ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΣΤΗ ΔΙΟΙΚΗΣΗ ΕΠΙΧΕΙΡΗΣΕΩΝ

Διπλωματική Εργασία

ΕΦΑΡΜΟΓΗ ΕΥΕΛΙΚΤΗΣ ΔΙΟΙΚΗΣΗΣ ΕΡΓΩΝ: Η ΠΕΡΙΠΤΩΣΗ ΕΤΑΙΡΕΙΑΣ ΛΟΓΙΣΜΙΚΟΥ

του

ΧΡΗΣΤΟΥ ΤΣΟΛΑΚΗ
Αριθμός Μητρώου : mba20026

ΕΠΙΒΛΕΠΟΝΤΑΣ: ΔΗΜΗΤΡΗΣ ΒΛΑΧΟΣ

Υποβλήθηκε ως απαιτούμενο για την απόκτηση του μεταπτυχιακού διπλώματος στη Διοίκηση Επιχειρήσεων

ΙΑΝΟΥΑΡΙΟΣ 2023

ΠΕΡΙΛΗΨΗ

Πολλοί οργανισμοί πιστεύουν ότι η αλλαγή δοκιμασμένων μεθόδων για την αύξηση της ταχύτητας παραγωγής και της ευελιξίας δεν συμβαδίζει με την εκπλήρωση των πιθανών κανόνων και απαιτήσεων που επιβάλει ο κλάδος τους. Με το λογισμικό να γίνεται όλο και πιο σημαντικό στην ιατρική με την πάροδο του χρόνου, οι οργανισμοί πρέπει να επανεξετάσουν τον τρόπο με τον οποίο προσεγγίζουν την ανάπτυξη ιατρικών συσκευών και του λογισμικού τους προκειμένου να ανταποκριθούν στις απαιτήσεις της αγοράς και τις ανάγκες των πελατών σε κλίμακα - και να συμβαδίσουν με τον ανταγωνισμό. Ένας από αυτούς τους τρόπους είναι χρησιμοποιώντας ευέλικτες μεθόδους, που μπορούν να επιταχύνουν την παραγωγή και να αυξήσουν την παραγωγικότητά τους. Αντικείμενο της παρούσας έρευνας, είναι να παρουσιαστεί η περίπτωση μιας εταιρείας ανάπτυξης λογισμικού για ιατρικά μηχανήματα και πως κατάφερε να μεταβεί από παραδοσιακό τρόπο ανάπτυξης (V-Model) σε ευέλικτο και συγκεκριμένα στο Scrum για όλη τη φάση της ανάπτυξης λογισμικού. Αναφερόμενοι σε όλες τις ιδιαιτερότητες που υπάρχουν λόγω συμμόρφωσης με κανονισμούς και πως αυτές ενσωματώθηκαν στον ευέλικτο τρόπο ανάπτυξης, θα αξιολογήσουμε τα προβλήματα που λύθηκαν αναφέροντας το τρόπο τους αφού πρώτα παραθέσουμε τα αποτελέσματα που συλλέχθηκαν από τους υπαλλήλους της εταιρείας μέσω αναδρομικών συναντήσεων σε διάστημα τριών ετών.

Λέξεις-κλειδιά: **λογισμικό, ευέλικτη ανάπτυξη, ευέλικτη διοίκηση έργου, ιατρικά μηχανήματα, μοντέλο καταρράκτη, scrum, v-model**

ABSTRACT

Many organizations believe that changing tried and tested methods to increase production speed and flexibility does not go hand in hand with meeting the possible regulations and requirements imposed by their industry. With software becoming increasingly important in medical field over time, organizations need to rethink how they approach the development of medical devices and their software to meet market demands and customer needs at scale – and keep up with the competition. One way is by using agile methods, which can accelerate production and increase their productivity. The objective of this research is to present the case of a software development company for medical devices and how it managed to move from a traditional way of development (V-Model) to an agile one and specifically to Scrum and use it for the whole phase of software development. After referring to all needs that exist due to compliance with regulations and how these were integrated into the agile method of development, we will evaluate the problems solved by indicating how this achieved after listing of course the results collected by the company's employees through retrospective meetings which occurred at the end of each software release at the course of over three years.

Keywords: **software, agile, project management, medical devices, waterfall, scrum, v-model**

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Κατάλογος Εικόνων

Κατάλογος Πινάκων

1. ΕΙΣΑΓΩΓΗ

Η παγκόσμια αγορά των ιατροτεχνολογικών προϊόντων αυξάνεται όλο και περισσότερο έχοντας σημαντικό αντίκτυπο στην παγκόσμια οικονομία αυτών όπως φαίνεται και από την αξία τους στην αγορά που από τα $495.46 δισ. το 2022 προβλέπεται να φτάσει στα $718.92 δισ. μέχρι το 2029 (*Medical Devices Market Size, Share & Growth | Report [2029],* n.d.).

Ένα βασικό χαρακτηριστικό πολλών ιατρικών συσκευών είναι αυτό των ενσωματωμένων συστημάτων λογισμικού (embedded software systems). Ουσιαστικά, τέτοια συστήματα είναι μηχανογραφικά συστήματα που είναι μοναδικά καθώς έχουν σχεδιαστεί για να εκτελούν συγκεκριμένες εργασίες σε συγκεκριμένη πλατφόρμα. Η πολυπλοκότητα και ο ρυθμός ανάπτυξης των λογισμικών αυτών αυξάνεται τις τελευταίες δεκαετίες. Ωστόσο, η ανάπτυξη τους προσθέτει διαφορετικές προκλήσεις στον λόγω της πολυπλοκότητάς τους.

Μερικές από αυτές περιλαμβάνουν τις έντονες εξαρτήσεις συμπεριλαμβανομένης της πλατφόρμας που θα χρησιμοποιηθεί, της εξάρτησης υλικού και λογισμικού όπως και της φύσης του προϊόντος που η λειτουργία τους απαιτεί την ομαλή συνεργασία σε πραγματικό χρόνο. Επίσης, για να υπάρχει πρόοδος συνήθως απαιτείται η συμβολή πολλών διαφορετικών ομάδων, συμπεριλαμβανομένων, για παράδειγμα, τους προγραμματιστές λογισμικού, μηχανικούς υλικού και πιθανώς ειδικούς στον ιατρικό τομέα. Η ποικιλομορφία αυτή απαιτεί έντονη διαδραστικότητα και επικοινωνία ανάμεσα στους πολλαπλούς τομείς.

Για να μπορέσουν να παρακάμψουν αυτές τις προκλήσεις, οι ομάδες συνήθως ακολουθούν μοντέλα ανάπτυξης που βασίζονται στη δημιουργία πλάνου (plan-driven) όπως το μοντέλο-V (V-model) και πρέπει να παρέχουν αποδεικτικά στοιχεία για τη διαδικασία ανάπτυξης λογισμικού τους ώστε να πάρουν προ-έγκριση για τη διάθεση το προϊόντος τους στην αγορά και αυτό γιατί πρέπει να συμμορφωθούν με τους κανονισμούς που ορίζονται από τον FDA ή τον Medical Device Directive (MDD) στην Ευρώπη (Munzner, 2003).

Ωστόσο, δημιουργήθηκε η ανάγκη για ένα καλύτερο πλαίσιο ανάπτυξης λογισμικού που θα αντιμετωπίσει την αξιοπιστία της ανάπτυξης ενός κρίσιμου λογισμικού. Οι

περισσότεροι οργανισμοί δεν υπαγορεύουν τον τρόπο με τον οποίο θα γίνει η ανάπτυξη σε χαμηλό επίπεδο (AAMI, 2012). Όλοι αυτοί οι οργανισμοί είναι περίπλοκοι και μεταβαλλόμενοι λόγω των κανονισμών που πρέπει να τροποποιούνται περιοδικά (Hugh et al., 2012).

Μια προσέγγιση που μπορεί να προσφέρει βοήθεια είναι η ανάπτυξη λογισμικού με ευέλικτες μεθόδους, όπου είναι ένα καυτό ζήτημα στα embedded software. Γενικά, οι ευέλικτες μέθοδοι συνιστούν υψηλό βαθμό συμμετοχής των πελατών, ικανότητα ενσωμάτωσης μεταβαλλόμενων απαιτήσεων όπως και σύντομους κύκλους ανάπτυξης με παραγόμενο λειτουργικό λογισμικό (Greer & Hamon, 2011).

2. ΒΙΒΛΙΟΓΡΑΦΙΚΗ ΑΝΑΣΚΟΠΗΣΗ

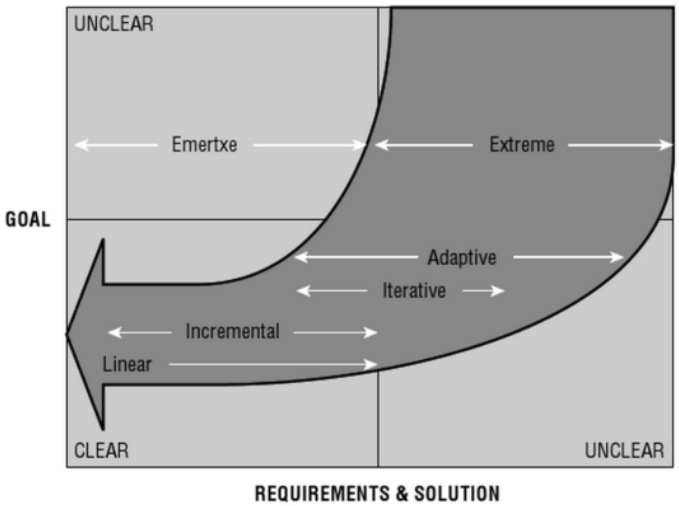
2.1 Διοίκηση έργων

Ως έργο ορίζεται μία προσωρινή δραστηριότητα που θα πρέπει να οδηγήσει σε ένα μοναδικό προϊόν, υπηρεσία ή αποτέλεσμα. Ένα έργο έχει σαφώς καθορισμένη αρχή και τέλος. Το τέλος του έργου επιτυγχάνεται όταν επιτυγχάνονται οι στόχοι του ή τερματίζεται για άλλους λόγους. (Project Management Institute, 2017b)

Η διοίκηση έργων ορίζεται ως «η εφαρμογή γνώσεων, δεξιοτήτων, εργαλείων και τεχνικών πάνω σε ένα έργο για την κάλυψη των απαιτήσεων του». Σύμφωνα με το PMBOK, η διαχείριση έργου αποτελείται από πέντε φάσεις, την έναρξη, τον προγραμματισμό, την εκτέλεση, την παρακολούθηση και τον έλεγχο και τη λήξη του. (Project Management Institute, 2017b). Στην παραδοσιακή διοίκηση έργου, αυτά εκτελούνται σειριακά ενώ στην ευέλικτη διοίκηση έργου με έναν πιο επαναληπτικό και προσαρμοστικό τρόπο. (Wysocki, 2019)

Βασιζόμενος στις πέντε φάσεις που αναφέρονται παραπάνω, ορίζονται πέντε διαφορετικά μοντέλα Διοίκησης Έργου το καθένα με το δικό του κύκλο ανάπτυξης (Project Management Life Cycle – PMLC). Τα μοντέλα αυτά είναι το γραμμικό (linear), το προσαυξητικό (incremental), το επαναληπτικό (iterative), το προσαρμοστικό (adaptive) και το ακραίο (extreme). (Wysocki, 2019)

Το γραμμικό και το προσαυξητικό εμπίπτουν στον παραδοσιακό τρόπο διοίκησης έργων, έχοντας χαμηλό επίπεδο αβεβαιότητας και σαφή στόχο. Το επαναληπτικό και προσαρμοστικό εμπίπτουν στον ευέλικτο τρόπο διοίκησης έργων, έχοντας υψηλό επίπεδο αβεβαιότητας αλλά και πάλι με σαφή στόχο. Το ακραίο μοντέλο έχει υψηλά επίπεδα αβεβαιότητας και ο στόχος του έργου είναι ασαφής. (Wysocki, 2019) Στην εικόνα 1 παρουσιάζονται τα μοντέλα ανάλογα με την σαφήνεια του στόχου και των απαιτήσεων.



Εικόνα 1 Μοντέλα διοίκησης έργου

2.2 Τρόποι διοίκησης έργου

2.2.1 Παραδοσιακός

Ο παραδοσιακός τρόπος διοίκησης έργου έχει εφαρμοστεί σε διάφορους κλάδους ανάπτυξης λογισμικού για πολλά χρόνια. Δίνεται σημαντική έμφαση στον προγραμματισμό και την προετοιμασία του έργου. Πριν την εκκίνησή του, ο προγραμματισμός των πόρων, η ενδελεχή καταγραφή της υλοποίησης, οι σαφώς συμφωνημένες απαιτήσεις αλλά και ο καθορισμένος στόχος του, πρέπει όλα να είναι ολοκληρωμένα (Wysocki, 2019).

Με το *γραμμικό* μοντέλο, κάθε διαδικασία εκτελείται σειριακά και ακριβώς μία φορά. Στο μοντέλο αυτό δεν ενθαρρύνονται οι αλλαγές του στόχου. Το προϊόν παραδίδεται εντός στόχου όπως ορίστηκε στην αρχή του έργου (Wysocki, 2019).

Το *προσαυξητικό* είναι παρόμοιο με το γραμμικό με διαφορά ότι μέρος του στόχου ορίζεται σταδιακά. Με αυτόν τον τρόπο υπάρχει καλύτερη ανατροφοδότηση και οι αλλαγές του στόχου ενθαρρύνονται για την επίτευξή του. Το βασικό πλεονέκτημα, συγκρινόμενο με το γραμμικό μοντέλο, είναι ότι το τελικό προϊόν θα βρίσκεται πιο κοντά στις ανάγκες του πελάτη (Wysocki, 2019).

Το μοντέλο-V (V-model) είναι μια παραλλαγή του γραμμικού μοντέλου που αναφέρθηκε πιο πάνω και θα αναλυθεί στο επόμενο κεφάλαιο.

2.2.2 Ευέλικτος

Το *επαναληπτικό* μοντέλο είναι μια βελτίωση του *προσαυξητικού* συμπεριλαμβάνοντας επιπλέον τον προγραμματισμό στη φάση της υλοποίησης. Σε κάθε επανάληψη δημιουργείται ένας δυνητικά παραδοτέος κώδικας (προϊόν) στον οποίο τα ενδιαφερόμενα μέρη μπορούν να δώσουν ανατροφοδότηση. Τα σχόλια είναι ευπρόσδεκτα και θεωρούνται αναπόσπαστο μέρος της διαδικασίας. Με βάση τα σχόλια και το συνολικό όραμα του προϊόντος, σχεδιάζεται, εκτελείται και αναπτύσσεται μια νέα επανάληψη. Με αυτόν τον τρόπο, η λύση δεν είναι πλήρως γνωστή στην αρχή, αλλά ορίζεται κατά τη διάρκεια του έργου (Wysocki, 2019).

Οι ευέλικτες μέθοδοι είναι τόσο προσαυξητικές όσο και επαναληπτικές. Προσαυξημένες επειδή οι εργασίες χωρίζονται σε μικρότερα κομμάτια και επαναληπτικές επειδή ο στόχος κάθε επανάληψης ορίζεται λίγο πριν από την έναρξη της. Αυτή η επαναληπτική φύση καθιστά τη διαδικασία ευέλικτη (Wysocki, 2019).

Όταν ο στόχος είναι σαφής αλλά η λύση και ο τρόπος να φτάσεις σε αυτόν ασαφή, ένα μοντέλο Ευέλικτης Διοίκησης Έργου (Agile Project Management – APM) πρέπει να χρησιμοποιηθεί. Αυτά τα είδη έργων ορίζονται ως σύνθετα και απαιτούν μη παραδοσιακές προσεγγίσεις για την επιτυχημένη εκτέλεση τους.

Στο μοντέλο Παραδοσιακής Διοίκησης Έργων (Traditional Project Management – TPM), όλες οι λεπτομέρειες του έργου είναι προγραμματισμένες εκ των προτέρων. Αυτή η προσέγγιση ονομάζεται βάσει σχεδίου (plan-driven).

Σε αντίθεση, τα APM έργα καθοδηγούνται από τις αλλαγές (change driven) (Wysocki, 2019). Αυτό σημαίνει ότι αντί να αποφεύγονται οι αλλαγές, ενθαρρύνονται. Αυτό αλλάζει τη σχέση ανάμεσα στην ομάδα ανάπτυξης του έργου και των εξωτερικών ενδιαφερόμενων μερών. Τα ενδιαφερόμενα μέρη εμπλέκονται περισσότερο στη διαδικασία και έτσι το τελικό αποτέλεσμα είναι πιο κοντά στο επιθυμητό. Από τη στιγμή που ο στόχος είναι ασαφής στην έναρξη του έργου, ορίζεται κατά τη διάρκεια του βασιζόμενο κυρίως στην ανατροφοδότηση (Rehman et al., 2010).

Οι εργασίες που χρειάζονται να γίνουν σε ένα έργο κυμαίνονται από καθορισμένες έως υψηλής αβεβαιότητας. Οι καθορισμένες εργασίες χαρακτηρίζονται από σαφείς διαδικασίες που έχουν αποδειχτεί επιτυχείς σε παρόμοια έργα στο παρελθόν. Η

παραγωγή ενός αυτοκινήτου, μιας ηλεκτρικής συσκευής ή ακόμα και ενός σπιτιού είναι παραδείγματα έργων με σαφείς διαδικασίες. Ο τρόπος παραγωγής και οι διαδικασίες του έργου, είναι συνήθως αρκετά κατανοητές και τα επίπεδα αβεβαιότητας και ρίσκου παραμένουν χαμηλά. Παραδείγματα έργων με υψηλή αβεβαιότητα συμπεριλαμβάνουν την ανάπτυξη ενός λογισμικού, τη κατασκευή νέων προϊόντων όπως και κυρίως τα περισσότερα μηχανικά έργα. Η αβεβαιότητα κάνει την εκτίμηση ενός έργου είναι δύσκολή. Τα έργα υψηλής αβεβαιότητας έχουν μεγάλα ποσοστά αλλαγών, πολυπλοκότητας και ρίσκου. Αυτά τα χαρακτηριστικά παρουσιάζουν προβλήματα στις προσεγγίσεις διοίκησης έργου που έχουν στόχο τον καθορισμό των απαιτήσεων εκ των προτέρων. Αντίθετα, οι ευέλικτες μέθοδοι δημιουργήθηκαν με σκοπό την ύπαρξη σύντομων κύκλων ανάπτυξης με γρήγορη προσαρμογή βασιζόμενη στην αξιολόγηση και την ανατροφοδότηση (Project Management Institute, 2017a).

2.3 Ευέλικτη διοίκηση έργου στην ανάπτυξη λογισμικού

Η εκθετική πρόοδος και εξέλιξη της τεχνολογίας, διαταράσσει συνεχώς τα έργα αλλά και τις ομάδες τους, λόγω των απαιτήσεων για την αμεσότερη αύξηση αξίας. Οι ευέλικτες προσεγγίσεις και τεχνικές, καταφέρνουν αποτελεσματικά να διαχειριστούν αυτές τις αλλαγές. Αναπτύχθηκαν για να παρέχουν μεγαλύτερη ικανοποίηση στους πελάτες, να συντομεύσουν τον κύκλο ζωής της ανάπτυξης, να μειώσουν τα ποσοστά σφάλματος και να ικανοποιήσουν τις μεταβαλλόμενες απαιτήσεις κατά τη διάρκεια της ανάπτυξης (Cho, 2008). Ο γρήγορος και με διαφάνεια τρόπος που οι πελάτες παραθέτουν τις εμπειρίες και τα σχόλια τους με την άνοδο των κοινωνικών δικτύων, αναγκάζει τους οργανισμούς να επικεντρωθούν φαινομενικά στην εξυπηρέτηση πελατών ώστε να παραμείνουν ανταγωνιστικοί (Project Management Institute, 2017a).

Όσο παλιότερος είναι ένας οργανισμός, είναι όλο και πιο επιρρεπείς στο να είναι υψηλής περιπλοκότητας και πιθανότητα αργός στην καινοτομία, έχοντας ως αποτέλεσμα να υπολείπεται σε ανταγωνιστικότητα και να καθυστερεί στην παράδοση νέων λύσεων στους πελάτες τους. Αυτοί οι οργανισμοί, ανταγωνίζονται με μικρότερους αλλά και με startups που μπορούν να παράγουν γρηγορότερα προϊόντα που ταιριάζουν στις ανάγκες των καταναλωτών. Αυτή η ταχύτητα με την οποία χρειάζεται να γίνονται οι αλλαγές, θα συνεχίσει να οδηγεί τους μεγάλους οργανισμούς να υιοθετούν ένα ευέλικτο τρόπο σκέψης ώστε να μπορούν να μένουν ανταγωνιστικοί και να μην μειώνεται το μερίδιο αγοράς τους (Project Management Institute, 2017a).

Οι ευέλικτες μεθοδολογίες είναι γνωστές για την έμφαση που δίνουν στην επικοινωνία αλλά και την ενεργή συμμετοχή των πελατών στην υλοποίηση του έργου (Petersen & Wohlin, 2009). Αυτό συνεπάγεται ότι οι διαπροσωπικές και κοινωνικές δεξιότητες είναι κρίσιμες για ολόκληρη την ομάδα των προγραμματιστών ώστε σε κάθε επανάληψη να μπορούν να παραδώσουν ολοκληρωμένες υλοποιήσεις επικοινωνώντας έτσι την πρόοδο και όποια προβλήματα αντιμετωπίζουν (Leau et al., 2012).

2.4 Ευέλικτες μεθοδολογίες

Διαφορετικές μέθοδοι έχουν και διαφορετικές πρακτικές. Κατά την επιλογή μιας μεθόδου για ένα συγκεκριμένο έργο, αυτές οι πρακτικές θα πρέπει να λαμβάνονται υπόψη προκειμένου να διασφαλιστεί ότι ταιριάζουν με το έργο. Δεν περιγράφουν όλες οι μέθοδοι πολλές πρακτικές, αλλά επικεντρώνονται σε άλλες πτυχές. Το Scrum περιγράφει πρακτικές με τη μορφή *γεγονότων (*events*), ρόλων* (roles) και *τεχνουργημάτων (*artifacts*)* (Schwaber & Sutherland, 2020). Ο ακραίος προγραμματισμός (Extreme Programming - XP) περιγράφει 5 βασικές αξίες, 15 αρχές, 13 πρωτογενείς πρακτικές και 11 επακόλουθες πρακτικές (Beck & Andres, 2004). Το Kanban περιγράφει πέντε αρχές όπως τις διατυπώνει ο (Griffiths & Project Management Institute., 2012) 1) Οπτικοποίηση της ροής εργασιών, 2) Περιορισμός των εργασιών σε εξέλιξη, 3) Διαχείριση της ροής, 4) Σαφή διαδικασίες και 5) Βελτίωση της συνεργατικότητας, που θα μπορούσαν επίσης να θεωρηθούν πρακτικές. Μια μέθοδος μπορεί να θεωρηθεί ως μια συλλογή βέλτιστων πρακτικών, αξιών ή/και αρχών, η οποία έχει αποδειχθεί ότι λειτουργεί για ορισμένους τύπους έργων. Παρακάτω θα αναλύσουμε το Scrum διότι είναι η μεθοδολογία που επιλέχθηκε από την εταιρεία που θα μελετήσουμε στο επόμενο κεφάλαιο.

2.5 Scrum

Σύμφωνα με τον οδηγό του Scrum (Schwaber & Sutherland, 2020), το Scrum είναι ένα επαναληπτικό και επαυξητικό πλαίσιο κανόνων, ιδεών και αξιών, εντός του οποίου οι άνθρωποι μπορούν να αντιμετωπίσουν σύνθετα προβλήματα, παρέχοντας προϊόντα με την υψηλότερη δυνατή αξία που δημιουργήθηκε ειδικά για τη διαχείριση έργων λογισμικού. Ορίστηκε γύρω στο 1995, αλλά πραγματικά αυξήθηκε η δημοτικότητά του μετά το 2001 όταν ιδρύθηκε η Ευέλικτη Συμμαχία (Agile Alliance).

Το Scrum είναι:

- Ελαφρύ
- Απλό στην κατανόηση
- Δύσκολο να το κυριαρχήσει κάποιος

Στο Scrum ορίζονται μόνο τρεις ρόλοι. Ο Product Owner, ο Scrum Master και η ομάδα. Ο Product owner ελέγχει τη λίστα των εκκρεμοτήτων του προϊόντος (product backlog) και καθορίζει και προτεραιοποιεί τις απαιτήσεις. Ο Scrum Master διασφαλίζει ότι ακολουθείται η διαδικασία Scrum και επιλύει όποια εμπόδια εμφανιστούν στην ομάδα προσπαθώντας να την κάνει πιο αποδοτική. Η ομάδα είναι τα μέλη που την απαρτίζουν όπως προγραμματιστές, γραφίστες κ.λπ (Schwaber & Sutherland, 2020).

Το Scrum ορίζει τέσσερεις συναντήσεις για επιθεώρηση και προσαρμογή κατά τη διάρκεια μιας επανάληψης του, το ονομαζόμενο sprint. Οι συναντήσεις αυτές βασίζονται στους πυλώνες της διαφάνειας, επιθεώρησης και προσαρμογής. Διαφάνεια σημαίνει πως οι προκύπτοντες διαδικασίες και δουλειά πρέπει να είναι ορατές σε αυτούς που εκτελούν τη δουλειά και σε αυτούς που τη λαμβάνουν. Συχνή επιθεώρηση και εντοπισμός πιθανόν προβλημάτων που εμποδίζουν την πρόοδο του συμφωνηθέντος στόχου. Προσαρμογή ως προς όποια απόκλιση σε διαδικασίες ή στην ποιότητα του προϊόντος σε σχέση με αυτή που έχει συμφωνηθεί. Αυτά να αναγνωρίζονται και να υφίσταται προσαρμογή όσο πιο σύντομα γίνεται ώστε να μειωθεί η απόκλιση όσο νωρίτερα γίνεται (Schwaber & Sutherland, 2020).

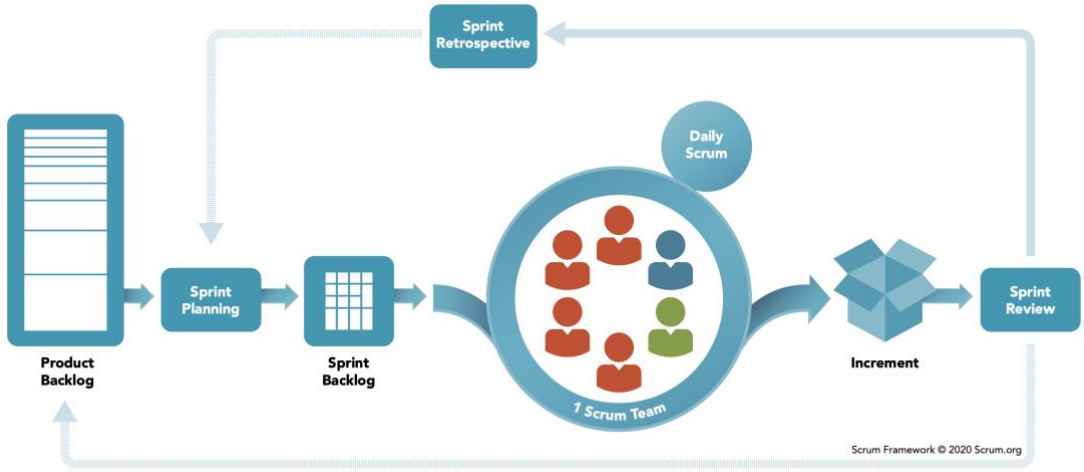
Το Scrum βασίζεται σε 5 αρχές Δέσμευση (Commitment), Εστίαση (Focus), Ανοικτότητα (Openness), Σεβασμό (Respect) και Θάρρος (Courage).

- Δέσμευση να επιτύχουν τους στόχους
- Εστίαση στους στόχους που ορίζει η ομάδα
- Ανοικτότητα μεταξύ της ομάδας και προς τα ενδιαφερόμενα μέρη για τη δουλειά και τις προκλήσεις που έρχονται αντιμέτωποι
- Σεβασμός ανάμεσα στα μέλη
- Θάρρος να κάνουν το σωστό και να δουλεύουν στην επίλυση προβλημάτων

Τα πιο σημαντικά αντικείμενα είναι η λίστα των εκκρεμοτήτων του προϊόντος (product backlog) που θα υλοποιηθούν στο μέλλον, η λίστα των εκκρεμοτήτων της επανάληψης (sprint backlog) που δουλεύονται τη συγκεκριμένη στιγμή και το κομμάτι του προϊόντος

(product increment) που είναι το αποτέλεσμα της δουλειάς που ολοκληρώθηκε κατά τη διάρκεια του sprint (Schwaber & Sutherland, 2020).

Η καρδιά του Scrum είναι το Sprint, που είναι ένα χρονικό διάστημα ενός μήνα ή λιγότερο κατά τη διάρκεια του οποίου δημιουργείται ένα ολοκληρωμένο, έτοιμο να χρησιμοποιηθεί και προς παράδοση λογισμικό. Τα Sprint έχουν σταθερή διάρκειες καθ' όλη τη διάρκεια μιας αναπτυξιακής προσπάθειας. Ένα νέο Sprint ξεκινά αμέσως μετά την ολοκλήρωση του προηγούμενου (Schwaber & Sutherland, 2020).



Εικόνα 2 Scrum

2.5.1 Ρόλοι στο Scrum

1. Ο **Scrum Master**, διευκολύνει τη διαδικασία υλοποίησης του Scrum. Το συγκεκριμένο άτομο πρέπει να έχει καλή γνώση του συστήματος που υλοποιείται και καλή κατανόηση των προτύπων ασφαλείας.
2. **Product Owner**, έχει πολύ κατανόηση της αγοράς και πρέπει να εκπροσωπεί τον πελάτη και να λαμβάνει αποφάσεις για λογαριασμό του. Είναι το υπεύθυνο πρόσωπο για τις απαιτήσεις του συστήματος.
3. Η **ομάδα Scrum** αποτελείται από τους μηχανικούς, άτομα που εκτελούν την επαλήθευση του προϊόντος, τον γραφίστα και τα άτομα υπεύθυνα για τα έγγραφα. Στην δική μας περίπτωση, ο Scrum Master είναι υπεύθυνος για τη δημιουργία των εγγράφων μιας και έχει εμπειρία από τον προηγούμενό του ρόλο.

2.5.2 Συναντήσεις (Events)

Οι συναντήσεις που καθορίζει το Scrum είναι ο σχεδιασμός (planning), η ημερήσια συνάντηση (daily scrum), η ανασκόπηση (sprint review) και η αναθεώρηση (retrospective). Κάθε συνάντηση είναι συγκεκριμένου χρόνου και σκοπό έχει τη γρήγορη πρόοδο και συνεχόμενη μεταφορά της πληροφορίας ανάμεσα σε όλα τα μέλη και τα ενδιαφερόμενα μέρη. Κάθε sprint διαρκεί από 2-4 εβδομάδες όπου η ομάδα δουλεύει σε συγκεκριμένες εργασίες.

2.5.2.1 Sprint Planning

Για να ξεκινήσει η ανάπτυξη, χρειάζεται να ξεκινήσει το sprint και αυτό σε μία συνάντηση που συμμετέχουν οι προγραμματιστές με τον Product Owner, οργανώνουν το sprint με τα κομμάτια δουλειάς που θα ασχοληθούν. Η συγκεκριμένη συνάντηση διαρκεί για ένα sprint 2 εβδομάδων μέγιστο 4 ώρες και πρέπει να καλυφθούν τρεις πτυχές.

Η πρώτη είναι ο στόχος του sprint που καθορίζεται από την ομάδα μαζί με τον Product Owner και καθορίζει το σκοπό του sprint με ένα συγκεκριμένο και μετρήσιμο τρόπο.

Έπειτα συζητάτε η διαθεσιμότητα και η δέσμευση της ομάδας. Εδώ υπολογίζονται τυχόν απουσίες, εκπαιδεύσεις κλπ., για να μπορέσουν να προβλέψουν και να δεσμευτούν στη δουλειά που μπορούν να ολοκληρώσουν.

Το τρίτο βήμα είναι η δημιουργία της λίστας των εργασιών που θα εργαστούν. Ο τρόπος επιλογής βασίζεται στη προτεραιότητα που έχει οριστεί για κάθε εργασία από τον Product Owner ανάλογα με την αξία που προσφέρουν στον πελάτη και στο προϊόν.

2.5.2.2 Daily Scrum

Καθημερινά γίνεται μία συνάντηση όλης της ομάδας για 15 λεπτά όπου ο κάθε προγραμματιστής αναφέρει με τι ασχολήθηκε εχθές, με τι θα ασχοληθεί σήμερα και αν αντιμετωπίζει κάποιο πρόβλημα. Τα υπόλοιπα μέλη μπορούν να ρωτήσουν ερωτήσεις για να καταλάβουν καλύτερα τα προβλήματα. Αν χρειαστεί εις βάθος ανάλυση, θα πρέπει να οργανωθεί μία νέα συνάντηση με στόχο την συζήτηση και επίλυση του προβλήματος. Με αυτή της σύντομη συνάντηση επιτυγχάνεται διαφάνεια ανάμεσα στη ομάδα έχοντας όλοι την τελευταία πληροφορία και βοηθάει ώστε να λυθούν τα προβλήματα αμέσως μόλις αναγνωρισθούν.

2.5.2.3 Sprint Review

Στο τέλος κάθε sprint γίνεται και η κριτική του. Είναι μια συνάντηση που είναι παρόντες η ομάδα και τα ενδιαφερόμενα μέρη και παρουσιάζεται η δουλειά που έχει ολοκληρωθεί. Αν υπάρχουν ποιοτικά προβλήματα ή κίνδυνοι, συζητιόνται και αποφασίζεσαι αν είναι αποδεκτά ή όχι ανάλογα την ανάλυση τους. Η συγκεκριμένη συνάντηση διαρκεί μέγιστο δύο ώρες.

2.5.2.4 Sprint Retrospective

Ανάμεσα στα sprint, στο τέλος του ενός και πριν την έναρξη του επόμενου, γίνεται μια ανασκόπηση. Σε αυτή τη συνάντηση συζητιόνται πιθανά προβλήματα που αντιμετώπισε η ομάδα ώστε να βελτιωθούν οι διαδικασίες και η ποιότητα του παραγόμενου. Πιθανά θέματα συζήτησης μπορεί να είναι εργασιακά, αλληλεπίδρασης των ατόμων, γύρω από εργαλεία, διαδικασίες και στο σύνολο τι πήγε καλά, τι χρήζει βελτίωσης και πως μπορεί να επιτευχθεί αυτή η βελτίωση. Η συνάντηση αυτή διαρκεί μέγιστο 1.5 ώρα.

Ο τρόπος που λειτουργούν όλα μεταξύ τους είναι ο εξής. Στην αρχή κάθε sprint, κατά τη διάρκεια του sprint planning, η ομάδα με τον Product Owner αποφασίζει τα κομμάτια δουλειάς που – ένα τμήμα του product backlog – που θα εργαστεί μέσα στο sprint. Κατά τη διάρκεια του sprint, στην καθημερινή συνάντηση συζητούν αν προχωράνε προς τον στόχο. Στο τέλος του sprint, στο sprint review, παρουσιάζουν στα ενδιαφερόμενα μέρη το ολοκληρωμένο κομμάτι δουλειάς που έχουν παράξει και το sprint ολοκληρώνεται με τη συνάντηση του retrospective όπου καταγράφουν τι πήγε καλά και σε ποια σημεία μπορούν να εστιάσουν για να βελτιώσουν σε επόμενο sprint.

2.6 Υβριδικοί Μέθοδοι

Τα τελευταία χρόνια, οι ευέλικτες μέθοδοι έχουν κερδίσει δημοτικότητα έναντι των παραδοσιακών μεθόδων, αφήνοντας τις επίσημες διαδικασίες υπερτερώντας οι ανεπίσημες συναντήσεις και επικοινωνίες για τη διαχείριση έργων. Αυτό είχε ως αποτέλεσμα η μηχανική απαιτήσεων, η αρχιτεκτονική του λογισμικού, η διασφάλιση ποιότητας και η διαχείριση κινδύνων να γίνονται ανεπίσημα ή ακόμα και να παραλείπονται εντελώς. Το σκεπτικό ήταν ότι αυτό θα εμπόδιζε τον ευέλικτο τρόπο εργασίας. Ωστόσο, οι εταιρείες που εργάζονταν με παραδοσιακό τρόπο, μεταβαίνοντας

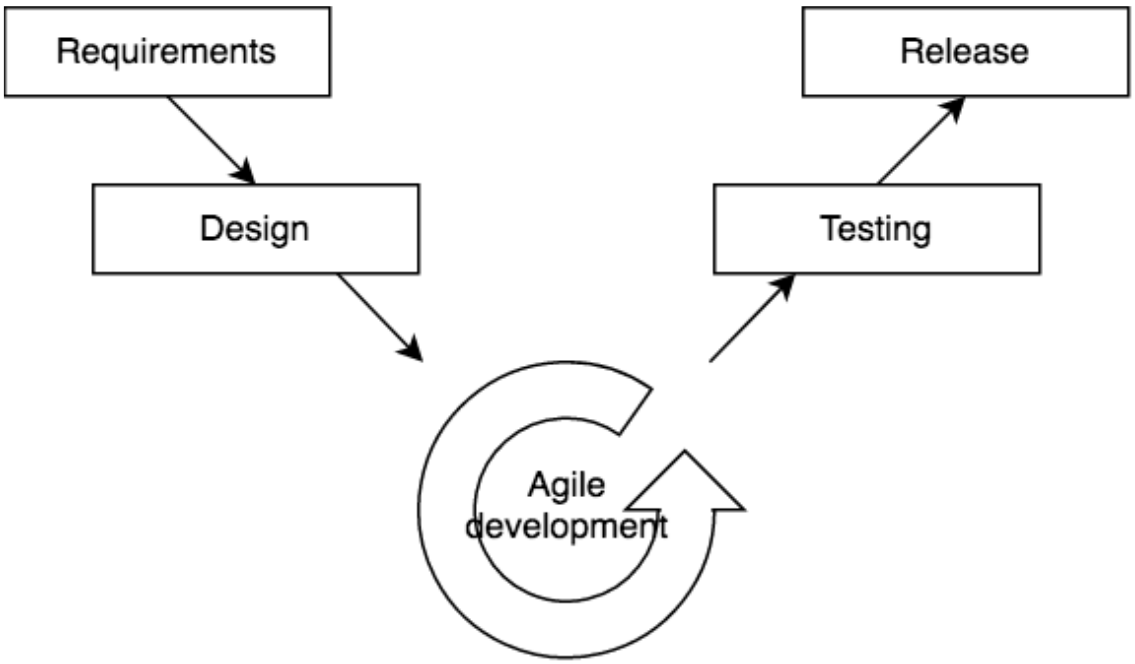
σε ευέλικτο, ή που δεσμεύονται από τήρηση κανονισμών, εφάρμοσαν στην πραγματικότητα μια υβριδική μορφή "water-scrum-fall", όπως έθεσε ο (West, 2011).

Με την τρέχουσα εμπειρία που οι ερευνητές και οι εταιρείες δουλεύουν με ευέλικτους τρόπους, προκύπτουν πολλές υβριδικές λύσεις που γεμίζουν τα κενά των τυπικών διαδικασιών στις ευέλικτες. Οι (McHugh et al., 2013) αναφέρουν ότι θα πρέπει να εφαρμόζονται μικτές μέθοδοι προκειμένου να επιτευχθεί η μέγιστη επιρροή στα έργα. Οι (Kuhrmann et al., 2017) παρατηρούν στη μελέτη τους ότι πράγματι ένας συνδυασμός παραδοσιακών και ευέλικτων μεθόδων έχει γίνει πραγματικότητα.

Αυτή η μέθοδος προέκυψε από τις διαφορετικές ανάγκες που έχουν οι προϊστάμενοι και οι προγραμματιστές: οι ευέλικτες μέθοδοι είναι συχνά προσαρμοσμένες για καθήκοντα που σχετίζονται με την ανάπτυξη, ενώ λείπουν οι διαδικασίες που απαιτούνται από τη διοίκηση, όπως η εκτίμηση χρονοδιαγράμματος και ο σχεδιασμός (Theocharis et al., 2015).

Στις ευέλικτες μεθόδους στην αρχιτεκτονική δίνεται λιγότερη έμφαση (Hanssen et al., 2018). Ενώ η πεποίθηση ήταν ότι η αρχιτεκτονική θα ακολουθούσε τη δουλειά που γίνεται στα σπριντ, πολλές ευέλικτες μέθοδοι σήμερα προτιμούν να ορίζουν την αρχιτεκτονική εκ των προτέρων.

Συνήθως ο συνδυασμός της ανάλυσης των απαιτήσεων, σχεδιασμού και αρχιτεκτονικής ακολουθούν τον παραδοσιακό τρόπο (το “water” κομμάτι) και η ανάπτυξη γίνεται βασισμένη σε κάποια ευέλικτη μέθοδο όπως το Scrum και η επαλήθευση και κυκλοφορία του λογισμικού γίνονται μετά την υλοποίηση (το “fall” κομμάτι) (West, 2011). Ένα παράδειγμα φαίνεται στην εικόνα 2.



Εικόνα 3 Υβριδικό μοντέλο

3. ΕΡΕΥΝΗΤΙΚΗ ΜΕΘΟΔΟΛΟΓΙΑ

3.1 Ερευνητική μέθοδος

Η μελέτη περίπτωσης ήταν ένας από τους πρώτους τύπους έρευνας που χρησιμοποιήθηκαν στον τομέα της ποιοτικής μεθοδολογίας. Έχουν χρησιμοποιηθεί σε μεγάλο βαθμό στις κοινωνικές επιστήμες και έχουν διαπιστώθηκε ότι είναι ιδιαίτερα πολύτιμες σε τομείς προσανατολισμένους στην πράξη (όπως η εκπαίδευση, διοίκηση και κοινωνική εργασία). Σήμερα, αντιπροσωπεύουν ένα μεγάλο μέρος της έρευνας που παρουσιάζονται σε βιβλία και άρθρα στην ψυχολογία, την ιστορία, την εκπαίδευση και την ιατρική, για να απαριθμήσω μόνο μερικές από τις θεμελιώδεις επιστήμες. Πολλά από αυτά που γνωρίζουμε σήμερα έχουν παραχθεί από την έρευνα μελέτης περιπτώσεων (Denzin & Lincoln, 2017).

Για την παρούσα μελέτη περίπτωσης θα χρησιμοποιηθεί ο τύπος της μοναδικής μελέτης περίπτωσης (single case study) και συγκεκριμένα με την χρονική μορφή της αναδρομής (retrospective) που εμπλέκετε η συλλογή δεδομένων και διαδικασιών μετά από την πάροδο μιας κατάστασης (Thomas, 2011).

Επιλέχθηκε ο συγκεκριμένος τρόπος διότι ο περιπτώσεις μελέτης είναι γενικά ισχυρές εκεί που οι ποσοτικές είναι ασθενέστερες. Τέσσερα από τα πλεονεκτήματα που έχουν εντοπίσει οι (George & Bennett, 2005) σε σύγκριση με τις ποσοτικές μεθόδους είναι: Η δυνατότητά τους να επιτύχουν υψηλή εννοιολογική εγκυρότητα, ισχυρές διαδικασίες για την εύνοια νέων υποθέσεων, χρησιμότητα για την στενή εξέταση του υποθετικού ρόλου των αιτιωδών μηχανισμών στο πλαίσιο των μοναδικών περιπτώσεων και τέλος την ικανότητά τους να αντιμετωπίζουν την αιτιώδη πολυπλοκότητα.

3.2 Διαδικασία συλλογής δεδομένων

Αρχικά θα παρουσιαστούν αναλυτικά όλα τα βήματα που ακολουθήθηκαν για τη μετάβαση του τρόπου λειτουργίας από παραδοσιακό σε ευέλικτο. Έπειτα, θα χρησιμοποιηθούν μια σειρά από ιστορικά δεδομένα από τους υπαλλήλους της εταιρείας και συγκεκριμένα από τους προγραμματιστές, τον Επικεφαλής Τεχνολογίας (CTO), τον υπεύθυνο του προγράμματος (Program Manager), τον Διευθυντή των Ρυθμιστικών Κανόνων (Director of Regulatory), την Διευθύντρια Διασφάλιση Ποιότητας (Director of Quality Assurance) και κάποιων υπαλλήλων από την υποστήριξη πελατών.

Τα δεδομένα θα συλλεχθούν από τις συναντήσεις ανατροφοδότησης που γίνονται στο τέλος κάθε έργου και είναι μια διαδικασία που ακολουθάει η εταιρεία από την πρώτη στιγμή για όλα τα τμήματα και όχι μόνο για την ανάπτυξη λογισμικού.

4. Η ΠΕΡΙΠΤΩΣΗ ΕΤΑΙΡΕΙΑΣ ΛΟΓΙΣΜΙΚΟΥ

4.1 Εισαγωγή

Ένας οργανισμός θεωρείτε ότι βρίσκεται σε «πτωτική» κατάσταση όταν οι οικονομικές του επιδόσεις πλήττονται για αυξημένη χρονική περίοδο σε βαθμό που η απόδοση του να φθίνει σε επίπεδο που τείνει να θέσει σε κίνδυνο τη βιωσιμότητά του (Cameron et al., 1987). Αντίθετα, μια κατάσταση ανάκαμψης είναι αυτή όπου μια εταιρεία που αντιμετωπίζει τα αίτια της παρακμής της υιοθετεί στρατηγικές που αντιστρέφουν την πορεία της πέρα από την επιβίωση και τελικά οδηγούν σε σταθερή κερδοφορία. (Barker & Duhaime, 1997). Οι (Lohrke et al., 2004) αναλύοντας τη διαδικασία ανάκαμψης, απεικονίζουν τρεις σχετικές φάσεις. Στην πρώτη φάση, ένας προβληματικός οργανισμός χρειάζεται να εντοπίσει τους κύριου λόγους που οδηγούν στη πτώση του, που συνήθως ορίζονται σε «αλλαγές στο περιβάλλον, εσωτερικές ελλείψεις, ή σε συνδυασμό και των δύο». Στη δεύτερη φάση, η διοίκηση της εταιρείας διατυπώνει και υλοποιεί τις κατάλληλες στρατηγικές για την αντιμετώπιση των πηγών που οδηγούν στην παρακμή. Τέλος, στη τρίτη φάση, αξιολογείται το αποτέλεσμα ανάκαμψης της εταιρείας που οδηγεί στη βελτίωση ή την αποτυχία της.

Εδώ θα παρουσιάσουμε πως ένας οργανισμός ύστερα από κάποια γεγονότα αποφάσισε να βελτιώσει τον τρόπο λειτουργίας τους με σημαντικά αποτελέσματα, αλλάζοντας τον τρόπο λειτουργίας του.

4.2 Η εταιρεία

Η εταιρεία που θα αναλυθεί είναι μια εταιρεία ανάπτυξης ιατρικών μηχανημάτων που ιδρύθηκε στην Αυστραλία το 1999 με γραφεία στην Αμερική και στη Θεσσαλονίκη. Αναπτύσσει συσκευές βιοεμπέδησης φασματοσκοπίας (bioimpedance spectroscopy - BIS) με έμφαση σε ιατρικές εφαρμογές για χρήση σε διάφορους τομείς υγείας. Έχει συνολικά ανθρώπινο δυναμικό περίπου 70 άτομα όπου τα 15 από αυτά εργάζονται στο κομμάτι της ανάπτυξης λογισμικού.

Η πρώτη της συσκευή χρησιμοποιούσε την BIS τεχνολογία για ετερόπλευρη ανάλυση πιθανού λεμφοιδήματος σε ασθενή. Είναι η πρώτη συσκευή που έχει λάβει έγκριση από τον Οργανισμό Τροφίμων και Φαρμάκων της Αμερικής (FDA) για την αξιολόγηση πιθανού λεμφοιδήματος σε πρώιμο στάδιο πριν την οπτική παραμόρφωση των άκρων του ασθενή. Το συγκεκριμένο μηχάνημα προϋπέθετε την ακινησία του ασθενή για αρκετά

λεπτά έχοντας και την επιπλέον δυσκολία σύνδεσης του με τον ασθενή. Λειτουργούσε με λογισμικό που έτρεχε μέσα στη συσκευή. Αποσύρθηκε από την αγορά το 2018 και αντικαταστάθηκε από της επόμενης γενιάς για την οποία αναπτύχθηκε και το λογισμικό για το οποίο θα αναλύσουν τον τρόπο ανάπτυξης του παρακάτω.

Η συνεχής εξέλιξη της τεχνολογίας στις καθημερινές συσκευές και ο εκσυγχρονισμός αυτών με τη διασύνδεσή τους στο διαδίκτυο (IoT) ώθησε σιγά σιγά και τα ιατρικά μηχανήματα προς την συγκεκριμένη κατεύθυνση. Η τελευταία συσκευή της εταιρείας χρησιμοποιεί tablet για την αλληλεπίδραση με αυτή μέσω μίας ειδικά ανεπτυγμένης εφαρμογής.

Στην συγκεκριμένη περίπτωση χρήσης θα αναπτύξουμε τις μεθόδους που χρησιμοποιήθηκαν για την ανάπτυξη αυτού του λογισμικού από την ομάδα ανάπτυξης λογισμικού της εταιρείας. Είναι η πρώτη φορά που γίνεται εσωτερικά η ανάπτυξη και όχι από κάποια εξωτερική συνεργασία που ήταν και ο μοναδικός τρόπος ανάπτυξης και συντήρησης του λογισμικού για τις προηγούμενες συσκευές.

Η δοκιμασία που έπρεπε να αντιμετωπίσει ήταν αρκετά μεγάλη μιας και από εταιρεία ανάπτυξης ιατρικών μηχανημάτων έπρεπε να μεταμορφωθεί σε εταιρεία τεχνολογίας.
Θα αναλύσουμε συγκεκριμένα το τμήμα της ανάπτυξης λογισμικού για το τελευταίο της προϊόν και την εξέλιξη της μέσω ευέλικτων μεθοδολογιών που χρησιμοποιήθηκαν και συγκεκριμένα του Scrum framework.

Τον Σεπτέμβριο του 2016 δημιουργήθηκε η ομάδα ανάπτυξης λογισμικού στην Ελλάδα. Μέχρι τον Μάιο του 2017 θα έπρεπε να υπάρχει η πρώτη λειτουργική έκδοση της εφαρμογής που θα έλεγχε τη νέα μηχανή.

Δεδομένου του περιορισμένου χρονικού διαστήματος, η ομάδα ξεκίνησε να λειτουργεί με τις υπάρχουσες διαδικασίες που ήταν γνωστές στην εταιρεία μέχρι εκείνη τη στιγμή και τις χρησιμοποιούσαν με τις εξωτερικές εταιρείες που συνεργάζονταν. Για την ανάπτυξη λογισμικού χρησιμοποιούσαν το V-μοντέλο (V-Model) που είναι και το πιο διαδεδομένο στην ανάπτυξη Κρίσιμης Ασφαλείας λογισμικού όπως αυτό ενός ιατρικού μηχανήματος.

4.3 Που βρισκόταν

Ένα πλάνο ανάπτυξης λογισμικού (Software Development Plan) που καταγράφονται όλες τις διαδικασίες που πρέπει να ακολουθηθούν κατά τη διάρκεια της ανάπτυξης είναι απαραίτητο να δημιουργηθεί για κάθε έργο ανάπτυξης λογισμικού. Σε αυτό θα πρέπει να συμπεριλαμβάνεται και ο κύκλος ανάπτυξης του λογισμικού (Software Development Life Cycle – SDLC) που καθορίζει τις δραστηριότητες που εμπλέκονται στη διαδικασία ανάπτυξης, την σειρά που εκτελούνται αλλά και τις εξαρτήσεις αυτών των δραστηριοτήτων, προσδιορίζοντας τα ορόσημα (milestones) και τα παραδοτέα που σχετίζονται με αυτά τα ορόσημα.

Στην εταιρεία χρησιμοποιούταν το μοντέλο-V (V-Model) που είναι η επέκταση του μοντέλου καταρράκτη (Waterfall) και η ανάπτυξη ενός προϊόντος γίνεται κάτω από 6 φάσεις οι οποίες είναι:

- Φάση 1 – Σχεδιασμός & Προγραμματισμός Ανάπτυξης
- Φάση 2 – Εισαγωγή σχεδιασμού
- Φάση 3 – Παραγωγή σχεδιασμού
- Φάση 4 – Επαλήθευση σχεδιασμού
- Φάση 5 – Επικύρωση σχεδιασμού
- Φάση 6 – Μεταφορά σχεδιασμού

Φάση 1

Στη πρώτη φάση, καταγράφεται το πλάνο που θα ακολουθηθεί για την ανάπτυξη του νέου προϊόντος. Το πλάνο περιλαμβάνει όλες τις κύριες δραστηριότητες που πρέπει να εκτελεστούν (Gantt chart), το οργανόγραμμα με όλους τους αρμόδιους, το κόστος, τα πιθανά ορόσημα (milestones), ανάλυση ρίσκου για το έργο, όπως και τον τρόπο που θα γίνει η επαλήθευση του έργου πριν την παράδοση του στο τέλος.

Φάση 2

Στη δεύτερη φάση καθιερώνονται οι απαιτήσεις που χρειάζονται για την ανάπτυξη του προϊόντος. Μία καλή δομή και βάση από απαιτήσεις (requirements) είναι το πιο σημαντικό που γίνεται στη φάση του σχεδιασμού, καθώς αυτές θα χρησιμοποιηθούν για την επικύρωση του παραδοτέου στο τέλος. Οι απαιτήσεις και οι προδιάγραφες που καταγράφονται περιλαμβάνουν απαιτήσεις Μάρκετινγκ, Συστήματος, Λογισμικού, Αρχιτεκτονικής Λογισμικού, Υλικού όπως και Λειτουργικές, Απόδοσης και Ασφαλείας.

Σχόλια/παράπονα από πελάτες, πληροφορίες από προηγούμενα έργα όπως και από ανταγωνιστές, λαμβάνονται υπόψιν κατά την καθιέρωση των απαιτήσεων.

Ακόμα, καθορίζονται τα πλάνα για τα επίπεδα ποιότητας που πρέπει να ακολουθηθούν όπως και με όλους τους κανόνες και τις προδιαγραφές που θα πρέπει να πληρούνται ανάλογα την αγορά στην οποία θα διατεθεί το προϊόν (π.χ. FDA, CE κλπ.).

Φάση 3

Στην τρίτη φάση, δημιουργείται η αρχιτεκτονική, αναλύεται, παράγεται, και στο τέλος αυτής, ολοκληρώνεται το λογισμικό βάση των απαιτήσεων της δεύτερης.

Φάση 4

Στη τέταρτη φάση γίνεται η επαλήθευση ότι τα παραδοτέα της τρίτης πληρούν τις προδιαγραφές που ορίστηκαν στη δεύτερη. Αν τα αποτελέσματα αυτής δεν είναι αποδεκτά τότε μένει να αποφασιστεί αν χρειάζεται κάποια αλλαγή και όλο το έργο να γυρίσει στο στάδιο της ανάπτυξης ,είτε να τερματιστεί το έργο.

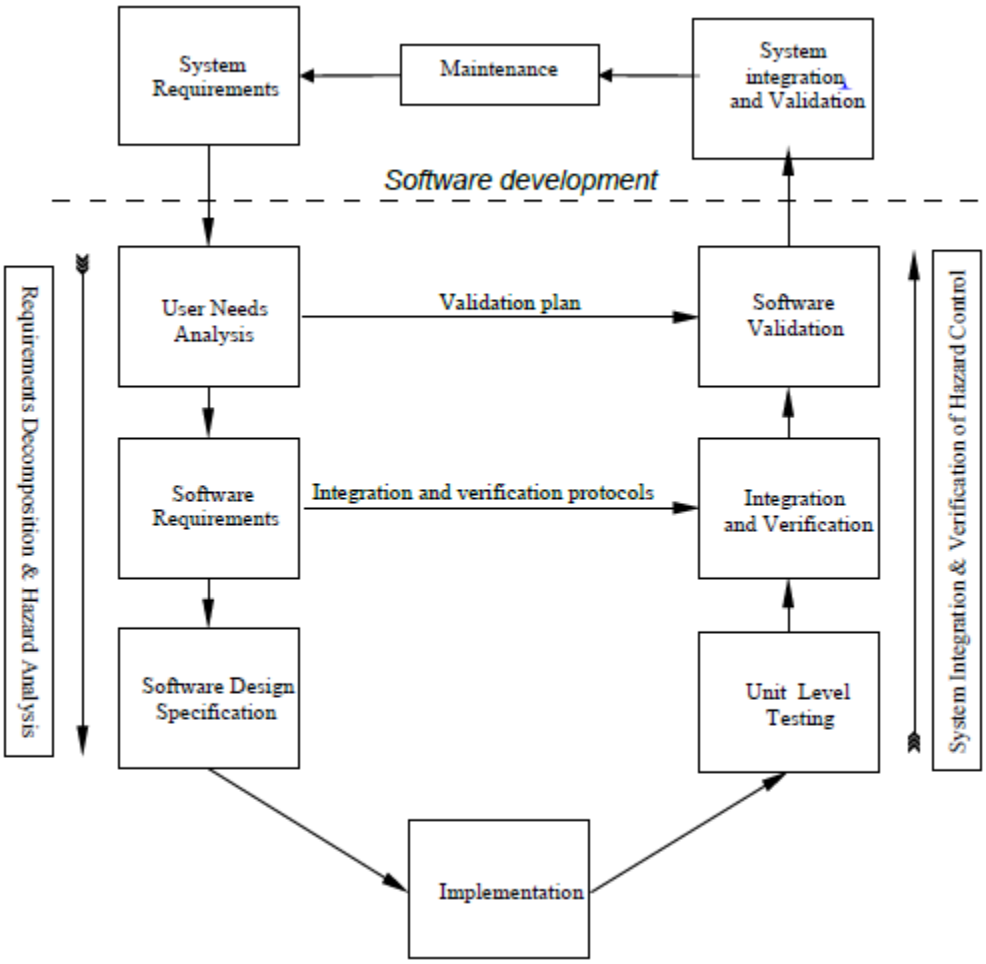
Φάση 5

Στην φάση αυτή, δημιουργείται ένα πλάνο που περιλαμβάνει τις διαδικασίες επικύρωσης που θα εκτελεστούν καταγράφοντας τα αποτελέσματά τους. Κάθε απαίτηση που ορίστηκε στη δεύτερη φάση θα πρέπει να επικυρώνεται από ένα ή περισσότερα τεστ. Το κάθε τεστ εκτελείται από κάποιον που δεν συμμετείχε στη διαδικασία ανάπτυξης λογισμικού και καταγράφεται το όνομα και η ώρα που εκτελέστηκε μαζί με το αποτέλεσμα. Σε αυτή τη φάση συνεπάγεται ολιστική δοκιμή του τελικού προϊόντος και επικυρώνεται ότι το παραγόμενο προϊόν πήγε σύμφωνα με το πλάνο και ότι το προϊόν είναι ικανό και πληροί τις απαιτήσεις για την καθορισμένη εφαρμογή ή προβλεπόμενη χρήση.

Στο πλαίσιο αυτό, διενεργούνται και τυχόν κλινικές αξιολογήσεις ή αξιολόγηση των επιδόσεων του ιατρικού προϊόντος, όπως απαιτείται από τους εθνικούς κανονισμούς.
Τα αποτελέσματα από τα τεστ, αξιολογούνται και πιθανά προβλήματα που βρίσκονται αναλύονται από τα ενδιαφερόμενα μέρη. Όσα πρέπει να επιλυθούν, ακολουθούν την παραπάνω διαδικασία.

Φάση 6

Στην τελευταία φάση παράγεται το τελικό παραδοτέο λογισμικό και είναι έτοιμο για διατεθεί στον πελάτη. Οποιαδήποτε απαραίτητη εκπαίδευση που χρειάζεται να γίνει στους τελικούς χρήστες πραγματοποιείται πριν ή κατά τη διάρκεια αυτής της φάσης.



Εικόνα 4 V-model Software Development Life Cycle

4.4 Τρέχων τρόπος ανάπτυξης

Κάθε νέα έκδοση ξεκινάει με τον καθορισμό των απαιτήσεων σε επίπεδο προϊόντος (product requirements) τα οποία καθορίζονται από την προϊοντική ομάδα. Εκεί θεσπίζονται σε ανώτατο επίπεδο οι επιθυμίες και οι στόχοι για την επόμενη έκδοση.

Με βάση τα product requirements και τη συμμετοχή του αρχιτέκτονα της ομάδας του λογισμικού, δημιουργούνται με περισσότερη λεπτομέρεια οι προδιαγραφές σε επίπεδο λογισμικού (software specifications).

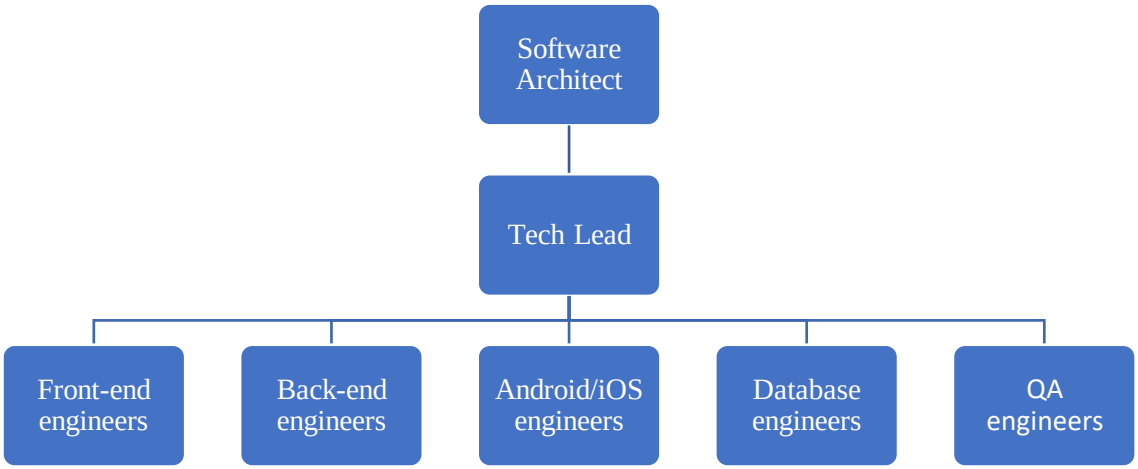
Από τα software specifications, ύστερα από εκτενέστερη ανάλυση, αναλύοντας σε επίπεδο ασφάλειας, λειτουργιών όπως και απόδοσης του συστήματος, καθοριζόταν οι περιπτώσεις χρήσης κάθε λειτουργίας (use cases). Αυτή είναι η βάση για τους μηχανικούς για να ξεκινήσουν την ανάπτυξη.
Ταυτόχρονα, η ομάδα των γραφιστών, δημιουργούσε τα αρχικά πρότυπα βασιζόμενοι στις απαιτήσεις.

Όλα τα παραπάνω έπρεπε να είναι έτοιμα πριν ξεκινήσει η ανάπτυξη λογισμικού ώστε να μπορέσει να βγει ένα πλάνο για την παράδοση τους τελικού προϊόντος. Οποιαδήποτε αλλαγή, επέβαλε να ξανά τρέξουν όλα τα βήματα σειριακά.
Στο τέλος της ανάπτυξης (τέλος 3ης φάσης), η ομάδα QA δημιουργούσε όλες τις περιπτώσεις επαλήθευσης (test cases) για τα product requirements, software specifications και use cases. Τα συγκεκριμένα τεστ θα χρησιμοποιηθούν κατά την εκτέλεση της 4ης και 5ης φάσης.

Κάθε ένα από τα product requirements, software specifications, use cases και test cases είχαν ένα μοναδικό αναγνωριστικό όπου χρησιμοποιείται στο τέλος για την ιχνηλασιμότητα μεταξύ τους. Η ιχνηλασιμότητα σε επίπεδο απαιτήσεων γινόταν από product requirements στα software specifications και use cases. Σε επίπεδο test cases γινόταν από product requirements σε test cases και από software specifications σε test cases.

4.5 Υπάρχουσα δομή και τρόπος λειτουργίας

Μέχρι να αρχίσει η μετάβαση σε ανάπτυξη με ευέλικτες μεθόδους, η ομάδα της ανάπτυξης λογισμικού είχε την παρακάτω δομή:



Τρόπος λειτουργίας

Βήματα	**Εκτέλεση από**
Ανάλυση της υλοποίησης (Analysis)	Tech Lead / Software Architect
Υλοποίηση από προγραμματιστή (Feature Development)	Engineers
Έλεγχος της υλοποίησης (Code Review)	Senior engineers
Επαλήθευση της υλοποίησης (QA Testing)	QA Team
Έλεγχος/Ενσωμάτωση της υλοποίησης (Final code review / Deployment)	Tech Lead / System Administrator

Πίνακας 1 Τρόπος λειτουργίας

4.6 Τα προβλήματα

Ύστερα από το τέλος κάθε έκδοσης κάθε τμήμα στην εταιρείας έκανε μια συνεδρίαση και κατέγραφε τι πήγε καλά και τι θα μπορούσε να βελτιωθεί. Σε διαδικαστικό επίπεδο εντοπίστηκαν τα παρακάτω και παρουσιάζονται σε τυχαία σειρά.

Συγχρονισμός ομάδων/πληροφορίας

Η ανάλυση γινόταν από τον Αρχιτέκτονα Λογισμικού (Software Architect) μαζί με τον Υπεύθυνο της ομάδας των Προγραμματιστών (Tech Lead). Λόγω εμπειρίας ήταν τα πιο κατάλληλα άτομα να κάνουν την τεχνολογική ανάλυση αλλά ήταν πολύ δεσμευτικό επειδή θα έπρεπε αυτοί να μεταφέρουν την πληροφορία και να συγχρονιστούν στην

έναρξη της υλοποίησης ώστε να μειωθούν οι εξαρτήσεις. Επειδή ποτέ δεν είναι εύκολο να προβλέψεις τα πάντα στην φάση της ανάλυσης, τις περισσότερες φορές κάτι νέο θα έβγαινε που δεν είχε προβλεφθεί με αποτέλεσμα να πρέπει να αξιολογηθεί η απαίτηση, η τεχνική υλοποίηση και μετά να μεταφερθεί στα υπόλοιπα συστήματα για να υλοποιηθεί και εκεί. Κάτι τέτοιο προκαλούσε αρκετή σύγχυση γιατί ένα από τα συστήματα μπορεί ήδη να ήταν στην υλοποίηση της επόμενης λειτουργίας και να έπρεπε να γυρίσει πίσω για τις νέες διορθώσεις είτε για την να προκαλέσει καθυστέρηση επειδή κάποιο άλλο πρόβλημα βγήκε την υλοποίηση που δουλεύουν εκείνη τη στιγμή.

Ταχύτητα

Αφού η ομάδα ακολουθούσε τα στάδια που αναφέρονται στον πίνακα 1 είχε ως αποτέλεσμα όλα τα τμήματα να μην δουλεύουν ταυτόχρονα σε μια συγκεκριμένη υλοποίηση (feature), επειδή υπήρχε διαφορά ταχύτητας, δυσκολιών, κλπ. Συνεπώς, ένα feature ενώ θα μπορούσε να ολοκληρώνεται σχετικά σύντομα, λόγω προτεραιοτήτων και συγχρονισμού καθυστερεί μέχρι να ολοκληρωθούν όλα τα κομμάτια του από κάθε σύστημα καθιστώντας το μη λειτουργικό. Το μεγαλύτερο όμως πρόβλημα ήταν ότι για να θεωρηθεί κάτι ολοκληρωμένο, θα έπρεπε να επιθεωρηθεί από τον υπεύθυνο τεχνολογίας και να ενσωματωθούν όλες η υλοποιήσεις σε μια τελική που θα μπορούσε να καταλήξει στο τελικό προϊόν. Όλος αυτός ο συγχρονισμός σε ένα τόσο γρήγορο περιβάλλον όπως η ανάπτυξη λογισμικού άρχισε να προκαλεί καθυστερήσεις και οι υλοποιήσεις να αργούν να είναι έτοιμες ολιστικά ενώ τα επιμέρους κομμάτια ήταν.

Ποιότητα

Ολοκλήρωση του feature γινόταν μόνο μετά το τελευταίο στάδιο. Λόγω των προβλημάτων συγχρονισμού και ταχύτητας που αναφέρθηκαν πιο πάνω, πολλά προβλήματα έβγαιναν στο τέλος που γινόταν η τελική επαλήθευση της υλοποίησης. Αυτό είχε σαν αποτέλεσμα λειτουργίες που δούλευαν νωρίτερα να μην δουλεύουν και να πρέπει γίνει εκ νέου υλοποίηση για τη διόρθωση τους. Ως τελικό αποτέλεσμα ήταν είτε να υπάρχει καθυστέρηση λόγω της ανάγκης για επίλυση όλων των προβλημάτων, είτε περισσότερα σφάλματα στο παραδοτέο προϊόν.

Διαφάνεια

Μέχρι να ολοκληρωθεί το τελευταίο στάδιο υλοποίησης ήταν πολύ δύσκολο να γνωρίζει ο κάθε ένας που βρισκόμαστε σαν ομάδα είτε από τους συναδέλφους, είτε από τους ανώτερους είτε από τα ενδιαφερόμενα μέρη. Αυτό είχε σαν αποτέλεσμα τη δημιουργία

αβεβαιότητας μέχρι την τελική στιγμή όπως και πιθανές αποκλίσεις που μπορεί να είχαν δημιουργηθεί επειδή οι ανάγκες θα έπρεπε να αλλάξουν.

Αποδιοργάνωση

Η ομάδα δούλευε όχι μόνο στην ανάπτυξη νέων εκδόσεων και λειτουργιών αλλά και στην συντήρηση των υπαρχουσών. Σε εβδομαδιαία βάση υπήρχαν αρκετές ανάγκες για διερεύνηση και επίλυση προβλημάτων. Η συνεχής αποδιοργάνωση της καθημερινότητας για την υποστήριξη, προκαλούσε αρκετή σύγχυση και καθυστερήσεις τόσο στην υλοποίηση των λειτουργιών όσο και στη διευθέτηση των προβλημάτων λόγω συντήρησης.

Μειωμένη ανατροφοδότηση

Όσο η ομάδα έκανε ανάπτυξη με παραδοσιακές μεθόδους η ανατροφοδότηση ήταν ελάχιστη και υπήρχε μόνο στη φάση της επαλήθευσης όπου είχε δημιουργηθεί ένα προϊόν ολοκληρωμένο και λειτουργικό. Αυτό τις περισσότερες φορές έκρυβε εκπλήξεις αφού δεν γνώριζαν τα ενδιαφερόμενα μέρη πως θα λειτουργούν ακριβώς πολλές από τις λειτουργίες δημιουργώντας αποκλίσεις από το αρχικό επιθυμητό αποτέλεσμα.

Αυξημένο ωράριο εργασίας

Η εργασία πέρα από το προβλεπόμενο ωράριο είχε καταλήξει σε ρουτίνα στο τέλος της 3ης και 4ης φάσης για όσες εκδόσεις βγήκαν τα πρώτα δύο χρόνια.

Σε ένα τόσο δυναμικό προϊόν ήταν πολύ δύσκολο να γίνει σωστή εκτίμηση της τελικής ημερομηνίας παράδοση με αποτέλεσμα να χρειάζονται επιπλέον ώρες εργασίας για να καλυφθεί η όποια απόκλιση.

Για την πρώτη έκδοση που η υλοποίηση της διήρκησε περίπου 9 μήνες, η ομάδα δούλευε 6 μέρες τη βδομάδα και τον τελευταίο μήνα για πάνω από 12 ώρες ημερησίως.

Στην επόμενη έκδοση λόγω συμπληρωματικού λογισμικού που έπρεπε να υλοποιηθεί, η εργασία συνέχισε να είναι για αρκετούς μήνες γύρω στις 11-12 ώρες καθημερινά αυτή τη φορά όμως με 5 μέρες τη βδομάδα εργασία.

Αλλαγές απαιτήσεων

Ένα από τα σημαντικότερα προβλήματα που είχε άμεση επιρροή στην καθημερινότητα της ομάδας ήταν η αλλαγή των απαιτήσεων. Παρόλο που με τον υπάρχων τρόπο, οι απαιτήσεις θα έπρεπε να είναι έτοιμες μετά την πρώτη φάση, σε ένα τμήμα που

λειτουργεί σε ρυθμούς startup ήταν ανέφικτο μιας και συνεχώς υπήρχαν δυσκολίες στην ανάπτυξη ενός νέου προϊόντος.

4.7 Γιατί agile

Η βιομηχανία ιατρικών μηχανημάτων υιοθετεί με βραδύτερο ρυθμό από άλλες βιομηχανίες ευέλικτες μεθόδους για την ανάπτυξη των συσκευών και του λογισμικού τους. Ο (Spence, 2006) κατέληξε ήδη στο συμπέρασμα ότι «*η ανάπτυξη με μια ευέλικτη μέθοδο είναι η προσέγγιση που ταιριάζει καλύτερα στην ανάπτυξη ιατρικών συσκευών που ρυθμίζονται από τον FDA*». Ωστόσο, οι (McHugh et al., 2013) σημειώνουν ότι η έλλειψη αποδεικτικών στοιχείων σε ρυθμιζόμενα περιβάλλοντα εμποδίζει την υιοθέτηση τους. Αναγνωρίζουν την ανάγκη για αλλαγή, δεδομένου ότι το λογισμικό γίνεται όλο και πιο αναπόσπαστο κομμάτι των παραγόμενων ιατρικών συσκευών. Αυτή η αλλαγή αντικατοπτρίζεται στους κανονισμούς όπου από το 2012, η Ευρωπαϊκή Ένωση θεωρεί το λογισμικό ενός ιατρικού μηχανήματος ενεργό μέρος του ιατρικού προϊόντος, γεγονός που δείχνει τη σημασία που έχει το λογισμικό, κατατάσσοντας το σε κατηγορία υψηλότερου κινδύνου που απαιτεί πιστοποίηση από κοινοποιημένο οργανισμό.

Παρόλο που οι επιχειρήσεις όπως αναφέρθηκε πιο πάνω είναι σκεπτικές στη δοκιμασία τεχνικών επειδή υπάρχει η αμφιβολία πως θα επιβραδύνει αναπόφευκτα η παραγωγικότητα, θα τις κάνει να χάσουν χρήματα και να διακόψουν τις καθημερινές λειτουργίες της, υπάρχουν δύο πολύ συγκεκριμένοι λόγοι που όσοι εργάζονται στην ανάπτυξη λογισμικού ιατρικών μηχανημάτων διστάζουν να δοκιμάσουν κάποια ευέλικτη μέθοδο.

Αρχικά πιστεύουν ότι δεν θα είναι σε θέση να επιτύχουν και να διατηρήσουν τη συμμόρφωση με τους κανόνες αν οι απαιτήσεις του προϊόντος (και επομένως το αποτέλεσμα) εξελίσσονται συνεχώς καθ' όλη τη διάρκεια του έργου. Αυτός είναι ο λόγος για τον οποίο τείνουν να τηρούν τις δοκιμασμένες μεθόδους, όπως του καταρράκτη ή το V-μοντέλο, οι οποίες σχεδιάζουν εξαντλητικά εκ των προτέρων χρησιμοποιώντας τη λίστα απαιτήσεων που πρέπει να πληρούνται πριν βγουν στην αγορά, γεγονός που θεωρητικά οδηγεί σε ομαλότερη συμμόρφωση.

Το δεύτερο είναι ότι ο συνδυασμός ανάπτυξης υλικού / λογισμικού θεωρείται μη εφικτός με τις ευέλικτες μεθόδους, καθώς χρησιμοποιήθηκαν αρχικά καθαρά σε περιβάλλοντα ανάπτυξης λογισμικού. Παρόλο που μπορεί νέος κώδικας να ενσωματωθεί σχετικά

γρήγορα και σε τακτά χρονικά διαστήματα, οι απαιτήσεις υλικού όπως η παραγωγή μηχανικών μερών (η οποία απαιτεί πολλά σχέδια μηχανικής και συνεργασία με εξωτερικούς προμηθευτές) συνήθως δεν μπορεί να παραχθεί με τον ίδιο ρυθμό μιας και δεν είναι τόσο εύκολο να αλλάξει το υλικό σε μεταγενέστερα στάδια παραγωγής όπως το λογισμικό (Narodetsky, n.d.).

Στην εταιρεία, η σκέψη για αλλαγή του τρόπου λειτουργίας και ανάπτυξης λογισμικού ήρθε ύστερα από την καταγραφή των προβλημάτων και μετά από έρευνα βρέθηκαν τα παρακάτω πλεονεκτήματα που έχουν οι ευέλικτες μέθοδοι που είναι ταυτόχρονα και κάποια από τα προβλήματα που αντιμετώπιζε η εταιρεία.

Αυξημένη ευελιξία και προσαρμοστικότητα

Λόγω των παραδοσιακών μοντέλων ανάπτυξης λογισμικού που ακολουθούνται συνήθως για την ανάπτυξη λογισμικού για ιατρικά μηχανήματα, η επαλήθευση του προϊόντος καθυστερεί μέχρι το τέλος της υλοποίησης που σημαίνει ότι τα σφάλματα και οι κίνδυνοι δεν αποκαλύπτονται μέχρι το τέλος του έργου. Με τις ευέλικτες μεθοδολογίες, η έμφαση δίνεται σε συνεχή επαλήθευση καθ’ όλη τη διάρκεια της διαδικασίας, πράγμα που σημαίνει ότι τα πιθανά προβλήματα μπορούν να αντιμετωπιστούν μόλις προκύψουν, με χαμηλότερο κόστος, χωρίς καθυστέρηση του έργου.

Μειωμένο ρίσκο

Όπως έχουμε ήδη αναλύσει παραπάνω, οι παραδοσιακές προσεγγίσεις ανάπτυξης λογισμικού αφήνουν τα προβλήματα να ανακαλυφθούν κοντά στο τέλος ενός έργου, καθώς το λογισμικό μόνο τότε γίνεται διαθέσιμο για δοκιμές. Ταυτόχρονα στο τέλος διατίθεται και για τροφοδότηση σχολίων από τα ενδιαφερόμενα μέρη. Με τις ευέλικτες μεθοδολογίες, οι κίνδυνοι μπορούν να εντοπιστούν σε όλη τη διαδικασία ανάπτυξης, χρησιμοποιώντας τακτικές όπως γρήγορη αποτυχία - γρήγορη εκμάθηση (fail fast – learn faster), επαλήθευση και επικύρωση των απαιτήσεων για την επίτευξη έγκαιρης διαχείρισης του κινδύνου.

Καλύτερη ποιότητα

Με τις ευέλικτες μεθοδολογίες, συνήθως η ποιότητα του λογισμικού είναι υψηλότερη επειδή το λογισμικό χτίζεται σταδιακά και υπάρχει συνεχή ανατροφοδότηση για πιθανά σχόλια και προβλήματα. Έτσι πιθανά σφάλματα διορθώνονται γρηγορότερα και να αλλάξει και η πορεία του έργου ευκολότερα λόγω της συνεχής ανατροφοδότησης.

4.8 Συμμόρφωση με κανονισμούς

Η ανάπτυξη λογισμικού για ιατρικά και ιατροτεχνολογικά μηχανήματα είναι ρυθμισμένη και απαιτεί να πληρούνται κάποια πρότυπα. Τρία είναι τα σημαντικά πρότυπα: ISO 13485 – διαχείριση ποιότητας για ιατροτεχνολογικά προϊόντα, ISO 14971 – διαχείριση κινδύνου για ιατροτεχνολογικά προϊόντα και IEC 62304 – λογισμικό ιατρικών συσκευών – διαδικασίες στον κύκλο ζωής ενός λογισμικού. Άμεσα και έμμεσα, η χρήση αυτών των προτύπων είναι απαραίτητη για τα ιατροτεχνολογικά προϊόντα από την Ευρωπαϊκή Ένωση, ο οποίος επιβάλλει να υπάρχει ένα σύστημα διαχείρισης ποιότητας και κινδύνου.

4.8.1 IEC 62304 - Διαδικασίες στον κύκλο ζωής ενός λογισμικού

Είναι το πρότυπου που παίζει κεντρικό ρόλο και περιγράφει τις διαδικασίες, εργασίες και δραστηριότητες που πρέπει να πληρούνται κατά την ανάπτυξη ενός λογισμικού για ιατρικά μηχανήματα. Το λογισμικό τοποθετείται στο πλαίσιο του V-model, περιγράφοντας φάσεις κατά σειρά σχεδιασμού, ανάλυσης απαιτήσεων, αρχιτεκτονικού σχεδιασμού, λεπτομερούς σχεδιασμού, υλοποίησης και επαλήθευσης, δοκιμών ολοκλήρωσης και κυκλοφορίας του συστήματος. Το πρότυπο υποθέτει και επιβάλλει ότι το λογισμικό αναπτύσσεται και συντηρείται στο πλαίσιο ενός συστήματος διαχείρισης ποιότητας και συστήματος διαχείρισης κινδύνων.

Το πλάνο ανάπτυξης και συντήρησης λογισμικού περιγράφει τις διαδικασίες, τους ρόλους και τους πόρους, τη διαμόρφωση του λογισμικού, τις δραστηριότητες επαλήθευσης και δοκιμών καθώς και τη συντήρησή του. Οι διαδικασίες περιλαμβάνουν τη διαδικασία ανάπτυξης λογισμικού, τη διαχείριση κινδύνων και τις διαδικασίες επίλυσης προβλημάτων. Οι πόροι περιλαμβάνουν το προσωπικό και οποιουσδήποτε τεχνικούς πόρους απαιτούνται. Για τις δραστηριότητες διαμόρφωσης, επαλήθευσης και δοκιμής του λογισμικού, συγκεντρώνονται τα ευρήματα της υλοποίησης και της εκτέλεσης και καταγράφονται στα αντίστοιχα έγγραφά τους.

Στο έγγραφο απαιτήσεων λογισμικού, καταγράφονται όλες οι απαιτήσεις για το λογισμικό. Αυτές οι απαιτήσεις μπορούν να γραφτούν ως ιστορίες χρηστών και κατηγοριοποιούνται σε λειτουργικές, ασφάλειας, βάσεων αλλά και κανονιστικές. Οι απαιτήσεις πρέπει να έχουν ένα μοναδικό αναγνωριστικό, ώστε να μπορεί να εκπληρωθεί η απαίτηση της ιχνηλασιμότητας κατά τη διαδικασία της ανάπτυξης.

Έπειτα οι απαιτήσεις μετατρέπονται σε αρχιτεκτονική. Το πρότυπο δεν καθορίζει πως μπορεί να συμβεί αυτό στην πραγματικότητα όμως είναι σε μορφή διαγραμμάτων και περιλαμβάνει συνήθως και τον τρόπο που αλληλοεπιδρούν άλλα μέρη με το λογισμικό (Eidel, 2023).

Επίσης το IEC 62304 επιβάλει την καταγραφή της επαλήθευσης του λογισμικού. Ο λόγος είναι πως πρέπει στο τέλος να είναι βέβαιο ότι το υλοποιημένο λογισμικό καλύπτει τις απαιτήσεις που ορίστηκαν στην αρχή του έργου. Υπάρχουν διάφοροι τρόποι επαλήθευσης όπως ο έλεγχος του πηγαίου κώδικα αλλά και διάφορες μορφές τεστ (εκτελέσιμα χειροκίνητα ή αυτόματα με εκτελέσιμο κώδικα).

Στη φάση του ελέγχου, περιλαμβάνονται όλα τα τεστ που επιβεβαιώνουν πως όλες οι απαιτήσεις έχουν καλυφθεί. Εδώ περιλαμβάνονται στοιχεία επαλήθευσης αλλά και απαιτήσεις που δεν μπορούν να ελεγχθούν αυτόματα και σε αυτή την περίπτωση είναι απαραίτητη η ύπαρξη ενός πλάνου που περιλαμβάνει τα αποδεικτικά στοιχεία μέσα από αρχεία, ρυθμίσεις ή ακόμα και από στιγμιότυπα οθόνης.

Τέλος για να εκδοθεί το τελικό λογισμικό οι απαιτήσεις που πρέπει να καλυφθούν είναι η ολοκλήρωση της επαλήθευσης και του ελέγχου, όλα τα ρίσκα και πιθανά προβλήματα να είναι καταγεγραμμένα και η τελική έκδοση να αποθηκευτεί κάπου με ασφάλεια και να είναι διαθέσιμη για τουλάχιστον τα επόμενα 10 χρόνια.

4.8.2 ISO 13485 - Διαχείριση ποιότητας για ιατροτεχνολογικά προϊόντα

Σχετικό με την εφαρμογή του IEC 62304, ως μέρος του συστήματος διαχείρισης ποιότητας είναι ότι το λογισμικό που χρησιμοποιείται εντός του οργανισμού πρέπει να είναι επικυρωμένο (ISO, 2016, 4.1.6). Αυτό σημαίνει ότι το λογισμικό που χρησιμοποιείται για την ανάπτυξη λογισμικού ιατρικών συσκευών, όπως και κάθε εργαλείο μέσα στην εταιρεία θα πρέπει να καταγράφονται και να τεκμηριώνεται ο λόγος επιλογής τους.

4.8.3 ISO 14971 - Διαχείριση κινδύνου για ιατροτεχνολογικά προϊόντα

Κάθε έργο έχει και κάποιο ρίσκο. Ρίσκο στην καθυστέρηση του παραδοτέου, στο προκαθορισμένο κεφάλαιο και πολλά άλλα. Η αναγνώρισή τους όμως βοηθάει στην

προετοιμασία πιθανής λύσης ή ακόμα καλύτερα την αποφυγή τους. Η ανάλυση ρίσκου είναι σημαντική κατά την υλοποίηση κρίσιμων λογισμικών όπως είναι τα ιατρικά και είναι απαραίτητη για να εγγυηθεί η ασφάλεια του προϊόντος. Ξεκινάει στη φάση του σχεδιασμού και εξελίσσεται κατά τη διάρκεια της ανάπτυξης μέχρι και μετά το τελικό παραδοτέο και είναι απαραίτητη ώστε να υπάρχει συμμόρφωση με το πρότυπο ISO 14971 έχοντας πάντα υπόψιν το ποσοστό ρίσκου προς τους ασθενείς. Στην εταιρεία που αναλύουμε, η ανάλυση ρίσκου γίνεται στο τέλος της πρώτης και έκτης φάσης.

Τα στάδια της διαδικασίας περιλαμβάνουν τον εντοπισμό κινδύνων και επικίνδυνων καταστάσεων, την εκτίμηση της επικινδυνότητας που συνοδεύεται από αυτές τις καταστάσεις και την αξιολόγηση τους. Η εκτίμηση του κινδύνου γίνεται από τη δυνητική βλάβη που μπορεί να προκαλέσει ένας κίνδυνος στο ασθενή, όπως και ανάλογα με τη πιθανότητα εμφάνισης ενός κινδύνου όπως παρουσιάζονται στους πίνακες παρακάτω.

		Severity				
		Negligible	Minor	Moderate	Critical	Catastrophic
Occurrence	Continuously Occurring	High	High	Intolerable	Intolerable	Intolerable
	Frequent	Moderate	Moderate	High	Intolerable	Intolerable
	Occasional	Low	Moderate	High	High	Intolerable
	Rare	Low	Low	Moderate	High	Intolerable
	Remote	Low	Low	Low	High	High

Πίνακας 2 Επικινδυνότητα κατάστασης σε ιατρικά μηχανήματα

Η αξιολόγηση των κινδύνων πραγματοποιείται σύμφωνα με τις απαιτήσει του παρακάτω πίνακα με προσπάθεια πάντα να μειωθεί κάποιο ρίσκο στο χαμηλότερο βαθμό.

Risk Level	Description
Intolerable	Unacceptable region. The level of risk is unacceptable for the product and must be reduced.
High	This level of risk is considered tolerable only if further reduction of the risk is not practicable in that the benefits of further reducing the risk do not outweigh the activities associated with the reduction.
Moderate	This level of risk is considered tolerable only if further reduction of the risk is not practicable in that the benefits of further reducing the risk do not significantly reduce the residual risk any further.
Low	Broadly acceptable region; the level of risk is acceptable for the product.

Πίνακας 3 Εκτίμηση κινδύνου σε ιατρικά μηχανήματα

Η ύπαρξη προτύπων είναι σημαντική, όπως αναφέρθηκε προηγουμένως, για να υποχρεώσει την εταιρεία να ακολουθήσει τις καλύτερες δυνατές πρακτικές.

4.9 Απαιτήσεις που έπρεπε να καλυφθούν

Ανεξάρτητα από ποια μεθοδολογία θα χρησιμοποιούταν, θα έπρεπε να καλύπτονται τα πρότυπα και οι κανονισμοί που αναλύθηκαν πιο πάνω και απαιτούνται για την ανάπτυξη λογισμικού για ιατρικά μηχανήματα. Πέρα από τη διαχείριση κινδύνου, παρακάτω αναφέρουμε μέσα από τα πρότυπα όλα αυτά που διαφοροποιούν την ανάπτυξη ενός λογισμικού για ιατρικά μηχανήματα σε αντίθεση με τα υπόλοιπα (Zamith & Gonçalves, 2018).

4.9.1 Διαδικασία ανάπτυξης

Τα πρότυπα απαιτούν τη δημιουργία ενός πλάνου που θα καταγράφει όλα τα στάδια ανάπτυξης αλλά και τη κατηγορία στην οποία εμπίπτει το παραγόμενο λογισμικό. Θα πρέπει να αναφέρεται η μέθοδος που ακολουθείται για την υλοποίηση, όπως και να αναφέρει όλα τα παραδοτέα από τις απαιτήσεις μέχρι τις τελικές αναφορές επικύρωσης του προϊόντος.

4.9.2 Διαδικασία συντήρησης

Θα πρέπει να υπάρχει μια διαδικασία για τη συντήρηση του λογισμικού μετά την παράδοσή του, καταγράφοντας με λεπτομέρεια ένα πλάνο για το πως θα λαμβάνονται, καταγράφονται, αξιολογούνται, επιλύονται αλλά και παρακολουθούνται πιθανά προβλήματα κατά τη μελλοντική συντήρησή του παραδοτέου.

4.9.3 Αναφορά και καταγραφή των προβλημάτων

Βασιζόμενοι στα παραπάνω πρότυπα, απαιτείται να υπάρχει μια αναφορά (report) για κάθε πρόβλημα που εντοπίζεται στο τελικό προϊόν και μπορεί να προκαλέσει πιθανό κίνδυνο. Σε αυτές τις αναφορές θα πρέπει πάντα να αναφέρεται ο τύπος, ο σκοπός που καλύπτεται αλλά και η κρισιμότητα του προβλήματος. Επιπλέον, λεπτομερείς στρατηγικές αλλά και ευρήματα για το πως προέκυψε το πρόβλημα μαζί με βήματα για τη διασφάλιση της επίλυσης όπως είναι η δημιουργία συγκεκριμένων τεστ.

4.9.4 Ιχνηλασιμότητα

Η ιχνηλασιμότητα είναι απαραίτητη για τη συμμόρφωση με το πρότυπο IEC 62304 και στοχεύει στην ασφάλεια. Χρησιμοποιείται συνήθως ανάμεσα στις απαιτήσεις και τεστ που εκτελούνται κατά τη φάση επαλήθευσης ή και συντήρησης του προϊόντος. Είναι ακόμα ένας τρόπος να οργανωθεί περισσότερο το έργο και ο κύκλος ζωής τους προϊόντος. Τώρα η ιχνηλασιμότητα έχει εξελιχθεί σε κάτι περισσότερο από τη χρήση της μόνο στις απαιτήσεις και έχει προχωρήσει στη διαχείριση σφαλμάτων, τη διαχείριση αλλαγών και γενικά των έργων (Regan et al., 2013).

4.10 Αλλαγή ομάδας - 2 ομάδες cross-functional

Για να μπορέσει να υποστηριχθεί η μετάβαση σε ανάπτυξη χρησιμοποιώντας το Scrum framework έπρεπε να γίνουν αλλαγές στη δομή, στους ρόλους και τα καθήκοντα της ομάδας.
Δημιουργήθηκαν δύο διατομεακές ομάδες (cross-functional teams). Η κάθε ομάδα περιλάμβανε έναν τουλάχιστον μηχανικό από κάθε σύστημα front-end, backend, database, android και QA.

Με αυτή την αναδιοργάνωση, χρειάστηκαν να γίνουν και εσωτερικές αλλαγές στους ρόλους των προγραμματιστών για να καλυφθούν οι ανάγκες. Η απόφαση πάρθηκε

συνεργατικά με τους προγραμματιστές βασιζόμενοι στο υπόβαθρό τους όπως και την επιθυμητή μελλοντική τους εξέλιξη.

4.11 Ρόλοι

Με τη μετάβαση στο Scrum, δύο νέοι ρόλοι απαιτήθηκαν να δημιουργηθούν για την καλύτερη και ορθότερη εφαρμογή του, αυτός του Scrum Master και του Product Owner. Ταυτόχρονα κρατήθηκε ο αποκλειστικός ρόλος του QA μέσα στη δημιουργία της Scrum ομάδας.

Τον ρόλο του Scrum Master τον πήρε ο Project Manager. Για να συμβεί αυτό χρειάστηκε εκπαίδευση για να μπορέσει να τον υποστηρίξει. Χρειάστηκε αρκετή μελέτη αλλά και εκπαίδευση από εξωτερικό εκπαιδευτή για την απόκτηση πιστοποίησης. Η αλλαγή του τρόπου σκέψης και η κατανόηση των βασικών αρχών των ευέλικτων μεθόδων ήταν και από τις πιο προκλητικές αλλαγές.

Για τον ρόλο του Product Owner τα πράγματα ήταν λίγο πιο περίπλοκα μιας και λόγω της δομής και ενασχόλησης ανατέθηκε στον Αρχιτέκτονα της ομάδας. Γνωρίζοντας το προϊόν και έχοντας έντονη επικοινωνία με τους stakeholders θεωρήθηκε ο κατάλληλος γι’ αυτή τη θέση. Χρειάστηκε η βοήθεια και εκπαίδευση από τον Product Manager ώστε να υποστηρίξει σταδιακά και ορθά τον ρόλο.

Στην περίπτωσή μας, δημιουργήθηκε και ο ρόλος του ατόμου διασφάλισης ποιότητας (**QA**) μέσα στην ομάδα, που διασφαλίζει ότι όλοι οι απαραίτητοι έλεγχοι ποιότητας εκτελούνται κατά τη διαδικασία ανάπτυξης. Υπάρχει ένα άτομο και για τις δύο ομάδες που είναι και ο υπεύθυνος για τη συνολική ποιότητα παράδοσης του προϊόντος.

Πως βοήθησαν και λειτουργήσαν οι ρόλοι αυτοί στην εταιρεία και στην ανάπτυξη.
Ο Scrum Master καθοδηγεί καθημερινά την ομάδα και τα ενδιαφερόμενα μέρη σε ένα πιο λιτό τρόπο λειτουργίας μέσα από την υλοποίηση του Scrum. Ταυτόχρονα βοηθάει τον Product Owner να δημιουργεί πιο δομημένα καθήκοντα και το κάθε ένα να προσφέρει αξία στον τελικό προϊόν.

Ο Product Owner άρχισε να είναι το κοινό άτομο αναφοράς για τις απαιτήσεις του προϊόντος και κάνοντας την ανάλυση του βοήθησε σε πιο ξεκάθαρη πληροφορία έχοντας ως αποτέλεσμα η ομάδα να δουλεύει σε στοιχεία που έχουν αξία.

Η ομάδα πλέον είναι αφοσιωμένη μόνο μέσα στα στοιχεία που πρέπει να δουλέψει μέσα στο sprint με αποτέλεσμα να μην αντιμετωπίζει τις όποιους περισπασμούς έρχονται επειδή ο Scrum Master βοηθάει στην διευθέτησή τους.

4.12 Εργαλεία που χρησιμοποιήθηκαν

Μέχρι πριν την απόφαση για βελτίωση των διαδικασιών ανάπτυξης, όλα τα έγγραφα δουλεύονταν και συντηρούνταν σε συμβατικά προγράμματα κειμενογράφου και υπολογιστικών φύλλων. Η έγκριση, συντήρηση, πρόσβαση σε ιστορικά δεδομένα όπως και η ταυτόχρονη συνεργασία και ενημέρωση πολλαπλών ανθρώπων, ήταν δύσκολη και κάποιες φορές αδύνατη. Για να λυθούν αυτά τα προβλήματα, χρειάστηκε να χρησιμοποιηθούν τα παρακάτω εργαλεία: Jira, Xray, R4J και QCBD.

1. Jira

   Το Jira είναι από τα πιο γνωστά εργαλεία διαχείρισης έργου. Η ομάδα το χρησιμοποιούσε και νωρίτερα απλά για την καταγραφή των εργασιών (tasks) που έπρεπε να εκτελεστούν. Με την εφαρμογή του Scrum, στήθηκε νέο Project στο Jira που υποστηρίζει το Scrum framework. Έγινε ευκολότερο να οργανωθεί η δουλειά, χρησιμοποιώντας ένα κοινό μέρος για την καταγραφή των εργασιών και των δύο ομάδων από τη στιγμή που δουλεύουν σε κοινό προϊόν.

2. R4J

   Η καταγραφή των απαιτήσεων γινόταν σε αρχεία κειμένου. Παρόλο που για την εγγραφή τους δεν υπήρχε κάποιο πρόβλημα, οι δυσκολίες ξεκίνησαν όταν έπρεπε να αναφερθούν σε αυτές όσοι έκαναν την ανάλυσή τους, δημιουργούσαν τεστ, tasks και οι προγραμματιστές όταν χρειάζονταν κάποια διευκρίνιση για την υλοποίηση. Το R4J είναι ένα πρόσθετο που εγκαθίσταται στο Jira και βοηθάει στην καταγραφή και οργάνωση των απαιτήσεων. Με το R4J υπήρχε άμεση σύνδεση με τα tasks στο Jira οπότε ήταν πολύ εύκολο να ανατρέξει κάποιος.

3. Xray

   Το Xray είναι ένα πρόσθετο που εγκαθίσταται πάνω στο Jira και βοηθάει στη διαχείριση των τεστ που πρέπει να εκτελεστούν για το λογισμικό.

   Μέχρι εκείνη τη στιγμή, όλα τα test αλλά και οι συσχετίσεις τους ανάμεσα στις απαιτήσεις καταγράφονταν σε υπολογιστικά φύλλα. Αυτό δημιουργούσε αρκετές δυσκολίες μιας και πέρα ότι τα συγκεκριμένα εργαλεία δεν έχουν φτιαχτεί για τέτοια δεδομένα, δυσκόλευε πολύ στην συντήρηση, οργάνωση και εκτέλεση των τεστ. Κάθε χρήστης που κατέγραφε τα σενάρια που θα χρησιμοποιηθούν έπρεπε

να δουλεύει σε δικό του αντίγραφο και στο τέλος να γίνεται ενοποίηση όλων σε ένα κοινό.

Ένα άλλο μεγάλο πρόβλημα ήταν πως για κάθε διαφορετικό σύστημα, έκδοση του λογισμικού που έπρεπε να βγει ή και επιδιόρθωση κάποιου σφάλματος, έπρεπε να δημιουργηθεί εκ νέου ένα στιγμιότυπο και χειροκίνητα να περαστούν τα τεστ. Πλέον με την χρήση ετικετών (labels) σε κάθε τεστ, είναι πολύ εύκολο να αντιληφθεί κάποιος σε ποιο σύστημα αναφέρεται ένα τεστ και να δημιουργήσει προκαθορισμένα σετ ανάλογα την ανάγκη.

Κατά την εκτέλεση των τεστ, μέχρι να αρχίσει η λειτουργία του Xray, ο κάθε ένας που εκτελούσε τα τεστ, έπρεπε να καταγράφει την ημερομηνία, ώρα, το όνομα του και την υπογραφή του δίπλα από κάθε αποτέλεσμα ώστε να πληροί τα κριτήρια των εγγράφων που πρέπει να δημιουργούνται στην ανάπτυξη λογισμικού για ιατρικά μηχανήματα. Η συγκεκριμένη διαδικασία πρόσθετε αρκετό χρόνο και με την εκτέλεση των τεστ μέσω του Xray, γινόταν αυτόματα αφού έπαιρνε τα στοιχεία του κάθε χρήστη από το σύστημα.

Αυτοί οι περιορισμοί δημιουργούσαν καθυστερήσεις και τα ποσοστά λάθους ήταν πάρα πολλά κατά την εκτέλεση τους διότι ήταν πολύ δύσκολο να συγχρονιστεί η πληροφορία για όλα τα συστήματα.

4. QCBD

   Όπως αναφέρθηκε πιο πάνω, στην ανάπτυξη λογισμικών για ιατρικά μηχανήματα, είναι απαραίτητο να υπάρχει ένα Σύστημα Διαχείρισης Ποιότητας (Quality Management System – QMS) για να υπάρχει συμμόρφωση με το πρότυπο ISO 13485. Τα συγκεκριμένα λογισμικά χρησιμοποιούνται για αρκετούς λόγους στον κύκλο ανάπτυξης. Εδώ θα αναφερθούμε στην αρχειοθέτηση των παραγόμενων αρχείων, την διαχείριση εκδόσεων και ιστορικού στα αρχεία όπως και την έγκρισή τους.

   Τα παραγόμενα αρχεία είτε πριν τη μετάβαση στα παραπάνω συστήματα είτε μετά, που περιλάμβαναν τις απαιτήσεις, τα τεστ που ελέγχουν το λογισμικού, τις συσχετίσεις αυτών όπως και τις αναφορές που έπρεπε να δημιουργηθούν μετά την εκτέλεση τους, έπρεπε να αποθηκεύονται και να είναι εύκολα προσβάσιμα και να υπάρχει η δυνατότητα να ανατρέξει κάποιος σε παλιότερη έκδοσή τους για λόγους ελέγχου.

Κάθε παραγόμενο αρχείο, αφού γινόταν αναθεώρηση από τους κατάλληλους ανθρώπους, έπρεπε να εγκριθούν για να θεωρηθούν τελικά και να μπορέσουν να χρησιμοποιηθούν. Η έγκριση μέχρι πριν γίνει η χρήση του συστήματος, γινόταν χειρόγραφα υπογράφοντας ο κάθε ένας το αρχείο και δίνοντάς το στον επόμενο μέχρι να μαζευτούν όλες οι κατάλληλες υπογραφές. Με το QCBD, ως εγκεκριμένο QMS, γινόταν με ηλεκτρονική υπογραφή κάτι που έκανε πολύ πιο εύκολη τη διαδικασία.

5. Μετάβαση από τα αρχεία στα συστήματα

   Για να μπορέσει η ομάδα να χρησιμοποιήσει τα εργαλεία αυτά, έπρεπε όλη η υπάρχουσα πληροφορία να μεταφερθεί σε αυτά. Η συγκεκριμένη διαδικασία έγινε σταδιακά σε περίοδο 6 μηνών ώστε να υποστηριχτεί ο όγκος της πληροφορίας. Για να γίνει αυτό, τα αρχεία μετατράπηκαν σε CSV και έγινε εισαγωγή τους στο Jira. Μεταφέρθηκαν όλες οι συσχετίσεις ανάμεσα στις απαιτήσεις και στα τεστ για να συνεχίσει να υπάρχει συμμόρφωση με τους κανόνες.

4.12.1 Πως λειτουργούν όλα μαζί

Οι απαιτήσεις καταγράφονται στο R4J αναλυτικά πριν από κάθε σπριντ. Κατά τη διάρκεια της ανάλυσής τους, δημιουργούνται και τα καθήκοντα στο Jira σε μορφή ιστοριών χρήστη (User stories). Εκεί συνδέονται όλες οι απαιτήσεις για να διασφαλιστεί ότι κάτι δεν παραλείπεται κατά την ανάπτυξη.

Μόλις ένα user story καταλήξει μέσα στο sprint και ξεκινήσει η υλοποίηση του, οι μηχανικοί διασφάλισης ποιότητας (Quality Assurance – QA) δημιουργούν τα σενάρια που θα τεστάρουν την υλοποίηση. Κάθε σενάριο τεστ που δημιουργείται είτε αυτόματο είτε χειροκίνητο καταγράφεται μέσα στο Xray και ταυτόχρονα συνδέεται με τις απαιτήσεις που έχουν καταγραφεί στο R4J. Έτσι διασφαλίζουμε ότι όλες οι απαιτήσεις έχουν επαληθευτεί και έμμεσα υπάρχει συμμόρφωση με τον κανονισμό που απαιτεί ιχνηλασιμότητα μεταξύ των απαιτήσεων, της υλοποίησης και της επαλήθευσης αυτής.

4.13 Διαδικασίες – Παραγωγή εγγράφων

Όπως αναφερθήκαμε παραπάνω, τα έγγραφα, η ορθή καταγραφή των απαιτήσεων και η επαλήθευση αυτών είναι απαραίτητα στην ανάπτυξη λογισμικού για ιατρικά μηχανήματα. Μέχρι πριν την υιοθέτηση του Scrum, η δημιουργία των εγγράφων για τις απαιτήσεις της υλοποίησης καθώς και τα τεχνικά έγγραφα που περιγράφουν την

αρχιτεκτονική που θα χρησιμοποιηθεί, γινόταν στην φάση 2, και η έγκρισή τους από τα ενδιαφερόμενα μέρη ήταν απαραίτητη για να ξεκινήσει η υλοποίηση.

Η καταγραφή των απαιτήσεων ενσωματώθηκε στη φάση της ανάπτυξης λογισμικού χρησιμοποιώντας το λογισμικό διαχείρισης έργων Jira μαζί με ένα πρόσθετο R4J. Οι απαιτήσεις καταγράφονταν στο σύστημα από τον Product Manager. Έπειτα, ύστερα από την ανάλυση τους σε τεχνικό και λειτουργικό επίπεδο, δημιουργούνται οι τεχνικές απαιτήσεις όπου οι προγραμματιστές θα ακολουθήσουν για την ανάπτυξη.
Κατά τη διάρκεια της ανάπτυξης, η ομάδα ανατρέχει στις απαιτήσεις είτε για την ανάπτυξη είτε για τη δημιουργία των τεστ που θα εκτελεστούν για την επαλήθευση της υλοποίησης. Κάθε απαίτηση θα πρέπει να τεστάρετε από τουλάχιστον ένα τεστ. Τα τεστ είναι συνδεμένα με τις απαιτήσεις μέσω του συστήματος. Με αυτόν τον τρόπο, θα χρησιμοποιηθεί αργότερα η πληροφορία αυτή για την δημιουργία των τελικών εγγράφων. Μέχρι πριν την ενσωμάτωση του εργαλείου ήταν μια χειροκίνητη διαδικασία που γινόταν σε έγγραφα και όχι σε σύστημα. Απαιτούσε πολύ χρόνο για να μειωθεί το ποσοστό λάθους.

Από τη στιγμή που όλα τα δεδομένα ήταν στο σύστημα, για να μπορέσουν να δημιουργηθούν τα έγγραφα που θα υπογράφονταν και θα καταχωρούνταν στο αρχείο έπρεπε να γίνει εξαγωγή και να περαστούν στο QCBD για να εγκριθούν και να αποθηκευτούν.

4.14 Οργάνωση - Meetings

Η διαδικασία ανάπτυξης λογισμικού για κρίσιμα έργα είναι αυστηρότερη από ό,τι τα υπόλοιπα. Οι κανονισμοί προβλέπουν την επαλήθευση της υλοποίησης για να εγγυηθεί η ασφάλεια και η πρόληψη ατυχημάτων. Αυτές οι διαδικασίες είναι ενσωματωμένες στο V-model, αλλά μπορούν να επιτευχθούν και στην ευέλικτη ανάπτυξη (Hanssen et al., 2018).

Η διαδικασία που ακολουθήθηκε βασίζεται στο Scrum. Για τη χρονική οργάνωση χρησιμοποιήθηκαν επαναλήψεις που ονομάζονται sprint και είναι το χρονικό διάστημα που διαρκεί η ανάπτυξη λογισμικού. Στην περίπτωση μας η διάρκεια κάθε sprint είναι δύο εβδομάδες. Η ανάπτυξη χωρίζεται σε τρεις φάσεις: τον προγραμματισμό του sprint, την εκτέλεση του και το κλείσιμο. Ανάμεσα σε αυτές τις φάσεις είναι ενσωματωμένες και οι διαδικασίες ανάλυσης κινδύνου και ποιοτικού ελέγχου.

Πριν την έναρξη του sprint, χρειάζεται να υπάρχει μία κοινή κατανόηση για το παραγόμενο ανάμεσα στους προγραμματιστές και τον πελάτη. Για να επιτευχθεί αυτό, θα πρέπει να συλλεχθούν όλες οι απαιτήσεις. Αυτές συγκεντρώνονται και αναλύονται αρχικά από τον Product Owner.

Sprint Refinement

Κατά τη διάρκεια του sprint δημιουργήθηκε ακόμα μία συνάντηση όπου οι προγραμματιστές με τον Product Owner συζητούν, αναλύουν και τμηματοποιούν τα κομμάτια της δουλειάς που έχουν να ασχοληθούν. Αυτό βοηθάει ώστε όταν χρειαστεί να ξεκινήσει η υλοποίηση ενός τμήματος δουλειάς, θα έχουν ξεκαθαριστεί οι περισσότερες πληροφορίες και θα είναι ξεκάθαρος ο στόχος που πρέπει να καλυφθεί με την ολοκλήρωσή του. Οι προγραμματιστές είναι υπεύθυνοι για να αξιολογήσουν και το μέγεθος της δουλειάς που θα απαιτηθεί για την ολοκλήρωση του κάθε τμήματος.

5 ΑΝΑΛΥΣΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΕΡΜΗΝΕΙΑ ΕΥΡΗΜΑΤΩΝ

Τα δεδομένα που θα χρησιμοποιήθηκαν για την προβολή των αποτελεσμάτων είναι παραθέσεις των υπαλλήλων και θα κατηγοριοποιηθούν σε δυο κατηγορίες, σε θέματα προς βελτίωση και τις επιτυχίες που αναγνωρίζουν κάθε φορά μετά την παράδοση μιας έκδοσης. Θα παραθέσουμε μόνο πληροφορίες σχετικά με την ανάπτυξη λογισμικού, τις διαδικασίες και τα εργαλεία. Θα αξιολογήσουμε τρεις χρονικές στιγμές, η πρώτη είναι η τελευταία έκδοση που αναπτύχθηκε με V-Model και έπειτα οι επόμενες δύο που αναπτύχθηκαν με Scrum.

Μεθοδολογία (Έτος)	Επιτυχίες	Προς Βελτίωση
V-Model (10/2018-6/2019)	• Πρώτη λειτουργική έκδοση σε πελάτη • Μετάβαση από τοπική σε αρχιτεκτονική cloud • Έτοιμα σχέδια από τον γραφίστα πριν την έναρξη υλοποίησης	• Αυξημένες ώρες εργασίας • Πρόβλημα με τις άδειες λόγω εξειδίκευσης • Χάνεται πληροφορία από τους Team Lead στην ομάδα • Χειροκίνητα τεστ, απαιτούν αρκετό χρόνο για εκτέλεση • Αλλαγές απαιτήσεων κατά τη διάρκεια της υλοποίησης • Τεχνικό χρέος στην υλοποίηση
Scrum (9/2019-5/2020)	• Ελάχιστες υπερωρίες • Το Jira βοήθησε στην για την καταγραφή των απαιτήσεων • Τα Sprint review βοήθησαν στην	• Βελτίωση στον τρόπο εκτίμησης φόρτου/χρόνου • Συνεχόμενη ενσωμάτωση του κώδικα (CI)

	παράδοση του επιθυμητού προϊόντος • Οι διαδικασίες βοήθησαν στην τηλεργασία • Συντονισμός ανάμεσα στην ομάδα ανάπτυξης και το business	• Περισσότερη αυτοματοποίηση στην επαλήθευση του προϊόντος
Scrum (6/2020-5/2021)	• Εκτέλεση των τεστ στο Jira • Πλήρης αυτοματοποίηση των εγγράφων • Μεγαλύτερο ποσοστό αυτόματης επαλήθευσης • Θεσπισμένες απαιτήσεις πριν από κάθε sprint • Ομαλή υλοποίηση με μειωμένο προσωπικό	• Παράδοση της έκδοσης στις γιορτές του Πάσχα • Μειωμένο ανθρώπινο δυναμικό • Παράδοση μικρότερων εκδόσεων • Κρίσιμες λειτουργίες υλοποιήθηκαν στο τέλος

Πίνακας 4 Ευρήματα ανάλυσης ανά μεθοδολογία

Πέρα από τα διαδικαστικά, θα αναλύσουμε και το ποσοστό των σφαλμάτων που παρουσιάστηκαν σε κάθε έκδοση μετά την παράδοση στο ίδιο διάστημα, όπως και των σφαλμάτων που βρέθηκαν κατά τη διάρκεια της ανάπτυξης.

Μεθοδολογία (Έτος)	Εντοπίστηκαν μετά την παράδοση	Εντοπίστηκαν και λύθηκαν κατά την υλοποίηση
V-Model (10/2018-6/2019)	12	87
Scrum (9/2019-5/2020)	6	101
Scrum (6/2020-5/2021)	5	424

Πίνακας 5 Αριθμός σφαλμάτων ανά μεθοδολογία

5.1 Συμπεράσματα

Από τα δεδομένα που έχουμε παραθέσει πιο πάνω βλέπουμε πως κατά τη διάρκεια της τελευταίας έκδοσης που υλοποιήθηκε με τη χρήση του V-Model, τα προβλήματα ήταν αρκετά και κυρίως αυτά που ανάγκασαν στην αναζήτηση λύσεων και την κατάληξη στις ευέλικτες μεθόδους και συγκεκριμένα στο Scrum. Παρατηρούμε ότι η έναρξη της επόμενης έκδοσης έγινε τρεις μήνες μετά την παράδοση και αυτό γιατί λόγω των αλλαγών των απαιτήσεων κατά τη διάρκεια της ανάπτυξης, λήφθηκαν αρκετές αποφάσεις που προκάλεσαν τεχνικό χρέος και αρχιτεκτονικές αποφάσεις που δεν ήταν βιώσιμες για την επέκταση του προϊόντος. Η συγκεκριμένη συντήρηση έπρεπε να γίνει για να μπορούν να υποστηριχθούν μελλοντικές εκδόσεις χωρίς προβλήματα στην απόδοση της εφαρμογής. Κατά τη διάρκεια αυτών των μηνών ταυτόχρονα γινόταν η ανάλυση των προβλημάτων και πως μπορούν να διορθωθούν που οδήγησε στην εξέλιξη και αλλαγή από V-Model σε Scrum.

Στην πρώτη έκδοση που βγήκε κοντά ένα έτος μετά με την εφαρμογή του Scrum, παρατηρούμε πως ήδη κάποια από τα προβλήματα άρχισαν να επιλύνονται. Σε αυτό βοήθησε όχι μόνο η αλλαγή στον τρόπο λειτουργίας αλλά και στα εργαλεία που άρχισε να χρησιμοποιεί η ομάδα βλέποντας το πρώτο θετικό να είναι στην ανάλυση των απαιτήσεων. Μειώθηκαν οι εκπλήξεις στην παράδοση του έργου αφού με την ανατροφοδότηση κάθε τέλος του sprint τα ενδιαφερόμενα μέρη μπορούσαν να εκφράσουν τα σχόλια τους για τα επόμενα βήματα και να προταθούν τυχόν αλλαγές πριν την έναρξη της υλοποίησης. Ένα ακόμα θετικό που παρατηρούμε είναι η ευκολία μετάβασης σε τηλεργασία την περίοδο της έξαρσης του Covid-19. Οι θεσπισμένες διαδικασίες και τα εργαλεία που είχαν αρχίσει να εφαρμόζονται βοήθησαν ώστε η περίοδος προσαρμογής να είναι ελάχιστες μέρες. Αυτό ήταν δυνατό διότι κατά το μετασχηματισμό του τρόπου λειτουργίας υπολογίστηκε και η συνεργασία με την ομάδα στην Αμερική. Παρόλο που το θετικό αντίκτυπο άρχισε να φαίνεται στην πρώτη έκδοση παρατηρούμε ότι υπάρχει αρκετό περιθώριο βελτίωσης στην εκτίμηση του φόρτου εργασίας και πόσο χρόνος θα απαιτηθεί περίπου για την υλοποίηση του και αυτό γιατί έπρεπε να αλλάξει ο τρόπος σκέψης και να μπορέσει η ομάδα να προχωρήσει από τα εκτιμήσεις και τη διαδικασία που ακολουθούταν κατά τη χρήση του V-Model.

Στην δεύτερη έκδοση του λογισμικού που παράχθηκε κάτω από την εφαρμογή του Scrum παρατηρούμε πως τα θετικά είναι ακόμα περισσότερα ειδικά στο κομμάτι της αυτοματοποίησης της παραγωγής των απαραίτητων εγγράφων που χρειάζονται για τη

συμμόρφωση με τους κανονισμούς. Είναι μια διαδικασία που μείωσε κατά 80% τον χρόνο παραγωγής των εγγράφων και αυτό επειδή η πληροφορία πλέον ήταν κεντροποιημένη στο Jira. Παράλληλα βλέπουμε και στα θετικά και στα προς βελτίωση σχόλια, το αποτέλεσμα που έφερε η ομάδα έχοντας λίγο πιο μειωμένο ανθρώπινο. Με τις διαδικασίες που έχει το Scrum όπως το Daily Scrum, ήταν πολύ εύκολο να βγουν στην επιφάνεια προβλήματα που μπορεί να αντιμετώπιζε κάποιος μηχανικός, να τα μοιράζεται και η ομάδα να προσπαθεί να βοηθήσει. Η διαφάνεια που ήρθε μέσω των διαδικασιών, βελτίωσε αρκετά τις εκπλήξεις και μπόρεσε να καλύψει και το κενό μια θέσης μηχανικού που μειώθηκε.

Επιπλέον πέρα από τις διαδικαστικές βελτιώσεις, υπήρξε και αρκετή βελτίωση στην ποιότητα του τελικού προϊόντος έχοντας λιγότερα σφάλματα στο τελικό προϊόν αλλά πιο βασικά βρίσκοντας περισσότερα σφάλματα κατά τη διαδικασία της ανάπτυξης. Αυτό έγινε εφικτό γιατί ως ενέργεια είχε παρθεί να ενσωματωθεί και η ανάπτυξη αυτοματοποιημένων τεστ κατά τη διάρκεια ανάπτυξης μέσα στο sprint. Έτσι μειώθηκε το ποσοστό ανθρώπινου λάθους και ο χρόνος που επενδύθηκε γύρισε πίσω στην ομάδα ώστε να μπορέσει να ασχοληθεί με κάτι άλλο.

Η εκπαίδευση μπορεί να θέσει τις βάσεις ώστε να επεκταθούν οι ευέλικτες μέθοδοι πέρα από την ανάπτυξη λογισμικού. Οι εκπαιδευτικοί στα σχολεία και στα πανεπιστήμια, αρχίζουν να χρησιμοποιούν ευέλικτες μεθόδους για την εκμάθηση. Τις χρησιμοποιούν για να δώσουν έμφαση στην θέσπιση προτεραιοτήτων. Η δια βίου αλληλεπίδραση, η ουσιαστική εκπαίδευση, οι αυτόνομα οργανωμένες ομάδες και η σταδιακή μάθηση που εκμεταλλεύονται τη φαντασία, ανήκουν στις ευέλικτες αρχές που μπορούν να αλλάξουν και να διαμορφώσουν τον τρόπο σκέψης όχι μόνο στην τάξη αλλά και στην πορεία τους έξω από αυτή (Briggs, 2014).

5.2 Προτάσεις για βελτίωση

Αντιλαμβανόμαστε ότι υπήρξαν αρκετές βελτιώσεις και θετικά αποτελέσματα από την υιοθέτηση νέων διαδικασιών. Η συνεχόμενη βελτίωση είναι ένα από τα κύρια κομμάτια των ευέλικτων μεθοδολογιών και πάνω σε αυτή την αρχή διακρίθηκαν τα παρακάτω σημεία που μελλοντικά θα μπορούσαν να βελτιώσουν ακόμα περισσότερο τον τρόπο λειτουργίας και τα αποτελέσματα αυτής.

Η περισσότερη προσοχή προς βελτίωση θα μπορούσε να είναι στο κομμάτι του ελέγχου του κώδικα χρησιμοποιώντας αυτόματα τεστ, κανόνες γραφής κώδικα κλπ. Με αυτό τον τρόπο τυχόν προβλήματα μπορούν να εντοπιστούν πολύ πριν φτάσουμε στην φάση της επαλήθευσης βελτιώνοντας την ποιότητα και τους χρόνο παράδοσης.

Ένα ακόμα κομμάτι που δεν χρήζει βελτίωσης και γίνονται βήματα προς τα εκεί είναι η δημιουργία διαδικασιών για συνεχή ενσωμάτωση νέων λειτουργιών (Continuous Integration – CI). Αυτή τη στιγμή εμπλέκει αρκετά χειροκίνητες διαδικασίες κυρίως στην αποσφαλμάτωση της διαδικασίας. Δεδομένου ότι χρειάζεται στο τέλος κάθε sprint να υπάρχει μια λειτουργική έκδοση, θα επιστρέψει αρκετός χρόνος στην ομάδα για να μπορέσει να λειτουργεί ακόμα πιο αποδοτικά.

Το επόμενο βήμα της συνεχής ενσωμάτωση νέων λειτουργιών είναι η συνεχή παράδοση στον πελάτη (Continuous Delivery – CD) και συνήθως υλοποιούνται και χρησιμοποιούνται μαζί αυτά τα δύο. Στην περίπτωση του λογισμικού για ιατρικά μηχανήματα υπάρχει η δυσκολία πως κάθε αλλαγή μπορεί να επιφέρει κίνδυνο προς τον ασθενή αν υπάρχει ελλιπής εκπαίδευση του προσωπικού που χρησιμοποιεί το λογισμικό γι’ αυτό και απαιτείται εκπαίδευση του προσωπικού που χρησιμοποιηθεί το λογισμικό μετά από κάθε νέα έκδοση. Στο κομμάτι που θα μπορούσε να βελτιωθεί η διαδικασία είναι να γίνεται παράδοση του τελικού προϊόντος αλλά ο πελάτης να μην έχει πρόσβαση στις νέες λειτουργίες και να ενεργοποιούνται μόνο όταν γίνει η κατάλληλη εκπαίδευση. Το προνόμιο σε θέμα χρόνου θα είναι αρκετό αφού η ομάδα θα μπορεί να δουλεύει σε επόμενες λειτουργίες και όταν αποφασιστεί να δημιουργηθεί μια νέα έκδοση, να γίνεται απλά ένας έλεγχος με αυτοματοποιημένο τρόπο για πιθανά σφάλματα.

Όσο αφορά στα εργαλεία που χρησιμοποιούνται, θα μπορούσε να γίνει κάποια στιγμή μετάβαση σε ένα σύστημα που περιλαμβάνει όλες τις λειτουργίες και διαχειρίζεται ολόκληρο τον κύκλο ζωής της ανάπτυξης (Application lifecycle management – ALM). Υπάρχουν συγκεκριμένα λογισμικά που πληρούν τις προϋποθέσεις για ανάπτυξη κρίσιμου λογισμικού καλύπτοντας όλους τους κανονισμούς που αναφέρθηκαν πιο πάνω. Ένα από αυτά είναι το Codebeamer. Το κόστος όμως σε σχέση με το συνδυασμό με τα υπάρχων συστήματα θα ήταν διαφορετικό.

6 ΒΙΒΛΙΟΓΡΑΦΙΑ

AAMI. (2012). *AAMI TIR45: 2012 Technical Information Report*. http://my.aami.org/aamiresources/previewfiles/TIR45_1208_PREVIEW.PDF

Barker, V. L., & Duhaime, I. M. (1997). Strategic change in the turnaround process: Theory and empirical evidence. *Strategic Management Journal*, *18*(1), 13–38. https://doi.org/10.1002/(sici)1097-0266(199701)18:1<13::aid-smj843>3.3.co;2-o

Beck, K., & Andres, C. (2004). *Extreme Programming Explained: Embrace Change, 2nd Edition (The XP Series)* (Vol. 2).

Briggs, S. (2014, February 22). *Agile Based Learning: What Is It and How Can It Change Education?* InformED. https://www.opencolleges.edu.au/informed/features/agile-based-learning-what-is-it-and-how-can-it-change-education/

Cameron, K. S., Kim, M. U., & Whetten, D. A. (1987). Organizational Effects of Decline and Turbulence. *Administrative Science Quarterly*, *32*(2), 222. https://doi.org/10.2307/2393127

Cho, J. (2008). Issues and Challenges of Agile Software Development With Scrum. *Issues In Information Systems*, *IX*(2), 188–195. https://doi.org/10.48009/2_iis_2008_188-195

Denzin, N. K., & Lincoln, Y. S. (2017). *The Sage handbook of qualitative research*. SAGE Publications, Inc.

Eidel, O. (2023). *Medical Device Software Architecture Documentation (IEC 62304)*. https://openregulatory.com/medical-device-software-architecture-documentation-iec-62304/

George, A. L., & Bennett, A. (2005). *Case Studies And Theory Development In The Social Sciences*. The MIT Press.

Greer, D., & Hamon, Y. (2011). Agile software development. *Software - Practice and Experience*, *41*(9), 943–944. https://doi.org/10.1002/spe.1100

Griffiths, M., & Project Management Institute. (2012). *PMI-ACP exam prep*. https://www.ptonline.com/articles/how-to-get-better-mfi-results

Hanssen, G. K., Stålhane, T., & Myklebust, T. (2018). *SafeScrum® – Agile Development of Safety-Critical Software*. Springer International Publishing. https://doi.org/10.1007/978-3-319-99334-8

Hugh, M. M., Caffery, F. M., Casey, V., & Pikkarainen, M. (2012). Integrating agile practices with a medical device software development lifecycle. *EuroSPI 2012*, *May*, 1–8. http://ulir.ul.ie/handle/10344/2524

Kuhrmann, M., Diebold, P., Münch, J., Tell, P., Garousi, V., Felderer, M., Trektere, K., McCaffery, F., Linssen, O., Hanser, E., & Prause, C. R. (2017). Hybrid software and system development in practice: Waterfall, scrum, and beyond. *ACM International Conference Proceeding Series*, *Part F128767*, 30–39. https://doi.org/10.1145/3084100.3084104

Leau, Y., Loo, W. K., Tham, W. Y., & Tan, S. F. (2012). *Software Development Life Cycle AGILE vs Traditional Approaches*. *37*(Icint), 162–167.

Lohrke, F. T., Bedeian, A. G., & Palmer, T. B. (2004). The role of top management teams in formulating and implementing turnaround strategies: A review and research agenda. *International Journal of Management Reviews*, *5–6*(2), 63–90. https://doi.org/10.1111/j.1460-8545.2004.00097.x

McHugh, M., McCaffery, F., Fitzgerald, B., Stol, K. J., Casey, V., & Coady, G. (2013). Balancing Agility and Discipline in a Medical Device Software Organisation. *Communications in Computer and Information Science*, *349 CCIS*, 199–210. https://doi.org/10.1007/978-3-642-38833-0_18

Medical Devices Market Size, Share & Growth | Report [2029]. (n.d.).

Munzner, R. F. (2003). *Entering the U.S. medical device market*. 3548-3550 Vol.4. https://doi.org/10.1109/IEMBS.2003.1280918

Narodetsky, L. (n.d.). *Introduction to Agile in Medical Device Development*. Retrieved November 22, 2022, from https://content.intland.com/blog/introduction-to-agile-in-medical-device-development

Petersen, K., & Wohlin, C. (2009). A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. *Journal of Systems and Software*, *82*(9), 1479–1490. https://doi.org/10.1016/j.jss.2009.03.036

Project Management Institute. (2017a). Agile practice guide. In *Choice Reviews Online*. https://pmi.bookstore.ipgbook.com/agile-practice-guide-products-9781628251999.php

Project Management Institute. (2017b). PMBOK® Guide Sixth Edition (PMI, 2017). In *Project Management Institute* (Vol. 6). http://www.citeulike.org/group/14887/article/9008974

Regan, G., McCaffery, F., Mc Daid, K., & Flood, D. (2013). Medical device standards' requirements for traceability during the software development lifecycle and implementation of a traceability assessment model. *Computer Standards & Interfaces*, *36*(1), 3–9. https://doi.org/10.1016/J.CSI.2013.07.012

Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide*. *November*.

Spence, J. W. (2006). *There has to be a better way! [software development]*. 272–278. https://doi.org/10.1109/adc.2005.47

Theocharis, G., Kuhrmann, M., Münch, J., & Diebold, P. (2015). Is water-scrum-fall reality? On the use of agile and traditional development practices. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9459*, 149–166. https://doi.org/10.1007/978-3-319-26844-6_11

Thomas, G. (2011). A Typology for the Case Study in Social Science Following a Review of Definition, Discourse, and Structure. *Http://Dx.Doi.Org/10.1177/1077800411409884*, *17*(6), 511–521. https://doi.org/10.1177/1077800411409884

West, D. (2011). Water-Scrum-Fall Is The Reality Of Agile For Most Organizations Today. *For Application Development & Delivery Professionals*, 1–15. http://www.storycology.com/uploads/1/1/4/9/11495720/water-scrum-fall.pdf

Wysocki, R. K. (2019). *Effective Project Management: Traditional, Agile, Extreme, Hybrid* (Vol. 33, Issue 3). Wiley. https://doi.org/10.1002/9781119562757

Zamith, M., & Gonçalves, G. (2018). Towards an Agile Development Model for Certifiable Medical Device Software - Taking Advantage of the Medical Device Regulation. *Proceedings of the 13th International Conference on Software Technologies*, 132–140. https://doi.org/10.5220/0006861301320140