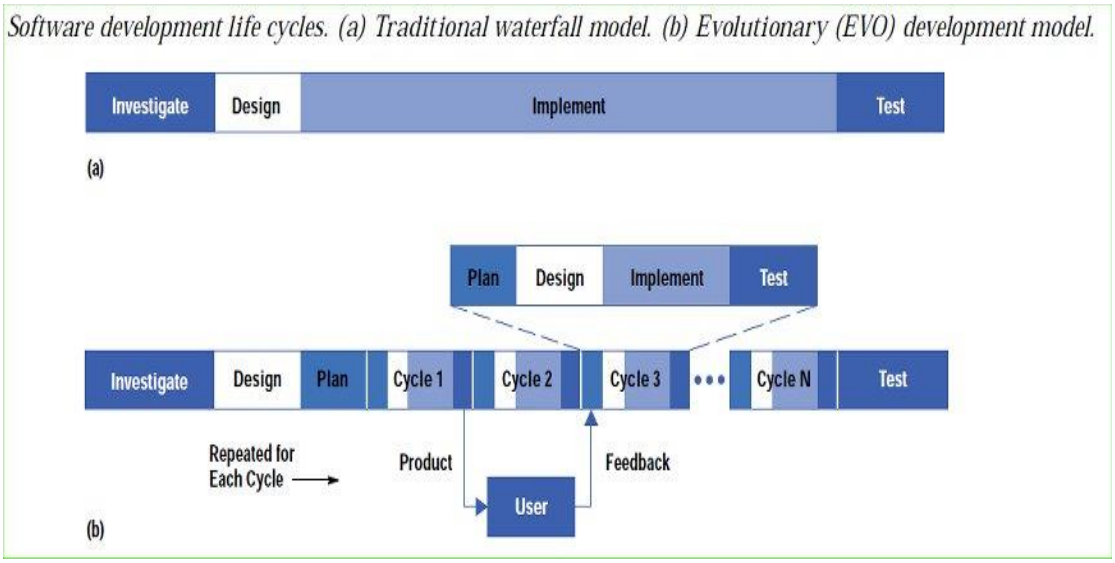
Τα πλεονεκτήματα της χρήσης του μοντέλου αυτού είναι τα εξής:

- Πρόωρη παράδοση τμημάτων του συστήματος, ακόμα και αν οι απαιτήσεις των χρηστών δεν είναι ξεκάθαρες.
- Η εξαγωγή των απαιτήσεων των χρηστών γίνεται από τη χρήση των πρόωρων εκδόσεων ως εργαλεία.

Τα μειονεκτήματα της χρήσης του μοντέλου αυτού είναι τα εξής:

- Δεν είναι ξεκάθαρη η εκτίμηση του τελικού κόστους και του συνολικού χρόνου κατά την έναρξη, μιας και οι απαιτήσεις δεν είναι ξεκάθαρες.
- Επιπρόσθετος χρόνος για ελέγχους
- Κίνδυνος διασφάλισης της αρχιτεκτονικής, από τον κίνδυνο πρόχειρης σύνδεσης των κομματιών.

Η σχηματική αναπαράσταση του μοντέλου έχει ως εξής:



Εικόνα 12 - Μοντέλο Εξελικτικής Ανάπτυξης / Παράδοσης (Evolutionary Delivery Model) - Πηγή: qualityguru.files.wordpress.com

2.6.6. Σπειροειδές Μοντέλο (Spiral Model)

Το Σπειροειδές Μοντέλο αναπτύχθηκε από τον Barry Boehm, το 1988, και παρουσιάστηκε με το άρθρο "A Spiral Model of Software Development and Enhancement" και, όπως αναφέρεται από τον ίδιο στο άρθρο, η δημιουργία του βασίζεται στη συλλογή εμπειριών για τη βελτίωση εφαρμογών καταρράκτη σε μεγάλα κυβερνητικά έργα λογισμικού.

Δεν είναι η πρώτη μεθοδολογία που εφαρμόζει επαναληπτική ανάπτυξη, αλλά ήταν η πρώτη που μίλησε για τη σημαντικότητά της. Κάθε στάδιο ξεκινά με ένα στόχο, ο οποίος ολοκληρώνεται με την συμμετοχή του χρήστη, ο οποίος μπορεί να αναθεωρήσει την εξέλιξη του έργου. Η ανάλυση και εφαρμογή παρουσιάζεται σε κάθε φάση του κύκλου ζωής.

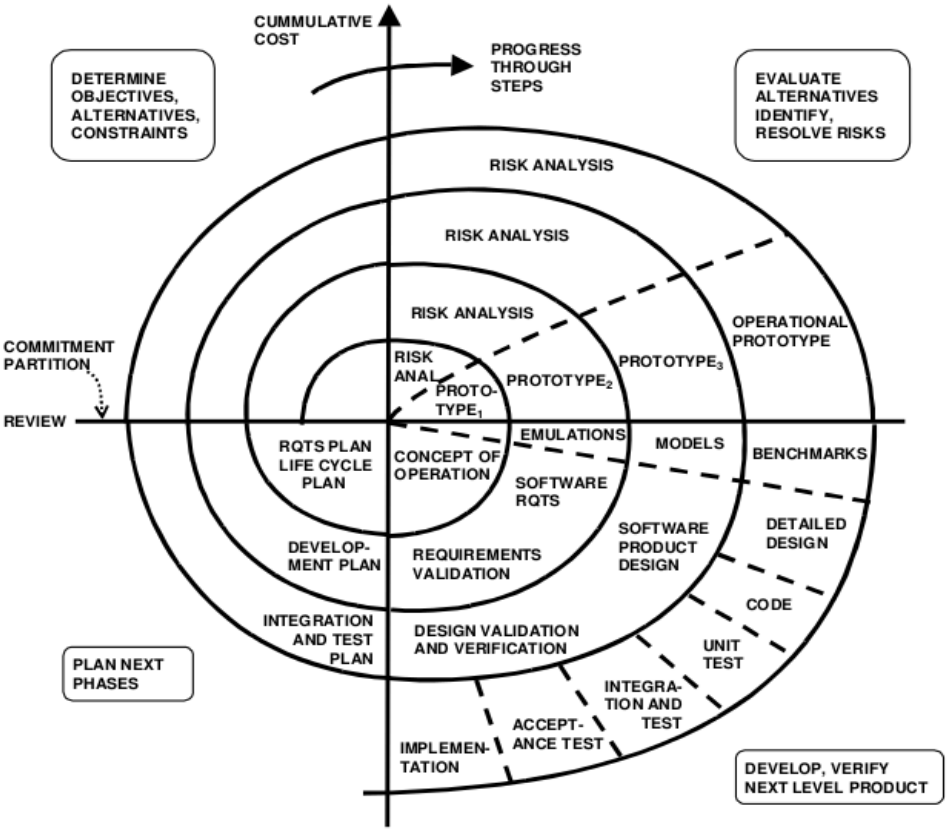
Τα **βασικά χαρακτηριστικά και πλεονεκτήματα** του σπειροειδούς μοντέλου είναι ότι δίνεται έμφαση στην εκτίμηση του ρίσκου σε κάθε κύκλο εργασιών και στην προσπάθειά του να το ελαχιστοποιήσουμε. Αυτό υλοποιείται με την κατάτμηση του έργου σε μικρότερα υπό-έργα με εμφανή την προοπτική αλλαγής ενός τμήματος κατά την ανάπτυξή του. Για τον λόγο αυτό, η ανάπτυξη του λογισμικού γίνεται σε πολλούς κύκλους εργασιών με σταδιακή επέκταση των λειτουργικών χαρακτηριστικών της εφαρμογής.

Στο μοντέλο αυτό διακρίνουμε τέσσερις κατηγορίες εργασιών:

- Ο προσδιορισμός των στόχων.
- Ο εντοπισμός και η επίλυση κινδύνων.
- Η εκτέλεση των διαδικασιών ανάπτυξης και η επαλήθευση.
- Οι εργασίες προγραμματισμού.

Οι **αδυναμίες** του μοντέλου αυτού είναι η πολυπλοκότητα του, η δυσκολία να οριστούν – αναλυθούν τα πολλά υπό-έργα, η δυσκολία να μείνει η ομάδα έργου εστιασμένη στους γενικούς στόχους του έργου, η ανάγκη ύπαρξης εξειδικευμένων στελεχών με ειδικές γνώσεις στη διαχείριση και η έλλειψη ορόσημων κατά την ανάπτυξη του έργου.

Η σχηματική αναπαράσταση του μοντέλου έχει ως εξής:



Εικόνα 13 - Σπειροειδές Μοντέλο (Spiral Model) - Πηγή: xbsoftware.com

2.6.7. Άλλα μοντέλα, και η ανάγκη της εξέλιξης

Τα μοντέλα κύκλου ζωής των έργων για την ανάπτυξη λογισμικού καθ' όλη τη διάρκεια ύπαρξής τους ήταν πολλά και σημαντικά, όπως είδαμε παραπάνω. Εκτός από τα παραπάνω, θα μπορούσαμε να προσθέσουμε ως σημαντικά και το **Μοντέλο του Πίδακα**, όπου η ανάπτυξη γίνεται με αντικειμενοστραφή φιλοσοφία και επαναχρησιμοποίηση των έτοιμων συστατικών. Το **Γενικό Μοντέλο Κύκλου Ζωής,** με ανάπτυξη σε κύκλους σύμφωνα με τα χαρακτηριστικά και τις δυνατότητες του κατασκευαστή. Δηλαδή, μια γενικευμένη μορφή των προηγούμενων μοντέλων κύκλου ζωής.

Στις αρχές της δεκαετίας του 1990, ο James Martin εισήγαγε την **Ταχεία Ανάπτυξη Εφαρμογής** (Rapid Application Development – RAD), που βασίζεται στην εξέλιξη και βελτίωση του **Μοντέλου Εξελικτικής Ανάπτυξης – Παράδοσης (Evolutionary Delivery Model)**, ενώ οι κύριοι στόχοι της συγκεκριμένης μεθοδολογίας είναι: η υψηλή ταχύτητα, η υψηλή ποιότητα και το χαμηλό κόστος. Χαρακτηριστικό γεγονός είναι ότι επιβάλλεται η ενεργή συμμετοχή του πελάτη αλλά και όλων των εμπλεκομένων. Ο έλεγχος του συστήματος γίνεται επαναληπτικά, τόσο από την ομάδα ανάπτυξης όσο και από τους χρήστες.

Επίσης το 1996, ο Philippe Krutchten, με σκοπό να συμπληρώσει και να ολοκληρώσει την γλώσσα UML (Unified Modelling Language), η οποία αποτέλεσε μια πρότυπη μέθοδο μοντελοποίησης λογισμικού, παρουσίασε τη διαδικασία **Rational Unified Process – RUP.** Και αυτή αναφέρεται στη λογική της αντικειμενοστραφής ανάλυσης, σχεδιασμού και ανάπτυξης λογισμικού. Σημαντικό χαρακτηριστικό της είναι η χρήση Σεναρίων Χρήσης (Use Cases), τα οποία χρησιμοποιούνται για τη μοντελοποίηση των απαιτήσεων των χρηστών.

Αναφορικά με όλα τα μοντέλα και τις διαδικασίες κύκλου ζωής (παραδοσιακά και μη) για την ανάπτυξη και παραγωγή ποιοτικού λογισμικού, υπήρξε η ανάγκη για κάτι νέο. Και αυτό γιατί όλες αυτές οι μέθοδοι χαρακτηρίστηκαν ως ανελαστικές, μη παραγωγικές και γραφειοκρατικές, παρά τη συνεχή εμφάνιση νέων πολλαπλών βελτιωμένων μεθοδολογιών. Συνοψίζοντας:

- Τα έργα λογισμικού αντιμετωπίζονται συχνά με την υπόθεση ότι η ανάπτυξη είναι μια προβλεπόμενη και επαναληπτική διαδικασία.
- Επικρατεί έλλειψη ευελιξίας σε ό, τι αφορά τις απαιτήσεις του πελάτη. Δεν καθορίζονται γενικά στα αρχικά στάδια και ακολουθεί αδυναμία προσαρμογής στα μεταγενέστερα.
- Μη σωστή χρήση – κατανομή πόρων.
- Η αδυναμία προσαρμογής στο νέο παγκοσμιοποιημένο επιχειρησιακό περιβάλλον.

2.7. «Ευέλικτες» Μεθοδολογίες (Agile Methodologies)

Μελετώντας κανείς τα μοντέλα κύκλου ζωής ενός λογισμικού που αναπτύχθηκαν παραπάνω κατά το παρελθόν, διαπιστώνει ότι, λόγω των μη ολοκληρωμένων μεθοδολογιών που ανέπτυσσαν οι σχεδιαστές λογισμικού, τα αποτελέσματα των προσπαθειών τους οδηγούνταν σε μεγάλο ποσοστό σε αδυναμία ολοκλήρωσης ή ακόμα και σε αποτυχία.

Η ανάγκη για ολοκληρωμένα αποτελέσματα οδήγησε στην ανάγκη για τη δημιουργία νέων μεθοδολογιών. Οι επιστήμονες μελέτησαν όλα τα μοντέλα ζωής και κατέληξαν σε μια αρκετά διαφορετική προσέγγιση, η οποία περιλαμβάνει: **επαναληπτική ανάπτυξη** (μικρές εκδόσεις του λογισμικού με ταχύτατους κύκλους ανάπτυξης), **απόλυτη συνεργασία με τον πελάτη** (ο οποίος συμμετέχει ενεργά στην υλοποίηση του έργου) και π**ροσαρμοστικότητα** (δυνατότητα να γίνονται συνέχεια αλλαγές).

Έτσι, το Φλεβάρη του 2001, μια ομάδα επιστημόνων συγκεντρώθηκε στο Lodge, στο χιονοδρομικό κέντρο Snowbird στα βουνά Wasatch της Γιούτα (ΗΠΑ), για φαγητό και σκι, αναζητώντας κοινό έδαφος για την ανάπτυξη λογισμικού μέσω των ευέλικτων μεθοδολογιών που αφορούν κυρίως τα έργα πληροφορικής. Μια πλειάδα μεθόδων έκανε την εμφάνιση της

στη λίστα μεθοδολογιών. Από αυτή τη συνάντηση έγινε η ίδρυση του **Agile Alliance** (Σύμπραξη Επιστημόνων και Φορέων για την Τεκμηρίωση και Προώθηση των Ευέλικτων Μεθοδολογιών), αλλά και του **Agile Manifesto**.

Η λέξη Agile στα ελληνικά μεταφράζεται ως ευέλικτος, άρα Agile θα μπορούσε να είναι και η ικανότητα να κινείται κανείς γρήγορα, άνετα, αεικίνητα, εύστροφα. Το λογισμικό τώρα αναπτύσσεται σε σύντομους επαναληπτικούς και αυξητικούς κύκλους, με δυνατότητα προσαρμογής στις αλλαγές που λάμβαναν χώρα στο διαρκώς μεταβαλλόμενο επιχειρησιακό περιβάλλον. Παράλληλα, αναπτύσσεται η φιλοσοφία του πρόσωπο με πρόσωπο με τον πελάτη, για τον περιορισμό της γραφειοκρατίας. Σαν προσαρμόσιμες μέθοδοι υποδέχονται εύκολα τις αλλαγές, ενώ είναι προσανατολισμένες στους ανθρώπους και όχι στις διαδικασίες υπόσχοντας ποιοτικότερο λογισμικό με ταχύτατη ανάπτυξη.

Οι πρώτες ευέλικτες μεθοδολογίες εμφανίστηκαν τη δεκαετία του 1990, ως εναλλακτική πρόταση έναντι των παραδοσιακών δύσκαμπτων διαδικασιών. Η διαδικασία της **Δομημένης Ανάλυσης και Σχεδίασης (Structured Analysis & Design Technique)** από τον Davis το 1992 έδωσε το εναρκτήριο λάκτισμα. Ακολούθησαν άλλες, κυριότερες Agile μεθοδολογίες, όπως οι: **Extreme Programming (XP)** από τον Beck το 1999, η **SCRUM** (μια από τις πιο διαδεδομένες μαζί με την XP) από τους Schwaber & Beedle το 2002, η **Crystal Family** από τον Cockburn το 2002, η **Feature Driven Development (FDD)** από τους Palmer & Felsing το 2002, η **Dynamic Systems Development Method (DSDM)** από τον Stapleton το 1997, η **Adaptive Software Development (ASD)** από τον Highsmith το 2000, η **Lean Development** από τον Poppendieck το 2001 και η **Διαδικασία της Ενοποιημένης Προσέγγισης (Unified Process)** από τον Kructhen το 2004.

2.7.1. Βασικές αξίες και αρχές

Όπως αναφέρθηκε και λίγο πιο πάνω, η λέξη «Agile» στα ελληνικά μεταφράζεται ως ευέλικτος, άρα Agile θα μπορούσε να είναι και η ικανότητα να κινείται κανείς γρήγορα, άνετα, αεικίνητα, εύστροφα. Το Μανιφέστο των ευέλικτων διαδικασιών αποτελείται από θεμελιώδες αρχές, αξίες και ιδιότητες.

Θα πρέπει να κατανοήσει κανείς ότι, όπως λένε οι εμπνευστές της, οι Agile μεθοδολογίες ΔΕΝ υποστηρίζουν ότι οι διαδικασίες, τα εργαλεία, η λεπτομερής τεκμηρίωση και τα συμβόλαια είναι ασήμαντα και επουσιώδη. Το αντίθετο μάλιστα. Απλώς, δίνουν περισσότερη σημασία, ενώ συνεισφορά έχει και ο πελάτης για το τελικό αποτέλεσμα.

Αρχικά, οι **βασικές τέσσερις θεμελιώδης αξίες (values)** που θεσπίστηκαν μαζί με την διατύπωση του Agile Manifesto είναι οι εξής:

1. **Άνθρωποι και αλληλεπιδράσεις ΑΝΤΙ Διαδικασιών και εργαλείων.** Στην αξία αυτή τονίζεται η σημασία του ανθρώπινου παράγοντα. Προγραμματιστές, μάνατζερ, διευθυντές έργων και πελάτες θα πρέπει να συνεργαστούν μεταξύ τους, αλληλοεπιδρώντας, για την καλύτερη ανάπτυξη του λογισμικού. Πρακτικές που συναντά κανείς εδώ είναι ο προγραμματισμός σε ζεύγη, η συνεχόμενη συμμετοχή του πελάτη στο έργο, η ύπαρξη μικρών ομάδων, κα. Συμπερασματικά, αν η συνεργασία μεταξύ των συμμετεχόντων ΔΕΝ στεφθεί με επιτυχία, η διαδικασία και τα εργαλεία δεν θα αποτελέσουν καμία σημαντική βοήθεια.
2. **Καθαρός κώδικας ΑΝΤΙ Γραπτής τεκμηρίωσης.** Βασικός στόχος της αξίας αυτής είναι να παραδώσει ένα ολοκληρωμένο λογισμικό απαλλαγμένο από τυχόν προβλήματα, ελλείψεις, κλπ. Από την μια δίνεται βαρύτητα στην παράδοση παραδοτέων και τεχνικών εγγράφων, ενώ αυτό μπορεί να αλλάζει συνεχώς, χωρίς να συγχρονίζεται με τις αλλαγές. Από την άλλη, βοηθά στη γνώση και τη διατήρηση πληροφοριών. Συμπερασματικά, θα πρέπει εξαρχής να αναπτύσσεται απλός και ποιοτικός κώδικας που να είναι πιο κατανοητός από τα μακροσκελή έγγραφα και διαγράμματα τεκμηρίωσης, τα οποία διαδραματίζουν δευτερεύοντα και συμπληρωματικό ρόλο.
3. **Συνεργασία με τον πελάτη ΑΝΤΙ Αυστηρών συμβολαίων.** Βασικός στόχος εδώ είναι η ενεργή συμμετοχή του πελάτη στο έργο, αναλαμβάνοντας να παίξει τον πρωταγωνιστικό ρόλο στις αποφάσεις. Η εποχή όπου οι κατασκευαστές ακολουθούσαν μια διαδικασία μακρόσυρτη για να πραγματοποιήσουν αλλαγές μέσω εγγράφων, προδιαγραφών, κλπ., τα οποία θα πρέπει να ακολουθήσουν μια διαδικασία για να υπογραφούν και να πάρουν την έγκριση του πελάτη, έχει περάσει. Γιατί αυτό ενέχει πολλούς κινδύνους, όπως την καθυστέρηση, την κατανάλωση των πόρων και, κυρίως, την αποτυχία του έργου. Τα συμβόλαια δεν είναι μια κακή πρακτική, αφού τίθενται τα όρια, μέσα στα οποία θα πρέπει να κινούνται τα συμβαλλόμενα μέρη. Συμπερασματικά, η στενή συνεργασία με τον πελάτη κρίνεται απαραίτητη, έτσι ώστε στο τέλος να καλύψει σε απόλυτο βαθμό τις ανάγκες του, ακολουθώντας ένα συμβόλαιο, όπου τα συμβαλλόμενα μέρη θα το προσπερνούν για τις ανάγκες τις ολοκλήρωσης του έργου, χτίζοντας σχέση εμπιστοσύνης.
4. **Ανταπόκριση σε αλλαγές ΑΝΤΙ Ακολουθούμενο σχέδιο.** Βασικός στόχος της τελευταίας θεμελιώδους αξίας είναι ότι κατά τη διάρκεια κατασκευής του λογισμικού, θα πρέπει να προσαρμοζόμαστε σε κάθε αλλαγή, η οποία είναι πολύ συχνό φαινόμενο και προκαλείται από διάφορες αιτίες. Από τη μια πλευρά, έχουμε την αδυναμία του πελάτη να εκφράσει το σύνολο των πραγματικών αναγκών πριν την έναρξη του έργου και την απαίτηση για γρήγορη ανάπτυξη και παράδοση του τελικού συστήματος. Από

την άλλη πλευρά, έχουμε ένα λεπτομερές πλάνο, που παραμένοντας προσκολλημένοι σε αυτό, πιθανόν να μην υιοθετούμε αλλαγές για διάφορους λόγους. Συμπερασματικά, εξαρχής θα πρέπει να καθορίζεται ένα πλάνο για την υλοποίηση ενός έργου, ενώ ταυτόχρονα οποιαδήποτε αλλαγή ή απόκλιση θα πρέπει να θεωρείται «φυσιολογική» από τους εμπλεκόμενους κάνοντάς τη αποδεκτή.

Οι «ευέλικτες» μεθοδολογίες δίνουν περισσότερη έμφαση σε αυτά που βρίσκονται στα αριστερά από αυτά που βρίσκονται στα δεξιά.

Παράλληλα με τις θεμελιώδεις αξίες του Agile Manifesto, στην διακήρυξη καθορίστηκαν και οι **12 αρχές (principles)** που καθοδηγούν την ανάπτυξη ενός λογισμικού υιοθετώντας ευέλικτο περιβάλλον. Αυτές είναι οι παρακάτω:

1. Μεγαλύτερη προτεραιότητα είναι η ικανοποίηση του πελάτη μέσω της πρόωρης και συνεχούς παράδοσης λογισμικού
2. Κάθε αλλαγή στις απαιτήσεις είναι καλοδεχούμενη, ακόμα και στα τελικά στάδια ανάπτυξης του λογισμικού, λειτουργώντας προς όφελος του πελάτη.
3. Η συχνή παράδοση λογισμικού κρίνεται απαραίτητη και θα πρέπει να γίνεται σε μικρά χρονικά διαστήματα, πχ σε λίγες εβδομάδες ή μήνες.
4. Ο πελάτης και η ομάδα του έργου θα πρέπει να δουλεύουν μαζί σε καθημερινή βάση καθ’ όλη τη διάρκεια του έργου.
5. Τα στελέχη κατά τη διάρκεια υλοποίησης θα πρέπει να έχουν κίνητρο, πχ μέσω ενός σωστού και οργανωμένου περιβάλλοντος, ενώ ταυτόχρονα θα υποστηρίζονται αν υπάρξει οποιαδήποτε ανάγκη.
6. Καθημερινές συναντήσεις με κατά πρόσωπο συνομιλίες, έτσι ώστε στην ομάδα ανάπτυξης να μεταφέρεται σωστά η πληροφορία.
7. Πρωταρχικό ρόλο διαδραματίζει η σωστή λειτουργία του λογισμικού ως δείκτης εξέλιξης του έργου.
8. Οι ευέλικτες μεθοδολογίες προωθούν μια βιώσιμη διαδικασία ανάπτυξης, έτσι ώστε όλοι οι εμπλεκόμενοι στο έργο να διατηρούν ένα σταθερό ρυθμό καθ’ όλη τη διάρκεια αυτού.
9. Συνεχής προσοχή στην τεχνική αρτιότητα και τον καλό σχεδιασμό με ενισχυμένη ευελιξία.
10. Η απλότητα είναι ουσιώδης.
11. Οι καλύτερες αρχιτεκτονικές, απαιτήσεις και σχέδια προκύπτουν από τις αυτό-διαχειριζόμενες ομάδες.

12. Η ομάδα ανάπτυξης σε τακτά χρονικά διαστήματα θα συζητά τους τρόπους και τις μεθόδους, έτσι ώστε να υπάρχει περισσότερη αποτελεσματικότητα μέσω του επαναπροσδιορισμού της συμπεριφοράς της.

Συμπερασματικά, όπως γίνεται αντιληπτό, για να αντιμετωπίσουμε τις παραπάνω αρχές και αξίες θα πρέπει να υιοθετηθούν κάποιες προϋποθέσεις για να γίνει η υιοθέτησή του. Αρχικά, θα πρέπει ο πελάτης να βρίσκεται στο έργο καθημερινά, έτσι ώστε να λαμβάνει μέρος στην ανάπτυξη του λογισμικού, στις αποφάσεις κα. Τα μέλη της ομάδας ανάπτυξης του λογισμικού θα πρέπει να βρίσκονται κοντά, για να συμμετέχουν μαζί με τον πελάτη στη δημιουργία του λογισμικού και στη λήψη των αποφάσεων. Έτσι, η τεκμηρίωση εδώ δε θα παίζει τον κυρίαρχο ρόλο, ενώ οι απαιτήσεις των χρηστών θα αναδύονται κατά τη διάρκεια ανάπτυξής του. Η ομάδα ανάπτυξης θα αποτελείται από προσωπικό, το οποίο θα έχει μεγάλη και εξειδικευμένη εμπειρία, έτοιμη να ανταποκριθεί σε κάθε αλλαγή που θα προκύπτει, ενώ, παράλληλα, μπορεί να περιορίσει τη διαδικασία αξιολόγησης των παραδοτέων σε μερικές ανεπίσημες ανασκοπήσεις και δοκιμές. Η επαναχρησιμοποίηση ενός μέρους ή ολόκληρου του τελικού λογισμικού δεν είναι αρχικός στόχος, μιας και το λογισμικό αναπτύσσεται επαναληπτικά. Τέλος, το κόστος των αλλαγών δεν αυξάνεται σημαντικά με την πάροδο του χρόνου.

2.7.2. Περιορισμοί και οφέλη με την υιοθέτηση των ευέλικτων μεθοδολογιών

Σύμφωνα με τους Cockburn και Turk (2002), με την υιοθέτηση των ευέλικτων μεθοδολογιών, και όλων των αρχών, αξιών και προϋποθέσεών τους αρχίζουν σιγά - σιγά να εμφανίζονται οι ανασταλτικοί παράγοντες και οι περιορισμοί από την χρήση τους.

Αναλυτικά έχουμε να κάνουμε με τους εξής **περιορισμούς**:

1. **Έλλειψη εξειδικευμένου προσωπικού.** Η έλλειψη εδώ έχει να κάνει με την έλλειψη γνώσεων από το διοικητικό, το οποίο δεν έχει γνώσεις της εφαρμογής των Ευέλικτων μεθοδολογιών.
2. **Άμεση επικοινωνία**. Ένας σημαντικός περιοριστικός παράγοντας είναι η επικοινωνία των μελών μιας ομάδας. Όπως έχουμε αναφέρει, η επικοινωνία μεταξύ των μελών, αλλά και με τον πελάτη, θα πρέπει να είναι άμεση. Και, αν είναι δυνατόν, διαπροσωπική. Αυτό αποτελεί κίνδυνο για τις εταιρίες, στις οποίες οι ομάδες έργων βρίσκονται σε διαφορετικά σημεία, ενώ θα πρέπει να υπάρχει συνεχόμενη επικοινωνία έστω και με σύγχρονα ψηφιακά μέσα.
3. **Διασφάλιση ποιότητας**. Εξίσου σημαντικής σημασίας είναι και η διασφάλιση της ποιότητας ενός λογισμικού. Είναι ένας αμφιλεγόμενος και πολύπλοκος παράγοντας. Γιατί από τη μια έχουμε την πλήρη υιοθέτηση και εκπλήρωση των διαδικασιών των διεθνών προτύπων, με ό, τι αυτό μπορεί να χρειαστεί, ενώ, από την άλλη, οι ευέλικτες

μεθοδολογίες έρχονται σε σύγκρουση με την πλήρη ακολουθία των διεθνών προτύπων που απαιτούν μεγάλη τεκμηρίωση. Επίσης, θα πρέπει να συνυπολογιστεί ότι πολλές εταιρίες έχουν επενδύσει σε τέτοιου είδους συστήματα, που είναι δύσκολο να αγνοηθούν.

4. **Τυποποίηση ή διαφοροποίηση απαιτήσεων των χρηστών.** Μια από τις σημαντικότερες αξίες των ευέλικτων μεθοδολογιών δηλώνει: «**Συνεργασία με τον πελάτη VS Αυστηρών συμβολαίων»,** που εδώ είναι ξεκάθαρο ότι οι νέες απαιτήσεις των χρηστών θα πρέπει να υιοθετούνται άμεσα, παραβλέποντας τις τυποποιημένες διαδικασίες και τα συμβόλαια. Αρά, θα πρέπει να λύνεται άμεσα κάθε παρεξήγηση μεταξύ των συμβαλλόμενων μελών.
5. **Γνώσεις και δεξιότητες ομάδας έργου**. Διαφοροποιούμενο λίγο από τον πρώτο περιορισμό, ο περιορισμός εδώ έχει να κάνει με το προσωπικό, το οποίο θα πρέπει να είναι όχι μόνο τεχνικά καταρτισμένο αλλά και να έχει γνώση και εκπαίδευση όσον αφορά τις ευέλικτες μεθοδολογίες.
6. **Θέματα κόστους**. Είναι γνωστό ότι τα έργα λογισμικού κατά το παρελθόν αντιμετώπιζαν θέματα κόστους εξαιτίας των μοντέλων κύκλου ζωής ανάπτυξης που ακολουθούσαν με αποτέλεσμα την υπερκοστολόγηση, ακόμα και την αποτυχία τους. Η επαναχρησιμοποίηση μερών ή τελικού αποτελέσματος, η αυτοματοποίηση διαδικασιών μπορεί να έχει ως αποτέλεσμα τη μείωση του κόστους.
7. **Μηχανισμός ελέγχου.** Θα πρέπει συνεχώς να αναπτύσσονται διαδικασίες ελέγχων της ομάδας ανάπτυξης, της επικοινωνίας, κλπ., έτσι ώστε οι εκδόσεις αλλά και το τελικό προϊόν να παράγεται με την επιθυμητή ποιότητα.
8. **Ασφάλεια.** Σημαντικός περιορισμός είναι και η εξασφάλιση της ασφάλειας, διότι συνήθως προκύπτουν μεγάλα εμπόδια κατά τη διάρκεια ανάπτυξης ενός λογισμικού.

Από την άλλη πλευρά, με την υιοθέτηση των ευέλικτων μεθοδολογιών έχουμε τα εξής **οφέλη:**

1. **Ταχύτερη ανάπτυξη λογισμικού.** Με την χρήση των ευέλικτων μεθοδολογιών έχουμε την αύξηση ανάπτυξης ενός λογισμικού, ενώ παράλληλα μειώνεται ο χρόνος παράδοσής του στον πελάτη και της έκδοσής του στους τελικούς χρήστες.
2. **Καλύτερη απόδοση της επένδυσης.** Όπως αναφέραμε και παραπάνω, στους περιορισμούς με τα θέματα κόστους η επαναχρησιμοποίηση μερών ή τελικού αποτελέσματος ενός λογισμικού ή η αυτοματοποίηση διαδικασιών μπορεί να έχει ως αποτέλεσμα τη μείωση του κόστους. Αυτό έχει ως αποτέλεσμα την μεγιστοποίηση της απόδοσης της επένδυσης.
3. **Πρόωρη ακύρωση προβληματικών έργων.** Όπως αναφέραμε και στα προηγούμενα κεφάλαια των ευέλικτων μεθοδολογιών, τα κλασσικά μοντέλα κύκλου ζωής πάσχουν

συχνά από την αδυναμία να εκτιμηθεί το μέγεθος του έργου στα αρχικά στάδια και από την εμφάνιση μεγάλων καθυστερήσεων - υπερβάσεων των χρονοδιαγραμμάτων, ακόμα και σε προχωρημένα στάδια της ανάπτυξης των έργων, με αποτέλεσμα ο πελάτης να διαθέσει περισσότερους πόρους, οδηγώντας τον στην ακύρωση του έργου και χάνοντας, έτσι, το σύνολο της επένδυσης του. Από την άλλη πλευρά, οι ευέλικτες μεθοδολογίες, τηρώντας όλες τις αρχές και αξίες του Ευέλικτου Μανιφέστου, δίνουν από νωρίς στον πελάτη την ευκαιρία να αναθεωρήσει το κόστος, το σχεδιασμό και τα οφέλη από το έργο, έτσι ώστε να αποφασίσει τη συνέχειά του ή όχι, με αποτέλεσμα να αντιμετωπίσει λιγότερες ζημιές.

4. **Καλύτερη ποιότητα.** Οι ευέλικτες μεθοδολογίες δίνουν έμφαση στην παραγωγή λογισμικού υψηλής ποιότητας, σε αντίθεση με τα παραδοσιακά μοντέλα κύκλου ζωής. Αυτό επιτυγχάνεται με την αύξηση του βαθμού επίτευξης των απαιτήσεων των χρηστών, μιας και ο πελάτης τώρα συμμετέχει σε όλη τη διάρκεια σχεδιασμού και ανάπτυξης του λογισμικού.
5. **Βελτιωμένος έλεγχος.** Με τις συνεχόμενες εκδόσεις του λογισμικού σε μικρά διαστήματα παρέχεται μια ολοκληρωμένη εικόνα, όσον αφορά τον έλεγχό του, παρά με τις παραδοσιακά μοντέλα, όπου η τεκμηρίωση είναι μεγαλύτερη. Επίσης, αυτό επιτυγχάνεται με την προσβασιμότητα των πληροφοριών από όλα τα συμβαλλόμενα μέλη.
6. **Μείωση της εξάρτησης σε συγκεκριμένα άτομα.** Όπως αναφέρθηκε μόλις παραπάνω, δηλαδή, με την προσβασιμότητα όλων των συμβαλλόμενων μελών σε όλες τις πληροφορίες και σε όλα τα στάδια ανάπτυξης ενός λογισμικού αποφεύγεται η κατάσταση, στην οποία η ανάπτυξη του εξαρτάται σημαντικά από ένα άτομο ή πολλά, με όποια προβλήματα μπορούν να δημιουργηθούν από αυτή την κατάσταση.
7. **Αυξημένη ευελιξία.** Όπως αναφέρθηκε πολύ συχνά στα προηγούμενα κεφάλαια, οι ευέλικτες μεθοδολογίες έχουν την ικανότητα να αλλάζουν σε όλα τα στάδια ανάπτυξης, ικανοποιώντας τις ανάγκες των πελατών δίνοντας τους ευελιξία σε σχέδια με τα παραδοσιακά μοντέλα. Εδώ έχουμε ως αποτέλεσμα να παραδοθεί στον πελάτη ένα λογισμικό που θα ικανοποιεί τις απαιτήσεις του, οι οποίες πολύ πιθανόν δεν είναι ξεκάθαρες εξαρχής, δίνοντάς του την δυνατότητα να αλλάξει σε σχέση με τα παραδοσιακά μοντέλα, που δεν επιτρέπουν τις αλλαγές αυτές κατά τη διάρκεια της ανάπτυξης τους.

Συμπερασματικά, η χρήση των ευέλικτων μεθοδολογιών, των αρχών, αξιών και προϋποθέσεών τους, λαμβάνοντας υπόψιν όλους τους περιορισμούς και τα οφέλη, θα οδηγήσει στην επιτυχία έκδοσης ενός λογισμικού, καλύπτοντας τις απαιτήσεις του κάθε απαιτητικού πελάτη. Αυτό αποδεικνύεται ξεκάθαρα, δηλαδή, μελετώντας κανείς διάφορα επιστημονικά άρθρα – μελέτες

περίπτωσης και εφαρμογής (Adapting Agile Practices in University Contexts, Masood, Z., Hoda, R. & Blincoe, K., 2018 // Challenges of adopting agile methods in a public organization, Nuottila, J., Aaltonen, K. & Kujala, J., 2016 // Critical success factors and barriers for lightweight software process improvement in Agile development: A literature review, Kouzari, E., Gerogiannis, V., Stamelos, I. & Kakarontzas G., 2015 // Identifying some important success factors in adopting agile software development practices, Misra, S. C., Kumar, V. & Kumar U., 2009) βλέπει ξεκάθαρα ότι οι περιορισμοί και τα οφέλη της εφαρμογής των ευέλικτων μεθοδολογιών συμπίπτουν.

2.7.3. Οι «Ευέλικτες» μεθοδολογίες

2.7.3.1. Ακραίος Προγραμματισμός (EXtreme Programming - XP)

Με τον όρο «ακραίος» προγραμματισμός εννοούμε την χρήση των ευέλικτων μεθοδολογιών σε «ακραία» επίπεδα. Η μεθοδολογία του «Ακραίου Προγραμματισμού», ή αλλιώς Extreme Programming (XP), παρουσιάστηκε για πρώτη φορά το 1999 από τον Beck. Η μεθοδολογία αυτή αποτέλεσε καινοτόμα γιατί, ενώ γνώριζαν όλες τις πρακτικές και δεν ήταν η πρώτη φορά που υπήρξαν, τώρα ενώθηκαν όλες μαζί με σκοπό να γίνει μια νέα μεθοδολογία για την ανάπτυξη λογισμικών. Η μεθοδολογία αυτή περιλαμβάνει μια σειρά από αρχές και αξίες που δίνονται εξαρχής στην ομάδα ανάπτυξης, που μέσω αυτών θα έχουμε την κάλυψη όλων των απαιτήσεων σε όλα τα στάδια ανάπτυξης.

Σύμφωνα με τον Beck, οι βασικές αρχές και αξίες του «Ακραίου Προγραμματισμού» είναι οι εξής: (1) **Επικοινωνία:** Ενίσχυση της επικοινωνίας μεταξύ όλων των συμβαλλόμενων μερών με σκοπό τη μείωση των προβλημάτων που δημιουργούνται κατά την υλοποίηση, (2) **Απλότητα:** Με βάση αυτή, θα μπορέσουμε να δουλέψουμε με ευελιξία, ταχύτητα και ευκολία. Αφορά όλες τις διαδικασίες ανάπτυξης ενός λογισμικού, (3) **Ανάδραση:** Σύμφωνα με την συνεχόμενη ανάδραση των εκδόσεων, θα έχουμε την πλήρη κάλυψη των απαιτήσεων του πελάτη, (4) **Θάρρος:** Εδώ αναφερόμαστε στην ικανότητα να εντοπίζονται οι αλλαγές και να λαμβάνονται αποφάσεις σε νέες προκλήσεις που προκύπτουν επιτυχημένα και (5) **Σεβασμός:** Αφορά το κομμάτι μεταξύ των συμβαλλόμενων μελών και τη δυνατότητα για ισότιμη συμμετοχή στην ανάπτυξη του λογισμικού.

Ο κύκλος ζωής ενός έργου, σύμφωνα με τον «Ακραίο Προγραμματισμό» για την ανάπτυξη ενός λογισμικού, αποτελείται από τις εξής φάσεις:

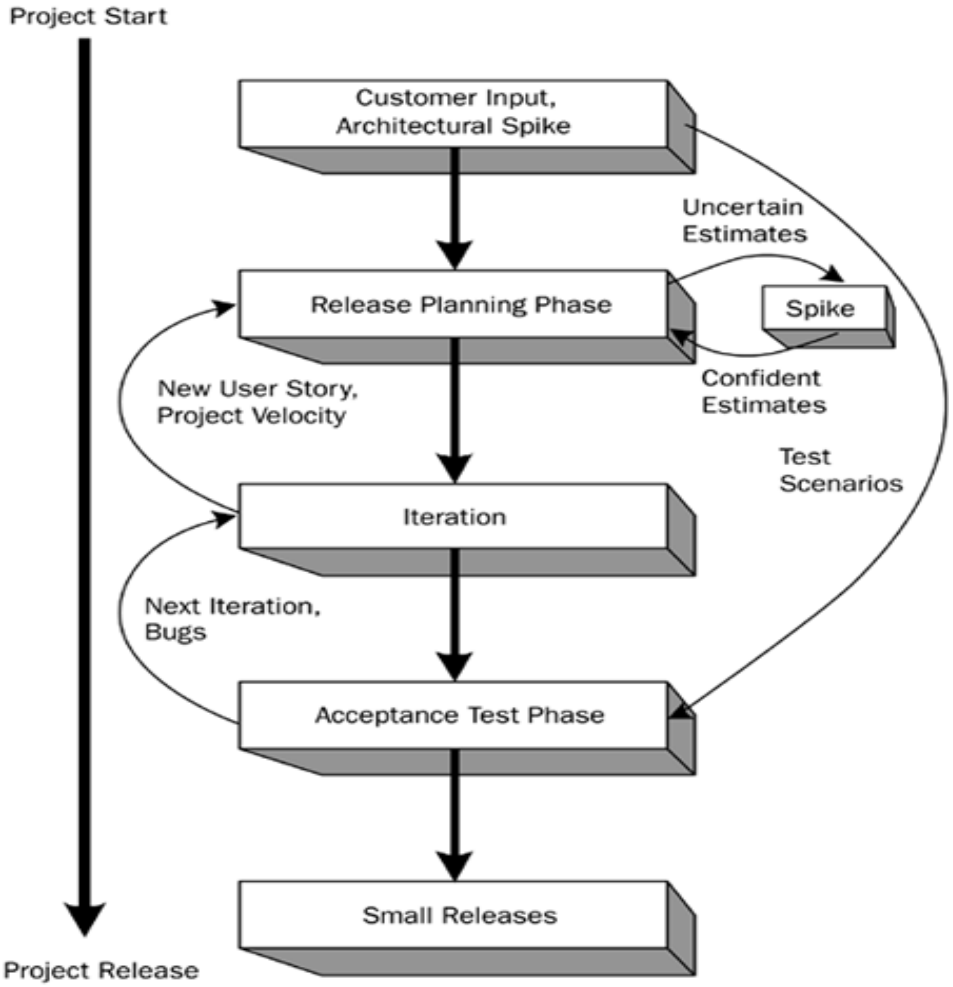
Διερεύνηση: Η φάση της διερεύνησης μπορεί να διαρκέσει από λίγες εβδομάδες έως λίγους μήνες, ενώ εξαρτάται από την ικανότητα της ομάδας να ανταπεξέλθει στις τεχνολογίες ανάπτυξης του λογισμικού. Αρχικά, θα πραγματοποιηθεί η καταγραφή των απαιτήσεων του πελάτη, έτσι ώστε να διαμορφωθεί ένα μοντέλο με επακόλουθο τον σχεδιασμό του

συστήματος. Η εκμαίευση των απαιτήσεων των χρηστών θα γίνεται μέσω των user stories (σε story cards) και των οποίων το αποτέλεσμα θα είναι ορατό με την κάθε έκδοση του λογισμικού. Παρόλο που ο κύκλος ζωής της XP δεν έχει ένα ξεκάθαρο στόχο, με τα user stories αυτός γίνεται εύκολα ορατός. Μετά ακολουθεί η επιλογή των εργαλείων, της τεχνολογίας και των πρακτικών που θα ακολουθηθούν για την έναρξη της υλοποίησης.

Προγραμματισμός: Η φάση του προγραμματισμού διαρκεί μερικές ημέρες. Κατά τη φάση αυτή ιεραρχούνται τα user stories, ενώ, παράλληλα, τα συμβαλλόμενα μέρη (πελάτης και ομάδα ανάπτυξης) συναποφασίζουν την πρώτη και τις επόμενες εκδόσεις του λογισμικού που θα περιλαμβάνουν τα χαρακτηριστικά εκείνα που έχει αποφασίσει ο πελάτης. Για να πραγματοποιηθεί η παραπάνω διαδικασία, θα πρέπει για κάθε user story να αναπτύσσεται η λίστα εργασιών της κάθε ιστορίας, ο χρόνος ανάπτυξης, η ημερομηνία παράδοσης και, τέλος, ποιες ιστορίες θα συμπεριληφθούν στην τελική έκδοση του λογισμικού.

- **Επαναλήψεις:** Η διάρκεια μιας επανάληψης από το σύνολο το επαναλήψεων για την πρώτη έκδοση είναι από μία έως τέσσερις εβδομάδες, με σκοπό στο τέλος, δηλαδή στο τέλος των επαναλήψεων, το λογισμικό να είναι έτοιμο για να εκδοθεί και να τεθεί σε λειτουργία. Η αρχή γίνεται με μια επανάληψη με φάσεις ανάλυσης, σχεδιασμού, προγραμματισμού, δοκιμών και ολοκλήρωσης. Σε κάθε επανάληψη προστίθενται νέα χαρακτηριστικά μέχρι να υλοποιηθεί το τελικό σύστημα. Με το τέλος μερικών επαναλήψεων έχουμε την παράδοση της πρώτης έκδοσης και των υπόλοιπων επαναλήψεων στο τέλος. Με την πρώτη έκδοση μπορούμε να πούμε ότι έχουμε μια πρώτη μορφή του συστήματος με την αρχιτεκτονική του τελικού συστήματος. Η επίτευξη αυτή, όπως και κάθε άλλη, γίνεται, όπως και στη φάση του προγραμματισμού, μέσω των user stories, όπου πάλι εδώ η τελική ευθύνη θα βαραίνει τον πελάτη.
- **Παραγωγή:** Η φάση αυτή περιλαμβάνει επαναλήψεις, οι οποίες έχουν πολύ μικρή χρονική διάρκεια και έχουν ως σκοπό την τελειοποίηση του λογισμικού και όχι τη λειτουργική βελτίωση. Άρα, στη φάση αυτή περιλαμβάνονται όλες οι τεχνικές δοκιμές και οι έλεγχοι, έτσι ώστε το λογισμικό να είναι έτοιμο προς παράδοση στον πελάτη. Τέλος, αν προκύψουν νέες ανάγκες, αν εντοπιστούν προβλήματα, ή αν το σύστημα επιδέχεται βελτιώσεις, τότε επιστρέφουμε στην προηγούμενη φάση και τα νέα χαρακτηριστικά που προκύπτουν θα διατεθούν στην επόμενη έκδοση.
- **Συντήρηση:** Στη φάση αυτή έχουμε τις διορθώσεις σφαλμάτων ή τυχόν προβλημάτων, αλλά και τις προκύπτουσες νέες εκδόσεις του λογισμικού. Στη φάση των νέων εκδόσεων ακολουθείται η διαδικασία των προηγούμενων φάσεων.
- **Απόσυρση:** Στη τελευταία φάση του κύκλου ζωής της XP θα υλοποιηθεί η τεκμηρίωση του λογισμικού. Σε αυτή τη φάση και, αφού ο πελάτης ΔΕΝ έχει άλλα

user stories προς υλοποίηση, μιας και το λογισμικό έχει ικανοποιήσει όλες τις απαιτήσεις, οι αλλαγές, οι οποίες πιθανόν να προκύψουν, δεν γίνονται αποδεκτές. Μπορεί, όμως, σε οποιαδήποτε στιγμή κατά τη διάρκεια της υλοποίησης ή της λειτουργίας να το αποσύρει, αφού δεν ικανοποιούνται οι απαιτήσεις του.



Εικόνα 14 - Ακραίος Προγραμματισμός (EXtreme Programming - XP) - Πηγή: flylib.com

Για την υλοποίηση του λογισμικού θα πρέπει να ορίζονται ρόλοι, έτσι ώστε ο καθένας να έχει την αντίστοιχη υπευθυνότητα. Ο **προγραμματιστής** γράφει κώδικα και κάνει τις δοκιμές με απλό και ακριβή τρόπο, ενώ απαιτείται καλή επικοινωνία με όλα τα υπόλοιπα μέλη. Ο **πελάτης** γράφει τα user stories και τα acceptance tests, τα βάζει σε ιεραρχία, ενώ λαμβάνει τις αποφάσεις για την υλοποίηση. Ο **ελεγκτής**, μέσω των διαφόρων εργαλείων ελέγχου, κάνει τακτικά ελέγχους στο σύστημα, κοινοποιεί τα αποτελέσματα και συντηρεί, ενώ παράλληλα βοηθά τον πελάτη στη συγγραφή των λειτουργικών ελέγχων. Ο **καταγραφέας** έχει σημαντικό ρόλο στη διαδικασία της XP, γιατί καταγράφει τις εκτιμήσεις, δίνει απολογιστικά στοιχεία βοηθώντας

στην βελτίωση της διαδικασίας, ενώ, τέλος, κάνει και την καταγραφή της προόδου υλοποίησης της κάθε επανάληψης και αξιολογεί κατά πόσο είναι εφικτή η επίτευξη του στόχου. Ο **καθοδηγητής** διαδραματίζει τον σημαντικότερο ρόλο στην XP, γιατί είναι υπεύθυνος για την τήρηση και εφαρμογή της μεθόδου σε όλα τα συμβαλλόμενα μέλη και αυτό απαιτεί την άριστη γνώση της. Ο **σύμβουλος** παρέχει την τεχνογνωσία του, όταν καταστεί ανάγκη, και είναι συνήθως ένας εξωτερικός συνεργάτης. Στην κορυφή της πυραμίδας, ο **διευθυντής** είναι εκείνος που λαμβάνει τις αποφάσεις, ενώ είναι εκείνος που επικοινωνεί με όλα τα προηγούμενα μέλη, έτσι ώστε να έχει την πλήρη εικόνα και τον έλεγχο της διαδικασίας.

Τέλος, για την υλοποίηση των φάσεων κατασκευής ενός λογισμικού απαιτείται ένα σύνολο βασικών πρακτικών, όπως παρουσιάστηκαν από τον Beck. Αυτές είναι:

1. **Μικρές εκδόσεις του λογισμικού:** Όταν λέμε μικρές εκδόσεις λογισμικού αναφερόμαστε στη διάρκεια και εδώ συστήνει τις λίγες ημέρες, βδομάδες μέχρι και δυο μήνες. Το λογισμικό εδώ αναπτύσσεται σταδιακά μέσω επαναλήψεων, οι οποίες σαν σύνολο αποτελούν μια έκδοση, η οποία θα πρέπει να γίνεται σε ένα σύντομο χρονικό διάστημα, όπως αναφέρθηκε στην αρχή.
2. **Απλός σχεδιασμός:** Σημαντικό, όπως έχει αναφερθεί σε όλες τις ευέλικτες μεθοδολογίες, είναι και το κομμάτι της απλουστευμένης διαδικασίας. Έτσι και στην XP αποφεύγεται η πολυπλοκότητα στο σχεδιασμό, με τη δυνατότητα απλών λύσεων υλοποίησης. Περιττά τμήματα κώδικα θα πρέπει να αφαιρούνται, ενώ οι νέες λειτουργίες θα πρέπει να ενσωματώνονται εύκολα.
3. **Παιχνίδι προγραμματισμού:** Κατά τη διάρκεια συνάντησης του πελάτη και της ομάδας ανάπτυξης και καθώς διερευνώνται - αναπτύσσονται τα user stories, αποφασίζεται η ιεράρχηση των απαιτήσεων, δηλαδή ποιες ιστορίες θα συμπεριληφθούν σε κάθε επανάληψη ή έκδοση και ποιες από αυτές θα απορριφθούν. Η παραπάνω διαδικασία ονομάζεται «παιχνίδι προγραμματισμού».
4. **Ανάπτυξη καθοδηγούμενη από δοκιμές:** Όπως αναφέρθηκε και στις φάσεις του κύκλου ζωής της XP, για τη σωστή ανάπτυξη του λογισμικού θα πρέπει να ακολουθούνται διαδικασίες ελέγχων και δοκιμών. Η τεχνική που ακολουθείται είναι η δημιουργία μονάδων ελέγχου κώδικα (uni-tests), του κώδικα, δηλαδή, που θα αναπτυχθεί. Στην πραγματικότητα αυτό γίνεται για να καθορίσουμε και να επιβεβαιώσουμε το τι πρόκειται να κάνει το κομμάτι του κώδικα.
5. **Επανακατασκευή:** Στη φάση αυτή, αφού έχουμε κάνει τις αλλαγές σε ένα τμήμα ενός κώδικα, αυτό αναπόφευκτα θα φέρει αλλαγές, επαναλήψεις, ισχυρές εξαρτήσεις, προβλήματα δομής, πολύπλοκες κλάσεις, κλπ. και σε άλλα σημεία του κώδικα. Στόχος

της ομάδας αυτής θα είναι η απλούστευση του τελικού κώδικα, έτσι ώστε να είναι κατανοητός και ευκολότερα επεκτάσιμος.

6. **Προγραμματισμός σε ζεύγη:** Εδώ ο κώδικας γράφεται από δυο προγραμματιστές ταυτόχρονα σε ένα υπολογιστή. Ο ένας εκ των δυο προγραμματιστών είναι εκείνος που συγγράφει, ενώ ο άλλος ταυτόχρονα παρατηρεί και αξιολογεί, ελέγχει, επανακατασκευάζει, κλπ. Οι ρόλοι μεταξύ των δυο προγραμματιστών δεν είναι στατικοί, έχουν μια δυναμικότητα και θα πρέπει συνεχώς να αλλάζουν.
7. **Συνεχής ενσωμάτωση αλλαγών:** Κατά τη διάρκεια της ανάπτυξης ενός λογισμικού, θα υπάρχουν πάντα αλλαγές, προσθήκες, διορθώσεις τμημάτων του κώδικα, οι οποίες θα πρέπει να ενσωματώνονται τουλάχιστον μια φορά την ημέρα. Με την ενσωμάτωση αυτή θα πρέπει να περάσει τη διαδικασία ελέγχου, έτσι ώστε οι αλλαγές να γίνουν αποδεκτές και να είναι πάντα επιτυχείς. Με την διαδικασία αυτή έχουμε πάντα ένα ελεγμένο σύστημα σε οποιαδήποτε χρονική στιγμή.
8. **Συμμετοχή του πελάτη:** Από την έναρξη μιας ανάπτυξης ενός λογισμικού, θα πρέπει να εξασφαλίζεται ότι ο πελάτης, που είναι και ο τελικός χρήστης του λογισμικού, θα πρέπει να είναι κοντά στην ομάδα ανάπτυξης. Στόχος είναι να μην υλοποιείται τίποτα χωρίς την έγκριση του πελάτη και το αναπτυχθέν λογισμικό να καλύπτει όλες του τις απαιτήσεις.
9. **Συλλογική κυριότητα:** Στη φάση αυτή, κάθε μέλος της ομάδας ανάπτυξης, εφόσον είναι ικανό, μπορεί να αλλάξει οποιαδήποτε στιγμή, οποιοδήποτε μέρος του κώδικα. Άρα, δεν υπάρχει κανένα πρόσωπο που να είναι αρμόδιο και υπεύθυνο για ένα κομμάτι κώδικα. Σημαντικό πλεονέκτημα της διαδικασίας αυτής είναι ο εύκολος εντοπισμός λαθών και ασαφειών προχωρώντας άμεσα στην διόρθωση τους.
10. **Συνεχής καταγραφή της εξέλιξης**: Θα πρέπει πάντα με την ολοκλήρωση της ανάπτυξης ενός λογισμικού να υπάρχει και η τεκμηρίωση. Στην τεκμηρίωση θα πρέπει να καταγράφονται όλα τα πραγματικά δεδομένα κατά τη διάρκεια υλοποίησης του έργου, με σκοπό την παραγωγή αναφορών και στατιστικών γραφημάτων, κα.
11. **Πρότυπα κωδικοποίησης:** Εφόσον έχουμε τον προγραμματισμό σε ζεύγη, η ομάδα ανάπτυξης, στην επικοινωνία μεταξύ των μελών της ή στην επικοινωνία της με τον πελάτη, θα πρέπει να ακολουθεί πρότυπα τα οποία θα είναι κοινά για την ανάπτυξη του κώδικα και του τελικού λογισμικού. Στόχος μας είναι να υπάρχει συνέπεια, επαρκής επικοινωνία και αλληλοκατανόηση μεταξύ των μελών.
12. **40 ώρες εργασίες:** Θα πρέπει να υπάρχει ένα όριο ωρών εργασίας κατά τη διάρκεια μιας εβδομάδας, ενώ θα πρέπει να αποφεύγονται τυχόν υπερωρίες. Στόχος είναι ένα ευχάριστο, δημιουργικό και αποδοτικό περιβάλλον εργασίας, αλλά, επίσης, να

παρατηρούνται τυχόν καθυστερήσεις για να εξασφαλιστεί ανάπτυξη χωρίς προβλήματα, έτσι ώστε να μη δημιουργούνται τυχόν προβλήματα στο έργο.

13. **Κοινός χώρος εργασίας:** Ο χώρος, στον οποίο θα αναπτύσσεται το λογισμικό, θα πρέπει να διασφαλίζει άμεση επικοινωνία κατά πρόσωπο, η οποία θα είναι συνεχής και απρόσκοπτη από όλα τα μέλη και, κυρίως, με τον πελάτη.
14. **Συμβολισμός:** Κατά τη διάρκεια της σύναψης της σύμβασης ή του αρχικού σχεδιασμού, θα πρέπει η ομάδα ανάπτυξης του λογισμικού μαζί με τον πελάτη να ορίζουν – περιγράφουν κανόνες – συμβολισμούς, οι οποίοι θα κάνουν ευκολότερη την επικοινωνία τους, αλλά και την κατανόηση του συστήματος.
15. **Απλοί κανόνες:** Οι κανόνες, οι οποίοι ορίζονται εξαρχής από όλα τα συμβαλλόμενα μέλη, θα πρέπει να ακολουθούνται, εφόσον όλοι έχουν συμμετάσχει στη δημιουργία τους. Στην περίπτωση που γίνει αντιληπτό ότι δεν μπορούν να ακολουθηθούν, θα πρέπει να αντικατασταθούν οποιαδήποτε στιγμή, εφόσον αυτό συμφωνηθεί από όλα τα μέλη.

2.7.3.2. Η μεθοδολογία Scrum

Η μεθοδολογία Scrum είναι μια μεθοδολογία ανάπτυξης λογισμικού και αφορά την οργάνωση, την λειτουργία, τον σχεδιασμό και τη διαχείριση ευέλικτων έργων. Είναι μια επαναληπτική και αυξητική μεθοδολογία ανάπτυξης έργων λογισμικού. Η ύπαρξή της οφείλεται σε μια ολιστική προσέγγιση του 1986 από τους Takeuchi και Nonaka. Ο όρος “Scrum” προέρχεται από το ράγκμπι, και σημαίνει μια στρατηγική παιχνιδιού. Το 1995 στο συνέδριο OOPSLA ’95, οι Sutherland και Schwaber όρισαν τα βασικά χαρακτηριστικά της μεθοδολογίας, ενώ σε ένα βιβλίο το 2001, οι Schwaber και Beedle, περιγράφουν αναλυτικά τη μέθοδο.

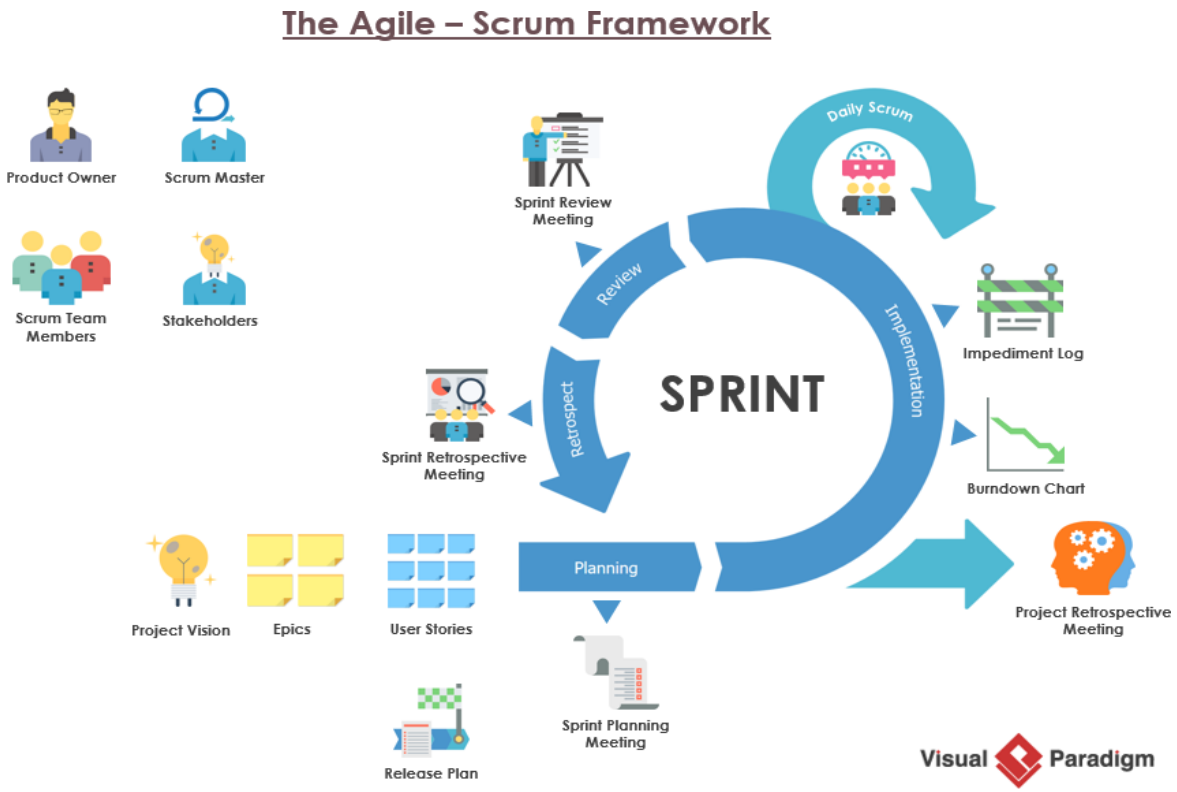
Ειδικά, είναι μια εμπειρική μέθοδος, η οποία αναπτύσσει τις ιδέες της θεωρίας των βιομηχανικών διαδικασιών ελέγχου ανάπτυξης πληροφοριακών συστημάτων. Δίνει έμφαση στο πως η ομάδα έργου θα λειτουργεί σωστά, έτσι ώστε να παράγει το λογισμικό ευέλικτα, σε συνεχή μεταβαλλόμενα περιβάλλοντα, με συνεχή έλεγχο και παρακολούθηση. Άρα, δεν είναι κάποια συγκεκριμένη τεχνική υλοποίησης ενός λογισμικού.

Υπάρχουν τρεις πρωταρχικές έννοιες που στηρίζουν την υλοποίηση της εμπειρικής μεθοδολογίας. Πρώτη σε σειρά έρχεται η **Διαφάνεια**, όπου ορισμένα τμήματα της διαδικασίας είναι ορατά στους υπεύθυνους για το αποτέλεσμα. Απαιτεί από τα τμήματα αυτά να είναι σε ένα κοινό πρότυπο, ώστε αυτοί που τα παρακολουθούν να έχουν μια κοινή άποψη. Ακολουθεί η **Επιθεώρηση**, που θα πρέπει να γίνεται συχνά την εξέλιξη, με σκοπό την ανεύρεση όλων των διαφορών. Ωστόσο, δε θα πρέπει να γίνεται υπερβολικά συχνά για να μην εμποδίζεται η εργασία ανάπτυξης του λογισμικού. Τέλος, η **Προσαρμογή** με το που επισημανθεί ότι ένα ή

περισσότερα στοιχεία της διαδικασίας απέχουν από τα πρότυπα, τότε το λογισμικό είτε θα πρέπει να μη γίνει αποδεκτό ή θα πρέπει να προσαρμοστεί. Η προσαρμογή θα πρέπει να γίνεται όσο το δυνατόν πιο νωρίς για την ελαχιστοποίηση της απόκλισης από τα πρότυπα.

Ο **κύκλος ζωής** ενός έργου, σύμφωνα με την μεθοδολογία «SCRUM» για την ανάπτυξη ενός λογισμικού, αποτελείται από τις εξής φάσεις:

- **The Pre-game phase:** Η φάση αυτή χωρίζεται στον Προγραμματισμό και στον Σχεδιασμό (Αρχιτεκτονική Λογισμικού). Αρχικά, ο **Προγραμματισμός** έχει ως σκοπό την ανάπτυξη του συστήματος, δημιουργώντας αρχικά μια Λίστα Εκκρεμοτήτων Προϊόντος (Product Backlog List) συμπεριλαμβάνοντας όλες τις απαιτήσεις του υπό ανάπτυξη συστήματος. Οι απαιτήσεις, όπως έχουμε πει, καθορίζονται από τον πελάτη και από την ομάδα ανάπτυξης. Αμέσως μετά ιεραρχούνται και, κατά την ανάπτυξη τους, συνεχώς ανανεώνονται και επικαιροποιούνται σύμφωνα με τις νέες ανάγκες. Επίσης, στον Προγραμματισμό, καθορίζεται και η ομάδα ανάπτυξης, τα εργαλεία που θα χρησιμοποιηθούν, η κατανομή των πόρων, η εκτίμηση των κινδύνων και ο τρόπος ελέγχου. Σε κάθε επανάληψη (iteration) η αναθεωρημένη Backlog List γίνεται γνωστή στην ομάδα με σκοπό να εφαρμοστεί από όλους. Στην φάση της **Αρχιτεκτονικής** γίνεται ο πλήρης σχεδιασμός του Λογισμικού, με βάση το Backlog List που έχει δημιουργηθεί νωρίτερα, έτσι ώστε να δημιουργηθεί ένα αρχικό πλάνο για το τι θα περιλαμβάνει τελικά το σύστημα. Στις περιπτώσεις ενός υπάρχοντος συστήματος, οι αλλαγές που παρατηρούνται, καταγράφονται και αυτές στο Backlog List.
- **The Development phase**: Με την φάση αυτή έχουμε την εφαρμογή του «Ευέλικτου» κομματιού στη μεθοδολογία της Scrum. Οι διάφορες τεχνικές και εξωτερικού περιβάλλοντος παράμετροι μπορούν να διαφοροποιηθούν κατά την υλοποίηση, ενώ παρακολουθούνται και ελέγχονται συνεχώς μέσω διάφορων πρακτικών κατά τη διάρκεια των Sprints. Όπου Sprints είναι οι κύκλοι επανάληψης κατά τη διάρκεια ανάπτυξης ενός λογισμικού στο οποίο υλοποιούνται οι λειτουργίες του συστήματος. Κάθε Sprint περιλαμβάνει την καταγραφή των απαιτήσεων, την ανάλυση, τον σχεδιασμό, την ανάπτυξη και, τέλος, την παράδοσή.
- **The Post-game phase**: Με την φάση αυτή έχουμε το κλείσιμο της συγκεκριμένης έκδοσης του λογισμικού. Η έναρξή της πραγματοποιείται με την κοινή απόφαση όλων των συμβαλλόμενων μελών ότι όλες οι απαιτήσεις έχουν ολοκληρωθεί. Εδώ το σύστημα έχει εκδοθεί και έχει τεθεί σε πλήρη λειτουργία, ενώ κάθε νέα απαίτηση δεν είναι δυνατόν να προστεθεί. Σε αυτή τη φάση έχουμε την προετοιμασία του λογισμικού για πλήρη λειτουργία τους τελικούς ελέγχους και την τεκμηρίωση του συστήματος.



Εικόνα 15 - Η μεθοδολογία Scrum - Πηγή: visual-paradigm.com

Για την υλοποίηση του λογισμικού θα πρέπει να ορίζονται **ρόλοι,** έτσι ώστε ο καθένας να έχει την αντίστοιχη υπευθυνότητα. Ο **Scum Master** είναι υπεύθυνος για να πιστοποιήσει αν το έργο υλοποιείται σύμφωνα με τις αξίες, τις πρακτικές και τους κανόνες της Scrum. Συνεργάζεται με όλα τα τμήματα του έργου λειτουργώντας ως συνδετικός κρίκος, ενώ είναι ικανός να διασφαλίσει την παραγωγικότητα του έργου. Ο **Ιδιοκτήτης του έργου** είναι εκείνος που καθορίζεται από τον Scrum Master, τον Πελάτη και τη Διοίκηση, λαμβάνοντας τις τελικές αποφάσεις, μιας και αρμοδιότητές του είναι η σωστή διαχείριση και ο έλεγχος του έργου. Η **ομάδα Scrum**, που κατά τη διάρκεια ανάπτυξης ενός λογισμικού μπορεί και να είναι περισσότερες από μια, δουλεύοντας παράλληλα, έχει ως αρμοδιότητα να αποφασίζει για όλες εκείνες τις ενέργειες που πρέπει να υλοποιηθούν με σκοπό την επίτευξη των στόχων του κάθε Sprint. Ο **Πελάτης** συμμετέχει ενεργά στη δημιουργία του Backlog List και στην ανασκόπησή της. Τέλος, η **Διοίκηση** είναι εκείνη η οποία θα είναι υπεύθυνη για τον καθορισμό των τελικών αποφάσεων, των στόχων και των απαιτήσεων του λογισμικού.

Τέλος, για την υλοποίηση των φάσεων κατασκευής ενός λογισμικού απαιτείται ένα σύνολο βασικών πρακτικών. Όπως αναφέραμε παραπάνω, η Scrum είναι μια εμπειρική μεθοδολογία χωρίς μια συγκεκριμένη μέθοδο ανάπτυξης. Για να έχουμε σωστή ανάπτυξη απαιτείται η χρήση συγκεκριμένων πρακτικών και εργαλείων που έχουν ως σκοπό την αποφυγή δημιουργίας

πολυπλοκότητας και αδυναμίας πρόβλεψης των επιπτώσεων των νέων απαιτήσεων. Αυτές είναι:

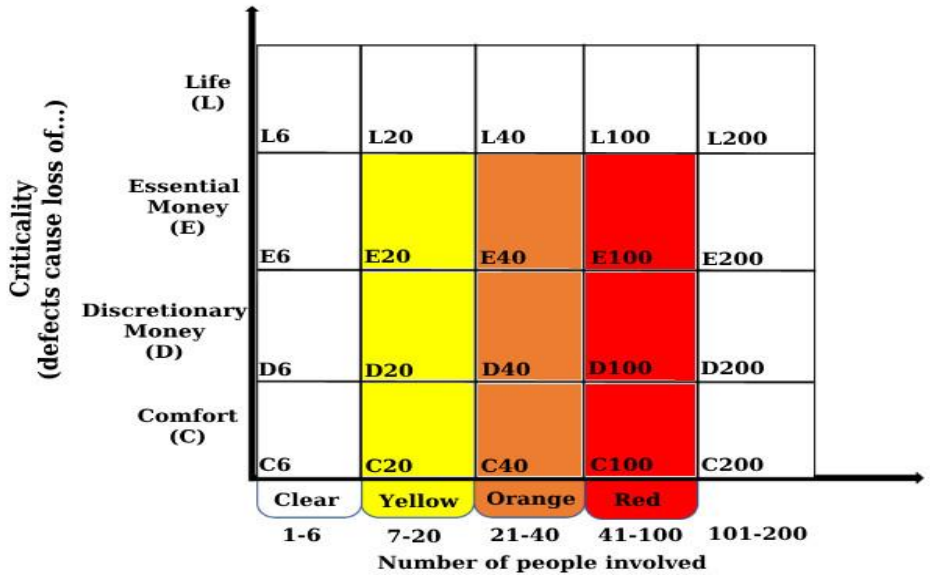
1. **Product Backlog:** Είναι μια λίστα, η οποία ενημερώνεται συνεχώς, και περιγράφει όλες τις τεχνικές και λειτουργικές απαιτήσεις για την ανάπτυξη ενός λογισμικού και στην οποία πραγματοποιείται η ιεράρχηση, έτσι ώστε να καθορίσει όλα τα χαρακτηριστικά που είναι απαραίτητα για το τελικό προϊόν. Υπεύθυνος για τη συντήρηση του Product Backlog είναι ο Ιδιοκτήτης του έργου.
2. **Effort Estimation:** Υπεύθυνος για την υλοποίηση της διαδικασίας είναι ο Ιδιοκτήτης του έργου μαζί με την ομάδα Scrum. Η εκτίμηση της προσπάθειας του ανθρώπινου δυναμικού είναι μια επαναληπτική διαδικασία του Product Backlog, ενώ αναλύεται λεπτομερώς το κάθε επί μέρους στοιχείο, όταν οι απαραίτητες πληροφορίες είναι διαθέσιμες.
3. **Sprint**: Ένα Sprint διαρκεί περίπου 28 ημέρες. Εδώ η ομάδα Scrum έχει ως σκοπό να υλοποιήσει ένα κομμάτι του τελικού συστήματος μέσα σε ένα Sprint. Πρόκειται για το κομμάτι του τελικού συστήματος, αφού έχει ολοκληρωθεί η Product Backlog List και έχει οργανωθεί η Συνάντηση Προγραμματισμού των Sprints, όπου ιεραρχείται η Product Backlog List υλοποιώντας το μέσα σε ένα Sprint.
4. **Sprint Planning Meeting**: Σε αυτή τη φάση, όλα τα συμβαλλόμενα μέρη του έργου συναντώνται με σκοπό να συναποφασίσουν αναλυτικά τις απαιτήσεις του μέρους του συστήματος, το οποίο θα υλοποιηθεί στο επόμενο Sprint. Αμέσως μετά, ο Scrum Master μαζί με την ομάδα του θα εστιάσει ακριβώς στο πως θα υλοποιηθούν οι απαιτήσεις κατά τη διάρκεια του συγκεκριμένου Sprint.
5. **Sprint Backlog**: Αποτελεί το σημείο εκκίνησης κάθε Sprint, ενώ παραμένει αμετάβλητη καθ’ όλη τη διάρκειά του. Αποτελεί ένα υποσύνολο περιεχομένων της Product Backlog List, με όλες τις απαιτήσεις, που θα υλοποιηθούν κατά τη διάρκεια του συγκεκριμένου Sprint, του Sprint Planning Meeting.
6. **Daily Scrum Meeting**: Οι καθημερινές συναντήσεις θα πρέπει να είναι μικρής διάρκειας των 15 περίπου λεπτών, στις οποίες μπορεί να συμμετέχει από τον Scrum Master και την ομάδα του μέχρι και η Διοίκηση, ο πελάτης και ο Ιδιοκτήτης του έργου. Σκοπός τους είναι ο συνεχής έλεγχος της πορείας εξέλιξης του έργου, ο καλύτερος προγραμματισμός και ο εντοπισμός και η καταγραφή των προβλημάτων.
7. **Sprint Review Meeting**: Πραγματοποιείται με το τέλος του κάθε Sprint, όπου ο Scrum Master με την ομάδα του συναντώνται με όλα τα υπόλοιπα συμβαλλόμενα μέλη του έργου και παρουσιάζουν τα αναλυτικά αποτελέσματα του Sprint που μόλις ολοκληρώθηκε, έτσι ώστε να αποφασιστούν οι επόμενες ενέργειες.

2.7.3.3. Μεθοδολογίες Crystal Family

Η Crystal Family αποτελεί μια οικογένεια μεθοδολογιών και αναφέρεται σε διαφορετικούς τύπους έργων, όπου επιλέγεται η πιο κατάλληλη μέθοδος περιλαμβάνοντας αρχές για να μπορέσει να προσαρμόζεται εύκολα, ανάλογα με τις απαιτήσεις του έργου. Πατέρας των «Κρυστάλλινων μεθοδολογιών» είναι ο Cockburn, ο οποίος μιλά για πρώτη φορά για αυτές τις μεθοδολογίες το 2002.

Η ονομασία του κρυστάλλου προέρχεται από το χαρακτηρισμό των έργων ως προς το μέγεθος και την κρισιμότητα, αντιστοιχώντας τη με εκείνη των ορυκτών, το χρώμα και τη σκληρότητα. Δηλαδή, κάθε μέλος της οικογένειας των Κρυστάλλινων μεθοδολογιών είναι χαρακτηρισμένο με ένα χρώμα, δείχνοντας πόσο εντατικά εφαρμόζεται η μεθοδολογία, αλλά και το βαθμό δυσκολίας του έργου. Τα μεγαλύτερα έργα, τα οποία έχουν περισσότερες απαιτήσεις, χαρακτηρίζονται με σκουρότερα χρώματα.

Η κρισιμότητα ενός έργου καθορίζεται από τις επιπτώσεις που θα έχει η αποτυχία λειτουργίας ενός συστήματος. Σύμφωνα με τον Cockburn, ο βαθμός κρισιμότητας χαρακτηρίζεται από τους χαρακτήρες: C, D, E και L. Όπου **C**: Comfort – Άνεση, **D**: Discretionary Money – Προαιρετικά Χρήματα, **E**: Essential Money – Ουσιαστικά Χρήματα και **L**: Life – Ζωή. Πρακτικά θα μπορούσαμε να πούμε ότι ο C μαρτυρά την κατάρρευση του συστήματος, έτσι ώστε να χαθεί η άνεση του χρήστη να το χρησιμοποιήσει, ενώ, αντίθετα, με το βαθμό L μπορεί με την κατάρρευση του συστήματος να έχουμε και την ανθρώπινη απώλεια του χρήστη!



Εικόνα 16 - Μεθοδολογίες Crystal Family - Πηγή: hangoutagile.com

Μερικά παραδείγματα μεθοδολογιών είναι: Crystal Clear (η πιο διαδεδομένη), Crystal Yellow, Crystal Orange, Crystal Red, Crystal Violet, Crystal Sapphire, Crystal Diamond, ξεκινώντας από αυτές που εφαρμόζονται στα πιο μικρά έργα μέχρι στα πιο μεγάλα.

Παρά την πλήρη κατηγοριοποίηση τους, οι μεθοδολογίες αυτές έχουν ορισμένους κανόνες, χαρακτηριστικά και αξίες που είναι κοινές σε όλες. Εδώ το έργο υλοποιείται σε επαναληπτικούς κύκλους ανάπτυξης (incremental development cycles), με μέγιστη διάρκεια επανάληψης τους τέσσερις μήνες, με προτιμητέα διάρκεια τον έναν έως τρεις μήνες.

Οι μεθοδολογίες αυτές ΔΕΝ υποστηρίζουν κατανεμημένη ανάπτυξη, γι' αυτό και οι ομάδες ανάπτυξης θα πρέπει να βρίσκονται πολύ κοντά. Επίσης, δε διευκρινίζεται εξαρχής το στυλ του κώδικα, η επιχειρηματική κουλτούρα και το επιθυμητό περιβάλλον τεχνολογίας (Qumer & Henderson - Sellers, 2008). Τέλος, δεν υπάρχει δέσμευση ως προς τις πρακτικές ανάπτυξης και τα εργαλεία που θα χρησιμοποιηθούν, επιτρέποντας την παράλληλη ανάπτυξη και άλλων μεθοδολογιών, όπως της XP, Scrum, κα. (Abrahamsson, 2003).

Η **Διαδικασία ανάπτυξης** των «Κρυστάλλινων» μεθοδολογιών περιλαμβάνει οδηγίες για τα πρότυπα πολιτικής, τα παραδοτέα, τα τοπικά θέματα, τα εργαλεία, τα πρότυπα και τους ρόλους που πρέπει να ακολουθηθούν κατά τη διάρκεια ανάπτυξης ενός λογισμικού.

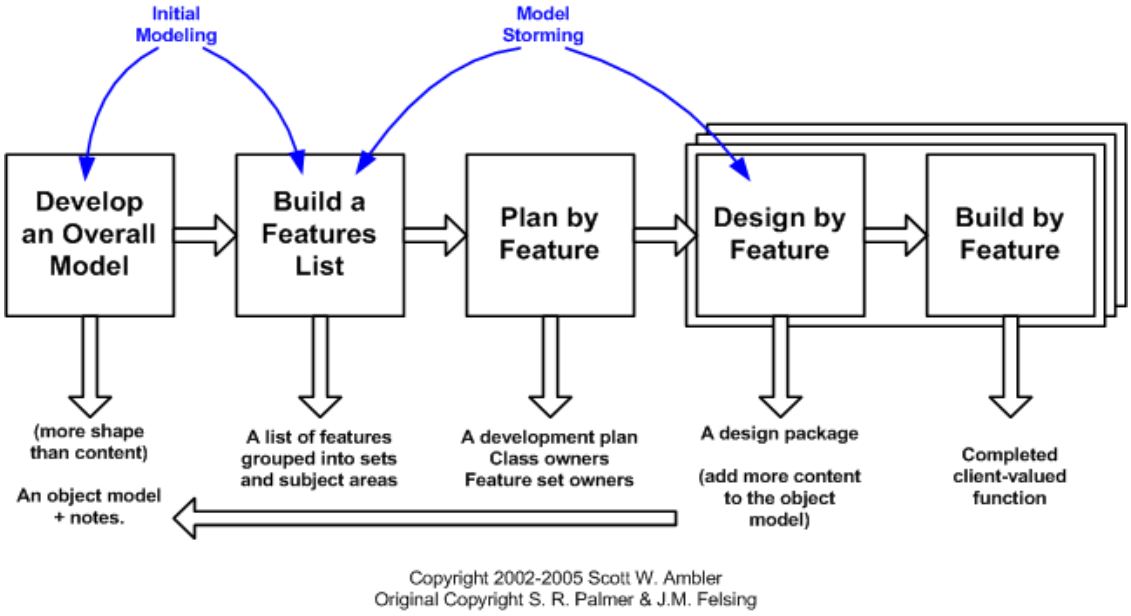
Για την υλοποίηση του λογισμικού θα πρέπει να ορίζονται **ρόλοι,** έτσι ώστε ο καθένας να έχει την αντίστοιχη υπευθυνότητα. Στην Crystal Clear οι βασικοί ρόλοι που υλοποιούν το έργο είναι: ο χορηγός, ο έμπειρος προγραμματιστής – σχεδιαστής, ο προγραμματιστής – σχεδιαστής και ο χρήστης. Οι ρόλοι αυτοί μπορεί να περιέχουν πολλαπλούς υπό-ρόλους. Αντίθετα, στην Crystal Orange, όπου υπάρχουν περισσότερες ομάδες, μπορούμε να έχουμε τους παραπάνω ρόλους με επιπρόσθετους αυτούς που ομαδοποιούνται σε αυτούς του σχεδιαστή συστήματος (system planning), αρχιτεκτονικής (architecture), τεχνολογίας (technology), λειτουργικοτήτων (functions), υποδομών (infrastructure) και εξωτερικών ελεγκτών (external test).

Τέλος, για την υλοποίηση των φάσεων κατασκευής ενός λογισμικού απαιτείται ένα σύνολο **βασικών πρακτικών - τεχνικών**. Αυτές είναι: (1) **Η Σκηνοθεσία (Staging)**, αφορά τον προγραμματισμό της υλοποίησης της κάθε υπό-έκδοσης του τελικού συστήματος, (2) **Η Αναθεώρηση & Ανασκόπηση (Revision & Review)**, όπου σε κάθε υπό-έκδοση έχουμε πολλές επαναλήψεις με ενέργειες, όπως: κατασκευή, επίδειξη και ανασκόπηση των στοιχείων, (3) **Η Παρακολούθηση (Monitoring)**, η εξέλιξη της υλοποίησης γίνεται σύμφωνα με τα παραδοτέα σε σχέση με την εξέλιξη και τη σταθερότητά τους, (4) **Η Στρατηγική Ολιστικής Ποικιλίας (Holistic Diversity Strategy)**, χρησιμοποιείται η στρατηγική αυτή για να χωρίσει τις μεγάλες ομάδες σε μικρότερες δια-λειτουργικές, με σκοπό όλες οι ομάδες να έχουν την απαραίτητη τεχνογνωσία σε πολλαπλές ειδικότητες, (5) **Η Ρύθμιση Μεθοδολογίας**

(Methodology – tuning), όπου μέσω των συνεντεύξεων και συναντήσεων εργασίας με τα στελέχη της ομάδας έργου επιχειρείται η «Ρύθμιση» της μεθοδολογίας, (6) **Οι Έλεγχοι Χρηστών (User Viewings)**, όπου στα μικρά έργα οι έλεγχοι θα γίνονται μια φορά στο τέλος από δυο χρήστες, ενώ στα μεγαλύτερα έργα οι έλεγχοι θα γίνονται τρεις φορές και (7) **Η Συνάντηση Απεικόνισης** (Reflection Workshops), όπου η ομάδα έργου πρέπει να οργανώνει συναντήσεις απεικόνισης, τόσο πριν όσο και μετά την ολοκλήρωση μιας υπό-έκδοσης.

2.7.3.4. Ανάπτυξη Βάσει Χαρακτηριστικών (Feature Driven Development - FDD)

Η μεθοδολογία μέσω Ανάπτυξης με βάσει τα Χαρακτηριστικά (**Feature Driven Development - FDD**) αποτελεί μια ελαφριά διαδικασία ανάπτυξης λογισμικού καθοδηγούμενη από μοντέλα, προσανατολισμένη προς τα συχνά, απτά και λειτουργικά αποτελέσματα. Τα χαρακτηριστικά αυτά καθορίζουν την μεθοδολογία ως ευέλικτη. Η μεθοδολογία αυτή αναφέρθηκε για πρώτη φορά από τον Peter Coad, το 2000, και αναπτύχθηκε περεταίρω στο βιβλίο των Palmer και Felsing, το 2002. Η FDD επικεντρώνεται στον σχεδιασμό και τον προγραμματισμό, διαφοροποιώντας την από τις άλλες μεθοδολογίες ανάπτυξης λογισμικών (Qumer & Henderson - Sellers, 2008). Συνδυάζοντας την ευέλικτη ανάπτυξη και την ανάπτυξη μέσω μοντέλων δίνουμε έμφαση στο αρχικό μοντέλο, τον καταμερισμό του έργου σε χαρακτηριστικά και τον σχεδιασμό κάθε χαρακτηριστικού σε επαναλήψεις, όπου η κάθε επανάληψη αποτελείται από τον σχεδιασμό και την ανάπτυξη. Έτσι, η μέθοδος αυτή κρίνεται ως κατάλληλη για την ανάπτυξη κρίσιμων συστημάτων σε σχέση με άλλες ευέλικτες μεθοδολογίες.



Εικόνα 17 - Ανάπτυξη Βάσει Χαρακτηριστικών (Feature Driven Development - FDD) - Πηγή: agilemodeling.com

Η **Διαδικασία Ανάπτυξης** της FDD αποτελείται από πέντε συνεχόμενες διαδικασίες, κατά τη διάρκεια των οποίων εκτελείται ο σχεδιασμός και η ανάπτυξη του συστήματος, όπως αναφέρθηκε παραπάνω. Η επαναληψιμότητα αυτή υποστηρίζει τους μηχανισμούς της ευέλικτης ανάπτυξης με ταχύτατη προσαρμογή σε αλλαγές της τελευταίας στιγμής. Κάθε επανάληψη έχει διάρκεια από μια έως τρεις εβδομάδες. Οι διαδικασίες αυτές είναι: η Ανάπτυξη ενός συνολικού μοντέλου (Develop an Overall Model), η Κατασκευή Λίστας Χαρακτηριστικών (Build a Features List), ο Προγραμματισμός βάσει Χαρακτηριστικών (Plan by Features), ο Σχεδιασμός και η Κατασκευή βάσει Χαρακτηριστικών (Design by Feature and Build by Feature).

Για την υλοποίηση του λογισμικού θα πρέπει να ορίζονται **ρόλοι,** έτσι ώστε ο καθένας να έχει την αντίστοιχη υπευθυνότητα. Η FDD χωρίζει τους ρόλους σε τρεις βασικές κατηγορίες:

- **Οι Βασικοί Ρόλοι (Key Roles):** ο υπεύθυνος έργου, ο υπεύθυνος αρχιτεκτονικής, ο υπεύθυνος ανάπτυξης, ο κύριος προγραμματιστής, ο ιδιοκτήτης κλάσης και οι ειδικοί πεδίου.
- **Οι Υποστηρικτικοί Ρόλοι (Supporting Roles):** ο υπεύθυνος έκδοσης, ο ειδικός στη γλώσσα προγραμματισμού, ο μηχανικός κατασκευής, ο κατασκευαστής εργαλείων και ο διαχειριστής του συστήματος.
- **Οι Επιπρόσθετοι Ρόλοι (Additional Roles):** οι ελεγκτές, οι προγραμματιστές και οι συντάκτες τεχνικών κειμένων.

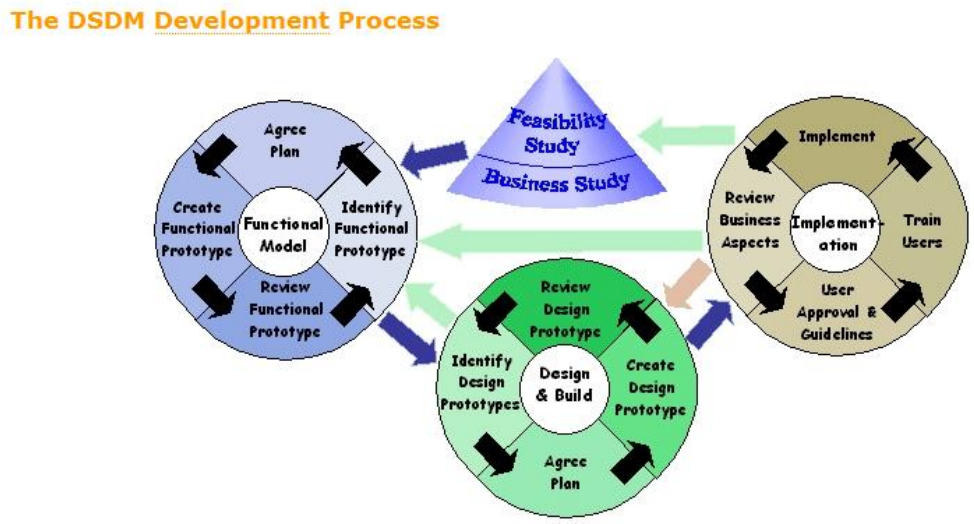
Κάθε μέλος της ομάδας έργου μπορεί να αναλαμβάνει πολλαπλούς ρόλους, ενώ ένας ρόλος μπορεί να μοιραστεί σε πολλά άτομα.

Τέλος, για την υλοποίηση των φάσεων κατασκευής ενός λογισμικού, απαιτείται ένα σύνολο **«καλών» πρακτικών και τεχνικών**, οι οποίες παρόλο που δεν είναι νέες, ο συγκεκριμένος συνδυασμός τους κάνει την FDD μοναδική, με μόνο σκοπό τη μεγιστοποίηση της ωφέλειας από την χρήση της μεθόδου, μιας και κάθε πρακτική ξεχωριστά από μόνη της δε μπορεί να διασφαλίσει κάτι τέτοιο. Η FDD περιλαμβάνει τις ακόλουθες πρακτικές – τεχνικές: Μοντελοποίηση Αντικειμένου Πεδίου (Domain Object Modeling), Ανάπτυξη βάσει Χαρακτηριστικών (Developing by Feature), Κυριότητα Κλάσης (Individual Class Code Ownership), Ομάδες Ανάπτυξης Χαρακτηριστικών (Feature Teams), Επιθεώρηση (Inspection), Τακτικές Ενσωματώσεις (Regular Builds), Διαχείριση Δομής / Διάρθρωσης (Configuration Management) και Αναφορές Προόδου Εξέλιξης (Progress Reporting).

2.7.3.5. Μέθοδος Ανάπτυξης Δυναμικών Συστημάτων (Dynamic Systems Development Method - DSDM)

Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, «*στις αρχές της δεκαετίας του 1990, ο James Martin εισήγαγε την* ***Ταχεία Ανάπτυξη Εφαρμογής*** *(Rapid Application Development – RAD), που βασίζεται στην εξέλιξη και βελτίωση του* ***Εξελικτικού Μοντέλου Παράδοσης (Evolutionary Delivery Model)****, ενώ οι κύριοι στόχοι της συγκεκριμένης μεθοδολογίας είναι: η υψηλή ταχύτητα, η υψηλή ποιότητα και το χαμηλό κόστος. Χαρακτηριστικό γεγονός είναι ότι επιβάλλεται η ενεργή συμμετοχή του πελάτη αλλά και όλων των εμπλεκομένων. Ο έλεγχος του συστήματος γίνεται επαναληπτικά τόσο από την ομάδα ανάπτυξης όσο και από τους χρήστες*». Παρά το γεγονός ότι η μέθοδος αυτή έφερε επανάσταση, αναπτύχθηκε άναρχα και χωρίς συγκεκριμένη δομή. Με στόχο τη δημιουργία ενός κοινά αποδεκτού πλαισίου εργασίας γεννήθηκε η DSDM, το 1994, στη Μεγάλη Βρετανία. Η επίσημη δημοσίευση της πρώτης έκδοσης έγινε το 1995, ενώ ακολούθησαν άλλες τρεις εκδόσεις, αρχίζοντας σιγά σιγά να υιοθετείται τόσο στον ιδιωτικό, όσο και στο δημόσιο τομέα στο Ηνωμένο Βασίλειο, ξεπερνώντας τα σύνορα της χώρας και καθιστώντας την μια από της σημαντικότερες μεθοδολογίες ανάπτυξης πληροφοριακών συστημάτων παγκοσμίως.

Η βασική ιδέα της DSDM είναι ότι, αντί να καθοριστούν η επιθυμητή λειτουργικότητα σε ένα προϊόν και οι πόροι για την υλοποίηση, είναι προτιμότερο να καθοριστούν αρχικά ο χρόνος και οι πόροι και, στη συνέχεια, να καθοριστεί ανάλογα η επιθυμητή λειτουργικότητα.



Εικόνα 18 - Μέθοδος Ανάπτυξης Δυναμικών Συστημάτων (Dynamic Systems Development Method - DSDM) - Πηγή: dsdmofagilemethodology.wdfiles.com

Η βασική φιλοσοφία της DSDM λέει ότι: Η ανάπτυξη του λογισμικού είναι μια συλλογική διαδικασία, η ιεράρχηση των απαιτήσεων του έργου είναι αναγκαία, η ανάπτυξη λογισμικού

μπορεί και πρέπει να είναι μια σταδιακή εμπειρία και η υψηλή τεχνογνωσία των τεχνικών, που εμπλέκονται στην ανάπτυξη του έργου, είναι βασική προϋπόθεση για το άριστο αποτέλεσμα του παραγόμενου αποτελέσματος.

Η **Διαδικασία Ανάπτυξης** της DSDM αποτελείται από πέντε φάσεις: τη Μελέτη Σκοπιμότητας (Feasibility Study), την Επιχειρησιακή Μελέτη (Business Study), την Επανάληψη Λειτουργικού Μοντέλου (Functional Model Iteration), την Επανάληψη Σχεδιασμού & Κατασκευής (Design & Build Iteration) και την Υλοποίηση (Implementation).

Για την υλοποίηση του λογισμικού θα πρέπει να ορίζονται **ρόλοι,** έτσι ώστε ο καθένας να έχει την αντίστοιχη υπευθυνότητα. Στη DSDM έχουμε 15 ρόλους χρηστών και προγραμματιστών. Οι κυριότεροι εξ' αυτών είναι οι έξης, σύμφωνα με τον Stapleton (1997):

- Προγραμματιστές και Έμπειροι Προγραμματιστές (Developers and Senior Developers)
- Τεχνικός Συντονιστής (Technical Coordinator)
- Χρήστης Πρεσβευτής (Ambassador User)
- Χρήστης Μέντορας (Adviser User)
- Οραματιστής (Visionary)
- Επιτελικός Χορηγός (Executive Sponsor)

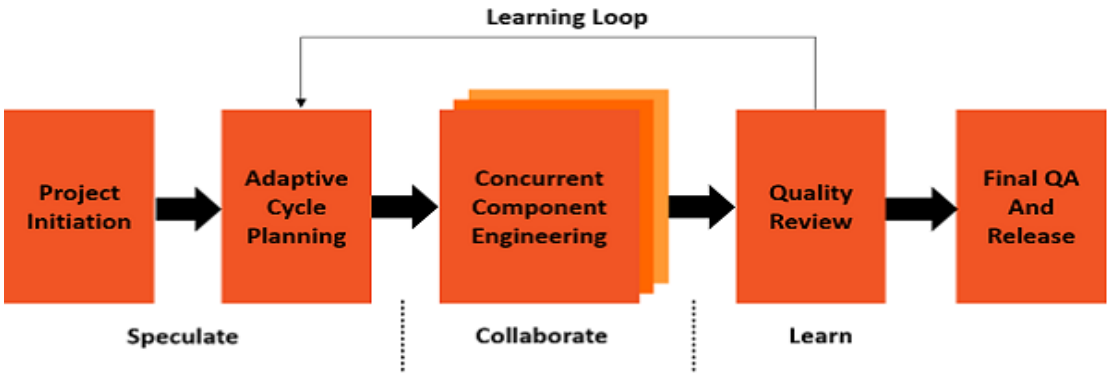
Τέλος, για την υλοποίηση των φάσεων κατασκευής ενός λογισμικού απαιτείται ένα σύνολο **βασικών πρακτικών - αρχών**. Οι εννέα αρχές που διέπουν την DSDM είναι: η συμμετοχή των χρηστών, η ενδυνάμωση της ομάδας έργου, η συχνή παράδοση, η αντιμετώπιση των σύγχρονων αναγκών της επιχείρησης, η επαναληπτική και σταδιακή ανάπτυξη, η δυνατότητα για ανατρεπτικές αλλαγές, ο καθορισμός του πλαισίου πριν από την έναρξη του έργου, ο έλεγχος καθ' όλη τη διάρκεια του έργου και τέλος η αποτελεσματική και αποδοτική επικοινωνία (Dyba & Dingsøyr, 2008).

2.7.3.6. Προσαρμοστική Ανάπτυξη Λογισμικού (Adaptive Software Development - ASD)

Η μέθοδος της ASD είναι μια αντικειμενοστραφής προσέγγιση, η οποία διευκολύνει την επαναλαμβανόμενη και γρήγορη παράδοση μεγάλων και πολύπλοκων συστημάτων με επαναληπτική και σταδιακή ανάπτυξη με τη συνεχή δημιουργία προτύπων. Αναπτύχθηκε και δημοσιεύθηκε το 2000 από τον James A. Highsmith III, και προήλθε από μια παλαιότερή του έρευνα για τις επαναληπτικές μεθόδους ανάπτυξης. Οι ομάδες έργου εδώ μπορεί να είναι, ως προς το μέγεθος και τη γεωγραφική τους εγκατάσταση, κατανεμημένες διαφορετικά.

Η **Διαδικασία Ανάπτυξης** της ASD παρουσιάζεται να παρέχει ένα πλαίσιο με επαρκή καθοδήγηση, έτσι ώστε να αποτρέψει ένα έργο από το χάος, αλλά όχι ώστε να αποτρέψει τη

δημιουργικότητα. Αυτή η διαδικασία χαρακτηρίζεται από τρεις φάσεις: Το στάδιο των **Εικασιών** (Speculate), της **Συνεργασίας** (Collaborate) και της **Εκμάθησης** (Learn). Τον όρο «Εικασία» τον χρησιμοποιούμε αντί του όρου «Σχεδιασμού», γιατί ένα «σχέδιο» αντιμετωπίζεται συνήθως σαν κάτι όπου η αβεβαιότητα θεωρείται αδυναμία και οι αποκλίσεις από αυτό δηλώνουν αποτυχία. Με τον όρο «Συνεργασία» τονίζεται η σημαντικότητα της ομαδικής και συντονισμένης προσπάθειας, έτσι ώστε να καλυφθούν οι αλλαγές κατά τη διάρκεια της ανάπτυξης ως μέσο ανάπτυξης των συστημάτων. Τέλος, με τον όρο «εκμάθηση» τονίζεται η ανάγκη αναγνώρισης και αντίδρασης στα λάθη και στις αλλαγές των απαιτήσεων κατά τη διάρκεια της ανάπτυξης.



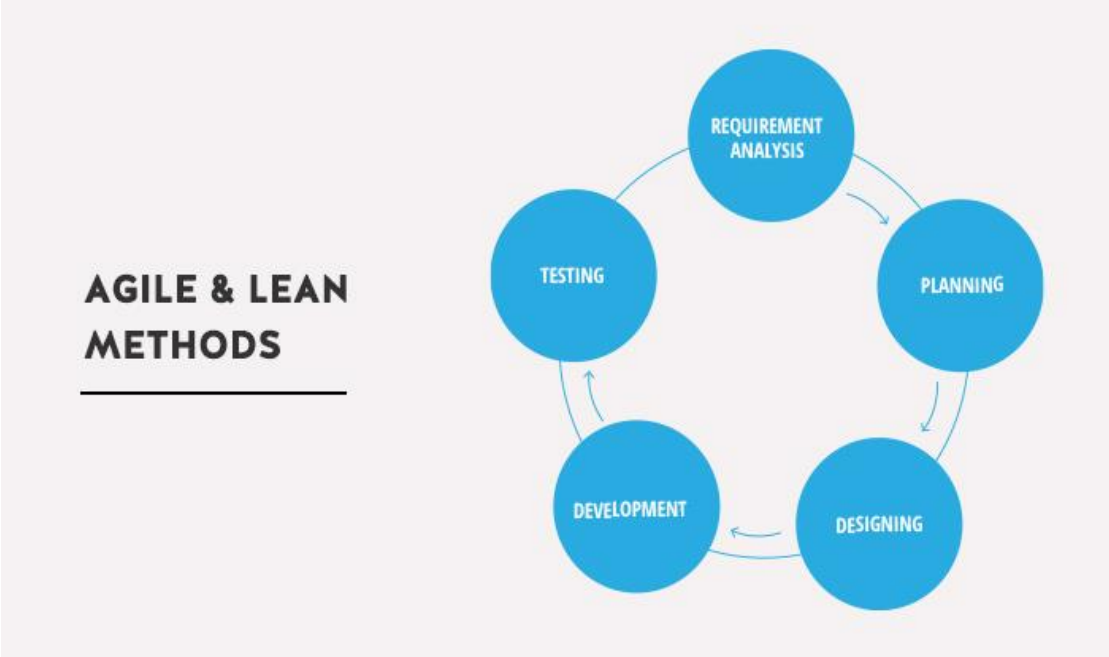
Εικόνα 19 - Προσαρμοστική Ανάπτυξη Λογισμικού (Adaptive Software Development - ASD) - Πηγή: tutorialspoint.com

Η ASD διακατέχεται κατά ένα μεγάλο μέρος από την επιχειρησιακή και διοικητική φιλοσοφία και ειδικότερα από τη σημασία των συνεργαζόμενων ομάδων και της ομαδικής εργασίας. Η προσέγγιση της ASD **δεν** περιγράφει δομές και ρόλους ομάδων. Οι ρόλοι με τις ευθύνες είναι πολλοί λίγοι.

Τέλος, για την υλοποίηση των φάσεων κατασκευής ενός λογισμικού εδώ οι **βασικές πρακτικές – τεχνικές είναι πολύ λίγες**. Ο Highsmith κατονομάζει ρητά τα ακόλουθα: επαναληπτική ανάπτυξη (iterative development), σχεδιασμός βάσει των χαρακτηριστικών / τμημάτων λογισμικού (feature based / component based planning) και ομάδες ανασκόπησης επικεντρωμένες στον πελάτη (customer focus group reviews). Εν κατακλείδι, η ASD ασχολείται με θέματα, όπως κοινωνικές, πολιτιστικές και ομαδικές δεξιότητες, που ευθυγραμμίζονται με την κατανεμημένη ανάπτυξη λογισμικού, αλλά δεν αναφέρει ρητά τίποτα σχετικό με το στυλ κώδικα που πρέπει να χρησιμοποιηθεί, το περιβάλλον τεχνολογίας ή την επιθυμητή επιχειρηματική κουλτούρα (Qumer & Henderson-Sellers, 2008).

2.7.3.7. Lean Software Development

Η Mary Poppendieck, με την ενασχόλησή της με τις μεθόδους Lean Manufacturing[1] και Total Quality Management[2] οδηγήθηκε στο συμπέρασμα ότι και οι μέθοδοι αυτές θα μπορούσαν να εφαρμοστούν για την ανάπτυξη λογισμικού πέραν της βιομηχανικής κατασκευής. Η προσαρμογή των μεθόδων αυτών στην ανάπτυξη λογισμικού είναι γνωστή ως Lean Software Development ή Lean Programming και έγινε γνωστή το 2001, από την Mary Poppendieck. Η μεθοδολογία αυτή δεν περιλαμβάνει κάποια συγκεκριμένη διαδικασία ανάπτυξης ούτε συγκεκριμένους ρόλους – υπευθυνότητες, αλλά προτείνει επτά συγκεκριμένες αρχές που πρέπει να ακολουθούνται στην ανάπτυξη λογισμικού (Poppendieck, 2003).



Εικόνα 20 - Lean Software Development - Πηγή: brainhub.eu

[1] Η μέθοδος Lean Manufacturing, γνωστή ως Σύστημα παραγωγής της Toyota, αναπτύχθηκε από τον Taiichi Ohno κατόπιν απαίτησης του Toyoda Sakichi, ιδιοκτήτη της Toyota, με σκοπό να καταστήσει εφικτή τη βιομηχανική παραγωγή αυτοκινήτων. Έχει δυο βασικές αρχές – αξίες, οι οποίες έχουν να κάνουν με την ταχύτητα ροής παραγωγής (rapid product flow) και την ενσωματωμένη ποιότητα (build in quality).

[2] Την ίδια χρονική περίοδο η μέθοδος Total Quality Management διδασκόταν στα πανεπιστήμια της Ιαπωνίας από τον Dr. W. Edwards Deming. Η Toyota αποφάσισε τη συγχώνευση των δύο μεθόδων για την ανάπτυξη μιας καλύτερης παραγωγικής διαδικασίας.

Οι επτά αρχές της μεθοδολογίας Lean Software Development είναι:

1. Εξάλειψη της σπατάλης (Eliminate Waste)
2. Ενίσχυση της μάθησης (Amplify Learning)
3. Αποφάσισε όσο το δυνατόν πιο αργά (Decide as late as possible)
4. Παράδωσε όσο το δυνατόν γρηγορότερα (Deliver as fast as possible)
5. Ενδυνάμωσε την ομάδα (Empower the team
6. Ενσωμάτωσε ακεραιότητα (Build integrity in)
7. Δες το σύνολο (See the whole)

Τέλος, σύμφωνα τις παραπάνω αρχές, και για κάθε μια από αυτές, προτάθηκαν είκοσι δύο εργαλεία που βοηθούν την κάθε αρχή να μετατραπεί σε ευέλικτη πρακτική. Αυτά είναι:

1.Εξάλειψη της σπατάλης (Eliminate Waste)

- Εργαλείο 1: Αναγνώριση της σπατάλης (Seeing Waste)
- Εργαλείο 2: Απεικόνιση της ροής ωφελιμότητας (Value Stream Mapping)

2.Ενίσχυση της μάθησης (Amplify Learning)

- Εργαλείο 3: Ανάδραση (Feedback)
- Εργαλείο 4: Επαναλήψεις (Iterations)
- Εργαλείο 5: Συγχρονισμός (Synchronization)
- Εργαλείο 6: Set – Based Development

3.Αποφάσισε όσο το δυνατόν πιο αργά (Decide as late as possible)

- Εργαλείο 7: Δικαίωμα Προαίρεσης (Options Thinking)
- Εργαλείο 8: The Last Responsible Moment
- Εργαλείο 9: Λαμβάνοντας Αποφάσεις (Making Decisions)

4.Παράδωσε όσο το δυνατόν γρηγορότερα (Deliver as fast as possible)

- Εργαλείο 10: Συστήματα Επιρροής (Pull Systems)
- Εργαλείο 11: Θεωρία Ουρών (Queuing Theory)
- Εργαλείο 12: Το κόστος της καθυστέρησης (Cost of Delay)

5.Ενδυνάμωσε την ομάδα (Empower the team)

- Εργαλείο 13: Αυτό-προσδιορισμός (Self – Determination)
- Εργαλείο 14: Παρακίνηση (Motivation)
- Εργαλείο 15: Ηγεσία (Leadership)
- Εργαλείο 16: Εξειδίκευση (Expertise)

Ενσωμάτωσε ακεραιότητα (Build integrity in)

- Εργαλείο 17: Αντιλαμβανόμενη Ακεραιότητα (Perceived Integrity)
- Εργαλείο 18: Εννοιολογική Ακεραιότητα (Conceptual Integrity)
- Εργαλείο 19: Επανακατασκευή (Refactoring)
- Εργαλείο 20: Δοκιμές (Testing)

Δες το σύνολο (See the whole)

- Εργαλείο 21: Μετρήσεις (Measurements)
- Εργαλείο 22: Συμβόλαια (Contracts)

2.8. Συμπεράσματα

Παρά την ευρύτερη χρήση της ευέλικτης ανάπτυξης λογισμικού τα τελευταία χρόνια, οι οργανισμοί δυσκολεύονται να υιοθετήσουν τις ευέλικτες πρακτικές. Και αυτό γιατί, όταν τις υιοθετήσουν, αντιμετωπίζουν διάφορες προκλήσεις στη διαδικασία υιοθέτησης, με το μεγαλύτερο ποσοστό εξ' αυτών να τις εγκαταλείπουν (Yu & Petter, 2014).

Από την πρώτη στιγμή που εμφανίστηκε η ανάγκη ανάπτυξης μεθοδολογιών ανάπτυξης λογισμικού, εμφανίστηκαν πάρα πολλές διαφορετικές διαδικασίες και μοντέλα διαδικασιών για να φτάσουμε στις σημερινές ευέλικτες μεθόδους, για την παραγωγή ποιοτικού λογισμικού. Αρχικά θα πρέπει να υπάρχει μια βαθιά γνώση όλων των παραδοσιακών μεθόδων και των πλεονεκτημάτων – αδυναμιών τους, έτσι ώστε να απορριφθούν και να μεταβούμε στις ευέλικτες μεθοδολογίες ανάπτυξης λογισμικού, ώστε να επιλέξουμε την κατάλληλη, η οποία θα πρέπει να ανταποκρίνεται, τελικά, στις ανάγκες του οργανισμού μας.

Συμπερασματικά, ένας οργανισμός για να είναι σίγουρος στο ποιο μοντέλο ανάπτυξης λογισμικού θα πρέπει να ακολουθήσει θα πρέπει να γνωρίζει εξαρχής τον στόχο τον οποίο θέλει να επιτύχει. Σίγουρα, μιας και διανύουμε την εποχής της 4ης Βιομηχανικής επανάστασης, όπου οι απαιτήσεις αλλάζουν συχνά και υπάρχει ανάγκη εξειδίκευσης, ενώ η παράδοση του προϊόντος θα πρέπει να είναι ταχύτατη, θα πρέπει να καταφεύγουμε στις μεθόδους ευέλικτης ανάπτυξης.

Η σελίδα αυτή είναι σκόπιμα λευκή.

Κεφάλαιο 3: Μεθοδολογία

Στο κεφάλαιο αυτό θα επιχειρήσουμε να αναδείξουμε και να αναπτύξουμε το βαθμό γνώσης των «ευέλικτων» μεθοδολογιών, τόσο από την πλευρά των εταιρειών λογισμικού στην ελληνική αγορά, όσο και από την πλευρά των στελεχών αυτών των επιχειρήσεων που ασχολούνται με την ανάπτυξη έργων λογισμικού. Συγκεκριμένα, θα περιγράψουμε αντίστοιχες έρευνες που διερευνούν ανάλογα ερωτήματα, δηλαδή το βαθμό επιτυχίας, τους παράγοντες επιτυχίας και τους παράγοντες, οι οποίοι δυσχεραίνουν την ολοκλήρωση ενός έργου κατασκευής λογισμικού. Στη συνέχεια, θα περιγραφούν και θα αναλυθούν τα ερευνητικά ερωτήματα που σχετίζονται με την έρευνά μας. Τέλος, στις επόμενες δυο ενότητες του κεφαλαίου, θα δοθεί η αναλυτική περιγραφή της έρευνας (διαδικασία, χαρακτηριστικά, τεχνικές, εργαλεία, κλπ.) και, μετά, η αναλυτική περιγραφή των στοιχείων της έρευνας (πληθυσμός, δείγμα, τρόποι επιλογής, κλπ.), έτσι ώστε να μεταβούμε στο τελευταίο κεφάλαιο της έρευνας, όπου θα δοθούν η ανάλυση των αποτελεσμάτων, τα τελικά συμπεράσματα και οι προτάσεις για το μέλλον.

3.1. Η εφαρμογή των «Ευέλικτων» Μεθοδολογιών

Η ευέλικτη ανάπτυξη λογισμικού, παρά τη μεγάλη υιοθέτηση και δημοτικότητα, η οποία παρατηρείται τα τελευταία δέκα κυρίως χρόνια, αντιμετωπίζει μεγάλη δυσκολία στην υιοθέτησή της, όπως παρατηρείται. Για τη μετάβαση από τις παραδοσιακές μεθόδους ανάπτυξης λογισμικού στις σύγχρονες ευέλικτες μεθόδους, απαιτείται βαθιά γνώση των μεθόδων αυτών, των πλεονεκτημάτων και των αδυναμιών τους, αλλά και της φύσης και της δομής του οργανισμού μέσα στον οποίο θα αναπτυχθούν, έτσι ώστε να επιλεχθεί η σωστή μέθοδος.

Ενδεικτικά θα αναφερθούν μερικά κριτήρια κάνοντας σύγκριση των παραδοσιακών μεθόδων με αυτές των ευέλικτων, όπως προέκυψε από τη βιβλιογραφική ανασκόπηση (Γεωργιάδου Α., 2014, Διερεύνηση της χρήσης ευέλικτων μεθόδων διοίκησης έργου). Αυτά είναι τα εξής:

1. **Θεμελιώδεις Υποθέσεις:** Τα συστήματα είναι πλήρως προσδιορίσιμα, προβλέψιμα, και μπορούν να αναπτυχθούν με σχολαστικό και εκτενή προγραμματισμό VS Προσαρμόσιμο λογισμικό υψηλής ποιότητας αναπτύσσεται από μικρές ομάδες, χρησιμοποιώντας τις αρχές της συνεχούς βελτίωσης του σχεδιασμού και των συχνών δοκιμών και βασίζεται στην ταχεία ανατροφοδότηση και την αλλαγή.
2. **Στόχος:** Προβλεψιμότητα και βελτιστοποίηση με σκοπό την Υψηλή διασφάλιση VS Εξερεύνηση ή προσαρμογή με σκοπό τη γρήγορη απόδοση αξίας.

3. **Απαιτήσεις:** Γνωστές από την αρχή σε μεγάλο βαθμό σταθερές: Σαφώς καθορισμένες και τεκμηριωμένες VS Επείγουσες με συχνές εναλλαγές, άγνωστες - ανακαλύπτονται κατά τη διάρκεια του έργου.
4. **Στυλ Διοίκησης:** Αυταρχικό με Εντολές και έλεγχο VS Αποκεντρωμένο - Ηγεσία και συνεργασία.
5. **Διοίκηση Γνώσης:** Ρητή VS Σιωπηρή.
6. **Επιθυμητή Δομή Οργανισμού:** Γραφειοκρατική με υψηλή τυποποίηση VS Ευέλικτη και Συμμετοχική ενθαρρύνοντας τη συνεργατική κοινωνική δράση.
7. **Αλλαγή:** Τάση αποστροφής στην αλλαγή VS Αποδοχή της αλλαγής.
8. **Εστίαση:** Στις διαδικασίες VS Στους ανθρώπους.
9. **Μέγεθος Έργου:** Μεγάλο VS Μικρό.
10. **Μέγεθος Ομάδας:** Μεγάλο VS Μικρό.
11. **Οργάνωση Ομάδας:** Δομημένες ομάδες VS Αυτοδιοικούμενες ομάδες
12. **Μέλη Ομάδας:** Κατανεμημένες ομάδες από ειδικούς - Προσανατολισμένοι στο σχεδιασμό, με επαρκείς δεξιότητες και πρόσβαση σε εξωτερική γνώση VS Ευέλικτοι, καταρτισμένοι, συνεργατικοί - Το ανώτερο τεχνικό προσωπικό βρίσκεται εγκατεστημένο στον ίδιο χώρο.
13. **Ανάθεση Ρόλων:** Ατομική όπου ευνοείται η εξειδίκευση VS Αυτό-διοικούμενες ομάδες όπου ευνοείται η εναλλαγή ρόλων.
14. **Επικοινωνία:** Επίσημη VS Ανεπίσημη.
15. **Πελάτες:** Χαμηλή συμμετοχή - παθητικός ρόλος VS Ο πελάτης θεωρείται ως ένα μέλος της ομάδας: Ενεργός ρόλος.
16. **Σχεδιασμός Έργου:** Καθοδηγούμενος από καθήκοντα και διαδικασίες VS Καθοδηγούμενος από τα χαρακτηριστικά του προϊόντος.
17. **Αρχικός Σχεδιασμός:** Πλήρης VS Ελάχιστος.
18. **Διαδικασία Ανάπτυξης Λογισμικού:** Καθολική προσέγγιση και λύση για να παρέχει προβλεψιμότητα και υψηλή διασφάλιση VS Ευέλικτη προσέγγιση, προσαρμοσμένη με κατανόηση των συνολικών συμφραζομένων αναγκών για να παρέχει ταχύτερη ανάπτυξη.
19. **Μοντέλο Ανάπτυξης:** Μοντέλο τύπου "Καταρράκτη" ή Spiral, ή παραλλαγή αυτών VS Μοντέλο εξελικτικής παράδοσης.
20. **Κύκλοι Έργου:** Περιορισμένοι VS Πολυάριθμοι.
21. **Χρησιμοποιούμενη Τεχνολογία:** Χωρίς περιορισμούς VS Ευνοεί την αντικειμενοστραφή τεχνολογία.
22. **Τεκμηρίωση:** Βαριά ή Εκτενής ρητή καταγραφή VS Ελαφριά (αντικαταστάθηκε από την επικοινωνία πρόσωπο με πρόσωπο).

23. **Επανασχεδιασμός:** Ακριβός VS Μικρές δαπάνες.
24. **Μέτρηση Επιτυχίας:** Συμμόρφωση με το σχεδιασμό VS Επιχειρηματική αξία.
25. **Απόδοση Επένδυσης (ROI):** Στο τέλος του έργου VS Σε αρχικά στάδια του έργου.
26. **Κίνδυνοι:** Κατανοητοί κίνδυνοι με μικρές επιπτώσεις VS Άγνωστοι κίνδυνοι με σημαντικές επιπτώσεις.

Συμπερασματικά, οι παραδοσιακές μέθοδοι, στις οποίες εφαρμόζονται κυρίως το μοντέλο καταρράκτη και το σπειροειδές μοντέλο, έχουν τα δικά τους πλεονεκτήματα ανταποκρινόμενα σε ορισμένες απαιτήσεις, όπως και οι ευέλικτες μέθοδοι, ενώ ταυτόχρονα έχουν μεγάλες αδυναμίες και μειονεκτήματα. Γι' αυτό και η επιλογή θα πρέπει να γίνεται με ιδιαίτερη προσοχή. Στη συνέχεια, αναλύονται δεκατέσσερις μελέτες περίπτωσης εφαρμογής υιοθέτησης των «ευέλικτων» μεθοδολογιών. Με βάση τα αποτελέσματα αυτών των μελετών περίπτωσης, θα αναλυθούν τα ερευνητικά μας ερωτήματα, τα οποία θα ακολουθήσουν την ανάπτυξης της έρευνας.

Συγκεκριμένα, εντοπίστηκαν ως συναφή και παρουσιάζονται ακολούθως τα εξής ερευνητικά άρθρα:

1. Προκλήσεις υιοθέτησης ευέλικτων μεθόδων από έναν δημόσιο οργανισμό (Nuottila, J., Aaltonen, K. & Kujala, J. (2016) Challenges of adopting agile methods in a public organization, International Journal of Information Systems and Project Management, Vol. 4, No. 3, 65-85),
2. Οι κρίσιμοι παράγοντες επιτυχίας και τα εμπόδια για την ελαφριά βελτίωση του λογισμικού σε μια ευέλικτη ανάπτυξη: μια ανασκόπηση της βιβλιογραφίας (Kouzari, E., Gerogiannis, V., Stamelos, I. & Kakarontzas G. (2015) Critical success factors and barriers for lightweight: software process improvement in agile development: A literature review, France, 10th International Conference on Software Engineering and Applications (ICSOFT-EA-2015), pages 151-159),
3. Προσδιορισμός σημαντικών παραγόντων επιτυχίας στην υιοθέτηση πρακτικών ανάπτυξης ευέλικτων λογισμικών (Misra, S. C., Kumar, V. & Kumar U. (2009) Identifying some important success factors in adopting agile software development practices, India, Elsevier, Volume 82, Issue 11, pp: 1869 – 1890),
4. Υιοθετώντας – Προσαρμόζοντας τις Agile πρακτικές σε πανεπιστημιακά πλαίσια (Masood, Z., Hoda, R. & Blincoe, K. (2018) Adapting Agile Practices in University Contexts, New Zealand, Elsevier, Volume 144, pp: 501 – 510),

5. Παράγοντες που σχετίζονται με την ευελιξία ανάπτυξης λογισμικού επιτυχημένων έργων (Sheffield, J., Lemétayer, J. (2012), Factors associated with the software development agility of successful projects, New Zealand, Elsevier, Volume 31, Issue 3, April 2013, Pages 459-472),
6. Ένα μοντέλο κατάλληλης έκτακτης ανάγκης για κρίσιμους παράγοντες επιτυχίας στα έργα ανάπτυξης λογισμικού. Σύγκριση ευέλικτων και παραδοσιακών μεθοδολογιών βάσει σχεδίου (Ahimbisibwe, A., Cavana, R., Daellenbach, U. (2013), A contingency fit model of critical success factors for software development projects. A comparison of agile and traditional plan-based methodologies, New Zealand, Journal of Enterprise Information Management, Vol. 28 No. 1, 2015, pp. 7-33),
7. Πλεονεκτήματα και εμπόδια πίσω από την επιτυχημένη ευέλικτη ανάπτυξη - πληροφορίες από τις τρεις εταιρείες εντάσεως λογισμικού στη Φινλανδία (Pikkarainen, M., Salo, O., Kuusela, R., Abrahamsson, P. (2011), Strengths and barriers behind the successful agile deployment – insights from the three software intensive companies in Finland, Springer Science – Business Media, Vol 17, pages: 675 – 702),
8. Προσδιορισμός ορισμένων κρίσιμων αλλαγών που απαιτούνται κατά την υιοθέτηση ευέλικτων πρακτικών σε παραδοσιακά προγράμματα ανάπτυξης λογισμικού (Misra, S. C., Kumar, V., Kumar, U. (2009), Identifying some critical changes required in adopting agile practices in traditional software development projects, Emerald Group Publishing Limited, International Journal of Quality & Reliability Management, Vol. 27 No. 4, 2010, pp. 451 – 474),
9. Πώς οι ανθρώπινες πτυχές εντυπώνουν τη μετάβαση και την υιοθέτηση ανάπτυξης λογισμικού Agile (Gandomani, T. J., Zulzalil, H., Ghani, A. A. A., Sultan, A. B. M. & Sharif, K. Y. (2014), How Human Aspects Impress Agile Software Development Transition and Adoption, International Journal of Software Engineering and Its Applications, Vol.8, No.1 (2014), pp: 129-148),
10. Παράγοντες επιτυχίας και αποτυχίας που επηρεάζουν την υλοποίηση του έργου χρησιμοποιώντας τη μεθοδολογία ανάπτυξης λογισμικού Agile (Dhir, S., Kumar, D. & Singh, V. B. (2019), Success and Failure Factors that Impact on Project Implementation Using Agile Software Development Methodology, Springer Nature Singapore Pte Ltd. 2019 M. N. Hoda et al. (eds.), Software Engineering, Advances in Intelligent Systems and Computing 731, pages: 647 – 654),

11. Οι παράγοντες που επηρεάζουν την επιτυχία των συνεχιζόμενων ευέλικτων προγραμμάτων ανάπτυξης λογισμικού (Tam, C., Moura, E. J. D.C., Oliveira, T. & Varajão, J. (2020), The factors influencing the success of on-going agile software development projects, Elsevier, International Journal of Project Management 38 (2020), pages: 165 – 176),
12. Παράγοντες επιτυχίας της ανάπτυξης ευέλικτων συστημάτων πληροφοριών (ASD): Μια ποιοτική μελέτη (Hummel, M. & Epp, A. (2015), Success Factors of Agile Information Systems Development: A Qualitative Study, 48th Hawaii International Conference on System Sciences),
13. Μια ανασκόπηση σχετικά με τους κρίσιμους παράγοντες επιτυχίας της ανάπτυξης λογισμικού Agile (Aldahmash, A., Gravell, A. M. & Howard, Y. (2017), A Review on the Critical Success Factors of Agile Software Development, Springer International Publishing AG 2017J, Stolfa et al. (Eds.): Euro SPI 2017, CCIS 748, pp. 504–512, 2017),
14. Μια εμπειρική μελέτη: Κατανόηση παραγόντων και εμποδίων για την εφαρμογή ευέλικτων μεθόδων στη Μαλαισία (Asnawi, A. L., Gravell, A. M. & Wills, G. B. (2010), An Empirical Study: Understanding Factors and Barriers for Implementing Agile Methods in Malaysia, 5th International Doctoral Symposium on Empirical Software Engineering - IDoESE)

3.1.1. Προκλήσεις υιοθέτησης ευέλικτων μεθόδων σε ένα δημόσιο οργανισμό

Ο **στόχος** του παρόντος ερευνητικού εγγράφου ήταν να εντοπίσει και να κατηγοριοποιήσει τις προκλήσεις που μπορεί να εμποδίσουν την αποδοτική υιοθέτηση των «ευέλικτων» μεθοδολογιών από το Υπουργείο Μεταφορών και Επικοινωνιών της Φινλανδίας. Δηλαδή, υλοποιήθηκε μια μελέτη περίπτωσης, μέσω ημι δομημένων συνεντεύξεων.

Τα **ευρήματα** ήταν πολλά, ενώ οι προκλήσεις **κατηγοριοποιήθηκαν** σε 7 τομείς, οι οποίοι είναι οι εξής:

1. Τεκμηρίωση
2. Εκπαίδευση, εμπειρία και δέσμευση
3. Επικοινωνία και συμμετοχή των ενδιαφερομένων
4. Ρόλοι ευέλικτου ρυθμιστή
5. Θέση των ευκίνητων ομάδων
6. Νομοθεσία
7. Πολυπλοκότητα αρχιτεκτονικής SW και ολοκλήρωση συστημάτων

3.1.2. Οι κρίσιμοι παράγοντες επιτυχίας και τα εμπόδια για την ελαφριά βελτίωση του λογισμικού σε μια ευέλικτη ανάπτυξη: μια ανασκόπηση της βιβλιογραφίας

Ο **στόχος** του παρόντος ερευνητικού εγγράφου ήταν, μέσω της βιβλιογραφικής έρευνας, να εντοπίσει τους παράγοντες επιτυχίας που οδηγούν στην μεγιστοποίηση της απόδοσης (ROI) στις μικρομεσαίες επιχειρήσεις, καθώς και τα εμπόδια, απαντώντας στα παρακάτω ερευνητικά ερωτήματα:

- Ποια είναι τα ειδικά χαρακτηριστικά των μικρομεσαίων επιχειρήσεων λογισμικού, όταν ακολουθούν τις ευέλικτες μεθοδολογίες;
- Ποιοι είναι οι κύριοι παράγοντες κρίσιμης επιτυχίας και τα εμπόδια που εμπλέκονται στο SPI;
- Πώς επηρεάζουν τους παράγοντες κρίσιμης επιτυχίας και τα εμπόδια την απόδοση επένδυσης (ROI) σε εταιρείες που εφαρμόζουν ελαφρύ SPI;

Τα ευρήματα ήταν πολλά, ενώ οι παράγοντες επιτυχίας αλλά και τα εμπόδια κατηγοριοποιήθηκαν ως εξής:

1. Δέσμευση
2. Συμμετοχή του προσωπικού
3. Εκπαίδευση
4. Πόροι
5. Ομάδες δράσης διαδικασίας
6. Εμπειρία προσωπικού
7. Καθοδήγηση
8. Κριτικές – Ανατροφοδότηση
9. Μεθοδολογία εφαρμογής
10. Παρακολούθηση
11. Επικοινωνία
12. Απόδοση επένδυσης
13. Ευαισθητοποίηση του SPI
14. Πρόσθετοι παράγοντες που δεν αναφέρονται συχνά στη βιβλιογραφία.

3.1.3. Προσδιορισμός σημαντικών παραγόντων επιτυχίας στην υιοθέτηση πρακτικών ανάπτυξης ευέλικτων λογισμικών

Ο **στόχος** του παρόντος ερευνητικού εγγράφου ήταν να εντοπίσει τους παράγοντες επιτυχίας από την οπτική γωνία των επαγγελματιών που ασκούν «ευελιξία», έτσι ώστε να οδηγήσουν τα έργα τους στην επιτυχή ολοκλήρωση. Ορίστηκε ένα υποθετικό πλαίσιο 14 υποθετικών παραγόντων, ενώ κλήθηκαν να απαντήσουν στο ακόλουθο ερευνητικό ερώτημα: «**Ποιοι είναι**

οι παράγοντες από την πλευρά των επαγγελματιών της Agile, που επιθυμούν να υιοθετήσουν τις Agile πρακτικές, οι οποίοι θα επηρεάσουν την επιτυχία των έργων».

Τα ευρήματα έδειξαν ότι, από την ανάλυση των ποσοτικών δεδομένων που ελήφθησαν, προέκυψαν **εννέα παράγοντες που έχουν στατιστικά σημαντικές επιπτώσεις σε σχέση με την επιτυχία, ενώ πέντε από αυτούς αποτελούν περιορισμούς για την ολοκλήρωση ενός έργου.** Οι παράγοντες που διαδραματίζουν σημαντικό ρόλο για την επιτυχία είναι οι εξής: Πελατοκεντρικά θέματα, Η ώρα απόφασης, Εταιρική κουλτούρα, Έλεγχος, Προσωπικά χαρακτηριστικά, Κοινωνική κουλτούρα, Εκπαίδευση – μάθηση, Ομαδική διανομή, Μέγεθος ομάδας, Σχεδίαση, Τεχνική ικανότητα και Επικοινωνία – διαπραγμάτευση.

3.1.4. Υιοθετώντας – Προσαρμόζοντας τις Agile πρακτικές σε πανεπιστημιακά πλαίσια

Στο παρόν ερευνητικό έγγραφο **βλέπουμε** την εφαρμογή από τους φοιτητές και τους καθηγητές των «ευέλικτων» μεθοδολογιών σε ένα Πανεπιστήμιο (Auckland). Η μελέτη έχει ως **σκοπό** να εντοπίσει τους περιορισμούς που αντιμετωπίζουν οι φοιτητές εφαρμόζοντας «ευέλικτες» πρακτικές σε ένα μάθημα, απαντώντας στα παρακάτω ερευνητικά ερωτήματα:

- Ποιοι είναι οι συνηθέστεροι περιορισμοί που αντιμετωπίζουν οι φοιτητές, ενώ ασκούν ευέλικτες μεθόδους σε ένα πανεπιστημιακό μάθημα;
- Ποιες ευέλικτες πρακτικές ασκούν οι μαθητές και πώς αυτές προσαρμόστηκαν ώστε να ταιριάζουν στο πανεπιστημιακό πλαίσιο;
- Ποιες προσαρμοσμένες πρακτικές θεωρούνται ωφέλιμες και αποτελεσματικές όσον αφορά τα αποτελέσματα και ποιες όχι;

Τα **ευρήματα** περιγράφουν τους **περιορισμούς,** τις **προσαρμογές,** την **αποτελεσματικότητα** και τη **σχέση μεταξύ περιορισμών και προσαρμογών** εφαρμόζοντας ευέλικτες πρακτικές, σε ένα πανεπιστημιακό πλαίσιο.

Οι **περιορισμοί** που εντοπίστηκαν από τους μαθητές έχουν να κάνουν με τα εξής: Περιορισμοί προγραμματισμού, Θέματα επικοινωνίας της ομάδας, Θέματα που σχετίζονται με πελάτες, Έλλειψη αφοσίωσης, Προσωπικές δεσμεύσεις και Τεχνικοί περιορισμοί.

Οι **προσαρμογές** που θα πρέπει να εφαρμοστούν άμεσα έχουν να κάνουν με:

- Τις πρακτικές του Scrum,
- Τις Συνεδριάσεις Sprint,
- Τους Scrum ρόλους,
- Σαν XP πρακτικές θα πρέπει να έχουμε προγραμματισμό ζευγών

Τέλος, τα **αποτελέσματα** που προκύπτουν αποδεικνύουν ότι οι προσαρμογές είχαν **θετικές** και **αρνητικές επιπτώσεις,** κάνοντας αντιληπτή την αποτελεσματικότητα.

3.1.5. Παράγοντες που σχετίζονται με την ευελιξία ανάπτυξης λογισμικού επιτυχημένων έργων

Ο **στόχος** του παρόντος ερευνητικού εγγράφου είναι μέσω των μελών των «ευέλικτων» κοινοτήτων όπως της PRINCE2, της PMI, κα. να συλλεχθούν μέσω συνεντεύξεων εμπειρικά αποδεικτικά στοιχεία σχετικά με το ερώτημα: ***ποιοι παράγοντες στο έργο και στο περιβάλλον του έργου είναι ενδεικτικοί της «ευελιξίας» σε επιτυχημένα έργα ανάπτυξης λογισμικού;*** Η **τρέχουσα έρευνα συμβάλλει** στη βιβλιογραφία, για την επιτυχία της μεθοδολογίας για ένα μοντέλο «βέλτιστης πρακτικής» μεθοδολογικής επιτυχίας.

Οι **ενδεικτικοί παράγοντες της ευελιξίας ανάπτυξης λογισμικού** στο περιβάλλον του έργου είναι: η **οργανωτική κουλτούρα** και η **ενδυνάμωση της ομάδας του έργου**.

Η τρέχουσα μελέτη έχει τουλάχιστον **έξι συγκεκριμένες επιπτώσεις**. **Πρώτον**, η εξέταση του κατάλληλου επιπέδου «ευελιξίας» ανάπτυξης λογισμικού θα πρέπει να διευρυνθεί. **Δεύτερον**, για να επιτευχθεί η επιτυχία του έργου, οι επαγγελματίες θα πρέπει να διασφαλίσουν την αφοσίωσή τους σε μια συγκεκριμένη μεθοδολογία ανάπτυξης. **Τρίτον**, η τρέχουσα μελέτη καταδεικνύει ότι η ανώτατη διοίκηση του έργου και ο πελάτης συνήθως αποτυγχάνουν να συμφωνήσουν ως προς το ποια μορφή «ευελιξίας» θα είναι κατάλληλη για την ανάπτυξη του δικού τους λογισμικού. **Τέταρτον**, τα ευρήματα δείχνουν ότι η ομάδα του έργου, η διαχείριση έργου, ο πελάτης και η ανώτατη διοίκηση πρέπει να υιοθετήσουν μια πιο «ευέλικτη» και προσαρμόσιμη προσέγγιση. **Πέμπτον**, οι ασκούντες «ευελιξία» θα πρέπει να σκεφτούν πώς να προσαρμόσουν τη διαδικασία, ώστε να ταιριάζουν καλύτερα στις ανάγκες τους. **Έκτον**, οι παράγοντες που είναι σημαντικοί για την επιτυχία του έργου μακροπρόθεσμα μπορεί να διαφέρουν από εκείνους που είναι ενδεικτικοί της «ευελιξίας».

Αυτή η μελέτη έχει **περιορισμούς**. **Πρώτον**, λόγω της πολυπλοκότητας των θεμάτων και της έλλειψης κατάλληλων ποσοτικών οργάνων στη μεθοδολογική επιτυχία της βιβλιογραφίας, τα ευρήματα αυτής της έρευνας δεν είναι οριστικά. Συνίσταται επιπλέον έρευνα. **Δεύτερον**, το μέγεθος του δείγματος παραμένει σχετικά χαμηλό. Ένα μεγαλύτερο μέγεθος δείγματος θα μπορούσε να παρέχει ακριβέστερα στατιστικά στοιχεία. **Τρίτον**, οι περισσότερες από τις απαντήσεις προέρχονταν από χώρες χαμηλής ισχύος. Αυτό περιόρισε την έρευνα της επίδρασης της απόστασης ισχύος. **Τέταρτον**, ορισμένα από τα δεδομένα που συλλέχθηκαν για δημογραφικούς σκοπούς (μέγεθος οργανισμού, βιομηχανία, χώρα, εμπειρία του ερωτώμενου και ρόλος) θα μπορούσαν να έχουν ταξινομηθεί ως παράγοντες που επηρεάζουν το έργο και το περιβάλλον του και να συμπεριληφθούν στην επίσημη ανάλυση.

Τέλος, αυτή η μελέτη έχει επίσης **οριοθετήσεις**. **Πρώτον**, λόγω μειωμένων πόρων, το πεδίο της έρευνας ήταν περιορισμένο. **Δεύτερον**, οι κατασκευές που χρησιμοποιούνται δεν βασίζονται σε θεωρητικά ορθές αντιλήψεις και δοκιμασμένα όργανα. **Τρίτον**, ενώ οι συνεντεύξεις επέτρεψαν στην έρευνα να επικεντρωθεί σε επιλεγμένα θέματα, πολλά ενδιαφέροντα θέματα παραμένουν εκτός πεδίου. **Τέταρτον**, το επίκεντρο της έρευνας ήταν στα μέλη των κοινοτήτων της πρακτικής ανάπτυξης λογισμικού.

3.1.6. Ένα μοντέλο κατάλληλης έκτακτης ανάγκης για κρίσιμους παράγοντες επιτυχίας για έργα ανάπτυξης λογισμικού. Σύγκριση ευέλικτων και παραδοσιακών μεθοδολογιών βάσει σχεδίου

Ο **σκοπός του παρόντος ερευνητικού άρθρου** είναι να εντοπίσει και να κατηγοριοποιήσει τους κρίσιμους παράγοντες επιτυχίας (ΚΠΕ) και να αναπτύξει ένα μοντέλο προσαρμοσμένης έκτακτης ανάγκης που αντιπαραβάλλει τις προοπτικές των παραδοσιακών μεθοδολογιών στις «ευέλικτες» μεθοδολογίες. Για να **πραγματοποιηθεί** ο σκοπός, **έγινε** μια συστηματική ανασκόπηση της βιβλιογραφίας (148 άρθρα), όπου οι 37 ΚΠΕ που εντοπίστηκαν να κατηγοριοποιηθούν σε 3 κατηγορίες. Αυτοί είναι: οι οργανωτικοί παράγοντες, οι παράγοντες της ομάδας και οι παράγοντες των πελατών.

Τα **ευρήματα** εντοπίστηκαν μέσα στις τρεις κύριες κατηγορίες των ΚΠΕ και οι μεμονωμένοι παράγοντες ταξινομούνται με βάση το πόσο συχνά έχουν αναφερθεί σε προηγούμενες μελέτες. Οι διαφορές σε αυτές τις κατατάξεις, καθώς και η μικτή εμπειρική υποστήριξη, υποδηλώνουν ότι η προηγούμενη έρευνα μπορεί να μην είχε θεωρηθεί επαρκώς όταν ήταν συγκεκριμένοι οι ΚΠΕ, και οι οποίοι θα επηρεάσουν την επιτυχία του έργου.

Τέλος, οι περιορισμοί της έρευνας έχουν να κάνουν με τη δοκιμή του μοντέλου προσαρμογής έκτακτης ανάγκης που αναπτύχθηκε χρησιμοποιώντας εμπειρικά δεδομένα. Δεν υπάρχει ευρεία συναίνεση μεταξύ ερευνητών και επαγγελματιών στην κατηγοριοποίηση των ΚΠΕ για προγράμματα ανάπτυξης λογισμικού. Ωστόσο, μέσω μιας εκτεταμένης αναζήτησης και ανάλυσης της βιβλιογραφίας για τους ΚΠΕ για έργα ανάπτυξης λογισμικού, η έρευνα παρέχει μεγαλύτερη σαφήνεια σχετικά με τις κατηγορίες των ΚΠΕ και πώς μπορούν να μοντελοποιηθούν οι άμεσες, έμμεσες και μετριοπαθείς επιπτώσεις τους στην επιτυχία του έργου.

3.1.7. Πλεονεκτήματα και εμπόδια πίσω από την επιτυχημένη ευέλικτη ανάπτυξη - πληροφορίες από τις τρεις εταιρείες εντάσεως λογισμικού στη Φινλανδία

Στόχος αυτής της έρευνας είναι να αναλύσει τα **πλεονεκτήματα και τα εμπόδια** σε επιλεγμένα πιλοτικά έργα από τρεις οργανισμούς λογισμικού, οι οποίοι επιχειρούν να αναπτύξουν τα έργα αυτά μέσω των «ευέλικτων» μεθοδολογιών. Ένας **άλλος στόχος** είναι να ολοκληρωθεί αυτή η ανάλυση με συστάσεις για την υποστήριξη εταιρειών λογισμικού που σχεδιάζουν «ευέλικτη» ανάπτυξη. Προκειμένου να **επιτευχθούν αυτοί οι στόχοι**, πραγματοποιήθηκαν τρεις σε βάθος διαχρονικές μελέτες περιπτώσεων στις οποίες διερευνήθηκε η ανάπτυξη «ευέλικτης» μεθοδολογίας σε καθεμία από τις τρεις εταιρείες.

Τα **Αποτελέσματα της Έρευνας** έδειξαν ότι τα δυνατά σημεία της ανάπτυξης λογισμικού με «ευέλικτες» μεθοδολογίες σχετίζονται κυρίως με την **επικοινωνία και τη συνεργασία** εντός της ομάδας, και μεταξύ της ομάδας και των ενδιαφερόμενων μερών κατά την εκτέλεση της ανάπτυξης. Άλλα προσδιορισμένα **πλεονεκτήματα** που προέκυψαν χρησιμοποιώντας την «ευέλικτη» ανάπτυξη λογισμικού ήταν οι **βελτιώσεις στις δοκιμές μονάδων, οι βελτιώσεις στη διαχείριση των απαιτήσεων και στη διαχείριση της ζήτησης των πελατών, οι δραστηριότητες βελτίωσης του σχεδιασμού και της εκτίμησης, η αυτοοργάνωση, οι μέθοδοι βελτίωσης και οι μαθησιακές πτυχές**. Ο βασικός **περιορισμός** αυτής της έρευνας είναι ότι τα δεδομένα που συλλέγονται βασίζονται κυρίως στις **υποκειμενικές απόψεις** των ερωτηθέντων από τις εταιρείες που μελετήθηκαν.

3.1.8. Προσδιορισμός ορισμένων κρίσιμων αλλαγών που απαιτούνται κατά την υιοθέτηση ευέλικτων πρακτικών σε παραδοσιακά προγράμματα ανάπτυξης λογισμικού

Το **πρωταρχικό κίνητρο** πίσω από αυτή την έρευνα είναι να επιχειρηθεί να εντοπιστούν μερικές από τις αλλαγές που πρέπει να υλοποιηθούν σε οργανισμούς που βρίσκονται σε διαδικασία μετάβασης στην εφαρμογή «ευελιξίας». Με βάση αυτό, είναι δυνατόν να βοηθηθούν οι διαχειριστές έργων που εμπλέκονται σε τέτοιες μεταβατικές δραστηριότητες. Γι΄αυτό και πραγματοποιήθηκε η παρούσα **εμπειρική μελέτη** βάσει της έρευνας.

Οι **αλλαγές** που απαιτούνται για την υιοθέτηση πρακτικών ASD από παραδοσιακά έργα, ώστε να μεταβούν σε «ευέλικτα» θα πρέπει να έχουν τα εξής χαρακτηριστικά:

- **Θεμελιακά Χαρακτηριστικά:** Ανεξάρτητα από το αν ένα έργο θέλει να μετασχηματιστεί όλο ή εν μέρει σε «ευέλικτο», θα πρέπει να επικεντρωθούμε πρώτα στους βασικούς τομείς και στη συνέχεια να στρέψουμε σταδιακά την προσοχή μας στους άλλους τομείς. Για να μπορέσει να γίνει αυτό, θα ήταν χρήσιμο να προσδιοριστούν οι βασικές αλλαγές που απαιτούνται για την υιοθέτηση ευέλικτων φιλοσοφιών σε ένα έργο.

- **Μεγάλες αλλαγές:** στην οργανωτική κουλτούρα, στο στυλ διαχείρισης, στις στρατηγικές διαχείρισης γνώσης και στις διαδικασίες ανάπτυξης.

Τέλος, οι μελέτες αυτής της φύσης συχνά συνοδεύονται από **διαφορετικούς περιορισμούς** διαφορετικής βαρύτητας. Είναι, επομένως, σημαντικό να δούμε τα αποτελέσματα της έρευνας που παρουσιάζονται σε αυτό το έγγραφο εντός των ορίων αυτών των περιορισμών. Αυτοί οι περιορισμοί που παρουσιάζονται θα πρέπει να εξεταστούν μαζί, έτσι ώστε να εμπνεύσουν μελλοντικές ερευνητικές εργασίες στο ίδιο θέμα. Οι **περιορισμοί** έχουν ως εξής: (1) Σε αυτό το έργο, προσπαθήσαμε να βρούμε αποδεικτικά στοιχεία βάσει ερευνών, (2) Οι μεταβλητές που χρησιμοποιούνται στη μελέτη είναι περίπλοκες από τη φύση τους, (3) Λόγω της περιορισμένης διαθεσιμότητας επαρκούς σχετικής βιβλιογραφίας, στο συγκεκριμένο πρόβλημα που αντιμετωπίζουμε σε αυτό το άρθρο, τα κομμάτια της βιβλιογραφίας που χρησιμοποιούνται για την έννοια κάθε κατασκευής δεν είναι πάντα επιστημονικά, (4) Η μεθοδολογία του να ζητάμε από λίγα άτομα να γνωμοδοτήσουν, χωρίς μια μεσολαβητική μεθοδολογία ομάδας εστίασης έχει τους περιορισμούς της, ακόμα και αν το ερωτηματολόγιο έχει δοκιμαστεί για την εγκυρότητα, τη χρησιμότητα και την αναγνωσιμότητά του πριν αποσταλεί για τη συλλογή δεδομένων από το πεδίο, (5) Σε αυτό το έργο, έχουμε περιοριστεί στην ανάλυση των συγκεντρωτικών δεδομένων που λαμβάνονται από διάφορους ερωτηθέντες διαφορετικών ηπείρων σε όλο τον κόσμο, (6) Στοχεύσαμε σε άτομα, αλλά τους ρωτήσαμε για πληροφορίες ομάδας (συμπεριλαμβανομένων άλλων ομάδων δέσμευσης) αλλά και για εταιρικά και για κοινωνικά ζητήματα, (7) Αυτή η μελέτη περιορίζεται στα όρια της υπόθεσης ότι τα δεδομένα που λαμβάνονται σε διαφορετικές λειτουργίες εργασίας είναι εξίσου σημαντικά, (8) Στην ερευνητική μεθοδολογία, λαμβάνουμε τις διάφορες μεταβλητές σε κάθε περιοχή και απλώς λαμβάνουμε τον μέσο όρο τους (διάμεσος).

3.1.9. Πώς οι ανθρώπινες πτυχές εντυπώνουν τη μετάβαση και την υιοθέτηση ανάπτυξης λογισμικού Agile

Το παρόν ερευνητικό έγγραφο επικεντρώθηκε στις ανθρώπινες πτυχές της διαδικασίας μετάβασης σε «ευελιξία», με τη διεξαγωγή μελέτης (Grounded Theory – Δημιουργία Θεωρίας) με τη συμμετοχή 32 εμπειρογνωμόνων Agile. Η **κύρια συμβολή** αυτού του ερευνητικού εγγράφου είναι η ταυτοποίηση και η ταξινόμηση των διαφορετικών ανθρώπινων πτυχών της διαδικασίας μετάβασης σε «ευελιξία». Η ανάλυση των συλλεχθέντων δεδομένων έδειξε ότι οι ανθρώπινες πτυχές της διαδικασίας μετάβασης σε «ευελιξία» μπορούν να ταξινομηθούν σε διάφορες κατηγορίες. Ενώ μερικά από αυτά αποτελούν εμπόδια στην αλλαγή, κάποια άλλα λειτουργούν ως επιταχυντές αλλαγής.

Η **αλλαγή της μεθόδου ανάπτυξης** από παραδοσιακές σε ευέλικτες μεθόδους **απαιτεί εκτεταμένες αλλαγές στη νοοτροπία των ανθρώπων και στις συμπεριφορές τους**. Αυτό σημαίνει ότι οι άνθρωποι διαδραματίζουν κρίσιμο ρόλο στην Ευέλικτη διαδικασία μετασχηματισμού / μετάβασης (Agile transformation / transition process - ATP) και μπορούν να λειτουργήσουν τόσο ως οδηγοί όσο και ως εμπόδια της μετάβασης και της υιοθέτησης του Agile. Αυτή η μελέτη διεξήχθη με βάση τη **Grounded Theory** ως ποιοτική ερευνητική μέθοδο. Η **ανάλυση δεδομένων** έγινε με κωδικοποίηση των δεδομένων και ξεκίνησε με τη συλλογή ορισμένων δεδομένων. Κατά τη διάρκεια της ανάλυσης δεδομένων, οι ερευνητές χρησιμοποίησαν θεωρητικά σημειώματα για να προσθέσουν κάποια περισσότερα σχετικά δεδομένα. Τα υπομνήματα βοήθησαν τους ερευνητές να αναπτύξουν ιδέες σχετικά με τις έννοιες και τις κατηγορίες και να ανακαλύψουν σχέσεις μεταξύ τους. Η **Δημιουργία Θεωρίας**, το τελευταίο βήμα της ανάλυσης δεδομένων, ήταν η δημιουργία θεωρίας που ονομάζεται «θεωρητική κωδικοποίηση».

Τα **αποτελέσματα** σε αυτή η μελέτη έδειξαν ότι οι άνθρωποι και η νοοτροπία, καθώς και οι συμπεριφορές τους, επηρεάζουν έντονα όλες τις δραστηριότητες που σχετίζονται με τον μετασχηματισμό. Οι σχετικές κατηγορίες και υποκατηγορίες έχουν να κάνουν με:

1. **Εμπόδια αλλαγής**: Οι ανθρώπινες πτυχές καθίσταντο σημαντικά εμπόδια στη διαδικασία αλλαγής στις εταιρείες λογισμικού. Τα εμπόδια είχαν να κάνουν με τα εξής: Αντίσταση στην αλλαγή, Πολιτιστικά θέματα, Έλλειψη συνεργασίας, Λανθασμένη νοοτροπία, Έλλειψη γνώσης.
2. **Επιτάχυνση αλλαγής:** Παρά τις προαναφερθείσες αρνητικές επιπτώσεις των ανθρώπων στη διαδικασία μετασχηματισμού, εμφανίστηκαν αρκετές ενδείξεις σχετικά με τις θετικές επιδράσεις των ανθρώπινων παραγόντων στο ATP. Αυτές είναι: Οι άνθρωποι αγοράζουν, η Διαχείριση αγορών, οι Πρωταθλητές και οι Υποστηρικτές.
3. **Οι αντιλήψεις των ανθρώπων σχετικά με τη διαδικασία αλλαγής:** Αυτή η μελέτη ανακάλυψε ότι οι αντιδράσεις των ανθρώπων στην αλλαγή σχετίζονται κυρίως με τις αντιλήψεις τους σχετικά με αυτή. Αυτές είναι: ο Ενθουσιασμός για την αλλαγή, η Ανησυχία για την αλλαγή, το Αδιάφορο για αλλαγή, το Αίσθημα της πραγματικής ανάγκης για αλλαγή και η Μη ρεαλιστική προσδοκία για ευελιξία.
4. **Παράγοντες κινήτρων:** Η μελέτη ανακάλυψε ότι οι άνθρωποι, ειδικά εκείνοι που δεν είχαν αρκετή βαθιά γνώση σχετικά με τις μεθόδους και τις αξίες του Agile, θα πρέπει να παρακινηθούν και να ενθαρρυνθούν για να εφαρμόσουν Agile. Η παροχή κινήτρων κάνει τους ανθρώπους θετικούς για την αλλαγή.

Τέλος, οι **περιορισμοί** έχουν να κάνουν με την εύρεση της μελέτης που βασίζεται στα δεδομένα, επειδή όλοι οι κώδικες, οι έννοιες, οι κατηγορίες και οι ιδιότητες συλλέχθηκαν

απευθείας από πραγματικά περιβάλλοντα. Αυτό το άρθρο δεν ισχυρίζεται ότι τα ευρήματά του είναι καθολικά, επειδή η πρόσβασή του σε κατάλληλους πόρους περιορίστηκε σε εκείνους τους συμμετέχοντες που είχαν συμμετάσχει εθελοντικά σε αυτήν την έρευνα. Ωστόσο, ισχυρίζεται ότι τα ευρήματά της περιγράφουν και χαρακτηρίζουν την περιοχή που μελετάται.

3.1.10. Παράγοντες επιτυχίας και αποτυχίας που επηρεάζουν την υλοποίηση του έργου χρησιμοποιώντας τη μεθοδολογία ανάπτυξης λογισμικού Agile

Στην «ευέλικτη» ανάπτυξη λογισμικού, υπάρχουν διαφορετικοί παράγοντες πίσω από την επιτυχία και την αποτυχία των έργων. Το παρόν ερευνητικό έγγραφο **αντιπροσωπεύει** τους παράγοντες επιτυχίας, αποτυχίας και μετριασμού στην «ευέλικτη» ανάπτυξη. **Παρουσιάζεται** μια μελέτη περίπτωσης ανάλογα με όλους αυτούς τους παράγοντες μετά την ολοκλήρωση των μικρών έργων. Τα τελικά αποτελέσματα παρατηρούνται με βάση την ανάλυση της αποτελεσματικότητας, της ακρίβειας, της διαχείρισης του χρόνου, της ανάλυσης του κινδύνου και της ποιότητας των προϊόντων του έργου. Τα τελικά αποτελέσματα προσδιορίζονται χρησιμοποιώντας διαφορετικές προσεγγίσεις.

Συγκεκριμένα, στο παρόν ερευνητικό έγγραφο πραγματοποιήθηκε μελέτη περίπτωσης με βάση την ανάπτυξη πέντε έργων. Τα έργα αναπτύχθηκαν από τους μαθητές σε πέντε διαφορετικές ομάδες. Παρατηρήθηκε ότι τα έργα αναπτύχθηκαν χρησιμοποιώντας διαφορετικές αναπτυξιακές προσεγγίσεις, όπως καταρράκτη, σπιράλ και ευέλικτη προσέγγιση, και εξετάστηκε ότι υπήρχαν διαφορές σε μη λειτουργικές δραστηριότητες με διαφορετικές μεθόδους. Η ανάλυση βασίστηκε στην αποτελεσματικότητα, την ακρίβεια, τη διαχείριση του χρόνου, την ανάλυση του κινδύνου και την ποιότητα των προϊόντων του έργου. Τέλος, το **αποτέλεσμα** ήταν ότι στη μεθοδολογία του Scrum τα αποτελέσματα ήταν καλύτερα από το μοντέλο καταρράκτη και σπιράλ, όπου η αποδοτικότητα, η ποιότητα του προϊόντος και η διαχείριση του χρόνου είναι υψηλές.

3.1.11. Οι παράγοντες που επηρεάζουν την επιτυχία των συνεχιζόμενων ευέλικτων προγραμμάτων ανάπτυξης λογισμικού

Δεδομένου ότι η «ευέλικτη» ανάπτυξη λογισμικού βασίζεται στο ανθρώπινο κεφάλαιο για να είναι επιτυχής, η έρευνα **επικεντρώθηκε** στην αποκάλυψη των ανθρωποπαραγόντων που συμβάλλουν στην επιτυχία αυτών των τύπων έργων. Μετά από μια εκτενή ανασκόπηση της βιβλιογραφίας, δημιουργήθηκε ένα μοντέλο παραγόντων που αποδείχτηκε ότι επηρέασαν την επιτυχία των έργων ανάπτυξης «ευέλικτου» λογισμικού, και έτσι η εγκυρότητά τους επανεξετάστηκε στα πλαίσια της χώρας της Πορτογαλίας. Το εννοιολογικό μοντέλο προσέφερε έναν συνοπτικό χαρακτηρισμό της διάστασης των ανθρωποπαραγόντων. Οι μελλοντικοί

ερευνητές καλούνται να επικυρώσουν και να αξιοποιήσουν το έργο της έρευνας, καθιστώντας το μοντέλο πιο λεπτομερές και αξιόπιστο.

Συνολικά 216 επαγγελματίες που ασκούν «ευελιξία» ερωτήθηκαν από μια ποικιλία επιχειρηματικών περιοχών. Τα **αποτελέσματα** που λήφθηκαν έδειξαν ότι η ικανότητα της ομάδας και η συμμετοχή των πελατών μπορούν να εξηγήσουν σε μεγάλο βαθμό τη διαφορά στην επιτυχία του έργου ανάπτυξης λογισμικού. Ωστόσο, δεν κατορθώσαμε να βρούμε στοιχεία που να υποδηλώνουν ότι τα προσωπικά χαρακτηριστικά, η εκπαίδευση και η μάθηση ή η κοινωνική κουλτούρα αποτελούν σημαντικούς παράγοντες σε αυτό το πλαίσιο. Ωστόσο, αυτά τα **ευρήματα ελέγχθηκαν και διασταυρώθηκαν** για την ορθότητά τους και υποστηρίχθηκαν από μια ομάδα εστίασης ποιοτικής προσέγγισης.

Αυτό το ερευνητικό κείμενο **προσφέρει** μια πολύτιμη συμβουλή σε επαγγελματίες που ασκούν «ευελιξία», επί του παρόντος ή που μελλοντικά θα συμμετάσχουν σε ένα «ευέλικτο» πρόγραμμα ανάπτυξης λογισμικού. Σύμφωνα με τα ευρήματα, οι διευθυντές ενθαρρύνονται να επιλέξουν μια ομάδα υψηλής ικανότητας και να προωθήσουν τη συμμετοχή και τη συνεργασία των πελατών, καθώς αυτοί οι παράγοντες είναι πιο πιθανό να οδηγήσουν ένα έργο ανάπτυξης λογισμικού στην επιτυχία. Τέλος, αυτή η έρευνα προσφέρει μια προσέγγιση μεικτής μεθόδου που παρέχει πιο θετικά και στέρεα συμπεράσματα από μια μεμονωμένη μέθοδο. Η εφαρμογή ενός συνδυασμού ποσοτικής και ποιοτικής προσέγγισης μας επέτρεψε να εμβαθύνουμε στην ανάλυση και στα ευρήματα.

3.1.12. Παράγοντες επιτυχίας της ανάπτυξης ευέλικτων συστημάτων πληροφοριών: Μια ποιοτική μελέτη

Στο παρόν ερευνητικό κείμενο πραγματοποιήθηκε μια ποιοτική μελέτη που βασίζεται στη συνέντευξη με 12 συμμετέχοντες προκειμένου να ρίξουμε περαιτέρω φως στους παράγοντες επιτυχίας της «ευέλικτης» υπό Ανάπτυξη Πληροφοριακών Συστημάτων (Information Systems Development – ISD), επειδή οι ποσοτικές μελέτες ανέφεραν αντικρουόμενα αποτελέσματα. Πρώτον, εντοπίστηκαν τέσσερις παράγοντες επιτυχίας από προηγούμενη βιβλιογραφία που αποτελούν την εννοιολογική βάση της έρευνάς μας. Αυτοί οι παράγοντες είναι **η ομάδα, ο πελάτης, η οργάνωση και η επικοινωνία**. Επεκτείναμε τα προηγούμενα ευρήματα ανοίγοντας το πλαίσιο του καθενός από αυτούς τους παράγοντες. Αυτό επετεύχθη με τον εντοπισμό των επί μέρους παραγόντων που επιτρέπουν την «ευέλικτη» επιτυχία των ISD. **Θεωρήθηκε** ότι η αυτονομία είναι το πιο σημαντικό χαρακτηριστικό της ομάδας, ενώ οι «ευέλικτες» αξίες και η διοικητική υποστήριξη είναι ζωτικής σημασίας για το οργανωτικό επίπεδο. Για τον ρόλο των πελατών, η συμμετοχή των πελατών κρίθηκε ως η πιο σημαντική, ενώ για την επικοινωνία, οι πρακτικές συνάντησης υπήρξαν ο πιο συχνά αναφερόμενος υπο-παράγοντας. Σε αντίθεση με

τις άλλες κατηγορίες, σημειώθηκε ότι λιγότεροι παράγοντες για το οργανωτικό επίπεδο αναφέρθηκαν από λιγότερους συμμετέχοντες. Κατά συνέπεια, τέθηκε την ακόλουθη ερευνητική ερώτηση: ***Πώς επηρεάζεται η επιτυχία του ευέλικτου ISD από την ομάδα, τον πελάτη, τον οργανισμό και την επικοινωνία;***

Τα αποτελέσματα χωρίστηκαν στις παρακάτω επί μέρους ενότητες παρουσιάζοντας τα αποτελέσματα για καθέναν από τους τέσσερις παράγοντες επιτυχίας που έχουν προσδιοριστεί προηγουμένως.

- Στην **Ομάδα** τα πιο συχνά αναφερόμενα χαρακτηριστικά της είναι: η αυτονομία, η προσωπική πειθαρχία, το κίνητρο, η συνεχόμενη εκμάθηση και η συνεργασία.
- Στην **Οργάνωση** το Agile SD χρειάζεται διαφορετική οργάνωση από τις παραδοσιακές μεθόδους που βασίζονται: στην Υποστήριξη διαχείρισης, στις Ευέλικτες αξίες και στα οργανωτικά επίπεδα.
- Στον **Πελάτη** τα χαρακτηριστικά πρέπει να είναι τα εξής: η Εστίαση πελατών και εγγύτητα, η Τακτική και έγκαιρη ανατροφοδότηση και η Αμοιβαία κατανόηση και εμπιστοσύνη με την ομάδα ανάπτυξης.
- Στην **Επικοινωνία** οι ερωτηθέντες μας επιβεβαιώνουν ότι η επικοινωνία είναι κρίσιμη για κάθε ευκίνητο έργο ASD. Τα χαρακτηριστικά της επικοινωνίας έχουν να κάνουν: με τις Πρακτικές συνάντησης (π.χ. καθημερινά Scrum) και τον Χώρο του γραφείου που βρίσκεται σε κοινή τοποθεσία.

Ολοκληρώνοντας, η παρούσα έρευνα παρείχε ποιοτικές πληροφορίες για τις βασικές αρχές της «ευέλικτης» υπό ανάπτυξης πληροφοριακών συστημάτων (ASD) και ενέπνευσε βαθύτερες ποιοτικές και ποσοτικές έρευνες στο μέλλον.

3.1.13. Μια ανασκόπηση σχετικά με τους κρίσιμους παράγοντες επιτυχίας της ανάπτυξης λογισμικού Agile

Η δεκαετία μεταξύ 2006 - 2016 παρήγαγε πολλές έρευνες που **επικεντρώθηκαν** στον εντοπισμό των παραγόντων που επηρεάζουν την επιτυχία της «ευέλικτης» ανάπτυξης λογισμικού. Το παρόν ερευνητικό έγγραφο **στοχεύει** να κάνει μια ανασκόπηση των κρίσιμων παραγόντων επιτυχίας της ευέλικτης έρευνας ανάπτυξης λογισμικού της παραπάνω δεκαετίας (2006-2016). Οκτώ προηγούμενες μελέτες έχουν επιλεγεί επειδή χρησιμοποίησαν εμπειρικές μεθοδολογίες για την επικύρωση των Κρίσιμων Παραγόντων Επιτυχίας (ΚΠΕ). Οι επιλεγμένες μελέτες εντόπισαν πολλούς παράγοντες επιτυχίας για «ευέλικτη» ανάπτυξη λογισμικού, από τους οποίους οι οκτώ έχουν επιλεγεί σε αυτό το έγγραφο επειδή προσδιορίστηκαν σε περισσότερες από μία μελέτες. Αυτοί είναι η στρατηγική παράδοσης, η ικανότητα και η κατάρτιση της ομάδας, οι ευέλικτες τεχνικές ανάπτυξης, η εμπλοκή των πελατών, η διαδικασία

διαχείρισης έργων, η οργανωτική κουλτούρα, η επικοινωνία και η υποστήριξη κορυφαίας διαχείρισης. Οι επιλεγμένοι παράγοντες έχουν ταξινομηθεί σε κατηγορίες που περιλαμβάνουν την Τεχνική, την Οργάνωση, τις Διαδικασίες και τα Άτομα.

Η **μελλοντική έρευνα** μπορεί να λειτουργήσει σχετικά με την προτεινόμενη ταξινόμηση και να διερευνήσει πώς αυτοί οι παράγοντες επιτυχίας σχετίζονται μεταξύ τους. Η μελλοντική εργασία μπορεί να μελετήσει τη σημασία αυτών των ΚΠΕ και το βάρος του κάθε παράγοντα. Υπάρχει μια συνεχής έρευνα για τη συλλογή δεδομένων από επαγγελματίες που ασκούν «ευελιξία» σε όλο τον κόσμο για να διερευνήσει τη σημασία των οκτώ προσδιορισμένων παραγόντων επιτυχίας. Ελήφθησαν απαντήσεις από 131 επαγγελματίες που ασκούν «ευελιξία», ενώ διερευνήθηκε η σχέση μεταξύ του αριθμού των ολοκληρωμένων επαναλήψεων στο έργο που αναπτύχθηκε με «ευελιξία» και της σημασίας των παραγόντων επιτυχίας. Τέλος, διερευνήθηκε ο ρόλος της εμπειρίας του ατόμου και της οργανωτικής εμπειρίας ασκώντας «ευελιξία» για την αντίληψη των αναγνωρισμένων παραγόντων επιτυχίας. Χρησιμοποιώντας την ανάλυση των παραγόντων, διερευνήθηκαν οι σχέσεις μεταξύ των οκτώ προσδιορισμένων παραγόντων επιτυχίας.

3.1.14. Μια εμπειρική μελέτη: Κατανόηση παραγόντων και εμποδίων για την εφαρμογή ευέλικτων μεθόδων στη Μαλαισία

Το παρόν ερευνητικό έγγραφο επικεντρώνεται στην εμπειρική εργασία για μια διδακτορική έρευνα. **Κύριος στόχος** της έρευνας είναι να εντοπίσει τους παράγοντες που μπορεί να επιφέρει η υιοθέτηση ή απόρριψη των «ευέλικτων» μεθόδων στη Μαλαισία. Η **συμβολή της μελέτης** είναι να μειωθεί το χάσμα μεταξύ των παραγόντων υιοθέτησης στη χώρα και την περιοχή όπου δεν έχουν υλοποιηθεί αντίστοιχες μελέτες. Ως μέρος αυτής της έρευνας, το αρχικό έργο περιελάμβανε τον προσδιορισμό των τρεχόντων επιπέδων χρήσης και αντίληψης της μεθοδολογίας στη χώρα. Μια πιλοτική μελέτη με τη χρήση ερωτηματολογίου πραγματοποιήθηκε σε δύο γλώσσες, στα Αγγλικά και στα Μαλαισιανά. Τα αρχικά αποτελέσματα έδειξαν ότι από τους 79 ερωτηθέντες άνω του 50% δεν έχουν επίγνωση των «ευέλικτων» μεθοδολογιών. Παρά την άγνοιά τους, οι περισσότεροι από τους ερωτηθέντες πιστεύουν στις «ευέλικτες» μεθοδολογίες. Η συνειδητοποίηση των «ευέλικτων» μεθόδων βρίσκεται επίσης στο ότι σημαντικό ρόλο διαδραματίζει η γλώσσα που χρησιμοποιούν οι ερωτηθέντες, οι οποίοι θωρούν εξίσου κρίσιμους τους τεχνικούς και κοινωνικούς παράγοντες προκειμένου να εκτιμηθούν τα οφέλη των Agile μεθόδων. Αυτό το εύρημα παρέχει ένα πολλά υποσχόμενο περιβάλλον για την εφαρμογή «ευέλικτων» μεθόδων στη χώρα. Η μελέτη αυτή χρησιμοποιείται για την παροχή κατευθυντήριων γραμμών σχετικά με τους παράγοντες και τα εμπόδια που εμφανίζονται με την υιοθέτηση των «ευέλικτων» μεθοδολογιών στη χώρα, όπου υπάρχουν λίγες γνώσεις και μελέτες σχετικά με τις μεθόδους. Τα αποτελέσματα

χρησιμοποιούνται, επίσης, για να καλύψουν το ερευνητικό κενό των ασαφών αποδεικτικών στοιχείων που βρέθηκαν σε προηγούμενες μελέτες.

3.1.15. Συγκεντρωτικός Πίνακας Παραγόντων Επιτυχίας και Περιορισμών

Συνοψίζοντας όλα τα παραπάνω ερευνητικά άρθρα (papers) – μελέτες περίπτωσης, που αφορούν κυρίως τους παράγοντες επιτυχίας και αποτυχίας ή περιορισμών εφαρμόζοντας τις Agile Μεθοδολογίες στις ελληνικές εταιρείες κατασκευής λογισμικού, καταλήγουμε στον παρακάτω συγκεντρωτικό πίνακα.

Paper	Παράγοντες Επιτυχίας - Υποθέτουμε	Παράγοντες Αποτυχίας – Περιορισμοί
1. Προκλήσεις υιοθέτησης ευέλικτων μεθόδων σε ένα δημόσιο οργανισμό	• Ανεβασμένη ευελιξία • Βελτιώσεις στην ικανοποίηση των πελατών του ποιότητα στο προϊόν/υπηρεσία και το κόστος	• Τεκμηρίωση • Εκπαίδευση, εμπειρία και δέσμευση • Επικοινωνία και συνεργασία των ενδιαφερομένων • Ρόλοι ευέλικτου ομαδικού • Θέση των ευέλικτων ομάδων • Νομοθεσία • Πολυπολύπλοκα φαρμακευτικά SW ή ιατρικού υλικού
2. Οι κρίσιμοι παράγοντες επιτυχίας και οι φραγμοί για την εφαρμογή μιας ευέλικτης μεθόδου σε οργανισμό ανάπτυξης λογισμικού βασισμένου σε βιβλιογραφική ανασκόπηση της βιομηχανίας	• Δέσμευση • Συμμετοχή του προσωπικού • Εκπαίδευση • Πόροι • Ομάδες δράσης διαδικασίας • Εμπειρία προσωπικού • Καθοδήγηση • Κρατικές – Αναπτυσσόμενη • Μεθοδολογία εφαρμογής • Παρακολούθηση • Επικοινωνία • Απόδοση επένδυσης • Ενσωμάτωση του SDI	• Δέσμευση • Συμμετοχή του προσωπικού • Εκπαίδευση • Πόροι • Ομάδες δράσης διαδικασίας • Εμπειρία προσωπικού • Καθοδήγηση • Κρατικές – Αναπτυσσόμενη • Μεθοδολογία εφαρμογής • Παρακολούθηση • Επικοινωνία • Απόδοση επένδυσης • Ενσωμάτωση του SDI

3.Προσδιορισμός σημαντικών παραγόντων επιτυχίας στην υιοθέτηση πρακτικών ανάπτυξης ευέλικτων λογισμικών	• **Οι οργανωτικοί παράγοντες**: Πελατοκεντρικά θέματα, Η ώρα απόφασης, Ομαδική διανομή, Μέγεθος ομάδας, Εταιρική κουλτούρα και Σχεδιασμός – έλεγχος • **Οι ανθρώπινοι παράγοντες**: Αρμοδιότητα, Προσωπικά χαρακτηριστικά, Επικοινωνία και διαπραγμάτευση, Κοινωνική κουλτούρα και Εκπαίδευση – μάθηση	
4.Υιοθετώντας – Προσαρμόζοντας τις Agile πρακτικές σε πανεπιστημιακά πλαίσια		• Περιορισμοί προγραμματισμού, • Θέματα επικοινωνίας ομάδας, • Θέματα που σχετίζονται με πελάτες, • Έλλειψη αφοσίωσης, • Προσωπικές δεσμεύσεις • Τεχνικοί περιορισμοί.
5.Παράγοντες που σχετίζονται με την ευελιξία ανάπτυξης λογισμικού επιτυχημένων έργων	• Η οργανωτική κουλτούρα • Η ενδυνάμωση της ομάδας έργου	• Πολυπλοκότητας των θεμάτων και της έλλειψης κατάλληλων ποσοτικών οργάνων στη μεθοδολογική επιτυχία της βιβλιογραφίας, τα ευρήματα αυτής της έρευνας είναι λιγότερο από οριστικά. Συνιστάται επιπλέον έρευνα. • **Δεύτερον**, το μέγεθος του δείγματος παραμένει σχετικά χαμηλό. Ένα μεγαλύτερο μέγεθος δείγματος θα μπορούσε να παρέχει ακριβέστερα στατιστικά στοιχεία. • **Τρίτον**, οι περισσότερες από τις απαντήσεις προέρχονταν από χώρες χαμηλής ισχύος. Αυτό περιόρισε την έρευνα της επίδρασης της απόστασης ισχύος. • **Τέταρτον**, ορισμένα από τα δεδομένα που συλλέχθηκαν για δημογραφικούς σκοπούς (μέγεθος οργανισμού, βιομηχανία, χώρα, εμπειρία του ερωτώμενου και ρόλος) θα μπορούσαν να έχουν ταξινομηθεί ως παράγοντες στο έργο και το περιβάλλον του και να συμπεριληφθούν στην επίσημη ανάλυση.

6.Ένα μοντέλο κατάλληλης έκτακτης ανάγκης για κρίσιμους παράγοντες επιτυχίας για έργα ανάπτυξης λογισμικού. Σύγκριση ευέλικτων και παραδοσιακών μεθοδολογιών βάσει σχεδίου	• Οι οργανωτικοί παράγοντες του πελάτη και η επιτυχία του έργου ανάπτυξης λογισμικού, • Οι Παράγοντες ομάδας και επιτυχία έργου ανάπτυξης λογισμικού, • Οι Παράγοντες πελατών και επιτυχία έργου ανάπτυξης λογισμικού, • Οι Παράγοντες έργου (μεταβλητές μεταβλητές)	
7.Περιεκτήματα και εμπόδια πίσω από την επιτυχημένη ευέλικτη ανάπτυξη - πληροφορίες από τις τρεις εταιρείες εντάσεως λογισμικού στη Φινλανδία	• Βεβαιωθείτε ότι η διαχείριση έχει δεσμευτεί και παρέχει συνεχή υποστήριξη για ευέλικτη ανάπτυξη • Βεβαιωθείτε ότι τόσο η ομάδα, η διοίκηση όσο και όλοι οι ενδιαφερόμενοι έχουν ένα σαφές όραμα, κατανόηση και επίγνωση των ευέλικτων μεθόδων • Δώστε την απαραίτητη ελευθερία στις ομάδες για να προσαρμόσετε τις ευέλικτες μεθόδους ώστε να ταιριάξει καλύτερα στις συγκεκριμένες ανάγκες τους • Κάντε τη συνεχή προσαρμογή του ευέλικτου μοντέλου διαδικασίας και σε οργανωτικό επίπεδο	• Βεβαιωθείτε ότι η Διοίκηση έχει δεσμευτεί και παρέχει συνεχή υποστήριξη για την ευέλικτη ανάπτυξη, • Βεβαιωθείτε ότι η Διοίκηση έχει σαφές όραμα, κατανόηση και επίγνωση των ευέλικτων μεθόδων, • Δώστε την απαιτούμενη ελευθερία στις ομάδες για να προσαρμόσετε τις ευέλικτες μεθόδους για να κάνετε την καλύτερη προσαρμογή στις συγκεκριμένες ανάγκες τους, • Κάντε τη Συνεχή Προσαρμογή του Μοντέλου Διαδικασίας με βάση την Ευελιξία και στο Οργανωτικό Επίπεδο

8.Προσδιορισμός ορισμένων κρίσιμων αλλαγών που απαιτούνται κατά την υιοθέτηση ευέλικτων πρακτικών σε παραδοσιακά προγράμματα ανάπτυξης λογισμικού	• Αλλαγές στην οργανωτική κουλτούρα, • Αλλαγές στο στυλ διαχείρισης, • Αλλαγές στις στρατηγικές διαχείρισης γνώσης, • Αλλαγές στις διαδικασίες ανάπτυξης.	• Σε αυτό το έργο, προσπαθήσαμε να βρούμε αποδεικτικά στοιχεία βάσει ερευνών, • Οι μεταβλητές που χρησιμοποιούνται στη μελέτη είναι κάθε περίπλοκος στη φύση τους, • Λόγω της περιορισμένης διαθεσιμότητας επαρκώς σχετικής συγκεκριμένης βιβλιογραφίας στο συγκεκριμένο πρόβλημα που αντιμετωπίζουμε σε αυτό το άρθρο, τα κομμάτια της βιβλιογραφίας που χρησιμοποιούνται για την έννοια κάθε κατασκευής δεν είναι πάντα επιστημονικά, • Η μεθοδολογία να ζητάμε από λίγα άτομα να γνωμοδοτήσουν χωρίς μια μεσολαβητική μεθοδολογία ομάδας εστίασης έχει τους περιορισμούς της, ακόμα και αν το ερωτηματολόγιο είχε δοκιμαστεί για την εγκυρότητα, τη χρησιμότητά του και αναγνωσιμότητα πριν αποσταλεί για τη συλλογή δεδομένων από το πεδίο, • Σε αυτό το έργο, έχουμε περιοριστεί στην ανάλυση των συγκεντρωτικών δεδομένων που λαμβάνονται από διάφορους ερωτηθέντες διαφορετικών ηπείρων σε όλο τον κόσμο, • Στοχεύσαμε άτομα, αλλά τους ρωτήσαμε για πληροφορίες ομάδας (συμπεριλαμβανομένων άλλων ομάδων δέσμευσης) και για εταιρικά και για κοινωνικά ζητήματα, • Αυτή η μελέτη περιορίζεται στα όρια της υπόθεσης ότι τα δεδομένα που λαμβάνονται σε διαφορετικές λειτουργίες εργασίας είναι εξίσου σημαντικά, • Στην ερευνητική μας μεθοδολογία, λαμβάνουμε τις διάφορες μεταβλητές σε κάθε περιοχή και απλώς λαμβάνουμε τον μέσο όρο τους (διάμεσος)

9.Πώς οι ανθρώπινες πτυχές εντυπωσιάζουν τη μετάβαση και την υιοθέτηση ανάπτυξης λογισμικού Agile	**Α. Ανθρώπινοι Παράγοντες:** • Οι άνθρωποι αγοράζουν, • η Διαχείριση αγορών, • οι Πρωταθλητές και οι Υποστηρικτές. **Β. Οι αντιδράσεις των ανθρώπων στην αλλαγή:** • Ενθουσιασμός για αλλαγή, • η Ανησυχία για την αλλαγή, • το Αδιάφορο για αλλαγή, • το Αίσθημα της πραγματικής ανάγκης για αλλαγή και • η Μη ρεαλιστική προσδοκία για ευελιξία. **Γ. Η παροχή κινήτρων κάνει τους ανθρώπους θετικούς για την αλλαγή**	• Αντίσταση στην αλλαγή, • Πολιτιστικά θέματα, • Έλλειψη συνεργασίας, • Λάθος νοοτροπία, • Έλλειψη γνώσης
10.Παράγοντες επιτυχίας και αποτυχίας που επηρεάζουν την υλοποίηση του έργου χρησιμοποιώντας τη μεθοδολογία ανάπτυξης λογισμικού Agile	**Α. Οργάνωση:** • Συνεργατική κουλτούρα • Εγκαταστάσεις για ευέλικτο περιβάλλον εργασίας **Β. Ομάδα:** • Κίνητρα μεταξύ των μελών της ομάδας • Η ομαδική εργασία μειώνει την πίεση εργασίας και αυξάνει τις γνώσεις • Ισχυρή σχέση μεταξύ των μελών της ομάδας, των ενδιαφερομένων και του πελάτη	**Α. Οργάνωση:** • Ο οργανωτικός πολιτισμός γίνεται κοινωνικά στενός • Πρόβλημα επικοινωνίας σε κατανεμημένο περιβάλλον • Ανακριβείς εκτιμήσεις στα αρχικά στάδια **Β. Ομάδα:** • Έλλειψη γνώσης και εμπειρίας • Έλλειψη διαχειριστικής ικανότητας • Κατανόηση του ζητήματος μεταξύ των μελών της ομάδας • Μια κακή σχέση με τους πελάτες

	Γ. Διαδικασία: • Η ομάδα ακολουθεί την ευέλικτη διαμόρφωση, έργο • διαδικασία διαχείρισης • Καλή επικοινωνία με καθημερινές συναντήσεις • Κατανόηση του επιπέδου του έργου και των νέων θεμάτων του **Δ. Τεχνική:** • Ακολουθώντας απλό σχεδιασμό • Συνεχής αποτύπωση του λογισμικού • Σωστός έλεγχος ολοκλήρωσης • Η ομάδα είναι δυναμικής φύσης και προσαρμόζει νέες τεχνικές ανάλογα με τη ζήτηση **Ε. Τεκμηρίωση:** • Ακριβής ποσότητα τεκμηρίωσης • Η λιγότερη τεκμηρίωση αποτρέπει την επιπλέον προσπάθεια •	**Γ. Διαδικασία:** • Απουσία πελάτη • Ασαφής προγραμματισμός και πεδίο εφαρμογής • Οι ρόλοι του έργου είναι σαφώς καθορισμένοι • Η προτεραιότητα των απαιτήσεων γίνεται το κύριο ζήτημα κατά την ανάπτυξη της διαδικασίας σε μια μικρή ομάδα **Δ. Τεχνική:** • Έλλειψη πόρων • Έλλειψη χρήσης νέων τεχνικών και εργαλείων • Πρακτικές λανθασμένης ευέλικτης προσέγγισης **Ε. Τεκμηρίωση:** • Λιγότερη τεκμηρίωση μπορεί να δημιουργήσει πρόβλημα για τα νέα μέλη
11.Οι παράγοντες που επηρεάζουν την επιτυχία των συνεχιζόμενων ευέλικτων προγραμμάτων ανάπτυξης λογισμικού	• Προσωπικά χαρακτηριστικά, • Εκπαίδευση και μάθηση, • Κοινωνική κουλτούρα, • Ομαδική ικανότητα • Συμμετοχή των πελατών	• Πολυπλοκότητα των μεταβλητών

12.Παράγοντες επιτυχίας της ανάπτυξης ευέλικτων συστημάτων πληροφοριών (ISD): Μια ποιοτική μελέτη	• Στην **Ομάδα** τα πιο συχνά αναφερόμενα χαρακτηριστικό της είναι: **η αυτονομία, η προσωπική πειθαρχία, το κίνητρο, η συνεχόμενη εκμάθηση και η συνεργασία.** • Στην **Οργάνωση** το Agile ISD χρειάζεται διαφορετική οργάνωση από τις παραδοσιακές μεθόδους που βασίζονται: στην **Υποστήριξη διαχείρισης**, στις **Ευέλικτες αξίες** και στα **Οργανωτικά επίπεδα.** • Στον **Πελάτη** τα χαρακτηριστικά πρέπει να είναι τα εξής: **η Εστίαση πελατών και εγγύτητα, στην Τακτική και έγκαιρη ανατροφοδότηση και στην Αμοιβαία κατανόηση και εμπιστοσύνη με την ομάδα ανάπτυξης.** • Στην **Επικοινωνία** τα χαρακτηριστικά της επικοινωνίας έχουν να κάνουν ως εξής: **με τις Πρακτικές συνάντησης (π.χ. καθημερινά Scrum) και τον Χώρο του γραφείου που βρίσκεται σε κοινή τοποθεσία.**	

13.Μια ανασκόπηση σχετικά με τους κρίσιμους παράγοντες επιτυχίας της ανάπτυξης λογισμικού Agile	• Η στρατηγική παράδοσης; • Η ικανότητα και η εκπαίδευση της ομάδας; • Οι τεχνικές ανάπτυξης Agile, • Η συμμετοχή των πελατών, • Η διοίκηση διαχείρισης έργων, • Η οργανωτική κουλτούρα, • Η επικοινωνία και • Η υποστήριξη κορυφαίας διαχείρισης	• Οι συγκρούσεις ανάπτυξης λογισμικού, • Οι διαπροσωπικές επικοινωνιακών διαδικασιών, και • Οι συγκρούσεις μεταξύ των ανθρώπων
14.Μια εμπειρική μελέτη: Κατανόηση παραγόντων και εμποδίων για την εφαρμογή ευέλικτων μεθόδων στη Μαλαισία	• Άνθρωποι • Οργανωτική Δομή • Περιβάλλοντικοί παράγοντες	• Η ποιότητα, • Η παράδοση; • Ο προϋπολογισμός • Οι άνθρωποι, • Ο οργανωτικός παράγοντας και • Οι απαιτήσεις

Οι **παράγοντες επιτυχίας,** έχουν να κάνουν κυρίως με τη Δέσμευση, την Τεκμηρίωση, την Εκπαίδευση – Εμπειρία, την Επικοινωνία – Συμμετοχή των ενδιαφερόμενων, τη Νομοθεσία, τους Πόρους, την Εταιρική Κουλτούρα, την Κοινωνική Κουλτούρα, την Ομαδική Εργασία, το Όραμα, το Στυλ Διαχείρισης – τη Στρατηγική – τις Διαδικασίες Ανάπτυξης και την Πειθαρχία.

Οι **παράγοντες αποτυχίας και οι περιορισμοί** έχουν να κάνουν κυρίως με την Τεκμηρίωση, την Εκπαίδευση – Εμπειρία – Δέσμευση, την Επικοινωνία – Συμμετοχή των Ενδιαφερόμενων, τη Νομοθεσία, την Πολυπλοκότητα του Σχεδιασμού, τους Πόρους, την Καθοδήγηση, το Feedback, τη Μεθοδολογία, την Πολυπλοκότητα των Μεταβλητών, την Έλλειψη ερευνών στον Ελληνικό χώρο και την Αντίσταση στην αλλαγή.

Συμπερασματικά, στην έρευνα μας, οι παραπάνω παράγοντες θα αποτελέσουν τη βάση στην έρευνα μας, έτσι ώστε να αποτελέσουν το μέτρο σύγκρισης των απαντήσεων που θα δοθούν από τους εφαρμοστές των Agile Μεθοδολογιών από τις Ελληνικές Εταιρίας κατασκευής λογισμικού. Έτσι, σύμφωνα με τα παραπάνω, θα είμαστε σε θέση να αποφανθούμε αν η εφαρμογή είναι εφικτή ή όχι στην ελληνική βιομηχανία.

3.2. Τα ερευνητικά ερωτήματα

Με το ξεκίνημα της νέας δεκαετίας, η Ελληνική κοινωνία κλήθηκε να αντιμετωπίσει μια νέα πραγματικότητα. Αυτή της ψηφιακής μετάβασης. Χιλιάδες προϊόντα και υπηρεσίες υπόκεινται πια στις τεχνολογίες και σε ό, τι αυτές προσφέρουν. Τι πραγματικά, όμως, συνέβη για να φτάσουμε εδώ; Πως μετασχηματίστηκε το τεράστιο γραφειοκρατικό κράτος σε νέο ψηφιακό; Πως αντιμετώπισαν οι δημιουργοί τη μετάβαση αυτή; Ήταν άραγε προετοιμασμένοι κατάλληλα, ή όχι; Το σίγουρο είναι πως στη νέα αυτή εποχή, της 4ης Βιομηχανικής Επανάστασης που τώρα διανύουμε με γοργούς ρυθμούς, η ύπαρξη προϊόντων και υπηρεσιών, που είναι αποτέλεσμα ενός λογισμικού, είναι ορατή.

3.2.1. Ποιοι είναι οι συνηθέστεροι περιορισμοί που αντιμετωπίζουν οι Ελληνικές εταιρείες λογισμικού, ενώ ασκούν Agile;

Σύμφωνα με όλα τα παραπάνω άρθρα, και τον συγκεντρωτικό πίνακα, καταλήξαμε στα εξής συμπεράσματα. Ότι οι συνηθέστεροι παράγοντες επιτυχίας και περιορισμοί συγκεντρωτικά έχουν να κάνουν κυρίως με τους ανθρώπινους, οργανωτικούς και περιβαλλοντικούς παράγοντες.

Στους **ανθρώπινους παράγοντες** βρίσκουμε την έλλειψη εκπαίδευσης – συνεχόμενης μάθησης, της εμπειρίας, της δέσμευσης, της επικοινωνίας, της συμμετοχής των ενδιαφερόμενων μελών και της αντίστασης στην αλλαγή. Στους **οργανωτικούς παράγοντες**

βλέπουμε την έλλειψη τεκμηρίωσης, την πολυπλοκότητα του σχεδιασμού, την πολυπλοκότητα των μεταβλητών, την έλλειψη μεθοδολογίας, feedback και καθοδήγησης. Στους **παράγοντες που περιβάλλουν το έργο**, θα αντιμετωπίσουμε την έλλειψη της νομοθεσίας, της διάθεσης πόρων, αλλά και των ερευνών στον ελλαδικό χώρο.

Συμπερασματικά, σύμφωνα με τη βιβλιογραφία και την έρευνα, θα θέσουμε τους δικούς μας παράγοντες, έτσι ώστε να δούμε αν αυτό ανταποκρίνεται και στη δική μας έρευνα και αν έχει πλήρη εφαρμογή και στις ελληνικές εταιρίες λογισμικού.

3.2.2. Ποιες «ευέλικτες» πρακτικές επιλέγουν να ακολουθήσουν οι Έλληνες μηχανικοί λογισμικού, και πώς αυτές οι πρακτικές προσαρμόστηκαν για να ανταποκριθούν στους επιχειρηματικούς στόχους της εταιρείας;

Μελετώντας κανείς τις ευέλικτες μεθοδολογίες, έτσι όπως αναπτύχθηκαν παραπάνω, θα διαπιστώσει μια πληθώρα από νέες πρακτικές. Το αποτέλεσμα αυτού είναι η επιτυχής ολοκλήρωση των έργων λογισμικού.

Οι πιο γνωστές Agile μεθοδολογίες είναι αυτή της Scrum και της XP. Άλλες γνωστές, και λιγότερο χρησιμοποιημένες, είναι αυτές των: Crystal Family, Feature Driven Development (FDD), Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD), Lean Development και η Διαδικασία της Ενοποιημένης Προσέγγισης (Unified Process).

Συμπερασματικά, στη δική μας έρευνα θα θεωρήσουμε γνωστές τις παραπάνω μεθόδους και θα ρωτήσουμε τους συμμετέχοντες ποια μεθοδολογία από τις παραπάνω χρησιμοποιούν για την κατασκευή λογισμικού. Επίσης, πως αυτές οι πρακτικές προσαρμόστηκαν ή όχι για να ανταποκριθούν στους επιχειρηματικούς στόχους της εταιρείας.

3.2.3. Ποιες προσαρμοσμένες πρακτικές θεωρούνται ωφέλιμες και αποτελεσματικές όσον αφορά τα αποτελέσματα, και ποιες όχι;

Έχοντας λάβει υπόψιν τις παραπάνω ευέλικτες πρακτικές, έτσι όπως αναπτύχθηκαν στην παρούσα διπλωματική μέσω της βιβλιογραφίας, αλλά και από το παραπάνω ερευνητικό ερώτημα, εφαρμοζόμενες πια από τους Έλληνες κατασκευαστές, θα εντοπίζουμε τις πρακτικές οι οποίες ήταν ωφέλιμες και αποτελεσματικές για την εταιρεία τους, ενώ τα αποτελέσματά τους ήταν ορατά, εκπληρώνοντας τους επιχειρηματικούς τους στόχους.

Συμπερασματικά, στο τέλος θα έχουμε πραγματοποιήσει μια νέα έρευνα, η οποία εστιάζει στις ελληνικές επιχειρήσεις λογισμικού και στόχευσης, στο να θέσει τις ερωτήσεις στα στελέχη των επιχειρήσεων αυτών, τα αποτελέσματα της οποίας θα αποτελέσουν κίνητρο για περαιτέρω

μελέτη, γνώση του Agile Management και για το τι πραγματικά γίνεται στην Ελλάδα της νέας δεκαετίας κατά το σχεδιασμό και την κατασκευή λογισμικού.

3.2.4. Συμπέρασμα ερευνητικών ερωτημάτων

Οι στόχοι της παρούσας έρευνας, σύμφωνα με τα παραπάνω ερευνητικά ερωτήματα έχουν ως εξής. Αρχικά, θα πρέπει να εντοπίσουμε τους παράγοντες επιτυχίας και τους περιορισμούς που αντιμετωπίζουν οι ελληνικές εταιρείες λογισμικού εφαρμόζοντας Agile. Στη συνέχεια, ποιες Agile μεθόδους χρησιμοποιούν οι Έλληνες κατασκευαστές λογισμικού για την ολοκλήρωση του έργου τους, προσαρμόζοντας τες ή όχι, για να επιτύχουν τους επιχειρηματικούς στόχους της εταιρείας τους. Τέλος, σύμφωνα με την Agile μεθοδολογία που επέλεξαν, θα δούμε αν είχαν θετικά αποτελέσματα με την χρήση της ή όχι.

Αυτό που θα πρέπει να καθορίσουμε εξαρχής και πριν την έναρξη της έρευνας είναι ότι θα πρέπει το ερωτηματολόγιο που θα διανεμηθεί, όπως αναφέρθηκε και παραπάνω, να βασίζεται σε standards. Αυτά θα οριστούν σύμφωνα με τα παραπάνω papers – ερευνητικά ερωτήματα, έτσι ώστε να υπάρξει ένα πλαίσιο, το οποίο να μπορεί να συγκριθεί με τα αποτελέσματά μας. Στα παρακάτω κεφάλαια αναλύεται γενικά η έρευνα και τα στοιχεία της, με βάση τη λεπτομερή περιγραφή.

3.3. Η έρευνα

Η παρούσα διπλωματική εργασία έχει προκύψει μέσω της βιβλιογραφικής έρευνας που πραγματοποιήθηκε κυρίως από τον διεθνή και εγχώριο επιστημονικό χώρο. Μελετήθηκε βιβλιογραφία από άρθρα επιστημονικών περιοδικών, διάφορες δημοσιεύσεις και πρακτικά συνεδρίων, ηλεκτρονικές και διαδικτυακές πηγές, στατιστικές έρευνες, συγγράμματα, διπλωματικές εργασίες (μεταπτυχιακού και διδακτορικού επιπέδου), που έχουν να κάνουν στο ένα μέρος τους με τις «ευέλικτες» μεθοδολογίες, ενώ στο άλλο εντοπίζουμε την εφαρμογή τους. Από την εφαρμογή 14 μελετών περίπτωσης, εντοπίσαμε τους παράγοντες επιτυχίας, αλλά και τους περιορισμούς εφαρμόζοντας τις «ευέλικτες» μεθοδολογίες κατά την διάρκεια ανάπτυξης ενός λογισμικού, με σκοπό την επιτυχή ολοκλήρωση του.

Αρχικά, από τις 14 μελέτες περίπτωσης, υλοποιήθηκε ένας ομαδοποιημένος πίνακας, όπου χωρίστηκαν οι παράγοντες επιτυχίας και οι περιορισμοί κατά την υλοποίηση ενός λογισμικού. Στη συνέχεια, κατηγοριοποιήθηκαν σε 3 κατηγορίες, αυτή των ανθρώπινων παραγόντων, αυτή των οργανωτικών παραγόντων και αυτή των εξωτερικών παραγόντων. Με βάση τα παραπάνω, ορίσαμε μερικούς υποθετικούς παράγοντες, έτσι ώστε να δουλέψουμε πάνω σε αυτούς το ερωτηματολόγιό μας.

Το ερωτηματολόγιό μας που αναπτύχθηκε, χωρίστηκε σε τρεις ενότητες. Στην πρώτη ενότητα ζητούνται οι πληροφορίες του συμμετέχοντα, όπως ηλικία, φύλλο, ρόλος στην εταιρεία, μέγεθος στην εταιρία, κλπ. Στη δεύτερη ενότητα αναπτύσσεται ο βαθμός γνώσης και εξοικείωσης με τις ευέλικτες μεθοδολογίες. Στην τρίτη ενότητα ερευνούμε τους παράγοντες εκείνους που συνεισφέρουν στην επιτυχία των έργων λογισμικού.

Για την υλοποίηση της έρευνας αναζητήθηκαν δεκάδες επιστήμονες, οι οποίοι ανήκουν στον χώρο της ανάπτυξης λογισμικού. Στάλθηκαν, δηλαδή, τα ερωτηματολόγια σε developers λογισμικού, διευθυντές έργων, κλπ. Οι συμμετέχοντες εργάζονται σε διάφορες νομικές μορφές εταιριών, από ατομικές εταιρίες μέχρι ευρωπαϊκές ή παγκόσμιας εμβέλειας εταιρίες, δημόσιους οργανισμούς και με διάφορες εργασιακής μορφής σχέσεις, δηλαδή, freelancers ή συμβάσεις έργου και αόριστης μορφής συνεργασία.

Το ερωτηματολόγιο δημιουργήθηκε με την δωρεάν χρήση του εργαλείου της google, το google forms. Αναπτύχθηκε σε ομοιόμορφη μορφή, εύχρηστη και με σύντομες περιεκτικές ερωτήσεις. Οι προσκλήσεις για τον κάθε συμμετέχοντα στάλθηκαν προσωπικά και οι απαντήσεις συλλέχθηκαν μέσω πάλι του ίδιου εργαλείου των google forms.

Για την ανάλυση των δεδομένων ακολουθήθηκε μια ποιοτική ανάλυση. Δηλαδή, αφού συλλεχθούν τα ερωτηματολόγια θα ξεκινήσει η εξαγωγή των αποτελεσμάτων. Συγκριτικά και με πίνακες, θα ελέγξουμε παράγοντες, όπως είναι η χρήση των μεθόδων, ποιοι από αυτούς επιδρούν θετικά και ποιοι περιορίζουν την εξέλιξη των διαδικασιών ανάπτυξης του λογισμικού για την επιτυχία ολοκλήρωσής του, και στο τέλος θα γίνει ένας σχολιασμός – μια κριτική σύμφωνα με τις ερευνητικές υποθέσεις.

Τέλος, «*Η έρευνα έχει σκοπό να προάγει την επιστημονική γνώση σύμφωνα με διεθνώς αποδεκτές επιστημονικές θεωρίες ή επεξεργασία νέων θεωριών, ικανών να αναγνωριστούν από τη διεθνή επιστημονική κοινότητα. Αποτελεί ταυτοχρόνως κοινωνικό και ατομικό αγαθό. Ως κοινωνικό αγαθό, προάγει την ανθρώπινη γνώση και την καινοτομία και συμβάλλει, έτσι, στη βελτίωση της ποιότητας της ζωής και στην ευημερία του κοινωνικού συνόλου. Αυτή η διάστασή της συνδέεται αναπόσπαστα με την ελευθερία των ερευνητών, χωρίς την οποία αυτή δεν είναι δυνατόν να διεξαχθεί. Υπό αυτήν την έννοια, η ερευνητική εργασία είναι και ατομικό αγαθό, που αποτυπώνεται θεσμικά με την κατοχύρωσή της ως αντικείμενο ατομικού δικαιώματος (ελληνικό σύνταγμα, Διακηρύξεις της UNESCO)*» (Πηγή: https://www.uoc.gr/research-at-uni/ethics). Σύμφωνα με τα παραπάνω ακολουθήθηκαν όλοι οι κανόνες δεοντολογίας για την εκπόνηση της παρούσας εργασίας.

3.4. Περιγραφή της έρευνας και των στοιχείων της

Η έρευνα έχει ως κύριο σκοπό να καταγράψει τον βαθμό επιτυχίας της εφαρμογής των «ευέλικτων» μεθοδολογιών από τις εταιρίες και τους κατασκευαστές λογισμικού στην Ελλάδα. Παρόλο που οι συγκεκριμένες μεθοδολογίες είναι παγκόσμια διαδεδομένες και αποδεκτές, στην ελληνική κοινότητα δεν έχουμε καταγεγραμμένη εικόνα της κατάστασης.

Με αυτή τη σκέψη οργανώσαμε την παρούσα έρευνα στοχεύοντας, αρχικά, να δούμε το επίπεδο γνώσης σε αναλυτές συστημάτων λογισμικού, σε μηχανικούς λογισμικού, σε προγραμματιστές και διαχειριστές έργων, που εργάζονται σε εταιρίες που αναπτύσσουν λογισμικό ή σε εταιρείες πληροφορικής. Αναλυτικά, η έρευνα χωρίστηκε σε τρεις ενότητες. Η πρώτη ενότητα αφορά γενικές πληροφορίες σχετικά με τον ρόλο σας ως συμμετέχοντα στην έρευνα. Η δεύτερη ενότητα παρουσιάζει ερωτήσεις αναφορικά με το βαθμό γνώσης και εξοικείωσης με τις «ευέλικτες» μεθοδολογίες. Τέλος, η τρίτη ενότητα, που είναι και η κύρια ενότητα του ερωτηματολογίου, παρουσιάζει ερωτήσεις που εξετάζουν τη συμβολή διαφόρων παραγόντων των «ευέλικτων» μεθοδολογιών αναφορικά με την επίπτωσή τους στην επιτυχία των έργων ανάπτυξης λογισμικού.

Πραγματοποιήθηκε έρευνα μεταξύ 01 Απρίλη 2021 και 15 Απρίλη 2021. Για την έρευνα δημιουργήθηκε ένα ερωτηματολόγιο, του οποίου η προώθηση έγινε προσωπικά, σε 70 επιστήμονες του χώρου, μέσω του διαδικτύου και των κοινωνικών δικτύων Facebook, LinkedIn, είτε από προσωπικές επαφές μέσω email. Στο Παράρτημα 1 διατίθεται ολόκληρο το ερωτηματολόγιο. Το ερωτηματολόγιο δημιουργήθηκε μέσω του δωρεάν εργαλείου της Google, το Google Forms. Στόχος ήταν να συλλέξουμε όσο το δυνατόν περισσότερες απαντήσεις. Η συλλογή των απαντήσεων έγινε απευθείας από τη φόρμα που διατίθεται από τη Google.

Κλείνοντας, στην παρούσα έρευνα θα παρουσιάσουμε μια ποιοτική μελέτη των παραγόντων που βοηθούν στην επιτυχία της εφαρμογής των «ευέλικτων» μεθοδολογιών. Η μέθοδος που θα χρησιμοποιηθεί για την ανάλυση αυτή θα είναι μέσω της κριτικής που θα γίνει από τα αποτελέσματα. Τεχνικά, το εργαλείο του Google forms, μας δίνει αυτόματα διαγράμματα με τα αποτελέσματα, έτσι ώστε να υλοποιηθεί η ανάλυση. Το πλαίσιο ερμηνείας θα γίνει σύμφωνα με τον Πίνακα 1, αλλά και τη βιβλιογραφία και τις έρευνες που αναπτύχθηκαν παραπάνω, έτσι ώστε να αποτελέσουν οδηγούς για τα δικά μας συμπεράσματα. Οι κύριοι περιορισμοί της έρευνάς μας ήταν ο χρόνος και το εύρος των επαγγελματιών, μιας και οι περισσότεροι είναι γνωστοί επαγγελματίες του χώρου, που ίσως να μην δώσουν μια καθαρή εικόνα στα αποτελέσματα από ότι ένα μεγαλύτερο εύρος επαγγελματιών.

Η σελίδα αυτή είναι σκόπιμα λευκή.

Κεφάλαιο 4: Αποτελέσματα & Προτάσεις

Στο κεφάλαιο αυτό θα επιχειρήσουμε να κάνουμε μια ανάλυση των αποτελεσμάτων της έρευνας. Δηλαδή, το τέταρτο και τελευταίο κεφάλαιο θα αναπτυχθεί ως εξής. Θα ξεκινήσουμε με την ανάλυση των αποτελεσμάτων, τα οποία εξαγάγαμε από την έρευνα. Θα ακολουθήσει μια συζήτηση των αποτελεσμάτων, δηλαδή, τι μας δείχνουν τα αποτελέσματα, πως ερμηνεύονται σύμφωνα με τη θεωρία και ποια από αυτά τα χαρακτηριστικά παρουσιάζουν ιδιαίτερο ενδιαφέρον για περαιτέρω μελέτη. Τέλος, θα κλείσουμε με την κατάθεση προτάσεων για περαιτέρω έρευνα στο μέλλον.

4.1. Ανάλυση αποτελεσμάτων

4.1.1. Οι υποθετικοί παράγοντες

Σε συνέχεια της παραπάνω έρευνας είχαμε ως αποτέλεσμα την συλλογή 31 απαντήσεων. Δεδομένου του περιορισμού του χρόνου, οι 31 απαντήσεις αποτελούν αντιπροσωπευτικό δείγμα για την ανάλυση μας, αφού αγγίζει σχεδόν το 50% των συνολικών συμμετεχόντων.

Για την ανάλυση των αποτελεσμάτων μέσω της παραπάνω μελέτης και ανάλυσης της βιβλιογραφίας και των δεκατεσσάρων μελετών περίπτωσης, θεωρήσαμε 25 υποθετικούς παράγοντες που οδηγούν στην επιτυχία ολοκλήρωσης ενός έργου ανάπτυξης λογισμικού.

Στη συνέχεια εξετάζουμε 25 παράγοντες που υποθετικά συμβάλλουν στην επιτυχή ολοκλήρωση των έργων ανάπτυξης λογισμικού με «ευέλικτες» μεθοδολογίες και είναι οι ακόλουθοι:

ΥΠ1. Συνεχόμενη εκπαίδευση του προσωπικού.

ΥΠ2. Αξιοποίηση συνεργατών με μεγάλη εμπειρία και εξειδίκευση.

ΥΠ3. Επικοινωνία μεταξύ των μελών της ομάδας έργου.

ΥΠ4. Επικοινωνία με τους πελάτες και τους τελικούς χρήστες.

ΥΠ5. Βαθμός συνεργασίας μεταξύ των συμμετεχόντων.

ΥΠ6. Απλότητα στη σχεδίαση.

ΥΠ7. Επανατροφοδότηση στην περίπτωση λαθών και αστοχιών.

ΥΠ8. Αποτελεσματική διοίκηση του έργου.

ΥΠ9. Διαθεσιμότητα ανθρώπινων πόρων.

ΥΠ10. Διαθεσιμότητα οικονομικών πόρων.

ΥΠ11. Δέσμευση, προσήλωση και υπευθυνότητα όλων των συμμετεχόντων στην εκπλήρωση των στόχων του έργου.

ΥΠ12. Τεχνικές γνώσεις και ικανότητες των μελών του έργου.

ΥΠ13. Ικανότητες και δεξιότητες επικοινωνίας.

ΥΠ14. Ικανότητα προσαρμογής στις αλλαγές, τόσο σε αυτές που αφορούν το έργο όσο και σε αυτές που αφορούν τις απαιτήσεις σε σχέση με το λογισμικό.

ΥΠ15. Αναλυτική τεκμηρίωση σε κάθε φάση της ανάπτυξης του έργου.

ΥΠ16. Ύπαρξη τυπικών διαδικασιών διαχείρισης του έργου και ιεραρχικών δομών διοίκησης.

ΥΠ17. Οι απαιτήσεις του προς ανάπτυξη λογισμικού είναι γνωστές εκ των προτέρων σε μεγάλο βαθμό και μεταβάλλονται ελάχιστα κατά τη διάρκεια του έργου.

ΥΠ18.Ύπαρξη καταγεγραμμένων αναλυτικών προδιαγραφών για τις απαιτήσεις του λογισμικού.

ΥΠ19. Ύπαρξη αναλυτικής σχεδίασης για το λογισμικό.

ΥΠ20. Λεπτομερής προγραμματισμός (σχεδίαση - χρονοπρογραμματισμός).

ΥΠ21. Προτεραιοποίηση όλων των απαιτήσεων από την αρχή του έργου.

ΥΠ22. Συνεχόμενη ανατροφοδότηση από τους πελάτες και τους τελικούς χρήστες.

ΥΠ23. Αναλυτικές διαδικασίες ελέγχου (εύρεση και διόρθωση λαθών).

ΥΠ24. Ύπαρξη ελέγχων εγκυρότητας.

ΥΠ25. Καθορισμός ρόλων και ευθυνών των μελών από την αρχή.

Σύμφωνα με όλα τα παραπάνω αναπτύχθηκε και το ερωτηματολόγιο το οποίο χωρίστηκε σε τρεις ενότητες. Στην έρευνα συμμετείχαν επιστήμονες από όλη την Ελλάδα, με διαφορετικές

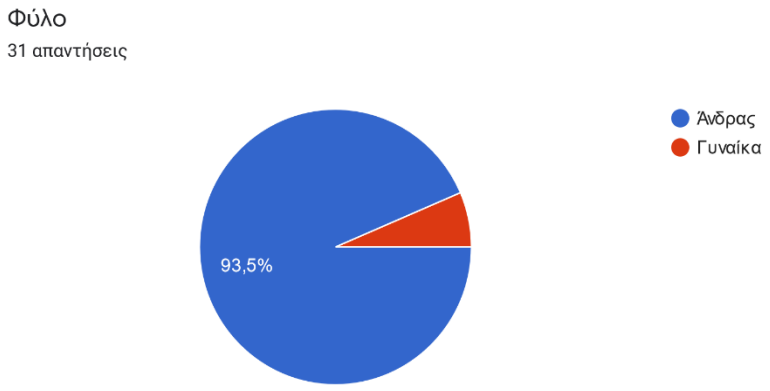
ιδιότητες, στα πιο επιτυχημένα πρότζεκτ πανελλαδικά ή ακόμα διακεκριμένα στην Ευρώπη, παγκόσμια, κλπ. Αυτό αποτελεί ένα κομμάτι από τον δεύτερο περιορισμό. Δηλαδή, οι επαγγελματίες που συμμετείχαν δεν ανήκουν σε ένα ευρύτερο πεδίο του σχεδιασμού – ανάπτυξης λογισμικών, αλλά μονό σε επιτυχημένα πρότζεκτ. Τέλος, τα αποτελέσματα και όλα τα στοιχεία είναι πάντα διαθέσιμα.

4.1.2. Αναλυτικά τα αποτελέσματα

Όπως αναφέρεται παραπάνω, ο σκοπός της παρούσας έρευνας ήταν να καταγραφούν οι παράγοντες επιτυχίας των «ευέλικτων» μεθοδολογιών από τις εταιρείες κατασκευής λογισμικού στην Ελλάδα. Το ερωτηματολόγιο δομήθηκε σε τρεις ενότητες, οι οποίες έδωσαν η καθεμία, ξεχωριστά αποτελέσματα. Αναλυτικά τα αποτελέσματα περιγράφονται παρακάτω.

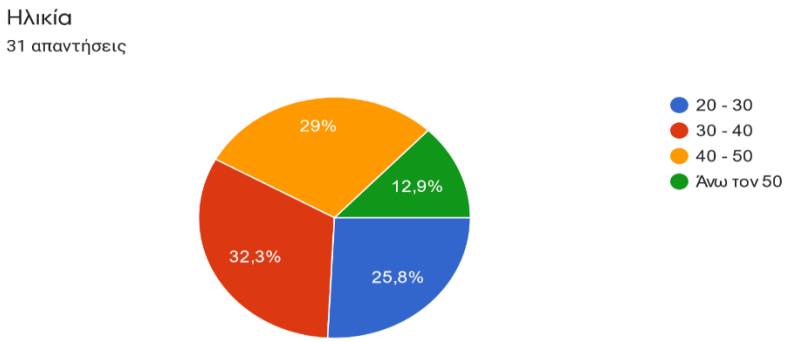
Η **πρώτη ενότητα** του ερωτηματολόγιου αφορά τις **γενικές πληροφορίες του συμμετέχοντα.** Αναλυτικά έχουμε τα εξής:

Ο παράγοντας του **φύλου** από τους 31 συμμετέχοντες, το 93,5% είναι άντρες (29) και το 6,5% είναι γυναίκες (2). Το διάγραμμα παρακάτω αναπαριστά σχηματικά την ποσόστωση αυτή.



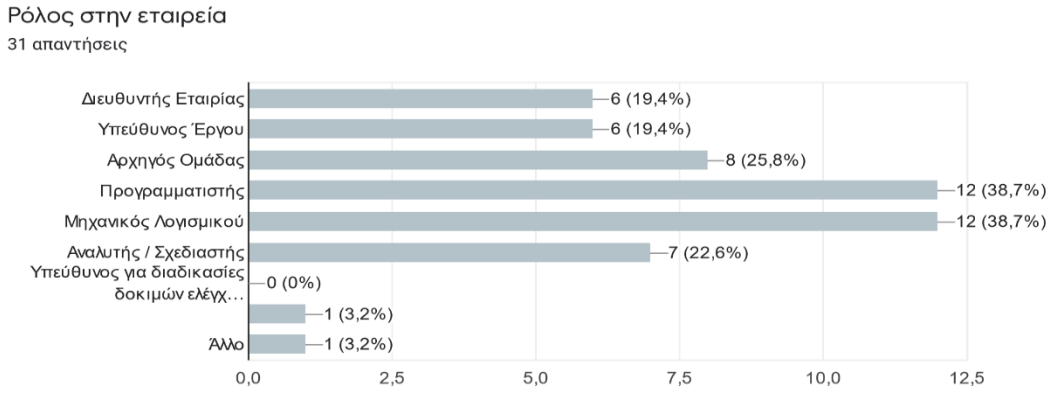
Εικόνα 21 - Γράφημα με την κατανομή του «Φύλου» των συμμετεχόντων

Όσο αφορά τον παράγοντα της **ηλικίας**, μεταξύ 20 – 30 ετών έχουμε το 25,8% (8), μεταξύ 30 – 40 ετών έχουμε το 32,3% (10), μεταξύ 40 – 50 ετών έχουμε το 29% (9) και τέλος, άνω των 50 ετών έχουμε το 12,9% (4) των συμμετεχόντων. Το διάγραμμα παρακάτω αναπαριστά σχηματικά την ποσόστωση αυτή.



Εικόνα 22 - Γράφημα της «ηλικιακής κατανομής» των συμμετεχόντων

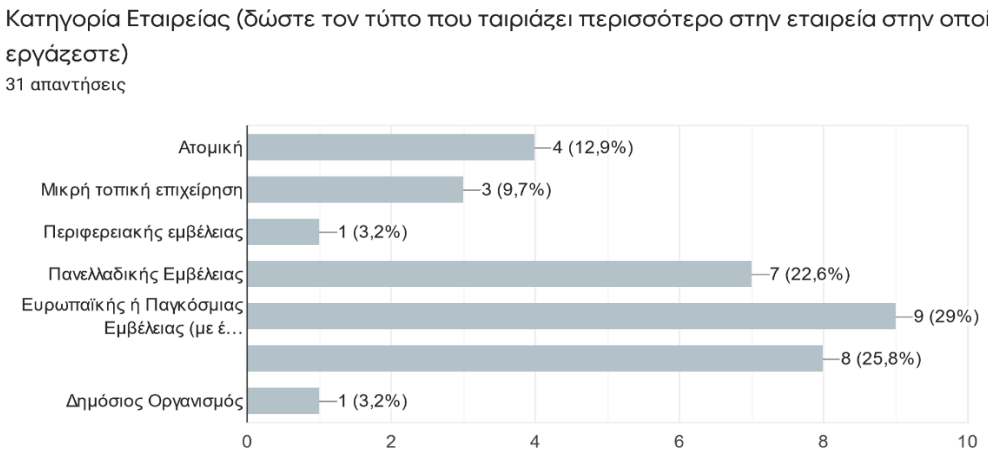
Στη συνέχεια, για το **ρόλο στην εταιρεία** έχουμε Διευθυντή Εταιρείας 19,4% (6), Υπεύθυνο έργου 19,4% (6), Αρχηγό Ομάδας 25,8% (8), Προγραμματιστή 38,7% (12), Μηχανικό Λογισμικού 38,7% (12), Αναλυτή / Σχεδιαστή 22,6% (7), Υπεύθυνο για τις διαδικασίες συντήρησης 3,2% (1) και Άλλο 3,2% (1). Το διάγραμμα παρακάτω αναπαριστά σχηματικά την ποσόστωση αυτή.



Εικόνα 23 - Γράφημα με την κατανομή των «ρόλων στην εταιρεία» των συμμετεχόντων

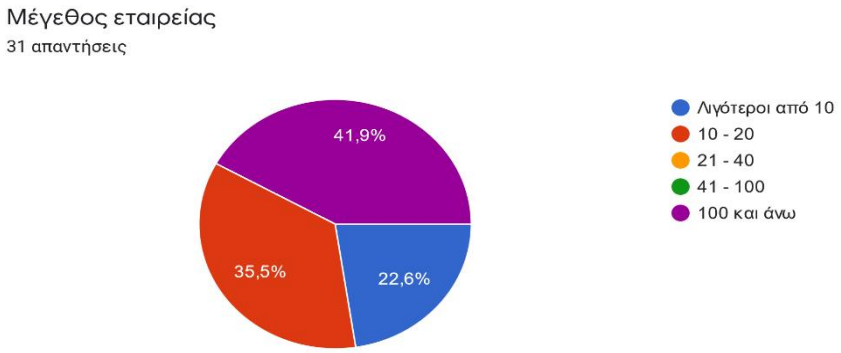
Οι απαντήσεις που δόθηκαν από τους συμμετέχοντες για το **αντικείμενο που πραγματοποιεί η εταιρεία** έχουν ως εξής: Διαχείριση και πώληση ψηφιακών ακουστικών ξεναγήσεων (Self-guided εφαρμογή), IoT – AI – Cloud / Edge Computing, Μάρκετινγκ τράπεζας, Slack bot για ασύγχρονα standup meetings, Smart Software solutions for transforming SME, Web Development, Πληροφορική, Compliance Software for the waste Industry, OnLine delivery, SSoftware platform for booking concerts, Διανομή Προϊόντων Πληροφορικής, Gambling – games, Ανάπτυξη - διαχείριση ιδιόκτητου erp / crm, Πλατφόρμα Λογισμικού ως υπηρεσία για διαχείριση εκδηλώσεων / εκθέσεων, Maritime Software and data analytics, Ανάπτυξη λογισμικού, Mobile Apps, Web / Mobile Development, Advertising – Digital Media, Αυτοματισμοί, Παροχή Υπηρεσιών Logistics, Υπηρεσίες υπολογιστικών συστημάτων Linux & τεχνολογιών Cloud, E-commerce, Έργα ανάπτυξης εκπαιδευτικού λογισμικού, Software Development, Ευρωπαϊκοί οργανισμοί και Finance.

Η **κατηγορία της εταιρείας** στην οποία ανήκει ο κάθε συμμετέχοντας αναλύεται ως εξής: Ατομική εταιρεία 12,9% (4), Μικρή τοπική επιχείρηση 9,7% (3), Περιφερειακής εμβέλειας 3,2% (1), Πανελλαδικής εμβέλειας 22,6% (7), Ευρωπαϊκής ή Παγκόσμιας εμβέλειας με έδρα ή εγκαταστάσεις στην Ελλάδα 29% (9), Ευρωπαϊκής ή Παγκόσμιας εμβέλειας με έδρα ή εγκαταστάσεις εκτός Ελλάδας 25,8% (8) και Δημόσιος Οργανισμός 3,2% (1). Το διάγραμμα παρακάτω αναπαριστά σχηματικά την ποσόστωση αυτή.



Εικόνα 24 - Γράφημα με την κατανομή της «κατηγορίας της εταιρείας» των συμμετεχόντων

Όσον αφορά το **μέγεθος της εταιρείας,** εταιρείες που έχουν λιγότερους από 10 υπαλλήλους αποτελούν το 22,6% (7), εταιρείες με 10 – 20 άτομα αποτελούν το 35,5% (11) και περισσότερα από 100 άτομα αποτελούν το 41,9% (13). Το διάγραμμα παρακάτω αναπαριστά σχηματικά την ποσόστωση αυτή.



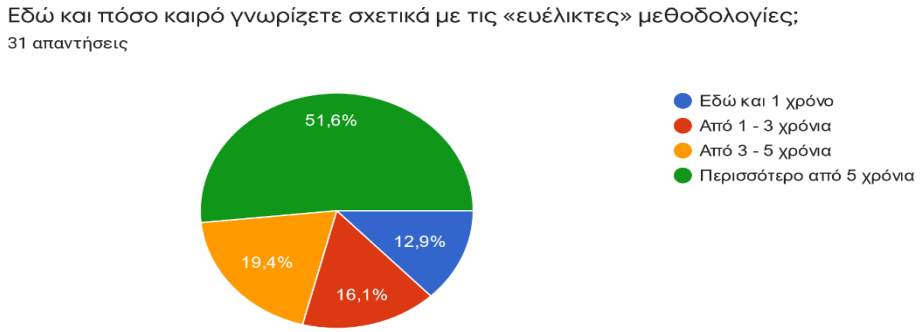
Εικόνα 25 - Γράφημα με την κατανομή του «μεγέθους της εταιρείας» των συμμετεχόντων

Το τελευταίο χαρακτηριστικό της πρώτης ενότητας αφορά το **τυπικό μέγεθος της ομάδας**, έτσι έχουμε ομάδες με λιγότερα από 5 άτομα αποτελούν το 38,7% (12), ομάδες με 5 – 10 άτομα αποτελούν το 35,5% (11), ομάδες με 10 – 20 άτομα αποτελούν το 19,4% (6) και ομάδες άνω των 40 ατόμων αποτελούν το 6,5% (2). Το διάγραμμα παρακάτω αναπαριστά σχηματικά την ποσόστωση αυτή.

Εικόνα 26 - Γράφημα με την κατανομή του «τυπικού μεγέθους της ομάδας» των συμμετεχόντων

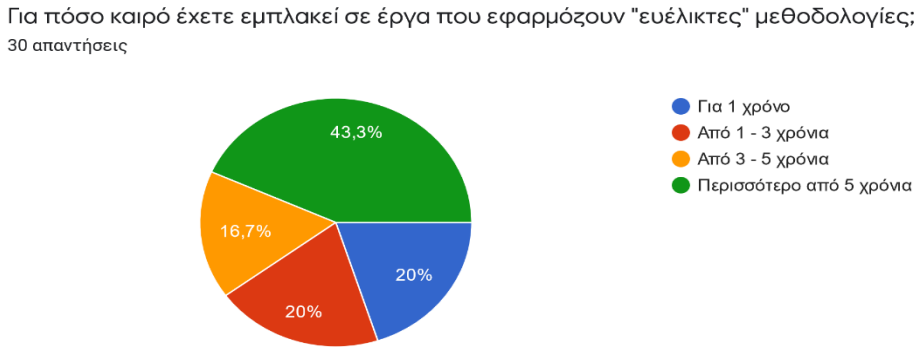
Στη **δεύτερη ενότητα**, αναπτύσσεται ο βαθμός γνώσης και εξοικείωσης των ευέλικτων μεθοδολογιών. Αναλυτικά έχουμε τα εξής χαρακτηριστικά:

Για την **χρονική περίοδο** γνώσης των «ευέλικτων» μεθοδολογιών έχουμε τα εξής: εδώ και 1 χρόνο αποτελεί το 12,9% (4), από 1 – 3 χρόνια αποτελεί το 16,1% (5), από 3 – 5 χρόνια αποτελεί το 19,4% (6) και περισσότερο από 5 χρόνια το 51,6% (16). Το διάγραμμα παρακάτω αναπαριστά σχηματικά την ποσόστωση αυτή.



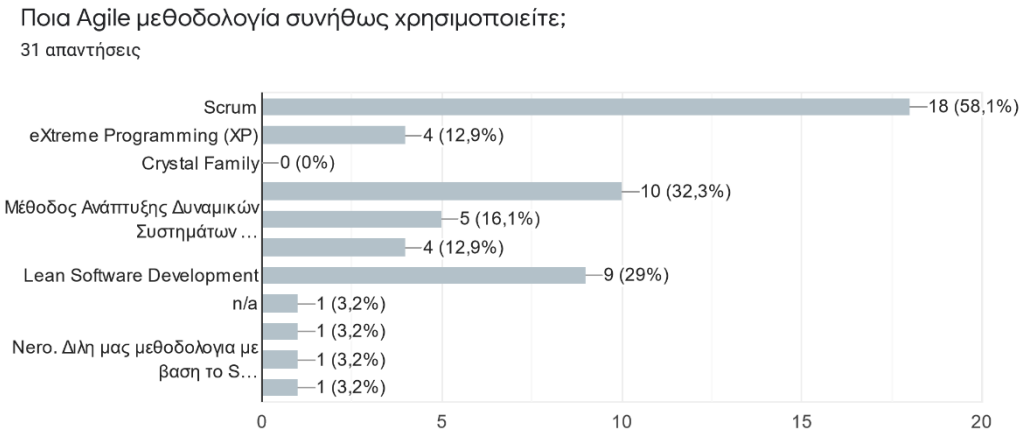
Εικόνα 27 - Γράφημα με την κατανομή του «χρόνου που γνωρίζουν» τις ευέλικτες μεθοδολογίες οι συμμετέχοντες

Για το **χρόνο εφαρμογής – εμπλοκής** τους σε έργα που εφαρμόζουν «ευέλικτες» μεθοδολογίες έχουμε τα εξής: για 1 χρόνο αποτελεί το 20% (6), για 1 – 3 χρόνια αποτελεί το 20% (6), για 3 – 5 χρόνια αποτελεί το 16,7% (5) και για περισσότερο από 5 χρόνια αποτελεί το 43,3% (13). Το διάγραμμα παρακάτω αναπαριστά σχηματικά την ποσόστωση αυτή.



Εικόνα 28 - Γράφημα με την κατανομή «της συμμετοχής σε έργα» που αναπτύσσονται με ευέλικτες μεθοδολογίες από τους συμμετέχοντες

Τελευταίο χαρακτηριστικό αφορά **η μεθοδολογία με την οποία ασχολείται ο κάθε συμμετέχοντας** και έχει ως εξής: Scrum με ποσοστό 58,1% (18), eXtreme Programming (XP) με ποσοστό 12,9% (4), FDD με ποσοστό 32,3% (10), DSDM με ποσοστό 16,1% (5), ASD με ποσοστό 12,9% (4) και με ποσοστό 3,2% (1) οι απαντήσεις Δε ξέρω / Δεν απαντώ, και δικές τους μεθοδολογίες που τις βάφτισαν BDD, Nero και Elements.



Εικόνα 29 - Γράφημα με την κατανομή «της μεθοδολογίας που ακολουθείται» από τον κάθε συμμετέχοντα

Στην **τρίτη ενότητα** ερευνούμε τους παράγοντες εκείνους που συνεισφέρουν στην επιτυχία των έργων λογισμικού. Στον παρακάτω πίνακα δίνονται συγκεντρωτικά οι απαντήσεις των συμμετεχόντων σχετικά με τους παράγοντες που επηρεάζουν θετικά την επιτυχία ενός έργου ανάπτυξης λογισμικού.

Πίνακας 2 - Αποτελέσματα Παραγόντων Επιτυχίας από τους συμμετέχοντες στην έρευνα

ΠΑΡΑΓΟΝΤΕΣ ΕΠΙΤΥΧΙΑΣ	ΠΟΛΥ	ΑΡΚΕΤΑ	ΛΙΓΟ	ΚΑΘΟΛΟΥ	ΔΕ ΓΝΩΡΙΖΩ	ΣΥΝΕΙΣΦΕΡΕΙ ΑΡΝΗΤΙΚΑ
1. Συνεχόμενη εκπαίδευση του προσωπικού.	**58,1**	**32,3**	**9,7**	**0**	**0**	**0**
2. Αξιοποίηση συνεργατών με μεγάλη εμπειρία και εξειδίκευση.	**51,6**	**38,7**	**9,7**	**0**	**0**	**0**
3. Επικοινωνία μεταξύ των μελών της ομάδας έργου.	**87,1**	**12,9**	**0**	**0**	**0**	**0**
4. Επικοινωνία με τους πελάτες και τους τελικούς χρήστες.	**54,8**	**38,7**	**6,5**	**0**	**0**	**0**
5. Βαθμός συνεργασίας μεταξύ των συμμετεχόντων.	**51,6**	**45,2**	**3,2**	**0**	**0**	**0**
6. Απλότητα στη σχεδίαση.	**38,7**	**45,2**	**16,1**	**0**	**0**	**0**
7. Επανατροφοδότηση στην περίπτωση λαθών και αστοχιών.	**45,2**	**51,6**	**3,2**	**0**	**0**	**0**
8. Αποτελεσματική διοίκηση του έργου.	**38,7**	**51,6**	**9,7**	**0**	**0**	**0**
9. Διαθεσιμότητα ανθρώπινων πόρων.	**32,3**	**41,9**	**22,6**	**3,2**	**0**	**0**
10. Διαθεσιμότητα οικονομικών πόρων.	**32,3**	**41,9**	**22,6**	**3,2**	**0**	**0**
11. Δέσμευση, προσήλωση και υπευθυνότητα όλων των συμμετεχόντων στην εκπλήρωση των στόχων του έργου.	**45,2**	**51,6**	**3,2**	**0**	**0**	**0**
12. Τεχνικές γνώσεις και ικανότητες των μελών του έργου.	**45,2**	**54,8**	**0**	**0**	**0**	**0**
13. Ικανότητες και δεξιότητες επικοινωνίας.	**29**	**54,8**	**16,1**	**0**	**0**	**0**
14. Ικανότητα προσαρμογής στις αλλαγές, τόσο σε αυτές που αφορούν το έργο όσο και σε αυτές που αφορούν τις απαιτήσεις σε σχέση με το λογισμικό.	**41,9**	**54,8**	**3,2**	**0**	**0**	**0**
15. Αναλυτική τεκμηρίωση σε κάθε φάση της ανάπτυξης του έργου.	**32,3**	**41,9**	**25,8**	**0**	**0**	**0**
16. Ύπαρξη τυπικών διαδικασιών διαχείρισης του έργου και ιεραρχικών δομών διοίκησης.	**19,4**	**35,5**	**35,5**	**3,2**	**3,2**	**3,2**
17. Οι απαιτήσεις του προς ανάπτυξη λογισμικού είναι γνωστές εκ των προτέρων σε μεγάλο βαθμό και μεταβάλλονται ελάχιστα κατά τη διάρκεια του έργου.	**16,1**	**38,7**	**32,3**	**12,9**	**0**	**0**
18. Ύπαρξη καταγεγραμμένων αναλυτικών προδιαγραφών για τις απαιτήσεις του λογισμικού.	**35,5**	**35,5**	**22,6**	**3,2**	**3,2**	**0**
19. Ύπαρξη αναλυτικής σχεδίασης για το λογισμικό.	**22,6**	**41,9**	**25,8**	**3,2**	**6,5**	**0**
20. Λεπτομερής προγραμματισμός (σχεδίαση - χρονοπρογραμματισμός).	**19,4**	**51,6**	**29**	**0**	**0**	**0**
21. Προτεραιοποίηση όλων των απαιτήσεων από την αρχή του έργου.	**16,1**	**48,4**	**32,3**	**3,2**	**0**	**0**
22. Συνεχόμενη ανατροφοδότηση από τους πελάτες και τους τελικούς χρήστες.	**32,3**	**48,4**	**19,4**	**0**	**0**	**0**
23. Αναλυτικές διαδικασίες ελέγχου (εύρεση και διόρθωση λαθών).	**46,7**	**50**	**3,3**	**0**	**0**	**0**
24. Ύπαρξη ελέγχων εγκυρότητας.	**38,7**	**51,6**	**9,7**	**0**	**0**	**0**
25. Καθορισμός ρόλων και ευθυνών των μελών από την αρχή.	**19,4**	**54,8**	**25,8**	**0**	**0**	**0**
ΜΕΣΟΣ ΟΡΟΣ	**38,01**	**44,572**	**15,5**	**1,284**	**0,516**	**0,128**

Σύμφωνα με τα παραπάνω στοιχεία, έχουμε αναλυτικά την πρώτη εικόνα των αποτελεσμάτων της έρευνάς μας, όπου εντοπίζουμε αν οι παραπάνω παράγοντες βοηθούν τους κατασκευαστές λογισμικού στην Ελλάδα να έχουν επιτυχημένα πρότζεκτ. Η ανάλυση των αποτελεσμάτων της έρευνας αναπτύσσεται παρακάτω.

4.2. Συζήτηση των αποτελεσμάτων

4.2.1. Τι δείχνουν τα αποτελέσματα και πως ερμηνεύονται

Τα σημαντικότερα συμπεράσματα κατά ενότητα της πραγματοποιθείσης έρευνας συνοψίζονται παρακάτω χωριστά.

Στην **πρώτη** ενότητα, έχουμε τα εξής αποτελέσματα – συμπεράσματα:

- Ο **ρόλος στην εταιρεία** έλαβε πολλαπλές – συνδυαστικές απαντήσεις. Ποσοστό 40% από τους συμμετέχοντες έχει τουλάχιστον δυο ή και περισσότερους ρόλους. Σύμφωνα με τις αρχές του Agile Μανιφέστο, την περιγραφή των μεθοδολογιών παραπάνω, και τις μελέτες περίπτωσης, είναι ξεκάθαρο ότι ο κάθε συμμετέχοντας στο έργο αναλαμβάνει ξεκάθαρα ένα ρόλο. Δεν αναφέρεται πουθενά η πολλαπλότητα ρόλων. Επίσης, η πολλαπλότητα ρόλων εντοπίζεται κυρίως στις εταιρείες που είναι μικρές ή μεσαίες, και είναι κύριο χαρακτηριστικό της Ελλάδας, που μπορεί πολλές φορές να οδηγήσει και στην αποτυχία ολοκλήρωσης των έργων.
- Το **αντικείμενο της εταιρείας** μας δείχνει ένα μεγάλο εύρος αντικειμένων εργασίας, το οποίο δείχνει ότι στην Ελλάδα έχουμε συνεχώς αναπτυσσόμενες εταιρείες σε διάφορα αντικείμενα. Επίσης, τα αποτελέσματα που δόθηκαν έχουν μια εγκυρότητα, δηλαδή έχουμε κοινότητα η οποία ασχολείται και εφαρμόζει αποκλειστικά και μόνο τις ευέλικτες μεθοδολογίες.
- Για την **κατηγορία της εταιρείας** έχουμε, επίσης, πολλαπλά – συνδυαστικά χαρακτηριστικά. Εντοπίσαμε ότι σχεδόν το 50% των συμμετεχόντων εργάζονται για μεγάλες εταιρείες Ευρωπαϊκής ή Παγκόσμιας εμβέλειας με εγκαταστάσεις είτε στην Ελλάδα, είτε στο εξωτερικό. Αξιοσημείωτο είναι ότι δυο εκ των συμμετεχόντων έδωσαν πολλαπλές απαντήσεις, πράγμα που πιθανώς ερμηνεύεται από το γεγονός ότι

οι συμμετέχοντες εργάζονται σε πολλά έργα, πράγμα που δείχνει την πιθανή χρήση πολλαπλών μεθόδων «ευέλικτων» μεθοδολογιών ανάπτυξης έργων λογισμικού.

- Το χαρακτηριστικό του **μεγέθους της εταιρείας** μας δείχνει ότι έχουμε μεγάλη διαφορά στο μέγεθος των εταιρειών. Δηλαδή, από τη μια πλευρά έχουμε εταιρείες που είναι μέχρι και 10 άτομα ή και 20 άτομα, και από την άλλη πλευρά το μεγαλύτερο ποσοστό, 43,8% των εταιρειών, απασχολούν περισσότερα από 100 άτομα. Σύμφωνα με το άρθρο «***Challenges of adopting agile methods in a public organisation***», των Nuottila, J., Aaltonen, K. & Kujala, J. (2016,) που αναλύθηκε στο κεφάλαιο 3.1.1., οι μικρές επιχειρήσεις και ομάδες αντιμετωπίζουν αρκετούς περιορισμούς, όπως: Ψυχολογικούς φραγμούς (κοινωνικό άγχος), Διαφορετική ιδεολογία, Οργανωτική ετοιμότητα (εμπλοκή διοίκησης), Συνεχόμενη εκπαίδευση – κατάρτιση, Συνεργασία, κλπ.
- Το χαρακτηριστικό του **τυπικού μεγέθους της ομάδας** μας δίνει τα εξής αποτελέσματα. Το 37,5% των συμμετεχόντων δηλώνουν ότι η ομάδα τους έχει λιγότερο από 5 άτομα, και ένα άλλο 37,5% των συμμετεχόντων δηλώνουν ότι η ομάδα τους έχει 5 – 10 άτομα. Σύμφωνα με τις Agile μεθοδολογίες, το μέγεθος της ομάδας δεν πρέπει να είναι μεγάλο, δηλαδή να μη ξεπερνά τα 8 με 10 άτομα. Άρα, το χαρακτηριστικό αυτό καλύπτει την ορθή εφαρμογή των μεθοδολογιών. Όμως, ένα ποσοστό της τάξεως του 18,8% έχει ομάδες των 10 – 20 ατόμων και ένα ποσοστό της τάξεως του 6,3% έχει ομάδες άνω των 40 ατόμων. Το συνολικό ποσοστό του 25,1% με ομάδες μεγαλύτερες από 10 άτομα, θα πρέπει να είναι ανησυχητικό από τους εφαρμοστές των Agile μεθοδολογιών, διότι οι μεγάλες ομάδες δεν είναι ευκίνητες - ευέλικτες για πολλούς λόγους.

Η **δεύτερη** ενότητα μας δίνει τα εξής αποτελέσματα – συμπεράσματα:

- Το χαρακτηριστικό για το **πόσο καιρό οι συμμετέχοντες γνωρίζουν για τις «ευέλικτες» μεθοδολογίες** μας έδωσε το 100% των απαντήσεων των συμμετεχόντων. Όλοι γνωρίζουν τις «ευέλικτες» μεθοδολογίες, εκ των οποίων το 50% γνωρίζουν για περισσότερο από 5 χρόνια. Το ποσοστό του 18,8% από 1 – 3 χρόνια και 3 – 5 χρόνια, ενώ θα μας προβληματίσει το ποσοστό του 12,5% που γνωρίζουν τις μεθοδολογίες για

λιγότερο από 1 χρόνο, διότι οι μεθοδολογίες, όπως είναι ευρύτερα γνωστές, έχουν εφαρμογή τα τελευταία 20 χρόνια.

- Το χαρακτηριστικό για το **πόσο καιρό εμπλέκονται σε έργα που εφαρμόζουν «ευέλικτες» μεθοδολογίες** μας έδωσε 97% των συμμετεχόντων ότι εμπλέκονται σε έργα που εφαρμόζουν τις μεθοδολογίες, ενώ ένα 3% των συμμετεχόντων δεν απάντησε καθόλου. Το ποσοστό του 3% που δεν συμμετέχει, ερμηνεύεται ως μη εμπλεκόμενο σε έργα πληροφορικής με εφαρμογή τέτοιων μεθόδων είτε λόγω μη γνώσης της εφαρμογής των μεθόδων είτε λόγω της ανάπτυξης έργων με τις παραδοσιακές μεθοδολογίες
- Το χαρακτηριστικό για το **ποια μεθοδολογία χρησιμοποιείται για την ανάπτυξη έργων λογισμικού** μας έδωσε μερικές εκπλήξεις. Αρχικά πρόκειται για ένα χαρακτηριστικό όπου έχουμε πολλαπλότητα στις απαντήσεις. Δηλαδή, σχεδόν το 50% των συμμετεχόντων ασκούν πολλαπλές μεθοδολογίες, και όχι μια. Δεν μας εγγυάται κατά πόσο ωφέλιμο και πρακτικό είναι αυτό. Η πιο διαδεδομένη μεθοδολογία είναι η Scrum με την εφαρμογή της να αγγίζει το 59,4%, σε αντίθεση με την Crystal Family που κανείς από τους συμμετέχοντες δεν την εφαρμόζει (0%). Ακολουθεί η FDD με ποσοστό 31,3%, η Lean με 28,1%, η DSDM με 15,6% και από 12,5% η XP και η ASD. Αξίζει να σημειωθεί ότι έχουμε 3 απαντήσεις από τους συμμετέχοντες που μας ενημερώνουν για την ανάπτυξη και χρήση δικών τους μεθόδων.

Η **τρίτη** ενότητα αποτελεί το κύριο μέρος της έρευνας όπου αναφέρονται χωριστά ένας - ένας οι παράγοντες που οδηγούν στην επιτυχία ενός έργου ανάπτυξης λογισμικού. Εδώ έχουμε τα εξής αποτελέσματα – συμπεράσματα:

- Όπως αναφέρθηκε στην δεύτερη ενότητα, σε ποσοστό 3% οι συμμετέχοντες δεν έχουν εμπλακεί σε έργα ανάπτυξης λογισμικού με χρήση των «ευέλικτων» μεθοδολογιών. Σε αυτή την ενότητα, όμως, συμμετείχαν κανονικά. Εικάζουμε πως οι συμμετέχοντες απάντησαν με βάση την υφιστάμενη κατάσταση και τη συμμετοχή τους σε έργα ανάπτυξης λογισμικού με παραδοσιακές μεθοδολογίες.
- Το 93% των συμμετεχόντων έχουν θετική γνώμη σχετικά με τη χρήση των «ευέλικτων» μεθοδολογιών, και χρησιμοποιώντας τες είτε με την αυτούσια διαδικασία

είτε με διάφορες τροποποιήσεις, ικανοποίησαν τον στόχο της εταιρείας τους για υλοποίηση των έργων με επιτυχία. Αξίζει να σημειωθεί ότι δεν είχαμε καμία αρνητική απάντηση (0%), σε ουδέτερη γνώμη βρίσκεται το 3,1% των συμμετεχόντων, ενώ άλλο ένα 3,1% σχολιάζει ότι θα πρέπει να εξετάζεται κάθε έργο χωριστά για το αν είναι κατάλληλο ή όχι να δεχτεί τέτοιου είδους εφαρμογή και ποιου είδους μεθοδολογία.

- Σύμφωνα με τον συνολικό Πίνακα 1 της τρίτης ενότητα, με όλους τους παράγοντες επιτυχίας έχουμε τα εξής αποτελέσματα. Ο μέσος όρος των παραγόντων επιτυχίας με απόκριση «Πολύ» έχει ποσοστό 38%, το «Αρκετά» έχει το 44,5%, το «Λίγο» έχει το 15%, το «Καθόλου» έχει το 1,28%, το «Δε γνωρίζω» έχει το 0,52% και, τέλος, το «Συνεισφέρει Αρνητικά» έχει το 0,13%. Σύμφωνα με τα παραπάνω αποτελέσματα, οι παράγοντες αυτοί συνεισφέρουν θετικά στην επιτυχία ενός έργου λογισμικού εφαρμόζοντας «ευέλικτες» μεθοδολογίες, μιας και το σύνολο των 82,5% επηρεάζει από «Αρκετά» έως «Πολύ».
- Οι **ανθρώπινοι παράγοντες** που βοηθούν στην επιτυχία ολοκλήρωσης του έργου περιλαμβάνουν τα εξής χαρακτηριστικά:
  - Όσον αφορά τη **συνεχόμενη εκπαίδευση,** το 90,4% των συμμετεχόντων απάντησαν «Αρκετά» έως «Πολύ», ενώ το υπόλοιπο 9,6% απάντησε «Λίγο». Σύμφωνα με τις θεωρίες του «ευέλικτου» μανιφέστου που αναπτύχθηκαν παραπάνω και σύμφωνα με τα 14 μελέτης περίπτωσης, η συνεχόμενη εκπαίδευση παίζει πρωταρχικό ρόλο στην επιτυχία της εφαρμογής. Άρα, το 100% των συμμετεχόντων εκπαιδεύεται στις νέες τεχνολογικές εξελίξεις «Λίγο» έως «Πολύ».
  - Η **επικοινωνία** χωρίζεται σε τρεις ενότητες, και παίζει σημαντικό ρόλο για την επιτυχία των έργων.
    - Η **επικοινωνία μεταξύ των μελών της ομάδας έργων**, όπου το 87,1% των συμμετεχόντων απάντησε «Πολύ», ενώ το υπόλοιπο 12,9% αποκρίθηκε με το «Αρκετά». Άρα, το σύνολο του 100% θεωρεί την επικοινωνία μεταξύ των μελών σημαντική για την επιτυχία, ενώ παράλληλα την εφαρμόζει.

- Η **επικοινωνία μεταξύ πελατών και τελικών χρηστών**, όπου το 54,8% των συμμετεχόντων αποκρίθηκε με το «Πολύ», το 38,7% με το «Αρκετά» και το 6,5% με το «Λίγο». Το σύνολο του 100% θεωρεί την επικοινωνία μεταξύ πελατών και τελικών χρηστών σημαντική για την επιτυχία.
- Οι **ικανότητες και δεξιότητες επικοινωνίας** του κάθε μέλους, όπου το 29% των συμμετεχόντων απάντησε «Πολύ», το 54,8% απάντησε με «Αρκετά», ενώ το 16% απάντησε «Λίγο». Το 16% που απάντησε «Λίγο», ίσως θα έπρεπε να αρχίζει να μας προβληματίζει, διότι η επικοινωνία όπως γνωρίζουμε παίζει σημαντικό ρόλο στην επιτυχία ανάπτυξης των έργων. Και εδώ, όπως φαίνεται, είναι ξεκάθαρο ότι σημαντικό ποσοστό των μελών της ομάδας των συμμετεχόντων ΔΕΝ έχει αναπτύξει ικανοποιητικά τις ικανότητες – δεξιότητες επικοινωνίας.

Η **επικοινωνία στο σύνολο** ως παράγοντας έχει καθοριστικό ρόλο στην επιτυχία, όπως είναι γνωστό. Στη δική μας έρευνα οι συμμετέχοντες στο 100% των απαντήσεων ανταποκρίθηκαν από «Λίγο» έως «Πολύ». Πράγμα που σημαίνει ότι ανταποκρίνεται θετικά στα πρότζεκτ, τα οποία αναπτύσσουν.

- Η **δέσμευση, η προσήλωση και η υπευθυνότητα όλων των συμμετεχόντων στην εκπλήρωση των στόχων του έργου** παίζει σημαντικό ρόλο στην επιτυχία. Έτσι το 45,2% αποκρίθηκε με «Πολύ», το 51,6% αποκρίθηκε με «Αρκετά», ενώ το 3,2% αποκρίθηκε με «Λίγο». Το σύνολο των συμμετεχόντων (100%) πιστεύει ότι η δέσμευση, η προσήλωση και η υπευθυνότητα οδηγούν στην επιτυχία και αυτό φαίνεται από τα αποτελέσματα των απαντήσεων τους.
- Ο **βαθμός συνεργασίας** μεταξύ όλων των συμμετεχόντων παίζει σημαντικό ρόλο στην επιτυχία. Το 51,6% απάντησε «Πολύ», το 45,2% απάντησε «Αρκετά», ενώ το 3,2% με «Λίγο». Το σύνολο των συμμετεχόντων (100%) πιστεύει στη συνεργασία για ένα επιτυχημένο αποτέλεσμα.

- Οι **τεχνικές γνώσεις και ικανότητες των μελών του έργου** είναι από τους πιο σημαντικούς παράγοντες επιτυχίας ανάπτυξης ενός έργου. Το 45,2% απάντησε «Πολύ», ενώ το υπόλοιπο 54,8% απάντησε «Αρκετά». Το σύνολο των συμμετεχόντων του 100% επιβεβαιώνει την καθοριστικότατα του παράγοντα της γνώσης.
- Η **ικανότητα προσαρμογής στις αλλαγές, τόσο σε αυτές που αφορούν το έργο, όσο και σε αυτές που αφορούν τις απαιτήσεις σε σχέση με το λογισμικό**, παίζει σημαντικό ρόλο στην επιτυχία. Το 41,9% απάντησε με «Πολύ», το 54,8% απάντησε με «Αρκετά», ενώ το 3,2% απάντησε με «Λίγο». Άρα, το σύνολο του 100% αναπτύσσουν ικανότητες προσαρμογής στις αλλαγές από «Λίγο» έως «Πολύ».

- **Οι οργανωτικοί παράγοντες** που βοηθούν στην επιτυχία ολοκλήρωσης του έργου περιλαμβάνουν τα εξής χαρακτηριστικά:
  - Η **αναλυτική τεκμηρίωση** σε κάθε φάση της ανάπτυξης του έργου μας έδωσε τα εξής αποτελέσματα. Το 32,3% αποκρίθηκε με το «Πολύ», το 41,9% αποκρίθηκε με «Αρκετά», ενώ το 26% αποκρίθηκε «Λίγο». Το 26% αποτελεί το ¼ των συμμετεχόντων, σημαντικό ποσοστό για την επιτυχία ανάπτυξης ενός έργου. Θα πρέπει να επανεξετάσουμε και να δώσουμε ιδιαίτερη σημασία στον παράγοντα αυτό, για να έχουμε επιτυχία.
  - Η **αξιοποίηση συνεργατών με μεγάλη εμπειρία και εξειδίκευση** αποτελεί σημαντικό κλειδί στην επιτυχία ανάπτυξης έργων λογισμικού. Το 51,6% πιστεύει «Πολύ», το 38,7% πιστεύει «Αρκετά», ενώ το 9,7% πιστεύει «Λίγο». Δηλαδή, το 100% των συμμετεχόντων συμμετέχει σε έργα όπου η αξιοποίηση των μελών παίζει σημαντικό ρόλο.
  - Η **Σχεδίαση** χωρίζεται σε 3 υποενότητες που παίζουν σημαντικό ρόλο στην επιτυχία των έργων. Αυτοί είναι:
    - Η **απλότητα του σχεδιασμού** παίζει σημαντικό ρόλο στην επιτυχία. Οι συμμετέχοντες για τον παράγοντα αυτό απάντησαν «Πολύ» το

38,7%, «Αρκετά» το 45,2%, ενώ «Λίγο» το 16%. Ο παράγοντας αυτός θα πρέπει να μας προβληματίσει, διότι από τα αποτελέσματα είναι φανερό ότι ο σχεδιασμός των προϊόντων στην Ελλάδα συνεχίζει να μην έχει την απόλυτη απλότητα. Θα πρέπει να επανεξεταστεί ο παράγοντας της απλότητας της σχεδίασης.

- Η **ύπαρξη αναλυτικής σχεδίασης για το λογισμικό** είναι ένας ακόμα παράγοντας που θα πρέπει να μας προβληματίσει εξαιτίας των αποτελεσμάτων. Το 22,6% των συμμετεχόντων απάντησε «Πολύ», το 41,9% απάντησε «Αρκετά», το 26% απάντησε «Λίγο», το 3,2% απάντησε «Καθόλου», ενώ το 6,5% απάντησε ότι «Δε γνωρίζει». Το σύνολο του 35,7%, δηλαδή λίγο παραπάνω από το 1/3 των συμμετεχόντων δεν έχει ξεκάθαρη εικόνα για τη σχεδίαση του προϊόντος που αναπτύσσει, οπότε θα πρέπει για γίνει ξεκάθαρο ότι θα πρέπει να υπάρχει αναλυτική σχεδίαση.

- Ο **λεπτομερής προγραμματισμός (για τη σχεδίαση και τον χρονοπρογραμματισμό)**, ο τελευταίος παράγοντας της σχεδίασης μας δίνει αποτελέσματα που θα πρέπει να αναλυθούν περαιτέρω. Αναλυτικά το 19,4% απάντησε «Πολύ», το 51,6% απάντησε «Αρκετά», ενώ το 29% απάντησε «Λίγο». Το 29% που αγγίζει το 1/3 των συμμετεχόντων βλέπει ότι ο προγραμματισμός δεν δουλεύει σωστά, με πιθανό αποτέλεσμα τη μη επιτυχία ολοκλήρωσης του έργου.

Συνολικά, **για τους τρεις παράγοντες της σχεδίασης,** οι οποίοι αποτελούν σημαντικό κλειδί στην επιτυχία ανάπτυξης ενός έργου, απέδειξαν σύμφωνα με τις απαντήσεις τους οι συμμετέχοντες ότι δεν υπάρχει ξεκάθαρη εικόνα. Ενώ, από τις 12 αρχές των «ευέλικτων» μεθοδολογιών είναι ξεκάθαρο ότι ο παράγοντας σχεδίαση παίζει πρωταρχικό και καθοριστικό ρόλο.

- Η **επανατροφοδότηση** και οι **διαδικασίες ελέγχου και εγκυρότητας** χωρίζονται σε τέσσερις ενότητες, διαδραματίζοντας σημαντικό ρόλο στη επιτυχία των έργων. Αναλυτικά έχουμε τα εξής στοιχεία:

  - Η **επανατροφοδότηση στην περίπτωση λαθών και αστοχιών** είναι σημαντική στην επιτυχία ανάπτυξης έργων. Το 45,2% των συμμετεχόντων απάντησε «Πολύ», το 51,6% απάντησε «Αρκετά», ενώ το 3,2% απάντησε «Λίγο». Συνολικά το 96,8% συμμετέχει σε έργα όπου η επανατροφοδότηση είναι συνεχόμενη για την αποφυγή προβλημάτων με σκοπό την επιτυχία.

  - Η **συνεχόμενη ανατροφοδότηση από τους πελάτες και τους τελικούς χρήστες** αποτελεί κλειδί στην επιτυχία ανάπτυξης των έργων. Το 32,3% των συμμετεχόντων αποκρίθηκε με «Πολύ», το 48,4% αποκρίθηκε με «Αρκετά», ενώ το υπόλοιπο 19,3% αποκρίθηκε με «Λίγο». Το σύνολο του 80,7% των συμμετεχόντων έχει συνεχόμενη ανατροφοδότηση με τους πελάτες ακολουθώντας τις αξίες των «ευέλικτων» μεθοδολογιών, έτσι ώστε να έχουμε επιτυχημένα έργα. Θα πρέπει να μας προβληματίσει το ποσοστό του 19% για την πλήρη επιτυχία των έργων.

  - Οι **αναλυτικές διαδικασίες ελέγχου (εύρεση και διόρθωση λαθών)** μας δίνουν τα εξής αποτελέσματα. Το 46,7% των συμμετεχόντων απάντησε «Πολύ», ενώ το 50% απάντησε «Αρκετά», ενώ το 3,3% απάντησε «Λίγο». Συμπερασματικά, το 96,7% του συνόλου των συμμετεχόντων ακολουθεί διαδικασίες ελέγχου. Πράγμα, που μας δείχνει ότι ο παράγοντας αυτός οδηγεί στην επιτυχία.

  - Η **ύπαρξη ελέγχων εγκυρότητας** έδωσε τις εξής πληροφορίες. Το 38,7% των συμμετεχόντων απάντησε «Πολύ», το 51,6% απάντησε «Αρκετά», ενώ το 9,7% απάντησε «Λίγο». Το σύνολο του 90,3% βρίσκεται σε πρότζεκτ με ανεπτυγμένους ελέγχους εγκυρότητας.

Οι υπο παράγοντες του **ελέγχου** και της **εγκυρότητας** μας έδωσαν θετικά αποτελέσματα, έτσι ώστε να έχουμε επιτυχημένα έργα. Η **συνεχόμενη ανατροφοδότηση από τους πελάτες και τους τελικούς χρήστες** αποτελεί κλειδί για την απόλυτη επιτυχία.

- Η **Διοίκηση** χωρίζεται σε τρεις υποενότητες, παίζοντας καθοριστικό ρόλο στην επιτυχία των έργων.
  - Η **Αποτελεσματική διοίκηση των έργων** είναι ένας παράγοντας ο οποίος βοηθά στην οργάνωση και, κυρίως, στην επιτυχία των έργων. Το 38,7% των συμμετεχόντων απάντησε «Πολύ», το 51,6% απάντησε «Αρκετά», ενώ το 9,7% απάντησε «Λίγο». Στο σύνολο του, το 100% των συμμετεχόντων συμφωνεί και έχει αποτελεσματική διοίκηση στα έργα του.
  - Ο **Καθορισμός ρόλων και ευθυνών των μελών από την αρχή** θα πρέπει να είναι ξεκάθαρος από την αρχή για την επιτυχία των έργων. Οι συμμετέχοντες απάντησαν με 19,4% το «Πολύ», το 54,8% απάντησε «Αρκετά», ενώ το 26% απάντησε «Λίγο». Το ποσοστό του 26% που αποτελεί το ¼ των συμμετεχόντων, δεν έχει ξεκάθαρο το ρόλο του. Δηλαδή, οι μικρές επιχειρήσεις πιθανόν να δίνουν πολλαπλούς ρόλους στους συμμετέχοντες, που ίσως και να προκαλεί σύγχυση στο ρόλο και στις ευθύνες.
  - Η **Ύπαρξη τυπικών διαδικασιών διαχείρισης του έργου και των ιεραρχικών δομών διοίκησης** είναι καθοριστικός παράγοντας επιτυχίας για την ανάπτυξη ενός έργου. Το 19,4% των συμμετεχόντων αποκρίθηκε «Πολύ», το 35,5% αποκρίθηκε «Αρκετά», το 36% αποκρίθηκε «Λίγο», ενώ από 3,2% αποκρίθηκε τα «Καθόλου», «Δε γνωρίζω» και «Συνεισφέρει Αρνητικά». Το σύνολο του 45,6% που αγγίζει ½ των συμμετεχόντων μαρτυρεί ότι ΔΕΝ υπάρχουν διαδικασίες διαχείρισης και δομές διοίκησης. Αυτό θα πρέπει να μας προβληματίσει και να επανακαθορίσουμε τον παράγοντα αυτό.

Άρα, η **Διοίκηση** σαν οργανωτικός παράγοντας, θα πρέπει να επαναξετεταστεί εξαρχής, γιατί τα αποτελέσματα των συμμετεχόντων μας δείχνουν ότι δεν είναι ξεκάθαρος ο ρόλος της.

- Οι **απαιτήσεις και προδιαγραφές** χωρίζονται σε τρεις ενότητες που διαδραματίζουν σημαντικό ρόλο στην επιτυχία των έργων.
  - Οι **απαιτήσεις του προς ανάπτυξη λογισμικού είναι γνωστές εκ των προτέρων σε μεγάλο βαθμό και μεταβάλλονται ελάχιστα κατά τη διάρκεια του έργου**: είναι καθοριστικός παράγοντας στην επιτυχία των έργων. Το 16,1% των συμμετεχόντων απάντησε «Πολύ», το 38,7% απάντησε «Αρκετά», το 32% απάντησε «Λίγο», ενώ το 12,9% απάντησε «Καθόλου». Το σύνολο του «Λίγο και Καθόλου» μας δίνει το άθροισμα του 44,9% που αγγίζει σχεδόν το ½ των συμμετεχόντων. Το αποτέλεσμα αυτό θα πρέπει να μας προβληματίζει αρκετά, διότι, σύμφωνα με τις αρχές του «Ευέλικτου» Μανιφέστου, θα πρέπει οι απαιτήσεις να είναι καθορισμένες εξ' αρχής και να μεταβάλλονται σύμφωνα με τις νέες ανάγκες. Η μη μεταβλητότητα είναι δείγμα παραδοσιακών μεθόδων ανάπτυξης.
  - **Η ύπαρξη καταγεγραμμένων αναλυτικών προδιαγραφών για τις απαιτήσεις του λογισμικού** παίζει σημαντικό ρόλο στην επιτυχία των έργων. Το 35,5% των συμμετεχόντων απάντησε «Πολύ» και «Αρκετά», το 23% απάντησαν «Λίγο» και το 3,2% απάντησε «Καθόλου» και «Δε γνωρίζω». Το σύνολο του 29,4% που αποτελεί σχεδόν το 1/3 των συμμετεχόντων ΔΕΝ έχει ξεκάθαρη ή καθόλου εικόνα των προδιαγραφών, πράγμα που δεν βοηθά στην επιτυχία του πρότζεκτ, όπως είναι γνωστό από τις αρχές του «ευέλικτου» μανιφέστου.
  - **Η προτεραιοποίηση όλων των απαιτήσεων από την αρχή** είναι καθοριστικός παράγοντας στην επιτυχία. Το 16,1% των συμμετεχόντων απάντησε «Πολύ», το 48,4% απάντησε «Αρκετά», το

32% απάντησε «Λίγο», ενώ το 3,2% απάντησε «Καθόλου». Και σε αυτό τον παράγοντα εντοπίζουμε ότι το 35,2% που ισοδυναμεί με λίγο παραπάνω από το 1/3 των συμμετεχόντων δεν έχει γνώση των προτεραιοτήτων.

Συνολικά, **οι απαιτήσεις και προδιαγραφές**, ως παράγοντες, παίζουν καθοριστικό ρόλο στην επιτυχία ανάπτυξης των έργων. Στην δική μας έρευνα είναι ξεκάθαρο ότι και οι τρεις υπο παράγοντες αυτού δεν έχουν τα επιθυμητά αποτελέσματα, σύμφωνα με την έρευνα. Γι' αυτό και θα πρέπει να επανεξετάσουμε εξαρχής τους παράγοντες αυτούς.

- Οι **εξωτερικοί παράγοντες** που βοηθούν στην επιτυχία ολοκλήρωσης του έργου περιλαμβάνουν τα εξής χαρακτηριστικά:
    - Η **διαθεσιμότητα των ανθρώπινων και οικονομικών πόρων** αποτελούν σημαντικούς παράγοντες στην επιτυχία ανάπτυξης ενός έργου λογισμικού. Το 32,3% απάντησε «Πολύ», το 41,9% απάντησε «Αρκετά», το 23% απάντησε «Λίγο», ενώ το 3,2% απάντησε «Καθόλου». Άρα στο σύνολο του 74,2% διατίθεται μεγάλο ποσοστό των ανθρώπινων και οικονομικών πόρων, ενώ το 25,8%, δηλαδή στο ¼ των συμμετεχόντων δεν διατίθενται οι πόροι, που παίζουν σημαντικό ρόλο στην επιτυχία. Θα πρέπει να επανεξετάσουμε τους δυο αυτούς παράγοντες, έτσι ώστε να έχουμε το 100% της επιτυχίας.

4.2.2. Μεμονωμένα αποτελέσματα που χρήζουν περαιτέρω – ιδιαίτερης ερμηνείας

Σε συνέχεια της έρευνάς μας αξίζουν να σημειωθούν μερικά χαρακτηριστικά – παράγοντες τα οποία μας έδωσαν ιδιαίτερα αποτελέσματα και χρήζουν ιδιαίτερης προσοχής. Είτε για περαιτέρω μελέτη, είτε γιατί οδηγούν απευθείας στην αποτυχία ανάπτυξης των έργων.

Το χαρακτηριστικό της **κατηγορίας της εταιρείας,** όπως αναφέρεται παραπάνω μας έδωσε ότι το 50% και παραπάνω των συμμετεχόντων απασχολείται σε εταιρείες μεγάλες είτε σε μεγάλα έργα. Αυτό θα πρέπει να μας προβληματίσει, διότι το χαρακτηριστικό αυτό αγγίζει και τους δυο περιορισμούς, αυτόν του εύρους των συμμετεχόντων και αυτόν της χρονικής διάθεσης της

έρευνας. Όπως αναφέρθηκε παραπάνω, η έρευνά μας απευθύνθηκε στο ευρύτερο περιβάλλον του ερευνητή, σε μικρό χρονικό διάστημα. Αυτό είχε το αποτέλεσμα όλοι οι συμμετέχοντες να εργάζονται σε μεγάλες εταιρείες, οι οποίες αναπτύσσουν έργα με «ευέλικτες» μεθοδολογίες. Δεν έχουμε, όμως, καθόλου εικόνα για το τι συμβαίνει σε μικρότερου εύρους επιχειρήσεις, και τι γίνεται με την εφαρμογή των μεθοδολογιών. Ταυτόχρονα, δυο από τους ερωτηθέντες, όπως διαπιστώσαμε, συμμετέχουν ταυτόχρονα σε περισσότερες από μια εταιρείες και έργα. Θα πρέπει να μας προβληματίσει και αυτό ιδιαίτερα, διότι μας είναι γνωστός, από τις αρχές του «ευέλικτου» μανιφέστου, ο ξεκάθαρος ρόλος ενός συμμετέχοντα σε ένα έργο για την ανάπτυξη των ικανοτήτων του πλήρως, κλπ. Παράλληλα, αυτό μπορεί να έχει και θετική επίδραση, εξαιτίας της περαιτέρω διεύρυνσης της γνώσης μέσω της χρήσης πολλαπλών «ευέλικτων» μεθόδων και απόκτησης σημαντικής εμπειρίας.

Το χαρακτηριστικό της **εφαρμογής των «ευέλικτων» μεθοδολογιών** χρήζει ιδιαίτερης ερμηνείας. Όπως αναφέρθηκε παραπάνω, για το πόσο καιρό γνωρίζετε τις «ευέλικτες» μεθοδολογίες, οι συμμετέχοντες στο 100% αποκρίθηκαν ότι γνωρίζουν τις μεθοδολογίες. Σε αυτή την ερώτηση αποκρίθηκε το 97% με τη συμμετοχή, ενώ το ποσοστό του 3% δεν απάντησε. Αυτό ερμηνεύεται πως ο συμμετέχοντας γνωρίζει τις μεθοδολογίες, αλλά στα έργα που συμμετέχει τα αναπτύσσει πιθανά με τις παραδοσιακές μεθόδους, ρισκάροντας την επιτυχία του αποτελέσματος.

Το χαρακτηριστικό **της μεθοδολογίας της οποίας ακολουθούν οι συμμετέχοντες** μας έδωσε αποτελέσματα, τα οποία μας κίνησαν το ενδιαφέρον. Δηλαδή, πέραν των κλασσικών απαντήσεων, έχουμε και τα εξής χαρακτηριστικά, τα οποία χρήζουν περαιτέρω διερεύνησης. Αρχικά, το 50% των συμμετεχόντων απάντησε ότι χρησιμοποιεί περισσότερες από μια μεθοδολογίες. Αυτό θα πρέπει να μας κάνει να αναρωτηθούμε αν και πόσο αυτό είναι ωφέλιμο και πρακτικό για εκείνους και την επιτυχία ολοκλήρωσης των έργων που συμμετέχουν. Στη συνέχεια, το 3% των συμμετεχόντων που αναφέρθηκαν αμέσως παραπάνω, επιβεβαιώνει πλήρως ότι ΔΕ χρησιμοποιεί «ευέλικτες» μεθοδολογίες, μιας και σε αυτή την ενότητα δεν απάντησαν καθόλου. Πιθανώς θα γίνεται χρήση παραδοσιακών μεθόδων ανάπτυξης έργων, ρισκάροντας την επιτυχία των έργων τους. Τέλος, θα πρέπει να μας προβληματίσει ιδιαιτέρα το εξής. Το 10% των συμμετεχόντων απάντησε ότι στις εταιρείες τους έχουν υιοθετήσει δικά

τους πρότυπα – πρακτικές – μεθοδολογίες και διαδικασίες ανάπτυξης λογισμικού, δηλαδή δεν ακολουθούν τυπικά τις μεθοδολογίες, και οι οποίες έχουν αποδειχτεί ότι ίσως έχουν κάποια αποτελέσματα. Αυτό χρήζει ιδιαίτερης προσοχής διότι δεν ξέρουμε κατά ποσό αυτό βοηθά και πιστοποιεί την επιτυχία των έργων, ενώ, παράλληλα, σε περίπτωση που η μεθοδολογία τους χρήζει επιτυχίας, θα ήταν καθοριστική και σημαντική να προστεθεί στην εφαρμογή της ευρύτερα.

Ο παράγοντας **της αναλυτικής τεκμηρίωσης** μας έδωσε ένα σημαντικό αποτέλεσμα, αυτό του 26%, δηλαδή του ¼ των συμμετεχόντων, που αποκρίθηκαν ότι υφίσταται «Λίγη» τεκμηρίωση στα έργα τα οποία αναπτύσσουν. Σύμφωνα με το άρθρο «***Success and Failure Factors that Impact on Project Implementation Using Agile Software Development Methodology***», των Dhir, S., Kumar, D. & Singh, V. B. (2019), που αναφέρθηκε παραπάνω, στο κεφάλαιο 3.1.10, μας δόθηκαν τα εξής στοιχεία. Η τεκμηρίωση παίζει καθοριστικό ρόλο στην επιτυχία των έργων, και αυτό γιατί η **λιγότερη τεκμηρίωση** αποτρέπει την επιπλέον προσπάθεια, μπορεί να δημιουργήσει προβλήματα στα μέλη, ή ακόμα να τα κάνει να μην εξοικειωθούν εύκολα με ένα νέο έργο και τις απαιτήσεις του.

Ο παράγοντας του **σχεδιασμού (Απλότητα σχεδιασμού, Ύπαρξη αναλυτικής σχεδίασης & Λεπτομερή Προγραμματισμού)**, σύμφωνα με την αρχή του «ευέλικτου» μανιφέστου για «Συνεχή προσοχή στην τεχνική αρτιότητα και καλό σχεδιασμό με ενισχυμένη ευελιξία» και με το άρθρο «***Success and Failure Factors that Impact on Project Implementation Using Agile Software Development Methodology***» των Dhir, S., Kumar, D. & Singh, V. B. (2019), που αναφέρθηκε παραπάνω στο κεφάλαιο 3.1.10, που αναφέρθηκαν παραπάνω, ξεκαθαρίζουν ότι ο σχεδιασμός είναι καθοριστικός παράγοντας για την επιτυχία ανάπτυξης των έργων. Στην έρευνά μας είχαμε τα εξής χαρακτηριστικά. Για την «**Απλότητα στο Σχεδιασμό**» το 16% των συμμετεχόντων απάντησε ότι υπάρχει «Λίγη» απλότητα στο σχεδιασμό των έργων που συμμετέχουν και αναπτύσσουν. Για την «**Ύπαρξη Αναλυτικής Σχεδίασης**», το 35,7% απάντησε ότι έχει αρνητική γνώμη. Δηλαδή, τα έργα στα οποία συμμετέχουν και αναπτύσσουν δεν έχουν σχεδίαση. Για τον «**Λεπτομερή Προγραμματισμό**» το 29% των συμμετεχόντων απάντησε ότι υπάρχει «Λίγος» προγραμματισμός στα έργα τους. Συμπερασματικά, **ο παράγοντας της σχεδίασης στο σύνολο** παίζει καθοριστικό ρόλο για την επιτυχία των έργων

χρησιμοποιώντας ευέλικτες μεθοδολογίες. Στην δική μας έρευνα, όπως είδαμε παραπάνω, και οι τρεις τομείς του σχεδιασμού χρήζουν περαιτέρω ανάλυσης, διότι οι συμμετέχοντες αποκρίθηκαν αρνητικά και στα τρία σκέλη του σε μεγάλο ποσοστό.

Στη συνέχεια, ο παράγοντας της **Διοίκησης (Αποτελεσματική Διοίκηση, Καθορισμός Ρόλων & Ευθυνών και Ύπαρξη τυπικών διαδικασιών & Ιεραρχικών Δομών)** θα πρέπει να μας προβληματίσει. Σύμφωνα με το άρθρο «***Success Factors of Agile Information Systems Development: A Qualitative Study***» των Hummel, M. & Epp, A. (2015) που αναπτύχθηκε παραπάνω στο κεφάλαιο 3.1.12, αλλά και των υπόλοιπων περιπτώσεων και θεωριών, η διοίκηση έχει διαφορετική οργάνωση από τις παραδοσιακές μεθόδους, που βασίζεται στην Υποστήριξη της Διοίκησης, στην «Ευελιξία» και στην γενική οργάνωση. Για την «**Αποτελεσματική Διοίκηση**» οι συμμετέχοντες σε απόλυτη σχεδόν πλειοψηφία απάντησαν ότι συμμετέχουν σε εταιρείες όπου η Διοίκηση παίζει σημαντικό και καθοριστικό ρόλο, πλην του ποσοστού του 10% που απαντά για «Λίγη» αποτελεσματικότητα των διοικούντων τους. Για τον «**Καθορισμό ρόλων και ευθυνών των μελών από την αρχή**», το 26% των συμμετεχόντων, δηλαδή το 1/4 αυτών, απάντησε πως γνωρίζει «Λίγο» τον ρόλο, ενώ το 54,8% απάντησε «Αρκετά». Σύμφωνα με τα αποτελέσματα αυτά το 80,8% στο σύνολο ΔΕΝ έχει ξεκάθαρο το ρόλο του στο έργο σε σχέση με το 19,2% που είναι πλήρως κατατοπισμένο στο έργο. Για την «**Ύπαρξη τυπικών διαδικασιών διαχείρισης του έργου και των ιεραρχικών δομών διοίκησης**» οι συμμετέχοντες έχουν αρνητική άποψη. Το 36% απάντησε πως υπάρχουν «Λίγες» διαδικασίες διαχείρισης και δομές διοίκησης, ενώ από 3,2% (δηλαδή το 9,6%) μοιράζονται το «Καθόλου», το «Δε γνωρίζω» και το «Συνεισφέρει Αρνητικά». Άρα, το σύνολο του 45,6% έχει εντελώς αρνητική άποψη, και μαζί με το 35,5% των συμμετεχόντων που πιστεύει ότι υπάρχουν «Αρκετά», οδηγούν στο ότι το 81,1% δεν είναι σίγουρο για τις διαδικασίες. Συμπερασματικά, και εδώ ο **παράγοντας της Διοίκησης στο σύνολο**, χρήζει επιπλέον βοήθειας, μιας και οι συμμετέχοντες δεν έχουν ξεκάθαρα πολλά πράγματα. Αυτό θα πρέπει να μας προβληματίσει, διότι η επιτυχία των έργων δε θα είναι σίγουρη στο τέλος.

Οι παράγοντες των **απαιτήσεων και των προδιαγραφών (Οι απαιτήσεις είναι γνωστές εκ των προτέρων & μεταβάλλονται ελάχιστα κατά τη διάρκεια ανάπτυξης, Ύπαρξη καταγεγραμμένων προδιαγραφών και Προτεραιοποίηση όλων των απαιτήσεων από την**

αρχή), σύμφωνα με αρκετές αρχές του «ευέλικτου» μανιφέστου, παίζουν καθοριστικό ρόλο για την επιτυχία του έργου. Ο παράγοντας του «**Οι απαιτήσεις του προς ανάπτυξη λογισμικού είναι γνωστές εκ των προτέρων σε μεγάλο βαθμό και μεταβάλλονται ελάχιστα κατά τη διάρκεια του έργου**» μας έδωσε ότι το 32% των συμμετεχόντων πιστεύει «Λίγο», ενώ το 12,9% έχει την άποψη ότι δε συνεισφέρει «Καθόλου». Άρα, το σύνολο του 45%, δηλαδή το ½ των συμμετεχόντων σχεδόν δεν έχει ξεκάθαρη εικόνα των απαιτήσεων από την αρχή, ενώ παράλληλα συμφωνούν ότι οι απαιτήσεις μεταβάλλονται ελάχιστα, σε αντίθεση με τις βασικές αρχές και αξίες του μανιφέστου που μιλούν ξεκάθαρα για μεταβλητότητα. Στη συνέχεια, ο παράγοντας της «**Ύπαρξη καταγεγραμμένων αναλυτικών προδιαγραφών για τις απαιτήσεις του λογισμικού**» μας έδωσε τα εξής, το 23% των συμμετεχόντων απάντησε πως γνωρίζει «Λίγο» για τις απαιτήσεις και τις προδιαγραφές του υπό ανάπτυξη λογισμικού, ενώ από 3,2% απάντησε «Καθόλου» και «Δε γνωρίζω». Άρα, το σύνολο του 30%, δηλαδή σχεδόν το 1/3 των συμμετεχόντων σχεδόν ΔΕΝ λαμβάνει γνώση για τις προδιαγραφές των απαιτήσεων του υπό ανάπτυξη λογισμικού, βάζοντας σε κίνδυνο την επιτυχία. Τέλος, για την «**Προτεραιοποίηση όλων των απαιτήσεων από την αρχή του έργου**», οι συμμετέχοντες στο σύνολο του 35,2%, δηλαδή πάνω από το 1/3 του συνόλου απάντησαν πως «Λίγο» και «Καθόλου» δεν έχουν γνώση των προτεραιοτήτων. Συμπερασματικά, **οι παράγοντες των απαιτήσεων και προδιαγραφών στο σύνολο** που παίζουν καθοριστικό ρόλο, σύμφωνα με τις αρχές του «ευέλικτου» μανιφέστου, και οδηγούν στην επιτυχία, στη δική μας έρευνα δεν έχουν τα επιθυμητά αποτελέσματα σύμφωνα με τους συμμετέχοντες, βάζοντας σε κίνδυνο την πορεία των υπό ανάπτυξη έργων.

Τέλος, στοιχείο της «**Συνεχόμενης ανατροφοδότησης από τους πελάτες και τους τελικούς χρήστες**», που ανήκει στην ομάδα παραγόντων της **επανατροφοδότησης και διαδικασιών ελέγχου & εγκυρότητας των οργανωτικών παραγόντων**, χρήζει περαιτέρω συζήτησης και ανάλυσης. Οι συμμετέχοντες εδώ έδωσαν αποτελέσματα της τάξεως του 20%, τα οποία λένε ότι επικρατεί «Λίγη» ανατροφοδότηση μεταξύ συμμετεχόντων και πελατών – τελικών χρηστών. Σύμφωνα με τις αρχές και αξίες του «ευέλικτου» μανιφέστου, η ανατροφοδότηση παίζει πρωταρχικό ρόλο στην επιτυχία ανάπτυξης των έργων. Άρα, το αποτέλεσμα της έρευνας και ο παράγοντας αυτός θα πρέπει να διερευνηθεί περαιτέρω για να έχουμε τα κατάλληλα αποτελέσματα.

Συμπερασματικά, μελετώντας κανείς τα αποτελέσματα της έρευνάς μας θα συνειδητοποιήσει πολλά ενδιαφέροντα πράγματα για το ποια στοιχεία αποτελούν το κλειδί για την επιτυχία κατά τη διάρκεια ανάπτυξης έργων λογισμικού. Θα ξεχωρίσει, όμως, και εκείνα που χρήζουν ιδιαίτερης - περαιτέρω ερμηνείας. Το χαρακτηριστικό της κατηγορίας της εταιρείας, η γνώση και εφαρμογή των «ευέλικτων» μεθοδολογιών, ποια μεθοδολογία ακολουθείται από τους συμμετέχοντες και οι παράγοντες της αναλυτικής τεκμηρίωσης, του σχεδιασμού, της διοίκησης, των απαιτήσεων – προδιαγραφών γενικά ως συνόλου, και μεμονωμένα της ανατροφοδότησης με τους πελάτες και τελικούς χρήστες θα έπρεπε να επανεξεταστούν ξανά. Όπως αποδεικνύεται από την έρευνά μας, οι συμμετέχοντες δεν έχουν καθαρή εικόνα, με αποτέλεσμα την αποτυχία των έργων στα οποία συμμετέχουν.

4.3. Συμπεράσματα & Προτάσεις για μελλοντική έρευνα

4.3.1. Συμπεράσματα

«Οι ευέλικτες μεθοδολογίες δεν είναι πανάκεια, ούτε μόδα, αλλά πρέπει να εξετάζεται το κάθε έργο ξεχωριστά ούτως ώστε να μπορούμε να διακρίνουμε την καταλληλότητα μιας «ευέλικτης» μεθοδολογίας, εφόσον το project το επιτρέπει»

Τα παραπάνω λόγια είναι η απάντηση ενός συμμετέχοντα στην έρευνα, στην τελευταία ερώτηση του ερωτηματολογίου, αν έχουν θετική ή αρνητική γνώμη με τη χρήση των «ευέλικτων» μεθοδολογιών οδηγώντας στην επιτυχία των έργων.

Ο στόχος της παρούσας διπλωματικής εργασίας ήταν να διερευνήσει αν εφαρμόζονται οι «ευέλικτες» μεθοδολογίες ανάπτυξης λογισμικού από τις ελληνικές εταιρείες και ποιοι από τους παράγοντες οδηγούν στην επιτυχία. Η εργασία ξεκίνησε με τον ορισμό του έργου, της διοίκησης έργων, ενώ πέρασε σιγά σιγά στο λογισμικό και την τεχνολογία λογισμικού. Στη συνέχεια, αναπτύχθηκε ο κύκλος ζωής των έργων με τα παραδοσιακά και «ευέλικτα» μοντέλα, ενώ αναφέρθηκαν 14 μελέτες περίπτωσης – εφαρμογής των μεθοδολογιών. Σύμφωνα με όλα τα παραπάνω, διεξήγαμε μια ποιοτική έρευνα, με μερικά μετρήσιμα χαρακτηριστικά που τα συγκρίναμε με αυτά της θεωρίας και των δεκατεσσάρων μελετών περίπτωσης. Τα αποτελέσματα μας έδωσαν τους παράγοντες εκείνους, οι οποίοι χρήζουν περαιτέρω

διερεύνησης, έτσι ώστε μαζί με τους υπόλοιπους παράγοντες και τα χαρακτηριστικά να έχουμε την επιτυχία στην ανάπτυξη των έργων μας.

Η εμπειρία του συγγραφέα στην παρούσα διπλωματική έρευνα οδήγησε στα παρακάτω συμπεράσματα:

- Αναζητώντας κανείς έρευνες και βιβλιογραφία με την εφαρμογή των «ευέλικτων» μεθοδολογιών ανάπτυξης λογισμικού στην Ελλάδα, θα συνειδητοποιήσει ότι δεν υπάρχουν. Πρόκειται για κάτι νέο και μοναδικό, το οποίο χρήζει ιδιαίτερης προσοχής για το πως θα πραγματοποιηθεί και το πλαίσιο στο οποίο θα ερμηνευτούν τα αποτελέσματα.
- Το χρονικό διάστημα, στο οποίο αναπτύχθηκε και διεξήχθη η έρευνα, ήταν περιορισμένο. Αυτό το στοιχείο θα πρέπει να ληφθεί υπόψιν ως περιορισμός της έρευνας.
- Το εύρος των συμμετεχόντων αποτελούνταν από επιτυχημένους επαγγελματίες, οι οποίοι συμμετέχουν σε επιτυχημένα έργα μεγάλης αποδοχής. Συμμετέχοντες από μικρότερης κλίμακας έργα ή από μικρές και περιφερειακές εταιρείες δεν ήταν εφικτό να εντοπιστούν, έτσι ώστε να έχουμε ορθότητα των απαντήσεων. Και αυτό το στοιχείο θα πρέπει να ληφθεί ως ένας ακόμη περιορισμός της έρευνάς μας.
- Αρκετοί από τους παράγοντες που κατατέθηκαν στους συμμετέχοντες αποδείχθηκαν ελλιπείς, με αποτέλεσμα τη μη πιθανή επιτυχή ανάπτυξη των έργων.

Τα παραπάνω στοιχεία είναι σαφές ότι αποτελούν ευρήματα της παρούσας μελέτης, ενώ σε καμία περίπτωση δε μπορούν να θεωρηθούν απόλυτα, καθώς εμπλέκονται ποιοτικοί παράμετροι που αναπτύχθηκαν στα προηγούμενα κεφάλαια. Ωστόσο, η παράθεσή τους παραπάνω αποτελεί σημείο αναφοράς, το οποίο αποτελεί τη βάση για να συναχθούν τα γενικά συμπεράσματα στο παρόν κεφάλαιο.

Η συνεισφορά της παρούσας έρευνας δεν είναι απλά να αναφέρει την εφαρμογή των «ευέλικτων» μεθοδολογιών ανάπτυξης λογισμικού από τους Έλληνες σχεδιαστές - κατασκευαστές, αλλά να επισημάνει τους παράγοντες επιτυχίας, αλλά και εκείνους που χρήζουν ιδιαίτερης προσοχής, μιας και οι επιτυχημένοι εφαρμοστές τους δεν τις αναπτύσσουν

σωστά. Επίσης να αποτελέσει ένα εγχειρίδιο για όσους θέλουν να γνωρίσουν τις εισαγωγικές έννοιες των μεθοδολογιών.

4.3.2. Μελλοντική έρευνα

Η φύση της παρούσας διπλωματικής εργασίας οδήγησε στην αποτύπωση ορισμένων ποσοτικών στοιχείων – χαρακτηριστικών των παραγόντων που οδηγούν στην επιτυχία των έργων ανάπτυξης λογισμικού, εφαρμόζοντας «ευέλικτες» μεθοδολογίες με βάση τα ποιοτικά στοιχεία που αναπτύσσονται από τη θεωρία.

Η παρούσα έρευνα θα μπορούσε να επεκταθεί σε:

- Μια μελέτη με μεγαλύτερη διάρκεια διεξαγωγής.
- Μια μελέτη με μεγαλύτερο εύρος συμμετεχόντων, έτσι ώστε να καλύπτεται σίγουρα όλη η αγορά.
- Μια μελέτη που θα έχει περισσότερους παράγοντες και ποσοτικά μετρήσιμα χαρακτηριστικά, έτσι ώστε να έχουμε μια ακριβή και μετρήσιμη εικόνα.
- Μια μελέτη, στην οποία ο υποψήφιος ερευνητής να καταφεύγει, αναζητώντας περισσότερα στοιχεία για την ελληνική αγορά.

Τέλος, ελπίζω η παρούσα διπλωματική εργασία να έδωσε αρκετά ερεθίσματα, έτσι ώστε οι υποψήφιοι μελλοντικοί εφαρμοστές των «ευέλικτων» μεθοδολογιών, αλλά και οι ερευνητές, να καταφεύγουν σε αυτή για την εκμαίευση στοιχείων για διεύρυνση των γνώσεων και των επαγγελματικών εμπειριών, αλλά και οργάνωση νέων περαιτέρω ερευνών με σκοπό και μόνο την επιτυχία των έργων.

Η σελίδα αυτή είναι σκόπιμα λευκή.

Βιβλιογραφία

- Masood, Z., Hoda, R. & Blincoe, K. (2018) Adapting Agile Practices in University Contexts, New Zealand, Elsevier, Volume 144, pp: 501 – 510, doi: https://doi.org/10.1016/j.jss.2018.07.011
- Nuottila, J., Aaltonen, K. & Kujala, J. (2016) Challenges of adopting agile methods in a public organization, International Journal of Information Systems and Project Management, Vol. 4, No. 3, 65-85, doi: 10.12821/ijispm40304
- Kouzari, E., Gerogiannis, V., Stamelos, I. & Kakarontzas G. (2015), Critical success factors and barriers for lightweight: software process improvement in agile development: A literature review, France, 10th International Conference on Software Engineering and Applications (ICSOFT-EA-2015), pages 151-159, doi: http://dx.doi.org/10.5220/0005555401510159 & http://www.icsoft-ea.org/
- Misra, S. C., Kumar, V. & Kumar U. (2009) Identifying some important success factors in adopting agile software development practices, India, Elsevier, Volume 82, Issue 11, pp: 1869 – 1890, doi: https://doi.org/10.1016/j.jss.2009.05.052
- Agile Alliance (2001), Manifesto for Agile Software Development, www.agilemanifesto.org
- Rubin K. (2012), Essential Scum – A practical guide to the most popular Agile Process, Addison – Wesley
- Aurum, A. & Wohlin C. (2005), Engineering and Managing Software Requirements, Springer
- Cohn, M. (2005), Agile Estimating and Planning, Pearson Education Inc
- Stellman, A. & Greene, I. (2006), Applied Software Project Management, O'Reilly Media
- Burke, R. (1999), Project Management - Planning and Control Techniques, J. Wiley & Sons LTD
- Stellman, A. & Greene, J. (2015), Learning Agile, O'Reilly Media
- Burke, R. & Barron, S. (2014), Project Management Leadership, J. Wiley & Sons LTD
- Cohn, M. (2010), Succeeding with Agile – Software Development using Scrum, Addison – Wesley
- Cohn, M. (2004), User Stories Applied – For Agile Software Development, Addison – Wesley
- Fitsilis, P. (2004), Designing Software – eshop Case Study, Hellenic Open University

- Papatheocharous, E. & Andreou, A. (2013), Evidence of Agile Adoption in Software Organizations: An Empirical Survey, F. McCaffery, R.V. O'Connor, and R. Messnarz (Eds.): EuroSPI 2013, CCIS 364, pp. 237–246, DOI: 10.1007/978-3-642-39179-8_21
- Katsikas, D. (2013), Modern Methods of Computer Project Management, University of Macedonia
- Rigopoulou, M. (2014), Scrum Methodology and its Application in Information Systems Development, University of Patras
- Pichliavas, D. (2015), Techniques Analysis for Development of Innovative Software Products, of the Extraction 's Methodology and the User's Requirements, Ethniko Metsovio Politechnio
- Monochrisou, E. (2010), Computing Project Management Methodologies: Agile methodologies and their use in public computing projects, University of Macedonia
- Sheffield, J., Lemétayer, J. (2012), Factors associated with the software development agility of successful projects, New Zealand, Elsevier, Volume 31, Issue 3, April 2013, Pages 459-472, doi: https://doi.org/10.1016/j.ijproman.2012.09.011
- Ahimbisibwe, A., Cavana, R., Daellenbach, U. (2013), A contingency fit model of critical success factors for software development projects. A comparison of agile and traditional plan-based methodologies, New Zealand, Journal of Enterprise Information Management, Vol. 28 No. 1, 2015, pp. 7-33, doi: 10.1108/JEIM-08-2013-0060
- Pikkarainen, M., Salo, O., Kuusela, R., Abrahamsson, P. (2011), Strengths and barriers behind the successful agile deployment – insights from the three software intensive companies in Finland, Springer Science – Business Media, Vol 17, pages: 675 – 702, DOI: 10.1007/s10664-011-9185-5
- Misra, S. C., Kumar, V., Kumar, U. (2009), Identifying some critical changes required in adopting agile practices in traditional software development projects, Emerald Group Publishing Limited, International Journal of Quality & Reliability Management, Vol. 27 No. 4, 2010, pp. 451 - 474, DOI: 10.1108/02656711011035147
- Gandomani, T. J., Zulzalil, H., Ghani, A. A. A., Sultan, A. B. M. & Sharif, K. Y. (2014), How Human Aspects Impress Agile Software Development Transition and Adoption, International Journal of Software Engineering and Its Applications, Vol.8, No.1 (2014), pp: 129-148, DOI: http://dx.doi.org/10.14257/ijseia.2014.8.1.12
- Dhir, S., Kumar, D. & Singh, V. B. (2019), Success and Failure Factors that Impact on Project Implementation Using Agile Software Development Methodology, Springer Nature Singapore Pte Ltd. 2019 M. N. Hoda et al. (eds.), Software Engineering,

Advances in Intelligent Systems and Computing 731, pages: 647 – 654, DOI: https://doi.org/10.1007/978-981-10-8848-3_62

- Tam, C., Moura, E. J. D.C., Oliveira, T. & Varajão, J. (2020), The factors influencing the success of on-going agile software development projects, Elsevier, International Journal of Project Management 38 (2020), pages: 165 – 176, DOI: https://doi.org/10.1016/j.ijproman.2020.02.001
- Hummel, M. & Epp, A. (2015), Success Factors of Agile Information Systems Development: A Qualitative Study, 48th Hawaii International Conference on System Sciences, DOI 10.1109/HICSS.2015.598
- Aldahmash, A., Gravell, A. M. & Howard, Y. (2017), A Review on the Critical Success Factors of Agile Software Development, Springer International Publishing AG 2017J, Stolfa et al. (Eds.): Euro SPI 2017, CCIS 748, pp. 504–512, 2017, DOI: 10.1007/978-3-319-64218-5_41
- Asnawi, A. L., Gravell, A. M. & Wills, G. B. (2010), An Empirical Study: Understanding Factors and Barriers for Implementing Agile Methods in Malaysia, 5th International Doctoral Symposium on Empirical Software Engineering (IDoESE)
- Επιτροπή Ηθικής Δεοντολογίας της Έρευνας του Πανεπιστημίου Κρήτης, Κανόνες ηθικής και δεοντολογίας, https://www.uoc.gr/research-at-uni/ethics

Η σελίδα αυτή είναι σκόπιμα λευκή.

Παραρτήματα

Ερωτηματολόγιο Έρευνας

Πανεπιστήμιο Θεσσαλίας

Σχολή: **Διοίκησης & Οικονομίας**

Τμήμα: **Διοίκησης & Επιχειρήσεων**

Μεταπτυχιακό Πρόγραμμα Σπουδών: «***Διοίκηση Έργων & Προγραμμάτων***»

Ερωτηματολόγιο στα πλαίσια της μεταπτυχιακής διατριβής με τίτλο:

«***Διερεύνηση του βαθμού χρήσης «ευέλικτων» μεθόδων μάνατζμεντ από Ελληνικές εταιρείες λογισμικού***»

Ονοματεπώνυμο φοιτητή: **Αθανάσιος Γκουρομπίνος / athagkou10@uth.gr**

Επιβλέπων καθηγητής: **Βασίλειος Γερογιάννης**

Το παρόν ερωτηματολόγιο απευθύνεται σε αναλυτές συστημάτων λογισμικού, σε μηχανικούς λογισμικού, σε προγραμματιστές και διαχειριστές έργων, που εργάζονται σε εταιρίες που αναπτύσσουν λογισμικό ή σε εταιρίες πληροφορικής.

Το ερωτηματολόγιο διεξάγεται στα πλαίσια έρευνας που πραγματοποιείται στη διπλωματική εργασία του υποψηφίου μεταπτυχιακού φοιτητή Αθανάσιου Γκουρομπίνου, με θέμα: «Διερεύνηση του βαθμού χρήσης «ευέλικτων» μεθόδων μάνατζμεντ από Ελληνικές εταιρείες λογισμικού». Η παρούσα εργασία πραγματοποιείται στο Μεταπτυχιακό Πρόγραμμα Σπουδών με τίτλο «Διοίκηση Έργων & Προγραμμάτων» της Σχολής Διοίκησης και Οικονομίας του Πανεπιστημίου Θεσσαλίας. Επιβλέπων στην συγκεκριμένη μεταπτυχιακή εργασία διατελεί ο κ. Βασίλειος Γερογιάννης, καθηγητής του Πανεπιστημίου Θεσσαλίας.

Στόχος του συγκεκριμένου ερωτηματολογίου είναι να διερευνήσει τους παράγοντες που συνεισφέρουν θετικά ή επηρεάζουν αρνητικά την επιτυχία εφαρμογής των «ευέλικτων» μεθοδολογιών ανάπτυξης λογισμικού στο περιβάλλον μιας εταιρείας, η οποία αναπτύσσει λογισμικό.

Σας διαβεβαιώνουμε ότι οι απαντήσεις που θα δοθούν στο παρακάτω ερωτηματολόγιο θα είναι απολύτως ανώνυμες και πως η συλλογή και η επεξεργασία των δεδομένων του ερωτηματολογίου, καθώς και η δημοσιοποίηση αυτών, στα πλαίσια της εκπόνησης της συγκεκριμένης διπλωματικής έρευνας, υπόκεινται στους νόμους του κράτους περί προστασίας δεδομένων. Η επεξεργασία και η χρήση τους θα γίνει, αποκλειστικά και μόνο, για τους σκοπούς της συγκεκριμένης εργασίας.

Το ερωτηματολόγιο χωρίζεται σε τρεις ενότητες. Η πρώτη ενότητα αφορά γενικές πληροφορίες σχετικά με τον ρόλο σας ως συμμετέχοντα στην έρευνα. Η δεύτερη ενότητα παρουσιάζει ερωτήσεις αναφορικά με το βαθμό γνώσης και εξοικείωσης με τις «ευέλικτες» μεθοδολογίες. Η τρίτη ενότητα, που είναι και η κύρια ενότητα του ερωτηματολογίου, παρουσιάζει ερωτήσεις που εξετάζουν τη συμβολή διαφόρων παραγόντων των «ευέλικτων» μεθοδολογιών αναφορικά με την επίπτωσή τους στην επιτυχία των έργων ανάπτυξης λογισμικού.

Α' Ενότητα: Γενικές πληροφορίες

Φύλο:

- Άνδρας
- Γυναίκα

Ηλικία:

- 20 - 30
- 30 - 40
- 40 - 50
- Άνω τον 50

Ρόλος στην εταιρεία:

- Διευθυντής Εταιρίας
- Υπεύθυνος Έργου
- Αρχηγός Ομάδας
- Προγραμματιστής
- Μηχανικός Λογισμικού
- Αναλυτής / Σχεδιαστής
- Υπεύθυνος για διαδικασίες δοκιμών ελέγχου
- Υπεύθυνος για τη διαδικασία συντήρησης
- Άλλο: …

Αντικείμενο των έργων που υλοποιεί η εταιρεία (περιγράψτε σύντομα):

…..

Κατηγορία Εταιρείας (δώστε τον τύπο που ταιριάζει περισσότερο στην εταιρεία στην οποία εργάζεστε):

- Ατομική
- Μικρή τοπική επιχείρηση
- Περιφερειακής εμβέλειας
- Πανελλαδικής Εμβέλειας
- Ευρωπαϊκής ή Παγκόσμιας Εμβέλειας (με έδρα ή εγκαταστάσεις στην Ελλάδα)
- Ευρωπαϊκής ή Παγκόσμιας Εμβέλειας (με έδρα και εγκαταστάσεις στο εξωτερικό)
- Δημόσιος Οργανισμός
- Άλλο: …

Μέγεθος εταιρείας:

- Λιγότεροι από 10
- 10 - 20
- 21 - 40
- 41 - 100
- 100 και άνω

Τυπικό μέγεθος ομάδας στα έργα που αναλαμβάνει η εταιρία:

- Λιγότεροι από 5
- 5 - 10
- 10 - 20
- 20 - 40
- Άνω των 40

Β’ Ενότητα: Ο βαθμός γνώσης και εξοικείωσης με τις «ευέλικτες» μεθοδολογίες

Με τον όρο «ευέλικτες» μεθοδολογίες (Agile Methodologies) ορίζουμε μια οικογένεια μεθοδολογιών ανάπτυξης λογισμικού, οι οποίες βασίζονται στις παρακάτω βασικές αρχές:

- Άτομα και αλληλεπιδράσεις αντί διαδικασιών και εργαλείων
- Δυναμικός κώδικας αντί γραπτής τεκμηρίωσης
- Συνεργασία με τον πελάτη αντί αυστηρών συμβολαίων
- Ανταπόκριση σε αλλαγές αντί ακολουθούμενου σχεδίου

Οι «ευέλικτες» μεθοδολογίες δίνουν περισσότερη έμφαση σε αυτά που βρίσκονται στα αριστερά από αυτά που βρίσκονται στα δεξιά.
Σε αυτή την κατηγορία μεθοδολογιών ανήκουν μεθοδολογίες, όπως αυτές των: Scrum, XP, Crystal Family, Feature Driven Development (FDD), Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD), Lean Development και Διαδικασία της Ενοποιημένης Προσέγγισης (Unified Process).

Εδώ και πόσο καιρό γνωρίζετε σχετικά με τις «ευέλικτες» μεθοδολογίες;

- Εδώ και 1 χρόνο
- Από 1 - 3 χρόνια
- Από 3 - 5 χρόνια
- Περισσότερο από 5 χρόνια

Για πόσο καιρό έχετε εμπλακεί σε έργα που εφαρμόζουν "ευέλικτες" μεθοδολογίες;

- Για 1 χρόνο
- Από 1 - 3 χρόνια
- Από 3 - 5 χρόνια
- Περισσότερο από 5 χρόνια

Ποια Agile μεθοδολογία συνήθως χρησιμοποιείτε;

- Scrum
- eXtreme Programming (XP)
- Crystal Family
- Ανάπτυξη Βάσει Χαρακτηριστικών (Feature Driven Development - FDD)
- Μέθοδος Ανάπτυξης Δυναμικών Συστημάτων (Dynamic Systems Development Method - DSDM)
- Προσαρμοστική Ανάπτυξη Λογισμικού (Adaptive Software Development - ASD)
- Lean Software Development
- Άλλο: …

Γ΄ Ενότητα: Οι παράγοντες των "ευέλικτων" μεθοδολογιών που συνεισφέρουν στην επιτυχία των έργων λογισμικού

Ζητάμε να αξιολογήσετε το βαθμό με τον οποίο οι ακόλουθοι παράγοντες συνεισφέρουν θετικά στην επιτυχία ενός έργου ανάπτυξης λογισμικού.

Σε ποιο βαθμό καθένας από τους παρακάτω παράγοντες είναι σημαντικός για την επιτυχία ενός έργου ανάπτυξης λογισμικού;

1. Συνεχόμενη εκπαίδευση του προσωπικού:

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

2. Αξιοποίηση συνεργατών με μεγάλη εμπειρία και εξειδίκευση:

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

3. Επικοινωνία μεταξύ των μελών της ομάδας έργου:

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

4. Επικοινωνία με τους πελάτες και τους τελικούς χρήστες:

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

5. Βαθμός συνεργασίας μεταξύ των συμμετεχόντων:

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

6. Απλότητα στη σχεδίαση:

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

7. Επανατροφοδότηση στην περίπτωση λαθών και αστοχιών:

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

8. Αποτελεσματική διοίκηση του έργου:

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

9. Διαθεσιμότητα ανθρώπινων πόρων:

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

10. Διαθεσιμότητα οικονομικών πόρων:

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

11. Δέσμευση, προσήλωση και υπευθυνότητα όλων των συμμετεχόντων στην εκπλήρωση των στόχων του έργου:

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

12. Τεχνικές γνώσεις και ικανότητες των μελών του έργου.

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

13. Ικανότητες και δεξιότητες επικοινωνίας:

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

14. Ικανότητα προσαρμογής στις αλλαγές, τόσο σε αυτές που αφορούν το έργο όσο και σε αυτές που αφορούν τις απαιτήσεις σε σχέση με το λογισμικό:

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

15. Αναλυτική τεκμηρίωση σε κάθε φάση της ανάπτυξης του έργου:

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

16. Ύπαρξη τυπικών διαδικασιών διαχείρισης του έργου και ιεραρχικών δομών διοίκησης:

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

17. Οι απαιτήσεις του προς ανάπτυξη λογισμικού είναι γνωστές εκ των προτέρων σε μεγάλο βαθμό και μεταβάλλονται ελάχιστα κατά τη διάρκεια του έργου:

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

18. Ύπαρξη καταγεγραμμένων αναλυτικών προδιαγραφών για τις απαιτήσεις του λογισμικού:

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

19. Ύπαρξη αναλυτικής σχεδίασης για το λογισμικό:

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

20. Λεπτομερής προγραμματισμός (σχεδίαση - χρονοπρογραμματισμός):

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

21. Προτεραιοποίηση όλων των απαιτήσεων από την αρχή του έργου:

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

22. Συνεχόμενη ανατροφοδότηση από τους πελάτες και τους τελικούς χρήστες:

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

23. Αναλυτικές διαδικασίες ελέγχου (εύρεση και διόρθωση λαθών):

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

24. Ύπαρξη ελέγχων εγκυρότητας:

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

25. Καθορισμός ρόλων και ευθυνών των μελών από την αρχή:

- Πολύ
- Αρκετά
- Λίγο
- Καθόλου
- Δε γνωρίζω
- Συνεισφέρει Αρνητικά

Ποια είναι η γνώμη σας σχετικά με τα αποτελέσματα της χρήσης των "ευέλικτων" μεθοδολογιών στην επιτυχία των έργων και στην υλοποίηση του στόχου της εταιρείας σας;

- Θετική
- Αρνητική
- Άλλο: …

Η σελίδα αυτή είναι σκόπιμα λευκή.

Αγγλική Έκδοση της Εργασίας

Ως παράρτημα της παρούσας εργασίας δίνουμε μια έκδοση της εργασίας στα αγγλικά. Αυτό θεωρήθηκε σκόπιμο προκειμένου μέσα από αυτό το κείμενο να προκύψουν ορισμένα ερευνητικά άρθρα.

University of Thessaly

Economy & Business School

Postgraduate Programme in

"Project & Programme Management"

Master's Thesis

Success factors in Agile Software Projects from Greek IT companies & Professionals

Author: **Athanasios Gkourompinos**

Supervisor: **Vasileios Gerogiannis**

Larissa 2021

This page is intentionally white.

University of Thessaly

Economy & Business School

Postgraduate Programme in

“Project & Programme Management”

Master’s Thesis

Success factors in Agile Software Projects from Greek IT companies & Professionals

Author: **Athanasios Gkourompinos**

Supervisor: **Vasileios Gerogiannis**

Approved by the three-member examination committee on May 10th, 2021

(Signature)	(Signature)	(Signature)
…….	…….	…….
Vasileios Gerogiannis	Leonidas Anthopoulos	Panagiotis Fitsilis
Professor – University of Thessaly	Professor – University of Thessaly	Professor – University of Thessaly

Larissa 2021

This page is intentionally white.

ATHANASIOS GKOUROMPINOS

Civil Infrastructure Engineer, Alexandrio University of Technological Education

This page is intentionally white.

Statement

This dissertation is submitted by Mr. Athanasios Gkourompinos as a partial fulfillment of the requirements of the Postgraduate Program in "Project & Program Management" of the University of Thessaly.

It is responsibly stated that this dissertation has been written by Mr. Athanasios Gkourompinos and has not been submitted or evaluated in the context of another postgraduate or undergraduate degree, in Greece or abroad.

Name:

Athanasios Gkourompinos

Signature:

This page is intentionally white.

To my brother Elias, who left us early.

This page is intentionally white.

Acknowledgments

To my nephews, Thomas and Dimitris, who are the foundations of my life.

To my supervisor, Professor Mr. Vassilios Gerogiannis, for his excellent cooperation, and all the horizons that were broadened by his teaching.

To my family and friends, who are always by my side.

To Mrs. Athanasia Ntalavika, for her editing.

This page is intentionally white.

Abstract

Since 2001, when the Agile Manifesto was established, there were many supporters who adopted its philosophy immediately, improving their daily lives by designing - constructing software. The fields of application are numerous and include Universities, Public Bodies, etc., however the main users are "Agile" professionals. All of them, through their study and work, contributed towards determining success factors and constrains for their respective projects.

Within the present dissertation it is attempted to present the basic concepts such as ‘project’, ‘project management’, ‘Software & Software Technology’, ‘Project Lifecycle and Models (Traditional & Agile)’, and especially ‘IT project management’, in a manner that aims to familiarize and educate readers without any pre-existing knowledge of said concepts.

Specifically, a qualitative study will be attempted, in which, via means of analysis, the level of application of "agile" methodologies by Greek software companies will be recorded and reported on. We aim to identify those factors that lead to the successful completion of projects using international literature and specifically 14 case studies of application of agile methodologies. Finally, this may provide the impetus for further conduct of case studies within the Greek market.

Key Words: Agile Management, success factors, constraints, Greek Software Companies, application

This page is intentionally white.

Περίληψη

Από το 2001 και μετά, όπου και ορίστηκε το Agile Μανιφέστο, υπήρξαν πολλοί οι υποστηρικτές που υιοθέτησαν τη φιλοσοφία του από την πρώτη στιγμή, βελτιώνοντας την καθημερινότητά τους σχεδιάζοντας - κατασκευάζοντας λογισμικό. Τα πεδία εφαρμογής και τα παραδείγματα πολλά, όπως Πανεπιστήμια, Δημόσιοι Φορείς, κλπ., αλλά κυρίως οι επαγγελματίες της Agile. Όλοι αυτοί με την μελέτη και την εργασία τους συνέβαλαν ώστε να καθορίσουν τους παράγοντες επιτυχίας και τους περιορισμούς των έργων.

Στην παρούσα εργασία θεωρούμε σκόπιμο να παρουσιάσουμε τις βασικές έννοιες προκειμένου κάποιος που δεν έχει γνώση πάνω στις βασικές έννοιες του έργου, της διοίκησης έργων, του Λογισμικού & της Τεχνολογίας Λογισμικού, του Κύκλου Ζωής των Έργων και των Μοντέλων του (Παραδοσιακά & Ευέλικτα), και ειδικότερα στην διαχείριση έργων πληροφορικής, ώστε να είναι σε θέση να κατανοήσει όλες αυτές τις έννοιες που θα αποτελέσουν το υπόβαθρο της εργασίας.

Συγκεκριμένα, θα επιχειρηθεί να γίνει μια ποιοτική έρευνα, στην οποία μέσω ανάλυσης θα καταγραφεί αν γίνεται εφαρμογή των «ευέλικτων» μεθοδολογιών από τις ελληνικές εταιρείες κατασκευής λογισμικού. Παράλληλα, θα εντοπίσουμε εκείνους τους παράγοντες που οδηγούν στην επιτυχία ολοκλήρωσης των έργων μέσω της διεθνούς βιβλιογραφίας και συγκεκριμένα μέσω 14 μελετών περίπτωσης εφαρμογής των «ευέλικτων» μεθοδολογιών, ενώ στο τέλος θα δοθεί το έναυσμα για περαιτέρω διενέργεια νέων μελετών περίπτωσης στην ελληνική αγορά.

Λέξεις κλειδιά: Agile Management, παράγοντες επιτυχίας, περιορισμοί, ελληνικές εταιρείες λογισμικού, εφαρμογή

This page is intentionally white.

Contents

This page is intentionally white.

List of Graphs

This page is intentionally white.

Table List

This page is intentionally white.

Chapter 1: Introduction

Over the centuries, since mankind felt the need to protect ourselves and those around us, we began building rough constructions which served this goal. Such early constructions were makeshift houses, but with the emergence of early human societies, mankind attempted to establish communication channels by constructing bigger works.

To successfully accomplish such works, there must have certainly existed influential bright minds, with brilliant ideas and methods. While many of these methods and ideas may not have survived till current times, these great projects are a testament to the pivotal role of management methods as well as the ability of the leaders constructing them.

The first visible examples of project management are recorded in France during the Middle Ages. There, the master builders led the work. Officially established project management appears in the early 1900s with Gantt, and a little later in the major modern projects reaching the present day. The science of Project Management is a science that is constantly evolving even today, since it first officially introduced as teaching material in Universities in 2000!

And while the evolution of man began to peak, simultaneously technologies began to develop that would make for a much-improved experience of our daily life. Turing in the 1930s begins to introduce computer science into our lives, paving the way for a new form of everyday life. The addition of information technology, software and a range of novel technologies into our lives and has demonstrably enhanced them. IT's extremely rapid development was assisted by the rapid advancement of electronic science and population demand.

Finally, the advent of the internet has complemented this development, leading to even greater technological advances. In the current era of the 4th Industrial Revolution we observe that we are facing economic and social developments, which only aim to improve the life expectancy of projects and people.

1.1. The Problem

In the early years of the development of the science of cybernetics and especially of information technology and software, there was no structured process focused on developing the final products. Especially in the 70's, with the rapid development of cybernetics, and in particular, software development projects, projects began to become increasingly complex, involving a large number of specialized staff, exemplifying the need to develop a project management methodology.

One of the first structured methodologies, which describes in concrete steps the whole development of a software development project, is that of the Serial Waterfall Model as given

by Royce in 1970. This methodology was already applied and used by him and his team. Several project development methodologies followed several years later. Some of them were: Boehm's Spiral Model in 1988, Connell's Rapid-Prototyping Model in 1989, Wong's Incremental (Staged) Delivery Model in 1984, Gilb's Evolutionary Delivery Model in 1988, the method Martin's Rapid Application Development (RAD) in 1991, Kruchten's 1996 Rational Unified Process (RUP) methodology, and others.

From all the above methodologies, it is evident that they find excellent application in projects that follow a series of developmental steps, in contrast to IT and software projects, which in general presents a series of problems and difficulties that mainly concern exceedances in terms of schedules, budget, quality and requirements of the customers.

Working and seeing all of the above, a team of 17 scientists gathered in February 2001 at Lodge, a Snowbird ski resort in the Wasatch Mountains of Utah, USA, for food and skiing, seeking to identify common ground for software development. From this meeting was born the "Agile Manifesto" (Agile Management). According to its values and principles, it now promises greater adaptability and response to change, more productive practices and less bureaucracy. The "Agile" methodologies propose a "revolutionary" approach to the usual practice, where the human factor (both the user-client and the members of the development team) is placed at a priority level.

The use of "Agile" methodologies has gradually become widely used worldwide. That is, it has found application in many areas of everyday life such as: universities, public services, companies, etc. and mainly in IT and software projects. Studies have demonstrated the success factors of these methodologies - practices worldwide.

Taking into account all of the above, and considering the fact that in the Greek reality, within the last 20 years many important projects have been implemented as well as large applications have been made to facilitate everyday life, this master's thesis aims to investigate the extent to which the application of "Agile" methodologies has been adopted by software companies and the Greek state.

1.2. Dissertation's Subject

The main goal of this dissertation is to initially identify and investigate whether Greek companies and company executives apply - work on "agile" methodologies. The current study further investigates which factors are considered successful and which are obstacles towards the completion of software projects, applying "agility". Firstly, we will set the hypothetical success factors of the "agile" practices, as they are mentioned in the international literature, and we will adapt them to the Greek reality. Then, based on all the above, all the scientific

terminology will be recorded, while, at the same time, a questionnaire will be created and distributed to the Greek software companies. After the above is completed, all the material distributed will be collected and the processing and analysis of the data will follow. According to the accumulated results, we will deduce if the "agile" methodologies are adopted by Greek software companies and which of the factors assist in the successful completion of the project.

1.2.1. Contribution

Therefore, the contribution of the present dissertation is analyzed in the following stages:

1.First, we will define the basic concepts in order for someone who does not have knowledge on the basic concepts of ‘project’, ‘project management’, ‘Software & Software Technology’, ‘Project Lifecycle and Models (Traditional)’, and especially in management IT projects, to become familiarized with all these concepts that will be the background of the work.

2. Next, we will focus on all the rules, types and models of "agile" methodologies, so that this new field of Management science is clear.

3. We will determine the success factors of a software, but also the limitations in the exercise "flexibility", according to the international literature, while at the same time we will have adapted them to the Greek data. At the same time, we will create a table with the success factors and the limitations.

4. A questionnaire will be available, which will function as a tool - a framework for determining the success factors, but also for identifying the limitations, applying "agile" methods in the Greek market.

5. An analysis of the results will be carried out, based on the quality characteristics specified above, to check whether the "agile" methodologies are applied in the Greek market and which of the factors lead to the successful completion of the projects.

6. The current situation in the Greek market will be recorded, so that this diploma thesis is a tool for immediate application and acquisition of knowledge of "agile" methods by each reader, candidate for the application of methodologies.

1.2.2. Text Organization

In total, this dissertation consists of four distinct chapters, including the first introductory chapter, which presents a general introduction, the problem, the subject and finally the contribution.

The second chapter presents a bibliographic overview on which the theoretical background of the work is based. That is, the conceptual framework and theories will be defined, the research questions and what emerges from the literature will be posed, while in the end the main conclusions will be drawn.

The third chapter will analyze the methodology of the work. Description of the research methods answering the research questions, with detailed descriptions of the methods separately. Also, a complete analysis of the research will be carried out alongside the choice of procedure which was followed.

In the fourth chapter, following the collection of research data, data analysis will be carried out. This will be done using appropriate techniques in conjunction with the research questions. The analysis will be thorough, based on the description through inductive analysis. The discussion of the results will follow, commenting critically and interpreting them according to the theoretical framework developed in the previous chapters and the findings of another research. In closing, this dissertation concludes with suggestions for the use of research in the broader scientific field. That is, to what extent can the present research be developed in the future and be used in general as a tool at an academic level acting as a tool for the application of methods by its prospective users.

This page is intentionally white.

Chapter 2: Bibliographic Review

In the following chapter we consider it appropriate to present the basic concepts, so that readers without prior knowledge of said concepts and, in particular, the management of computer projects, become familiar to them, since these will comprise the background of the work. Specifically, this chapter will attempt to explain the terms of the Project, Project Management, Software & Software Technology, Software Requirements, Project Lifecycle & Models (Traditional & Agile).

2.1. Project

As mentioned earlier, it was initially the need for protection of the individual and those close to them that provided the impetus for rough constructions. This was followed at a later stage, when early societies started to appear, by a need for communication and connection which resulted in the construction of larger projects. What is a project and how is it created?

Every organization performs a basic function: **it transforms resources into actions.** The same process is done by all organizations. From the human body, to, for example, companies, public bodies, non-profit - governmental organizations, etc.

It may be assumed from our current viewpoint that in order for early construction achievements to materialize, since the birth of human life, there would have certainly been special people who unknowingly influenced the creation of and helped leapfrog Project Management, however little information in regards with this process is available due to lack of early written records. The great projects of the past such as the tower of Babel, the pyramids of Egypt, the Great Wall, etc. projects before the classical historical times taught us that people were attempting to achieve something. The first examples of project management records can be found in France in the Middle Ages with the Master Builders. The first systematic attempt to achieve a better result through the production process was made in the 19th century in a cast iron factory in the USA. Responsible for this effort was the Chief Engineer of the factory, F. W. Taylor (1856 - 1915), who today is considered the father of Management.

From the time of Taylor onwards, project management began to gain widespread acceptance, especially in the field of industry, of technical projects, while special emphasis was placed on military development projects such as the Polaris program, the NASA space program, Apollo, the development of smart bombs, etc. For the last fifty years, project management has been treated as a separate field of scientific and professional study.

So, what is a project?

"**A temporary effort to create a unique product or service**," according to PMBOK, Project Management Institute.

"**An effort in which human, material and financial resources are organized in a new way in order to be used in the execution of tasks with specific specifications, in time and financial constraints so as to achieve a beneficial result determined by quantitative and qualitative criteria**" according to Turner JR, in 1990

"**A project is a temporary administrative structure in which resources are allocated to undertake a unique, new and transient effort to manage inherent uncertainty and the need for coordination and integration to bring about beneficial change**," said Turner J.R. in 2002.

Comparing the above we can say that: **project** is a temporary effort that aims to create a unique product or service. Temporary because it means that every project has a predetermined end, while by uniqueness we mean that the product or service differs in a distinct way from all similar products or services.

Alternatively, we could analyze it as follows: a project is a work, in which all resources (human - material - financial) are organized in an original way, with the aim of undertaking a specific object of work that has its own specifications and is subject to constraints (cost and time), so as to produce a beneficial change which will be determined by quantitative and qualitative criteria.

In detail, a project is also considered a series of interdependent activities with specific characteristics, such as:

- Specific start and end dates
- Well defined goals
- Production of a specific result
- Does not repeat the same series of activities
- Consumption of money, time, human and material resources

One of the successful project models is that of the 4-D model which consists of the following 4 stages:

- D1 - Define the project
- D2 - Design the project process
- D3 - Deliver the project
- D4 - Develop the process

While the 4 basic procedures of a project are the following:

- **Initiating:** concepts, set up

- **Planning:** tasks, estimates, dependencies, constraints, assignments
- **Executing:** optimizing, reporting, updating
- **Closing:** evaluating

The process of developing a project occurs when we have the satisfaction of the need using mechanisms such as: human resources, knowledge, experience, capital, tools - techniques, technology, taking into account all the constraints such as: financial, legal, ethical, logical, chronological, etc.

How is a project **different** from a process - a set of functions?

The **similarities** in both projects and procedures are as follows:

- consume resources
- are limited by budget, time, resources, etc.
- are programmed, executed and controlled

While the **differences** are separate for each case. In projects, the differences have to do with the following:

- They are unique
- They are temporary

The **procedures** have to do with the following:

- They are repetitive
- They are in development

Therefore, the success of a project is judged by meeting the requirements - needs of the customer, according to the time available and the agreed budget, in an acceptable by the customer manner. The failure of a project on the other hand is judged by the unmet requirements - the needs of the client, which obviously resulted by incorrect recording, so as not to produce a correct and realistic design, consuming extra resources by exhausting them for other purposes.

So, according to all the above, the goal of a project is to satisfy the goal triangle that consists of the three constraints: quality, cost and time (Figure 1). In detail we have the following (Figure 2):

- Increased range = increased time + increased cost
- Restricted time = increased cost + reduced range
- Restricted budget = increased time + reduced range

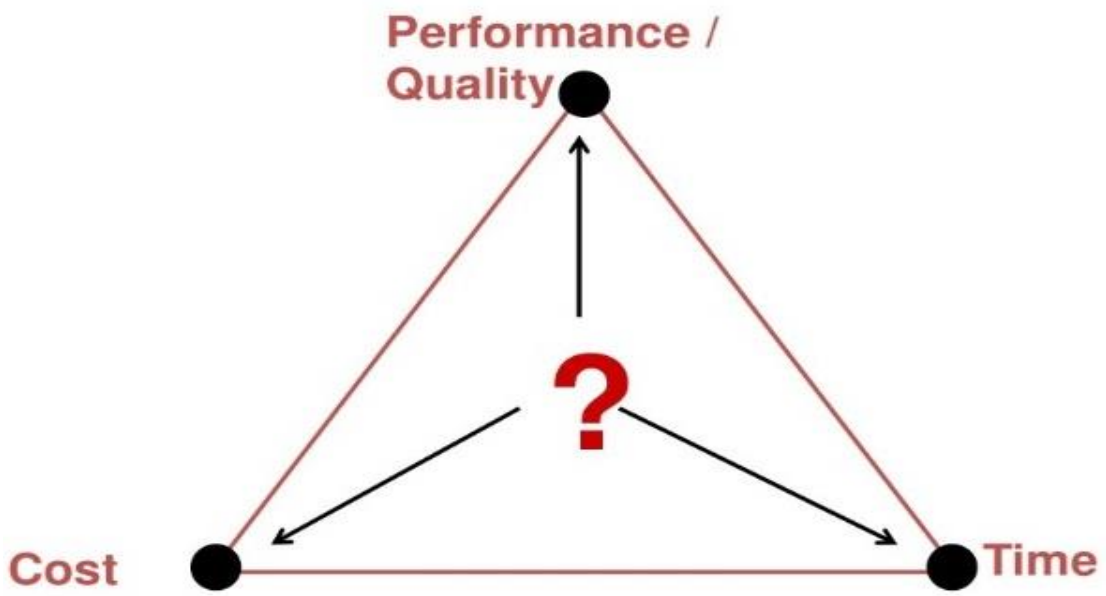


Figure 1 - The triangle of targets – Source: *slideplayer.gr*

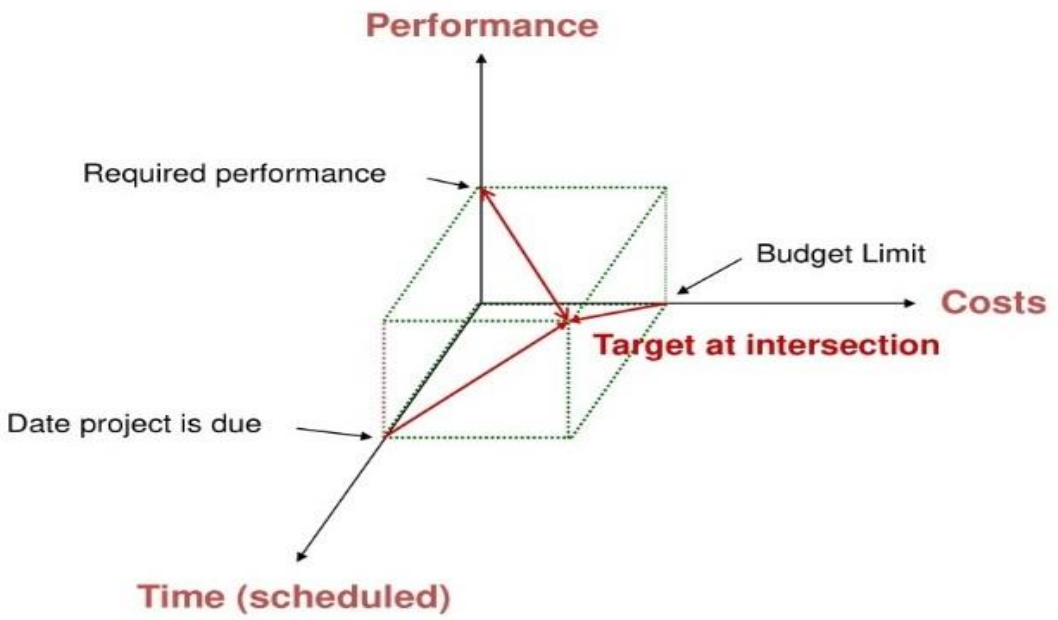


Figure 2 - The objectives of the project – Source: *slideplayer.gr*

2.2. Project Management

The science that deals with the organization of projects is the science of Project Management. "Project Management" aims to answer the questions concerning: why, what, how, who, how much does a project cost and when, from the moment of conception until the moment of implementation, destruction, abandonment, etc. Objective goals are the balance between time (i.e., the life of the project), money (i.e., the lowest cost) and quality (i.e., optimizing the quality).

Alternatively, we could say that Project Management is the process in which we apply knowledge, skills, tools and techniques when performing project activities with the sole aim of meeting the requirements and expectations of all participants.

We could also say that it is a process of integration of everything that needs to be done as the project goes through its life cycle meeting all the objectives.

2.3. Software – Software Technology

Over the last 20 years, the rapid development of information technology and software, provided products and services that simplified processes in our daily lives. The adoption of these processes and products arose first mainly within the most developed countries on the planet and found immediate application in all areas of life. The Greek reality is very different. In recent years IT may have begun to enter the lives of Greeks timidly, but Greece was severely lagging behind in simplified processes and products, due in part to the half-learnedness of the majority of the population and especially the elderly, and due to a larger extent to the huge amounts of bureaucracy.

The new decade that began almost a year ago brought us face to face with a great danger, that of the Coronavirus pandemic, while at the same time introducing the use of technology in all areas of everyday life. The Greeks were confronted almost overnight with the widespread use of computer command programs and other smart devices and services, while the state had to rapidly undergo a monumental digital transformation.

As a result, everyone is now confronted with software literacy. What is software? **Software** is comprised of all the programs or commands of a computer and all other electronic devices. **Software** is all the data structures with permanent or temporary storage that allow programs to manage information that always requires documentation.

Software could be said to be a complex technical task. And this is because the technical projects have a tangible (material) substance and are described - they are perceived. Software has an imaginary substance. It is perceived only with its use, while it is not easily described, standardized and unique. Its development is accomplished by defining all the requirements from the beginning. This is why the "software crisis", which is still relevant today, was mentioned by Bauer (1968) in the early years of software development. Also, in 1968, at a NATO conference, the following was emphasized:

- With the software crisis we have the characterization of the problem.
- With "Software Engineering" we have an idea for a solution.

Therefore, in order for a piece of software to be fit for use, it must consist of the following components: software components, elements, artifact, that include the following features: Executable and source code, specifications - reports, general texts, drawings, diagrams, etc.

"**Software engineering**", as art talk, is the science that describes how a technical construction, including software, will be performed.

Software technology is defined as "***the systematic approach to the analysis, design, evaluation, implementation, control, maintenance, and redesign of software, i.e., the application of engineering to software***" (Laplante, 2007).

Software Technology describes - defines good quality Software according to the following:

- The actions that are implemented
- The products issued
- The description of the specifications - standards of the final products and actions
- The methods of controls, verification and quality assurance.

That is why the need for a "Good Software" program is self-evident and imperative since the role of computers in the economy, production, information, entertainment, education, etc., is fulfilled only with Software that performs its tasks properly, is produced with low cost, with minimal production time and always in excellent quality.

The **object** of Software Technology is the scientific foundation of the life cycle, the production process, the way of description - maintenance and the framework of use of a Software program. The above procedures are very demanding since the Software is difficult by nature and as a final product. Another object of Software Technology is to look for systematic ways to show what components we need to build so that a software program can do its job properly.

Finally, what is "quality - good" Software and how do I implement it? It is certainly a difficult question as cost, quality and development time are mutually exclusive.

2.4. Software System Requirements

For proper software development, one of the processes that should always be followed first is the specification, regardless of the life cycle model that follows. At the same time one of the most important tasks during the development of a software program is to determine the specifications of the requirements. This is because it is a difficult and creative process, requiring special communication knowledge, structured - critical thinking and systematic approach. The methodological framework for carrying out this work is defined by **Software Technology**.

*"**A requirement of the system is the description of a task that should be performed by one of the components of the system (people, machines, software) or a feature that a system should have**"* (V. Veskoukis, 2015).

The process of identifying the services that the customer requires from a system, and the constraints under which the system operates and develops, make the **technology demanding**. The requirements themselves are the descriptions of the services - the constraints of the system that are produced in the process of the requirements technology.

But what is a requirement? **Requirement** could be anything, from something very abstract like a service - a constraint to a detail, a mathematical definition of our system. So, the requirements can have a dual function. They can be a basis for negotiating a contract, so they should leave room for different interpretations. They can be the basis for the contract itself, so they should be defined in detail. Both of these forms can play the role of requirements.

"A requirement of the software is a function that it must perform or a condition that must be satisfied when its construction is completed" (V. Veskoukis, 2015).

The types of requirements are divided into **user requirements, system requirements and software design specification. User requirements** include statements in natural language and diagrams of the services provided by the system and its operational constraints (written for customers). They are addressed to the customer and supplier management staff, end users, customer engineers and system architects. **System requirements** include a structured document detailing system functions, services, and operating constraints. They encompass precisely defined characteristics as regards the implementation as part of the contract between the customer and the contractor. They are aimed at end users, customer engineers, system architects and vendor developers. The **software design specification** consists of a structured document that describes in detail the operating requirements, design constraints, idioms, performance requirements, and other system operating requirements. This is aimed at the customer's engineers, system architects and vendor developers.

Specifications are driven primarily by business objectives (high level), operating environment and organizational environment. The following stakeholders participate in the development of the specifications: users, customers, regulators and development engineers. Data may even come from other sources such as market studies.

We can divide the requirements into two major categories. **Functional requirements and non-functional requirements. Functional requirements** should be Complete and Consistent, describing the tasks (functions) that the software should perform, the interaction between the system and its environment, and finally describing how the system behaves when specific stimuli is received. The **non-functional requirements** will describe features that the software

must have, which do not concern the execution of any function by it but also the specifications that limit the options for finding the solution to the problem. Examples of non-functional requirements are user requirements, reliability, performance, support, design, implementation, communication with other systems, databases and physical requirements.

The life cycle of receivables includes:

- **Elicitation techniques:** through interviews, scripts, archetype construction, structured meetings, observation and existing internal and external documentation.
- **Analysis and negotiation:** which is done by describing the conceptual model that represents the world, and the requirements model that represents the requirements of the software. UML diagrams such as Class Diagrams, Activity Diagrams and Statechart Diagrams are used to describe the ideal model. UML diagrams such as Class Diagrams, Statechart Diagrams, Use Case Diagrams, Sequence Diagrams, and Collaboration Diagrams are used to describe the requirements model.
- **Specification:** Describes the capabilities of the system. Must be Complete and Consistent.
- **Validation:** Refers to error checking, consistency, completeness, realism and verifiability.

Finally, **requirements management** should be processed via specification management which includes specification, change management process, traceability policies (sources of claims, link between claims and link to claims with software design) and support tools. Also, the description of the requirements is often compiled in natural language. The following common mistakes should be avoided when describing requirements: Long-term reasoning with parentheses, use of ambiguous terms, presentation of multiple requirements as a requirement, inconsistent usage of multiple different terms to denote the same meaning and mixing between functional - non-functional requirements. Therefore, the role of requirements in the construction of a software program is a complex task and is an important factor that should be taken into account from the beginning.

2.5. Project Life Cycle

With the term **Project Life Cycle**, we define the sequence of actions required from the conception of an idea of a project to its implementation. First the idea should be identified, then it should be developed - expressed through a work plan which can be implemented in practice and then evaluated. The ideas are identified in the context of a strategy agreed upon by the developer and the contractor. The important milestones between the life cycle phases represent the points of decision making of great importance, while the phases in many cases may overlap.

“Every project and especially every IT project has different characteristics and therefore a different Life Cycle” (Clarke).

The life cycle of a project provides the structure that ensures the participation of those involved in the project, as well as the availability of the necessary information that may be useful during the life of the project.

The concept of Life Cycle Management (PCM) of a project was introduced by the European Commission in the early 1990s, with the aim of improving the quality of project design and management and the effectiveness of EU-funded projects. This methodology was developed in the late 1980s after research pertaining to the conclusions of the evaluation showed that projects were poorly executed for many different reasons. Setting the final goal, a project should have measurable SMART features (Specific, Measurable, Achievable, Realistic, Time - Limited).

The **key elements of a conventional life cycle**, according to Clarke, are:

- **Phase,** is the set of interrelated activities
- **Task** is a specific activity with a defined purpose.
- **Milestone or Deliverable** is a certain result of a Phase or Work.

In detail, a conventional Life Cycle contains the following phases:

- **Project Planning:** Initially, the purpose of the project and those involved in it are determined. Cost and resource analysis is performed. Deliverable of this phase is the agreed Terms of Reference and the Project Plan.
- **Requirement Analysis:** In this phase, the "what" the software will do is determined, by all those involved in the project and the end users. The participation of all the above in the design of the product is necessary. Deliverable at this stage is the agreed System Requirements Statement (SRS).
- **System Design:** This section develops "how" the software will work according to SRS. Deliverable of this phase is the System Design Specification.
- **Implementation (Development):** In the implementation we include the evaluation and collection of the existing software. The writing of new software, the definition of the detailed specifications - activities, the integration of all the elements in the final product, as well as the level of testing of the software. Deliverable of this phase is the complete application that satisfies all of the above.
- **Implementation:** Shortly before its final version, the software is used by end users. Delivery of this phase is the acceptance of system operation.
- **Operation:** In the final phase, the software is used more widely by users who are tasked with identifying possible problems, omissions or any changes and additions.

Although the phases seem to follow a sequence, completing each other first, in reality this sequence is seldom followed.

2.6. Project Life Cycle Models

Software development in the early years of computer science was dominated by the programmer, whose skills influenced the end result. The lack of a structured product delivery process was also evident, while the end user contribution was virtually zero, and did pose major concerns for the manufacturers.

The term "**Code & Fix Model**" was later used to describe the software development process, which included two stages of development, that of writing the code, and that of correcting any problems. This model created a series of problems, which made it clear, from early stages that the integration of other things was needed for proper development.

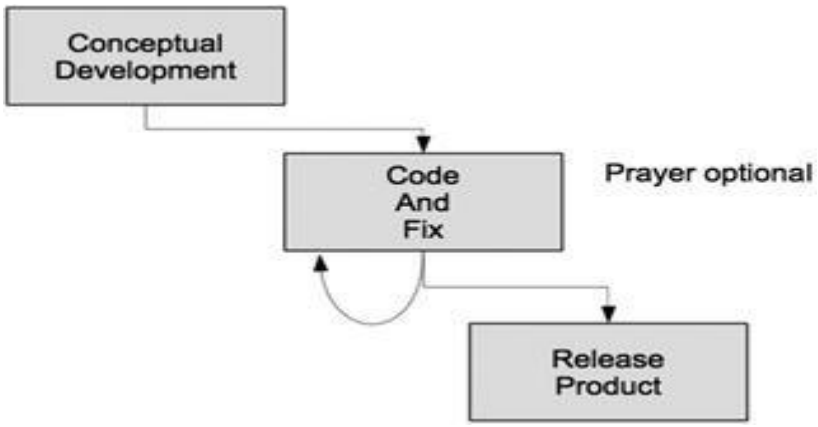


Figure 3 - Code & Fix Model – Source: *media.springernature.com*

At the same time, the need for a more structured way of developing software projects has emerged as a concern even from Bennington as early as 1956, transferring his experiences from developing large projects. The model he described is that of the "**Stagewise Model**", which consists of a series of successive stages.

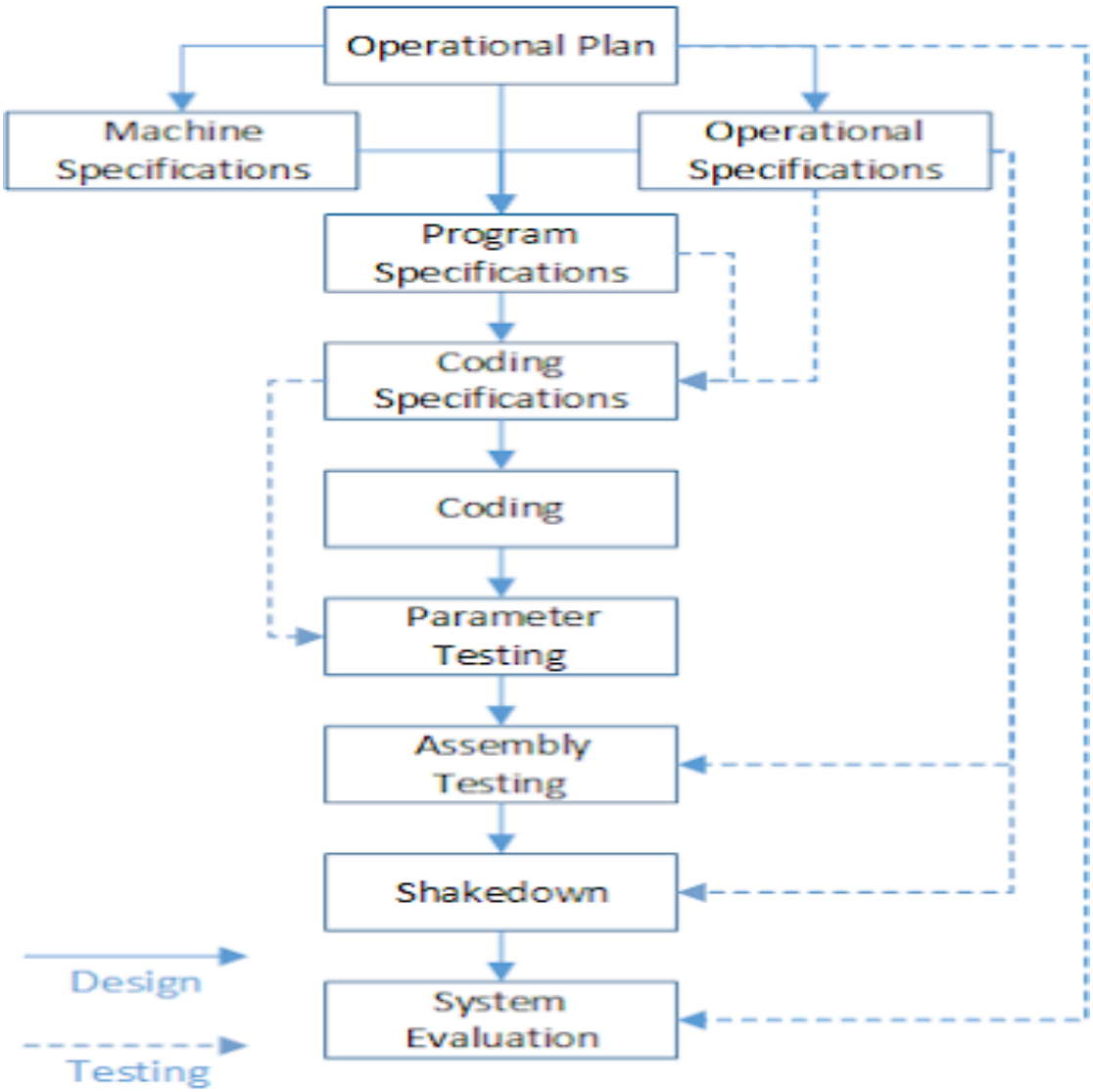


Figure 4 - Stagewise Model - *lh3.googleusercontent.com*

2.6.1. Waterfall Model

After several years of testing, in the early 1970s Royce presented the first attempt to describe a complete software development methodology. Its original name was "Large Systems Development Management", while later it was given the name "Waterfall", which is the evolution of the "Stagewise Model".

According to the author, there are two steps in software development, regardless of size or complexity. The first stage is that of **Analysis** which is followed by a **stage of coding**.

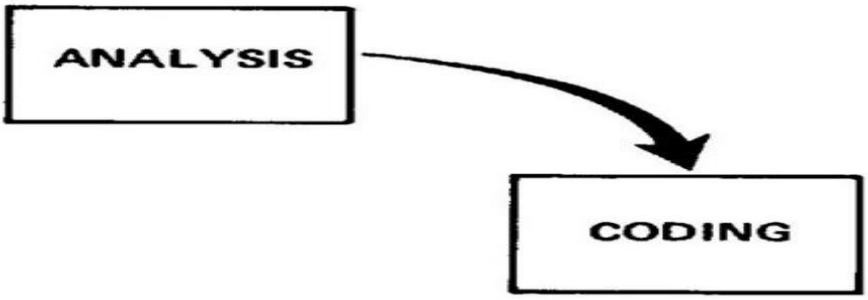


Figure 5 - 1st Step of Waterfall Model – Source: *arialdomartini.wordpress.com*

In the case of small systems, this sequence projects perfectly, while for large systems, additional development steps should be introduced which will contribute (Contribute) to the final product, increasing the development costs for it and which should therefore be monitored.

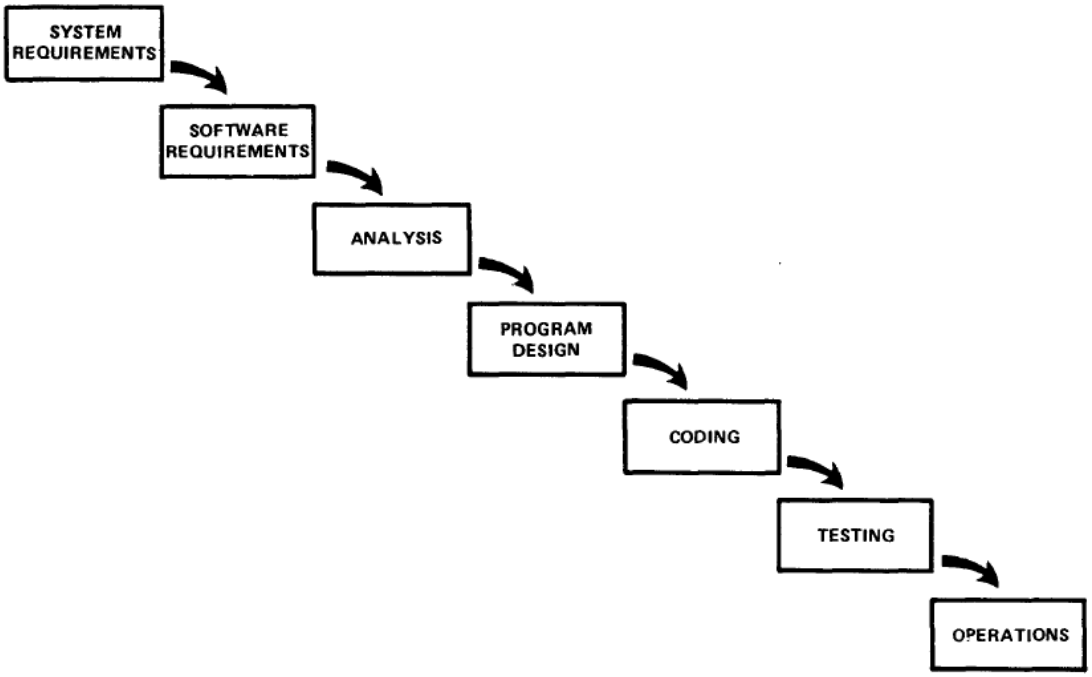


Figure 6 - 2nd Step of Waterfall Model – Source: *researchgate.net*

This introduces the two phases of requirements analysis, design and testing which are between the requirements. Expanding on the matter, the author emphasized the iteration relationship between the developmental phases. That is, as the steps progress, the design becomes more detailed, and there is an interaction between the preceding and succeeding steps, and very rarely with the further removed ones.

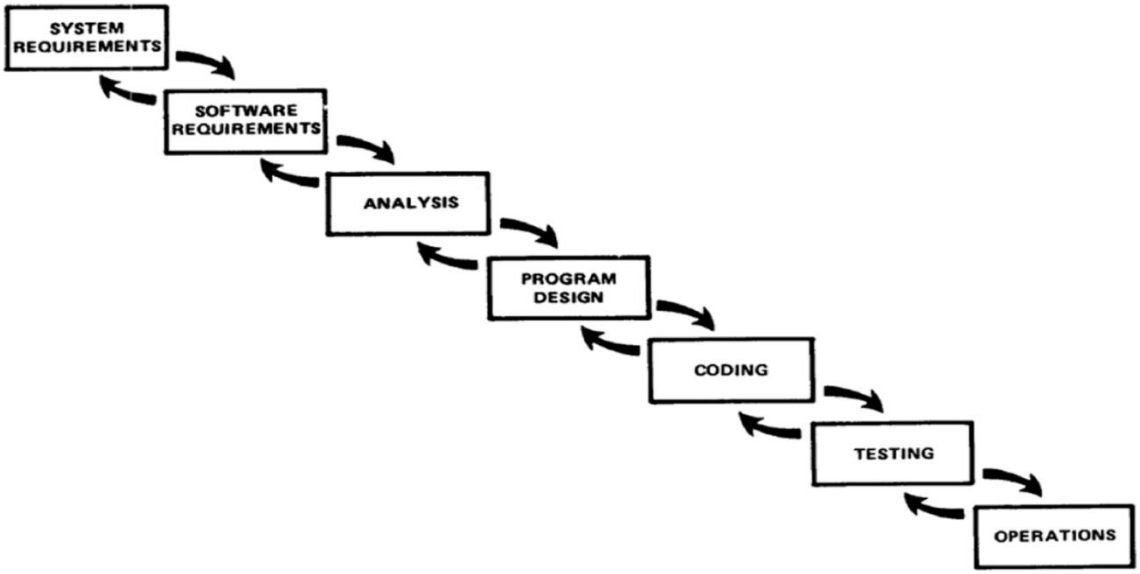


Figure 7 - 3rd Step of Waterfall Model – Source: *changearc.files.wordpress.com*

So, in detail, the phases that the "Waterfall Model" goes through are as follows:

1. **Requirements Analysis:** One of the major problems of software project failure is the incorrect recording of system requirements. This task should be performed from the outset and with utmost detail, by the people in charge of the system through interviews or other methods of elicitation, and in collaboration with future end users proceed to record the specifications and requirements. This helps the customers to fully understand what will be developed.
2. **Design:** At the beginning, an initial Preliminary Program Design is made without analysis, noting that it will not fail due to technical reasons (it can precede both the requirements phase and the analysis). The following is a general design of the system architecture. Therefore, a large degree of documentation is required at this stage. That is, 6 different documents are required until the "Design" phase is completed, which are the following: Software Requirements, Preliminary Design, Interface Design, Final Design, Test Plans and Operating Instructions.
3. **Implementation:** In this phase the software is completed including all its components. It is important that the system has been developed at least twice, so that the final version of the software that will be delivered to the customer, does not display any potential issues.

4. **Verification**: At this stage the final inspection of the software, as well as all of its subparts, is completed. This stage determines whether all the specifications are being met alongside ensuring that a complete system has been produced.
5. **Maintenance:** This phase takes place after the software has been completed and delivered to users. Also included here are possible changes that need to be made, bugs, improvements or even new requirements of the end users. Royce added that Design, Inspection and Testing are considered to be one of the most important phases in building software because it requires the greatest use of available resources. At the same time, he spoke about the importance of the client's involvement in the project. Based on all of the above, the final "Waterfall" model as proposed by Royce (1970) has the following structure:

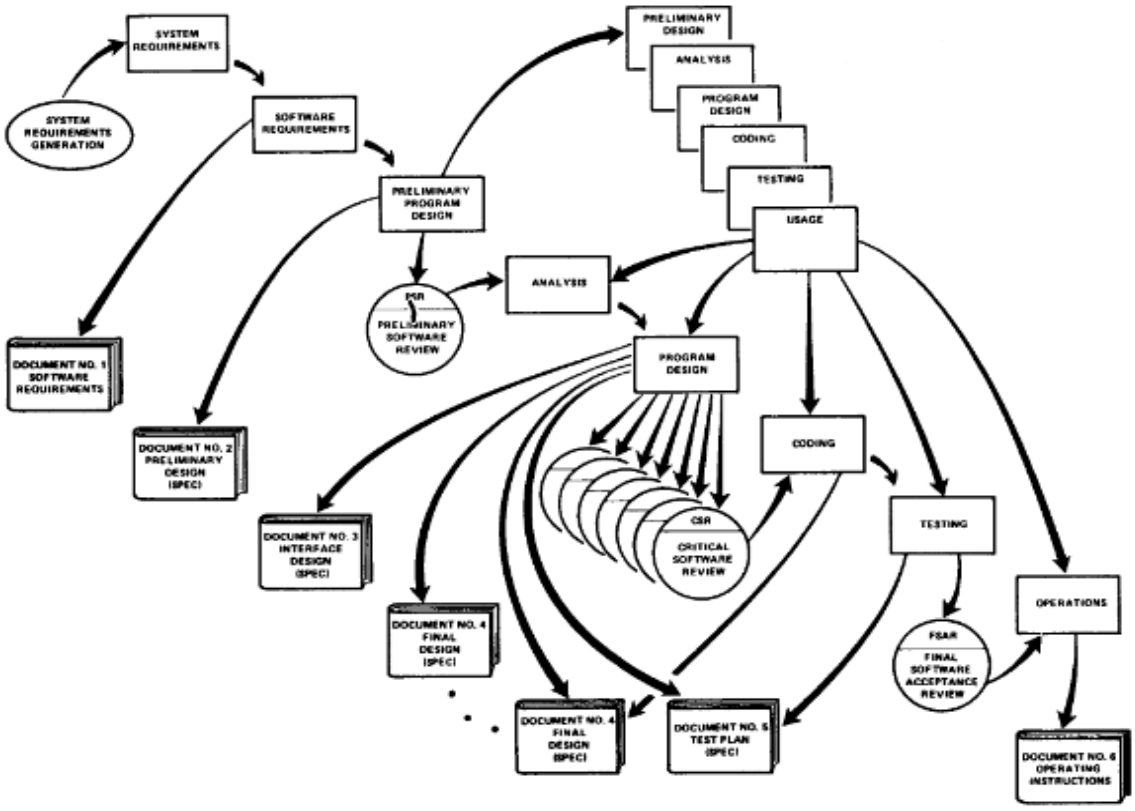


Figure 8 - Developed version of Waterfall Model – Source: *beza1e1.tuxen.de*

The **advantages** of this methodology are: (1) wide acceptance and use, (2) forming the basis for all newer methodologies, (3) clear phases of development with specific deliverables (ease of management - how to measure and develop the project) and (4) facilitating the drafting of contracts with the user requirements essentially constituting the essence of the contract.

The **disadvantages** of this methodology are: (1) the general dependence on documentation (production of forms), (2) the emphasis on the development of requirements analysis in documents, (3) the lack of flexibility, since in order to start a new task, the previous one must have first been completed, (4) the phase in which a project is located does not always reflect its actual development. In conclusion, the "Waterfall Model" is still one of the most established methodologies for the development and management of IT projects, although it has received and continues to receive strong criticism. According to the principles presented here, all subsequent models were based.

2.6.2. Rapid - Prototyping Model

From the mid-1980s, new methodologies began to be developed that sought to address the limitations of the Waterfall model. Typical examples are the Rapid - Prototyping model (Connell 1989) and the Spiral model (Bohem 1988), the Incremental (staged) Delivery model (Wong 1984) and the Evolutionary Delivery model (Gilb 1988), as well as the Rapid Application Development method (RAD) (Martin 1991) which is essentially an improvement on the former. Finally, in 1996 the Krutchten Rational Unified Process (RUP) methodology appeared, which introduced the logic of Object-Oriented Software Analysis & Design.

The **Prototyping model methodology** is a development model that builds partially incomplete versions of the software to be released. Each of these versions is called a **prototype**, which includes features and functions of the final software, but may be different from the final software. This method is used because it allows system users to evaluate software engineers' proposals by actually testing a prototype product, without needing to understand and evaluate specifications and descriptions of the software under construction, while simultaneously improving it. When all the requirements are met then it can be turned into a final productive software.

The prototype method is **an iterative process** like all models and contains basic development phases, where each new version meets the requirements of the users more precisely. The steps are as follows:

1. Identify and record the needs of the user.
2. Fast development of the functional prototype.
3. Test the prototype, where users are encouraged to express their views on ambiguities - questions and to determine which needs are not being fully met.
4. Prototype correction and improvement, where user-based designers change and improve the prototype.

The procedure followed is the continuous repetition of steps 3 and 4 until we have the final satisfaction, and the final product with all specifications met. Based on this **it is possible to get an opinion** on the application of the software at an earlier stage. This methodology also helps when there is **NO certainty about the requirements from the outset** by customers or end users.

The **main advantage** of this particular model is that it helps the user understand the system from the outset, while at the same time integrating it into the production process. On the other hand, an important disadvantage is that the system is designed and developed according to the waterfall model with all the consequences that this entails.

In conclusion, the **Rapid - Prototyping Model** presented by Connell in 1989, in order to be properly implemented requires an understanding of the business problem of each company, for which a solution must be developed. The schematic representation of the model is as follows:

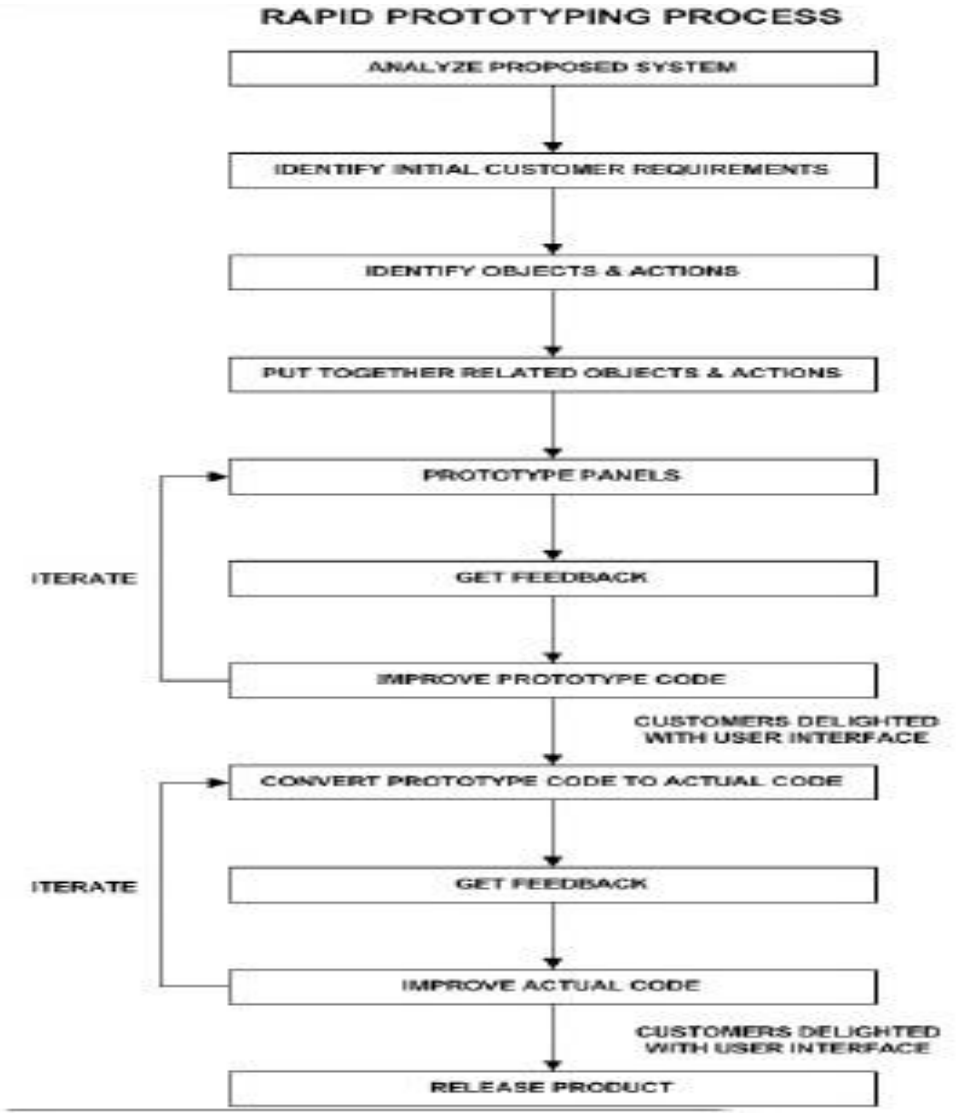


Figure 9 - Rapid / Prototyping Model – Source: *image.slidesharecdn.com*

2.6.3. V - Model

The "**V-model**" software development adopts many of the features of the waterfall model or we could say it is also a kind of waterfall, as the process that follows begins with a downward slope, followed by an upward slope.

The basic idea of this model is to determine the design and implementation requirements, on the left side of the letter "V" following a downward trend, while the unit and system control has an upward trend and is on the right side of the "V". Mistakes are slowly identified in their project life cycle, such as in the waterfall methodology.

The model adopted the letter "V" from the word **Verification VS Validation**. **Verification** answers the question of whether the product has been correctly manufactured, and concerns the process of comparing a product with the specifications or standards for debugging. The design is verified by comparing it with the requirements, while the code is verified by comparing it with the design. **Validation** answers the question of whether the right product has been produced, and is the process of comparing a product to high-level requirements, using a validation method to determine if the system does what the requirements expect of it, in order to be validated. The same methods are also used in verification.

The schematic representation of the model is as follows:

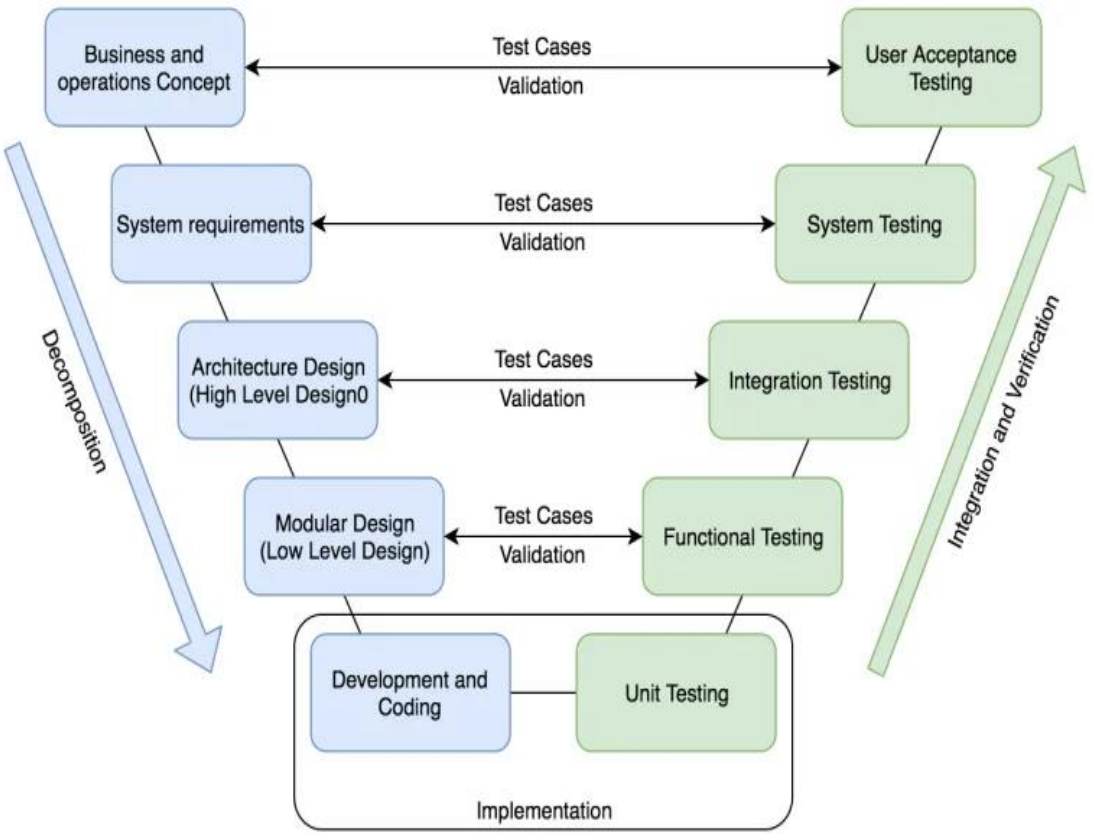


Figure 10 - V Model – Source: *i0.wp.com/melsatar.blog*

2.6.4. Incremental (Staged) Delivery Model

The **Incremental Staged Delivery Model** combines the serial development of the waterfall model, while at the same time it has the repetitive development of the prototype production model. The central idea is to divide the software into sections that are developed individually, while for each of them the method of the waterfall is followed, while at the final stage the merging takes place. From the beginning, the requirements of the users are determined, a small functional sub-part of the final system is constructed, and concurrently the other functions of the system are gradually added to it after a series of repetitive versions.

The main **advantages** of the methodology are the following:

- The user suggests improvements after the delivery of a subsection.
- From the first issues the client can control the return on investment.

The **disadvantages** of this methodology are:

- Upon completion of a subsection, it is not possible to return to it
- The use of the waterfall in these sub-sections.
- Any significant change in the design of the next subsection resulting from customer comments is not possible, and is made only in appearance and not in functionality and architecture.

The schematic representation of the model is as follows:

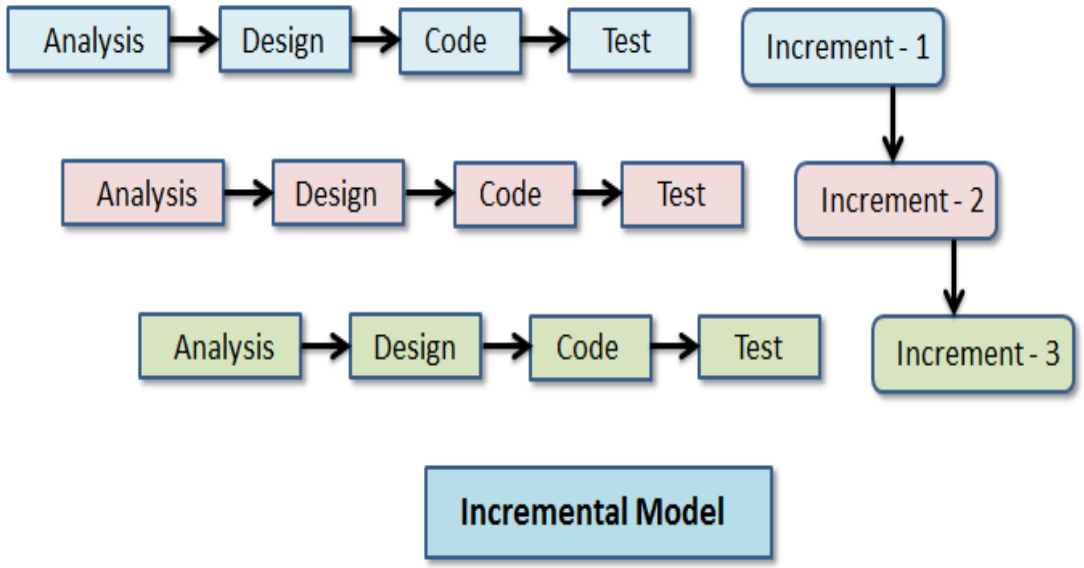


Figure 11 - Incremental (Staged) Delivery Model – Source: *cdn.educba.com*

2.6.5. Evolutionary Delivery Model

In the **Evolutionary Delivery Model**, the software development cycle is divided into smaller iterative cycles where at the end of each is recorded the user feedback, which enriches the knowledge and experience of the manufacturers for the subsequent cycles. Repeat cycles last from two to four weeks until customer requirements are met. The difference between this model and the previous one is the serial implementation and not the parallel development cycle of a software program. That is, at the end of each cycle and after the user's comments, the next cycle is designed.

The **advantages** of using this model are the following:

- Early delivery of parts of the system even if the requirements of the users are not clear.
- Exporting user requirements is done by using early versions as tools.

The **disadvantages** of using this model are:

- The estimate of the final cost and total time at the start is not clear as the requirements are not clear.
- Additional time for inspections
- Danger of securing the architecture, from the risk of improper merging of the pieces.

The schematic representation of the model is as follows:

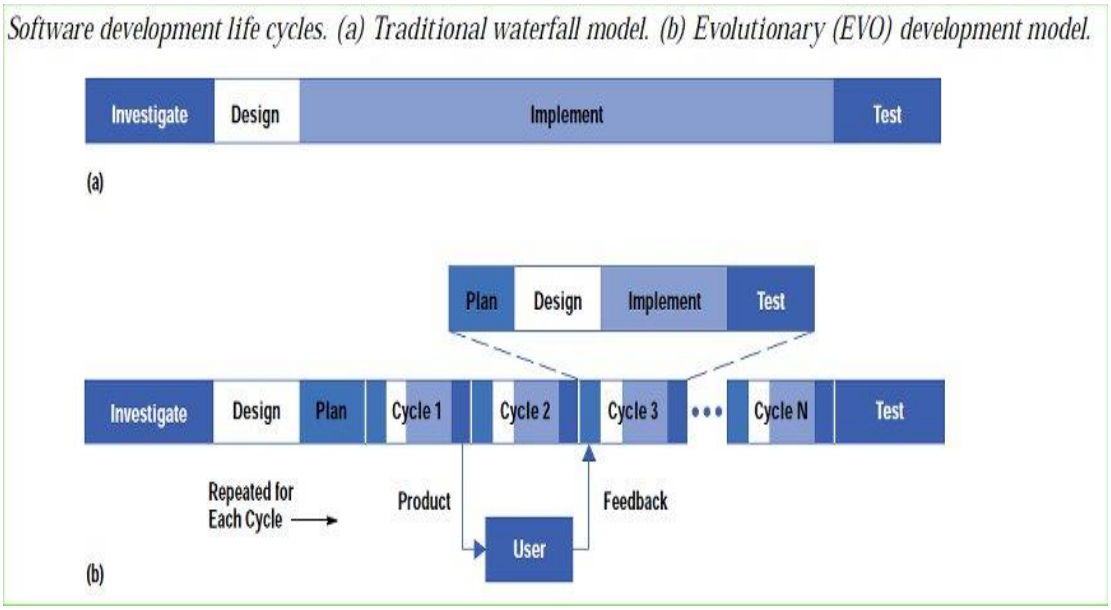


Figure 12 - Evolutionary Delivery Model – Source: *qualityguru.files.wordpress.com*

2.6.6. Spiral Model

The **Spiral Model** was developed by Barry Boehm in 1988 and presented with the article "A Spiral Model of Software Development and Enhancement", and as mentioned by him in the

article, its creation is based on the collection of experiences to improve waterfall applications in large government software projects.

It is not the first methodology to apply iterative development, but it was the first to talk about its importance. Each stage begins with a goal, which was completed with the participation of the user who can review the progress of the project. The analysis and application are presented in each phase of the life cycle.

The **main features and advantages** of the spiral model are that emphasis is placed on risk assessment in each turnover and on trying to minimize it. This is realized by dividing the project into smaller sub-projects with the obvious prospect of changing a sub system during its development. For this reason, software development is done in many workflows with gradual expansion of the functional features of the application.

In this model we distinguish four categories of tasks:

- Defining goals.
- Identifying and resolving risks.
- Execution of development processes and verification.
- The planning tasks.

The **weaknesses** of this model are its complexity, the difficulty to define - analyze the many sub-projects, the difficulty to keep the project team focused on the overall objectives of the project, the need for specialized executives with specialized knowledge in management and the lack of milestones during project development. The schematic representation of the model is as follows:

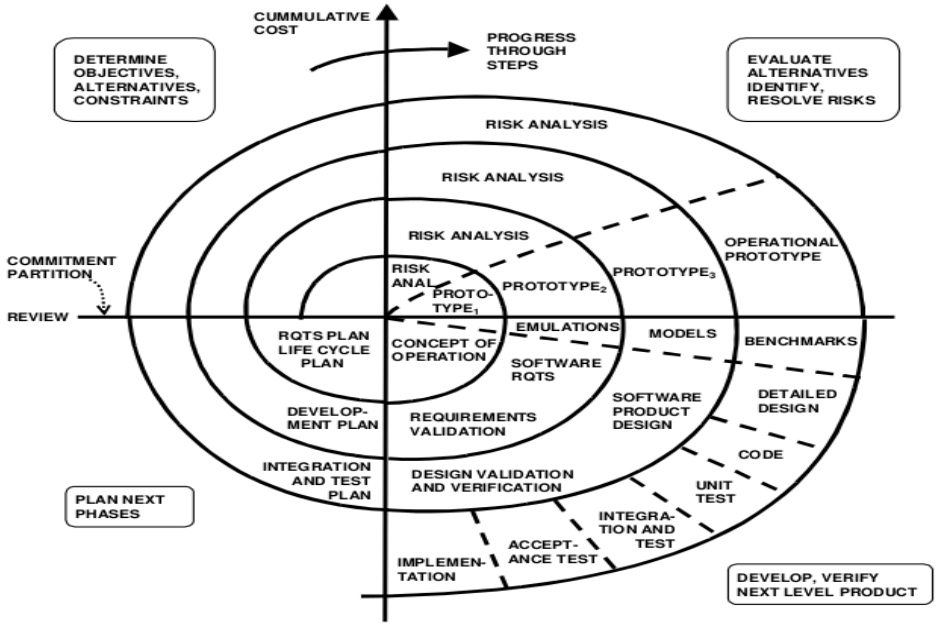


Figure 13 - Spiral Model – Source: *xbsoftware.com*

2.6.7. Other models, and the need for evolution

The life cycle models of software development projects throughout their existence were many and important as we saw above. In addition to the above, we could add as important the **Fountain (Pidakas) Model**, where the development follows an object-oriented philosophy and reuse of ready-made components. The **General Life Cycle Model** with development in cycles according to the characteristics and capabilities of the manufacturer. That is, a generalized form of previous life cycle models.

In the early 1990s, James Martin introduced **Rapid Application Development (RAD),** which is based on the evolution and improvement of the **Evolutionary Delivery Model**, while the main objectives of this methodology are: the high speed, high quality and low cost. A characteristic fact is that the active participation of the customer and all involved is required. The system is checked repeatedly by both the development team and the users.

Also, in 1996, Philippe Krutchten, in order to enhance and complete the UML language (Unified Modeling Language) which was a standard method of software modeling, presented the methodology **Rational Unified Process - RUP**. And this refers to the logic of object-oriented software analysis, design and development. An important feature is the use of Use Cases, which are used to model the requirements of users.

Regarding all life cycle models and processes (traditional and non-traditional), for the development and production of quality software there was a need for something new. This is because all of these methods have been described as inflexible, unproductive and bureaucratic, despite the constant emergence of new, multiple improved methodologies.

Summarizing:

- Software projects are often treated with the assumption that development is a predictable and iterative process.
- There is a lack of flexibility in terms of customer requirements. They are generally not defined in the initial stages and it is impossible to adapt to the later ones.
- Improper use - allocation of resources.
- The inability to adapt to the new globalized business environment.

2.7. Agile Methodologies

Examining the software life cycle models developed above in the past, one finds that due to the incomplete methodologies developed by software designers, the results of their efforts were either largely difficult to complete or even completely failed.

The need for integrated results led to the need to create new methodologies. Scientists studied all life models and came up with a fairly different approach, which includes: **iterative development** (small versions of software with rapid development cycles), **complete collaboration with the client** (who is actively involved in the implementation of the project) and **adaptability** (possibility of continuous changes).

Therefore, in February 2001, a team of scientists gathered at Lodge, a snowbird ski resort in the Wasatch Mountains of Utah, USA, for food and skiing, seeking common ground for software development through Agile IT project-specific methodologies. A number of methods have appeared in the list of methodologies. From this meeting the **Agile Alliance** (Partnership of Scientists and Bodies for the Documentation and Promotion of Agile Methodologies), but also the **Agile Manifesto** were established.

The word Agile translates in Greek to Flexible, so Agile could be the ability to move quickly, comfortably, be in flux and flexible. Software is now being developed in short iterative and incremental cycles, with the ability to adapt to the changes that have taken place from the ever-changing business environment. At the same time, its philosophy is developed face to face with the client, in order to reduce bureaucracy. As adaptable methods, they easily incorporate changes, while they are people-oriented and not process-oriented, promising better software with rapid development.

The first Agile methodologies appeared in the 1990s, as an alternative to the traditional rigid procedures. The **Structured Analysis & Design Technique process** by Davis in 1992 kicked off. Other major Agile methodologies followed: **Extreme Programming (XP)** by Beck in 1999, **SCRUM** (one of the most common with XP) by Schwaber & Beedle in 2002, **Crystal Family** by Cockburn in 2002, **Feature Driven Development (FDD)** by Palmer & Felsing in 2002, **Dynamic Systems Development Method (DSDM)** by Stapleton in 1997, **Adaptive Software Development (ASD)** by Highsmith in 2000, **Lean Development** by Poppendieck in 2001 and the **Unified Process** by Kructhen in 2004.

2.7.1. Basic Values & Principles

As mentioned above, the word "Agile" in Greek translates to Flexible, so Agile could be the ability to move quickly, comfortably, be mobile, be flexible. The Manifesto of Agile Processes consists of certain fundamental principles, values and qualities.

It should be understood that, as its creators say, Agile methodologies do NOT claim that procedures, tools, detailed documentation and contracts are trivial and insignificant. In fact, the opposite is true. They just elevate the importance and contribution of the customer towards the end result.

Initially, the **four core Values** enshrined in the wording of the Agile Manifesto are:

1. **People and Interactions VS Processes and tools**. This value emphasizes the importance of the human factor. Developers, managers, project managers and clients need to work together, interacting, to better develop software. Practices that one encounters here are the planning in pairs, the continuous participation of the client in the project, the existence of small groups, etc. In conclusion, if the cooperation between the participants is NOT successful, the process and tools will not be of any significant help.
2. **Pure Code VS Written documentation.** The main goal of this value is to deliver a complete piece of software devoid from any issues, deficiencies, etc. On the one hand emphasis is placed on the delivery of deliverables and technical documents and while this can change constantly, without synchronizing with changes. On the other hand, it helps in the retention of knowledge and information. In conclusion, a simple and good quality code should be produced from the outset, which is better understood than detailed documents and documentation diagrams as the latter ultimately play a secondary and complementary role.
3. **Cooperation with the client VS Strict contracts.** The main goal here is the active participation of the client in the project, undertaking to play the leading role in the decision making. Gone are the the times of manufacturers going through lengthy processes to make changes through documentation, specifications, etc. followed by additional processes of signing off changes after obtaining customer approval. As this involves many risks such as delays, depletion of resources and most importantly the project failure, contracts are not an unnecessary practice, as they determine the limits within which the signing parties should operate. In conclusion, a close cooperation with the client is considered necessary, so that in the end their needs are fully met, mostly adhering to a contract which can, however, be superseded by mutual agreement should the needs of the completion of the project change. This way a relationship of trust is built and maintained.
4. **Responding to changes VS Following plan.** The main goal of the last fundamental value is that during the construction of the software, we must adapt to any change, which is a very common phenomenon and can be affected by various factors. On the one hand we have to deal with the inability of the client to express all the real needs prior to the start of the project alongside the demand for rapid development and delivery of the final system. On the other hand, we have a detailed plan that, while remaining attached to it, we may not adopt changes for various reasons. In conclusion, a plan for the implementation of a project should be defined from the outset, while at the same

time any change or deviation should be considered "normal" by those involved, and therefore acceptable.

"Agile" methodologies place more emphasis on what is on the left than on what is on the right.

Along with the fundamental values of the Agile Manifesto, the proclamation set out the **12 Principles** that guide the development of the software by adopting an Agile environment. These are the following:

1. The Highest priority is the customer's satisfaction through early and continuous software delivery
2. Any change in requirements is welcome, even in the final stages of the software development, working towards the benefit of the customer.
3. Frequent software delivery is considered necessary, and should be done at short intervals, e.g., in a few weeks or months.
4. The client and the project team should work together on a daily basis throughout the project.
5. The executives during implementation should be motivated e.g., through a proper and organized environment, while at the same time they should be supported if the need arises.
6. Daily meetings with face-to-face conversations so that the information is transferred correctly to the development team.
7. The primary role is the proper operation of the software as an indicator of project progress.
8. Agile methodologies promote a sustainable development process so that all stakeholders maintain a steady pace throughout the project.
9. Constant attention to technical perfection and good design with enhanced flexibility.
10. Simplicity is essential.
11. The best architectures, requirements and designs, result from self-managed teams.
12. The development team will regularly discuss ways and methods so that there may be more effectiveness through redefining its behavior.

In conclusion, as it is understood, in order to confront the above principles and values, some conditions must be adopted in order for it to be adopted. Initially, the client must be at work every day to take part in the software development, decisions, etc. The members of the software development team should be close by to participate with the client in creating the software and making the decisions. So, the documentation does not play the dominant role, while the requirements of the users emerge during its development. The development team will consist of staff who have extensive and specialized experience, ready to respond to any change that

may occur, while at the same time they may limit the process of evaluating deliverables to some informal reviews and tests. Reusing part or the entirety of the final software is not an initial goal, as the software is evolving continuously. Finally, the cost of change should not increase significantly over time.

2.7.2. Constraints and benefits with adoption of Agile Methodologies

According to Cockburn and Turk (2002), with the adoption of Agile methodologies, and their respective principles, values and conditions, inhibitions and restrictions relating to their uses are slowly beginning to emerge.

We have several **constraints**:

1. **Lack of specialized staff.** The lack here pertains to the administration's lack of knowledge as regards the application of Agile Methodologies.

2. **Direct communication.** An important restrictive factor is the communication among team members. As we have mentioned, the communication among the members but also between the team and the client should be direct, and where possible interpersonal. This is a risk for companies in which project teams are located in different places, while there should be continuous communication even with the use of modern digital media.

3. **Quality assurance.** Ensuring the quality of a software program is equally important. It is a controversial and complex factor. Because, on the one hand we have the full adoption and fulfillment of the procedures of international standards, with whatever this may entail, while on the other hand the Agile methodologies come into conflict with the full sequence of international standards that require great documentation. It should also be taken into account that many companies have invested in such systems, which is difficult to ignore.

4. **Standardization or differentiation of user requirements.** One of the most important values of Agile methodologies states: "**Working with the VS Strict Contracts client**", which emphasizes that new user requirements should be adopted immediately, ignoring standard procedures and contracts. Therefore, any misunderstanding between the parties should be resolved immediately.

5. **Project team knowledge and skills.** Slightly different from the first constraint, the constraint here has to do with staff who should be both technically trained and also have knowledge and training in Agile methodologies.

6. **Cost issues.** It is well known that software projects in the past faced cost issues due to the development life cycle models they followed resulting in overpricing to failure. Re-using of parts or the end result, process automation can result in cost reduction.

7. **Control mechanism.** Development team control procedures, communication, etc. should be constantly developed so that the early versions as well as the final product will be produced with the desired quality.

8. **Security.** Ensuring security is also a significant limitation, as during this software development major obstacles may arise.

On the other hand, by adopting Agile methodologies we have the following **benefits**:

1. **Faster software development.** With the use of Agile methodologies, we have the increase of software development while at the same time we achieve reduction of the delivery time to the customer, and its release to end users.
2. **Better return on investment.** As mentioned above in the case of cost constraints, reusing parts ort final version of a software program, or automating processes can reduce the cost. This results in maximizing the return on investment.
3. **Early cancellation of problematic projects.** As mentioned in earlier chapters on Agile methodologies, classical life cycle models are often proven deficient in estimating the size of a project in the early stages, and this may result in the occurrence of long delays - overdue schedules even in the advanced stages of the development of the project. As a result, the client may not wish to allocate more resources, leading them to cancel the project and losing all initial investment. On the other hand, Agile methodologies, adhering to the entirety of principles and values of the Agile Manifesto, give the client the opportunity early on to review the cost, the design and the benefits of the project, so that they can decide whether to continue or not, resulting in less damage.
4. **Better quality.** Agile methodologies emphasize on the production of high-quality software, in contrast to the traditional life cycle models. This is accomplished by increasing the degree of achievement of user requirements, since the customer is in this case involved throughout the design and development of the software.
5. **Improved control.** With continuous versions of the software at short intervals, a complete picture of its quality control is provided, compared to traditional models where the validation takes longer. This is also achieved through the accessibility of information by all parties.
6. **Reduction of dependence on specific individuals.** As mentioned above, that is, with the accessibility of information by all parties, and at all stages of the development of a software program, the situation in which its development is highly dependent on one person or many, can be avoided, alongside any potential problems which may arise from this situation.

7. **Increased flexibility.** As mentioned, throughout earlier chapters, Agile methodologies offer the opportunity to change at all stages of development to meet the customer's needs by giving them flexibility in designs with the traditional models. This can provide the customer with a software program satisfying their requirements, which most likely were not clear at the early stages, allowing them to change, compared to the traditional models, which prohibit these changes during their development.

In conclusion, the use of Agile methodologies, their principles, values and conditions, taking into account all the limitations and benefits, can lead to the success of publishing a software program, meeting the requirements of each demanding client. This is clearly demonstrated by studying various scientific articles - case studies and their application (Adapting Agile Practices in University Contexts, Masood, Z., Hoda, R. & Blincoe, K., 2018 // Challenges of adopting agile methods in a public organization, Nuottila, J., Aaltonen, K. & Kujala, J., 2016 // Critical success factors and limitations for the lightweight software process improvement in Agile development: A literature review, Kouzari, E., Gerogiannis, V., Stamelos, I. & Kakarontzas G., 2015 // Identifying some important success factors in adopting agile software development practices, Misra, SC, Kumar, V. & Kumar U., 2009), clearly sees that the limitations and benefits of implementing Agile methodologies coincide.

2.7.3. Models of Agile Management

2.7.3.1. EXtreme Programming - XP

By "extreme" programming we refer to the use of Agile methodologies at "extreme" levels. The Extreme Programming (XP) methodology was first introduced in 1999 by Beck. This methodology was innovative because, while all practices where pre-existing and well known, at this point they were all merged in order to produce a single novel methodology for software development. This methodology includes a series of principles and values given to the development team at the outset and through which we achieve the satisfaction of all requirements, at all stages of development.

According to Beck, the basic principles and values of "Extreme Planning" are:

1. **Communication:** Enhancing communication among all parties in order to reduce implementation problems,
2. **Simplicity:** With simplicity as the basis, we can work with flexibility, speed and ease. It concerns all the development processes of a Software program,
3. **Feedback:** According to the continuous feedback of the early versions, we can ensure the full satisfaction of customer requirements,

4. **Courage:** Here we refer to the ability to identify changes and make decisions to successfully deal with new challenges that may arise and
5. **Respect:** Concerns the relationship among the parties and the possibility for equal participation in the development of the software.

The life cycle of a project according to "Extreme Programming" for the development of a software program consists of the following phases:

- **Investigation:** The investigation phase can last from a few weeks to a few months, depending on the team's ability to cope with the software development technologies. Initially, the customer's requirements are recorded and used to produce a model for the consequent design of the system. The elicitation of user requirements is accomplished through user stories (in story cards), the result of which is obvious with each updated version of the software. Although the XP life cycle does not present a clear goal, this becomes easily visible using user stories. What finally follows is the selection of tools, technology and practices through which the implementation is initiated.
- **Planning:** The planning phase lasts for a few days. During this phase, the user stories are ranked, while concurrently the parties (client and development team), co-decide the first and the subsequent versions of the software that will include those features that have been decided by the client. In order to perform the above process, the task list of each story, the development time, the delivery date, and finally the stories chosen to be included in the final version of the software must be developed for each user story.
- **Repetitions:** The duration of a repetition of the total repetitions for the first version lasts from one to four weeks, so that in the end, i.e., the end of the repetitions, the software is ready to be released and put into operation. The start is accomplished with a repetition with phases of analysis, design, planning, testing and completion. New features are added to each iteration until the final system is implemented. At the end of some iterations, we have the delivery of the first version, and the rest of the iterations finally follow. With the first version we can expect that we have an initial format of the system with the architecture of the final system. This is accomplished as in the programming phase, through user stories, where again the final responsibility falls on the customer.
- **Production:** This phase includes repetitions that have a very short duration, and are aimed at perfecting the software, without functional improvements. So, in this phase encompasses all the technical tests and inspections, so that the software is ready to be delivered to the customer. Finally, if new needs arise, problems are identified, or the system can be improved then we return to the previous phase and the new features that arise become available in the next version.

- **Maintenance:** In this phase we have the corrections of errors or any problems, also in addition to the resulting new versions of the software. In the phase of the new versions, the procedure of the previous phases is followed.
- **Withdrawal:** In the last phase of the XP life cycle the software documentation is implemented. At this stage, and since the client has NO other user stories to implement, since the software has met all the requirements, any changes that may occur are not accepted. However, the customer may withdraw it at any time during implementation or operation if their requirements are not met.

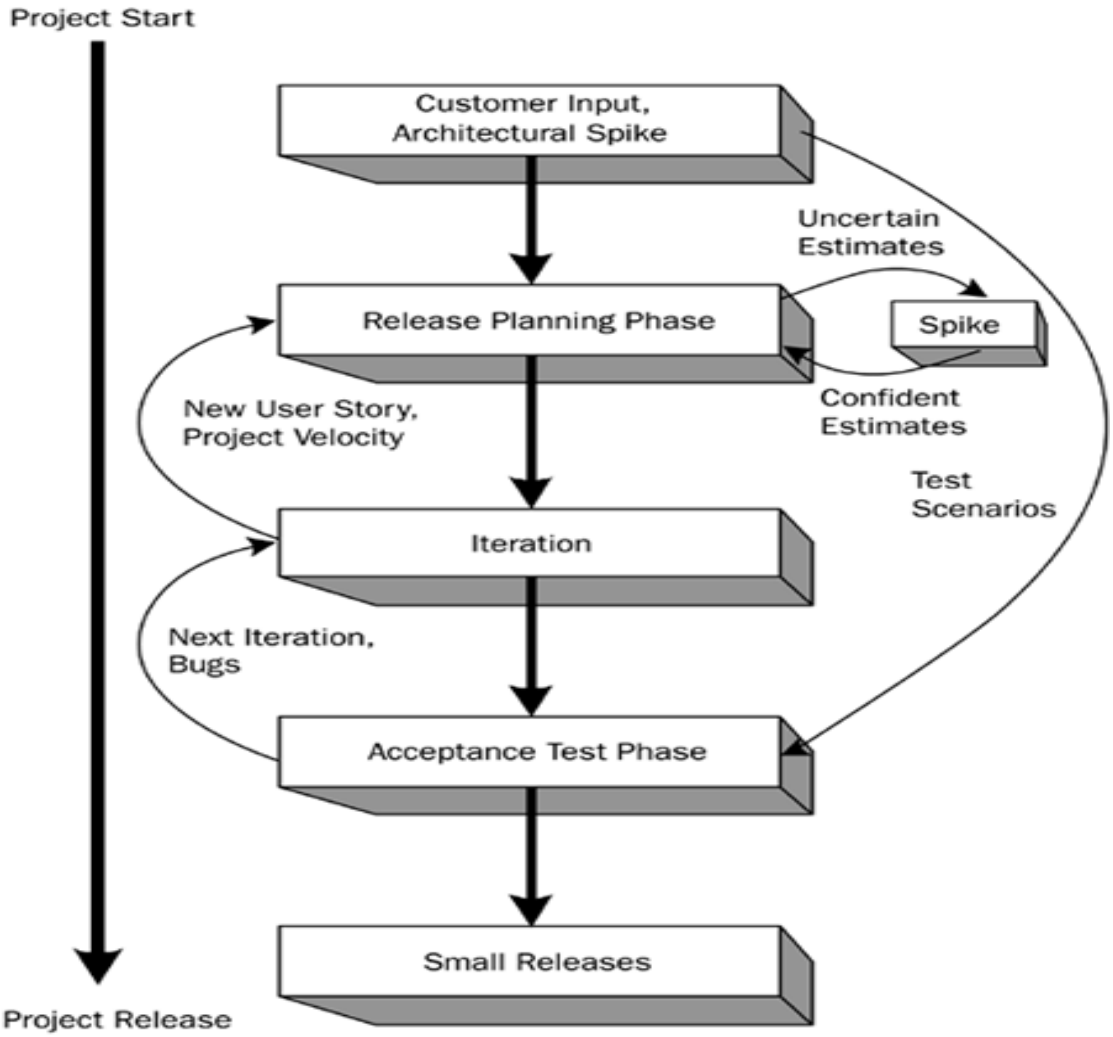


Figure 14 - EXtreme Programming (XP) – Source: *flylib.com*

For the implementation of the software, roles should be defined so that everyone has the corresponding responsibility. The **developer** writes the code and performs the tests in a simple and accurate way, while efficient communication with all other members is required. The **client** writes the user stories and the acceptance tests, puts them in a hierarchy, while making the decisions for the implementation. The **auditor**, through the various control tools, regularly

checks the system, communicates the results and maintains, while at the same time helps the client to write the functional audits. The **recorder** is an important role for the XP process, because it records the estimates, provides reports helping to improve the process, and finally records the progress of the implementation of each iteration and evaluates whether it is possible to achieve the goal. The **mentor** has the most important role for XP, because they are responsible for the observance and application of the method to all parties, and this requires excellent knowledge. The **consultant** provides their expertise when the need arises and is usually an external partner. At the top of the pyramid, **the manager** is the decision maker, while also communicating with all the previous members so that they maintain the complete picture and control of the process. Finally, the implementation of the software development phases requires a set of basic practices as presented by Beck. These are:

1. **Small versions of software:** Small versions of software refers to duration, and here Beck recommends a few days, weeks and up to two months. The software here is developed gradually through iterations, which as a whole comprise a version that should be completed within a short period of time as mentioned at the beginning.
2. **Simple design:** as mentioned earlier, with regards to all Agile methodologies process simplification is significant. This also prevents the complexity of the design in XP, with the possibility of simple implementation solutions. Unnecessary code snippets should be removed, and new features should be easy to integrate.
3. **Programming game:** During the meeting between the client and the development team, and while the user stories are explored - developed, it is decided to prioritize the requirements, i.e., which stories will be included in each iteration or release, and which of them will be rejected. The above process is called "Programming games".
4. **Test-guided development:** As mentioned in the XP life cycle phases, for proper software development, inspections and testing procedures should be followed. The technique followed is the creation of code control units (uni-tests) of the code to be developed. This is actually undertaken in order to determine and confirm the expected function of each code piece.
5. **Reconstruction:** In this phase since we have introduced alterations in a part of a code, there will inevitably be changes, iterations, strong dependencies, structure problems, complex classes, etc. in other parts of the code. The goal of this group will be to simplify the final code so that it may be understandable and more easily scalable.
6. **Programming in pairs**: Here the code is written by two programmers simultaneously on one computer. One of the two programmers is the one who writes, while the other simultaneously observes and evaluates, controls, reconstructs, etc. The roles of the two programmers are not static but dynamic and they must constantly switch.

7. **Continuous integration of changes:** During the development of a software program, it is always anticipated that there will be changes, additions and corrections of parts of the code which should be integrated at least once a day. Following this process of integration, one is expected to go through the testing process so that the changes are accepted and successfully assimilated. With this process we may always expect a system to be in check at any time.
8. **Customer Involvement:** From the start of software development, it should be ensured that the client, who is also the end user of the software, should interact closely with the development team. Goals should absolutely be set with the customer's approval therefore ensuring that the developed software meets all of the customer's requirements.
9. **Collective ownership:** At this stage, each member of the development team, if capable, may change any part of the code at any time. In effect, there is no single person who is solely responsible for a piece of code. An important advantage of this process is the easy identification of errors and ambiguities which may then immediately be corrected.
10. **Continuous recording of the development:** Upon the completion of the development of a software program, there should always exist the associated documentation. The documentation should record and present all the accumulated data during the implementation of the project, in order to produce reports and statistical graphs, etc.
11. **Coding standards**: Where the programming is undertaken in pairs the development team, in intra-team communications or communications with the client, should follow standards that are common for the development of the code and the final software. The goal is to have consistency, adequate communication and mutual understanding among those involved.
12. **40 hours of work:** A limit of working hours during a week is advised and overtime should be avoided. The aim is a pleasant, creative and efficient work environment, but also the identification of any delays, enabling a development process devoid of problems that may negatively impact the project.
13. **Shared project space:** The space in which the software is being developed should allow for direct face-to-face communication which should be continuous and uninterrupted for everyone involved, while of particular importance here is the customer's participation.
14. **Symbolism:** While composing the contract or the initial design, the software development team together with the client should define - describe the rules - the symbolism, which will make both their communication as well as understanding of the system easier.

15. **Simple rules:** The rules, which are set at the beginning, should be followed by all the contracting members. In case it becomes apparent that they cannot be followed, they should be replaced at any time, if agreed by all members.

2.7.3.2. Scrum Methodology

The Scrum methodology is a software development methodology, and concerns the organization, operation, design and management of Agile projects. It is an iterative and incremental methodology for developing software projects. It owes its existence to a holistic approach of 1986 by Takeuchi and Nonaka. The term "Scrum" was adopted from rugby, and connotes a game strategy. In 1995 at the OOPSLA '95 conference Sutherland and Schwaber defined the basic features of the methodology, while in a book published in 2001, Schwaber and Beedle describe the method in detail.

Specifically, it is an empirical method, which develops the ideas of the theory of industrial processes control information systems development. Emphasizing on how the project team will function properly to produce software agility, in ever-changing environments with constant control and monitoring. So, it is not a specific technique of implementing software.

There are three primary concepts that underpin the implementation of empirical methodology. First in line is **Transparency**, where some parts of the process are visible to those responsible for the result. It demands these sections to be in a common standard so that those who tend to them can share a common point of view. This is followed by the **Inspection**, where the development should be done frequently, in order to identify existing differences. It should not be undertaken too often so as not to interfere with the software development work. Finally, **Customization**, when one or more elements of the process are marked as deviating from the standards, then the software should either not be accepted or should be adapted. The adjustment should be made as early as possible to minimize any deviation from the standards.

The **life cycle** of a project according to the "SCRUM" methodology for the development of software consists of the following phases:

• **The Pre-game phase:** This phase is divided into Programming and Design (Software Architecture). Initially, the Programming aims at the development of the system, initially creating a Product Backlog List, including all the requirements of the system under development. The requirements as we have said are determined by the client and the development team. Subsequently, they are ranked, and during their development they are constantly renewed and updated to the new needs. Also, in Planning, the development team, the tools to be used, the allocation of resources, the risk assessment and the type of control are determined. In each iteration, the revised Backlog List

becomes known to the team for implementation by everyone. In the Architecture phase, the complete design of the Software is undertaken, based on the Backlog List that was been created beforehand so as to create an initial plan detailing what the system will eventually include. In the case of an existing system, the observed changes are also recorded in the Backlog List.

• **The Development phase:** With this phase we have the application of the "Agile" part in the Scrum methodology. The various technical and external environmental parameters can be differentiated during the implementation and constantly monitored through various practices during Sprints. Sprints are the iteration cycles during software development in which the system functions are implemented. Each Sprint includes requirements recording, analysis, design, development and finally delivery.

• **The Post-game phase:** In this phase the specific version of the software is finalized. It is initiated by the joint decision of all parties, that all requirements have been met. Here the system has been issued and put into full operation, while every new requirement cannot be added. In this phase we have the preparation of the software for full operation, the final checks as well as the documentation of the system.

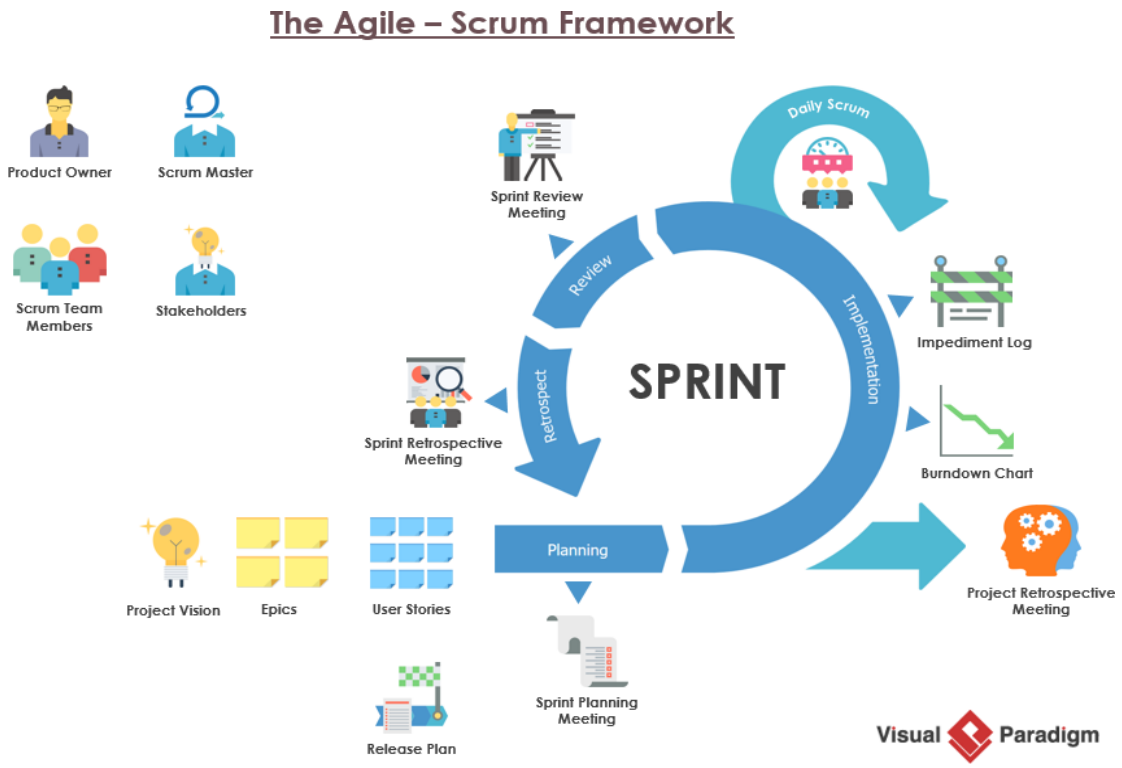


Figure 15 - SCRUM Methodology – Source: *visual-paradigm.com*

For the implementation of the software, **roles** should be defined so that everyone has the corresponding responsibility. The **Scrum Master** is responsible for certifying whether the project is implemented in accordance with Scrum values, practices and rules. They collaborate with all parts of the project acting as a link, while being able to ensure the productivity of the project. The **Project Owner** is the one delegated by the Scrum Master, the Client and the Management, making the final decisions, since the former's responsibilities entail the proper management and control of the project. The **Scrum team**, who during the development of a software program can amount to more than one, working in parallel, has the responsibility to decide on all those actions that must be implemented in order to achieve the goals of each Sprint. The **Customer** actively participates in the creation of the Backlog List and in its review. Finally, the **Management** is responsible for finalizing decisions, goals and requirements of the software.

Finally, the implementation of the phases of software development requires a set of basic practices. As mentioned above, Scrum is an empirical methodology without a specific development method. In order to have proper development, it is necessary to use specific practices and tools that prevent the emergence of increased complexity and weaknesses in predicting the consequences of the new requirements. These are:

1. **Product Backlog:** It is a list, which is constantly updated, and describes all the technical and operational requirements for the development of a software program, and in which the hierarchy takes place, in order to determine all the features that are necessary for the final product. Responsible for maintaining the Product Backlog is the Project Owner.

2. **Effort Estimation:** Responsible for the implementation of the process is the Project Owner along with the Scrum team. Assessing the human resource effort is an iterative process of the Product Backlog, while analyzing in detail each individual item when the necessary information is available.

3. **Sprint:** A Sprint lasts for about 28 days. Here the Scrum team aims to implement a piece of the final system within a Sprint. It concerns the part of the final system, after the Product Backlog List has been completed, and the Sprints Scheduling Meeting has been organized, in which the Product Backlog List is ranked by implementing it within a Sprint.

4. **Sprint Planning Meeting:** In this phase, all the contracting parties of the project meet in order to co-decide in detail the requirements of the part of the system that will be implemented in the next Sprint. Subsequently, the Scrum Master and their team will focus on exactly how the requirements are to be implemented during this Sprint.

5. **Sprint Backlog:** It is the starting point of every Sprint, while it remains unchanged throughout. It is a subset of the contents of the Product Backlog List with all the requirements that will be implemented during the specific Sprint of the Sprint Planning Meeting.

6. **Daily Scrum Meeting:** Daily meetings should be short, about 15 minutes, and can be attended by the Scrum Master and their team alongside the Management, the client and the Project Owner. Their purpose is the continuous control of the progress of the project, the best planning and the identification and recording of problems.

7. **Sprint Review Meeting:** It takes place at the end of each Sprint, where the Scrum Master and their team meet with all the other project partners and present the detailed results of the Sprint just completed, so that the next steps may be decided.

2.7.3.3. Crystal Family Methodology

The Crystal Family is a family of methodologies, and refers to different types of projects, where the most appropriate method is selected, including principles, so that it can easily be adapted to the requirements of the project. The father of "Crystal Methodologies" is Cockburn, who first discusses these methodologies in 2002.

The name of the crystal comes from the characterization of the projects in terms of size and criticality, corresponding to that of minerals, color and hardness. That is, each member of the family of Crystal Methodologies is characterized by a color, showing how intensively the methodology is applied, but also the degree of difficulty of the project. Larger projects with increased demands are characterized by darker colors.

The criticality of a project is determined by the consequences that the failure of a system will have. According to Cockburn, the degree of criticality is characterized by the characters: C, D, E and L. Where **C**: Comfort, **D**: Discretionary Money, **E**: Essential Money and **L**: Life. In practice we could say that C denotes system collapse paired with loss of comfort of the end user to use it, while in contrast L may denote that the collapse of the system, may also have consequences for the end user as severe as human loss!

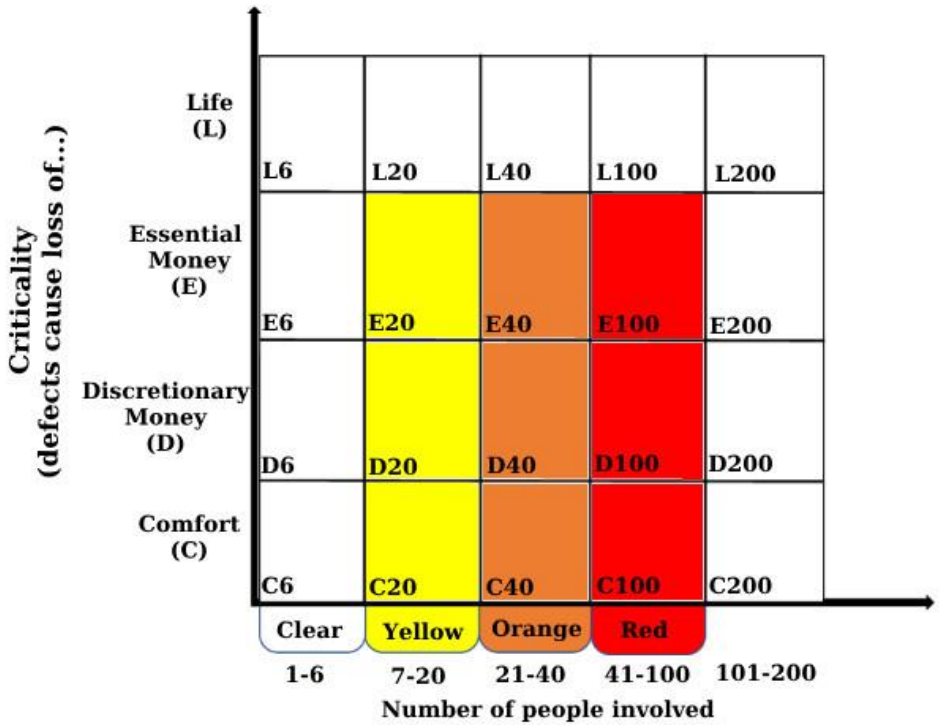


Figure 16 - Crystal Family Methodology - *hangoutagile.com*

Some examples of methodologies are: Crystal Clear (the most common), Crystal Yellow, Crystal Orange, Crystal Red, Crystal Violet, Crystal Sapphire, Crystal Diamond, ranging from those applied to the smallest projects, up to the largest.

Despite their complete categorization, these methodologies have certain rules, characteristics and values that are common to all. Here the project is implemented in incremental development cycles (incremental development cycles), with a maximum duration of repetition of four months, with a preferred duration of one to three months.

These methodologies do NOT support distributed growth; Therefore, development teams should be very close. Additionally, the style of the code, the business culture, and the desired technology environment are not specified from the outset (Qumer & Henderson - Sellers, 2008). Finally, there is no commitment to the development practices and tools that will be used, allowing for the parallel development of other methodologies such as XP, Scrum, etc. (Abrahamsson, 2003).

The **Crystal Methodology Development Process** includes instructions on policy standards, deliverables, local issues, tools, standards, and roles to follow when developing software.

As regards the implementation of the software, **roles** should be defined so that everyone has the corresponding responsibility. In Crystal Clear, the key roles of the project are: the sponsor,

the experienced programmer-designer, the programmer-designer and the user. These roles can have multiple sub-roles. In contrast to the others, in Crystal Orange, where there are more teams, we can have the above roles with additional ones grouped in those of system planning, architecture, technology, functions, infrastructure) and external auditors.

Finally, the implementation of the phases of the construction of software, requires a set of basic practices - techniques. These are: (1) **The Staging**, concerns the planning of the implementation of each sub-version of the final system, (2) **The Revision & Review**, where in each sub-version we have many repetitions with actions such as: construction, demonstration and review of data, (3) **Monitoring,** the evolution of the implementation is undertaken according to the deliverables in relation to their evolution and stability, (4) **The Holistic Diversity Strategy,** this strategy is used to divide large groups into smaller interoperable ones, in order for all groups to have the necessary know-how in multiple specialties, (5) **Methodology - tuning**, where, through interviews and workshops with the executives of the project team, the "Adjustment" of the methodology is attempted, (6) **User Viewings,** where in small projects the checks are performed once at the end by two users, while in larger projects the checks occur three times (7) At the **Reflection Workshops,** the project team should organize visualization meetings, both before and after the completion of a sub-version.

2.7.3.4. Feature Driven Development - FDD

Feature Driven Development (FDD) methodology is a lightweight, model-driven software development process focused on frequent, tangible and functional results. These features define the methodology as Agile. This methodology was first reported by Peter Coad in 2000 and was further developed in a book by Palmer and Felsing published in 2002. The FDD focuses on designing and programming, differentiating it from other software development methodologies (Qumer & Henderson - Sellers, 2008). Combining Agile development and model development we emphasize the original model, the division of the project into features, the design of each feature into iterations, where each iteration consists of design and development. Thus, this method is considered suitable for the development of critical systems in relation to other Agile methodologies.

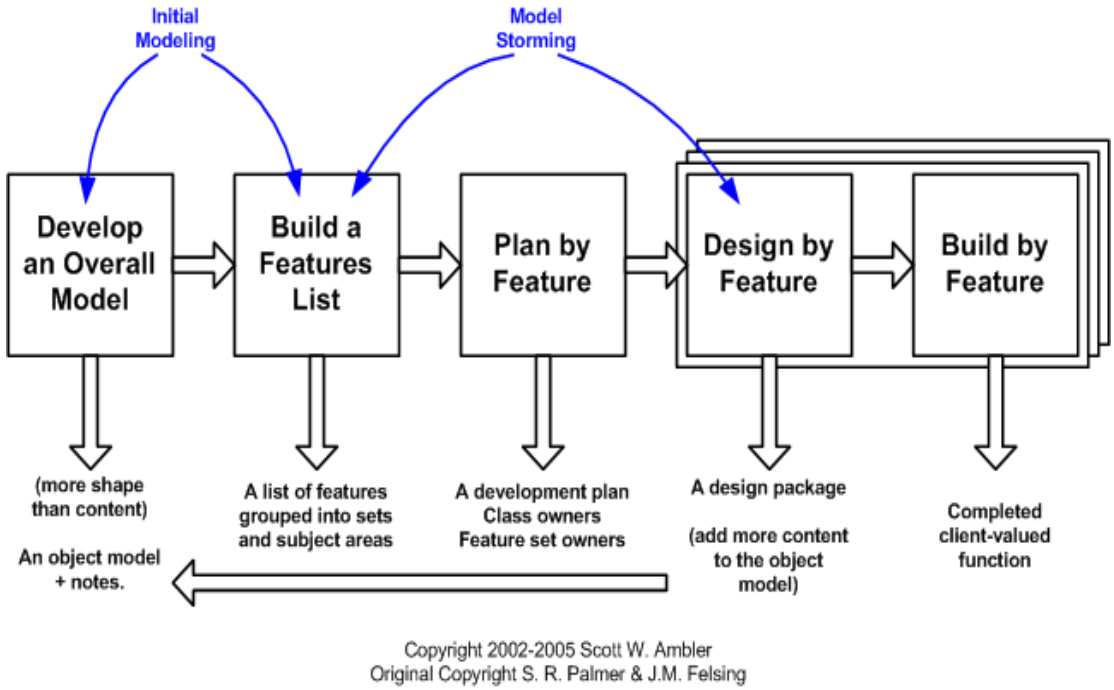


Figure 17 - Feature Driven Development (FDD) – Source: *agilemodeling.com*

The **FDD Development Process** consists of five consecutive processes, during which the design and development of the system is performed as mentioned above. This repeatability supports Agile growth mechanisms with rapid adaptation to last minute changes. Each repetition lasts from one to three weeks. These processes are: Develop an Overall Model, build a Features List, Plan by Features, Design by Feature and Build by Feature.

For the implementation of the software, roles should be defined so that everyone assumes the corresponding responsibility. FDD divides roles into three main categories:

- **Key Roles:** the project manager, the architecture manager, the development manager, the main developer, the class owner and the field experts.
- **Supporting Roles:** the publisher, the programming language specialist, the construction engineer, the tool maker and the system administrator.
- **Additional Roles:** the controllers, the programmers and the authors of technical texts.

Each member of the project team can take on multiple roles, while one role can may be shared by many people.

Finally, the implementation of the construction phases of a software program requires a set of **"good" practices and techniques**, which although not new, their combination makes FDD unique, with the sole purpose of maximizing the benefit of using the method, since each practice alone cannot ensure such a thing. FDD includes the following practices - techniques: Domain

Object Modeling, Developing by Feature, Individual Class Code Ownership, Feature Teams, Inspection, Regular Builds, Configuration Management and Progress Reporting.

2.7.3.5. Dynamic Systems Development Method - DSDM

As mentioned in a previous chapter, "*in the early 1990s, James Martin introduced* ***Rapid Application Development (RAD),*** *which is based on the evolution and improvement of the* ***Evolutionary Delivery Model****, with the main objectives of this methodology being: high speed, high quality and low cost. A characteristic fact is that the active participation of the customer and all involved is required. The control of the system is done repeatedly by both the development team and the users*". Although this method was revolutionary, it was developed in a random fashion and without specific structure. DSDM was born in 1994 in Great Britain with the aim of creating a commonly accepted work framework. The first edition was officially published in 1995, followed by three more editions, slowly being adopted in both the private and public sectors in the UK, crossing the country's borders and becoming one of the most important information systems development methodologies worldwide.

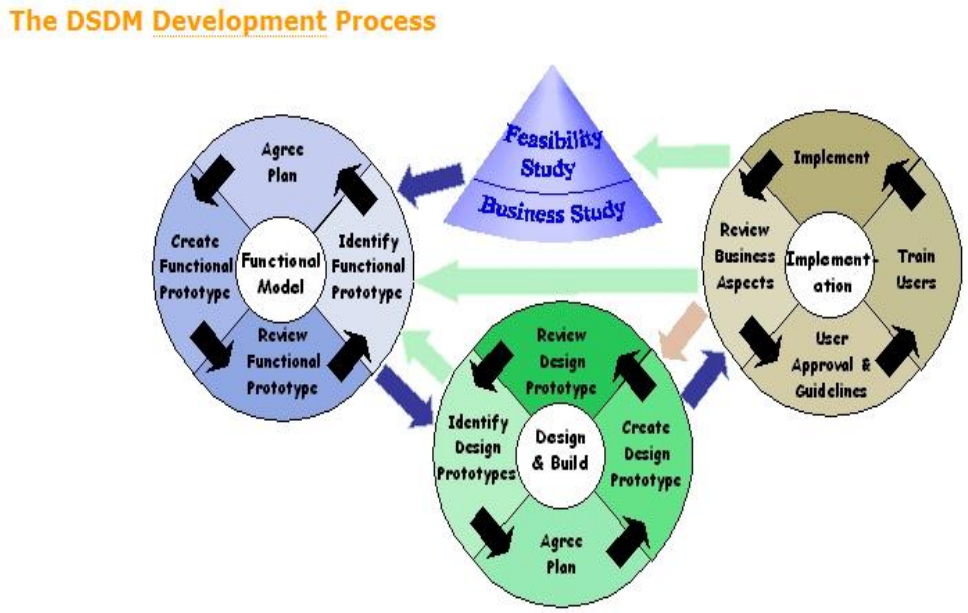


Figure 18 - Dynamic Systems Development Method (DSDM) – Source: *dsdmofagilemethodology.wdfiles.com*

The basic idea of DSDM is that, instead of determining the desired functionality in a product and the resources for implementation, it is preferable to first determine the time and resources, and then determine the desired functionality accordingly.

DSDM's core philosophy is that: The Software development is a collective process, prioritization of project requirements is necessary, the software development can and should be a gradual experience and the high know-how of the technicians involved in project development is a basic condition for the excellent result of the end product.

The **DSDM Development Process** consists of five phases: The Feasibility Study, the Business Study, the Functional Model Iteration, the Design & Build Iteration and the Implementation (Implementation).

For the implementation of the software, roles should be delegated so that everyone assumes the corresponding responsibility. At DSDM we have 15 user and developer roles. The most important of these are the following according to Stapleton (1997):

- Developers and Senior Developers
- Technical Coordinator
- Ambassador User
- Adviser User
- Visionary
- Executive Sponsor

Finally, the implementation of the software development phases requires a set of **basic practices - principles**. The nine principles that govern DSDM are: user involvement, empowerment of the work team, frequent delivery, meeting the modern needs of the business, repetitive and gradual development, the possibility of subversive change, setting the framework before the start of the project, control throughout the project and finally effective and efficient communication (Dyba & Dingsøyr, 2008).

2.7.3.6. Adaptive Software Development - ASD

The ASD method is an object-oriented approach, which facilitates the repetitive and fast delivery of large and complex systems with iterative and step-by-step development with continuous pattern creation. Developed and published in 2000 by James A. Highsmith III, it is the result of an earlier study of repetitive development methods. The project teams here may be different in size and geographical location.

The **ASD Development Process** is designed to provide a framework with adequate guidance to prevent a project from becoming chaotic, but to not stifle creativity. This process is characterized by three phases: **The Speculate stage, the Collaborate stage and the Learn stage**. We use the term "Guess" instead of "Plan" because a "plan" is usually treated as something where uncertainty is considered a weakness and deviations from it indicate failure. The term "Collaboration" emphasizes the importance of teamwork and coordinated effort to support changes during development as a means of developing systems. Finally, the term "learning" emphasizes the need to recognize and respond to errors and changes in requirements during development.

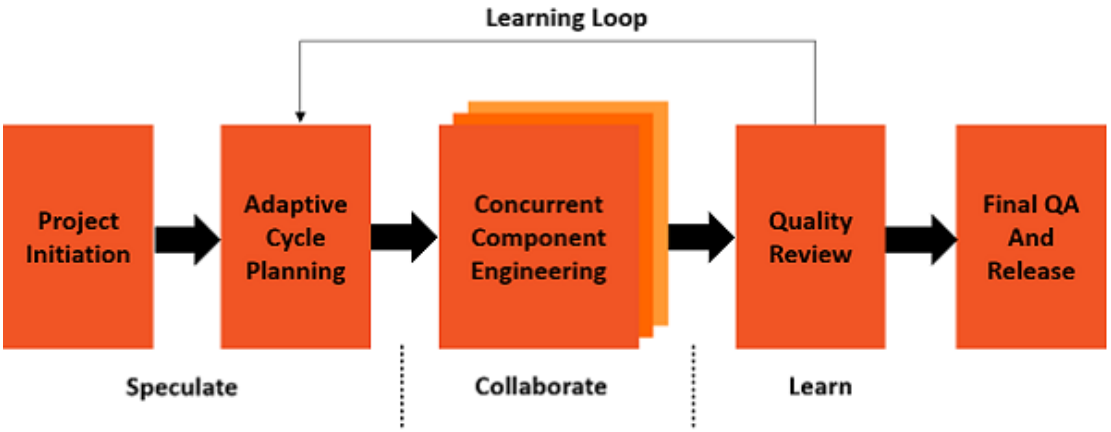


Figure 19 - Adaptive Software Development (ASD) – Source: *tutorialspoint.com*

ASD is largely occupied by the business and management philosophy and especially by the importance of collaborating teams and teamwork. The ASD approach does not describe the group structures and roles. Roles with increased responsibilities roles with accountability very few.

Finally, as regards the implementation of the construction phases of **a software program the basic practices - techniques are very few.** Highsmith explicitly names the following: iterative development, feature based / component-based planning, and customer focus group reviews. In conclusion, ASD addresses issues such as social, cultural, and team skills that align with distributed software development, but does not explicitly state anything about the style of the code that should be used, the technology environment, or the desired business culture (Qumer & Henderson - Sellers, 2008).

2.7.3.7. Lean Software Development

Mary Poppendieck, by engaging in methods **Lean Manufacturing**[3] and **Total Quality Management**[4] concluded that these methods could also be applied to software development beyond industrial manufacturing. The adaptation of these methods to software development is known as Lean Software Development or Lean Programming and was introduced in 2001 by

[3] The Lean Manufacturing method, known as the Toyota Production System, was developed by Taiichi Ohno at the request of Toyota owner Toyoda Sakichi to make industrial car production possible. It has two basic principles - values which have to do with the speed of production flow (rapid product flow) and the built-in quality (build in quality).

[4] At the same time, the Total Quality Management method was taught in Japanese universities by Dr. W. Edwards Deming. Toyota has decided to merge the two methods to develop a better production process.

Mary Poppendieck. This methodology does not include a specific development process, specific roles - responsibilities, but suggests seven specific principles to be followed in software development (Poppendieck, 2003).

The seven principles of the Lean Software Development methodology are:

1. Eliminate Waste
2. Amplify Learning
3. Decide as late as possible (Decide as late as possible)
4. Deliver as fast as possible (Deliver as fast as possible)
5. Strengthen the team (Empower the team)
6. Build integrity in
7. See the whole

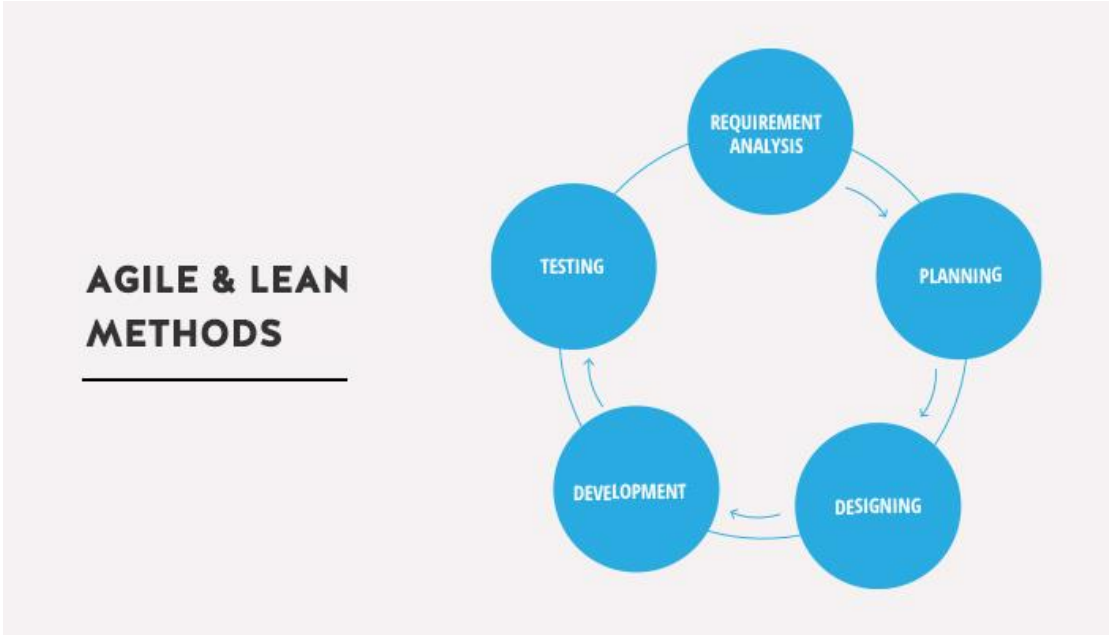


Figure 20 - Lean Software Development – Source: *brainhub.eu*

Finally, according to the above principles, and for each of them, twenty-two tools were proposed that help each authority become an Agile practice. These are:

1.Eliminate Waste

- Tool 1: Identifying Wasting (Seeing Waste)
- Tool 2: Value Stream Mapping

2.Amplify Learning

- Tool 3: Feedback

- Tool 4: Iterations
- Tool 5: Synchronization
- Tool 6: Set - Based Development

3.Decide as late as possible

- Tool 7: Options Thinking
- Tool 8: The Last Responsible Moment
- Tool 9: Making Decisions

4.Deliver as fast as possible

- Tool 10: Pull Systems
- Tool 11: Queuing Theory
- Tool 12: The Cost of Delay

5.Strengthen the team (Empower the team)

- Tool 13: Self-Determination
- Tool 14: Motivation
- Tool 15: Leadership
- Tool 16: Expertise

6.Build integrity in

- Tool 17: Perceived Integrity
- Tool 18: Conceptual Integrity
- Tool 19: Refactoring
- Tool 20: Testing

7.See the whole

- Tool 21: Measurements
- Tool 22: Contracts

2.8. Conclusions

Despite the widespread use of Agile software development in recent years, organizations are finding it difficult to adopt Agile practices. This is because, upon attempting to adopt them, they are faced with various challenges, resulting in the majority of organizations abandoning them (Yu & Petter, 2014).

From the moment the need to develop software development methodologies arose, many different processes and process models emerged to reach today's Agile methods for producing

quality software. Initially, an organization should be deeply knowledgeable of all traditional methods and their strengths - weaknesses, so that they may be discarded and switched with Agile software development methodologies, in order to choose the right one, which should ultimately meet the needs of the organization.

In conclusion, for an organization to be certain of which software development model should be followed, it –the organization-needs to be aware from the beginning of the goal it seeks to achieve. Certainly, since we are traversing the era of the 4th Industrial Revolution with frequently changing requirements, there is a need for specialization and the delivery of the product should be very fast, therefore adoption of the methods of agile development is necessary.

This page is intentionally white.

Chapter 3: Methodology

In this chapter we attempt to highlight and develop the degree of knowledge of "agile" methodologies, both from the viewpoint of software companies in the Greek market, and from the viewpoint of the executives of these companies involved in the development of software projects. Specifically, we aim to describe corresponding surveys that explore similar questions, namely the degree of success, the success factors and the factors that hinder the completion of a software development project. Next, the research questions related to our research will be described and analyzed. Finally, in the next two sections of the chapter, the detailed description of the research will be given (procedure, characteristics, techniques, tools, etc.) followed by the detailed description of the research elements (population, sample, selection methods, etc.), so that we may reach the last chapter of the research, where the analysis of the results, the final conclusions and the proposals for the future are given.

3.1. The application of Agile Methodologies

Adoption of the Agile software development, although popular in mostly the last decade, is still found to encounter great difficulties. The transition from the traditional software development methods to the modern Agile ones requires in-depth knowledge of these methods, their strengths and weaknesses, and the nature and structure of the organization to be developed in order to choose the correct method.

Indicatively, some criteria will be mentioned, comparing the traditional methods to those of the Agile ones as it emerged from the literature review (Georgiadou A., 2014, Investigation of the use of Agile methods of project management). These are the following:

1. **Fundamental Assumptions:** Systems are fully identifiable, predictable, and can be developed with meticulous and extensive vs scheduling high quality custom software is developed by small teams, using the principles of continuous design improvement and frequent testing and based on rapid feedback and change.
2. **Objective:** Predictability and optimization for High vs Assurance Exploration or adaptation for fast value return.
3. **Requirements:** Known from the beginning, largely stable: Clearly defined and documented vs Urgent with frequent alternations, unknown - discovered during the project.
4. **Management Style:** Authoritarian with Commands and Control vs Decentralized - Leadership and Cooperation.
5. **Knowledge Management:** Explicit vs Tacit.

6. **Desired Organization Structure:** Bureaucratic with high standardization VS Agile and Participatory encouraging collaborative social action.
7. **Change:** Tendency to change vs Accept change.
8. **Focus:** Processes vs Humans.
9. **Project Size:** Large vs Small.
10. **Group Size:** Large vs Small.
11. **Team Organization:** Structured vs Groups Self-governing teams
12. **Team Members:** Distributed teams of experts - Design-oriented, with sufficient skills and access to external knowledge vs Agile, trained, collaborative - Senior technical staff located in the same area.
13. **Role Assignment:** Individual where vs specialization is favored Self-managed groups where role rotation is favored.
14. **Communication:** Official vs Unofficial.
15. **Clients:** Low participation - passive role vs the client is considered a member of the team: Active role.
16. **Project Design:** Guided by vs tasks and procedures Guided by product features.
17. **Original Design:** Full VS Minimum.
18. **Software Development Process:** A Universal Approach and Solution to Provide Predictability and High vs Guarantee Agile approach, tailored with an understanding of the overall contextual needs to deliver faster growth.
19. **Development Model:** "Waterfall" or "Spiral" type model, or variant of these vs Evolutionary delivery model.
20. **Project Cycles:** Limited vs Numerous.
21. **Technology Used:** Unlimited VS Favors object-oriented technology.
22. **Documentation:** Heavy or Extensive explicit recording vs Light (replaced by face-to-face communication).
23. **Redesign:** Expensive vs Low cost.
24. **Measuring Success:** Complying with vs Design Business Value.
25. **Return on Investment (ROI):** At the end of the project vs in the early stages of the project.
26. **Risks:** Understood risks with minor impact vs Unknown risks with significant impact

In conclusion, the traditional methods, to which the waterfall model and the spiral model are mainly applied, have their own advantages corresponding to certain requirements similarly to the Agile methods, while at the same time they have great weaknesses and disadvantages. That is why the choice should be made with extreme caution. Fourteen case studies of the application of the adoption of "agile" methodologies are here being analyzed. Our research questions which

will follow the development of the research, will be analyzed based on the results of these case studies,

Specifically, the following research articles have been identified as relevant and are presented below:

15. Nuottila, J., Aaltonen, K. & Kujala, J. (2016), Challenges of adopting agile methods in a public organization, International Journal of Information Systems and Project Management, Vol. 4, No. 3, 65-85,
16. Kouzari, E., Gerogiannis, V., Stamelos, I. & Kakarontzas G. (2015), Critical success factors and barriers for lightweight: software process improvement in agile development: A literature review, France, 10th International Conference on Software Engineering and Applications (ICSOFT-EA-2015), pages 151-159,
17. Misra, S. C., Kumar, V. & Kumar U. (2009), Identifying some important success factors in adopting agile software development practices, India, Elsevier, Volume 82, Issue 11, pp: 1869 – 1890,
18. Masood, Z., Hoda, R. & Blincoe, K. (2018), Adapting Agile Practices in University Contexts, New Zealand, Elsevier, Volume 144, pp: 501 – 510,
19. Sheffield, J., Lemétayer, J. (2012), Factors associated with the software development agility of successful projects, New Zealand, Elsevier, Volume 31, Issue 3, April 2013, Pages 459-472,
20. Ahimbisibwe, A., Cavana, R., Daellenbach, U. (2013), A contingency fit model of critical success factors for software development projects. A comparison of agile and traditional plan-based methodologies, New Zealand, Journal of Enterprise Information Management, Vol. 28 No. 1, 2015, pp. 7-33),
21. Pikkarainen, M., Salo, O., Kuusela, R., Abrahamsson, P. (2011), Strengths and barriers behind the successful agile deployment – insights from the three software intensive companies in Finland, Springer Science – Business Media, Vol 17, pages: 675 – 702,
22. Misra, S. C., Kumar, V., Kumar, U. (2009), Identifying some critical changes required in adopting agile practices in traditional software development projects, Emerald Group Publishing Limited, International Journal of Quality & Reliability Management, Vol. 27 No. 4, 2010, pp. 451 – 474,
23. Gandomani, T. J., Zulzalil, H., Ghani, A. A. A., Sultan, A. B. M. & Sharif, K. Y. (2014), How Human Aspects Impress Agile Software Development

Transition and Adoption, International Journal of Software Engineering and Its Applications, Vol.8, No.1 (2014), pp: 129-148,

24. Dhir, S., Kumar, D. & Singh, V. B. (2019), Success and Failure Factors that Impact on Project Implementation Using Agile Software Development Methodology, Springer Nature Singapore Pte Ltd. 2019 M. N. Hoda et al. (eds.), Software Engineering, Advances in Intelligent Systems and Computing 731, pages: 647 – 654,
25. Tam, C., Moura, E. J. D.C., Oliveira, T. & Varajão, J. (2020), The factors influencing the success of on-going agile software development projects, Elsevier, International Journal of Project Management 38 (2020), pages: 165 – 176,
26. Hummel, M. & Epp, A. (2015), Success Factors of Agile Information Systems Development: A Qualitative Study, 48th Hawaii International Conference on System Sciences,
27. Aldahmash, A., Gravell, A. M. & Howard, Y. (2017), A Review on the Critical Success Factors of Agile Software Development, Springer International Publishing AG 2017J, Stolfa et al. (Eds.): Euro SPI 2017, CCIS 748, pp. 504–512, 2017,
28. Asnawi, A. L., Gravell, A. M. & Wills, G. B. (2010), An Empirical Study: Understanding Factors and Barriers for Implementing Agile Methods in Malaysia, 5th International Doctoral Symposium on Empirical Software Engineering - IDoESE

3.1.1. Challenges of adopting Agile methods in a public organisation

The **aim** of this research paper was to identify and categorize the challenges that may hinder the efficient adoption of "agile" methodologies by the Finnish Ministry of Transport and Communications. A case study was implemented, through semi-structured interviews.

Of the numerous **findings**, the observed challenges were **categorized** into 7 areas, which are the following:

1. Documentation

2. Training, experience and commitment

3. Communication and participation of stakeholders

4. Agile regulator roles

5. Position of agile teams

6. Legislation

7. SW architecture complexity and systems integration

3.1.2. Critical success factors and barriers for lightweight software process improvement in Agile development: A literature review

The **aim** of this research paper was to identify the success factors that lead to maximizing performance (ROI) in small and medium enterprises, as well as the obstacles, through literature review, by answering the following research questions:

- What are the special features of small and medium-sized software companies when they follow agile methodologies?
- What are the main critical success factors and barriers involved in SPI?
- How do critical success factors and barriers affect return on investment (ROI) in companies applying light SPIs?

Of the numerous **findings**, the success factors and obstacles were categorized as follows:

1. Commitment

2. Involvement of staff

3. Education

4. Resources

5. Process action groups

6. Staff experience

7. Guidance

8. Reviews - Feedback

9. Application methodology

10. Monitoring

11. Communication

12. Return on investment

13. Awareness of SPI

14. Additional factors that are not often mentioned in the literature.

3.1.3. Identifying some important success factors in adopting Agile software development practices

The **aim** of this research paper was to identify the success factors from the perspective of practicing "agile" professionals, so as to lead their projects to successful completion. A hypothetical framework of 14 hypothetical factors was defined, and they were asked to answer the following research question: "***What are the factors that will influence the success of projects from the Agile professionals' perspective who wish to adopt Agile practices?"***

From the analysis of the quantitative data obtained, it was observed that **nine factors emerge as having statistically significant effects in relation to success, while five of them are constraints on the completion of a project**. The factors that play an important role for success are: Customer-centric issues, Decision time, Corporate culture, Control, Personal characteristics, Social culture, Training - learning, Group distribution, Team size, Design, Technical ability and Communication - negotiation.

3.1.4. Adapting Agile practices in university contexts

In this research paper we **observe** the application by students and professors of "agile" methodologies at a University (Auckland). The **aim** of the study is to identify the constraints that students face by applying "agile" practices in a course, answering the following research questions:

- What are the most common constraints faced by students while practicing agile methods in a university course?
- What agile practices do students practice and how have they been adapted to fit the university context?
- Which adapted practices are considered beneficial and effective in terms of results and which are not?

The **findings** describe limitations, adaptations, effectiveness, and the relationship between constraints and adaptations by applying agile practices, within a university context.

Restrictions identified by students relate to: Scheduling Restrictions, Team Communication Issues, Customer Relationship Issues, Lack of Commitment, Personal Commitments, and Technical Constraints.

The adjustments that must be implemented immediately concern:

- Scrum practices,
- Sprint Meetings,
- Scrum roles,
- As XP practices we should have pair programming

Finally, the results show that the adjustments had both positive and negative effects, making the effectiveness perceptible.

3.1.5. Factors associated with the software development agility of successful projects

The **aim** of this research paper is, through use of the members of the "agile" communities such as PRINCE2, PMI, etc., to gather empirical evidence through interviews on the question: **which factors in the project and in the project, environment is indicative of "agility" in successful software development projects**? Current **research contributes** to the literature on the success of the methodology for a model of "best practice" methodological success.

The **indicative factors of software development agility** in the project environment are: the organizational culture and the empowerment of the project team.

The current study identifies at least **six specific implications**. **First**, consideration of the appropriate level of "agility" in software development must be broadened. **Second,** in order for the project to be successful, professionals must ensure their commitment to a specific development methodology. **Third,** the current study shows that top project management and the client often fail to agree on what form of "agility" would be appropriate for developing their own software. **Fourth,** the findings show that the project team, project management, client and top management must take a more "agile" and adaptable approach. **Fifth,** "agility" practitioners must think about how to tailor the process to better suit their needs. **Sixth,** the factors that are important for the success of the project in the long run may differ from those that are indicative of "agility".

This study suffered from some **constraints. First,** due to the complexity of the topics and the lack of appropriate quantitative instruments in the methodological success of the literature, the findings of this research were not definitive. Additional research is recommended. **Second,** the sample size remains relatively low. A larger sample size could provide more accurate statistics. **Third**, most of the answers came from low-power countries. This limited the investigation of the effect of the power distance. **Fourth,** some of the data collected for demographic purposes (organization size, industry, country, respondent experience and role) could have been classified as influencing the project and its environment and are included in the formal analysis.

Finally, this study also has **limitations. First,** due to reduced resources, the scope of the research was limited. **Second,** the constructions used are not based on theoretically correct perceptions and tested instruments. **Third,** while interviews have allowed research to focus on selected topics, many interesting topics were not included in the questions. **Fourth,** the focus of the research was on community members of software development practice.

3.1.6. A contingency fit model of critical success factors for software development projects. A comparison of Agile and traditional plan-based methodologies

The **purpose** of this research paper is to identify and categorize critical success factors (CRCs) and to develop a customized emergency model that contrasts the perspectives of traditional methodologies with "agile" methodologies. In order to **achieve** the purpose, a systematic review of the literature was performed (148 articles), where the 37 CCIs identified were categorized into 3 categories. These are: the organizational factors, the team factors and the customer factors.

The **findings** were identified within the three main categories of CCIs and individual factors are classified based on how often they had been reported by previous studies. Differences in these rankings, as well as mixed empirical support, suggest that previous research may not have been sufficiently considered when the CCIs were specific, and which would affect the success of the project.

Finally, the **constraints of the research** relate to the testing of the emergency adaptation model developed using empirical data. There is no broad consensus among researchers and professionals on the categorization of CCIs for software development programs. However, through an extensive review and analysis of the CIP literature for software development projects, this research provides greater clarity about the categories of CCIs and how their direct, indirect and moderate effects on project success can be modeled.

3.1.7. Strengths and barriers behind the successful Agile deployment – insights from the three software intensive companies in Finland

The **aim** of this research is to analyze the **advantages and disadvantages** of selected pilot projects by three software organizations, which attempted to develop these projects through "agile" methodologies. Another **goal** is to complete this analysis with recommendations to support software companies planning "Agile" development. In order to achieve these goals, three in-depth case studies were conducted in which the development of an "agile" methodology in each of the three companies was investigated.

The **results of the Research** showed that the strengths of software development with "agile" methodologies are mainly related to **communication and collaboration** within the team, and between the team and stakeholders during the execution of the development. Other identified **benefits** of using "agile" software development include **improvements in unit testing, improvements in claims management and customer demand management, design and appraisal improvement activities, self-organization, improvement methods and learning aspects.** The main **constraint** of this research is that the data collected are based mainly on the **subjective views** of the respondents from the companies studied.

3.1.8. Identifying some critical changes required in adopting Agile Practices in traditional software development projects

The **primary motivation** behind this research is to try to identify some of the changes that must be implemented in organizations which are in the process of transitioning to the "agility" application. Based on this, it is possible to assist project managers involved in such transitional activities. That is why the present **empirical study** was conducted based on the research.

The **changes** required to adopt ASD practices from traditional projects to "agile" ones should have the following characteristics:

- **Fundamental Features:** Whether a project needs to be fully or partially transformed into "agile", we should focus first on the key areas and then gradually turn our attention to the other areas. To be able to do this, it would be useful to identify the key changes needed to adopt agile philosophies in a project.
- **Big changes:** in organizational culture, management style, knowledge management strategies and development processes.

Finally, studies of this nature are often accompanied by **different constraints** of varying severity. It is therefore important to look at the research results presented in this paper within the limits of these restrictions. These limitations presented should be considered together in order to inspire future research on the same subject. The **constraints** are as follows: (1) In this project, we have tried to find evidence based on research, (2) The variables used in the study are complex in nature, (3) Due to the limited availability of sufficient relevant literature, specifically a problem we face in this article is that the pieces of literature used for the concept of each construction are not always scientific, (4) The methodology of asking a few people for advice, without a mediation focus group methodology has its limitations, even if the questionnaire has been tested for validity, usefulness and readability before being sent for field data collection, (5) In this work, we have limited ourselves to analyzing aggregate data obtained from different respondents on different continents in around the world, (6) We targeted individuals, but asked them for group information (including additional linked teams) (7) This study is limited to the assumption that data obtained in different job functions is equally important, (8) In the research methodology, we obtain the various variables into each area and use their average (median).

3.1.9. How human aspects impress Agile software development transition and adoption

This research paper focused on the human aspects of the process of transition to "agility", by conducting a study (Grounded Theory) with the participation of 32 Agile experts. The **main contribution** of this research paper is the identification and classification of the different

human aspects of the process of transition to "agility". The analysis of the collected data showed that the human aspects of the process of transition to "agility" can be classified into several categories. While some of them are obstacles to change, others act as accelerators of change.

Changing the method of development from traditional to agile methods **requires extensive changes in people's mentality and behaviors**. This means that people play a critical role in the Agile transformation / transition process (ATP) and can act as both drivers and barriers to Agile transition and adoption. This study was conducted based on **Grounded Theory** as a qualitative research method.

Data analysis was performed by coding the data and started with the collection of some data. During data analysis, the researchers used theoretical notes to add some more relevant data. The memoirs helped researchers develop ideas about concepts and categories and discover relationships between them. **Creating Theory**, the last step in data analysis, was creating a theory called "theoretical coding."

The **results** in this study showed that people and their mentality, as well as their behaviors, strongly influence all activities related to transformation. The relevant categories and subcategories have to do with:

1. **Barriers to change**: The human aspects became significant barriers to the change process in software companies. The obstacles arose due to the following: Resistance to change, Cultural issues, Lack of cooperation, Wrong mentality, Lack of knowledge.
2. **Accelerate change:** Despite the aforementioned negative effects of humans on the transformation process, there has been considerable evidence of the positive effects of human factors on ATP. These are: People Buy, Market Management, Champions and Supporters.
3. **People's perceptions of the change process:** This study found that people's reactions to change are mainly related to their perceptions of it. These are: Enthusiasm for change, Anxiety for change, Indifference for change, Feeling the real need for change and Unrealistic expectation for flexibility.
4. **Motivational factors:** The study found that people, especially those who did not have enough in-depth knowledge of Agile methods and values, should be motivated and encouraged to apply Agile. Motivating makes people positive about change.

Finally, the constraints relate to finding the data-based study, because all the codes, concepts, categories, and properties were collected directly from real environments. These authors do not claim that their findings are universal because their access to appropriate resources was limited to those participants who volunteered for their research. However, they claim that their findings describe and characterize the area under study.

3.1.10. Success and failure factors that impact on project implementation using Agile software development methodology

In "agile" software development, different factors exist behind the success and failure of projects. This research paper **represents** the factors of success, failure and mitigation in "agile" growth. **A case study is presented** pertaining to all these factors after the completion of the small projects. The final results are observed based on the analysis of the efficiency, accuracy, time management, risk analysis and quality of the project products. The final results are determined using a variety of approaches.

Specifically, in this research paper a case study was carried out based on the development of five projects. The projects were developed by the students in five different groups. It was observed that the projects were developed using different developmental approaches, such as waterfall, spiral and agile approach, and it was determined that there were differences in non-functional activities with different methods. The analysis was based on efficiency, accuracy, time management, risk analysis and project product quality. Finally, the **result** was that in Scrum methodology the results were better than the waterfall and spiral model, where efficiency, product quality and time management are high.

3.1.11. The factors influencing the success of on-going Agile software development projects

Since "agile" software development relies on human capital to be successful, research has **focused** on uncovering the human factors that contribute to the success of these types of projects. After an extensive review of the literature, a model of factors was found to have been shown to influence the success of "agile" software development projects, and thus their validity was re-examined within the country of Portugal. The conceptual model provided a concise characterization of the human factor dimension. Future researchers are invited to validate and leverage the research work, making the model more detailed and reliable.

A total of 216 "agility" professionals were interviewed from a variety of business areas. The **results obtained** indicated that the ability of the team and the involvement of customers can largely explain the difference in the success of the software development project. However, we have not been able to find evidence to suggest that personal characteristics, education and learning or social culture are important factors in this context. These **findings were further tested and cross-referenced** for accuracy and supported by a quality approach focus group.

This research paper **offers** valuable advice to professionals who are currently practicing "agility" or who will be participating in an "agile" software development program. According to the findings, executives are encouraged to select a high-capacity team and promote customer engagement and collaboration, as these factors are more likely to lead a software development

project to success. Finally, this research offers a mixed method approach that provides more positive and solid conclusions than a single method. The application of a combination of quantitative and qualitative approach allowed us to delve into the analysis and the findings.

3.1.12. Success factors of Agile information systems development: A qualitative study

In this research paper, a qualitative study based on interviewing 12 participants was conducted in order to shed further light on the success factors of the "agile" Information Systems Development (ISD), because quantitative studies reported conflicting results. Initially, four success factors were identified from previous literature that form the conceptual basis of their research. These factors are **the team, the customer, the organization and the communication**. They extended the previous findings by opening the context of each of these factors. This was achieved by identifying the individual factors that enable the "agile" success of ISDs. Autonomy **was considered** to be the most important feature of the team, while "agile" values and administrative support are vital for the organizational level. As regards the role of the clients, the participation of the clients was judged as the most important, while in regards with communication, the meeting practices were the most frequently mentioned sub-factor. In contrast to the other categories, it was noted that fewer factors for the organizational level were reported by fewer participants. Therefore, the following research question was asked: ***How is the success of agile ISD affected by team, customer, organization and communication?***

The **results were divided** into the following sections showing the results for each of the four previously identified success factors.

- In the **Group** the most frequently mentioned characteristics are: autonomy, personal discipline, motivation, continuous learning and cooperation.
- In **Organization** Agile SD needs a different organization from traditional methods based on: Management Support, Agile Values and organizational levels.
- In the **Customer** the characteristics should be the following: Customer Focus and Proximity, Regular and timely feedback and Mutual understanding and trust with the development team.
- In **Communication** their respondents confirm that communication is critical to any agile ASD project. The characteristics of Communication have to do with: Meeting Minutes (e.g., daily Scrum) and Office space located in a common location.

In conclusion, the present research provided qualitative information on the basic principles of "agile" information systems under development (ASSD) and inspired deeper qualitative and quantitative research in the future.

3.1.13. A review on the critical success factors of Agile software development

The decade between 2006 and 2016 produced a lot of research that **focused** on identifying the factors that influence the success of "agile" software development. This research paper **aims** to provide an overview of the critical success factors of the agile software development research of the above decade (2006-2016). Eight previous studies have been selected because they used empirical methodologies to validate Critical Success Factors (CRCs). The selected studies identified many success factors for "agile" software development, eight of which have been selected in this paper because they were identified in more than one study. These are delivery strategy, team capability and training, agile development techniques, customer engagement, project management process, organizational culture, communication and top management support. The selected factors are classified into categories that include Technique, Organization, Procedures, and Individuals.

Future research can examine the proposed classification and explore how these success factors relate to each other. Future work may also study the importance of these CCIs and the weight of each factor. There is ongoing research into data collection by "agile professionals around the world to explore the importance of the eight identified success factors. Responses were received from 131 professionals who practice "agility", while the relationship between the number of completed repetitions in the project developed with "agility" and the importance of success factors was investigated. Finally, the role of individual experience and organizational experience was explored by exercising "agility" for the perception of recognized success factors. Using factor analysis, the relationships between the eight identified success factors were investigated.

3.1.14. An empirical study: Understanding factors and barriers for implementing Agile methods in Malaysia

This research paper focuses on empirical work for a doctoral research. The **main purpose** of the research is to identify the factors that can lead to the adoption or rejection of "agile" methods in Malaysia. The **contribution of the study is** to reduce the gap between the adoption factors in the country and the region where similar studies have not been implemented. As part of this research, the initial project involved identifying current levels of use and perception of methodology in the country. A pilot study using a questionnaire was conducted in two languages, English and Malay. The initial results showed that more than 50% of the 79 respondents were not aware of the "agile" methodologies. Despite their ignorance, most of the respondents believe in "agile" methodologies. Awareness of "agile" methods is also found in the important role played by the language used by respondents, who equate equally critical technical and social factors in order to assess the benefits of Agile methods. This finding

provides a promising environment for the application of "agile" methods in the country. This study is used to provide guidelines on the factors and obstacles that arise with the adoption of "agile" methodologies in the country, where there is little knowledge and studies on methods. The results are also used to fill the research gap of the vague evidence found in previous studies.

3.1.15. Summary table of success factors and constraints

Summarizing all of the above papers that mainly concern the factors of success and failure or limitations while applying the Agile Methodologies to Greek software companies, we arrive at the following summary table.

Table 1 - Success and failure factors or constraints by applying Agile Methodologies

Paper	Success – Adoption Factors	Failure – Constraints Factors
1. Challenges of adopting Agile methods in a public organisation	• Increased flexibility • Improvements in customer satisfaction, quality, productivity and cost	• Documentation • Training, experience and commitment • Communication and participation of stakeholders • Flexible regulator roles • Position of flexible teams • Legislation • SW architecture complexity and systems integration
2. Critical success factors and barriers for lightweight software process improvement in Agile Development: A literature review	• Commitment • Staff participation • Education • Resources • Process action groups • Staff experience • Guidance • Reviews - Feedback • Application methodology • Monitoring • Contact • Return on investment • Awareness of SPI	• Commitment • Staff participation • Education • Resources • Process action groups • Staff experience • Guidance • Reviews - Feedback • Application methodology • Monitoring • Contact • Return on Investment • Awareness of SPI

3. Identifying some important success factors in adopting Agile Software Development Practices	• **The organizational factors:** Customer-centric issues, Decision time, Group distribution, Team size, Corporate culture and Planning - control • **Human factors:** Responsibility, Personal characteristics, Communication and negotiation, Social culture and Education - learning	
4. Adapting Agile practices in university contexts		• Scheduling restrictions, • Group communication issues, • Customer issues, • Lack of commitment, • Personal commitments, • Technical restrictions
5. Factors associated with the software development agility of successful projects	• Organisational Culture • The strengthening of the project team	• **First,** due to the complexity of the topics and the lack of appropriate quantitative instruments in the methodological success of the literature, the findings of this research are less than definitive. Additional research is recommended. • **Second,** the sample size remains relatively low. A larger sample size could provide more accurate statistics. • **Third,** most of the answers came from low-power countries. This limited the investigation of the effect of the power distance. • **Fourth,** some of the data collected for demographic purposes (organization size, industry, country, respondent experience and role) could have been classified as factors in the project and its environment and included in the formal analysis.

6. A contingency fit model of critical success factors for software development projects. A comparison of Agile and traditional plan-based methodologies	• The client's organizational factors and the success of the software development project, • Team factors and software development project success, • Client Factors and software development project success, • Project Factors (mitigation variables)	
7. Strengths and barriers behind the successful Agile deployment – insights from the three software intensive companies in Finland	• Make sure management is committed and provides ongoing support for flexible development • Make sure that both the team, the management and all stakeholders have a clear vision, understanding and awareness of flexible methods • Give groups the freedom they need to adapt flexible methods to better suit their specific needs • Constantly adapt the flexible process model to an organizational level	• Ensure that Management is committed and provides ongoing support for flexible development, • Ensure that Management has a clear vision, understanding and awareness of flexible methods, • Give groups the freedom they need to adapt flexible methods to better suit their specific needs, • Continuously Adjust the Process Model Based on Flexibility and Organizational Level

8. Identifying some critical changes required in adopting Agile Practices in traditional software development projects	• Changes in organizational culture, • Changes in management style, • Changes in knowledge management strategies, • Changes in development processes.	• In this project, we tried to find evidence based on research, • The variables used in the study are all complex in nature, • Due to the limited availability of sufficiently relevant specific literature on the specific problem we face in this article, the pieces of literature used for the concept of each construction are not always scientific, • The methodology of asking a few people to give their opinion without a mediation focus group methodology has its limitations, even if the questionnaire was tested for validity, usefulness and readability before being sent for field data collection, • In this project, we have limited ourselves to the analysis of aggregate data obtained from various respondents from different continents around the world, • We targeted individuals, but asked them for group information (including other engagement groups) and corporate and social issues, • This study is limited to the assumption that data obtained in different job modes is equally important, • In our research methodology, we take the various variables in each area and simply take their average (median)
9. How human aspects impress Agile software development transition and adoption	**A. Human Factors:** • People buy, • Market Management, • Champions and Supporters.	• Resistance to change, • Cultural issues, • Lack of cooperation, • Wrong mentality, • Lack of knowledge

	B. People's reactions to change: • Enthusiasm for change, • Concern for change, • The Indifferent to change, • The feeling of the real need for change and • The unrealistic expectation of flexibility. **C. Motivating makes people positive about change**	
10. Success and failure factors that impact on project implementation using Agile software development methodology	**A. Organization:** • Collaborative culture • Facilities for a flexible working environment **B. Group:** • Motivation between team members • Teamwork reduces workload and increases knowledge • Strong relationship between team members, stakeholders and the client	**A. Organization:** • Organizational culture becomes socially narrow • Communication problem in a distributed environment • Inaccurate estimates in the early stages **B. Group:** • Lack of knowledge and experience • Lack of management capacity • Understanding the issue among team members • A bad relationship with customers

	C. Procedure: • The team follows the flexible configuration, project • Management process • Good communication with daily meetings • Understanding the level of the project and its new topics **D. Technique:** • Following a simple design • Continuous fixation of the software • Proper completion control • The team is dynamic in nature and adapts new techniques to demand **E. Documentation:** • Exact amount of documentation • Less documentation prevents extra effort	**C. Procedure:** • Absence of customer • Unclear programming and scope • The roles of the project are clearly defined • Prioritization of requirements becomes the main issue when developing the process in a small group **D. Technique:** • Lack of resources • Lack of use of new techniques and tools • Wrong agile approach practices **E. Documentation:** • Less documentation can be a problem for new members
11. The factors influencing the success of on-going Agile software development projects	• Personal characteristics, • Education and learning, • Social culture, • Team ability • Customer participation	• Complexity of variables

12. Success factors of Agile information systems development: A qualitative study	• In the **Team** the most frequently mentioned characteristics are: **autonomy, personal discipline, motivation, continuous learning and cooperation.** • In **Organization,** Agile ISD needs a different organization than traditional methods based on: **Management Support, Flexible Values and Organizational Levels.** • In the **Customer** the characteristics must be the following: **Customer Focus and proximity, Regular and timely feedback and Mutual understanding and trust with the development team.** • In **Communication** the characteristics of communication have to do as follows: with **the Meeting Practices (e.g., daily Scrum) and the Office Space located in a common location.**	
13. A review on the critical success factors of Agile software development	• The delivery strategy, • The ability and training of the team, • Agile development techniques, • Customer participation, • The project management process, • Organizational culture, • Communication and • Top management support	• Software development conflicts, • Business process disputes, and • Conflicts between people

14. An empirical study: Understanding factors and barriers for implementing Agile methods in Malaysia	• People • Organizational structure • Environmental factors	• The quality, • Tradition, • The budget, • People, • The organizing factor, and • The requirements

Success factors are mainly related to Commitment, Documentation, Training - Experience, Communication - Stakeholder Participation, Legislation, Resources, Corporate Culture, Social Culture, Teamwork, Vision, Style Management - Strategy - Development Processes and Discipline.

Failure factors and constraints are mainly related to Documentation, Training - Experience - Commitment, Communication - Stakeholder Participation, Legislation, Planning Complexity, Resources, Guidance, Feedback, Methodology, Complexity of Variables, Lack of research in Greece and Resistance to change.

In conclusion, the above factors will form the basis of our research, acting as the measure of comparison of the answers given by the implementers of Agile Methodologies from the Hellenic Software Company. Thus, based on the above factors we aim to be able to determine whether the application of Agile methodologies is feasible within the Greek industry or not.

3.2. The research questions

With the beginning of the new decade, Greek society was confronted with a new reality. That of the digital transition. Thousands of products and services are now subject to these technologies and to what they have the potential of offering. But what really happened to drive us to this moment? How was this huge bureaucratic state transformed into a new digital one? How did the creators deal with this transition? Were they properly prepared or not? What is certain is that in this new era of the 4th Industrial Revolution we are currently traversing at a fast pace, the existence of products and services based on software is clearly visible.

3.2.1. What are the most common constraints faced by Greek software companies while practicing Agile?

According to previously presented articles and the summary table, we have reached the following conclusions. That the most common success factors and constraints collectively, are intimately related to human, organizational and environmental factors.

As regards the human factors we observe the lack of education - continuous learning, experience, commitment, communication, stakeholder participation and resistance to change. In regards with **organizational factors,** we observe the lack of documentation, the complexity of design, the complexity of variables, and the lack of methodology, feedback and guidance. As regards the **factors that surround the project**, we are confronted by the lack of legislation, allocation of resources, as well as the lack of research in Greece.

In conclusion, according to the literature review and previous research, we contrast our own factors in order to identify possible correspondence to our own research, and whether this is fully applicable to Greek software companies.

3.2.2. What "Agile" practices do Greek software engineers choose to follow, and how have these practices been adapted to meet the company's business goals?

Considering the Agile methodologies as developed above, one will find a plethora of new practices. The result is the successful completion of software projects.

The best-known Agile methodologies are those of the Scrum and XP. Other well-known and less widely used are the Crystal Family, the Feature Driven Development (FDD), the Dynamic Systems Development Method (DSDM), the Adaptive Software Development (ASD), the Lean Development, and the Unified Process.

In conclusion, in our research we will consider the above methods as known, and we will inquire of the participants which of the above methodologies they use for the construction of software. We additionally inquire whether these practices had to be adapted in order to meet the company’s goals or not.

3.2.3. Which custom practices are considered beneficial and effective in terms of results, and which are not?

Having taken into account the aforementioned agile practices, as developed in this dissertation through literature and further expanded upon using the above research question, as currently applied by Greek manufacturers, we aim to identify the practices that were beneficial and effective for their company, while their respective results were visible, fulfilling their business objectives.

In conclusion, by the end we will have conducted new research, which focuses on Greek software and targeting companies, directly asking questions to company executives. The results of the current undertaking aim to act as an incentive for further study, awareness of Agile Management and the relevant milestones occurring in Greece of the new decade during the design and construction of software.

3.2.4. Conclusion of research questions

The objectives of the present research, according to the above research questions are as follows. Initially, we must identify the success factors and constraints that Greek software companies face by applying Agile. Subsequently, determine which Agile methods Greek software

manufacturers use to complete their project, and whether specific adaptations are required or not, to achieve the business goals of their company. Finally, we aim to determine if the Agile methodology chosen by each company had favorable or unfavorable results.

What we need to solidify before the start of the research is that the questionnaire that will be distributed should, as mentioned above, be based on standards. These will be defined based on the above papers - research questions so that there is a framework that can be directly comparable to our results. Within the following chapters the research alongside obtained data are being analyzed, based on the detailed description.

3.3. The research

The present dissertation has emerged through the bibliographic review of research carried out mainly by international and domestic scientific fields. The bibliography which was of interest was comprised of articles in scientific journals, various publications and conference proceedings, electronic and online sources, statistical research, books, diploma theses (postgraduate and doctoral level), which relate in one part with the "Agile" methodologies, while in the other we aim to locate their application. From the application of 14 case studies, success factors have been identified alongside the constraints and limitations of applying the "Agile" methodologies during the development of a software program in order to successfully complete it.

Initially, from the 14 case studies, a grouped table was implemented, whereby the success factors and limitations regarding implementation of a software program were separated. They were then assigned to 3 categories, that of human factors, that of organizational factors and that of external factors. Based on the above, we have defined some hypothetical factors, so that our questionnaire may be compiled with them as a basis.

Our questionnaire was divided into three sections. The first section requires information pertaining to the participants' age, gender, role in the company, company size, etc. within the second section, the degree of knowledge and familiarity with Agile methodologies is investigated. Finally, the third section investigates the factors that may contribute to the success of software projects.

Dozens of software developers were requested to participate in the research by forwarding the questionnaires to software developers, project managers, etc. The participants work in various legal form companies ranging from single owner companies, to European or multi-national companies, public organizations and with various types of employment, i.e., freelancers or project contracts.

The questionnaire was created using the freely available google tool, google forms. It was developed in a uniform format, easy to use, and with short comprehensive questions. Invitations for each participant were sent in person, and responses were collected again through the same google forms tool.

For data analysis, a qualitative approach was followed. Specifically, after the questionnaires were collected, the export of the results was started. Comparatively and using tables, we investigated factors such as the use of methods, which of them had a positive effect and which ones limited the evolution of software development processes for successful completion. Subsequently, a commentary - critique is presented based on research cases.

Finally, "This *research aims to promote scientific knowledge in accordance with internationally accepted scientific theories or the elaboration of new theories, capable of being recognized by the international scientific community. It is both a social and an individual good. As a social good, it promotes human knowledge and innovation and thus contributes to improving the quality of life and the well-being of society as a whole. This dimension is inextricably linked to the freedom of researchers, without which it cannot be carried out. In this sense, research work is also an individual good, which is institutionally reflected by its registration as an object of individual right (Greek Constitution, UNESCO Declarations)*". (Source: https://www.uoc.gr/research-at-uni/ethics). In accordance with the above, all the rules of ethics were followed for the elaboration of the present work.

3.4. Description of the research and collected data

The main purpose of this research is to record the degree of success of the application of "Agile" methodologies by companies and software manufacturers in Greece. Although the specific methodologies are globally widespread and accepted, the Greek community does not to date possess a detailed perspective of the current status.

With this in mind, we organized the present research with the aim of first investigating the level of familiarity of software systems analysts, software engineers, programmers and project managers, working in software development companies or IT companies. The research was divided into three sections in detail. The first section provides general information about your role as a research participant. The second section presents questions regarding the degree of knowledge and familiarity with the agile methodologies. Finally, the third section, which is also the main section of the questionnaire, presents questions that examine the contribution of various factors affecting agile methodologies regarding their impact on the success of software development projects.

A survey was conducted between 01 April 2021 and 15 April 2021. A questionnaire was created for the research, which was forwarded personally, to 70 scientists in the field, via the internet and the social networks such as Facebook and LinkedIn, or through personal contacts via email. The entire questionnaire is available in Annex 1. The questionnaire was created using Google's free tool, Google Forms. The goal was to collect as many answers as possible. The answers were collected directly from the form available from Google.

In conclusion, in the present research we will present a qualitative study of the factors that promote the success of the application of the "Agile" methodologies. The method that will be employed for this analysis will arise from the critique of the collected results. Technically, the Google forms tool automatically provides charts based on the datasets so that the analysis may be applied. The interpretation framework will be based on Table 1, as well as the literature and research discussed earlier, serving as guidelines for our own conclusions. The main constraints of our research were time limitations alongside the range of participating professionals, as many are established professionals in the field, and this may not offer as clear a picture of the results as would a wider range of professionals.

This page is intentionally white.

Chapter 4: Results & Suggestions

This fourth and final chapter focuses on the analysis of the research results and will be developed as follows. An analysis of the results which were extracted from the research is initially presented followed bya discussion of the results concerning our observations, their interpretation based on theory as wells as characteristics that may be of particular interest for further study. Finally, we will end this dissertation with the submission of proposals for future research.

4.1. Result Analysis

4.1.1. The hypothetical factors

The above-described research culminated in the collection of 31 answers. Given the time constraints, the 31 responses present a satisfactory representative sample for our analysis, as it covers almost 50% of the total participants.

For the analysis of the results presented in the above study, analysis of the literature and the fourteen case studies, we considered 25 hypothetical factors that lead to the successful completion of a software development project with "agile" methodologies, which are listed below:

HF1. Continuous staff training.

HF2. Utilization of partners with great experience and specialization.

HF3. Communication between project team members.

HF4. Communication with customers and end users.

HF5. Degree of cooperation between the participants.

HF6. Simplicity in design.

HF7. Feedback in case of errors and failures.

HF8. Effective project management.

HF9. Availability of human resources.

HF10. Availability of financial resources.

HF11. Commitment, dedication and responsibility of all participants in fulfilling the objectives of the project.

HF12. Technical knowledge and skills of project members.

HF13. Communication skills and abilities.

HF14. Ability to adapt to changes, both project-related and software-related requirements.

HF15. Detailed documentation in each phase of the project development.

HF16. Existence of standard project management procedures and hierarchical management structures.

HF17. The requirements of the software to be developed are well known in advance and do not change much during the project.

HF18. Existence of recorded detailed specifications for the software requirements.

HF19. Existence of detailed design for the software.

HF20. Detailed planning (design - scheduling).

HF21. Prioritize all requirements from the beginning of the project.

HF22. Ongoing feedback from customers and end users.

HF23. Analytical control procedures (finding and correcting errors).

HF24. Existence of validity checks.

HF25. Defining roles and responsibilities of members from the beginning.

Upon consideration of the factors listed above, the questionnaire was developed and was divided into three sections. The research included scientists from all over Greece, with different qualities, involved with highly successful projects nationwide as well as in Europe and worldwide, etc. As was mentioned earlier regarding one of the constraints of the current research, it is reiterated that the professionals who participated do not belong to a wider field of the design - software development, but only to certain successful projects.

4.1.2. Detailed Results

As mentioned above, the purpose of this study was to record the success factors of the Agile methodologies by software companies in Greece. The questionnaire was structured in three sections, each of which gave separate results. The results are described in detail below.

The **first section** of the questionnaire concerns **general information about the participant**. We have the following in detail:

The **gender factor**, out of the 31 participants, 93.5% are men (29) and 6.5% are women (2). The diagram below schematically represents this quota.

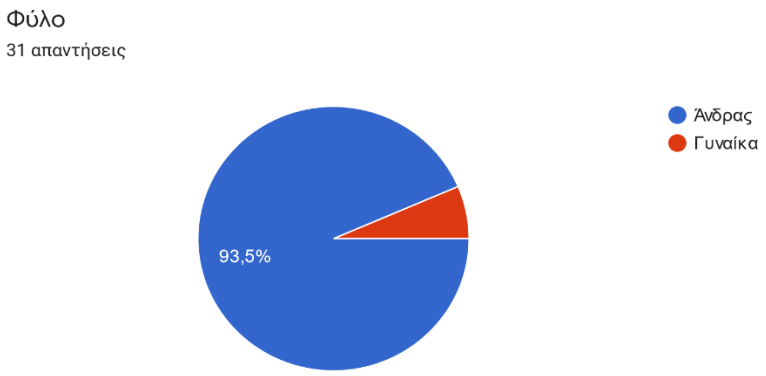


Figure 21 - Graph with the distribution of the "Gender" of the participants

Regarding the **age factor**, 20 - 30-year-olds amount 25.8% (8) of the total number of participants, 30 - 40-year-olds amount to 32.3% (10), 40 - 50-year-olds amount to 29% (9) and finally over 50-year-olds amount to 12.9% (4). The diagram below schematically represents this quota.

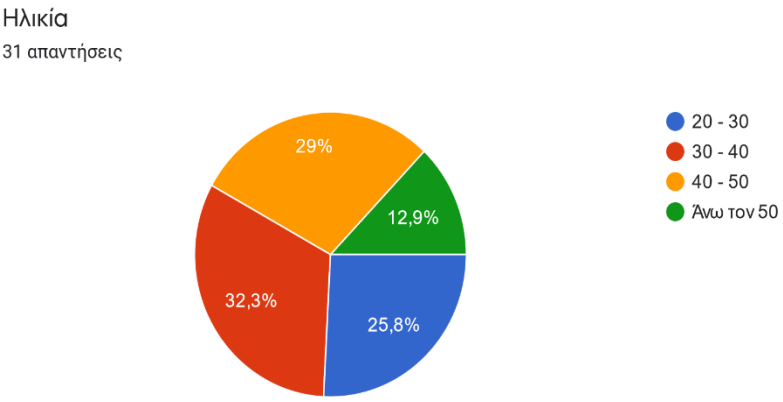


Figure 22 - Graph of the "age distribution" of the participants

Subsequently, as regards **role in the company,** Company Managers amount to 19.4% (6), Project Managers 19.4% (6), Team Leaders 25.8% (8), Developers 38.7% (12), Software Engineers 38.7% (12), Analysts / Designers 22.6% (7), those Responsible for maintenance procedures 3.2% (1) and Other 3.2% (1). The diagram below schematically represents this quota.

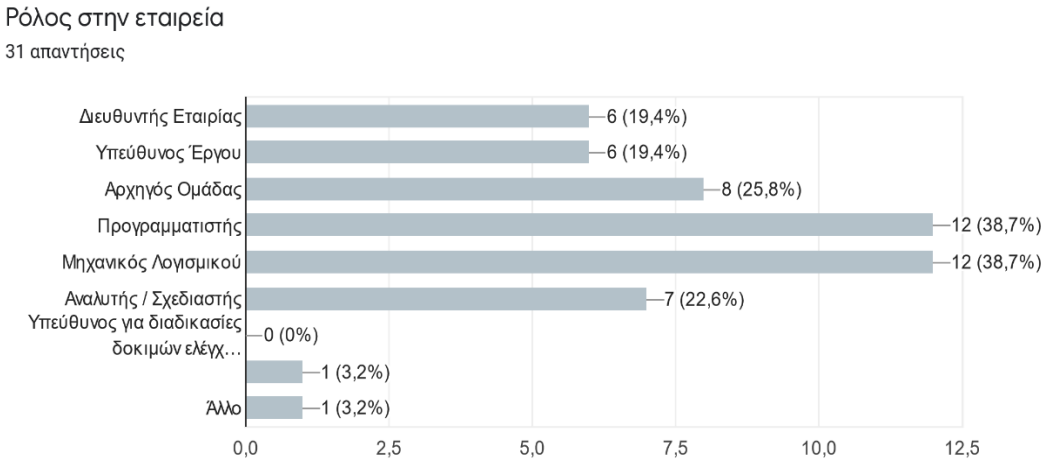


Figure 23 - Graph with the distribution of "roles in the company" of the participants

The answers given by the participants for **the object carried out by the company** are as follows: Management and sale of digital audio tours (Self-guided application), IoT - AI - Cloud / Edge Computing, Bank marketing, Slack bot for asynchronous standup meetings, Smart Software solutions for transforming SME, Web Development, Informatics, Compliance Software for the waste Industry, Online delivery, Software platform for booking concerts, Distribution of Informatics Products, Gambling - games, Development - management of proprietary erp / crm, Software Platform as management service events / exhibitions, Maritime Software and data analytics, Software Development, Mobile Apps, Web / Mobile Development, Advertising - Digital Media, Automation, Logistics Services, Linux Computer Services & Cloud Technologies, E-commerce, Educational Software Development Projects, Software Development, European agencies and Finance.

The distribution for **category of company** to which each participant belongs is as follows: Sole proprietorship 12.9% (4), Small local enterprise 9.7% (3), Regional scope 3.2% (1), Panhellenic scope 22.6% (7), European or Global with headquarters or facilities in Greece 29% (9), European or Global with headquarters or facilities outside Greece 25.8% (8) and Public Organization 3.2% (1). The diagram below schematically represents this quota.

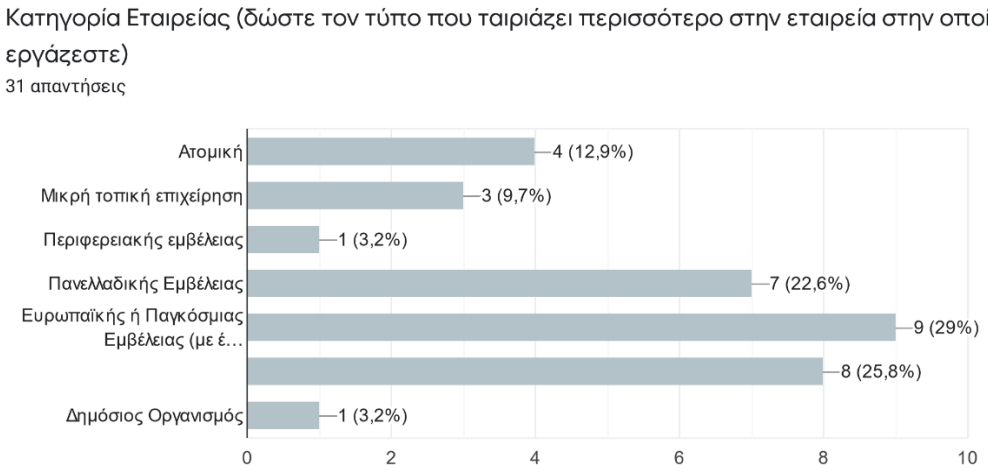


Figure 24 -Graph with the distribution of the "company category" of the participants

In terms of **company size,** companies with less than 10 employees make up 22.6% (7), companies with 10-20 people make up 35.5% (11) and with more than 100 people make up 41.9% (13). The diagram below schematically represents this quota.

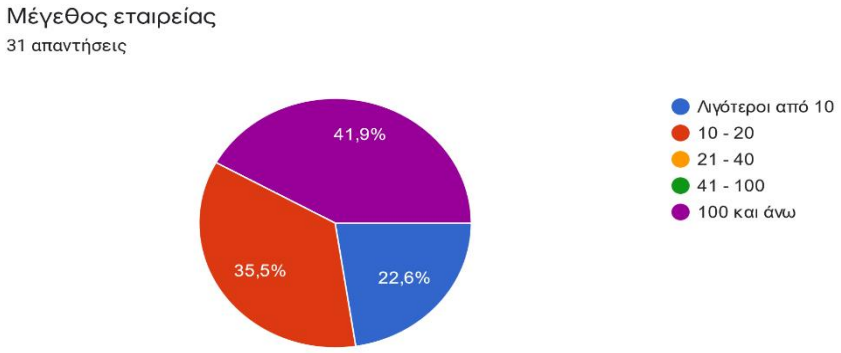


Figure 25 - Graph with the distribution of the "size of the company" of the participants

The last feature of the first section concerns the **typical group size**, so we observe groups with less than 5 people that make up 38.7% (12), groups with 5 - 10 people that make up 35.5% (11), groups with 10 - 20 people that make up 19.4% (6) and groups over 40 people that make up 6.5% (2). The diagram below schematically represents this quota.

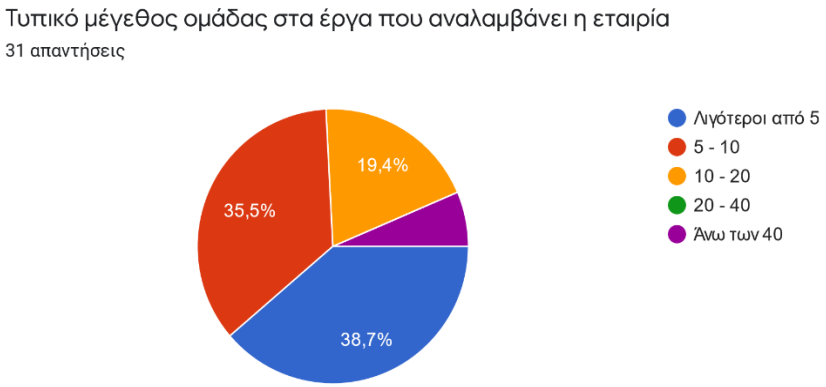


Figure 26 - Graph with the distribution of the "typical group size" of the participants

In the **second section**, the degree of knowledge and familiarity of the agile methodologies is developed. In detail, we have the following characteristics:

For the **period of knowledge** of "agile" methodologies we have the following: for up to 1 year it is 12.9% (4), between 1 - 3 years it is 16.1% (5), between 3-5 years it is 19.4% (6) and above 5 years is 51.6% (16). The Diagram below schematically represents this quota.

Figure 27 - Graph with the distribution of "time they know" the Agile methodologies the participants

For the time of **implementation - their involvement** in projects that apply "agile" methodologies we have the following: for up to 1 year, it is 20% (6), between 1 - 3 years it is 20% (6), between 3-5 years it is 16.7% (5) and above 5 years is 43.3% (13). The diagram below schematically represents this quota.

Figure 28 - Graph with the distribution of "participation in projects" developed with Agile methodologies by the participants

The last feature concerns **the methodology that each participant deals with and is distributed as follows**: The Scrum with a percentage of 58.1% (18), the eXtreme Programming (XP) with a percentage of 12.9% (4), the FDD with a percentage of 32.3% (10), the DSDM with a percentage of 16.1% (5), The ASD with a percentage of 12.9% (4) and finally with a percentage of 3.2% (1) the answers “I do not know / I do not wish to answer”, and their own custom methodologies which were named BDD, Nero and Elements.

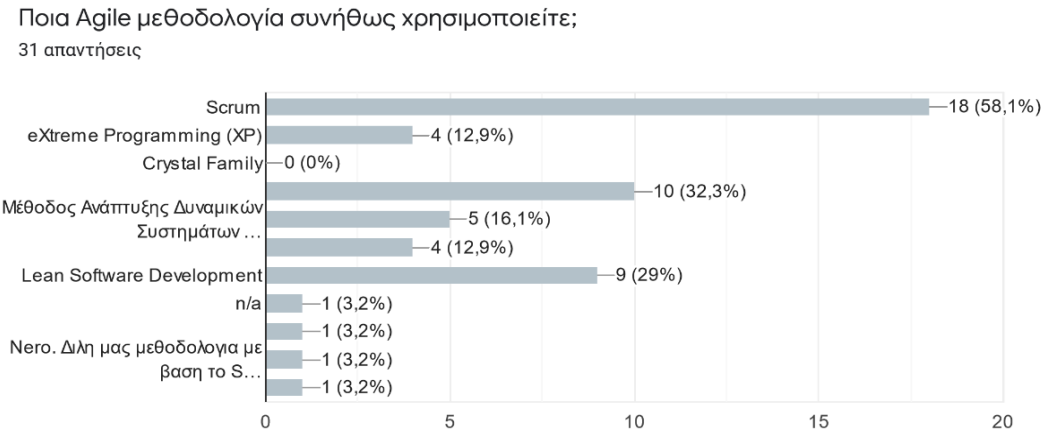


Figure 29 - Graph with the distribution of "the methodology followed" by each participant

In the third section we research those factors that contribute to the success of the software projects. The following table summarizes the participants' answers on the factors that positively affect the success of a software development project.

Table 2 - Results of Success Factors by the participants in the research

SUCCESS FACTORS	VERY	ENOUGH	LITTLE	NONE	DON'T KNOW	NEGATIVELY CONTRIBUTE
1. Ongoing staff training.	58,1	32,3	9,7	0	0	0
2. Utilization of partners with great experience and specialization.	51,6	38,7	9,7	0	0	0
3. Communication among the project team members.	87,1	12,9	0	0	0	0
4. Communication with the customers and the final users.	54,8	38,7	6,5	0	0	0
5. Degree of cooperation among the participants.	51,6	45,2	3,2	0	0	0
6. Simplicity in the design.	38,7	45,2	16,1	0	0	0
7. Feedback in case of errors and failures.	45,2	51,6	3,2	0	0	0
8. Effective project management.	38,7	51,6	9,7	0	0	0
9. Availability of human resources.	32,3	41,9	22,6	3,2	0	0
10. Availability of financial resources.	32,3	41,9	22,6	3,2	0	0
11. Commitment, dedication and responsibility of all the participants in fulfilling the objectives of the project.	45,2	51,6	3,2	0	0	0
12. Technical knowledge and skills of the project members.	45,2	54,8	0	0	0	0
13. Communication skills and abilities.	29	54,8	16,1	0	0	0
14. Ability to adapt to changes, both project-related and software-related requirements.	41,9	54,8	3,2	0	0	0
15. Detailed documentation in each phase of the project development.	32,3	41,9	25,8	0	0	0
16. Existence of standard project management procedures and hierarchical management structures.	19,4	35,5	35,5	3,2	3,2	3,2
17. The requirements of the software to be developed are well known in advance and do not change much during the project	16,1	38,7	32,3	12,9	0	0
18. Existence of recorded detailed specifications for the software requirements.	35,5	35,5	22,6	3,2	3,2	0
19. Existence of detailed design for the software.	22,6	41,9	25,8	3,2	6,5	0
20. Detailed planning (design - scheduling).	19,4	51,6	29	0	0	0
21. Prioritize all the requirements since the beginning of the project.	16,1	48,4	32,3	3,2	0	0
22. Ongoing feedback from the customers and the final users.	32,3	48,4	19,4	0	0	0
23. Detailed control procedures (error detection and correction).	46,7	50	3,3	0	0	0
24. Existence of validity checks.	38,7	51,6	9,7	0	0	0
25. Defining the roles and the responsibilities of the members since the beginning.	19,4	54,8	25,8	0	0	0
ΜΕΣΟΣ ΟΡΟΣ	**38,01**	**44,572**	**15,5**	**1,284**	**0,516**	**0,128**

The data depicted in table 2 above provides the first detailed view of our results by which it is possible to identify whether the above factors enable software developers in Greece to have successful projects. The analysis of the research results is expanded upon below.

4.2. Discussion of results

4.2.1. What the results show and how we interpret them

The most important conclusions deduced from each section as presented within our research, are summarized separately below.

In the **first section**, we have the following results - conclusions:

- The **role in the company**, received multiple - combined answers. 40% of the participants have at least two or more roles. According to the principles of the Agile Manifesto, the description of the methodologies above, and the case studies, make it evident that each participant in the project clearly assumes a role. The multiplicity of roles is not mentioned anywhere. Furthermore, the multiplicity of roles is found mainly in companies of small or medium size, and is the main feature of Greek companies, which may often lead to project failure.
- The **object of the company**, demonstrates a wide range of work objects, which indicates that in Greece we observe companies constantly developing upon various objects. Additionally, the results obtained display a validity, i.e., we have a community that deals and applies exclusively with the agile methodologies.
- For the **company category**, we observe multiple - combination features. We found that almost 50% of the participants work for large companies of European or global scope with facilities either in Greece or abroad. It is noteworthy that two of the participants gave multiple answers, which probably explains why the participants work on multiple projects, and indicates the possible use of multiple methods of the "agile" software project development methodologies.
- The factor of **the size of company** indicates that large differences are found in company sizes. On the one hand we have companies that employ up to 10 or 20 people, and on the other hand the largest percentage of 43.8% represents companies with personnel of more than 100 people. According to the article "***Challenges of adopting agile methods in a public organization***", by Nuottila, J., Aaltonen, K. & Kujala, J. (2016) analyzed in chapter 3.1.1. Small businesses and groups face several constraints such as: Psychological barriers (social stress), Different ideology, Organizational readiness (management involvement), Continuing education - training, Cooperation, etc.

- The **typical size of the group** offers the following results. 37.5% of the participants state that their group has less than 5 people and another 37.5% of the participants state that their group has between 5 - 10 people. According to the Agile methodologies, the size of the group should not exceed 8 to 10 people. Therefore, for the majority of participants the correct application of agile methodologies is observed. However, a percentage of 18.8% represents groups of 10-20 people and a percentage of 6.3% represents groups of more than 40 people. That a combined 25.1% includes groups of more than 10 people, should be of concern to the practitioners of Agile methodologies, as large groups are not agile or flexible for a number of reasons.

The **second section** gives us the following results - conclusions:

- The factor pertaining to **the length of time participants have been aware of "Agile" methodologies**, for which 100% of the participants answered in the affirmative. All participants had been aware of "agile" methodologies, while 50% of them had been aware of the methodologies for more than 5 years. 18.8% of the participants had been aware of agile methodologies for 1-3 years and 3-5 years collectively, while a final 12.5% had only been aware for 1 year or less. This was surprising since the methodologies have been widely known and applied for the past 20 years.
- The characteristic pertaining to **the length of time participants have been involved in projects that apply the "Agile" methodologies,** for which 97% of the participants answered in the affirmative, while 3% of the participants did not answer at all. The percentage of 3% who did not answer, is interpreted as not being involved in IT projects using such methods either due to lack of knowledge of the application of such methods or due to being involved with projects employing traditional methodologies
- The characteristic pertaining to **which methodology is used to develop the software projects** resulted in some surprises. Initially, a multiplicity of answers is obtained. Almost 50% of the participants practice multiple methodologies rather than just one. There is of course no guarantee of this practice being useful or practical. The most common methodology is the Scrum, being applied by 59.4% of the participants, in contrast to the Crystal Family which is not applied by any of the participants (0%). Scrum is followed by the FDD with 31.3%, Lean with 28.1%, DSDM with 15.6, the XP and ASD with 12.5%. It is worth noting that, 3 respondents stated that they developed and use their own custom methods.

The **third section** is the main part of the research where the factors that lead to the success of a software development project are listed. Here we have the following results - conclusions:

- As mentioned in the second section, 3% of participants have not been involved with software development projects using the "agile" methodologies. In the current section however, the participation could be considered as "normal"-"adequate". It is assumed here that the participants responded based on their current situation and their participation in software development projects employing traditional methodologies.
- 93% of the participants have a positive opinion regarding the use of "agile" methodologies, and stated that having used them either in their original format or after various modifications, they have satisfied their company's goal to implement the projects successfully. It is worth noting that no negative responses were obtained (0%), while 3.1% of the participants maintain a neutral opinion and another 3.1% commented that each project should be considered separately for the suitability and applicability of this type of methodology.
- According to Table 1 presented in the third section, considering all of the success factors we obtain the following results regarding the contribution of all of them collectively towards success. The average of the success factors with a response of "A lot" has a percentage of 38%, "Enough" has 44.5%, "A little" has 15%, "Not at all" has 1.28%, " I do not know "has 0.52% and finally" Contributes Negatively", has 0.13%. According to the results above, these factors contribute positively to the success of software project by applying the "agile" methodologies, since the sum of 82.5% fluctuates between "Enough" and "A lot".
- The **human factors** that help in the successful completion of the project include the following characteristics:
  - Regarding **continuing education**, 90.4% of the participants answered "Enough" to "A lot", while the remaining 9.6% answered "A little". According to the theories of the "agile" manifesto developed above, in all of the 14 case studies, continuing education plays a key role to the success of the application. So, 100% of the participants are trained in new technological developments either "A Little" to "A lot".
  - **Communication** is divided into three sections, and plays an important role in the success of projects.
    - **The Communication among the members of the project team**, where 87.1% of the participants answered "A lot", while the remaining 12.9% responded with "Enough". Therefore, 100% consider communication

among the members important for success, while at the same time implementing it.

- **Communication among the customers and the end users**, where 54.8% of the participants responded "A lot", 38.7% "Enough" and 6.5% "A little". A total of 100% consider communication among customers and the end users important for success.
- **The communication skills and the abilities of each member**, where 29% of the participants answered " a lot", 54.8% answered «Enough", while 16% answered "A little". The 16% who answered "A little" pose reason for concern, because communication as is known, plays an important role in the success of project development. It is evident here that a significant percentage of the participants has NOT sufficiently developed the required communication abilities and skills.

Communication as a whole play a key role to success as is widely known. In our survey, 100% of the participants answered "A little" to "A lot" suggesting communication is a positive factor in project development.

- **The commitment, dedication and responsibility of participants in fulfilling the objectives of the project plays** an important role in success. Thus, 45.2% responded "A lot", 51.6% responded with «Enough", while 3.2% responded "A little". All of the participants (100%) believe that commitment, dedication and responsibility led to success, and this is evident as becomes apparent from their responses.
- **The degree of cooperation** of all participants plays an important role in success. 51.6% answered "A lot", 45.2% answered "Enough", while 3.2% answered "A little". All of the participants (100%) believe in collaboration promoting successful outcomes.
- The **technical knowledge and skills of project members,** represent one of the most important factors in project success. 45.2% answered "A lot", while the remaining 54.8% answered "Enough". The total of 100% of participants confirms the determining role of the knowledge as a factor. The ability to adapt to both project-related and software-related changes plays an important role in success. 41.9% answered "A lot", 54.8% answered "Enough", while 3.2% answered "A little". So, 100% agree that developing skills to adapt to change, is important.

• The **organizational factors** that help in the successful completion of the project include the following characteristics:

- The **detailed documentation** in each phase of project development, provided the following results. 32.3% responded with "A lot", 41.9% responded with "Enough", while 26% answered "A little". 26% represents one quarter of the participants, which shows a significant percentage of people do not consider documentation to be that important for successful project development.
- **Utilizing partners with extensive experience and expertise** is an important key to the success of software project development. 51.6% answered "A lot", 38.7% answered "Enough", while 9.7% answered "A Little". 100% of the participants participate in projects where the proper utilization of experienced members plays an important role.
- **Design** is divided into 3 subsections that play an important role in projects success. They are:
    - **Simplicity of design** plays an important role in success. The participants for this factor answered "A lot" 38.7%, "Enough" 45.2%, while "A little" 16%. This factor should be of concern because it is obvious from the collected results that product design in Greece continues to lack simplicity. The design simplicity factor needs to be reconsidered and re-examined.
    - **The existence of detailed design for the software,** is another factor of concern based on our results. 22.6% of the participants answered "A lot", 41.9% answered "Enough", 26% answered "A little", 3.2% answered "Not at all", while 6.5% answered that "I do not know". The total of 35.7%, i.e., a little over 1/3 of the participants do not have a clear picture of the design of the product they are developing, therefore awareness of the importance of detailed design should be increased.
    - **Detailed planning (for design and scheduling),** the last factor of the design aspects provided results that require further analysis. 19.4% answered "A lot", 51.6% answered "Enough", while 29% answered "A little". The 29% who represent 1/3 of the participants see that planning does not progress properly, with a possible outcome the potential failure to complete the project.

  Overall, the participants do not have a clear perception of **the three design factors** that constitute important key factors contributing to the success of project development. The 12 principles of "Agile" methodologies however, make it clear that the design factor plays a primary and decisive role.

- **Feedback, verification and validation procedures** are divided into four sections playing an important role in the success of projects. We obtained the following:
    - **Feedback in case of errors and failures**, is important in the success of project development. 45.2% of the participants answered "A lot", 51.6% answered "Enough", while 3.2% answered "A little". A total of 96.8% participate in projects where feedback is continuous to avoid problems prohibiting success.
    - **Continuous feedback from customers and end users** is key to the success of project development. 32.3% of the participants answered "A lot", 48.4% answered "Enough", while the remaining 19.3% answered "A little". A total of 80.7% of the participants establish continuous feedback with customers following the values of "Agile" methodologies, resulting in successful projects. It is of concern that 19% do not share this opinion.
    - The **detailed control procedures (finding and correcting errors),** provide the following results. 46.7% of the participants answered "A lot", while 50% answered "Enough", while 3.3% answered "A little". In conclusion, 96.7% of all participants follow control procedures.
    - The **existence of validity checks** provided the following information. 38.7% of the participants answered "A lot", 51.6% answered "Enough", while 9.7% answered "A little". The total of 90.3% works on projects with developed validity checks.

  The sub factors of **control and validity** are perceived as positives towards the development successful projects. **Continuous feedback from customers and end users is** the main key to ultimate success.
- **Management** is divided into three subsections playing a key role in the success of projects.
    - **Effective project management** is a factor that helps in both the organization and especially in project success. 38.7% of the participants answered "A lot", 51.6% answered "Enough", while 9.7% answered "A little". A total of 100% of the participants agree and enjoy effective management in their projects.
    - **Defining the roles and responsibilities of the members from the beginning**, should be clear from the outset for the success of projects. The participants answered "A lot" with 19.4%, 54.8% answering "Enough", while 26% answered "A little". The percentage of 26%, which is 1/4 of the participants, do not have a clear role. This indicates that small businesses

are likely to give participants multiple roles, which may lead to confusion as regards responsibilities.

- **The existence of standard project management procedures and hierarchical management structures**, is a key success factor for the development of a project. 19.4% of the participants answered "A lot", 35.5% answered "Enough", 36% answered "A little", while from 3.2% answered "Not at all", "I do not know" and "Contributes Negatively" ». A total of 45.6% of the participants indicate that there are NO management procedures and management structures. This is of concern and we should be able to redefine this factor.

Therefore, **Management** as an organizational factor, should be re-examined from its conception, because the results representing our participants indicate that Management's role is not clear.

- The **requirements and specifications** are divided into three sections that play an important role in the success of the projects.
    - The **requirements of the software to be developed are well known in advance and change little during the project**. This is a key factor pertaining to the success of projects. 16.1% of the participants answered "A lot", 38.7% answered "Enough", 32% answered "A little", while 12.9% answered "Not at all". The sum of "Little and Not at all" answers is a staggering 44.9%, almost half of the participants. This result of great concern because according to the principles of the "Agile" Manifesto, the requirements should be defined from the outset, and should be changed according to emerging needs. Immutability is an example of traditional development methods.
    - The **existence of recorded detailed specifications for software requirements**, plays an important role in project success. 35.5% of the participants answered "A lot" and "Enough", 23% answered "A little" and 3.2% answered "Not at all" or "I do not know". The total of 29.4%, which is almost the 1/3 of the participants, do NOT have a clear idea of the specifications and some have no idea at all. This does not promote project success according to the principles of the "Agile" manifesto.
    - **Prioritizing all the requirements from the start** is a key factor to success. 16.1% of the participants answered "A lot", 48.4% answered "Enough", 32% answered "A little", while 3.2% answered "Not at all". As regards this factor, we observe that 35.2% which is equivalent to a little

more than 1/3 of the participants are not aware of the priorities. Overall, the requirements and the specifications, as factors, play a crucial role in the success of project development. In our research, it is clear that none of the three sub-factors are given the desired importance. That is why we need to reconsider these factors.

- **External factors** that contribute to the successful completion of the project include the following characteristics:
  - **The availability of human and financial resources are** important factors in the success of software projects. 32.3% answered "A lot", 41.9% answered "Enough", 23% answered "A little", while 3.2% answered "Not at all". Thus, for sum of 74.2%, a large percentage of human and financial resources is available, while for 25.8%, i.e., in ¼ of the participants, the resources are not available, negatively affecting project success. We need to reconsider these two factors so that 100% success may be achieved.

4.2.2. Individual Results that need further – special interpretation

Further considering and expanding our research, it is worth noting some characteristics - factors that have provided a particular set of results and therefore need special attention. The following discussion could be incentive for further study or could aid in identifying what leads to failure in project development.

The characteristic of the **company category** as mentioned above, indicated that more than 50% of the participants are employed in large companies or in large projects. This concerns us because this feature touches on both limitations, the one dealing with the range of participants and the time constraints of the research. As mentioned above, our research was addressed to a wide environment of researchers in a short time. This had the undesired effect of all the participants being employed by large companies, which develop projects with "Agile" methodologies. But this does not offer insights into what is happening in smaller companies and further what is happening with the application of the methodologies. Furthermore, two of the respondents, as we found out, are involved with more than one company and project at the same time. This should be of particular concern to us, because we discussed that from the introduction of the "Agile" manifesto, clearly defined roles of participants in a single project is of importance because this allows them to develop their skills fully, etc. On the other hand, this can have a positive effect, due to further expansion of knowledge through the use of multiple "agile" methods and the gaining of additional significant experience.

The characteristic of the **application of the "Agile" methodologies** needs special interpretation. As mentioned above, concerning the period of one being aware of "Agile" methodologies, 100% of the participants answered in the affirmative. 97% answered they are aware of their own participation, while3% did not know the answer. The interpretation of these answers could suggest that although the participants are aware of the methodologies, they may be developing relevant skills via the use of traditional methods within their respective projects, risking project success.

The characteristic of the **methodology followed by the participants** offered results that peaked our interest. In addition to the classical answers, we observe the following characteristics which need further investigation: Initially, 50% of the participants answered that they use more than one methodology. This raises some doubts regarding the benefits and practicalities of such a practice with regards to the successful completion of their respective projects. Furthermore, 3% of the participants confirm that they do NOT use "Agile" methodologies, since they did not provide an answer at all for this section. The Traditional project development methods are probably in use, risking project success. Finally, 10% of the participants answered that their companies have adopted their own standards - practices - methodologies and software development procedures, i.e., they do not formally follow the methodologies, and these custom methods have been proven to achieve results. This needs special attention because we cannot know to what extent these aids or guarantees the success of the projects, but if their methodology does ensure success, then it would be crucial for it to be implemented for a wider audience.

The factor of the **analytical documentation**, provided a significant result of 26%, i.e., ¼ of the participants, responded that there is "Little" documentation of the projects they develop. According to the article "***Success and Failure Factors that Impact on Project Implementation Using Agile Software Development Methodology***" by Dhir, S., Kumar, D. & Singh, VB (2019), mentioned above in chapter 3.1.10, documentation plays a crucial role in the success of the projects. **Less documentation** although it prevents extra effort, it can create problems for members especially by limiting access to project requirements.

The **design factor (Simplicity of design, Existence of detailed design & Detailed Planning)**, according to the principle of the "Agile" manifesto for "Continuous attention" to technical perfection and good design with enhanced agility" and the article "***Success and Failure Factors that Impact on Project Implementation Using Agile Software Development Methodology*** » by Dhir, S., Kumar, D. & Singh, VB (2019), cited above in chapter 3.1.10, it is clear that design is a key factor for successful development of projects. In our research we obtained the following characteristics. For "**Simplicity in Design**" 16% of the participants answered that there is

"Little" simplicity in the design of the projects they participate in and develop. For the "**Existence of Detailed Design**", 35.7% answered that they have a negative opinion. This means, the projects in which they participate and develop have no design. For "**Detailed Planning**" 29% of the participants answered that there is "Little" planning in their projects. In conclusion, the design factor as a whole play a key role to the success of the projects using the Agile methodologies. In our research, as we saw earlier, all three areas of design require further analysis because the participants responded largely negatively to each of the three parts of it.

The factor of **Management (Effective Management, Defining the Roles & the Responsibilities and the Existence of formal procedures & Hierarchical Structures),** is where our attention is subsequently focused. According to the article "***Success Factors of Agile Information Systems Development: A Qualitative Study***" by Hummel, M. & Epp, A. (2015) developed in chapter 3.1.12, as well as to the other cases and theories, the management factor presents a different organization from compared to traditional methods, which is based on the Management Support, the "agility" and the general organization. Regarding "***Effective Management***" the participants in their vast majority answered that they participate in companies where the Management plays an important and decisive role. 10% of the respondents felt that «A Little» better described the efficiency of their managers. For "***Defining the roles and the responsibilities of the members since the beginning***", 26% of the participants, i.e., ¼ of them, answered that they comprehend the role «A little», while 54.8% answered "Enough". According to these results, 80.8% of the respondents do NOT have a clear understanding of their role in the project compared to 19.2% who are fully aware of their responsibilities within the project. As regards the "***Existence of formal project management procedures and hierarchical management structures***" the participants have a negative opinion. 36% answered that there are "Few" managements procedures and administration structures, while 3.2% (i.e., 9.6%) share the answers of "Not at all", "I do not know" and "Contributes Negatively". Thus, a total of 45.6% maintain a completely negative opinion, along with 35.5% of the participants who believe that there are "Enough" management procedures and hierarchical structures. The fact here then is that 81.1% are not sure about the existence of procedures. In conclusion therefore, it appears that the **factor of Management as a whole**, needs to be further promoted since the participants do not demonstrate familiarity with the importance of this particular factor. This could have negative consequences and ultimately lead to project failure.

The **factors of the requirements and specifications (Requirements are known in advance & change little during development, Existence of recorded specifications and Prioritization of all the requirements since the beginning),** according to several principles of the “Agile" manifesto, play a decisive role for project success. The factor "**The requirements of the software to be developed are known in advance to a large extent and**

change little during the project" received the following answers: 32% of the participants believe "A little", while 12.9% are of the opinion that it contributes «Not at all". Therefore, the total of 45%, i.e., about ½ of the participants do not have a clear idea of the requirements from the start, while at the same time they agree that the requirements change little although the basic principles and values of the manifesto clearly suggest variability. The factor of "**Existence of recorded detailed specifications for the software requirements**" resulted in the following: 23% of participants answered that they know "A little" about the requirements and specifications of the software under development, while 3.2% replied "Not at all" and "I do not know". Thus, a total of 30%, i.e., almost 1/3 of the participants do NOT know the specifications of the requirements of the software under development, and this may ultimately endanger project success. Finally, for the "**Prioritization of all the requirements since the beginning of the project**" 35.2% of the participants, i.e., more than 1/3 of the total answered with "Little" and "Not at all" indicating a lack of knowledge regarding the priorities. In conclusion, the factors of the **requirements and the specifications as a whol**e that play a decisive role according to the principles of the "Agile" manifesto, and lead to a success, appear from our conducted research to not have the desired results according to the participants answers, endangering the course of project development.

Finally, the element of the "**Continuous feedback from customers and the end users**" which belongs to the group of the factors of **feedback and control procedures & validity of organizational factors**, requires further discussion and analysis. Only 20% of the participants answered in the affirmative, which demonstrates that there is "Little" feedback among the participants and the customers - end users. According to the principles and the values of the "agile" manifesto, feedback plays a key role towards the success of project development. Therefore, the results of this portion of the research and this factor in particular could be a topic of further investigation in order to obtain a better understanding of the situation.

In conclusion, after studying the results of our research one may draw conclusions pertaining to a number of interesting aspects regarding the elements that are the key to success during software development. We can additionally single out and pay special attention and attempt further interpretation of a number of factors. The characteristics of company category, knowledge and application of the "agile" methodologies, which methodology is followed by the participants, alongside the factors of detailed documentation, planning, management, requirements - specifications in general as a whole, and individually feedback with customers and end users require further consideration. As our research shows, the participants do not have a clear idea of the above characteristics and factors and this may result in the failure of the projects in which they participate.

4.3. Conclusions & Suggestions for future research

4.3.1. Conclusions

The "**Agile *methodologies are not a panacea, nor a fad, but each project must be considered separately so that we can distinguish the appropriateness of an 'agile' methodology if the project allows it***."

The above words were quoted by a participant that took part in our investigation, answering for the last question of the questionnaire provided. The question was whether they have a positive or negative opinion of using the "agile" methodologies leading to the success of the projects.

The aim of this dissertation was to investigate whether the "agile" software development methodologies are applied by Greek companies, and which of the factors lead to their successful application and project success. The work began defining the project and project management, while slowly moving on to examining the software and software technology. Then the life cycle of projects with traditional as compared to the "agile" models was developed, while 14 case studies - application of the methodologies were reported. Based on all of the above, we conducted a qualitative research, with some measurable features being compared to theory and the fourteen case studies. The results highlighted those factors that require further investigation so that when combined with the other factors and characteristics they may increase the possibility of success in project development.

The author's experience in the present dissertation led to the following conclusions:

- Aiming to identify research and literature with the application of "agile" software development methodologies in Greece, one quickly realizes their lack thereof. This is something novel and therefore unique, requiring special attention in the manner that it may be undertaken and within the context in which the results are interpreted.
- The time period in which the research was developed and conducted was limited. This should be taken into account as a limitation of the research.
- The range of the participants consisted of accomplished professionals, who participate in successful projects of great acceptance. Participants from smaller projects or from small and regional companies could not be identified, in order to expand the potential pool of varying opinions and answers. This element should be taken as another limitation of our research.
- Several of the factors submitted to the participants proved to be incomplete, resulting in the unlikely successful development of the projects.

The findings of the present study as arrived at based on data interpretation cannot be considered as absolute, as they involve qualitative parameters developed in the previous chapters.

However, their discussion serves as a reference point which is the basis for drawing the general conclusions in this chapter.

The contribution of the present research is not to simply mention the application of the "Agile" software development methodologies by Greek designers - manufacturers, but to point out the success factors, including which ones require special attention. The successful implementers of the methodology and its proper development were also part of this study.

4.3.2. Future Research

The nature of this dissertation has led to the conception of certain quantitative elements - characteristics of the factors that lead to the success of software development projects, applying the "Agile" methodologies based on the qualitative elements developed by the theory.

The present investigation could be extended to:

- A study with a longer duration
- A study with a wider range of participants, so that the whole market is represented by the data
- A study that will have more factors and quantifiable characteristics, so that may gain a more accurate and measurable view.
- A study in which the prospective researcher should resort to looking for more information about the Greek market.

Finally, I hope that the present dissertation has given enough impetus, so that the prospective future applicants of these "Agile" methodologies, but also the researchers who may resort to it, in order to extract data to expand their knowledge and their professional experience, as well as to organize further research for the sole purpose of project success.

This page is intentionally white.

Bibliography

- Masood, Z., Hoda, R. & Blincoe, K. (2018) Adapting Agile Practices in University Contexts, New Zealand, Elsevier, Volume 144, pp: 501 – 510, doi: https://doi.org/10.1016/j.jss.2018.07.011
- Nuottila, J., Aaltonen, K. & Kujala, J. (2016) Challenges of adopting agile methods in a public organization, International Journal of Information Systems and Project Management, Vol. 4, No. 3, 65-85, doi: 10.12821/ijispm40304
- Kouzari, E., Gerogiannis, V., Stamelos, I. & Kakarontzas G. (2015), Critical success factors and barriers for lightweight: software process improvement in agile development: A literature review, France, 10th International Conference on Software Engineering and Applications (ICSOFT-EA-2015), pages 151-159, doi: http://dx.doi.org/10.5220/0005555401510159 & http://www.icsoft-ea.org/
- Misra, S. C., Kumar, V. & Kumar U. (2009) Identifying some important success factors in adopting agile software development practices, India, Elsevier, Volume 82, Issue 11, pp: 1869 – 1890, doi: https://doi.org/10.1016/j.jss.2009.05.052
- Agile Alliance (2001), Manifesto for Agile Software Development, www.agilemanifesto.org
- Rubin K. (2012), Essential Scum – A practical guide to the most popular Agile Process, Addison – Wesley
- Aurum, A. & Wohlin C. (2005), Engineering and Managing Software Requirements, Springer
- Cohn, M. (2005), Agile Estimating and Planning, Pearson Education Inc
- Stellman, A. & Greene, I. (2006), Applied Software Project Management, O'Reilly Media
- Burke, R. (1999), Project Management - Planning and Control Techniques, J. Wiley & Sons LTD
- Stellman, A. & Greene, J. (2015), Learning Agile, O'Reilly Media
- Burke, R. & Barron, S. (2014), Project Management Leadership, J. Wiley & Sons LTD
- Cohn, M. (2010), Succeeding with Agile – Software Development using Scrum, Addison – Wesley
- Cohn, M. (2004), User Stories Applied – For Agile Software Development, Addison – Wesley
- Fitsilis, P. (2004), Designing Software – eshop Case Study, Hellenic Open University

- Papatheocharous, E. & Andreou, A. (2013), Evidence of Agile Adoption in Software Organizations: An Empirical Survey, F. McCaffery, R.V. O'Connor, and R. Messnarz (Eds.): EuroSPI 2013, CCIS 364, pp. 237–246, DOI: 10.1007/978-3-642-39179-8_21
- Katsikas, D. (2013), Modern Methods of Computer Project Management, University of Macedonia
- Rigopoulou, M. (2014), Scrum Methodology and its Application in Information Systems Development, University of Patras
- Pichliavas, D. (2015), Techniques Analysis for Development of Innovative Software Products, of the Extraction 's Methodology and the User's Requirements, Ethniko Metsovio Politechnio
- Monochrisou, E. (2010), Computing Project Management Methodologies: Agile methodologies and their use in public computing projects, University of Macedonia
- Sheffield, J., Lemétayer, J. (2012), Factors associated with the software development agility of successful projects, New Zealand, Elsevier, Volume 31, Issue 3, April 2013, Pages 459-472, doi: https://doi.org/10.1016/j.ijproman.2012.09.011
- Ahimbisibwe, A., Cavana, R., Daellenbach, U. (2013), A contingency fit model of critical success factors for software development projects. A comparison of agile and traditional plan-based methodologies, New Zealand, Journal of Enterprise Information Management, Vol. 28 No. 1, 2015, pp. 7-33, doi: 10.1108/JEIM-08-2013-0060
- Pikkarainen, M., Salo, O., Kuusela, R., Abrahamsson, P. (2011), Strengths and barriers behind the successful agile deployment – insights from the three software intensive companies in Finland, Springer Science – Business Media, Vol 17, pages: 675 – 702, DOI: 10.1007/s10664-011-9185-5
- Misra, S. C., Kumar, V., Kumar, U. (2009), Identifying some critical changes required in adopting agile practices in traditional software development projects, Emerald Group Publishing Limited, International Journal of Quality & Reliability Management, Vol. 27 No. 4, 2010, pp. 451 - 474, DOI: 10.1108/02656711011035147
- Gandomani, T. J., Zulzalil, H., Ghani, A. A. A., Sultan, A. B. M. & Sharif, K. Y. (2014), How Human Aspects Impress Agile Software Development Transition and Adoption, International Journal of Software Engineering and Its Applications, Vol.8, No.1 (2014), pp: 129-148, DOI: http://dx.doi.org/10.14257/ijseia.2014.8.1.12
- Dhir, S., Kumar, D. & Singh, V. B. (2019), Success and Failure Factors that Impact on Project Implementation Using Agile Software Development Methodology, Springer Nature Singapore Pte Ltd. 2019 M. N. Hoda et al. (eds.), Software Engineering, Advances in Intelligent Systems and Computing 731, pages: 647 – 654, DOI: https://doi.org/10.1007/978-981-10-8848-3_62

- Tam, C., Moura, E. J. D.C., Oliveira, T. & Varajão, J. (2020), The factors influencing the success of on-going agile software development projects, Elsevier, International Journal of Project Management 38 (2020), pages: 165 – 176, DOI: https://doi.org/10.1016/j.ijproman.2020.02.001
- Hummel, M. & Epp, A. (2015), Success Factors of Agile Information Systems Development: A Qualitative Study, 48th Hawaii International Conference on System Sciences, DOI 10.1109/HICSS.2015.598
- Aldahmash, A., Gravell, A. M. & Howard, Y. (2017), A Review on the Critical Success Factors of Agile Software Development, Springer International Publishing AG 2017J, Stolfa et al. (Eds.): Euro SPI 2017, CCIS 748, pp. 504–512, 2017, DOI: 10.1007/978-3-319-64218-5_41
- Asnawi, A. L., Gravell, A. M. & Wills, G. B. (2010), An Empirical Study: Understanding Factors and Barriers for Implementing Agile Methods in Malaysia, 5th International Doctoral Symposium on Empirical Software Engineering (IDoESE)
- Research Ethics Committee of the University of Crete, Rules of Ethics and Conduct, https://www.uoc.gr/research-at-uni/ethics

This page is intentionally white.

Annexes

Research Questionnaire

University of Thessaly

Management & Economics School

Department of **Business Administration**

Postgraduate Program: **"Project & Program Management"**

Questionnaire in the context of the postgraduate dissertation entitled:

"***Investigation of the degree of use of" Agile "management methods by Greek software companies***"

Student name: **Athanasios Gourobinos / athagkou10@uth.gr**

Supervising professor: **Vassilios Gerogiannis**

This questionnaire is addressed to software systems analysts, software engineers, programmers and project managers, working in software development companies or IT companies.

The questionnaire is conducted in the context of research carried out in the diploma thesis of the postgraduate student Athanasios Gourobinos, on the subject: "Investigation of the degree of use the "agile "management methods by Greek software companies". The present work is carried out in the Postgraduate Program entitled "Project Management & Programs" of the School of Management and Economics of the University of Thessaly. Mr. Vassilios Gerogiannis, professor at the University of Thessaly, is supervising this postgraduate thesis.

The purpose of this questionnaire is to investigate the factors that positively contribute or negatively affect the success of the application of the “agile” software development methodologies in the environment of a company that develops software.

We assure you that the answers given to the following questionnaire will be completely anonymous and that the collection and the processing of the questionnaire data, as well as their publication, in the context of this dissertation, are subject to state data protection laws. Their processing and use will be used exclusively, for the purposes of this work.

The questionnaire is divided into three sections. The first section provides general information about your role as a research participant. The second section presents questions regarding the degree of knowledge and familiarity with the "agile" methodologies. The third section, which is also the main section of the questionnaire, presents questions that examine the contribution

of various factors of the "agile" methodologies regarding their impact on the success of the software development projects.

E-mail*

Section A: General information

Sex:

- Man
- Woman

Age:

- 20 - 30
- 30 - 40
- 40 – 50
- Over 50

Role in the company:

- Company Manager
- Project director
- Team leader
- Developer
- Software Engineer
- Analyst / Designer
- Responsible for control test procedures
- Responsible for the maintenance process
- Other:

Object of the projects implemented by the company (describe briefly):

…

Company Category (give the type that best suits the company you work for):

- Individual
- Small local business
- Regional scope
- Pan-Hellenic Scope
- European or Global Scope (with headquarters or facilities in Greece)
- European or Global Scope (with headquarters and facilities abroad)
- Public Organization
- Other:

Company size:

- Less than 10
- 10 – 20
- 21 – 40
- 41 – 100
- 100 and over

Typical group size in the projects undertaken by the company:

- Less than 5
- 5 – 10
- 10 – 20
- 20 – 40
- Over 40

Section B: The degree of knowledge and familiarity with the "Agile" methodologies

The term "Agile Methodologies" defines a family of software development methodologies, which are based on the following basic principles:

- People and interactions instead of processes and tools
- Dynamic code instead of written documentation
- Cooperation with the client instead of strict contracts
- Responding to changes instead of following a plan

The «Agile» methodology places more emphasis on what is on the left than on what is on the right.

This category of methodologies includes methodologies such as: The Scrum, XP, the Crystal Family, the Feature Driven Development (FDD), the Dynamic Systems Development Method (DSDM), the Adaptive Software Development (ASD), the Lean Development and the Unified Approach Process (Unified Process).

How long have you known about "agile" methodologies?

- For 1 year
- From 1 - 3 years
- From 3 - 5 years
- More than 5 years

How long have you been involved in projects that the apply "Agile" methodologies?

- For 1 year
- From 1 - 3 years
- From 3 - 5 years
- More than 5 years

Which Agile methodology do you usually use?

- Scrum
- eXtreme Programming (XP)
- Crystal Family
- Feature Driven Development (FDD)
- Dynamic Systems Development Method (DSDM)
- Adaptive Software Development (ASD)
- Lean Software Development
- Other:

Module C: The factors of the "agile" methodologies that contribute to the success of the software projects

We Ask you to rate the extent to which the following factors contribute positively to the success of a software development project.

To what extent is each of the following factors important to the success of a software development project?

1. Ongoing staff training:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

2. Utilization of partners with great experience and specialization:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

3. Communication among the project team members:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

4. Communication with the customers and the final users:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

5. Degree of cooperation among the participants:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

6. Simplicity in the design:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

7. Feedback in case of errors and failures:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

8. Effective project management:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

9. Availability of human resources:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

10. Availability of financial resources:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

11. Commitment, dedication and responsibility of all participants in fulfilling the objectives of the project:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

12. Technical knowledge and skills of project members:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

13. Communication skills and abilities:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

14. Ability to adapt to changes, both project-related and software-related requirements:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

15. Detailed documentation in each phase of the project development:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

16. Existence of standard project management procedures and hierarchical management structures:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

17. The requirements of the software to be developed are well known in advance and do not change much during the project:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

18. Existence of recorded detailed specifications for the software requirements:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

19. Existence of detailed design for the software:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

20. Detailed planning (design - scheduling):

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

21. Prioritize all the requirements since the beginning of the project:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

22. Ongoing feedback from the customers and the final users:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

23. Detailed control procedures (error detection and correction):

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

24. Existence of validity checks:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

25. Defining roles and responsibilities of the members since the beginning:

- Very
- Enough
- A little bit
- Not at all
- Do not know
- Contributes Negatively

What is your opinion about the results of using the "Agile" methodologies in the success of the projects and in achieving your company's goal?

- Positive
- Negative
- Other: …