Εθνικό Μετσόβιο Πολυτεχνείο (ΕΜΠ)

Σχολή Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών (ΣΕΜΦΕ)

ΜΑΘΗΜΑΤΙΚΗ ΠΡΟΤΥΠΟΠΟΙΗΣΗ ΣΕ ΣΥΓΧΡΟΝΕΣ ΤΕΧΝΟΛΟΓΙΕΣ ΚΑΙ ΤΗ ΧΡΗΜΑΤΟΟΙΚΟΝΟΜΙΚΗ

Agile Μετρικά και Μεθοδολογίες Ανάλυσης Δεδομένων από Συστήματα Διαχείρισης Εργασιών

ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Βατικιώτης Φώτιος

Τριμελής Επιτροπή:

Βεσκούκης Β, ΕΜΠ (ΣΑΤΜ), v.vescoukis@cs.ntua.gr
Στεφανέας Π, ΕΜΠ (ΣΕΜΦΕ), petros@math.ntua.gr
Τσανάκας Π, ΕΜΠ (ΣΗΜΜΥ), panag@cs.ntua.gr

Αθήνα, Ιούλιος 2023

Βατικιώτης Φώτιος Ιούλιος 2023

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τους Ανδρέας Δημάκης, Κωνσταντίνος Φωτόπουλος και τις Scrum ομάδες με τις οποίες συνεργάστηκα όλο αυτό τον καιρό. Επίσης, θα ήθελα να ευχαριστήσω θερμά τον κ. Βασίλειο Βεσκούκη για την επίβλεψη της διπλωματικής και την εξαιρετική συνεργασία μας. Σε κάθε απορία ή αναζήτηση που έκανα ήταν πάντοτε διαθέσιμος να μου προσφέρει τις γνώσεις και την εμπειρία του για την κατανόηση και την εμβάθυνση σε διαφορετικούς τομείς.

Σε αυτό το σημείο θα ήθελα να ευχαριστήσω ανθρώπους εκτός του στενού ακαδημαϊκού κύκλου οι οποίοι υπήρξαν σημαντικοί πόλοι στην ζωή μου. Θα ήθελα αρχικά να ευχαριστήσω τους φίλους μου και τον άνθρωπό μου Μαρία Μιχαηλίδου για την συνεχή συμπαράσταση, την βοήθεια και την κατανόηση που έδειξε κατά την διάρκεια των μεταπτυχιακών σπουδών μου.Το μεγαλύτερο ευχαριστώ το οφείλω στους γονείς μου και τον αδελφό μου για την στήριξη, την εμπιστοσύνη, τις συμβουλές, τα ιδανικά και την αγάπη που μου δείχνουν όλα αυτά τα χρόνια.

Η παρούσα εργασία είναι αφιερωμένη στον παππού και την γιαγιά μου.

Περίληψη

Οι Ευέλικτοι τρόποι ανάπτυξης λογισμικού αποτελούν έναν από τους τομείς με έντονη δραστηριότητα, προσανατολισμένο στην καινοτομία. Σκοπός τους είναι η δημιουργία μιας πιο διαφανούς και συνεργατικής ανάπτυξης λογισμικού, η οποία παρέχει άμεσα αξία στους τελικούς χρήστες και εξελίσσεται συνεχώς λαμβάνοντας υπόψιν την μεταβολή των συνθηκών. Στόχος τους είναι η δημιουργία μιας απλούστερης και αποδοτικότερης ανάπτυξης λογισμικού επικεντρωμένης στον πελάτη.

Αυτή η διατριβή εξερευνά διάφορες δημοφιλείς προσεγγίσεις Ευέλικτης ανάπτυξης λογισμικού, εξετάζοντας μετρήσεις που μπορούν να χρησιμοποιηθούν από οργανισμούς και ομάδες ανάπτυξης για να βελτιώσουν τη διαφάνεια και να προωθήσουν την αυτο-βελτίωση αυτών. Είναι σημαντικό να σημειωθεί ότι η χρησιμότητα αυτών των μετρήσεων μπορεί να ποικίλει ανάλογα με τη εκάστοτε προσέγγιση ανάπτυξης λογισμικού που επιλέγεται από έναν οργανισμό, είτε αυτή είναι Ευέλικτη είτε όχι.

Για να αντιμετωπίσει αυτήν τη δυνητική ασυμβατότητα, η διατριβή παρουσιάζει μια αυτοματοποιημένη μέθοδο για τη συλλογή και την οπτικοποίηση δεδομένων που είναι ειδικά προσαρμοσμένη στις Ευέλικτες μεθόδους ανάπτυξης. Στη συνέχεια, αναλύεται ένα δείγμα δεδομένων για να αξιολογηθεί η εφικτότητα χρήσης των ίδιων μετρήσεων σε περιπτώσεις όπου οι ομάδες ανάπτυξης ακολουθούν μη Ευέλικτες πρακτικές.

Τέλος, η διατριβή παρουσιάζει τα ευρήματα της μελέτης, μαζί με προτάσεις για μελλοντικές έρευνες που μπορούν να στηριχθούν σε αυτά τα αποτελέσματα. Με την εξέταση των πορισμάτων αυτής της μελέτης, οι ερευνητές και οι επαγγελματίες μπορούν να βελτιώσουν και να επεκτείνουν τις προτεινόμενες μετρήσεις, στοχεύοντας σε συνεχή βελτίωση των πρακτικών ανάπτυξης λογισμικού.

Abstract

Agile software development is one of the most active, innovation-oriented areas. Its purpose is to improve transparency and collaboration in software development that provides immediate value to the end users and continuously evolves in response to changing circumstances. Its goal is to create a simpler and more efficient customer-centric software development.

This thesis explores several popular approaches to Agile software development, examining metrics that can be used by organizations and development teams to improve transparency and promote self-improvement. It is important to note that the usefulness of these metrics can vary depending on the software development approach chosen by the organization, whether it is Agile or not.

To address this potential incompatibility, the thesis presents an automated method for data collection and visualization that is specifically tailored to Agile development methods. A sample of data is then analyzed to assess the feasibility of using the same metrics in cases where development teams follow non-Agile practices.

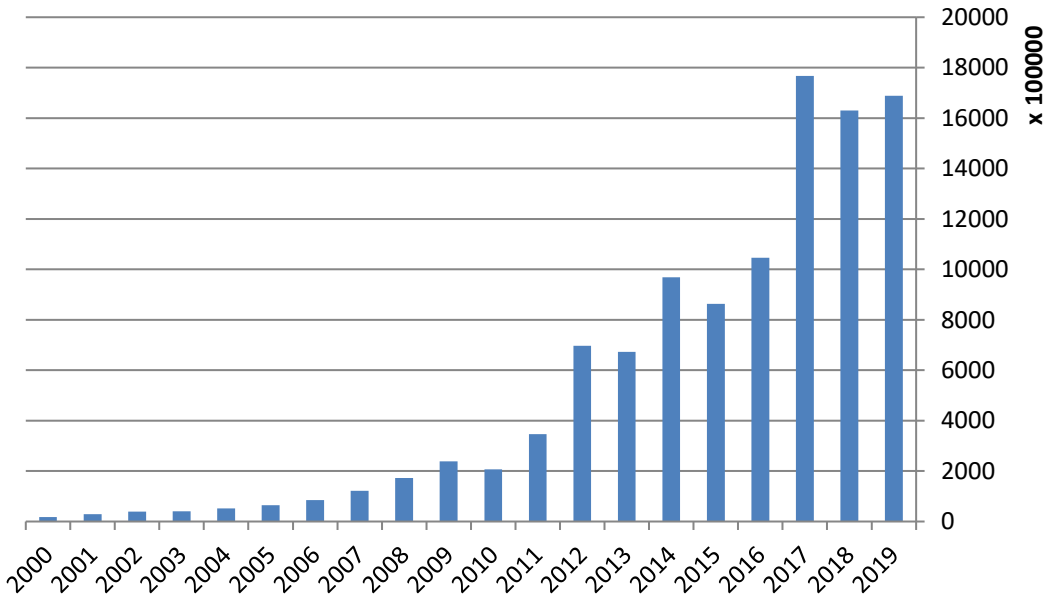
Finally, the thesis presents the findings of the study, along with suggestions for future research that can build on these results. By considering the findings of this study, researchers and practitioners can refine and extend the proposed metrics, aiming for continuous improvement of software development practices.

Περιεχόμενα

1. Εισαγωγή

Η εκθετική ανάπτυξη των διαδικτυακών εφαρμογών και των ιστοσελίδων έχει αναπτύξει την ανάγκη κατανόησης των χρηστών και των τρόπων αλληλεπίδρασης αυτών με διαδικτυακές εφαρμογές και ιστότοπους. Κατανοώντας τις ανάγκες των χρηστών, οι εταιρείες καταφέρνουν να βελτιώσουν τα τελικά προϊόντα τους και συνεπώς τις υπηρεσίες τους. Με αυτόν τον τρόπο, οι εταιρείες γίνονται πιο ανταγωνιστικές καλύπτοντας τις ανάγκες των χρηστών και ελαχιστοποιώντας τους κινδύνους τους.

Εκτός από την μελέτη της αλληλεπίδρασης των χρηστών με τους ιστότοπους, για την βελτιστοποίηση των τελικών προϊόντων τους, οι σύγχρονες εταιρίες αναπτύσσουν μηχανισμούς για την μέτρηση της απόδοσης των ομάδων ανάπτυξης λογισμικών κατά την διάρκεια της συνεργασίας και της ανάπτυξης των τεχνικών λύσεων. Οι μηχανισμοί αυτοί αποτελούν μετρικά λογισμικού, τα οποία μπορούν να αναδείξουν κατά πόσο ένα σύστημα ή και μια διαδικασία διαθέτουν κάποια ιδιότητα. Επιπλέον βοηθούν στην εκτίμηση προόδου, της υγείας και της ποιότητας ενός λογισμικού κατά τη διαδικασία ανάπτυξής του. Η διαθεσιμότητα αποτελεσματικών μετρικών στα πρώτα στάδια της ανάπτυξης λογισμικού επιτρέπει την καλύτερη διαχείριση των μεταγενέστερων βημάτων όπως όταν για παράδειγμα απαιτούνται διορθωτικές ενέργειες οι οποίες μπορούν εύκολα να βελτιώσουν την ποιότητα του τελικού προϊόντος.



Εικόνα 1: *Ιστότοποι από το 2000 ως το 2019*[1].

Στον παρακάτω πίνακα αποτυπώνεται αριθμητικά η εκθετική αύξηση των ιστότοπων.

[1] Πηγή: NetCraft and Internet Live Stats (elaboration of data by Matthew Gray of MIT and Hobbes' Internet Timeline and Pingdom).

Πίνακας 1: *Αριθμός ιστότοπων ανά τα χρόνια*[2].

Χρονιά	Αριθμός ιστότοπων[3]	Ποσοστό βάση της προηγούμενης χρονιάς
2019	1,689,023,048	4%
2018	1,630,322,579	-8%
2017	1,766,926,408	69%
2016	1,045,534,808	21%
2015	863,105,652	-11%
2014	968,882,453	44%
2013	672,985,183	-3%
2012	697,089,489	101%
2011	346,004,403	67%
2010	206,956,723	-13%
2009	238,027,855	38%
2008	172,338,726	41%
2007	121,892,559	43%
2006	85,507,314	32%
2005	64,780,617	26%
2004	51,611,646	26%
2003	40,912,332	6%
2002	38,760,373	32%
2001	29,254,370	71%
2000	17,087,182	438%

Αυτή η Μεταπτυχιακή Διπλωματική Εργασία μελετάει και προτείνει ένα σύνολο μετρικών λογισμικού που βασίζονται και μπορούν να εφαρμοστούν σε διαφορετικά frameworks ανάπτυξης λογισμικού, όπως το Scrum, το Kanban, το Lean, κλπ. Τα μετρικά αυτά συμβάλλουν στη βελτίωση της απόδοσης των συνεργατικών ομάδων ανάπτυξης λογισμικού και βελτιώνουν την ποιότητα της διαδικασίας ανάπτυξης και εξέλιξης των τελικών παραδοτέων που αναπτύσσονται σε έναν οργανισμό. Επιπλέον, τα περισσότερα από αυτά τα μετρικά μπορούν να εφαρμοστούν εύκολα σε οποιεσδήποτε καθημερινές δραστηριότητες για την διασφάλιση της αξιοπιστίας, της συντήρησης και της χρηστικότητας του τελικού προϊόντος. Για τη καλύτερη και αποτελεσματικότερη κατανόηση του προβλήματος, θα εστιάσουμε σε συγκεκριμένα frameworks τα οποία χαρακτηρίζονται από την Ευέλικτη μεθοδολογία (Agile methodology) και χρησιμοποιούνται αδρά στην ανάπτυξη λογισμικού τα τελευταία χρόνια.

[2] Πηγή: NetCraft and Internet Live Stats (elaboration of data by Matthew Gray of MIT and Hobbes' Internet Timeline and Pingdom).
[3] Με τον όρο ιστότοπο εννοούμε ένα μοναδικό hostname.

Πιο συγκεκριμένα, η Ευέλικτη μεθοδολογία είναι μια προσέγγιση στην ανάπτυξη λογισμικού σύμφωνα με την οποία οι απαιτήσεις εξελίσσονται κατά την διάρκεια ανάπτυξης του λογισμικού. Αυτό επιτυγχάνεται με την συχνή διάθεση νέων λειτουργιών στους τελικούς χρήστες και στην ανατροφοδότηση αυτών με σκοπό την ανάπτυξη και την ολοκλήρωση της τελικής λύσης-υπηρεσίας. Για την επιτυχία αυτού του εγχειρήματος, είναι απαραίτητη η ανάπτυξη αυτό-οργανωμένων και διαλειτουργικών ομάδων ανάπτυξης λογισμικού και η ύπαρξη ενός δίαυλου επικοινωνίας των ομάδων με τους τελικούς χρήστες. Η Ευέλικτη μεθοδολογία ή Agile methodology, υποστηρίζει τον προσαρμοστικό σχεδιασμό, την εξελικτική ανάπτυξη, την έγκαιρη παράδοση και τη συνεχή βελτίωση και ενθαρρύνει την ταχεία και ευέλικτη απόκριση στην αλλαγή. Η υιοθέτηση Ευέλικτων πρακτικών και αξιών βελτιώνει την ευελιξία των επαγγελματιών λογισμικού, των ομάδων και των οργανισμών και ελαχιστοποιεί τον κίνδυνο να μην ικανοποιηθούν οι προσδοκίες των πελατών ή/και των τελικών χρηστών.

Στο επόμενο κεφάλαιο παρουσιάζεται το Agile μανιφέστο όπως και κάποια από τα πιο διαδεδομένα Ευέλικτα πλαίσια (frameworks) ανάπτυξης λογισμικού που χρησιμοποιούν τις αρχές της Ευέλικτης μεθοδολογίας.

2. Ευέλικτα Πλαίσια Ανάπτυξης Λογισμικού

Όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο, σήμερα οι περισσότερες από τις εταιρείες που αναπτύσσουν διαδικτυακές λύσεις και λογισμικό, ακολουθούν τα ευέλικτα πλαίσια ανάπτυξης λογισμικού για να προσφέρουν το τελικό προϊόν στους χρήστες/πελάτες τους. Η έννοια Ευέλικτο πλαίσιο ή Agile framework, δεν αποτελεί μια Μεθοδολογία, ή ένας συγκεκριμένος τρόπος ανάπτυξης λογισμικού, ή ένα πλαίσιο, είτε μια διαδικασία. Το Agile είναι ένα σύνολο αξιών και αρχών, μια συλλογή από πεποιθήσεις που χρησιμοποιούν οι ομάδες πληροφορικής για να λάβουν αποφάσεις σχετικά με τον τρόπο ανάπτυξης λογισμικού. Το Agile δεν λαμβάνει αποφάσεις για μια ομάδα, αλλά οι ομάδες γίνονται ευέλικτες όταν λαμβάνουν αποφάσεις με βάση τις αξίες και τις αρχές του Agile.

Το κατά πόσο μια ομάδα ανάπτυξης λογισμικού είναι Ευέλικτη (Agile) μπορεί να περιγραφεί από τον τρόπο που λειτουργεί και παραδίδει ένα έργο αλλά και από τον τρόπο λήψης αποφάσεων εντός της ομάδας. Εφόσον το Agile δεν είναι μεθοδολογία, πρέπει να απαντήσουμε στην ερώτηση: «Τι αντιπροσωπεύουν οι Agile μεθοδολογίες;». Αλλά πριν απαντήσουμε σε αυτό το ερώτημα, θα ορίσουμε τι είναι Μεθοδολογία.

> Η **Μεθοδολογία** είναι ένα σύνολο μεθόδων, διαδικασιών και κανόνων για έναν συγκεκριμένο κλάδο. Περιλαμβάνει τη θεωρητική ανάλυση του συνόλου των μεθόδων και των αρχών που σχετίζονται με έναν κλάδο της γνώσης. Οι μεθοδολογίες προσφέρουν τη θεωρητική βάση για την κατανόηση της μεθόδου, του συνόλου μεθόδων ή των βέλτιστων πρακτικών που μπορούν να εφαρμοστούν σε μια συγκεκριμένη περίπτωση.

Αν και το Agile δεν αποτελεί μεθοδολογία, υπάρχουν πολλές μεθοδολογίες που μπορούν να χρησιμοποιήσουν οι ομάδες για να ακολουθήσουν τις αρχές και τις αξίες του Agile. Για να ορίσουν καλύτερα τη μεθοδολογία Agile, επαγγελματίες λογισμικού από διάφορα υπόβαθρα συγκεντρώθηκαν για να σχηματίσουν την Agile Alliance που δημιούργησε το Agile Μανιφέστο. Το Agile Μανιφέστο αποτελείται από τέσσερις θεμελιώδεις αξίες και δώδεκα υποστηρικτικές αρχές που καθοδηγούν την προσέγγιση Agile στην ανάπτυξη λογισμικού. Κάθε πλαίσιο Agile εφαρμόζει τις τέσσερις αξίες με διαφορετικούς τρόπους, αλλά όλες βασίζονται σε αυτές για να καθοδηγήσουν την ανάπτυξη και την παράδοση λογισμικού υψηλής ποιότητας που λειτουργεί.

Στο επόμενο κεφάλαιο θα παρουσιάσουμε το Agile Μανιφέστο όπως και την ιστορία πίσω από αυτό.

2.1. Το Agile Μανιφέστο

Πριν από σχεδόν 21 χρόνια (11-13 Φεβρουαρίου 2001), 17 προγραμματιστές λογισμικού συγκεντρώθηκαν στο Snowbird της Γιούτα για να προτείνουν έναν νέο τρόπο ανάπτυξης λογισμικού. Μέσω αυτής της εργασίας, οι υπογράφοντες του Μανιφέστου κατάλαβαν πόσο μεγάλο αντίκτυπο είχε η ανάπτυξη αυτών των αρχών στον τομέα της ανάπτυξης λογισμικού — αλλά δεν είχαν ιδέα πόσο γρήγορα οι ιδέες τους θα εξαπλωθούν πέρα από τον κλάδο τους. Το σύνολο των αξιών και των αρχών περιγράφονται σε αυτό το κεφάλαιο.

2.1.1. Οι τέσσερις αξίες του Agile Μανιφέστου

Όπως αναφέραμε και σε προηγούμενο κεφάλαιο, το Agile Μανιφέστο αποτελείται από τέσσερις θεμελιώδεις αξίες και δώδεκα υποστηρικτικές αρχές. Σε αυτό το κεφάλαιο θα περιγράψουμε τις τέσσερις θεμελιώδεις αξίες του Agile Μανιφέστου.

1. Τα άτομα και οι αλληλεπιδράσεις είναι πιο σημαντικά από τις διαδικασίες και τα εργαλεία

Η πρώτη αξία στο Agile Μανιφέστο είναι "Άτομα και αλληλεπιδράσεις πάνω από διαδικασίες και εργαλεία". Είναι εύκολο να γίνει κατανοητό το να εκτιμάτε τους ανθρώπους περισσότερο από τις διαδικασίες ή τα εργαλεία, επειδή είναι οι άνθρωποι που ανταποκρίνονται στις επιχειρηματικές ανάγκες και καθοδηγούν τη διαδικασία ανάπτυξης. Εάν η διαδικασία ή τα εργαλεία καθοδηγούν την ανάπτυξη, η ομάδα ανταποκρίνεται λιγότερο στις αλλαγές και είναι λιγότερο πιθανό να καλύψει τις ανάγκες των πελατών. Η επικοινωνία είναι ένα παράδειγμα της διαφοράς μεταξύ της αξίας των ατόμων έναντι αυτής των διαδικασιών. Στην περίπτωση των ατόμων, η επικοινωνία είναι ρευστή και συμβαίνει όταν προκύπτει ανάγκη. Στην περίπτωση της διαδικασίας, η επικοινωνία είναι προγραμματισμένη και απαιτεί συγκεκριμένο περιεχόμενο.

2. Ένα πλήρως λειτουργικό λογισμικό είναι πιο σημαντικό από την πλήρη τεκμηρίωση για τον τρόπο λειτουργίας αυτού

Ιστορικά, παρατηρήθηκε πως έχει δαπανήθει τεράστιος χρόνος για την τεκμηρίωση του προϊόντος για την ανάπτυξή του και για την τελική παράδοση. Τεχνικές προδιαγραφές, τεχνικές απαιτήσεις, τεχνικό ενημερωτικό δελτίο, έγγραφα σχεδιασμού διεπαφής, σχέδια δοκιμών, σχέδια τεκμηρίωσης και εγκρίσεις που απαιτούνται για το καθένα από αυτά. Ο κατάλογος ήταν εκτενής και ήταν η αιτία για τις μεγάλες καθυστερήσεις στην ανάπτυξη και συνεπώς στην τελική παράδοση του τελικού προϊόντος. Το Agile δεν καταργεί σε καμία περίπτωση την τεκμηρίωση, αλλά τη βελτιστοποιεί με μια μορφή που δίνει στον προγραμματιστή ό,τι χρειάζεται για να κάνει την εργασία χωρίς να κολλήσει σε μικρολεπτομέρειες. Το Agile τεκμηριώνει τις απαιτήσεις ως ιστορίες χρηστών, οι οποίες επαρκούν για έναν προγραμματιστή λογισμικού για να ξεκινήσει το έργο της δημιουργίας μιας νέας λειτουργίας.

Το Agile Μανιφέστο εκτιμά την τεκμηρίωση, αλλά εκτιμά περισσότερο το λειτουργικό λογισμικό εργασίας που αποτελεί και ο τελικός και πιο ουσιαστικός στόχος των ομάδων ανάπτυξης λογισμικού.

3. Συνεργασία πελατών είναι πιο σημαντική από τις διαπραγματεύσεις συμβολαίων

Η διαπραγμάτευση είναι η περίοδος κατά την οποία ο πελάτης και ο υπεύθυνος προϊόντος επεξεργάζονται τις λεπτομέρειες μιας παράδοσης, με σημεία κατά μήκος της διαδρομής που ακολουθεί ο τελικός χρήστης όπου οι λεπτομέρειες μπορούν να επαναδιαπραγματευθούν. Η συνεργασία είναι ένα εντελώς διαφορετικό θέμα. Με μοντέλα ανάπτυξης όπως το Waterfall, οι πελάτες διαπραγματεύονται τις απαιτήσεις για το προϊόν, συχνά με μεγάλη λεπτομέρεια, πριν ξεκινήσουν οποιαδήποτε εργασία. Αυτό σήμαινε ότι ο πελάτης συμμετείχε στη διαδικασία ανάπτυξης πριν ξεκινήσει η ανάπτυξη και μετά την ολοκλήρωσή της, αλλά όχι κατά τη διάρκεια της διαδικασίας. Το Agile Μανιφέστο περιγράφει έναν πελάτη που ασχολείται και συνεργάζεται σε όλη τη διαδικασία ανάπτυξης με την ομάδα ανάπτυξης. Αυτό μειώνει το ρίσκο να μην καλυφθούν οι ανάγκες του πελάτη

από την ανάπτυξη λογισμικού. Έτσι οι ευέλικτες μέθοδοι μπορούν να περιλαμβάνουν τον πελάτη κατά διαστήματα κατά την σχεδίαση αλλά και την υλοποίηση για περιοδικές επιδείξεις, αλλά ένα έργο θα μπορούσε εξίσου εύκολα να έχει έναν τελικό χρήστη ως καθημερινό μέρος της ομάδας ο οποίος θα παρακολουθεί όλες τις συναντήσεις, διασφαλίζοντας ότι το προϊόν ανταποκρίνεται στις επιχειρηματικές ανάγκες του πελάτη.

4. Η ανταπόκριση στην αλλαγή είναι πιο σημαντική από την εκτέλεση ενός προδιαγεγραμμένου σχεδίου

Η παραδοσιακή ανάπτυξη λογισμικού θεωρούσε την αλλαγή ως έξοδο, επομένως έπρεπε να αποφευχθεί. Η πρόθεση ήταν να αναπτυχθούν λεπτομερή, περίτεχνα σχέδια, με ένα καθορισμένο σύνολο χαρακτηριστικών με μεγάλο αριθμό εξαρτήσεων οι οποίες παραδίδονται με μια συγκεκριμένη σειρά, έτσι ώστε η ομάδα να μπορεί δουλέψτε το επόμενο κομμάτι του παζλ.

Με το Agile, η σύντομη διάρκεια μιας επανάληψης σημαίνει ότι οι προτεραιότητες μπορούν να μετατοπιστούν από επανάληψη σε επανάληψη και μπορούν να προστεθούν νέα χαρακτηριστικά στην επόμενη επανάληψη. Η άποψη του Agile είναι ότι οι αλλαγές πάντα βελτιώνουν ένα έργο. Οι αλλαγές παρέχουν πρόσθετη αξία.

Οι μεθοδολογίες Agile επιτρέπουν στις ομάδες ανάπτυξης λογισμικού να τροποποιήσουν τις διαδικασίες και να τις κάνουν να ταιριάζουν στην ομάδα και όχι το αντίστροφο. Έτσι λοιπόν σε αυτό το σημείο είναι σημαντικό να αναφέρουμε ότι δύο ομάδες που ακολουθούν την ίδια μέθοδο ανάπτυξης λογισμικού ακόμα και στον ίδιο οργανισμό μπορεί να λειτουργούν εντελώς διαφορετικά η μια από την άλλη, αρκεί να παραδίδουν αποτελεσματικά και ποιοτικά.

2.1.2. Οι δώδεκα αρχές του Agile Μανιφέστου

Οι Δώδεκα Αρχές είναι οι κατευθυντήριες αρχές για τις μεθοδολογίες που περιλαμβάνονται στον τίτλο «The Agile Movement». Περιγράφουν μια κουλτούρα στην οποία η αλλαγή είναι ευπρόσδεκτη και ο πελάτης είναι το επίκεντρο της εργασίας. Επιδεικνύουν επίσης την πρόθεση του κινήματος όπως περιγράφεται από τον Alistair Cockburn, έναν από τους υπογράφοντες στο Agile Μανιφέστο, το οποίο είναι να φέρει την ανάπτυξη σε ευθυγράμμιση με τις επιχειρηματικές ανάγκες.

Οι δώδεκα αρχές της ευέλικτης ανάπτυξης περιλαμβάνουν:

1. Ικανοποίηση πελατών μέσω της έγκαιρης και συνεχούς παράδοσης λογισμικού – Οι πελάτες είναι πιο χαρούμενοι όταν λαμβάνουν λειτουργικό λογισμικό σε τακτά χρονικά διαστήματα, αντί να περιμένουν μεγάλες χρονικές περιόδους μεταξύ των εκδόσεων.
2. Δεχθείτε τις μεταβολές στις απαιτήσεις σε όλη τη διαδικασία ανάπτυξης – Η δυνατότητα αποφυγής καθυστερήσεων όταν αλλάζει μια απαίτηση ή ένα αίτημα για υλοποίηση.
3. Συχνή παράδοση λειτουργικού λογισμικού – Το Scrum για παράδειγμα το οποίο αποτελεί ένα Ευέλικτο πλαίσιο ανάπτυξης λογισμικού τηρεί αυτήν την αρχή, καθώς η ομάδα δραστηριοποιείται σε sprints ή επαναλήψεις λογισμικού που διασφαλίζουν την τακτική παράδοση του λειτουργικού λογισμικού.
4. Συνεργασία μεταξύ των μετόχων της επιχείρησης και των προγραμματιστών καθ' όλη τη διάρκεια του έργου – Οι καλύτερες αποφάσεις λαμβάνονται όταν η επιχειρηματική και η

τεχνική ομάδα είναι ευθυγραμμισμένες καθώς οι αποφάσεις λαμβάνονται ή και επικοινωνούντα έγκαιρα μειώνοντας τον χρόνο προσαρμογής των ομάδων ανάπτυξης.

5. Υποστηρίξτε, εμπιστευτείτε και παρακινήστε τα άτομα που απαρτίζουν την ομάδα ανάπτυξης λογισμικού – Οι ομάδες με κίνητρα είναι πιο πιθανό να είναι πιο αποδοτικές και να κάνουν καλύτερα την δουλεία τους από τις ομάδες που δεν είναι χαρούμενες.
6. Ενεργοποιήστε τις αλληλεπιδράσεις πρόσωπο με πρόσωπο – Η επικοινωνία είναι πιο επιτυχημένη όταν οι ομάδες ανάπτυξης συστεγάζονται.
7. Το λογισμικό εργασίας είναι το κύριο μέτρο προόδου – Η παράδοση λειτουργικού λογισμικού στον τελικό πελάτη είναι ο απόλυτος παράγοντας που μετρά την πρόοδο.
8. Ευέλικτες διαδικασίες για την υποστήριξη ενός σταθερού ρυθμού ανάπτυξης – Οι ομάδες δημιουργούν μια επαναλαμβανόμενη και διατηρήσιμη ταχύτητα με την οποία μπορούν να παραδώσουν λειτουργικό λογισμικό και το επαναλαμβάνουν με κάθε νέα εκδοχή.
9. Η προσοχή στις τεχνικές λεπτομέρειες και το σχέδιο ενισχύουν την ευελιξία – Οι σωστές δεξιότητες και ο καλός σχεδιασμός διασφαλίζουν ότι η ομάδα μπορεί να διατηρήσει το ρυθμό, να βελτιώνει συνεχώς το προϊόν και να διατηρήσει την αλλαγή.
10. Απλότητα – Αναπτύξτε τόσα όσα είναι αρκετά για να ολοκληρώσετε τη δουλειά αυτή τη δεδομένη χρονική στιγμή.
11. Οι αυτοοργανωμένες ομάδες παραδίδουν εξαιρετικές αρχιτεκτονικές, απαιτήσεις και σχέδια – Τα ικανά και με κίνητρα μέλη της ομάδας που έχουν δύναμη λήψης αποφάσεων, αναλαμβάνουν την ευθύνη, επικοινωνούν τακτικά με άλλα μέλη της ομάδας και μοιράζονται ιδέες που προσφέρουν ποιοτικά προϊόντα.
12. Τακτικοί προβληματισμοί για το πώς να γίνει η ομάδα πιο αποτελεσματική – Η αυτοβελτίωση, η βελτίωση της διαδικασίας, η προώθηση δεξιοτήτων και τεχνικών, βοηθούν τα μέλη της ομάδας να εργάζονται πιο αποτελεσματικά.

Η πρόθεση του Agile είναι να ευθυγραμμίσει την ομάδα ανάπτυξης με τις επιχειρηματικές ανάγκες, έτσι η επιτυχία του Agile θα μπορεί να είναι εμφανής. Τα ευέλικτα έργα είναι πελατοκεντρικά και ενθαρρύνουν την καθοδήγηση και τη συμμετοχή των πελατών. Ως αποτέλεσμα, το Agile έχει εξελιχθεί σε μια γενική άποψη της ανάπτυξης λογισμικού σε όλη τη βιομηχανία λογισμικού και σε μια βιομηχανία από μόνη της.

2.2. Agile Πλαίσια Ανάπτυξης Λογισμικού

Το Agile αντιπροσωπεύει μια γενική φιλοσοφία για την ανάπτυξη λογισμικού, δίνοντας έμφαση στην αξία των συχνών επαναλήψεων για την ικανοποίηση των πελατών. Ως εκ τούτου, ένα Agile πλαίσιο μπορεί να οριστεί ως μια συγκεκριμένη προσέγγιση ανάπτυξης λογισμικού που βασίζεται στην Agile φιλοσοφία που διατυπώνεται στο Agile Μανιφέστο.

Συνήθως αναφέρονται τα Agile πλαίσια ανάπτυξης ως μεθοδολογίες ή ακόμα και διαδικασίες. Όμως όπως αναφέραμε και στο προηγούμενο κεφάλαιο, μια από τις τέσσερις αξίες του μανιφέστου αναφέρει πως η φιλοσοφία έχει την μεγαλύτερη βαρύτητα «Άτομα και αλληλεπιδράσεις πάνω από διαδικασίες και εργαλεία». Οι περισσότερες Agile ομάδες χρησιμοποιούν τα Agile πλαίσια ως αφετηρία για τον μετασχηματισμό τους και τελικά προσαρμόζουν τα στοιχεία για να καλύψουν τις ανάγκες τους.

Πολλοί σύγχρονοι οργανισμοί χρησιμοποιούν τα πιο δημοφιλή πλαίσια ανάπτυξης, τα οποία τροποποιούν και επαναλαμβάνουν τις δικές τους ευέλικτες διαδικασίες. Παρακάτω, σε αυτό το

κεφάλαιο παρουσιάζονται, συνοπτικά, τα πιο κοινά χρησιμοποιούμενα και καλά τεκμηριωμένα πλαίσια για ευέλικτη ανάπτυξη λογισμικού. Αναφορικά αυτά είναι:

- Scrum Framework
- Kanban Framework
- Lean Software Development (LSD)
- eXtreme Programming (XP)
- The Crystal Method
- Dynamic Systems Development Method (DDSM)
- Feature Driven Development (FDD)
- Adaptive Software Development (ASD)

Εικόνα 2: *Agile Πλαίσια Ανάπτυξης Λογισμικού [24].*

2.2.1. Scrum Πλαίσιο Ανάπτυξης Λογισμικού

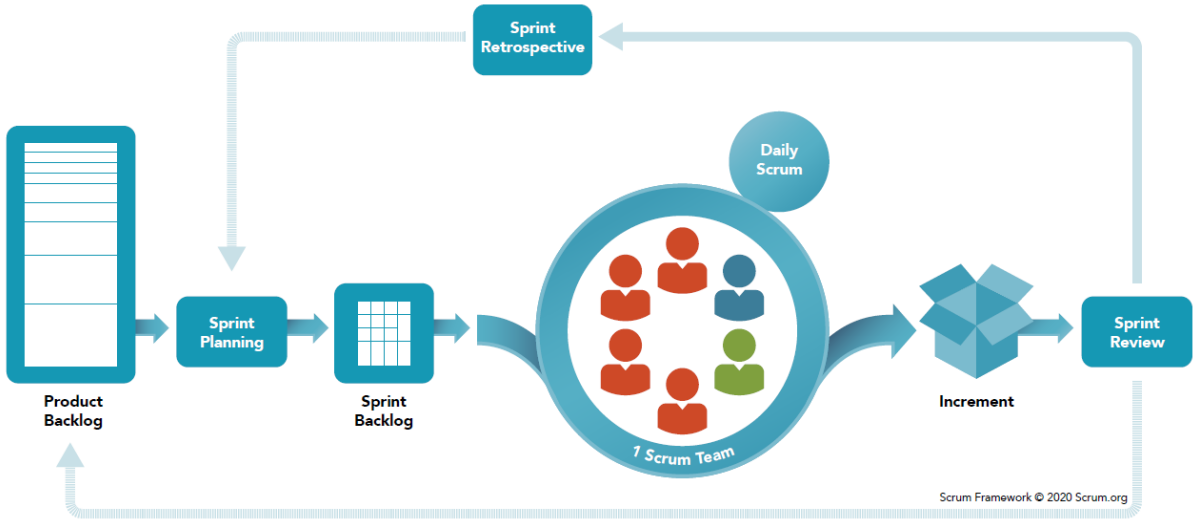
Το Scrum είναι ένα πλαίσιο για την ανάπτυξη, την παράδοση και τη συντήρηση σύνθετων προϊόντων το οποίο βασίζεται στον Scrum οδηγό [1] του οποίου οι δημιουργοί Ken Shwaber και Jeff Sutherland έγραψαν για να εξηγήσουν το πλαίσιο ξεκάθαρα και συνοπτικά. Στο Scrum οι άνθρωποι μπορούν να αντιμετωπίσουν σύνθετα, προσαρμοστικά προβλήματα με σκοπό να προσφέρουν προϊόντα της υψηλότερης δυνατής αξίας με παραγωγικό και δημιουργικό τρόπο. Κύριος σκοπός του Scrum είναι η ανάδειξη ενός λειτουργικού λογισμικού, μέσω της ευελιξίας σε συνδυασμό με την διαφαινόμενη επικοινωνία, τη συνεργασία και τις επιχειρηματικές πραγματικότητες.

Το Scrum βασίστηκε στην εμπειρική θεωρία ελέγχου διεργασιών, η οποία είναι επίσης γνωστή ως Εμπειρισμός. Ο Εμπειρισμός λέει ότι η γνώση κερδίζεται από την εμπειρία στη λήψη αποφάσεων με βάση αυτά που γνωρίζουμε. Η κύρια βάση του Scrum αποτελούν ο έλεγχος του κινδύνου και η βελτιστοποίηση της προβλεψιμότητας. Οι τρεις πυλώνες που υποστηρίζουν την κάθε εφαρμογή του εμπειρικού ελέγχου της διαδικασίας είναι: η διαφάνεια, η επιθεώρηση και η προσαρμογή.

Οι κύριοι διακριτοί ρόλοι του Scrum που έχουν οριστεί σε αυτό το πλαίσιο ανάπτυξης είναι:

1. Η **ομάδα των Προγραμματιστών (DEV team) α**που αποτελείται από άτομα που εργάζονται μαζί στα Sprints για την επίτευξη του στόχου του έργου.
2. Την/ον **Scrum Master** που διασφαλίζει ότι η ομάδα ακολουθεί τους κανόνες του Scrum.
3. Την/ον **Product Owner** που εκπροσωπεί τον πελάτη, δίνει προτεραιότητα στις καθυστερήσεις και συντονίζει τις προσπάθειες της ομάδας Scrum.

Έτσι λοιπόν για να συνοψίσουμε, το Scrum είναι μια επαναληπτική διαδικασία η οποία έχει συγκεκριμένη διάρκεια και ονομάζεται Sprint (συνήθως δύο εβδομάδες) κατά την οποία παίρνουν μέρος γεγονότα με προκαθορισμένη σειρά ώστε να μειωθεί η πολυπλοκότητα και να βοηθήσει τις Scrum Ομάδες να προσφέρουν προϊόντα της υψηλότερης δυνατής αξίας με παραγωγικό και δημιουργικό τρόπο. Τα γεγονότα όπως και τα τεχνουργήματα που χρησιμοποιούνται στο Scrum παρουσιάζονται στην Εικόνα 3 όπως και περιγράφονται αναλυτικότερα σε αυτό το κεφάλαιο.



Εικόνα 3: *The Scrum flow [1].*

<u>Γεγονότα (events) του Scrum πλαισίου ανάπτυξης:</u>

1. **Sprint:** Τα Sprints μπορούν να χαρακτηριστούν ως η "καρδία" του Scrum μιας και όλη η εργασία που απαιτείται για την επίτευξη του στόχου προϊόντος, συμπεριλαμβανομένων των Sprint Planning, των Daily Scrums, του Sprint Review και του Sprint Retrospective, πραγματοποιούνται κατά την διάρκεια ενός Sprint. Τα Sprints στο Scrum έχουν σταθερή διάρκεια, συνήθως δύο εβδομάδες, για την δημιουργία συνέπειας. Ένα νέο Sprint ξεκινά αμέσως μετά την ολοκλήρωση του προηγούμενου.
2. **Sprint Planning:** Το Sprint Planning εκκινεί το Sprint παρουσιάζοντας τις εργασίες που θα εκτελεστούν για αυτό. Το σχέδιο δράσης που προκύπτει, δημιουργείται από τη συλλογική

εργασία ολόκληρης της Scrum Ομάδας. Ο Product Owner διασφαλίζει ότι οι συμμετέχοντες είναι προετοιμασμένοι να συζητήσουν τα πιο σημαντικά στοιχεία του Product Backlog και τον τρόπο με τον οποίο αντιστοιχίζονται στον στόχο του προϊόντος. Τα κύρια θέματα που καλύπτονται κατά την διάρκεια αυτού του γεγονότος περιλαμβάνουν:

i. Γιατί είναι πολύτιμες οι εργασίες αυτού του Sprint;
ii. Τι μπορεί να παραδοθεί σε αυτό το Sprint;
iii. Πως θα υλοποιηθεί και θα παραδοθεί το επιλεγμένο σύνολο της εργασίας;
iv. Ποιος είναι ο κοινός μας στόχος σαν ομάδα που καλούμαστε να πετύχουμε σε αυτό το Sprint;

Με αυτό τον τρόπο επιτυγχάνεται ο ευθυγραμμισμός της ομάδας ανάπτυξης και η κοινή στοχοθέτηση. Ο Product Owner είναι ο μοναδικός ο οποίος μπορεί να αποφασίσει εάν θα μπουν επιπλέον εργασίες ή θα αφαιρεθούν εργασίες από το Sprint ή σε ακραίες περιπτώσεις εάν θα ακυρωθεί ή αλλάξει εντελώς ένα Sprint. Σημείωση: Η προσθαφαίρεση εργασιών από το Sprint φυσικά μπορεί να επηρεάσει και τα τελικά παραδοτέα που σε αυτή την περίπτωση ο Scrum Master όπως και κάθε μέλος της ομάδας είναι εκεί για να το υπενθυμίσουν στον Product Owner και να προστατεύσουν την ομάδα από τέτοιου είδους αλλαγές.

2. **Daily Scrum:** Ο σκοπός του Daily Scrum είναι η επιθεώρηση της προόδου της ομάδας ανάπτυξης προς τον στόχο του Sprint και η προσαρμογή του Sprint Backlog για το σύνολο της του επόμενου Sprint. Το Daily Scrum είναι μια συνάντηση διάρκειας 15 λεπτών για την Scrum Ομάδα. Για την μείωση της πολυπλοκότητας, το γεγονός αυτό πραγματοποιείται την ίδια ώρα και στο ίδιο μέρος κάθε εργάσιμη ημέρα του Sprint. Τα Daily Scrums βελτιώνουν την επικοινωνία, βοηθούν στον εντοπισμό εμποδίων, προωθούν τη γρήγορη λήψη αποφάσεων και, κατά συνέπεια, εξαλείφουν την ανάγκη για άλλες συναντήσεις. Κατά την διάρκεια αυτού του event, δίνεται ο λόγος και ο χρόνος σε όλα τα μέλη της ομάδας για να αναδείξουν τους προβληματισμούς τους σχετικά με τις εργασίες του Sprint και να ζητήσουν βοήθεια αλλά και για να προγραμματίσουν τις εργασίες της ημέρας.
3. **Sprint Review:** Ο σκοπός του Sprint Review είναι η επιθεώρηση του αποτελέσματος του Sprint και ο καθορισμός μελλοντικών προσαρμογών του Product Backlog. Η Scrum Ομάδα παρουσιάζει τα αποτελέσματα της δουλειάς της σε βασικούς ενδιαφερόμενους και συζητείται η πρόοδος προς τον στόχο του προϊόντος. Κατά τη διάρκεια του Sprint Review, η Scrum ομάδα και οι ενδιαφερόμενοι αξιολογούν τι επιτεύχθηκε στο Sprint και τι έχει αλλάξει στο περιβάλλον τους. Με βάση αυτές τις πληροφορίες, οι συμμετέχοντες συνεργάζονται για το τι πρέπει να κάνουν στη συνέχεια.
4. **Sprint Retrospective:** Ο σκοπός του Sprint Retrospective είναι ο σχεδιασμός τρόπων για την αύξηση της ποιότητας των παραδοτέων και της αποτελεσματικότητας της ομάδας. Η Scrum Ομάδα επιθεωρεί πώς πήγε το τελευταίο Sprint όσον αφορά τα άτομα, τις αλληλεπιδράσεις, τις διαδικασίες, τα εργαλεία κ.α. Τα στοιχεία που ελέγχονται συχνά διαφέρουν ανάλογα με τον τομέα εργασίας. Οι υποθέσεις που τους παρέσυραν εντοπίζονται, διερευνάται η προέλευσή τους και ορίζονται ενέργειες για την αντιμετώπισή τους. Επίσης, η Scrum ομάδα εντοπίζει τις πιο χρήσιμες αλλαγές για τη βελτίωση της αποτελεσματικότητάς της. Το Sprint Retrospective ολοκληρώνει το Sprint. Μέχρι το τέλος του Sprint Retrospective, η Scrum Ομάδα θα πρέπει να έχει εντοπίσει βελτιώσεις που θα εφαρμόσει στο επόμενο Sprint. Η εφαρμογή αυτών των βελτιώσεων στο επόμενο Sprint είναι η προσαρμογή στην επιθεώρηση της ίδιας της Scrum ομάδας ως προς τον τρόπο που λειτουργεί ως ανεξάρτητη οντότητα εντός του οργανισμού. Παρόλο που μπορεί να πραγματοποιηθούν βελτιώσεις ανά πάσα στιγμή, το Sprint Retrospective αποτελεί μια επίσημη ευκαιρία εστίασης στην επιθεώρηση και την προσαρμογή.

<u>Τεχνουργήματα (artifacts) του Scrum πλαισίου ανάπτυξης:</u>

1. **Product Backlog:** Το Product Backlog είναι μια ταξινομημένη λίστα με το σύνολο των εργασιών που θα συμβάλουν στην βελτίωση του τελικού προϊόντος και αποτελεί η μοναδική πηγή εργασίας που ανατίθεται στην Scrum Ομάδα. Τα στοιχεία του Product Backlog που μπορούν να γίνουν από την Scrum Ομάδα μέσα σε ένα Sprint, θεωρούνται έτοιμα και αναλύονται κατά το Sprint Planning. Η βελτιστοποίηση του Product Backlog είναι η πράξη της ανάλυσης και περαιτέρω προσδιορισμού των στοιχείων του σε μικρότερα και ακριβέστερα στοιχεία. Αυτή είναι μια συνεχής δραστηριότητα για την προσθήκη λεπτομερειών, όπως η περιγραφή, τα επίπεδα προτεραιότητας και το μέγεθος. Τα χαρακτηριστικά ποικίλλουν συχνά ανάλογα με τον τομέα εργασίας. Οι προγραμματιστές που θα κάνουν την εργασία είναι υπεύθυνοι για το μέγεθος. Ο Product Owner μπορεί να επηρεάσει τους προγραμματιστές βοηθώντας τους να κατανοήσουν και να επιλέξουν τις προτεραιότητες.
2. **Sprint Backlog:** Το Sprint Backlog αποτελείται από το Sprint goal, τον στόχο του Sprint ο οποίος περιγράφει το γιατί. Γιατί επιλέχτηκαν αυτά τα αντικείμενα από το Product Backlog για υλοποίηση. Τα αντικείμενα προς υλοποίηση (Ιστορίες Χρηστών ή User Stories) που περιγράφουν το τι. Καθώς και ένα σχέδιο δράσης για την παράδοση του Product Increment που περιγράφει το πώς. Το Sprint Backlog είναι ένα σχέδιο από και προς τους προγραμματιστές της ομάδας. Είναι μια εξαιρετικά ορατή εικόνα σε πραγματικό χρόνο της δουλειάς που οι προγραμματιστές σχεδιάζουν να ολοκληρώσουν κατά τη διάρκεια του Sprint προκειμένου να επιτύχουν τον στόχο της Sprint. Κατά συνέπεια, το Sprint Backlog ενημερώνεται σε όλο το Sprint καθώς μαθαίνονται περισσότερα.
3. **Product Increment:** Το παράγωγο ενός Sprint είναι η προσαύξηση του προϊόντος ή Product Increment. Ένα Product Increment είναι ένα συγκεκριμένο σκαλοπάτι προς τον συνολικό στόχο του τελικού προϊόντος. Κάθε νέο Increment θα πρέπει να είναι χρησιμοποιήσιμο και αποτελεί ένα προσθετικό σε όλα τα προηγούμενα το οποίο επαληθεύεται διεξοδικά κατά την διάρκεια του Sprint από την ομάδα ανάπτυξης, διασφαλίζοντας ότι όλα λειτουργούν όπως πρέπει μαζί. Αξίζει να σημειωθεί πως η ομάδα ανάπτυξης μπορεί να δημιουργήσει πολλαπλά Increments σε ένα Sprint. Το σύνολο των Increments που παραδίδονται κατά την διάρκεια του Sprint παρουσιάζεται από την ομάδα ανάπτυξης στο Sprint Review, υποστηρίζοντας έτσι τον εμπειρισμό. Ωστόσο, ένα Increment μπορεί να παραδοθεί στους ενδιαφερόμενους πριν από το τέλος του Sprint για την συλλογή σχόλιων και για την υποστήριξη της ανατροφοδότησης.

<u>Τα κύρια χαρακτηριστικά του Scrum:</u>

Τα κύρια χαρακτηριστικά του Scrum αποτελούν η κύρια αιτία που το κάνανε τόσο δημοφιλές. Πιο συγκεκριμένα:

Το Scrum είναι ένα **ελαφρύ πλαίσιο ανάπτυξης** το οποίο αποτελείται από κανόνες και πρακτικές που είναι λίγοι σε αριθμό και εύκολο να ακολουθηθούν.

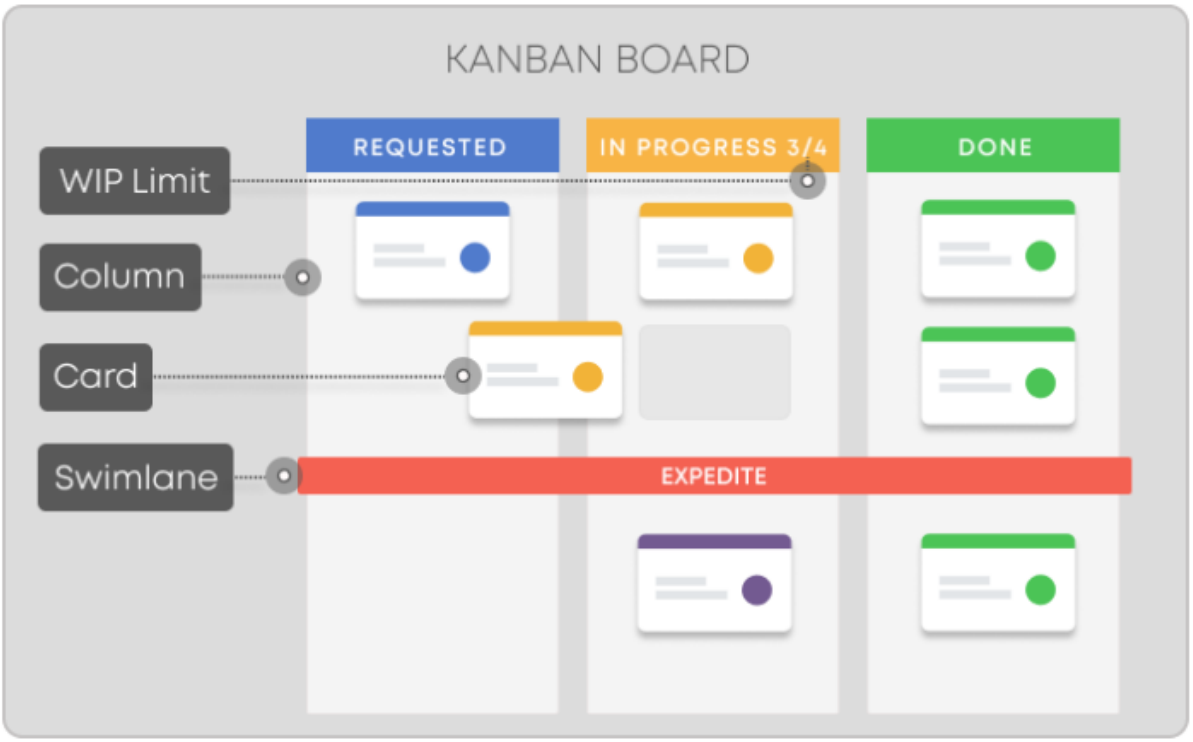
Το Scrum είναι **απλό στην κατανόηση**, πράγμα που το καθιστά εύκολο στην κατανόηση σαν πλαίσιο ανάπτυξης για όλους.

Επειδή υπάρχει μεγάλη διαφορά μεταξύ της κατανόησης του Scrum και της εφαρμογής του σε πραγματικά έργα, είναι **δύσκολο** αυτοί που το χρησιμοποιούν **να το κατακτήσουν και να το ακολουθούν σωστά**. Για να γίνει αυτό θα πρέπει όσο εργάζονται σε αυτό το πλαίσιο να

κατανοήσουνε σωστά γιατί κάθε στοιχείο είναι στη θέση του, πώς αποδίδει αξία ως διακριτό στοιχείο όπως επίσης και σε σχέση με τα άλλα στοιχεία του.

2.2.2. Kanban

Το Kanban είναι μια δημοφιλής μέθοδος διαχείρισης ροής εργασιών για τον καθορισμό, τη διαχείριση και τη βελτίωση των υπηρεσιών ενός οργανισμού. Βοηθά στην οπτικοποίηση της εργασίας, στην μεγιστοποίηση της αποτελεσματικότητας και της συνεχούς βελτίωσης. Η εργασία αποτυπώνεται σε πίνακες Kanban (Εικόνα 4), επιτρέποντάς την βελτιστοποιήση της παράδοσης εργασίας και της διαφάνειας σε πολλές ομάδες για τη διαχείρισης ακόμη και των πιο σύνθετων έργων σε ένα ενιαίο περιβάλλον.



Εικόνα 4: *Kanban Board [25].*

Η ιαπωνική λέξη "kanban", που σημαίνει "οπτικός πίνακας" ή "σημάδι", χρησιμοποιείται ως ο ορισμός της έννοιας της διαδικασίας από τη δεκαετία του 1950. Η διαδικασία αυτή αναπτύχθηκε και εφαρμόστηκε για πρώτη φορά από την Toyota ως σύστημα χρονοπρογραμματισμού για την έγκαιρη κατασκευή. Από την άλλη πλευρά, ο όρος "Kanban" με κεφαλαία γράμματα είναι γνωστός και σχετίζεται με την εμφάνιση της "Μεθόδου Kanban", η οποία ορίστηκε για πρώτη φορά το 2007.

Η Μέθοδος Kanban ακολουθεί ένα σύνολο αρχών και πρακτικών για τη διαχείριση και τη βελτίωση των ροών εργασίας. Είναι μια εξελικτική, μη διασπαστική μέθοδος που προωθεί σταδιακές βελτιώσεις στις διαδικασίες ενός οργανισμού. Εάν ακολουθούνται αυτές τις αρχές και πρακτικές, τότε το Kanban θα χρησιμοποιείται για τη μεγιστοποίηση των οφελών στις επιχειρηματικές διαδικασίες – βελτίωση ροών, μείωση του χρόνου παράδοσης, αύξηση της αξίας για τον πελάτη, μεγαλύτερη προβλεψιμότητα κλπ. Αντικείμενα ζωτικής σημασίας για κάθε επιχείρηση ή οργανισμό.

Οι τέσσερις θεμελιώδεις αρχές και οι έξι Βασικές Πρακτικές της Μεθοδολογίας Kanban παρουσιάζονται παρακάτω:

Οι τέσσερις θεμελιώδεις αρχές του Kanban:

Εκκινήστε τις εργασίες με την υπάρχουσα δομή: Η Μέθοδος Kanban τονίζει έντονα ότι δεν είναι απαραίτητο να γίνουν αμέσως αλλαγές στην υπάρχουσα εγκατάσταση/διαδικασία. Προτείνεται η εφαρμογή του Kanban απευθείας στην ροή εργασίας και οποιεσδήποτε αλλαγές απαιτούνται μπορούν να προκύψουν σταδιακά σε μια χρονική περίοδο με ρυθμό που η ομάδα αισθάνεται άνετα.

Συμφωνήστε να επιδιώξετε σταδιακή, εξελικτική αλλαγή: Το Kanban ενθαρρύνει τις ομάδες ανάπτυξης και τους οργανισμούς να κάνουν μικρές σταδιακές αλλαγές αντί να κάνουν ριζικές αλλαγές που μπορεί να οδηγήσουν σε αντίσταση. Η μέθοδος του Εμπειρισμού και της σταδιακής προσαρμογής των αλλαγών ανάλογα με τις ανάγκες της ομάδας ανάπτυξης με αυτό τον τρόπο υποστηρίζονται.

Αρχικά, σεβαστείτε τους υπάρχοντες ρόλους, τις ευθύνες και τους τίτλους εργασίας: Σε αντίθεση με άλλες μεθόδους, το Kanban δεν επιβάλλει οργανωτικές αλλαγές από μόνο του. Επομένως, δεν είναι απαραίτητο να γίνουν αλλαγές στους υπάρχοντες ρόλους και λειτουργίες που μπορεί να αποδίδουν καλά. Η ομάδα ανάπτυξης εντοπίζει και εφαρμόζει από κοινού τις αλλαγές που απαιτούνται. Αυτές οι τρεις αρχές βοηθούν τους οργανισμούς να ξεπεράσουν την τυπική συναισθηματική αντίσταση και τον φόβο της αλλαγής που συνήθως συνοδεύουν οποιεσδήποτε πρωτοβουλίες αλλαγής σε έναν οργανισμό.

Ενθαρρύνετε τις πράξεις ηγεσίας σε όλα τα επίπεδα: Το Kanban ενθαρρύνει τη συνεχή βελτίωση σε όλα τα επίπεδα του οργανισμού και λέει ότι οι ηγετικές πράξεις δεν πρέπει να προέρχονται μόνο από ανώτερα στελέχη. Άτομα σε όλα τα επίπεδα μπορούν να παρέχουν ιδέες και να επιδείξουν ηγετικό ρόλο στην εφαρμογή αλλαγών για τη συνεχή βελτίωση του τρόπου με τον οποίο προσφέρουν τα προϊόντα και τις υπηρεσίες τους.

Οι έξι Βασικές Πρακτικές του Kanban:

Οραματιστείτε τη ροή της εργασίας: Αυτό είναι το θεμελιώδες πρώτο βήμα για την υιοθέτηση και την εφαρμογή της μεθόδου Kanban. Είναι απαραίτητη η οπτικοποιήση – είτε σε φυσικό πίνακα είτε σε ηλεκτρονικό πίνακα Kanban, τα βήματα της διαδικασίας που χρησιμοποιείται αυτήν τη στιγμή για την παράδοση της εργασίας ή των υπηρεσιών. Ανάλογα με την πολυπλοκότητα της διαδικασίας και το μείγμα εργασίας (τους διαφορετικούς τύπους αντικειμένων εργασίας που εργάζονται οι ομάδες ανάπτυξης και παραδίδουν), ο πίνακας Kanban μπορεί να είναι πολύ απλός έως πολύ περίπλοκος. Μόλις οπτικοποιθεί η διαδικασία, τότε μπορεί να οπτικοποιθεί και η τρέχουσα εργασία που εκτελείται από την ομάδα ανάπτυξης.

Περιορίστε την εργασία σε εξέλιξη - Work in Progress (WIP): Ο περιορισμός της εργασίας σε εξέλιξη (WIP) είναι θεμελιώδης για την εφαρμογή του Kanban. Περιορίζοντας το WIP, ενθαρρύνεται η ομάδα ανάπτυξης να ολοκληρώσει την εργασία που έχει στη διάθεσή της πρώτα πριν αναλάβει νέα εργασία. Επομένως, οι εργασίες που βρίσκονται σε εξέλιξη πρέπει να ολοκληρωθούν και να επισημανθούν ότι έχουν ολοκληρωθεί. Αυτό δημιουργεί χωρητικότητα στο σύστημα, έτσι ώστε η ομάδα να μπορεί να ξεκινήσει νέα δουλειά. Αρχικά, μπορεί να μην είναι εύκολο να αποφασιστεί ποια θα πρέπει να είναι

τα WIP όρια. Στην πραγματικότητα, η εργασία μπορεί να ξεκινήσει χωρίς WIP όρια και να παρατηρηθεί η αρχική εργασία που είναι σε εξέλιξη καθώς η ομάδα ανάπτυξης χρησιμοποιεί το Kanban. Μόλις τα δεδομένα που συλλέγονται είναι επαρκή, τότε ορίζονται τα όρια WIP για κάθε στάδιο της ροής εργασιών (κάθε στήλη του πίνακα Kanban) ως ίσα με το μισό του μέσου όρου WIP.

Συνήθως, πολλές ομάδες ξεκινούν με όριο WIP από 1 έως 1,5 φορές τον αριθμό των ατόμων που εργάζονται σε ένα συγκεκριμένο στάδιο. Ο περιορισμός του WIP και η τοποθέτηση των ορίων WIP σε κάθε στήλη του πίνακα όχι μόνο βοηθά τα μέλη της ομάδας να ολοκληρώσουν πρώτα αυτό που κάνουν πριν αναλάβουν νέα πράγματα - αλλά επίσης ενημερώνουν τον πελάτη και τους άλλους ενδιαφερόμενους ότι υπάρχει περιορισμένη ικανότητα για εργασία για οποιοδήποτε ομάδα – και πρέπει να προγραμματίσουν προσεκτικά τι δουλειά που ζητούν από τις ομάδες ανάπτυξης να ολοκληρώσουν.

Διαχείριση ροής: Η διαχείριση και η βελτίωση της ροής είναι η ουσία του συστήματος Kanban αφού εφαρμοστούν πρώτα οι δύο πρώτες πρακτικές. Ένα σύστημα Kanban βοηθά στην διαχείριση της ροής επισημαίνοντας τα διάφορα στάδια της ροής εργασίας και την κατάσταση της εργασίας σε κάθε στάδιο. Ανάλογα με το πόσο καλά έχει οριστεί η ροή εργασίας και τα Όρια WIP, παρατηρείται είτε μια ομαλή ροή εντός των ορίων WIP είτε συσσωρεύονται οι εργασίες καθώς κάτι μένει στάσιμο και συνεπώς αρχίζει να συγκρατεί τη χωρητικότητα των υπόλοιπων. Όλα αυτά επηρεάζουν το πόσο γρήγορα περνά η εργασία από την αρχή έως το τέλος της ροής. Το Kanban βοηθά την ομάδα να αναλύσει το σύστημα και να κάνει προσαρμογές για τη βελτίωση της ροής, ώστε να μειωθεί ο χρόνος που απαιτείται για την ολοκλήρωση κάθε εργασίας.

Μια βασική πτυχή αυτής της διαδικασίας παρατήρησης της εργασίας και επίλυσης/εξάλειψης των σημείων συμφόρησης είναι η μελέτη των ενδιάμεσων σταδίων αναμονής και ο υπολογισμός του χρόνου όπου τα αντικείμενα εργασίας παραμένουν σε αυτά. Η μείωση του χρόνου που δαπανάται σε αυτά τα στάδια αναμονής είναι το κλειδί για τη μείωση του χρόνου παράδοσης – Cycle Time. Καθώς βελτιώνεται η ροή, η παράδοση της εργασίας της ομάδας γίνεται πιο ομαλή και προβλέψιμη. Καθώς γίνεται πιο προβλέψιμη, γίνεται πιο εύκολο για τον οργανισμό να αναλάβει αξιόπιστες δεσμεύσεις έναντι των πελατών του σχετικά με το πότε θα ολοκληρωθεί οποιαδήποτε εργασία κάνετε τους ενδιαφέρει. Η βελτίωση της αξιοπιστίας των ομάδων ανάπτυξης είναι και ο σκοπός του Kanban.

Κάντε σαφείς τις πολιτικές διαδικασίας: Ως μέρος της οπτικοποίησης της διαδικασίας, προτείνεται και ο ορισμός και η απεικόνιση των πολιτικών της ομάδας ανάπτυξης (κανόνες διαδικασίας ή κατευθυντήριες γραμμές) για το πώς ολοκληρώνει την εργασία της. Διατυπώνοντας σαφείς κατευθυντήριες γραμμές διαδικασίας, δημιουργείται μια κοινή βάση για όλους τους συμμετέχοντες ώστε να κατανοήσουν πώς να παραδώσουν οποιοδήποτε είδος εργασίας στο σύστημα. Οι πολιτικές μπορούν να είναι σε επίπεδο πίνακα Kanban, σε επίπεδο περιοχής ή ακόμα και στήλης. Μπορεί να είναι μια λίστα ελέγχου βημάτων που πρέπει να γίνουν για κάθε τύπο στοιχείου εργασίας, κριτήρια εισόδου-εξόδου για κάθε στήλη ή οτιδήποτε άλλο βοηθά τα μέλη της ομάδας να διαχειρίζονται σωστά τη ροή της εργασίας στον πίνακα. Παραδείγματα ρητών πολιτικών περιλαμβάνουν τον ορισμό του πότε ολοκληρώνεται μια εργασία (Definition of Done - DoD), την περιγραφή μεμονωμένων λωρίδων ή στηλών κ.λπ. Οι πολιτικές πρέπει να ορίζονται ρητά και απεικονίζονται συνήθως στην κορυφή του πίνακα και σε κάθε περιοχή και στήλη.

Εφαρμογή βρόχων ανατροφοδότησης: Οι βρόχοι ανάδρασης αποτελούν αναπόσπαστο μέρος κάθε καλού συστήματος. Η Μέθοδος Kanban ενθαρρύνει και βοηθά στην εφαρμογή βρόχων σχολίων διαφόρων ειδών – στάδια ανασκόπησης στη ροή εργασίας του πίνακα Kanban, μετρήσεις και αναφορές και μια σειρά οπτικών ενδείξεων που παρέχουν συνεχή ανατροφοδότηση σχετικά με την πρόοδο της εργασίας –ή την έλλειψή της– στο σύστημά. Η ιδέα της έγκαιρης λήψης σχολίων είναι ζωτικής σημασίας για την παράδοση του τελικού παραδοτέου στον πελάτη στο συντομότερο δυνατό χρόνο. Οι βρόχοι ανάδρασης είναι κρίσιμοι για τη διασφάλιση αυτού.

Συνεργατική Βελτίωση, Πειραματική Εξέλιξη: Η μέθοδος Kanban είναι μια εξελικτική διαδικασία βελτίωσης. Βοηθά στην υιοθέτηση μικρών αλλαγών και την σταδιακή βελτίωση που μπορεί να χειριστεί εύκολα από την ομάδα ανάπτυξης. Πρακτικά, η ομάδα ανάπτυξης σχηματίζει μια υπόθεση, τη δοκιμάζει και κάνει αλλαγές ανάλογα με το αποτέλεσμα της δοκιμής αυτής. Το βασικό καθήκον μιας ομάδας που εφαρμόζει τις αρχές Lean/Agile, είναι να αξιολογεί συνεχώς τις διαδικασίες της και να βελτιώνεται συνεχώς όσο χρειάζεται και όσο γίνεται.

Ο αντίκτυπος κάθε αλλαγής που κάνει μπορεί να παρατηρηθεί και να μετρηθεί χρησιμοποιώντας μετρικά. Χρησιμοποιώντας αυτά τα μετρικά, μπορεί να αξιολογηθεί εάν μια αλλαγή βοηθά την ομάδα να βελτιωθεί ή όχι και να αποφασίσει εάν θα τη διατηρήσει ή θα δοκιμάσει κάτι άλλο.

2.2.3. Lean Software Development (LSD)

Η Lean Software Development (LSD) μεθοδολογία χρησιμοποιείται για τον εξορθολογισμό και τη βελτιστοποίηση της διαδικασίας ανάπτυξης λογισμικού. Μπορεί επίσης να αναφέρεται ως στρατηγική ελάχιστου βιώσιμου προϊόντος (Minimum Viable Product - MVP), καθώς αυτοί οι τρόποι σκέψης μοιάζουν πολύ, μιας και οι δύο έχουν σαν σκοπό την επιτάχυνση της ανάπτυξης εστιάζοντας σε νέα παραδοτέα.

Η Toyota έχει πιστωθεί ότι εμπνέει την προσέγγιση του Lean Development, η οποία προορίζεται για τη βελτιστοποίηση της παραγωγής και την ελαχιστοποίηση της σπατάλης. Βλέποντας τη Lean Development προσέγγιση της Toyota, πολλές άλλες κατασκευαστικές ομάδες άρχισαν να ακολουθούν την ίδια στρατηγική. Η Lean Development μεθοδολογία υιοθετήθηκε για πρώτη φορά στην ανάπτυξη λογισμικού το 2003 και μετονομάστηκε σε Lean Software Development.

Η LSD έχει αποδειχθεί ότι βελτιώνει την ανάπτυξη λογισμικού με τους εξής τρόπους:

1. Η LSD αφαιρεί τα περιττά στάδια της διαδικασίας κατά το σχεδιασμό λογισμικού, έτσι ώστε να εξοικονομεί χρόνο καθώς απλοποιεί τη διαδικασία ανάπτυξης.
2. Με εστίαση στο MVP, η LSD δίνει προτεραιότητα σε βασικές λειτουργίες, ώστε να εξαλείφει τον κίνδυνο να ξοδευτεί χρόνος σε εκδόσεις ή και αναλύσεις που δεν έχουν αξία για το τελικό προϊόν.
3. Αυξάνει τη δύναμη εμπλοκής της ομάδας ανάπτυξης καθώς συμμετέχουν όλο και περισσότερα μέλη, με άμεσο αποτέλεσμα να βελτιστοποιείται η συνολική ροή εργασίας και να μειώνονται οι απώλειες.

<u>Βασικές αρχές της LSD:</u>

1. **Εξάλειψη των αποβλήτων:** Για τον εντοπισμό και την εξάλειψη των αποβλήτων π.χ. του περιττού κώδικα, της καθυστέρησης των διεργασιών, της αναποτελεσματικής επικοινωνίας, του προβλήματος ποιότητας, του διπλασιασμού των δεδομένων, της καταγραφής περισσότερων εργασιών στο αρχείο καταγραφής από όσες έχουν ολοκληρωθεί, κ.λπ. πραγματοποιούνται τακτικές συναντήσεις από τους Project Managers. Αυτό επιτρέπει στα μέλη της ομάδας να επισημαίνουν σφάλματα και να προτείνουν αλλαγές στην επόμενη για την επόμενη έκδοση.
2. **Γρήγορη παράδοση:** Ο μακροχρόνιος προγραμματισμός ήταν η βασική επιτυχία των επιχειρήσεων, αλλά με το πέρασμα του χρόνου διαπιστώθηκε ότι οι μηχανικοί αφιερώνουν πολύ χρόνο στην κατασκευή πολύπλοκων συστημάτων με ανεπιθύμητα χαρακτηριστικά. Έτσι, κατέληξαν σε μια στρατηγική MVP που είχε ως αποτέλεσμα τα προϊόντα κατασκευής γρήγορα που περιλάμβαναν λίγη λειτουργικότητα και λανσάρισαν το προϊόν στην αγορά και βοηθούσε στην καταγραφή και την μελέτη των αντιδράσεων των χρηστών. Μια τέτοια προσέγγιση τους επιτρέπει να βελτιώσουν το προϊόν με βάση τα σχόλια των πελατών.
3. **Ενίσχυση γνώσης - Amplify Learning:** Η μάθηση βελτιώνεται μέσω της άφθονης αναθεώρησης κώδικα, μιας συνάντησης που ισχύει μεταξύ ομάδων. Εξασφαλίζεται επίσης ότι δεν συσσωρεύεται συγκεκριμένη γνώση από έναν μηχανικό που γράφει ένα συγκεκριμένο κομμάτι κώδικα, ώστε να χρησιμοποιείται η μέθοδος του paired programming.
4. **Ποιότητα κατασκευής:** Το LSD έχει να κάνει με την πρόληψη της σπατάλης, προσέχοντας να μην θυσιάζεται η ποιότητα. Οι προγραμματιστές συχνά εφαρμόζουν προγραμματισμό βάσει δοκιμών για να εξετάσουν τον κώδικα πριν γραφτεί. Η ποιότητα μπορεί επίσης να αποκτηθεί για να λαμβάνεται συνεχής ανατροφοδότηση από τα μέλη της ομάδας και τους διαχειριστές έργων.
5. **Σεβασμός στην ομαδική εργασία:** Το LSD εστιάζει στην ενδυνάμωση των μελών της ομάδας αντί να τα ελέγχει. Δημιουργώντας μια ατμόσφαιρα συνεργασίας, διατηρείται τέλεια ισορροπία όταν υπάρχουν σύντομες προθεσμίες και τεράστιος φόρτος εργασίας. Αυτή η μέθοδος γίνεται πολύ σημαντική όταν νέα μέλη εντάσσονται σε μια καλά εδραιωμένη ομάδα.
6. **Καθυστέρηση της δέσμευσης:** Στην παραδοσιακή διαχείριση έργων συμβαίνει συχνά όταν γίνεται μια αίτηση για υλοποίηση μιας νέας λειτουργικότητας η οποία μετά την πάροδο ενός χρόνου, αποδεικνύεται ότι είναι εντελώς ακατάλληλο για την αγορά. Η μέθοδος LSD αναγνωρίζει αυτήν την απειλή και αφήνει περιθώρια βελτίωσης αναβάλλοντας τις μη αναστρέψιμες αποφάσεις μέχρι να ολοκληρωθεί όλο το πείραμα. Αυτή η μεθοδολογία κατασκευάζει πάντα το λογισμικό ως ευέλικτο, έτσι ώστε η νέα γνώση να είναι διαθέσιμη και οι μηχανικοί να μπορούν να κάνουν βελτιώσεις.
7. **Βελτιστοποίηση ολόκληρου του συστήματος:** Η αρχή του lean επιτρέπει στους μάνατζερ να χωρίσουν ένα ζήτημα σε μικρά συστατικά μέρη για να βελτιστοποιήσουν τη ροή εργασίας της ομάδας, να δημιουργήσουν ενότητα μεταξύ των μελών και να εμπνεύσουν μια αίσθηση κοινής ευθύνης που έχει ως αποτέλεσμα τη βελτίωση της απόδοσης της ομάδας.

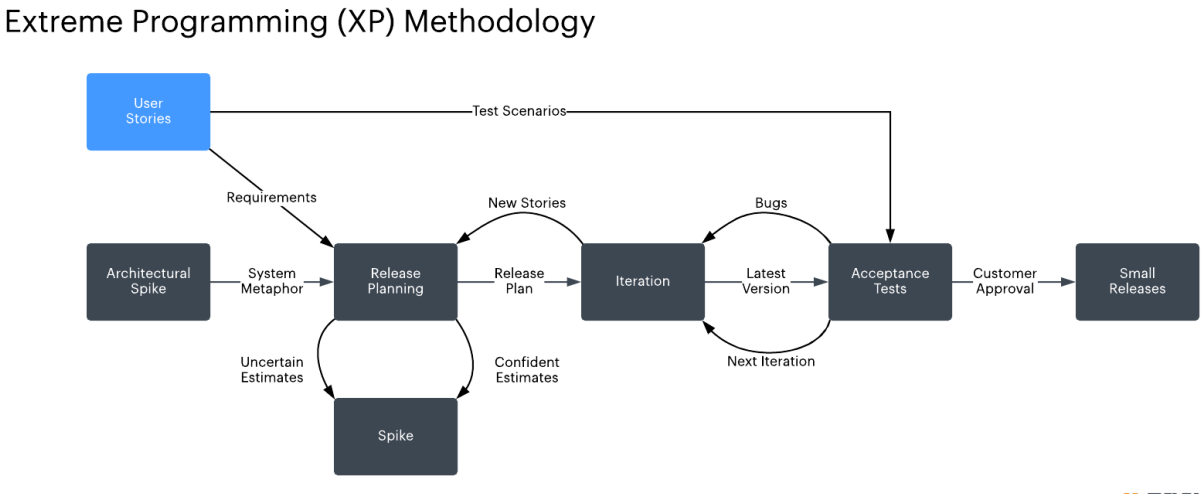
Η LSD μεθοδολογία είναι μία από τις προληπτικές προσεγγίσεις που οδηγεί τις ομάδες ανάπτυξης στην παραγωγικότητα και την οργανωτικότητα. Συνδέεται στενά με τη μεθοδολογία Agile, την εμπειρία ανταλλαγής γνώσεων, τη γρήγορη παράδοση προϊόντων. Όλες οι διαδικασίες και τα στάδια ανάπτυξης έχουν κατασκευαστεί με ακρίβεια με σκοπό να παραδίδεται το τελικό προϊόν με το ελάχιστο κόστος και εγκαίρως.

2.2.4. eXtreme Programming (XP)

Το Extreme Programming (XP)[4] είναι ένα ευέλικτο πλαίσιο ανάπτυξης λογισμικού που στοχεύει στην παραγωγή λογισμικού υψηλότερης ποιότητας όταν οι προδιαγραφές του προϊόντος αλλάζουν συχνά. Το XP είναι το πιο συγκεκριμένο από τα ευέλικτα πλαίσια όσον αφορά τις κατάλληλες πρακτικές μηχανικής για την ανάπτυξη λογισμικού εστιάζοντας στην ομαδική εργασία. Οι managers, οι πελάτες και οι προγραμματιστές είναι όλοι ίσοι εταίροι σε μια συνεργατική ομάδα η οποία αυτό-οργανώνεται γύρω από το πρόβλημα για να το λύσει όσο το δυνατόν πιο αποτελεσματικά.

Το Extreme Programming βελτιώνει ένα έργο λογισμικού με πέντε βασικούς τρόπους την επικοινωνία, την απλότητα, την ανατροφοδότηση, το σεβασμό και το θάρρος. Οι Extreme Programmers επικοινωνούν συνεχώς με τους πελάτες και τους συναδέλφους τους προγραμματιστές. Διατηρούν τον σχεδιασμό τους απλό και καθαρό. Λαμβάνουν σχόλια δοκιμάζοντας το λογισμικό τους από την πρώτη μέρα. Παραδίδουν το σύστημα στους πελάτες όσο το δυνατόν νωρίτερα και εφαρμόζουν αλλαγές όπως προτείνεται. Κάθε μικρή επιτυχία βαθαίνει το σεβασμό τους για τη μοναδική συνεισφορά κάθε μέλους της ομάδας. Με αυτό το θεμέλιο, οι Extreme Programmers είναι σε θέση να ανταποκριθούν με θάρρος στις μεταβαλλόμενες απαιτήσεις και την τεχνολογία.

Η πιο εκπληκτική πτυχή του Extreme Programming είναι οι απλοί κανόνες του και το γεγονός ότι μοιάζει πολύ με ένα παζλ με πολλά μικρά κομμάτια που μεμονωμένα δεν έχουν νόημα. Όταν όμως αυτά τα κομμάτια συνδυάζονται μαζί μπορεί κάποιος να σχηματίσει μια πλήρη εικόνα. Οι κανόνες μπορεί να φαίνονται δύστροποι και ίσως ακόμη και αφελείς στην αρχή, όμως αυτοί βασίζονται σε υγιείς αξίες και αρχές. Στην Εικόνα 5 παρουσιάζεται ο τρόπος με τον οποίο αυτοί οι κανόνες συνδυάζονται μαζί και στο κεφάλαιο αυτό επίσης παρουσιάζονται οι κανόνες αυτοί, όπως και οι αξίες του XP.



Εικόνα 5: *Extreme Programming [26].*

<u>Οι κανόνες του Extreme Programming:</u>

Planning

[4] http://www.extremeprogramming.org/

1. Το σύνολο του έργου θα πρέπει να χωρίζεται σε μικρότερα κομμάτια τα οποία αποτελούν και οι επαναλήψεις του έργου - Iterations.
2. Το Iteration Planning αποτελεί η αρχή του εκάστοτε Iteration.
3. Τα User Stories που περιγράφουν τις ανάγκες των τελικών χρηστών της εφαρμογής γράφονται και οργανώνονται για παράδοση στην λήξη του κάθε Iteration.
4. Σκοπός του XP είναι η ανάπτυξη και παράδοση συχνών μικρών εκδόσεων.
5. Το Release Planning δημιουργεί το χρονοδιάγραμμα των νέων εκδόσεων.

Διαχείριση

1. Η ομάδα ανάπτυξης θα πρέπει να συνεργάζεται σε έναν ειδικό ανοιχτό χώρο εργασίας.
2. Θα πρέπει να οριστεί ένας βιώσιμος ρυθμός ανάπτυξης.
3. Θα πρέπει να οριστεί μια καθημερινή συνάντηση stand-up.
4. Θα πρέπει να μετράται η ταχύτητα του έργου.
5. Τα μέλη της ομάδας ανάπτυξης θα πρέπει να μετακινούνται και να αλλάζουν θέσεις.
6. Το XP και οι διαδικασίες αυτού θα πρέπει να διορθώνονται όταν παρατηρείται ότι δεν λειτουργούν σωστά. Για την υποστήριξη αυτού του κανόνα, θα πρέπει να υπάρχουν σημεία και μηχανισμοί ανατροφοδότησης τα οποία βοηθούν όλα τα μέλη του έργου ή του οργανισμού να προτείνουν λύσεις και να αξιολογούν τις καταστάσεις.

Σχεδιασμός

1. Η απλότητα αποτελεί ένας πολύ βασικός παράγοντας σχετικά με τον σχεδιασμό.
2. Χρησιμοποιήστε κάρτες Class, Responsibilities, and Collaboration (CRC) στις συνεδρίες σχεδίασης. Αυτές οι κάρτες ορίζουν την περιοχή στην οποία το μέρος της εργασίας εφαρμόζεται, τους υπευθύνους για αυτό το μέρος εργασίας και τα άτομα με τα οποία αυτοί θα πρέπει να συνεργαστούν.
3. Θα πρέπει να ενθαρρύνεται και να προτείνονται οι λύσεις τύπου Spike για την μείωση κινδύνου. Οι Spike λύσεις αποτελούν τεχνικές εργασίες που συνήθως αφορούν διερεύνηση και βοηθούν την ομάδα των μηχανικών να αναγνωρίσουν δυσκολίες που μπορεί να υπάρχουν για την ολοκλήρωση της εργασίας αυτής.
4. Θα πρέπει να μην προστίθεται λειτουργικότητα νωρίς.
5. Θα πρέπει η ομάδα των μηχανικών να ανακατασκευάζει τον πηγαίο κώδικα όποτε και όπου αυτό είναι δυνατόν.

Υλοποίηση

1. Θα πρέπει οι πελάτες να είναι πάντα διαθέσιμοι για την υποστήριξη των μηχανικών σε περιπτώσεις όπου οι διευκρινήσεις απαιτούνται. Για αυτό το λόγο το XP αναφέρει ότι όλοι είναι ίσοι και δεν υπάρχουν διακριτά επίπεδα ιεραρχίας ανάμεσα στα άτομα που εργάζονται πάνω στην ανάπτυξη του προϊόντος.
2. Θα πρέπει ο κώδικας να βασίζεται σε συμφωνημένα πρότυπα – πράγμα που βοηθάει τους μηχανικούς να λειτουργούν συλλογικά υπό μια συγκεκριμένη κατεύθυνση.
3. Τρόποι για τον έλεγχο του πηγαίου κώδικα θα πρέπει να αναπτύσσονται από τα πρώτα βήματα, όπως για παράδειγμα τα unit tests.
4. Θα πρέπει όλος ο κωδικός παραγωγής να είναι αναπτυγμένος ανά ζεύγη μηχανικών. Αυτό βοηθάει στην άμεση ανατροφοδότηση και τον διαμοιρασμό γνώσης εντός της ομάδας ανάπτυξης – κατά την ανάπτυξη του πηγαίου κώδικα.

5. Θα πρέπει μόνο ένα ζεύγος ενσωματώνει κώδικα κάθε φορά μέσω ενός και μοναδικού υπολογιστή ενοποίησης για την μείωση της πολυπλοκότητας και την αποφυγή σφαλμάτων.
6. Η ομάδα ανάπτυξης θα πρέπει να ενσωματώνει νέα κομμάτια κώδικα ανά συχνά χρονικά διαστήματα.
7. Θα πρέπει να καλλιεργηθεί και να επικροτείται η χρήση συλλογικής ιδιοκτησίας.

Έλεγχος

1. Όλος ο κώδικας πρέπει να έχει unit tests.
2. Ο κώδικας για να ανέβει παραγωγή θα πρέπει να έχει περάσει επιτυχώς όλα τα unit tests.
3. Όταν εντοπιστεί ένα σφάλμα, δημιουργούνται νέα tests τα οποία θα εξυπηρετούν στον εντοπισμό του ίδιου σφάλματος για το άμεσο μέλλον.
4. Τα Acceptance tests εκτελούνται συχνά και η βαθμολογία αυτών θα δημοσιεύεται προς όλο τον οργανισμό.

<u>Οι αξίες του Extreme Programming:</u>

Απλότητα: Η ομάδα ανάπτυξης κάνει ό,τι χρειάζεται και έχει ζητηθεί, αλλά όχι περισσότερο. Αυτό θα μεγιστοποιήσει την αξία που έχει δημιουργηθεί για την επένδυση που έχει πραγματοποιηθεί μέχρι σήμερα. Η ομάδα ανάπτυξης θα κάνει μικρά απλά βήματα προς τον στόχο και θα μετριάζει τις αποτυχίες καθώς συμβαίνουν. Θα δημιουργηθεί κάτι για το οποίο είναι περήφανοι και θα το διατηρήσουν μακροπρόθεσμα με λογικό κόστος.

Επικοινωνία: Όλοι είναι μέρος της ομάδας και επικοινωνούν πρόσωπο με πρόσωπο καθημερινά. Εργάζονται μαζί για τα πάντα, από τις απαιτήσεις μέχρι τον κώδικα. Δημιουργούν την καλύτερη λύση που μπορούν στο πρόβλημά μαζί.

Ανατροφοδότηση: Η ομάδα ανάπτυξης θα λάβει σοβαρά υπόψη την δέσμευσή της σε κάθε επανάληψη παρέχοντας λειτουργικό λογισμικό. Θα επιδεικνύει το λογισμικό της νωρίς και συχνά, στη συνέχεια θα ακούει προσεκτικά και θα κάνει όποιες αλλαγές χρειάζονται. Οι διαδικασίες θα προσαρμόζονται στο έργο συνεχώς.

Σεβασμός: Όλοι δίνουν και νιώθουν τον σεβασμό που τους αξίζει ως πολύτιμο μέλος της ομάδας. Οι προγραμματιστές σέβονται την τεχνογνωσία των πελατών και το αντίστροφο. Η διοίκηση σέβεται το δικαίωμά όλων να αναλαμβάνουν την ευθύνη και να λαμβάνουν εξουσία για τη δική τους εργασία.

Κουράγιο: Θα λέγεται η αλήθεια για την πρόοδο και τις εκτιμήσεις. Δεν τεκμηριώνονται δικαιολογίες για αποτυχία επειδή ο σχεδιασμός γίνεται με σκοπό, την επιτυχία. Ο φόβος της αποτυχίας θα μειώνεται επειδή κανείς δεν δουλεύει ποτέ μόνος. Η προσαρμογή στις αλλαγές όποτε αυτές συμβούν είναι δεδομένη.

2.2.5. The Crystal Method

Η μεθοδολογία Crystal αναπτύχθηκε από έναν Αμερικανό επιστήμονα, τον Alistair Cockburn που εργαζόταν στην IBM. Αποφάσισε να μην επικεντρωθεί σε αναπτυξιακές στρατηγικές βήμα προς βήμα, αλλά να αναπτύξει την ομαδική συνεργασία και την επικοινωνία. Μερικά από τα χαρακτηριστικά της μεθόδου του Cockburn's Crystal ήταν:

- Η ανθρώπινη δυναμική, δηλαδή το έργο θα πρέπει να είναι ευέλικτο και οι άνθρωποι να συμμετέχουν στο κομμάτι εργασίας που προτιμάνε.
- Οι προσαρμοστικές προσεγγίσεις, δηλαδή, η ομάδα ανάπτυξης θα πρέπει να μην έχει σταθερά εργαλεία, αλλά να μπορεί να αλλάξει ανά πάσα στιγμή για να ανταποκριθεί στις συγκεκριμένες ανάγκες της.
- Το πλαίσιο ανάπτυξης να είναι εξαιρετικά ελαφρύ, δηλαδή αυτή η μεθοδολογία δεν απαιτεί πολλή τεκμηρίωση.

Η Crystal μεθοδολογία χρησιμοποιεί ένα σύνολο από διάφορες αναπτυξιακές προσεγγίσεις. Αρχικά, η προσέγγιση καθορίζεται λαμβάνοντας υπόψη τις επιχειρηματικές απαιτήσεις και τις ανάγκες του έργου. Στην συνέχεια, ανάλογα με τις ανάγκες του έργου μπορεί να κατηγοριοποιηθεί ανάλογα με τις ανάγκες της ομάδας ανάπτυξης σε διαφορετικές χρωματικές αποχρώσεις όπως παρουσιάζονται παρακάτω:

Crystal clear: Η ομάδα αποτελείται από μόνο 1-6 μέλη που είναι κατάλληλη για βραχυπρόθεσμα έργα όπου τα μέλη εργάζονται σε ενιαίο χώρο εργασίας.

Crystal yellow: Η ομάδα αποτελείται από 7-20 μέλη, όπου τα σχόλια λαμβάνονται από πραγματικούς χρήστες. Αυτή η παραλλαγή περιλαμβάνει τις αυτοματοποιημένες δοκιμές που επιλύουν τα σφάλματα πιο γρήγορα και μειώνουν τη χρήση υπερβολικών και λεπτομερών τεκμηριώσεων.

Crystal orange: Η ομάδα αποτελείται από 21-40 μέλη και χωρίζεται ανάλογα με τις λειτουργικές δεξιότητες των μελών. Το έργο συνήθως διαρκεί από 1 έως 2 χρόνια και απαιτούνται νέες εκδόσεις του κάθε 3 με 4 μήνες.

Crystal red: Η ανάπτυξη λογισμικού καθοδηγείται από 40-80 μέλη όπου οι ομάδες μπορούν να σχηματιστούν και να χωριστούν ανάλογα με τις απαιτήσεις.

Crystal Maroon: Περιλαμβάνει έργα μεγάλου μεγέθους όπου το μέγεθος της ομάδας είναι 80-200 μέλη. Οι μέθοδοι ανάπτυξης είναι διαφορετικές και προσαρμόζονται σύμφωνα με τις απαιτήσεις του λογισμικού.

Ιδιότητες της Crystal Μεθοδολογίας:

1. **Συχνή παράδοση:** Επιτρέπει την συχνή παράδοση του κώδικα δοκιμής του προϊόντος σε πραγματικούς χρήστες με άμεσο σκοπό την ανατροφοδότηση.
2. **Ανακλαστική Βελτίωση:** Δεδομένου ότι υπάρχουν πάντα τομείς όπου το προϊόν μπορεί να βελτιωθεί, οι ομάδες ανάπτυξης μπορούν μέσω της ανατροφοδότησης που λαμβάνουν από πραγματικούς χρήστες, να βελτιώσουν τις μελλοντικές τους πρακτικές και το ίδιο το προϊόν.
3. **Οσμωτική επικοινωνία:** Ο Alistair δήλωσε ότι το να υπάρχουν οι ομάδες στην ίδια φάση ή κατάσταση, είναι πολύ σημαντικό καθώς επιτρέπει τη ροή πληροφοριών μεταξύ των μελών μιας ομάδας και την εύρεση λύσεων και κοινών τακτικών για την κοινή πρόοδο.
4. **Προσωπική ασφάλεια:** Δεν υπάρχουν κακές προτάσεις σε μια ομάδα ανάπτυξης που ακολουθεί την Crystal μεθοδολογία, τα μέλη της ομάδας πρέπει να αισθάνονται ασφαλή να συζητούν ιδέες ανοιχτά χωρίς κανένα φόβο.
5. **Συγκέντρωση:** Κάθε μέλος της ομάδας ξέρει ακριβώς τι πρέπει να κάνει, κάτι που του επιτρέπει να εστιάσει την προσοχή του. Αυτό ενισχύει την αλληλεπίδραση της ομάδας και την εργασία προς τον ίδιο στόχο.

6. **Εύκολη πρόσβαση σε ειδικούς χρήστες:** Βελτιώνει την επικοινωνία της ομάδας με τους χρήστες αφού η ίδια λαμβάνει τακτικά σχόλια από πραγματικούς χρήστες.
7. **Τεχνικά εργαλεία:** Περιέχει πολύ συγκεκριμένα τεχνικά εργαλεία που θα χρησιμοποιηθούν από την ομάδα ανάπτυξης λογισμικού κατά τη διάρκεια της δοκιμής, διαχείρισης και διαμόρφωσης. Αυτά τα εργαλεία επιτρέπουν στην ομάδα να εντοπίσει οποιοδήποτε σφάλμα σε μικρό χρόνο.

2.2.6. Dynamic Systems Development Method (DDSM)

Η μέθοδος Dynamic System Development Method ή DSDM είναι μια μακροχρόνια προσέγγιση που χρησιμοποιείται με επιτυχία από το 1994 για την ταχεία ανάπτυξη και παράδοση λογισμικού. Με την επαναληπτική και σταδιακή προσέγγισή που ακολουθεί αυτή η μέθοδος, ήταν ευέλικτη πολύ πριν γραφτεί το Agile Μανιφέστο και προχώρησε πιο μακριά από το επίπεδο ομάδας, καθιστώντας το μια ευέλικτη μεθοδολογία κλιμάκωσης μέσα σε οργανισμούς. Με αυτή την προσέγγιση ενισχύεται η επίλυση πολλών από τα προβλήματα που εντοπίζονται σε πιο παραδοσιακά μοντέλα ανάπτυξης λογισμικού. Για παράδειγμα, η DSDM μεθοδολογία προτείνει την χρήση τακτικών διευκολυνόμενων workshops και ενθαρρύνει την ομάδα να αλληλεπιδράσει, κάτι που βοηθά στην επίλυση των φυσικών προβλημάτων επικοινωνίας που αντιμετωπίζουν πολλά έργα.

Η DSDM μεθοδολογία εστιάζει στην έγκαιρη παράδοση, επειδή η παράδοση του 80% των λειτουργιών είναι καλύτερο από την παράδοση όλων των λειτουργιών αργά. Μιας και οι πελάτες ή οι χρήστες του τελικού προϊόντος είναι άνθρωποι, καθώς οι γνώσεις τους για το προϊόν αυξάνονται κατά τη φάση ανάπτυξης, οι ίδιοι τείνουν να αλλάζουν γνώμη. Επομένως, η προσέγγιση της DSDM για την κατανόηση των απαιτήσεων των πελατών είναι να εμπλέξει έναν πελάτη στην ανάπτυξη της λύσης. Έτσι η έγκαιρη συμμετοχή του πελάτη, βοηθά στην αποφυγή προβλημάτων, σταματά τη δημιουργία ανεπιθύμητων λειτουργιών και διασφαλίζει ότι οι πελάτες και συνεπώς και η επιχείρηση παίρνουν αυτό που χρειάζονται. Βοηθά επίσης στην αποφυγή υπερβολικής μηχανικής μιας και οι λειτουργικότητες που δεν χαρακτηρίζονται ως απαραίτητες από τον πελάτη σταματάνε να αναπτύσσονται. Όλες αυτές οι θετικές επιπτώσεις διασφαλίζουν ότι ο πελάτης θα έχει απόδοση της επένδυσής του το συντομότερο δυνατό.

<u>Η φιλοσοφία της DSDM μεθοδολογίας:</u>

Η φιλοσοφία της DSDM είναι ότι κάθε έργο πρέπει να ευθυγραμμίζεται με σαφώς καθορισμένους στρατηγικούς στόχους και να επικεντρώνεται στην έγκαιρη παράδοση πραγματικών οφελών για την επιχείρηση.

Αυτό επιτυγχάνεται όταν όλοι οι ενδιαφερόμενοι:

1. Κατανοούν και συνεισφέρουν στο όραμα και τους στόχους της επιχείρησης.
2. Είναι εξουσιοδοτημένοι να λαμβάνουν αποφάσεις στον τομέα της εμπειρογνωμοσύνης τους.
3. Συνεργάζονται για να προσφέρουν μια κατάλληλη επιχειρηματική λύση.
4. Συνεργάζονται για την παράδοση σε συμφωνημένα χρονοδιαγράμματα σύμφωνα με τις επιχειρηματικές προτεραιότητες.
5. Αποδεχθούνε ότι η αλλαγή είναι αναπόφευκτη καθώς η κατανόηση της λύσης μεγαλώνει με την πάροδο του χρόνου.

Η φιλοσοφία DSDM υποστηρίζεται από οκτώ αρχές που χτίζουν τη νοοτροπία και τις συμπεριφορές που είναι απαραίτητες για να ζωντανέψει η φιλοσοφία.

Οι ίδιες οι αρχές υποστηρίζονται από τέσσερα "P":

1. People: Τα άτομα που έχουν προκαθορισμένους ρόλους και ευθύνες.
2. Process: Μια ευέλικτη διαδικασία που επιτρέπει έναν επαναληπτικό και σταδιακό κύκλο ζωής για την διαμόρφωση της ανάπτυξης και της παράδοσης.
3. Products: Τα προϊόντα που περιγράφονται με σαφήνεια.
4. Practices: Τις συνιστάμενες πρακτικές που βοηθούν στην επίτευξη των βέλτιστων αποτελεσμάτων.

Οι οκτώ αρχές της DSDM μεθοδολογίας:

Οι οκτώ αρχές της DSDM ζωντανεύουν τις Agile αξίες καθοδηγώντας την ομάδα στη στάση και τη νοοτροπία που πρέπει να υιοθετήσει προκειμένου να αποδίδει με συνέπεια, παραμένοντας παράλληλα ευέλικτη.

Αρχή 1 – Εστίαση στην επιχειρηματική ανάγκη

Κάθε απόφαση που λαμβάνεται κατά τη διάρκεια ενός έργου θα πρέπει να εξετάζεται υπό το πρίσμα του πρωταρχικού στόχου του έργου - να παραδοθεί αυτό που χρειάζεται η επιχείρηση, όταν πρέπει.

Αρχή 2 –Έγκαιρη παράδοση

Η έγκαιρη παροχή μιας λύσης είναι ένα πολύ επιθυμητό αποτέλεσμα για ένα έργο και είναι πολύ συχνά ο πιο σημαντικός παράγοντας επιτυχίας.

Αρχή 3 – Συνεργασία

Η συνεργασία ενθαρρύνει την αυξημένη κατανόηση, τη μεγαλύτερη ταχύτητα και την κοινή ιδιοκτησία, που επιτρέπουν στις ομάδες να αποδίδουν σε επίπεδο που υπερβαίνει το άθροισμα των μερών τους.

Αρχή 4 – Μην διακυβεύετε ποτέ την ποιότητα

Στη DSDM μεθοδολογία, θα πρέπει να συμφωνείται στην αρχή το επίπεδο ποιότητας παράδοσης. Όλες οι εργασίες πρέπει να στοχεύουν στην επίτευξη αυτού του επιπέδου ποιότητας – ούτε περισσότερο ούτε λιγότερο.

Αρχή 5 – Σταδιακή δόμηση μέσω εταιρικών θεμελίων

Η DSDM μεθοδολογία υποστηρίζει πρώτα την κατανόηση του εύρους του επιχειρηματικού προβλήματος που πρέπει να λυθεί και της προτεινόμενης λύσης, αλλά όχι με τέτοια λεπτομέρεια ώστε το έργο να παραλύει από υπερβολικά λεπτομερή ανάλυση των απαιτήσεων.

Αρχή 6 – Επαναληπτική ανάπτυξη

Είναι απαραίτητος ο συνδυασμός Επαναληπτικής και Αυξητικής Ανάπτυξης, συχνών επιδείξεων και ολοκληρωμένης αναθεώρησης για την ενθάρρυνση της έγκαιρη ανατροφοδότηση. Η υιοθέτηση της

αλλαγής ως μέρος αυτής της εξελικτικής διαδικασίας επιτρέπει στην ομάδα να συγκλίνει σε μια ακριβή επιχειρηματική λύση.

Αρχή 7 – Συνεχής και ξεκάθαρη επικοινωνία

Η κακή επικοινωνία αναφέρεται συχνά ως η μεγαλύτερη αιτία αποτυχίας του έργου. Οι πρακτικές DSDM έχουν σχεδιαστεί ειδικά για τη βελτίωση της αποτελεσματικότητας της επικοινωνίας τόσο για ομάδες όσο και για άτομα.

Αρχή 8 – Επίδειξη ελέγχου

Η χρήση καλά καθορισμένων συναντήσεων, με σταθερά σημεία αναθεώρησης, έχουν σχεδιαστεί για να βοηθήσουν τον Υπεύθυνο Έργου και την υπόλοιπη ομάδα ανάπτυξης να ακολουθήσουν αυτήν την αρχή.

2.2.7. Feature Driven Development (FDD)

Η Feature Driven Development (FDD) μεθοδολογία γεφυρώνει το χάσμα μεταξύ των παραδοσιακών ελεγχόμενων προσεγγίσεων όπως waterfall και των αναδυόμενων διαδικασιών που συναντώνται σε Agile προσεγγίσεις όπως η ανάπτυξη με γνώμονα τη συμπεριφορά των χρηστών, το Extreme Programming και το Scrum. Επίσης η FDD χρησιμοποιείται σε περιπτώσεις όπου απαιτείται η ανάπτυξη ενός μεγάλου έργου σε έναν Agile οργανισμό ή εταιρία.

Οι ιδρυτές του FDD αποτελούν οι Jeff De Luca και ο Peter Coad, μαζί με τους συνεργάτες τους. Το FDD εφαρμόστηκε για πρώτη φορά το 1997 για ένα νέο προϊόν της Τράπεζας της Σιγκαπούρης. Το προϊόν χρειάστηκε 15 μήνες για να ολοκληρωθεί με 50 ανθρώπους να δουλεύουν για αυτό. Μετά την επιτυχημένη ολοκλήρωσή του, το FDD ξαναχρησιμοποιήθηκε για ένα δεύτερο προϊόν το οποίο ολοκληρώθηκε επιτυχώς από 250 άτομα.

Το FDD χωρίζεται σε πέντε διακριτά στάδια τα οποία περιγράφονται παρακάτω:

1. Την ανάπτυξη του αρχικού μοντέλου: Το πρώτο βήμα στο FDD είναι η περιγραφή ενός αρχικού μοντέλου - αντικειμένου τομέα. Οι ομάδες των προγραμματιστών και των ειδικών τομέα (π.χ. μελλοντικοί χρήστες στην εταιρεία του πελάτη) καθορίζουν ένα μοντέλο που αντικατοπτρίζει την κοινή τους κατανόηση για τα σημαντικά αντικείμενα στο νέο σύστημα. Ο αρχιτέκτονας τους υποστηρίζει και τους καθοδηγεί σε αυτή τη δουλειά. Όταν ολοκληρωθούν αυτά, όλα τα μοντέλα εξετάζονται για να βρεθεί το αρχικό μοντέλο που θα χρησιμοποιηθεί για τα επόμενα βήματα. Αυτό μπορεί να είναι ένα από τα μοντέλα ή ένα νέο μοντέλο που θα αποτελεί σύνθεση από τα αρχικά.

2. Την ανάπτυξη της λίστας των δυνατοτήτων: Ακολουθεί η λίστα χαρακτηριστικών — όλη η λειτουργικότητα που πρέπει να προσφέρει το προϊόν. Για να επιτευχθεί αυτό, το μοντέλο τομέα χωρίζεται σε υποτομείς που ο καθένας αντιπροσωπεύει μια επιχειρηματική λειτουργία. Για κάθε υποτομέα, στη συνέχεια αναφέρονται όλα τα χαρακτηριστικά που απαιτούνται για την υποστήριξη του πελάτη μέσω του προϊόντος σε αυτήν την επιχειρηματική λειτουργία. Τα χαρακτηριστικά που σχετίζονται στενά ομαδοποιούνται σε σύνολα χαρακτηριστικών.

3. Τον σχεδιασμό ανά δυνατότητα/λειτουργικότητα: Το τρίτο βήμα είναι η σχεδίαση της σειράς με την οποία θα αναπτυχθούν τα χαρακτηριστικά και τα σύνολα των χαρακτηριστικών.

Παράγοντες που πρέπει να ληφθούν υπόψη είναι:

- Ποια χαρακτηριστικά χρειάζονται πιο επειγόντως οι χρήστες του πελάτη σας.
- Ποιες λειτουργίες θα προσφέρουν τη μεγαλύτερη αξία το συντομότερο.
- Οι εκτιμήσεις χρόνου ανάπτυξης —τα μεγαλύτερα χαρακτηριστικά σε απαιτούμενο χρόνο ανάπτυξης θα πρέπει να υλοποιηθούν πρώτα.
- Οι τεχνικές εξαρτήσεις μεταξύ των χαρακτηριστικών.
- Οι κίνδυνοι που συνδέονται με λειτουργίες — για παράδειγμα δοκιμές συμμόρφωσης, χρήση νέας τεχνολογίας, εκμάθηση νέων πλαισίων και βιβλιοθηκών.
- Οι διαθέσιμοι άνθρωποι, οι δεξιότητες και η τεχνογνωσία τους.
- Ο φόρτος εργασίας.

Το αποτέλεσμα είναι μια προτεραιοποιημένη λίστα με όλα όσα πρέπει να υλοποιηθούν. Είναι σαν να ξεκινά η υλοποίηση ενός έργου με Scrum με ένα πλήρως γεμάτο (αλλά όχι εκλεπτυσμένο) ανεκτέλεστο backlog.

4. Τον σχεδιασμό ανά δυνατότητα/λειτουργικότητα: Κάθε ομάδα δυνατοτήτων εργάζεται στον λεπτομερή σχεδιασμό των λειτουργιών που της έχουν ανατεθεί για την τρέχουσα επανάληψη. Οι προγραμματιστές και οι ειδικοί τομέα, βοηθούμενοι από τον επικεφαλής αρχιτέκτονα και τον επικεφαλής προγραμματιστή, χρησιμοποιούν τεχνικές μοντελοποίησης UML, όπως διαγράμματα κλάσεων, διαγράμματα ακολουθίας και διαγράμματα μετάβασης κατάστασης, για να καθορίσουν τι θα κάνει το χαρακτηριστικό και πώς θα το κάνει. Δουλεύοντας σε αυτά τα συγκεκριμένα σχέδια, η γνώση για το προϊόν αυξάνεται και ενημερώνει τυχόν αλλαγές που απαιτούνται στο αρχικό μοντέλο αντικειμένου τομέα. Σε κάθε επανάληψη, στο τέλος της φάσης σχεδιασμού γίνεται μια ανασκόπηση από ολόκληρη την ομάδα που διασφαλίζει τον εντοπισμό των αλληλεπικαλύψεων, τον εντοπισμό των εξαρτήσεων και το ότι όλες οι ομάδες γνωρίζουν το συνολικό μοντέλο και το σχέδιο και παραμένουν εντός της αρχικής ιδέας του προϊόντος.

5. Την υλοποίηση ανά δυνατότητα/λειτουργικότητα: Στη συνέχεια, κάθε ομάδα δυνατοτήτων εργάζεται για να μετατρέψει το σχέδιό της σε λειτουργικό λογισμικό, να το δοκιμάσει και να συλλέξει σχόλια από ειδικούς τομέα για να επαληθεύσει ότι η λειτουργία έχει την αναμενόμενη συμπεριφορά. Όταν όλα είναι εντάξει, οι δουλειά της ομάδας ενσωματώνεται με όλα όσα έχουν κατασκευαστεί σε αυτήν και σε προηγούμενες επαναλήψεις.

Οι κύριοι διακριτοί ρόλοι του FDD που έχουν οριστεί σε αυτό το πλαίσιο ανάπτυξης είναι:

Ο **Υπεύθυνος Έργου (Project Manager)** που επιβλέπει ολόκληρο το έργο και χρησιμοποιεί παραδοσιακές και όχι ευέλικτες τεχνικές διαχείρισης.

Ο **Υπεύθυνος Ανάπτυξης (Development Manager)** που ασχολείται με τις καθημερινές δραστηριότητες και λειτουργεί ως μέντορας για λιγότερο έμπειρα μέλη της ομάδας.

Ο **Επικεφαλής Αρχιτέκτονας (Chief Architect)** που είναι υπεύθυνος για τη συνολική σχεδίαση του συστήματος και καθοδηγεί τους ειδικούς και τους προγραμματιστές του τομέα κατά τη διάρκεια των αρχικών και επαναληπτικών βημάτων σχεδιασμού.

Ο **Επικεφαλής Προγραμματιστής (Chief Programmer)** που είναι ένας έμπειρος προγραμματιστής που βοηθά τις ομάδες στην ανάλυση και το σχεδιασμό κατά τη διάρκεια των βημάτων σχεδίασης και υλοποίησης.

Ο **Κάτοχος Κλάσης (Class Owner)** που είναι κάθε προγραμματιστής που είναι υπεύθυνος για το σχεδιασμό, την κωδικοποίηση, τη δοκιμή και την τεκμηρίωση μιας κλάσης που υλοποιεί μέρος του μοντέλου αντικειμένου τομέα στον κώδικα. Ένας ειδικός τομέα (domain Expert) που κατανοεί το αντικείμενο δραστηριοποίησης του πελάτη και το μέρος που θα υποστηρίξει το νέο προϊόν (σύστημα) ειδικότερα. Βοηθά τις ομάδες ανάπτυξης να κατανοήσουν πώς λειτουργεί η επιχείρηση και τι χρειάζεται.

2.2.8. Adaptive Software Development (ASD)

Η Adaptive Software Development (ASD) είναι μια μεθοδολογία ανάπτυξης λογισμικού που δημιουργήθηκε από τους Jim Highsmith και Sam Bayer, ένα δίδυμο διαχειριστών έργων. Η ASD υποστηρίζει ότι ο σωστός τρόπος για να επιτευχθεί η επιτυχία όταν πρόκειται για σύνθετα έργα λογισμικού είναι η συνεχής μάθηση σε όλη τη διάρκεια του έργου, χωρίς να τηρούνται άκαμπτα σχέδια.

Στην ASD, τα έργα πραγματοποιούνται σε έναν επαναληπτικό κύκλο που αποτελείται από τρεις αλληλοκαλυπτόμενες φάσεις: την μελέτη (speculation), την συνεργασία (collaboration) και μάθηση (learning). Η φάση της μελέτης είναι αυτό που συνήθως αποκαλείται προγραμματισμός (planning) σε άλλες μεθοδολογίες, ευέλικτες ή μη. Αναγνωρίζει ρητά το παράδοξο της δημιουργίας σχεδίων σε ένα ταχέως εξελισσόμενο, περίπλοκο, σενάριο. Η φάση της συνεργασίας δείχνει τη σημασία της στενότερης και συνεχούς συνεργασίας, όχι μόνο εντός της ομάδας ανάπτυξης, αλλά και μεταξύ των προγραμματιστών και των τελικών χρηστών του λογισμικού. Τέλος, η φάση εκμάθησης υποδεικνύει ότι όλοι όσοι εμπλέκονται στη διαδικασία ανάπτυξης λογισμικού θα μαθαίνουν συνεχώς σε όλο το έργο.

Η ASD, ως προσέγγιση, προσανατολίζεται προς την υψηλή ταχύτητα: χρησιμοποιεί τεχνικές όπως επαναληπτικούς κύκλους με χρονικά πλαίσια, προγραμματισμό βάσει κινδύνου και ταυτόχρονη επίτευξη γρήγορης παράδοσης αξίας στον πελάτη, ενώ επίσης αγκαλιάζει την αβεβαιότητα και το σχεδόν χάος που είναι εγγενείς σε πολύπλοκα έργα υψηλού κινδύνου.

Κατά τη διάρκεια των επαναλήψεων, οι προγραμματιστές όχι μόνο κατασκευάζουν νέα στοιχεία αλλά κάνουν και τις απαραίτητες τροποποιήσεις σε προϋπάρχοντα. Επιπλέον η βάση της ASD μεθοδολογίας είναι το επιθυμητό αποτέλεσμα και όχι η νέα λειτουργικότητα όπως συνηθίζεται σε άλλες ευέλικτες μεθοδολογίες. Ως επιθυμητό αποτέλεσμα, ορίζεται το σύνολο των χαρακτηριστικών που υλοποιούνται και παραδίδονται μαζί. Πιο συγκεκριμένα, υπάρχουν τρεις τύποι εξαρτημάτων στο ASD: τα κύρια εξαρτήματα, τα στοιχεία τεχνολογίας και τα εξαρτήματα υποστήριξης. Τα κύρια στοιχεία αναφέρονται στις ίδιες τις επιχειρησιακές λειτουργίες. Τα στοιχεία τεχνολογίας, από την άλλη πλευρά, αποτελούνται από κομμάτια τεχνολογίας ή υποδομής που πρέπει να υπάρχουν για να εφαρμοστούν τα κύρια στοιχεία. Τα στοιχεία τεχνολογίας περιλαμβάνουν όχι μόνο υλικό και υποδομή δικτύου, αλλά και λογισμικό όπως λειτουργικά συστήματα, βάσεις δεδομένων, πλαίσια και άλλα. Αυτό που συμβαίνει συχνά είναι ότι τέτοια εξαρτήματα θα είναι ήδη έτοιμα για χρήση. Εάν δεν είναι, πρέπει να αντιστοιχίζονται σε επαναλήψεις για υλοποίηση. Τέλος, τα στοιχεία υποστήριξης

περιλαμβάνουν οτιδήποτε άλλο δεν προβλέπεται από τους άλλους δύο τύπους. Για παράδειγμα, εκπαιδευτικό υλικό, τεκμηρίωση της τεχνικής υλοποίησης κ.α.

Όπως και σε άλλες μεθοδολογίες, οι επαναλήψεις στην ASD είναι χρονικά καθορισμένοι, όχι μόνο στο επίπεδο κάθε μεμονωμένου κύκλου ανάπτυξης, αλλά και σε επίπεδο έργου. Οι κύκλοι ASD βασίζονται στον κίνδυνο και είναι ανεκτικοί στις αλλαγές. Βασική αρχή της ASD σχετική με την διαχείριση κινδύνου είναι η εξής: εάν μια δεδομένη απόφαση είναι καταδικασμένη να αποτύχει, είναι καλύτερο και φθηνότερο να το μάθετε νωρίτερα παρά αργότερα. Οπότε τα επιθυμητά αποτελέσματα που κρύβουν τους περισσότερους κινδύνους, επιλέγονται να υλοποιούνται πρώτα στην ASD μεθοδολογία για την αποδοτικότερη διαχείριση κινδύνου. Ανεκτικοί στις αλλαγές σημαίνει ότι, αντί να υπάρχει η υπόθεση ότι οι περισσότερες δραστηριότητες θα γίνουν όπως έχουν προγραμματιστεί, ξεκινάει η κάθε επανάληψη με την προϋπόθεση ότι όλα θα αλλάζουν, και ότι αλλάζουν συχνά. Στην ASD μεθοδολογία, οι ομάδες δημιουργούν περιβάλλοντα ανθεκτικά στην αλλαγή, ώστε η αλλαγή να μπορεί να διαχειρίζεται καλύτερα. Ένας από τους τρόπους με τους οποίους αυτό επιτυγχάνεται είναι μέσω της υιοθέτησης μικρότερων χρονικών πλαισίων.

Οι βασικές αξίες των ομάδων στην ASD μεθοδολογία:

Στην ανάπτυξη προσαρμοστικού λογισμικού, οι ομάδες λειτουργούν σύμφωνα με τις 4 βασικές αξίες: αμοιβαία εμπιστοσύνη, αμοιβαίος σεβασμός, αμοιβαία συμμετοχή και αμοιβαία δέσμευση.

Βασικοί Ρόλοι και Ευθύνες:

Η ASD μαθοδολογία αυτοπροσδιορίζεται ως ένα πλαίσιο εννοιών, πρακτικών και κατευθυντήριων γραμμών. Ως εκ τούτου, δεν είναι υπερβολικά ρυθμιστικό. Ωστόσο, υποστηρίζει ορισμένους βασικούς ρόλους και ευθύνες που παρουσιάζονται παρακάτω.

Ο **Εκτελεστικός Χορηγός** είναι ένα ενιαίο σημείο επαφής με τη διοίκηση του οργανισμού. Αυτό το άτομο παίζει έναν ρόλο που μοιάζει κάπως με αυτόν ενός κατόχου προϊόντος (Product Owner) στο Scrum. Οι κύριες αρμοδιότητές του είναι να ορίζει τους επιχειρηματικούς στόχους και να εγκρίνει τους πόρους.

Η **βασική ομάδα** που αποτελείται συνήθως από προγραμματιστές που είναι υπεύθυνοι για την ανάπτυξη και την μετέπειτα συντήρηση του τελικού προϊόντος. Σημαντικό στοιχείο της ASD είναι ότι σημαντικά άτομα από τον οργανισμό για το έργο που αναπτύσσεται θα πρέπει να είναι μέλη της βασικής ομάδας.

Στην ASD, η διαχείριση έργου γίνεται από τους **Project Managers**.

3. Προτεινόμενα Μετρικά βάση της Agile Μεθοδολογίας Ανάπτυξης Λογισμικού

Όπως είδαμε στο προηγούμενο κεφάλαιο, η Agile μεθοδολογία έχει διαδοθεί ραγδαία στον σημερινό επιχειρηματικό κόσμο. Έχουν αναπτυχθεί πλαίσια ανάπτυξης λογισμικού τα οποία βασίζονται στις αξίες και τις αρχές του Agile Μανιφέστου και στην θεωρία του εμπειρισμού. Έχουν ως κύριο στόχο την απόδοση αξίας στους πελάτες ή στους τελικούς χρήστες του προϊόντος του οργανισμού και συνεπώς στην απόδοση αξίας στον ίδιο τον οργανισμό.

Σκοπός αυτού του κεφαλαίου είναι να προτείνει μετρικά και τρόπους απεικόνισης για την συνεχή ανατροφοδότηση και την υποστήριξη των ομάδων ανάπτυξης να ενημερώνονται τόσο για την ίδια την ομάδα, την αξία που προσφέρουν στους χρήστες τους αλλά και για τον πηγαίο κώδικα του προϊόντος τους.

Οι ομάδες ανάπτυξης που ακολουθούν την Agile μεθοδολογία, μέσω των πλαισίων ανάπτυξης, ακολουθούν κοινές πρακτικές στον τρόπο εργασίας τους (όπως για παράδειγμα την ποντοδότηση των εργασιών μέσω Story Points) και μεγάλο μέρος της ανάλυσης και των προτάσεων βασίζεται σε αυτό σαν δεδομένο. Ταυτόχρονα τα μετρικά αυτά θα χρησιμοποιηθούν για να αξιολογηθούν δεδομένα από ένα Σύστημα Καταγραφής Εργασιών όπου οι ομάδες ανάπτυξης δεν χρησιμοποιούσαν Agile πρακτικές και συνεπώς θα ελεγχθεί η εφαρμογή των μετρικών αυτών και σε μη Agile περιβάλλοντα.

Η πρόκληση λοιπόν είναι να προτείνουμε, να περιγράψουμε και να μελετήσουμε μετρικά με βάση τις Agile μεθοδολογίες τα οποία θα βοηθήσουν Agile και μη Agile ομάδες να λαμβάνουν με εύκολο και γρήγορο τρόπο ανατροφοδότηση για την επίδοσή τους, την επίδοση του προϊόντος που αναπτύσσουν και των τεχνικών χαρακτηριστικών αυτού.

Για τον σκοπό αυτό το παρακάτω κεφάλαιο έχει χωριστεί στις εξής κατηγορίες που αποτελούν και τους βασικούς πυλώνες της διπλωματικής εργασίας:

3.1 Βασικά Agile Μετρικά: Σε αυτό το κεφάλαιο παρουσιάζονται τα πιο διαδεδομένα Agile Μετρικά. Επιπλέον προτείνονται αλγόριθμοι για τον υπολογισμό αυτών.
3.2 Μετρικά Συνεργατικότητας: Τα μετρικά συνεργατικότητας μπορούν να χωριστούν σε δύο μεγάλες κατηγορίες. Τη συνεργατικότητα εντός την ομάδας ανάπτυξης αλλά και εκτός αυτής. Σε αυτό το κεφάλαιο παρουσιάζονται και οι δύο πτυχές όπως και κάποιοι αποτελεσματικοί τρόποι για την εκτίμηση, την παρακολούθηση αυτών αλλά και την πρόβλεψη πιθανών καθυστερήσεων στην παράδοση του τελικού έργου.
3.3 Μετρικά Παράδοσης: Με την έννοια μετρικά παράδοσης εννοούνται αυτά τα οποία βοηθούν την ομάδα ανάπτυξης να αναγνωρίζει και να προβλέπει πόσο μακριά είναι από την τελική παράδοση ενός έργου, τα επίπεδα αξιοπιστίας της ίδιας της ομάδας, όπως επίσης και πόσο γρήγορα ανταποκρίνεται η ίδια η ομάδα σε ανάγκες. Να σημειωθεί πως σε αυτό το κεφάλαιο παρουσιάζονται εκτός από αυτά τα μετρικά και γραφικές απεικονίσεις που βοηθούν στη καταγραφή και την μελέτη της αποτελεσματικότητας της ομάδας ανάπτυξης στην παράδοση αυτών που έχουν συμφωνηθεί για παράδοση σε ένα Sprint.
3.4 Μετρικά Ποιότητας: Τα μετρικά ποιότητας δίνουν την δυνατότητα στην ομάδα ανάπτυξης να έχει εικόνα σχετικά με την ποιότητα των τελικών παραδοτέων τα οποία παρέχει στο ευρύ

κοινό. Με την καταγραφή των σφαλμάτων τα οποία εμφανίζονται στην παραγωγή όπως και αυτών που εμφανίζονται στα δοκιμαστικά περιβάλλοντα, δίνεται η δυνατότητα στις ομάδες να λαμβάνουν αποφάσεις σχετικά με το εάν "τρέχουν" πολύ. Έτσι λοιπόν αυτά τα μετρικά, συναρτήσει των Μετρικών Παράδοσης βοηθούν τις Ομάδες Ανάπτυξης να γίνονται πιο αποτελεσματικές.

3.5 Μετρικά Προστιθέμενης Αξίας Προϊόντος: Τα μετρικά προστιθέμενης αξίας παρέχουν πληροφορίες σχετικά με αξία που παρέχουν οι αλλαγές που γίνονται από τις ομάδες ανάπτυξης στο τελικό προϊόν και την αξία που δίνουν στους τελικούς χρήστες του προϊόντος.

3.6 Μετρικά για Αρχιτεκτονικές Λογισμικών και Οντολογιών: Η εικόνα για την αποδοτικότητα και την επεκτασιμότητα του πηγαίου κώδικα αποτελεί αναγκαία για τους Οργανισμούς. Έτσι λοιπόν παρουσιάζονται κάποια μετρικά και ο τρόπος αποτύπωσης αυτών σε διαγράμματα.

Σε αυτό το σημείο αξίζει να σημειωθεί πως αυτά τα μετρικά δεν προτείνονται σε καμία περίπτωση για την παρακολούθηση και την σύγκριση ομάδων ανάπτυξης λογισμικού. Αυτά τα μετρικά θα πρέπει να μοιράζονται και να είναι διαφανή σε όλο τον οργανισμό και ακόμα περισσότερο στην ίδια την ομάδα ανάπτυξης. Τα μετρικά αυτά προτείνονται για την υποστήριξη των Ευέλικτων πλαισίων ανάπτυξης. Μιας και με την βοήθεια αυτών και την πλήρη διαφάνεια, δίνεται η δυνατότητα στις ομάδες ανάπτυξης να δοκιμάζουν νέες πρακτικές και διαδικασίες και μέσω τον μετρικών να καταλαβαίνουν εάν οι αλλαγές αυτές είχαν θετικά αποτελέσματα στο τελικό προϊόν και στην δουλειά τους. Συνεπώς, τα μετρικά αυτά αποτελούν υποστηρικτικά εργαλεία για Ευέλικτες ομάδες ανάπτυξης λογισμικού.

3.1. Βασικά Agile Μετρικά

3.1.1. Story Points

Το βασικότερο μετρικό στο οποίο βασίζεται όλη η θεωρία του Scrum όπως και άλλων Frameworks είναι τα Story Points. Τα Story Points είναι μονάδες μέτρησης για την έκφραση μιας εκτίμησης της πολυπλοκότητας ή της συνολικής προσπάθειας που απαιτείται για την πλήρη υλοποίηση ενός Story ή Ticket ή οποιασδήποτε άλλης εργασίας που ανατίθεται στην ομάδα ανάπτυξης. Με τον όρο πολυπλοκότητα ή συνολική προσπάθειας, εννοούμε το πόσο δύσκολο ή τη συνολική προσπάθεια της ομάδας ανάπτυξης βάση του Definition of Done (DoD) που αποτελεί μια λίστα που εκφράζει τις εργασίες που πρέπει να ολοκληρωθούν για να θεωρηθεί ένα User Story ως ολοκληρωμένο. Τα Story Points περιγράφουν την περιπλοκότητα μια εργασίας και δεν θα πρέπει να αντιστοιχίζονται με το χρονικό πλαίσιο το οποίο απαιτείται για την υλοποίηση αυτής. Επιπλέον τα Story Points συνήθως, ακολουθούν την Fibonacci ακολουθία (1, 2, 3, 5, 8, 13, …) πράγμα που βοηθάει τα μέλη μιας ομάδας να συγκλίνουν στις αποφάσεις σχετικά με τα Story Points που αναλογούν σε ένα Story ή Ticket.

Υπάρχουν πολλές τεχνικές ώστε μια ομάδα να μπορεί να ποντοδοτεί και να συγκλίνει προς μια κοινή άποψη σχετικά με την συνολική προσπάθεια που απαιτείται για την ολοκλήρωση ενός User Story. Μια κοινή τεχνική είναι ο ορισμός του τι εκφράζει η ομάδα ανάπτυξης ως μέσο (Reference Story Points), δηλαδή 5 ή 8 ώστε να μπορεί να έχει ένα σημείο αναφοράς και να ποντοδοτεί αποτελεσματικά. Στη συνέχεια η διαδικασία ποντοδότησης γίνεται με το Poker Planning όπως το έχουν ονομάσει, δηλαδή κάθε μέλος της ομάδας ανάπτυξης έχει στην κατοχή του κάρτες οι οποίες περιέχουν την ακολουθία Fibonacci (μια κάρτα για κάθε αριθμό από το 1 μέχρι το 21) και αφού περιγραφεί η εργασία που θα πρέπει να γίνει αλλά και ο τρόπος με τον οποίο θα ελεγχθεί αυτή η

υλοποίηση, όλη η ομάδα που ποντοδοτεί εμφανίζει την κάρτα με την οποία θεωρεί ότι εκφράζει την πολυπλοκότητα αυτής της εργασίας. Σε περίπτωση όπου ένα ή περισσότερα μέλη της ομάδας αποκλίνουν από την κοινή γνώμη, τότε τους δίνεται ο λόγος και εξηγούν τον λόγο για τον οποίο ποντοδότησαν παραπάνω ή παρακάτω από την υπόλοιπη ομάδα. Στη συνέχεια επαναλαμβάνεται η ίδια διαδικασία μέχρι όλα τα μέλη της ομάδας να συμφωνήσουν σε έναν κοινό αριθμό. Με αυτό το τον τρόπο ενισχύεται η επικοινωνία της ομάδας όπως και η κοινή κατανόηση στο τι είναι αυτό το οποίο καλούνται να υλοποιήσουν και επιπλέον και με ποιόν τρόπο θα γίνει αυτό όπως και με ποιόν τρόπο θα γίνει το τεστ. Να σημειωθεί πως αυτός ο τρόπος ποντοδότησης ακολουθείται από πολλές ομάδες ανάπτυξης λογισμικού και έχει παρατηρηθεί πως όσο περισσότερο χρόνο συνεργάζεται μια ομάδα ανάπτυξης μεταξύ της τόσο πιο εύκολα συγκλίνει σε κοινές κάρτες, όσο και ποντοδοτεί πιο αποτελεσματικά.

3.1.2. Ταχύτητα της Ομάδας Ανάπτυξης Λογισμικού – Team Velocity

Η Ταχύτητα της ομάδας ανάπτυξης Λογισμικού ή Team Velocity, είναι μια εξαιρετικά απλή αλλά ταυτόχρονα ισχυρή μέθοδος για την ακριβή μέτρηση του ρυθμού παράδοσης επιχειρηματικής αξίας των ομάδων ανάπτυξης λογισμικού. Είναι μια ένδειξη του μέσου ποσού του Product Backlog που μετατράπηκε σε παραγωγικό προϊόν κατά τη διάρκεια ενός Sprint από μια ομάδα ανάπτυξης λογισμικού. Έτσι, για τον υπολογισμό της ταχύτητας μιας ομάδας ανάπτυξης, πρέπει απλώς να προσθέσουμε τις εκτιμήσεις των Stories και των Tickets που έχουν δοθεί από την ομάδα ανάπτυξης τα οποία παραδόθηκαν με επιτυχία στο πέρας ενός Sprint. Οι ομάδες ανάπτυξης επίσης καθορίζουν την ποσότητα των Story Points στα οποία θα δεσμευτούν για να παραδώσουν στο επόμενο Sprint. Έτσι λοιπόν, η Ταχύτητα της ομάδας ανάπτυξης μπορεί να υπολογιστεί ως εξής:

$$Team\,Velocity\ (TV) = TSPC$$

Όπου:

Total Story Points Completed (TSPC): Εκφράζει το σύνολο των Story points που ολοκληρώθηκαν στο προηγούμενο Sprint.

Για την αποτελεσματική χρήση της Ταχύτητας της ομάδας ανάπτυξης λογισμικού, προτείνεται ο υπολογισμός του μέσου για τα προηγούμενα 6 Sprints. Επίσης, θα πρέπει να λαμβάνονται υπόψιν και οι απουσίες των μελών της ομάδας ανάπτυξης κατά αυτά τα έξι προηγούμενα Sprints. Πιο συγκεκριμένα ο υπολογισμός της μέσης Ταχύτητας της ομάδας ανάπτυξης λογισμικού μπορεί να υπολογιστεί ως εξής:

$$Average\ Team\ Velocity\ (ATV) = \frac{TSPC}{TAWD} = \frac{\sum_{i=-6}^{0} TSPC_n}{\sum_{i=-6}^{0}(WD_n * TMC_n - PL_n)}$$

Όπου n=0, εκφράζει το ενεργό Sprint, n=1 το επόμενο Sprint, n=-1 το προηγούμενο Sprint κοκ. Επίσης,

- Working Days (WD): Εκφράζει τις εργατοημέρες του Sprint. Να σημειωθεί ότι οι εργατοημέρες ενός Sprint συνήθως είναι 10 μιας και το κάθε Sprint διαρκεί δύο εβδομάδες.

Αλλά υπάρχει πιθανότητα να είναι και λιγότερες λόγο αργιών ή μικρότερων σε διάρκεια Sprints, πράγμα που επηρεάζει την Δέσμευση της ομάδας ανάπτυξης.

- Team Members Count (TMC): Ο αριθμός της ομάδας ανάπτυξης που θα συνεισφέρει για την ολοκλήρωση των User Stories που θα συνθέσουν το Sprint Backlog. Να σημειωθεί ότι και αυτός ο αριθμός είναι δυναμικός μιας και η δυναμικότητα των ομάδων ανάπτυξης μπορεί να αλλάζει (αποχώρηση ενός μέλους της ομάδας, νέο μέλος ομάδας, κλπ).
- Planned Leaves (PL): Οι άδειες που μπορεί να έχουν λάβει κάποια μέλη της ομάδας ανάπτυξης.

3.1.3. Δέσμευση της Ομάδας Ανάπτυξης Λογισμικού – Team Commitment

Η Δέσμευση της ομάδας ανάπτυξης Λογισμικού ή Team Commitment, αποτελεί συμπληρωματικό μετρικό της Ταχύτητας αλλά και της Αξιοπιστίας των ομάδων ανάπτυξης λογισμικού. Στο Scrum, κατά την διάρκεια του Planning οι ομάδα ανάπτυξης καλείται να διαλέξει από το Product Backlog ένα σύνολο από τα Διαθέσιμα Stories και Tickets τα οποία θα πρέπει να παραδώσει κατά το πέρας του Sprint. Το άθροισμα λοιπόν των Story Points από τα Stories και τα Tickets στα οποία δεσμεύτηκε η ομάδα ανάπτυξης να παραδώσει στο επόμενο Sprint αποτελεί η Δέσμευση της ομάδας ανάπτυξης ή Team Commitment.

$$Team\ Commitment\ (TC) = ATV - (\frac{ATV}{FWD * TMC} * FPL)$$

Όπου:

- Average Team Velocity (ATV): Που εκφράζει την μέση Ταχύτητα της ομάδας ανάπτυξης Λογισμικού. Όταν αναφερόμαστε στην μέση Ταχύτητα της ομάδας ανάπτυξης λογισμικού, αναφερόμαστε στην μέση τιμή για τα προηγούμενα έξι Sprints – όπως παρουσιάσαμε στην προηγούμενη ενότητα.
- Forecasted Peoples Leaves (FPL): Εκφράζει τον αριθμό αδειών που προβλέπονται να λάβουν μέρος μέσα στο Sprint.

3.2. Μετρικά Συνεργατικότητας

Τα Μετρικά Συνεργατικότητας μπορούν να παρέχουν ορατότητα για το εάν μια Agile ομάδα συνεργάζεται αποτελεσματικά, τόσο εντός ομάδας όσο και εκτός ομάδας (με άλλες ομάδες για παράδειγμα). Επίσης, μπορούν να παρέχουν εικόνα για το ηθικό και τον παλμό της ομάδας. Τα Μετρικά Συνεργατικότητας τείνουν να αποτελούν κορυφαίοι δείκτες, προσφέροντας έγκαιρα προειδοποιητικά σημάδια πιθανών περιοχών προβλημάτων τα οποία χρήζουν αντιμετώπισης για την ομαλή λειτουργία τόσο της ομάδας αλλά ταυτόχρονα για όλο τον οργανισμό.

Τα Μετρικά Συνεργατικότητας που παρουσιάζονται παρακάτω είναι τα ακόλουθα:

- Ο Αριθμός εξαρτήσεων μεταξύ των ομάδων
- Έλεγχος υγείας της ομάδας ανάπτυξης

3.2.1. Αριθμός εξαρτήσεων μεταξύ ομάδων

Ένα από τα κύρια χαρακτηριστικά των Agile ομάδων είναι η διαλειτουργικότητα. Δηλαδή το γεγονός ότι οι ομάδες μπορούν να λειτουργούν ατομικά και να παραδίδουν κομμάτια προϊόντος στους τελικούς χρήστες. Ωστόσο, καθώς ένας οργανισμός κλιμακώνεται και το προϊόν γίνεται πιο περίπλοκο, είναι συχνά απαραίτητο για τις ομάδες να συνεργάζονται για να αυξήσουν την τελική αξία του τελικού προϊόντος. Έτσι σε ένα Agile περιβάλλον, οι εξαρτήσεις εκφράζουν τις καταστάσεις όπου μία ομάδα ανάπτυξης αδυνατεί να παραδώσει από μόνη της ένα κομμάτι προϊόντος στους τελικούς χρήστες [2].

Επειδή οι ομάδες έχουν δημιουργηθεί με σκοπό την ανεξαρτησία τους, το να βασίζεται μια ομάδα σε μία άλλη μπορεί να προκαλέσει τριβές, που τελικά καθυστερούν τα έργα. Επίσης, οι εξαρτήσεις μπορούν να επηρεάσουν την προτεραιοποίηση κάποιων εργασιών που μπορεί να είναι απαραίτητες μιας και οι εξαρτήσεις τους μπορεί να μην τους επιτρέπουν να ολοκληρωθούν. Γενικότερα λοιπόν, οι εξαρτήσεις αποτελούν ένα από τους κύριους λόγους καθυστέρησης παράδοσης που μπορούν να επηρεάσουν και τις σχέσεις μεταξύ των ανθρώπων σε έναν οργανισμό [3].

Οι κατηγορίες στις οποίες μπορούν να χωριστούν οι εξαρτήσεις είναι τρείς:

1. Εξαρτήσεις γνώσης:
   Οι εξαρτήσεις γνώσης παρουσιάζονται όταν απαιτούνται πληροφορίες για την πρόοδο του έργου. Υπάρχουν τέσσερείς τύποι εξάρτησης γνώσης.
   i. Η εξάρτηση γνώσης λόγο απαίτησης, αυτός ο τύπος εξάρτησης εκφράζει τις περιπτώσεις όπου η γνώση ενός συγκεκριμένου τομέα δραστηριοποίησης της ομάδας ανάπτυξης απουσιάζει από αυτή ή τις περιπτώσεις όπου μια απαίτηση δεν είναι γνωστή και πρέπει να εντοπιστεί ή να αναγνωριστεί, πράγμα που επηρεάζει την πρόοδο του έργου.
   ii. Η εξάρτηση γνώσης λόγο εξειδίκευσης, μια κατάσταση όπου οι τεχνικές πληροφορίες ή οι πληροφορίες εργασίας είναι γνωστές μόνο από ένα συγκεκριμένο άτομο ή άλλη ομάδα.
   iii. Η εξάρτηση γνώσης λόγο έλλειψης κατανομής εργασίας, εκφράζει μια κατάσταση όπου ποιος κάνει τι, και πότε, δεν είναι γνωστό, πράγμα που επηρεάζει την πρόοδο του έργου.
   iv. Η εξάρτηση γνώσης λόγο ιστορικού, μια κατάσταση όπου απαιτείται γνώση σχετικά με αποφάσεις του παρελθόντος.
2. Εξαρτήσεις εργασίας:
   Οι εξαρτήσεις εργασίας περιγράφουν τις περιπτώσεις όπου για την ολοκλήρωση μίας εργασίας απαιτείται η ολοκλήρωση μιας άλλης εργασίας, όταν δηλαδή υπάρχει σημείο μπλοκαρίσματος μεταξύ των εργασιών. Υπάρχουν δύο είδη εξαρτήσεων εργασίας:
   i. Η εξάρτηση εργασίας λόγο δραστηριότητας, αφορά μια κατάσταση που απαιτεί την ολοκλήρωση μίας δραστηριότητας πριν την εκκίνηση μιας άλλης.
   ii. Η εξάρτηση εργασίας λόγο επιχειρηματικής διαδικασίας, εκφράζει την κατάσταση όπου μία υπάρχουσα επιχειρηματική διαδικασία απαιτεί την ολοκλήρωση των εργασιών με μία συγκεκριμένη σειρά.
3. Εξαρτήσεις πόρων:
   Οι εξαρτήσεις πόρων εμφανίζονται όταν ένας πόρος απαιτείται για την ολοκλήρωση μίας εργασίας. Οι εξαρτήσεις πόρων μπορούν να χωριστούν σε δύο κατηγορίες:

i. Η εξάρτηση πόρων λόγο οντοτήτων που εκφράζει μια κατάσταση όπου ένας πόρος (άνθρωπος, τόπος ή αντικείμενο) δεν είναι διαθέσιμος.
ii. Τεχνικού χαρακτήρα, μια κατάσταση όπου μία τεχνική πτυχή επηρεάζει την πρόοδο του έργου. Για παράδειγμα όταν ένα τεχνικό κομμάτι κώδικα πρέπει να αλληλοεπιδράσει με ένα άλλο και η απουσία του επηρεάζει την πρόοδο του έργου.

Για να μπορούμε να προβλέψουμε τις πιθανές καθυστερήσεις μιας εργασίας η οποία έχει εξαρτήσεις από διαφορετικές ομάδες ανεξαρτήτου κατηγορίας, προτείνεται η παρακάτω εξίσωση:

$$Chance\ of\ on\ Time\ Delivery = \frac{1}{2^n}$$

Η εξίσωση αυτή περιγράφει την πιθανότητα του να παραδοθεί μια εργασία στον προγραμματισμένο χρόνο της εάν υπάρχει εξάρτηση μεταξύ διαφορετικών ομάδων. Όπου n ο αριθμός των εξαρτήσεων μεταξύ διαφορετικών ομάδων. Έτσι λοιπόν, όσες περισσότερες εξαρτήσεις υπάρχουν για την ολοκλήρωση μιας εργασίας, μειώνεται η πιθανότητα η εργασία αυτή να παραδοθεί στον χρόνο όπου αυτή έχει προγραμματιστεί για παράδοση.

3.2.2. Έλεγχος υγείας της ομάδας ανάπτυξης

Ο Έλεγχος υγείας της ομάδας ανάπτυξης επιτρέπει στον Scrum Master (στην περίπτωση όπου η ομάδα ανάπτυξης λειτουργεί βάση του Scrum πλαισίου ανάπτυξης λογισμικού) και σε άλλους ενδιαφερόμενους, συμπεριλαμβανομένου και των μελών της ίδιας της ομάδας ανάπτυξης, να καταλαβαίνουν εύκολα πότε μια ή περισσότερες ομάδες παρουσιάζουν δυσλειτουργίες. Κάνοντας τακτικούς ελέγχους, οι ομάδες θα πρέπει να είναι σε θέση:

- Να αυξήσουν την εμπιστοσύνη μεταξύ των ατόμων και την ψυχολογική ασφάλεια μέσα στην ομάδα.
- Τα άτομα της ομάδας να έχουν καλύτερες σχέσεις μεταξύ τους.
- Να δημιουργηθεί μια κουλτούρα εποικοδομητικής κριτικής και ανατροφοδότησης μεταξύ των μελών της ομάδας αλλά και γενικότερα όλου του οργανισμού.
- Να αναγνωριστούν προβληματικές περιοχές και να θεσπιστεί σχέδιο δράσης για την αντιμετώπιση αυτών.

Υπάρχουν πολλοί και διαφορετικοί τρόποι για τον έλεγχο υγείας μιας ομάδας ανάπτυξης. Σε αυτό το σημείο θα παρουσιάσουμε ένα δομημένο τρόπο για να κρατάμε ένα ιστορικό σχετικά με την υγεία της ομάδας σε διαφορετικές θεματικές μέσω της ανατροφοδότησης από τα μέλη της. Σε περιβάλλοντα όπου η ψυχολογική ασφάλεια των μελών της ομάδας είναι υψηλά, η ανατροφοδότηση που λαμβάνουμε από την ίδια την ομάδα είναι πολύ πιο κοντά στην πραγματικότητα. Παρακάτω θα παρουσιάσουμε έναν έλεγχο υγείας ομάδας το οποίο ουσιαστικά είναι ένα ερωτηματολόγιο αξιολόγησης με συγκεκριμένες προδιαγεγραμμένες απαντήσεις τριών επιπέδων:

1. Πράσινο: Είμαστε δυνατοί σαν ομάδα σε αυτό τον τομέα.
2. Κίτρινο: Είμαστε σε μέτρια επίπεδα σε αυτό τον τομέα σαν ομάδα και θα μπορούσαμε να βελτιωθούμε.
3. Κόκκινο: Υπάρχει εμφανής αδυναμία και θα πρέπει να βελτιωθούμε άμεσα σαν ομάδα σε αυτόν τον τομέα.

Με αυτή την απλή διαβάθμιση και τον χρωματισμό των απαντήσεων, γίνεται πολύ ευκολότερη η ανάγνωση των ιστορικών αποτελεσμάτων πράγμα πολύ σημαντικό για την αναγνώριση των κατηγοριών που χρήζουν βελτίωσης ανά περιόδους. Να σημειωθεί πως ο χρωματισμός των επιπέδων μπορεί οριστεί βάση του μέσου όρου των βαθμολογιών που συλλέγουμε από την συμμετοχή της ομάδας ανάπτυξης στον Έλεγχο υγείας.

Ενδεικτικά κάποιες κατηγορίες και οι ερωτήσεις αυτών που τίθενται στην ομάδα ανάπτυξης προς απάντηση είναι οι ακόλουθες[5]:

Πίνακας 2: *Κατηγορίες και περιοχές αξιολόγησης.*

Κατηγορία	Περιοχή αξιολόγησης
Ιδιοκτήτης έργου – Product Owner	Υπάρχει ένας επικεφαλής που είναι υπεύθυνος για το αποτέλεσμα αυτού του έργου. Αυτός πρέπει να είναι κάποιος του οποίου ο χρόνος είναι αφιερωμένος τουλάχιστον κατά 80% σε αυτό και που μπορεί να πρωταγωνιστήσει στην αποστολή εντός και εκτός της ομάδας.
Ισσοροπία της ομάδας ανάπτυξης	Οι ρόλοι και οι ευθύνες της ομάδας είναι σαφείς και συμφωνημένες. Το έργο έχει ανθρώπους με το σωστό συνδυασμό δεξιοτήτων.
Κοινή κατανόηση των στόχων	Η ομάδα έχει κοινή αντίληψη για το γιατί είναι εδώ, το πρόβλημα/ανάγκη, είναι πεπεισμένη για την ιδέα, έχει αυτοπεποίθηση ότι έχει αυτό που χρειάζεται και εμπιστεύεται ο ένας τον άλλον.
Αξία και μετρήσεις	Είναι σαφές τι σημαίνει επιτυχία από την επιχειρηματική αλλά και την οπτική γωνία των χρηστών, και υπάρχει μια μοναδική προσφορά αξίας για τους χρήστες και ταυτόχρονα για την επιχείρηση. Η επιτυχία ορίζεται, με ένα κοινό στόχο και τον τρόπο που μετράμε την επίτευξη αυτού.
Απόδειξη της ιδέας	Έχει δημιουργηθεί και δοκιμαστεί κάποιο είδος επίδειξης, που δείχνει γιατί πρέπει να λυθεί αυτό το πρόβλημα και καταδεικνύει την αξία του.
Σκοπός του έργου	Το έργο συνοψίζεται σε μια σελίδα και μοιράζεται με οποιονδήποτε, ώστε να κατανοήσει τον σκοπό του έργου και την αξία του.
Διαχειριζόμενες εξαρτήσεις	Σαφής κατανόηση της πολυπλοκότητας, της υποδομής, των κινδύνων, των πόρων, της προσπάθειας και του χρονοδιαγράμματος. Σαφής κατανόηση από ποιους εξαρτιόμαστε και από ποιους εξαρτώνται από εμάς.
Ταχύτητα της ομάδας ανάπτυξης (Velocity)	Η ομάδα σημειώνει σταδιακή πρόοδο στέλνοντας συγκεκριμένες επαναλήψεις στους ενδιαφερόμενους (και ακόμη καλύτερα στην παραγωγή), μαθαίνοντας στην πορεία και εφαρμόζοντας τα διδάγματα που αντλήθηκαν, με αποτέλεσμα μεγαλύτερη επιτυχία.

[5] Πηγή: https://www.atlassian.com/team-playbook/health-monitor

Ο Έλεγχος υγείας της ομάδας ανάπτυξης προτείνεται να παίρνει μέρος ανά τακτά χρονικά διαστήματα μέσα στο έτος (κάθε δύο Sprints ή κάθε μήνα) και να αποθηκεύονται τα ιστορικά δεδομένα για την υποστήριξη λήψης αποφάσεων για την ομάδα και την αντιμετώπιση πιθανών δυσλειτουργιών αυτής. Ένα παράδειγμα τέτοιων ιστορικών αναφορών παρουσιάζεται στην Εικόνα 6:

Εικόνα 6: *Ιστορική αναφορά Ελέγχου υγείας της ομάδας ανάπτυξης.*

Σε αυτή την απεικόνιση ουσιαστικά κάθε στήλη απεικονίζει τις επαναλήψεις στις οποίες ο Έλεγχος υγείας της ομάδας έλαβε μέρος (π.χ. κάθε Sprint) και η κάθε γραμμή απεικονίζει τον χρωματισμό που έχουμε θέσει και περιγράψει παραπάνω.

Η πρακτική του Ελέγχου υγείας μπορεί να αποτυπωθεί για περισσότερες από μια ομάδες πράγμα που βοηθάει στην αποτύπωση της γενικότερης εικόνας ενός οργανισμού σε περιοχές αξιολόγησης που μπορεί να θεωρεί ο οργανισμός ότι χρήζουν διερεύνησης. Αυτή η πρακτική όμως δεν θα πρέπει να χρησιμοποιείται για την σύγκριση μεταξύ διαφορετικών ομάδων ανάπτυξης.

3.3. Μετρικά Παράδοσης

Οι μετρήσεις παράδοσης παρέχουν ορατότητα σχετικά με την συχνότητα ανάπτυξης λειτουργιών λογισμικού και την παροχή αυτών στους τελικούς χρήστες ή πελάτες. Ως εκ τούτου, οι μετρήσεις παράδοσης επικεντρώνονται στην ικανότητα μιας ομάδας ανάπτυξης και συνεπώς ενός οργανισμού να προσφέρει με συνεπή, βιώσιμο ρυθμό νέες λειτουργικότητες προς τους τελικούς χρήστες.

Οι μετρήσεις παράδοσης τείνουν να είναι δείκτες καθυστέρησης, δίνοντάς μας πληροφορίες για την ωριμότητα των εργαλείων και των διαδικασιών ενός οργανισμού που διέπουν τη διαχείριση των τεχνουργημάτων κώδικα και την παρακολούθηση φυσικής ή εικονικής υποδομής.

3.3.1. Αριθμός User Stories και Story points

Οι μετρήσεις αυτές υπολογίζονται ως μια απλή μέτρηση ή μια μέτρηση σταθμισμένη, βάση διαφορετικών κατηγοριών όπως τον τύπο της εργασίας (απλή εργασία, σφάλμα κώδικα, technical dept, κλπ.) την πολυπλοκότητα της ιστορίας (όπως απλή, μεσαία ή σύνθετη) τον αριθμό των εργασιών που παραδόθηκαν κατά την διάρκεια ενός Sprint κ.α. Για παράδειγμα, σε ένα έργο με 120 User Stories, η πρόοδος της κυκλοφορίας μπορεί να παρακολουθηθεί ως 80 ολοκληρωμένα και αποδεκτά, 20 ολοκληρωμένα στην ανάπτυξη αλλά μη αποδεκτά και 20 ακόμη προς ανάπτυξη, εκ τω οποίων τα προς ανάπτυξη τα 5 User Stories αποτελούν σύνθετα (>=13 Story Points), τα 8 μεσαίας πολυπλοκότητας (>13 και <=5 Story Points) και τα 7 αποτελούν απλές εργασίες (<5 Story Points). Συνολικά απομένουν 168 Story points προς ανάπτυξη.

Με αυτά τα δεδομένα σε συνάρτηση της ταχύτητας παράδοσης μιας ομάδας ανάπτυξης, η ομάδα και ο συντονιστής του έργου έχουν την δυνατότητα να προβλέψουν το Sprint και συνεπώς την ημερομηνία παράδοσης του έργου.

3.3.2. Αξιοπιστία της Ομάδας Ανάπτυξης Λογισμικού – Team Reliability

Όπως αναφέραμε και στο προηγούμενο κεφάλαιο, η Αξιοπιστία της ομάδας ανάπτυξης είναι άμεσα συνδεδεμένη με την Ταχύτητα και την Δέσμευση. Πιο συγκεκριμένα η αξιοπιστία μιας ομάδας ανάπτυξης μπορεί να υπολογιστεί από τον λόγο της Δέσμευσης προς την Ταχύτητα της ομάδας αυτής. Ο λόγος αυτός μας υπολογίζει την Αξιοπιστία της ομάδας ανάπτυξης για κάθε Sprint. Για να είμαστε όμως πιο ακριβείς και να έχουμε ένα πιο ουσιαστικό μετρικό για τις ομάδες, μπορούμε να υπολογίσουμε την Αξιοπιστία των ομάδων για ένα σύνολο Sprints αθροίζοντας απλά τα μετρικά τους για τα προηγούμενα Sprints και υπολογίζοντας τον λόγο των αθροισμάτων. Ένας καλός αριθμός Sprints είναι τα 6 Sprints πράγμα που μας βοηθά να καταλάβουμε καλύτερα τις δυναμικές και την αξιοπιστία των ομάδων για ένα σύντομο χρονικό διάστημα.

$$Reliability = \frac{\sum_{i=-6}^{0} Velocity_n}{\sum_{i=-6}^{0} Commitment_n}$$

Αυτό το μετρικό αποτελεί ένα από τα σημαντικότερα για ομάδες που συνεργάζονται πολύ καιρό σε συνάρτηση πάντα με την Ταχύτητα της ομάδας για παρελθοντικά Sprints. Αυτό επειδή εάν μια ομάδα είναι έμπειρη (τα μέλη αυτής συνεργάζονται πολλά χρόνια μεταξύ τους) θα πρέπει να αυξάνει την Ταχύτητά της και να παραδίδει περισσότερα σε κάθε sprint. Εάν η ομάδα ανάπτυξης δεν παίρνει ρίσκα και έχει υψηλή Αξιοπιστία δεν σημαίνει ότι είναι και αποτελεσματική.

3.3.3. Μεταφερόμενα Story Points – Carry over

Με τον όρο Μεταφερόμενα Story Points ή Carry over, αναφερόμαστε στα Story Points τα οποία η ομάδα ανάπτυξης δεν κατάφερε να ολοκληρώσει βάση Definition of Done κατά τον προκαθορισμένο χρόνο του Sprint και αναγκάστηκε να μεταφέρει στο επόμενο. Όπως καταλαβαίνουμε, αυτό το μετρικό μπορεί να επηρεάσει την Αξιοπιστία της ομάδας. Εάν όμως η ομάδα ανάπτυξης κατά την διάρκεια του Sprint αναγκάστηκε να αναλάβει κάποιο άλλο User Story το οποίο είχε τα ίδια ακριβώς Story Points με το User Story το οποίο μετέφερε στο επόμενο Sprint και κατάφερε να το παραδώσει μαζί με όλη την υπόλοιπη δουλειά που είχε οργανώσει κατά το planning, τότε η Αξιοπιστία της θα

παραμείνει στο 100%. Να σημειωθεί ότι για την σωστή λειτουργία της ομάδας ανάπτυξης, η ανταλλαγή των User Stories (αυτό που υλοποίησε και δεν ήταν μέρος του Sprint Backlog έναντι αυτού που υλοποίησε και ήταν εκτός) θα πρέπει να γίνεται σε συνεννόηση με τον Product Owner της ομάδας. Έτσι λοιπόν, τα μεταφερόμενα Story Points ουσιαστικά αποτυπώνουν τον αριθμό των Story Points τα οποία είχαν προγραμματιστεί να υλοποιηθούν στο ενεργό Sprint βάση του Commitment της ομάδας, αλλά μεταφέρθηκαν για υλοποίηση στο αμέσως επόμενο.

3.3.4. Απρογραμμάτιστα Story Points - Unplanned

Τα απρογραμμάτιστα Story Points ή Unplanned, περιγράφουν τον όγκο δουλειάς – σε Story Points – ο οποίος προστέθηκε σε ένα ενεργό Sprint μετά την έναρξή του. Όπως γνωρίζουμε από το Scrum, οι ομάδες ανάπτυξης είναι υπεύθυνες για την ανάπτυξη αλλά και την παραγωγική υποστήριξη των υλοποιήσεων που παραδίδουν. Έτσι λοιπόν, σε κάποια sprints είναι πιθανό να χρειάζεται να καταβάλουν κάποιο από τον χρόνο τους στην παραγωγική υποστήριξη είτε αυτό είναι για την επίλυση κάποιου σφάλματος (bug) ή για την υποστήριξη κάποιον τμημάτων του οργανισμού. Αυτά τα tickets μπορούν να χαρακτηριστούν ως απρογραμμάτιστα και μέσω των σύγχρονων συστημάτων διαχείρισης εργασιών να σημειώνονται με κάποια επιγραφή (label).

Για τον λόγο αυτό μια ομάδα ανάπτυξης μπορεί να χρησιμοποιεί και ένα Απρογραμμάτιστο Ρυθμιστή (Unplanned Buffer) ο οποίος της δίνει την δυνατότητα να έχει προβλέψει εκ των προτέρων το γεγονός ότι θα χρειαστεί ένα προκαθορισμένο αριθμό Story Points για την παραγωγική υποστήριξη για το επόμενο Sprint μιας και για παράδειγμα θα ενεργοποιήσει στην παραγωγή μία νέα υλοποίηση η οποία θα επιφέρει επιπλέον δουλειά για την ομάδα. Να σημειωθεί πως τα Story Points που περιγράφουν τον Απρογραμμάτιστο Ρυθμιστή μπορούν να υπολογιστούν και εμπειρικά από τον μέσο όρο των Story Points τα οποία αναγκάστηκε η ομάδα ανάπτυξης να καταπιαστεί στα έξι προηγούμενα Sprints και ήταν Απρογραμμάτιστα. Έτσι ο Απρογραμμάτιστος Ρυθμιστής μπορεί να υπολογιστεί μέσω της ακόλουθης εξίσωσης:

$$Unplanned\ Buffer\ (UB) = \frac{\sum_{i=-6}^{0} \sum Unplanned\ Story\ Points_n}{6}$$

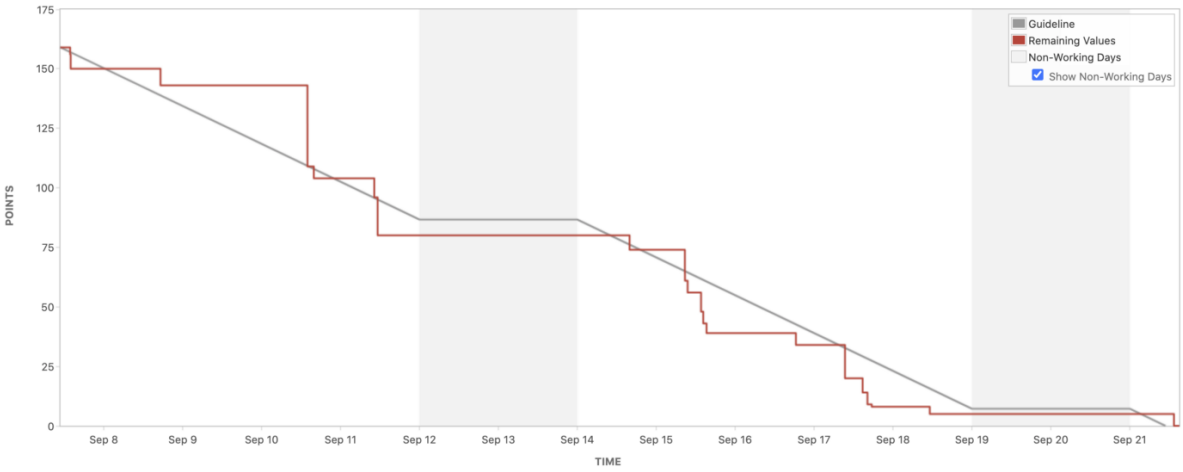
3.3.5. Sprint Burndown Chart

Το Sprint Burndown Chart είναι μια γραφική αναπαράσταση της εργασίας που απομένει σε ένα Sprint συναρτήσει του χρόνου. Χρησιμοποιείται συχνά σε Ευέλικτα πλαίσια ανάπτυξης λογισμικού όπως το Scrum. Ωστόσο, τα Sprint Burndown Charts μπορούν να εφαρμοστούν σε οποιοδήποτε έργο που περιέχει μετρήσιμη πρόοδο με την πάροδο του χρόνου.

Συνήθως, σε ένα Sprint Burndown Chart, αναπαρίσταται η εργασία που απαιτείται για την ολοκλήρωση του Sprint (άξονας y) συναρτήσει του χρόνου (άξονας χ). Αυτή η γραφική παράσταση αποτελεί απαραίτητο εργαλείο για την πρόβλεψη ολοκλήρωσης του συνόλου της εργασίας ενός Sprint που έχει οριστεί από την ομάδα ανάπτυξης. Αυτό επιτυγχάνεται με την καθημερινή ενημέρωση του γραφήματος κατά την διάρκεια του Daily Scrum, όπου η ομάδα ανάπτυξης ενημερώνει το Sprint Burn Down και σχεδιάζει την υπόλοιπη εργασία της ημέρας. Το Burndown Chart είναι ένα απαραίτητο εργαλείο για μια ομάδα Scrum για τους ακόλουθους λόγους:

- Για την παρακολούθηση της αλλαγής του πεδίου εφαρμογής του έργου και των αλλαγών.
- Για να κρατήσουμε την ομάδα εντός προγραμματισμού.
- Για την σύγκριση της προγραμματισμένης εργασίας με την πρόοδο της ομάδας ανάπτυξης.

Για την δημιουργία αυτού του γραφήματος, θα πρέπει να καθορίσουμε το σύνολο της εργασίας που έχει καθοριστεί αθροίζοντας τις όλες τις εκτιμήσεις του Sprint Backlog. Επίσης θα πρέπει να καθορίσουμε το ποσό της εργασίας που απομένει για ένα Sprint, το οποίο είναι το άθροισμα της εργασίας που απομένει για την ολοκλήρωση της εργασίας που έχει αποτυπωθεί στο Sprint Backlog κάθε μέρα. Με την χρήση των παραπάνω δεδομένων έχουμε την δυνατότητα να δημιουργήσουμε το Burndown Chart όπως φαίνεται στην Εικόνα 7.



Εικόνα 7: *Παράδειγμα ενός Sprint Burndown Chart από το JIRA.*

Όπως αποτυπώνεται στην Εικόνα 7, στον άξονα x αντικατοπτρίζεται ο χρόνος ενός Sprint, ενώ στον άξονα y αποτυπώνεται το σύνολο των απαιτούμενων Story Points για την ολοκλήρωση της εργασίας που έχει οριστεί από την ομάδα ανάπτυξης μέσα στο ενεργό Sprint. Με ανοιχτό γκρι χρώμα στο φόντο του γραφήματος, αναδεικνύονται οι ημέρες όπου η ομάδα ανάπτυξης δεν εργάζεται (12-14 Σεπτεμβρίου και 19-21 Σεπτεμβρίου). Με γκρι γραμμή αποτυπώνεται η κατευθυντήρια γραμμή η οποία αναδεικνύει την ιδανική κλήση και με κόκκινο αποτυπώνεται η πρόοδος της ομάδας ανάπτυξης συναρτήσει του χρόνου. Έτσι λοιπόν εάν η ομάδα ανάπτυξης βρίσκεται σε καλά επίπεδα σε σχέση με το υπόλοιπο της απαιτούμενης εργασίας και του χρόνου, η κόκκινη γραμμή θα βρίσκεται κάτω από την γκρι. Ενώ, αντίθετα εάν η ομάδα ανάπτυξης έχει καθυστερήσει, τότε η κόκκινη γραμμή θα βρίσκεται πάνω από την γκρι. Στο δικό μας παράδειγμα της Εικόνας 7, η ομάδα ανάπτυξης βρισκόταν πίσω από το ιδανικό πλάνο από τις 8-10 Σεπτεμβρίου και στην συνέχεια, από της 11 Σεπτεμβρίου και μετά ήταν συνεπής και μπροστά από την κατευθυντήρια.

3.3.6. Release Burndown

Το Release Burndown Chart είναι μια γραφική αναπαράσταση της εργασίας που απομένει για την ολοκλήρωση ενός έργου συναρτήσει των Sprints. Δεδομένου ότι ένα Sprint (για Scrum ομάδες) μπορεί να περιέχει εργασίες από διάφορα epics έργων και εκδόσεων, είναι σημαντικό να παρακολουθείται και να μελετάται τόσο η πρόοδος των μεμονωμένων Sprints όσο και των έργων που αναπτύσσονται από τις ομάδες ανάπτυξης.

Συνήθως, σε ένα Release Burndown Chart, η εργασία που απαιτείται για την ολοκλήρωση του epic ή του έργου αναπαρίσταται στον κάθετο άξονα y (μπορεί να αποτυπώνεται σε ώρες, μέρες ή και σε Story Points) και τα Sprints στον οριζόντιο x άξονα. Αυτή η γραφική παράσταση αποτελεί απαραίτητο εργαλείο για την πρόβλεψη ολοκλήρωσης του συνόλου της εργασίας ενός epic ή έργου. Αυτό επιτυγχάνεται με την ανανέωση της γραφικής παράστασης μετά το πέρας κάθε Sprint από τον Scrum Master της ομάδας. Το Release Burndown Chart είναι ένα απαραίτητο εργαλείο για την ομάδα ανάπτυξης για τους ακόλουθους λόγους:

- Βοηθάει να έχουμε μια εικόνα σχετικά με ποια έργα ασχολήθηκαν οι ομάδες ανάπτυξης σε κάθε Sprint.
- Δείχνει ξεκάθαρα την πρόοδο της ομάδας ανάπτυξης και τον υπολειπόμενο όγκο εργασίας που απομένει για την ολοκλήρωση των έργων.
- Το γράφημα βοηθά την ομάδα ανάπτυξης αλλά και τον οργανισμό να γνωρίζει εάν όλα προχωρούν ομαλά και εντός του καθορισμένου χρόνου για όλα τα έργα που έχει αναλάβει να παραδώσει η ομάδα ανάπτυξης.
- Τα ζητήματα, τα προβλήματα και τα ρίσκα μπορούν να αναγνωριστούν χρησιμοποιώντας το γράφημα πολύ νωρίς και μπορούν να ληφθούν μέτρα για την αντιμετώπισή τους.
- Βοηθά να επικεντρωθούν οι προσπάθειες της ομάδας προς τη σωστή κατεύθυνση.
- Δίνει κίνητρο και βοηθά στην επίτευξη στόχων της ομάδας ανάπτυξης σχετικά με τη δουλειά της.
- Δείχνει στην Ομάδα Ανάπτυξης την επιτυχία συγκεκριμένων Sprints, σε συσχέτιση με τα προκαθορισμένα έργα και epics.

Για την δημιουργία αυτού του γραφήματος, θα πρέπει να αθροίσουμε το σύνολο της εργασίας που απομένει για την ολοκλήρωση του έργου από το Product Backlog. Στη συνέχεια θα πρέπει να ορίσουμε τις εκδόσεις του έργου και βάση της δυναμικότητας (capacity) της ομάδας ανάπτυξης, να τις αντιστοιχήσουμε σε Sprints. Έτσι λοιπόν θα έχουμε όλα τα απαραίτητα δεδομένα για την δημιουργία του Release Burndown Chart.

3.3.7. Velocity Rate/Chart & Velocity Deviation

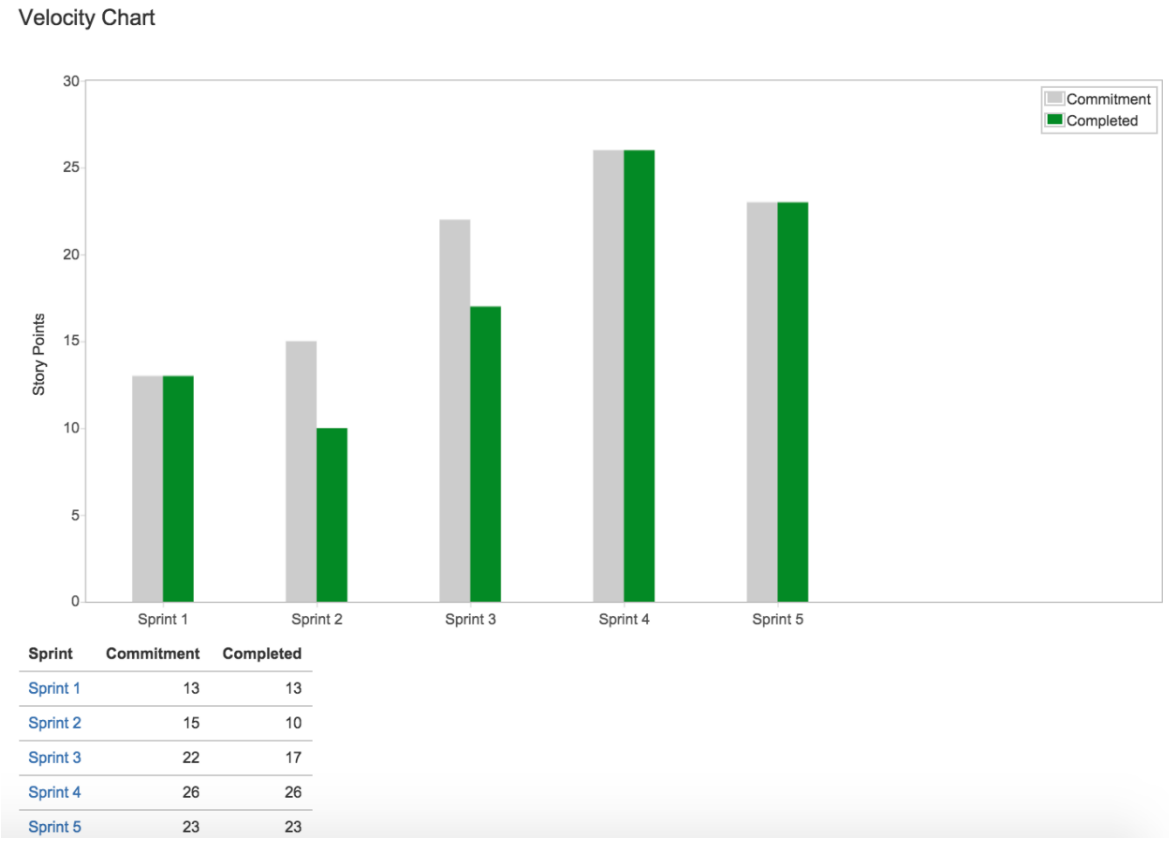
Όπως έχουμε δει και σε προηγούμενη ενότητα, το Velocity αποτελεί ο μέσος αριθμός από Story Points που μπορεί να ολοκληρώσει μια ομάδα ανάπτυξης κατά τη διάρκεια ενός Sprint. Αποτελεί εργαλείο για την πρόβλεψη δέσμευσης της ομάδας (team commitment). Ο Product Owner μπορεί να χρησιμοποιήσει αυτό το μετρικό για να προβλέψει πόσο γρήγορα μια ομάδα μπορεί να εργαστεί πάνω στο Product Backlog και να προβλέψει την παράδοση ενός ή και περισσότερων έργων.

Το Velocity Chart αποτυπώνει την πρόβλεψη δέσμευσης έναντι της ολοκληρωμένης εργασίας της ομάδας ανάπτυξης σε πολλά Sprints. Όσο περισσότερα δεδομένα για Sprints υπάρχουν για μια ομάδα με παρόμοια σύνθεση, τόσο πιο ακριβής θα είναι η πρόβλεψη. Για να καταλάβουμε καλύτερα αυτό που περιγράφουμε παραπάνω περιγράφεται το εξής παράδειγμα. Έστω ότι μια ομάδα ανάπτυξης έχει προβλέψει ότι για την ολοκλήρωση ενός epic χρειάζεται συνολικά πεντακόσια (500) Story Points και έστω ότι το Velocity της ίδιας ομάδας είναι πενήντα (50) Story Points κατά μέσο όρο για τα προηγούμενα έξι (06) Sprints. Έτσι μπορεί να προβλεφθεί πως η ομάδα ανάπτυξης θα χρειαστεί συνολικά δέκα (10) Sprints για την ολοκλήρωση του epic με την προϋπόθεση ότι δεν θα εργάζεται παράλληλα και για άλλα epics.

Το Velocity Chart είναι ένα απαραίτητο εργαλείο για την ομάδα ανάπτυξης για τους ακόλουθους λόγους:

- Βοηθάει την ομάδα ανάπτυξης να κάνει καλύτερες προβλέψεις.
- Αποτελεί το κύριο εργαλείο για την πρόβλεψη του χρόνου παράδοσης έργων.
- Βοηθάει την ομάδα ανάπτυξης να παρουσιάσει και να εμβαθύνει σε πιθανά προβλήματα που είχε ένα Sprint.

Να σημειωθεί πως το Velocity δεν αποτελεί μια ακριβής μονάδα μέτρησης μιας και τα Sprints επηρεάζονται από πολλούς παράγοντες όπως για παράδειγμα νέα μέλη που μπαίνουν στην ομάδα ανάπτυξης, νέες διαδικασίες, αλλαγές στο πεδίο εφαρμογής του έργου, κλπ. Όμως μπορεί να αποτελέσει ένα εργαλείο για την μελέτη αποτελεσματικότητας που μπορεί να επιφέρει μια αλλαγή διαδικασίας ή και της Ομάδας Ανάπτυξης. Επιπλέον αξίζει να σημειωθεί πως το Velocity δεν μπορεί να αποτελέσει αντικείμενο σύγκρισης 2 διαφορετικών ομάδων ανάπτυξης.



Sprint	Commitment	Completed
Sprint 1	13	13
Sprint 2	15	10
Sprint 3	22	17
Sprint 4	26	26
Sprint 5	23	23

Εικόνα 8: *Παράδειγμα ενός Velocity Chart από το JIRA.*

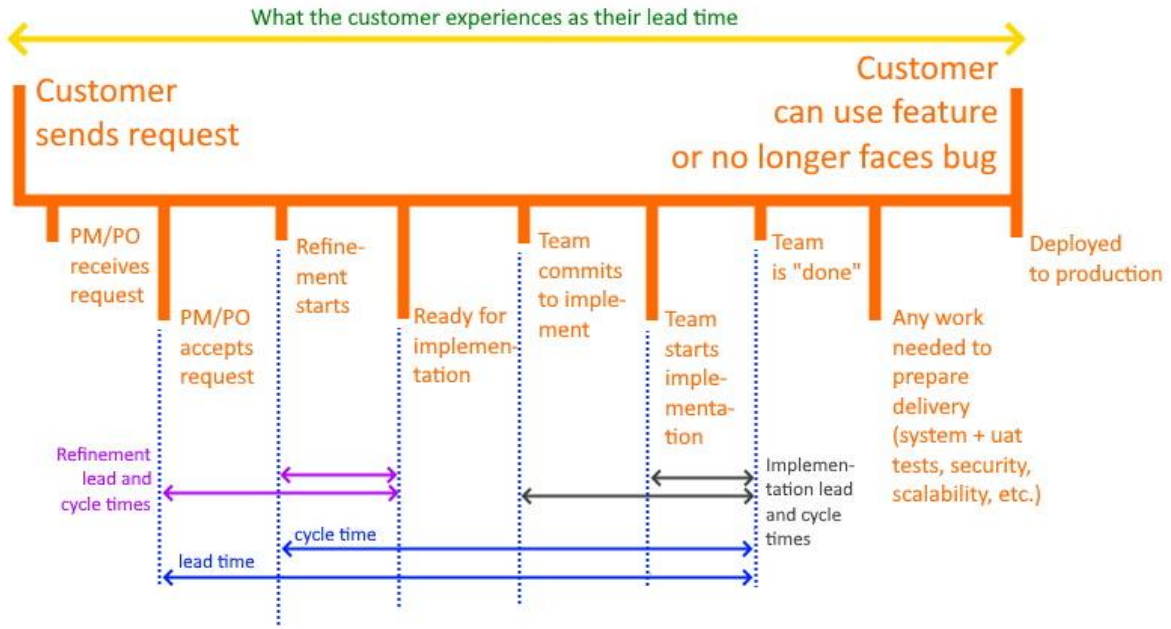
Στην Εικόνα 8 παρουσιάζεται ένα παράδειγμα ενός Velocity Chart. Με γκρι χρώμα έχουμε την πρόβλεψη δέσμευσης της ομάδας ανάπτυξης και με πράσινο το ποσό της εργασίας που παραδόθηκε από την ομάδα ανάπτυξης. Στον άξονα x έχουμε τα Sprints (5 στο δικό μας παράδειγμα) ενώ στον άξονα y τα Story Points. Από το συγκεκριμένο παράδειγμα μπορούμε να αναγνωρίσουμε πως η ομάδα ανάπτυξης παρέδωσε όλη την εργασία που είχε προβλεφθεί για τα Sprints: 1, 4 και 5. Αντίθετα φαίνεται ότι δεν κατάφερε να παραδώσει την εργασία που είχε προβλεφθεί για τα Sprints: 2 και 3. Επίσης για το επόμενο Sprint μπορούμε να υπολογίσουμε το Velocity της ομάδας προσθέτοντας το

σύνολο των Story Points τα οποία παραδόθηκαν προς τον αριθμό των Sprints: (13+10+17+26+23)/5= 17.8.

3.3.8. Cycle Time & Lead Time – Control Chart

Οι έννοιες Cycle Time και Lead Time αποτελούν έννοιες οι οποίες χρησιμοποιούνται στις γραμμές παραγωγής. Πιο συγκεκριμένα, ο όρος Cycle Time αποτελεί ο χρόνος από την εκκίνηση ενός task ή user story έως την χρονική στιγμή που είναι έτοιμη για παράδοση. Από την άλλη, ο όρος Lead Time γνωστός ως Lean και Toyota Production System, αποτελεί ο χρόνος από την δημιουργία της ανάγκης έως την χρονική στιγμή που είναι έτοιμη για παράδοση. Αυτά τα δύο μετρικά χρησιμοποιούνται κυρίως στο Kanban framework, μιας και τα αιτήματα εξυπηρετούνται σειριακά.

Τα δύο μετρικά περιγράφουν τον χρόνο που απαιτείται για την τελική παράδοση ενός user story ή task. Η οδοποιός διαφορά των δύο αποτελεί ο χρόνος στον οποίο ένα user story ή task βρίσκεται στο backlog της ομάδας ανάπτυξης το οποίο μπορεί να υπολογιστεί εάν για ένα συγκεκριμένο user story αφαιρέσουμε το Cycle Time από το Lead Time.



Εικόνα 9: *Παράδειγμα Cycle Time και Lead Time [27].*

Για την αποτύπωση των Lead και Cycle Time μετρικών σε διαγράμματα υπάρχουν πολλές πρακτικές. Εμείς θα παρουσιάσουμε το διάγραμμα Control Chart που παρουσιάζεται στην Εικόνα 9 το οποίο χρησιμοποιείται για την απεικόνιση των Lead και Cycle Time μετρικών.

Το Control Chart μπορεί να χρησιμοποιηθεί για:

- Την αναδρομική ανάλυση της απόδοσης της ομάδας.
- Την μέτρηση της επίδρασης μιας αλλαγής διαδικασίας στα επίπεδα παραγωγικότητας της ομάδας ανάπτυξης.
- Την παρουσίαση της απόδοσης της ομάδας ανάπτυξης σε εξωτερικούς ενδιαφερόμενους.

- Για τον ορισμό νέων στόχων της ομάδας ανάπτυξης σχετικά με την παραγωγικότητά της.

Εικόνα 10: *Παράδειγμα ενός Control Chart από το JIRA.*

Έτσι λοιπόν, στον άξονα x αποτυπώνονται οι ημερομηνίες μετάβασης των εργασιών από τις επιλεγμένες καταστάσεις βάση της ροής εργασίας της ομάδας ανάπτυξης[6]. Ο άξονας y παρουσιάζει τον χρόνο (ημέρες) που έχουν παρέλθει για ένα user story ή task που δουλεύεται – όσο χαμηλότερα είναι τα νούμερα τόσο καλύτερα τα αποτελέσματα. Η κόκκινη γραμμή αναδεικνύει τον μέσο χρόνο που χρειάζεται η ομάδα ανάπτυξης για μετακινήσει ένα user story ή task από την μία κατάσταση στην άλλη (από τις επιλεγμένες). Η μπλε γραμμή αποτυπώνει τον κινητό μέσο όρο που χρειάζεται η ομάδα ανάπτυξης για μετακινήσει ένα user story ή task από την μία κατάσταση στην άλλη (από τις επιλεγμένες). Η σκιασμένη μπλε περιοχή παρουσιάζει την τυπική απόκλιση μεταξύ του μεμονωμένου σημείου δεδομένων (πράσινες κουκκίδες που αποτελούν είτε ένα user story ή task ή ένα σύνολο αυτών) και του κυλιόμενου μέσου όρου.

3.3.9. Μετρικά νέων κυκλοφοριών (Release metrics)

Τα Μετρικά των νέων κυκλοφοριών δίνουν την δυνατότητα σε έναν οργανισμό να παρακολουθεί τις νέες κυκλοφορίες των προϊόντων του και να προβλέπει προβληματικές καταστάσεις. Πιο συγκεκριμένα:

[6] Να σημειωθεί πως οι καταστάσεις που θα επιλέξουμε από την ροή εργασιών ορίζει στο Control Chart εάν θέλουμε να μελετήσουμε το Cycle ή Lead Time.

- Συχνότητα των προγραμματισμένων κυκλοφοριών: Αυτό το μετρικό περιγράφει την συχνότητα των πετυχημένων κυκλοφοριών ανά ένα προκαθορισμένο χρονικό διάστημα (ανά χρόνο, τετράμηνο, μήνα ή και βδομάδα).
- Συχνότητα των μη προγραμματισμένων κυκλοφοριών: Αυτό το μετρικό περιγράφει την συχνότητα των πετυχημένων μη προγραμματισμένων κυκλοφοριών ανά ένα προκαθορισμένο χρονικό διάστημα (ανά χρόνο, τετράμηνο, μήνα ή και βδομάδα). Οι μη προγραμματισμένες κυκλοφορίες γίνονται όταν κάποιο σφάλμα έχει αναγνωριστεί στην παραγωγή και η νέα κυκλοφορία είναι απαραίτητη για την επίλυσή του.
- Χρόνος που απαιτείται για την έκδοση νέων κυκλοφοριών: Αυτό το μετρικό εκφράζει τον χρόνο που απαιτείται από την υπεύθυνη ομάδα για την έκδοση νέων κυκλοφοριών. Αυτό το μετρικό εκφράζει τον χρόνο σε λεπτά ή και ώρες. Τέλος, από την συλλογή δεδομένων πολλών περιπτώσεων είναι δυνατός και ο υπολογισμός του μέσου χρόνου που μπορεί να μας δίνει και μια τάση (προς αύξηση ή μείωση) ανά περιόδους αλλά και συναρτήσει άλλων μεταβλητών όπως όγκου δουλειάς (σε Story Points) ή και αριθμό νέων δυνατοτήτων.
- Προγραμματισμένος χρόνος που η εφαρμογή δεν είναι διαθέσιμη λόγω έκδοσης νέας κυκλοφορίας: Πολλές εφαρμογές δεν χρειάζεται να μην είναι διαθέσιμες στην παραγωγή (downtime) για να εκδοθεί μια νέα κυκλοφορία. Σε κάθε περίπτωση όμως οι οργανισμοί είναι καλό να καταγράφουν και να παρακολουθούν τον χρόνο που απαιτείται ώστε η εφαρμογή να είναι πάλι διαθέσιμη στο ευρύ κοινό μετά από την έκδοση νέας κυκλοφορίας.
- Μη προγραμματισμένος χρόνος που η εφαρμογή δεν είναι διαθέσιμη λόγω έκδοσης νέας κυκλοφορίας: Ο μη προγραμματισμένος χρόνος που η εφαρμογή δεν είναι διαθέσιμη είναι απαραίτητο μετρικό για έναν οργανισμό μιας και περιγράφει κατά πόσο η υπηρεσία που παρέχουμε στους πελάτες είναι διαθέσιμη 24/7.

3.4. Μετρικά Ποιότητας

Οι μετρήσεις ποιότητας παρέχουν ορατότητα στον βαθμό στον οποίο τα προϊόντα που παραδίδονται προς τους τελικούς χρήστες λειτουργούν όπως προορίζονται να λειτουργήσουν. Ορισμένες μετρήσεις ποιότητας, όπως η κάλυψη δοκιμής μονάδας (unit testing coverage), αποτελούν κορυφαίοι δείκτες. Για παράδειγμα, αν πάρουμε δεδομένο ότι οι δοκιμές μονάδων δοκιμάζουν κάτι σημαντικό, η εκτέλεση των δοκιμών μονάδας μπορούν να αποτρέψουν την εμφάνιση προβλημάτων αργότερα. Άλλες μετρήσεις ποιότητας, όπως το ποσοστό διαφυγής ελαττωμάτων, είναι δείκτες καθυστέρησης, επειδή μας λένε μετά το γεγονός ότι έχουμε ένα κενό στην προσέγγιση δοκιμών μας.

Να σημειωθεί πως τα Μετρικά Ποιότητας και τα Μετρικά Παράδοσης συνδυαστικά, μπορούν να αποτελέσουν πολύ καλά εργαλεία για τις ομάδες ανάπτυξης. Πιο συγκεκριμένα, εάν μια ομάδα ανάπτυξης έχει πολύ καλή Αξιοπιστία και αυξάνει την ταχύτητα της κάθε 2 με 3 Sprints, αλλά παρουσιάζονται πολλά σφάλματα ανά επανάληψη (που θα αναλύσουμε παρακάτω), τότε θα πρέπει να μειώσει την Ταχύτητα παράδοσής της και να επικεντρωθεί στην ποιότητα των τελικών παραδοτέων.

3.4.1. Ελαττώματα και σφάλματα ανά επανάληψη & Χρόνος επίλυσης ελαττώματος/σφάλματος

Σε αυτό το σημείο είναι σημαντικό να ορίσουμε τους όρους ελαττώματα και σφάλματα τα οποία εμφανίζονται στον κώδικα που αναπτύσσει μια ομάδα ανάπτυξης λογισμικού.

> **Ελάττωμα (defect) πηγαίου κώδικα:** Όταν το πραγματικό αποτέλεσμα αποκλίνει από το αναμενόμενο αποτέλεσμα **κατά τη δοκιμή μιας εφαρμογής λογισμικού ή ενός προϊόντος**, αυτό αποτελεί ελάττωμα του πηγαίου κώδικα. Ως εκ τούτου, οποιαδήποτε απόκλιση από τις προδιαγραφές που αναφέρονται στο έγγραφο προδιαγραφών λειτουργίας του προϊόντος ή στο User Story όταν αυτό είναι υπό ανάπτυξη, τότε αυτό είναι ελάττωμα. Να σημειωθεί πως σε πολλούς οργανισμούς αυτό μπορεί να αναφέρεται ως σφάλμα.
> **Σφάλμα (bug) πηγαίου κώδικα:** Όταν το πραγματικό αποτέλεσμα αποκλίνει από το αναμενόμενο αποτέλεσμα **στο παραγωγικό περιβάλλον της εφαρμογής λογισμικού ή ενός προϊόντος**, αυτό αποτελεί σφάλμα του πηγαίου κώδικα. Ως εκ τούτου, οποιαδήποτε απόκλιση από τις προδιαγραφές που αναφέρονται στο έγγραφο προδιαγραφών λειτουργίας ή το User Story του προϊόντος στην παραγωγή, τότε αυτό ονομάζεται σφάλμα του πηγαίου κώδικα.

Έτσι λοιπόν τρία αποτελεσματικά μετρικά για την ποιότητα των παραδοτέων μιας ομάδας είναι:

1. Ο αριθμός των ελαττωμάτων ή των σφαλμάτων ανά επανάληψη: Για τις επαναλήψεις, όσο αφορά τα ελαττώματα μπορεί να μετρηθεί ανά Sprint ενώ για τα σφάλματα ανά εκδοχή του πηγαίου κώδικα. Επιπλέον οι μετρήσεις μπορούν να υπολογιστούν σταθμισμένα ως προς την σοβαρότητα των ελαττωμάτων ή των σφαλμάτων που βρέθηκαν (μικρής, μεσαίας, μεγάλης) χρησιμοποιώντας βάρη. Παραδείγματα υπολογισμών:
   $$Mean\ Defect\ OR\ Bug\ Rate = \frac{\sum_{i=-6}^{0} Total\ number\ of\ Defects\ OR\ Bugs\ of\ Sprint_n}{6}$$
   $$Mean\ Defect\ OR\ Bug\ Rate = \frac{\sum_{i=-6}^{0}(x_n * Total\ number\ of\ Defects\ OR\ Bugs\ of\ Sprint_n)}{6}$$
   Όπου με x_nτα βάρη των ελαττωμάτων ή σφαλμάτων που έχουν οριστεί.
2. Το ποσοστό διαφυγής ελαττωμάτων περιγράφει το ποσοστό των ελαττωμάτων τα οποία κατέληξαν να βγουν στην παραγωγή και τελικά να καταγραφούν σαν σφάλματα προς το σύνολο των ελαττωμάτων. Να σημειωθεί σε αυτό το σημείο ότι τα σφάλματα αποτελούν τα πιο κοστοβόρα ως προς την επίλυση μιας και η πιθανότητα να χρειάζονται περισσότεροι πόροι για την επίλυσή τους είναι αυξημένοι (πόροι ανάλυσης, σχεδιασμού και υλοποίησης).
   $$Defect\ escape\ rate = \frac{number\ of\ bugs\ recordered\ in\ production\ after\ the\ release}{number\ of\ defects\ recordered\ in\ staging\ before\ the\ release}$$
3. Ο μέσος χρόνος απόκρισης της ομάδας ανάπτυξης από την αναφορά των ελαττωμάτων ή των σφαλμάτων μέχρι την επίλυσή τους.

3.4.2. Κάλυψη δοκιμών μονάδας και αυτόματων δοκιμών

Η κάλυψη δοκιμών μονάδας (unit test coverage) περιγράφει το ποσοστό του κώδικα το οποίο μπορεί να δοκιμαστεί μεμονωμένα. Για αυτό το σκοπό υπάρχουν πολλά πλαίσια δοκιμής μονάδων, συμπεριλαμβανομένων των MSTest, NUnit, xUnit και MSpec. Το θετικό των δοκιμών μονάδας είναι ότι δίνεται μια κατά προσέγγιση απεικόνιση του πόσο καλά ελεγμένη είναι μια βάση κώδικα λογισμικού. Από την άλλη, οι δοκιμές μονάδων είναι ακριβώς αυτό, μια δοκιμή μιας ενιαίας μονάδας. Όλες οι μονάδες σε ένα σύστημα μπορεί να λειτουργούν τέλεια, αλλά αυτό δεν εγγυάται ότι το σύστημα τελικά θα κάνει αυτό που είναι σχεδιασμένο να κάνει. Για αυτό το λόγο, οι δοκιμές ενσωμάτωσης (integration) και αποδοχής (acceptance) είναι ζωτικής σημασίας για τη διασφάλιση της λειτουργικότητας του λογισμικού και η κάλυψη δοκιμών μονάδας δεν λαμβάνει υπόψη αυτές τις δοκιμές.

Για να μπορεί μια ομάδα ανάπτυξης να γνωρίζει κατά πόσο το σύστημα κάνει τελικά αυτό που είναι σχεδιασμένο να κάνει, τρέχει ένα σύνολο από δοκιμές οι οποίες περιγράφουν όλα τα μονοπάτια τα οποία μπορεί να ακολουθήσει ένας χρήστης της εφαρμογής. Αυτές οι δοκιμές αρχικά καταγράφονται και εκτελούνται με χειροκίνητο τρόπο. Όσο το σύστημα μεγαλώνει, το να τρέχουν πριν από κάθε έκδοση όλες οι δοκιμές χειροκίνητα γίνεται όλο και πιο δύσκολο και χρονοβόρο. Για αυτό το λόγο έχουν αναπτυχθεί εργαλεία τα οποία υποστηρίζουν την αυτοματοποίηση αυτών των δοκιμών. Έτσι λοιπόν, η κάλυψη των αυτόματων δοκιμών περιγράφει το ποσοστό του συνόλου των δοκιμών που γίνονται με αυτόματο τρόπο έναντι του συνολικού αριθμού των δοκιμών που έχουν καταγραφεί.

$$Automation\ test\ coverage = \frac{number\ of\ automation\ tests}{number\ of\ total\ tests}$$

Τα παραπάνω υποστηρίζουν μηχανισμούς που δίνουν αποτελέσματα σχετικά με το ποσοστό επιτυχίας των δοκιμών μετά την εκτέλεσή τους. Όπως για παράδειγμα το ποσοστό επιτυχίας αυτοματισμού το οποίο είναι ένα τυπικό ποσοστό του πόσες δοκιμές αυτοματισμού πέρασαν. Αυτό το ποσοστό είναι χρήσιμο για την κατανόηση της σταθερότητας της σουίτας αυτοματισμού καθώς και της αποτελεσματικότητάς της. Επιπλέον, ο χρόνος ο οποίος απαιτείται για να ελεγχθεί όλος ο πηγαίος κώδικας μέσω των αυτόματων δοκιμών αποτελεί άλλο ένα μετρικό σημαντικό για την αποτελεσματικότητα του κώδικα της σουίτας δοκιμών.

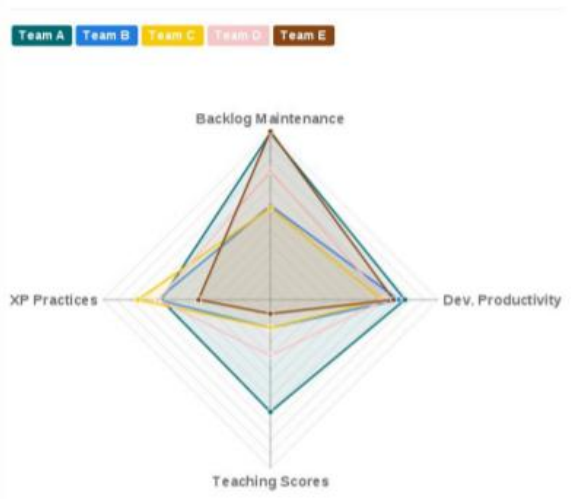
3.4.3. Τυπική παράβαση - Standard violation

Η Τυπική παράβαση (Standard violation) μπορεί να χρησιμοποιηθεί για την παρακολούθηση του αριθμού των προτύπων που παραβιάζονται ανά Sprint. Η μέτρηση χρησιμοποιείται για την επιβολή προτύπων κωδικοποίησης ή σχεδίασης. Η χρήση αυτής της μέτρησης επιβάλλει πειθαρχία εντός της ομάδας σχετικά με την ποιότητα του κώδικα και των παραδοτέων. Επομένως, αυτή η μέτρηση χρησιμοποιείται συχνά για να κατευθύνει την ομάδα προς τις σωστές συμπεριφορές κατά την ανάπτυξη.

Πιο συγκεκριμένα, μερικές προτάσεις και μετρικά που μπορούν να χρησιμοποιηθούν ως πρότυπα είναι τα ακόλουθα:

- Αριθμός των εργασιών/Story Points που προστίθενται στο Sprint μετά την εκκίνησή του.
- Αριθμός των εργασιών που δεν ακολουθούν το προκαθορισμένο από την ομάδα ανάπτυξης Definition of Done (DoD). Το DoD αποτελεί μια μορφή λίστας κριτηρίων που βοηθάει μια ομάδα ανάπτυξης να ορίσει εάν μία εργασία έχει παραδοθεί.
- Αριθμός των εργασιών που προστίθενται στο Sprint χωρίς να είναι έτοιμα. Να σημειωθεί πως στο Scrum υπάρχει ο όρος Definition of Ready (DoR). Το DoR αποτελεί μια μορφή λίστας ελέγχου κριτηρίων που θα βοηθούσε στη λήψη απόφασης μιας ομάδας ανάπτυξης σχετικά με το αν θα ξεκινήσει να εργάζεται σε κάτι.
- Αριθμός των εργασιών που εμφανίζονται σε πολλά Product Backlogs διαφορετικών ομάδων ανάπτυξης.
- Αριθμός μεγάλων εργασιών που βρίσκονται στο Product Backlog.

- Αριθμός των ατελειών (defects) ανά Sprint ή ανά εργασία.



Εικόνα 11: *Ένα διάγραμμα radar το οποίο απεικονίζει 5 ομάδες ανάπτυξης και 4 κατηγορίες τυπικής παράβασης (Backlog Maintenance, XP Practices, Dev. Productivity, Teaching Scores) [4].*

Τα παραπάνω μπορούν να χρησιμοποιηθούν ως βάση για την ανάπτυξη μετρικών μελέτης αποτελεσματικότητας της ομάδας ανάπτυξης αλλά και ολόκληρου του συστήματος που αλληλοεπιδρά με την ομάδα για την σχεδίαση και ανάπτυξη εργασιών. Τα μετρικά τυπικής παράβασης μπορούν να χωριστούν σε κατηγορίες όπως φαίνεται στην Εικόνα 11. Αυτές οι κατηγορίες μπορούν να βοηθήσουν την ίδια την ομάδα ανάπτυξης όπως και το σύστημα να καταλάβει και εν συνεχεία να βελτιώσει τα αδύναμά της σημεία.

3.4.4. Υποστήριξη μετά την παράδοση – After Release Support

Αυτό το μετρικό περιγράφει την συνολική ομαδική ενέργεια που πρέπει να καταβάλει η ομάδα ανάπτυξης – σε Story Points – μετά την ενεργοποίηση μιας νέας έκδοσης ή ενός νέου τεχνικού χαρακτηριστικού στην παραγωγή.

Με τη χρήση αυτού του μετρικού η ομάδα ανάπτυξης έχει την δυνατότητα να προβλέπει την συνολική προσπάθεια που απαιτείται κατά την διάρκεια της ενεργοποίησης νέων υλοποιήσεων στην παραγωγή. Επίσης, την βοηθά να παρακολουθεί και να αντιλαμβάνεται κατά πόσο οι υλοποιήσεις που παραδίδει είναι ποιοτικές. Εάν δηλαδή μια ομάδα χρειάζεται x Story Points για την υλοποίηση και παράδοση μιας νέας έκδοσης και παράλληλα παρατηρήσουμε ότι χρειάζεται και επιπλέον Story Points που ξεπερνάνε το 20% του x, τότε η ομάδα ανάπτυξης θα πρέπει να βελτιώσει την ποιότητα των παραδοτέων της. Επίσης αυτό το μετρικό μας βοηθάει να έχουμε μια καλύτερη εικόνα του τεχνικού χρέους το οποίο μπορεί να μεγαλώνει όσο παράγεται νέος κώδικας και ανεβαίνει στην παραγωγή. Έτσι για παράδειγμα, εάν παρατηρείται αύξηση του ποσοστού του χρόνου που απαιτείται από την ομάδα ανάπτυξης ώστε να υποστηρίξει την νέα υλοποίηση στην παραγωγή έναντι του χρόνου που απαιτείται για να υλοποιήσει την υλοποίηση αυτή, τότε σημαίνει ότι παράλληλα αυξάνεται και το τεχνικό χρέος.

Τέλος, ο αριθμός των Sprint ο οποίος απαιτείται για την υποστήριξη μιας νέας υλοποίησης από την ομάδα ανάπτυξης μπορεί να συζητηθεί και να συμφωνηθεί με την ίδια την ομάδα, καλό είναι όμως να είναι σταθερός και προτείνεται να είναι δύο Sprints.

3.5. Μετρικά Προστιθέμενης Αξίας Προϊόντος

Οι μετρήσεις αξίας παρέχουν ορατότητα σε έναν οργανισμό για τον βαθμό στον οποίο τα προϊόντα που παρέχουν επιτρέπουν στους χρήστες τους να επιτύχουν τους στόχους τους και στον βαθμό στον οποίο οι επενδύσεις του οργανισμού παρέχουν αποδόσεις που αναμένονταν.

3.5.1. Παραδοθείσα Επιχειρηματική Αξία

Η Παραδοθείσα Επιχειρηματική Αξία μπορεί να μετρηθεί με βάση τα Story Points, τον αριθμό των εργασιών ή ένα αφηρημένο μέτρο που μετρά την αξία που αποδίδει η επιχείρηση σε ένα χαρακτηριστικό ή μια εργασία. Πολλά από τα αποτελέσματα αυτών των μετρικών αποτελούν ανατροφοδότηση από τους ίδιους τους χρήστες του προϊόντος πράγμα που βοηθάει της ομάδες ανάπτυξης και τους υπεύθυνους προϊόντος να προσαρμόζονται στις ανάγκες των χρηστών τους.

Πιο συγκεκριμένα μια επιχείρηση μπορεί να ορίσει την επιχειρηματική αξία ως μια σειρά μεταβλητών τα οποία έχουν διαφορετικά βάρη. Σαν παραδείγματα μπορούμε να χρησιμοποιήσουμε τα συνολικά έσοδα της επιχείρησης, την κερδοφορία, την πίστη των πελατών, το ποσοστό διατήρησης πελατών, το μερίδιο αγοράς, κλπ.

Αυτό το μετρικό μπορεί να βοηθήσει την ομάδα ανάπτυξης να υπολογίσει την ταχύτητά της και να καθορίσει το χρόνο για την ολοκλήρωση μιας έκδοσης. Για παράδειγμα, όταν πραγματοποιηθεί το 80 τοις εκατό της επιθυμητής επιχειρηματικής αξίας, μπορεί κάποιος να αποφασίσει ότι αρκεί να ονομαστεί ως μια πρώτη έκδοση.

Μερικά από τα πιο διαδεδομένα μετρικά παραδοθείσας αξίας που μπορεί ένας οργανισμός να συλλέξει από τους ίδιους τους πελάτες του είναι:

- Ο Δείκτης Προσπάθειας Πελατών - Customer Effort Score (CES): Ένας γενικός ορισμός του Δείκτη Προσπάθειας Πελατών είναι μια έρευνα πελατών που μετρά πόσο εύκολο ήταν για αυτούς να αλληλοεπιδράσουν με μια επιχείρηση ή ένα προϊόν της (επίλυση ενός προβλήματος με την υποστήριξη πελατών, πραγματοποίηση μιας αγοράς, εγγραφή σε μια δοκιμή κ.λπ.).
- Η Βαθμολογία Ικανοποίησης Πελατών - Customer Satisfaction Score (CSAT): Όμοια με τον Δείκτη Προσπάθειας Πελατών, μέσα από μια έρευνα πελατών ο λόγος των θετικών απαντήσεων των πελατών προς τις αρνητικές απαντήσεις μπορεί να μας δώσει το ποσοστό το οποίο περιγράφει την Βαθμολογία Ικανοποίησης Πελατών.
- Καθαρή Βαθμολογία Υποστηρικτών - Net Promoter Score (NPS): Η Καθαρή Βαθμολογία Υποστηρικτών είναι μια μέτρηση αφοσίωσης και ικανοποίησης πελατών. Αυτή λαμβάνεται από τους ίδιους τους πελάτες μέσω της ερώτησης “Πόσο πιθανό είναι να προτείνετε το προϊόν ή την υπηρεσία αυτή σας σε άλλους;” Μέσω μια κλίμακας από το 0 έως το 10.

3.6. Μετρικά για Αρχιτεκτονικές Λογισμικών και Οντολογιών

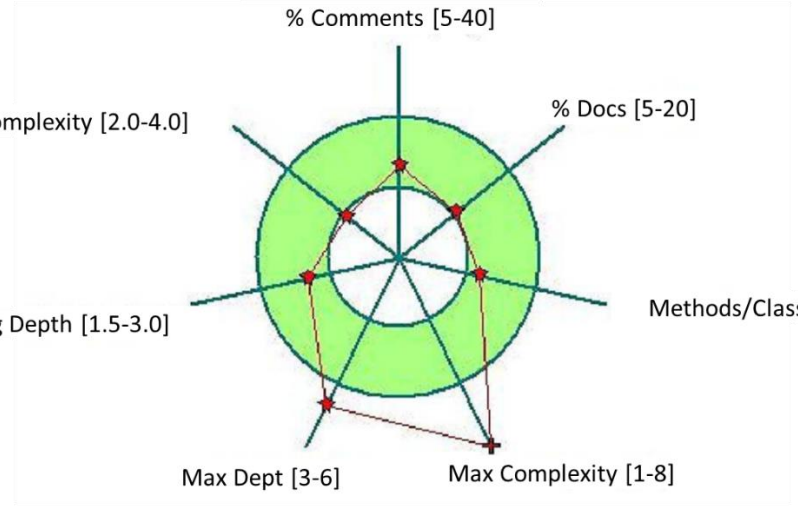
Σε αυτό το κεφάλαιο θα παρουσιάσουμε ένα σύνολο από εργαλεία και μετρικά τα οποία μας επιτρέπουν να γνωρίζουμε ποια μέρη ενός συστήματος – πηγαίου κώδικα – παρουσιάζουν προβλήματα, ποια είναι η πολυπλοκότητα και η κατανάλωση μνήμης κλπ. Αυτές οι τεχνικές και οι μετρήσεις μπορούν να εφαρμοστούν συμπληρωματικά στα μετρικά που παρουσιάσαμε στις προηγούμενες ενότητες ώστε οι ομάδες ανάπτυξης και οι ενδιαφερόμενοι να έχουν μια ολιστική εικόνα για την ίδια την ομάδα, το προϊόν και τον πηγαίο κώδικα του προϊόντος – σε περιπτώσεις όπου το προϊόν αποτελεί μια πλατφόρμα ή μια εφαρμογή, κλπ.

Ορισμένες τεχνικές και εργαλεία αναπτύσσονται με την πάροδο του χρόνου στη μηχανική λογισμικού για την εκτέλεση διαγνωστικών εφαρμογών και την οπτικοποίηση μετρικών σχετικών με μεγάλα κομμάτια κώδικα για την κατανόηση των προδιαγραφών και των επιδόσεών του. Πράγμα που βοηθάει τους μελετητές του να έχουν μια γενική εικόνα χωρίς να χρειάζεται να ψάχνουν συνεχώς μικρά κομμάτια κώδικα και να χάνουν πολύ χρόνο για να σχηματίσουνε μια γενικότερη εικόνα σχετικά με την ποιότητα του κώδικα. Έτσι λοιπόν, μια ομάδα ανάπτυξης που θέλει να εστιάσει στον ανασχεδιασμό του πηγαίου κώδικα για να τον κάνει πιο αποτελεσματικό και να έχει αποτελέσματα, μπορεί να χρησιμοποιήσει αυτές τις τεχνικές και τα εργαλεία για να το πετύχει. Επιπλέον, κατά την διάρκεια των Sprints οι ομάδες ανάπτυξης μπορούν να αφιερώνουν ένα ποσοστό του capacity τους για την διαχείριση του Technical Dept (ο όρος αυτός δεν αναλύεται επιπλέον στα πλαίσια αυτής της διπλωματικής εργασίας).

Αρχικά για να μπορέσουμε να παρουσιάσουμε και να αναλύσουμε αυτά τα εργαλεία και τις τεχνικές αναπαράστασης δεδομένων, θα πρέπει να ορίσουμε ένα σύνολο Οντολογιών. Έχουν γίνει άφθονες έρευνες στον τομέα των μετρήσεων λογισμικού. Έχουν προταθεί πολλές μετρήσεις και μέτρα αντικειμενοστρεφούς λογισμικού και έχουν γίνει κάποιες κύριες προτάσεις και εργασίες [5]. Για την απλούστευση της ανάλυσης σε αυτό το σημείο να σημειωθεί ότι στο πλαίσιο αυτής της διπλωματικής εργασίας δεν θα εμβαθύνουμε στον όρο Οντότητες άλλα θα παρουσιάσουμε μετρικά και τρόπους αναπαράστασης δεδομένων μεγάλων συστημάτων. Έτσι λοιπόν, οι ακόλουθες ενότητες περιγράφουν ένα σύνολο γνώσεων λογισμικού και τεχνικών αρχιτεκτονικής ανάλυσης που θα έχουν μεγάλο αντίκτυπο στη διαδικασία ανάλυσης της αρχιτεκτονικής, της εξέλιξης του πηγαίου κώδικα, της αξιολόγησης και αναθεώρησης μεγάλων κομματιών πηγαίου κώδικα.

3.6.1. Γραφήματα Matrix Kiviat

Πολλές φορές, χρειάζονται περισσότερες από μία μετρήσεις για την κατανόηση, την αξιολόγηση ή τον έλεγχο ενός προϊόντος λογισμικού, μιας διαδικασίας, μιας υπηρεσίας ή ενός έργου. Ένας τρόπος για να εμφανιστεί μια συνοπτική προβολή ενός συνόλου μετρήσεων είναι να χρησιμοποιήσουμε ένα γράφημα Kiviat, που ονομάζεται επίσης πολικό γράφημα, διάγραμμα ραντάρ ή διάγραμμα αράχνης [6]. Τα γραφήματα Kiviat αποτελούν ευέλικτο εργαλείο γραφικής απεικόνισης, το οποίο παρουσιάζει διαφορετικές μετρήσεις έναντι μέγιστων και ελάχιστων σημείων αναφοράς (Εικόνα 12).



Εικόνα 12: *Διάγραμμα Kiviat [28].*

Κατά την εφαρμογή αυτής της αρχιτεκτονικής αναπαράστασης σε Μεγάλες Οντολογίες ή Οντολογίες που δημιουργούνται σε συνεργασία μεταξύ ομάδων, οποιοσδήποτε αρχιτέκτονας γνώσης θα έχει λεπτομερείς πληροφορίες σχετικά με την κατάσταση της οντολογίας και όλα τα προβλήματα μετρήσεων της Οντολογίας αυτής.

Ένα ακόμη εξαιρετικό βήμα είναι να συνδυαστεί αυτό το γράφημα με προηγούμενες εκδόσεις οντολογίας που ελέγχονται από σύστημα ελέγχου πηγής. Επομένως, το διάγραμμα Kiviat θα έχει μια απεικόνιση των χρονοεξαρτώμενων, πολυμεταβλητών συνόλων δεδομένων σχετικών με την οντολογία κατά την διαρκεια εξέλιξης της οντολογίας. Επίσης, θα έχουμε ένα ιστορικό για κάθε πίνακα οντολογίας που μας δίνει τη δυνατότητα να καταλάβουμε πότε ακριβώς κάποια μήτρα οντολογίας είχε πρόβλημα ή όταν κάποια μήτρα ξεκίνησε την απόκλιση των τυπικών γνωστών τιμών για τη συγκεκριμένη μήτρα.

Έτσι για κάθε μήτρα Οντολογίας αυτό που προτείνεται να μετράται είναι:

Ο αριθμός των σχόλιων στον πηγαίο κώδικα συναρτήσει του συνολικού κώδικα. Αυτός ο λόγος μπορεί να εξαχθεί σε επίπεδο χαρακτήρων ή και γραμμών. Επίσης, το ποσοστό στο οποίο είναι αποδεκτό να μην υπάρχουν σχόλια και να είναι αποδεκτό, μπορεί να οριστεί από τον αρχιτέκτονα γνώσης ανάλογα από την περίπτωση.

Η μέση πολυπλοκότητα του κώδικα που μπορεί να μετράται από τον μέσο όρο των ένθετων συνθηκών εντός του κώδικα που δημιουργούνται από ενσωματωμένα IF, ELSE, WHILE, FOR κλπ.

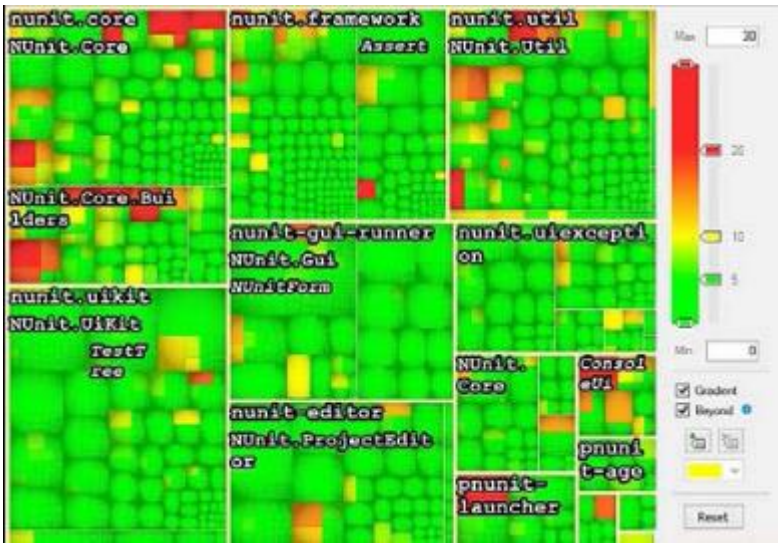
Την μέγιστη πολυπλοκότητα του κώδικα που μπορεί να μετράται από τον απόλυτο αριθμό των ένθετων συνθηκών στο πιο σύνθετο σημείο του κώδικα. Αυτό το σημείο μπορεί να εμφανίζεται στον αρχιτέκτονα γνώσης για να τον βοηθήσει να το αναζητήσει στον κώδικα και να το διορθώσει.

Το μέσο τεχνικό χρέος, που μπορεί να μετράται από το σύνολο του κώδικα που δεν χρησιμοποιείται καθόλου προς το σύνολο του κώδικα. Πάλι σε αυτό το σημείο οι δύο λόγοι μπορούν να υπολογίζονται από τον αριθμό των χαρακτήρων, των γραμμών κα.

Το μέγιστο τεχνικό χρέος, που αναγνωρίζει το σημείο του κώδικα που δεν χρησιμοποιείται καθόλου και είναι μεγάλο.

3.6.2. Χάρτες Θερμότητας – Heat Maps

Ένας χάρτης θερμότητας (Εικόνα 13) είναι μια γραφική αναπαράσταση δεδομένων όπου οι μεμονωμένες τιμές που περιέχονται σε μια μήτρα αναπαρίστανται ως χρώματα.

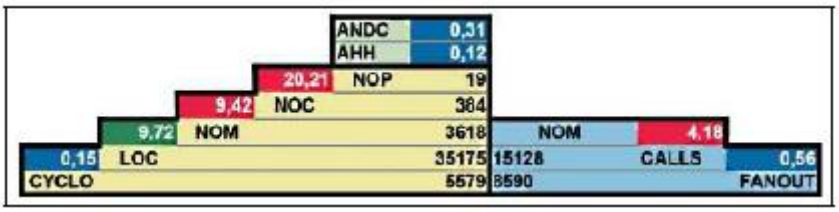


Εικόνα 13: *Χάρτης θερμότητας [28].*

Όλες οι μετρήσεις οντολογίας που συζητήθηκαν μπορούν να αναπαρασταθούν ως χάρτης θερμότητας. Ένας αρχιτέκτονας γνώσης μπορεί να καταλάβει από έναν θερμικό χάρτη μιας συγκεκριμένης μήτρας οντολογίας γρήγορα ποια μέρη έχουν προβλήματα. Αυτό είναι ένα εύχρηστο και χρήσιμο εργαλείο ειδικά σε περίπτωση μεγάλων οντολογιών ή σε περίπτωση οντολογιών που εξελίσσονται συνεχώς από πολλές ομάδες ανάπτυξης λογισμικού.

3.6.3. Πυραμίδες Οπτικοποίησης Μεγέθους και Πολυπλοκότητας – Size and Complexity Pyramid

Ο σκοπός της πυραμίδας μεγέθους και πολυπλοκότητας (επίσης γνωστή ως πυραμίδα επισκόπησης) είναι να παρέχει μια επισκόπηση υψηλού επιπέδου οποιουδήποτε συστήματος. Αυτό το επιτυγχάνει συγκεντρώνοντας σε ένα μέρος μερικές από τις πιο σημαντικές μετρήσεις ενός συστήματος λογισμικού (Μέγεθος και πολυπλοκότητα, σύζευξη και κληρονομικότητα). Το αριστερό μέρος περιγράφει το μέγεθος και την πολυπλοκότητα, ενώ το δεξιό μέρος περιγράφει τη σύζευξη συστήματος. Το επάνω μέρος περιγράφει τη χρήση κληρονομικότητας [7].

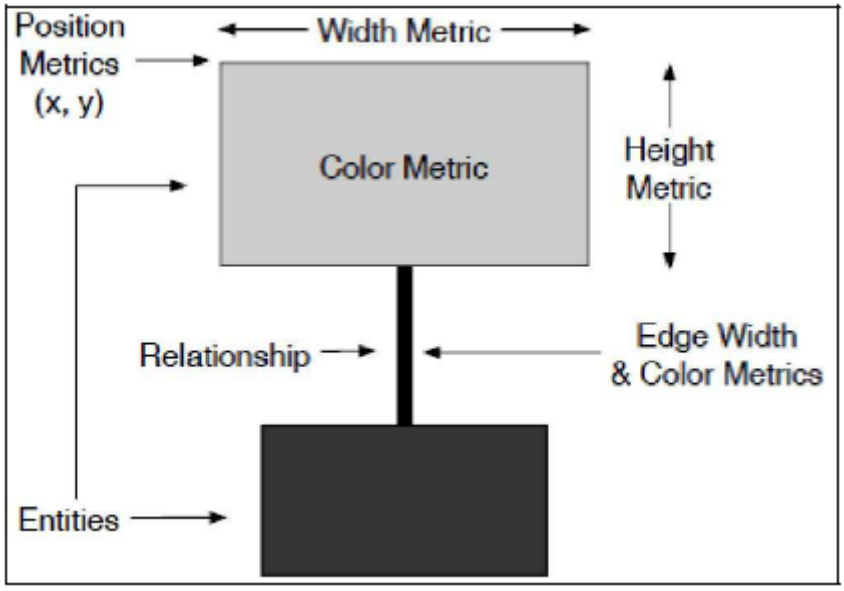


Εικόνα 14: *Πυραμίδα Οπτικοποίησης Μεγέθους και Πολυπλοκότητας – Size and Complexity Pyramid [28].*

Η βασική ιδέα πίσω από την πυραμίδα επισκόπησης είναι να συγκεντρωθούν σε ένα μέρος οι πιο σημαντικές μετρήσεις ενός αντικειμενοστρεφούς συστήματος, έτσι ώστε ένας μηχανικός να μπορεί να δει και να ερμηνεύσει ό,τι είναι απαραίτητο για να αποκτήσει μια πρώτη εικόνα του συστήματος.

3.6.4. Πολυμετρικές Προβολές – Polymetric Views

Μια Πολυμετρική Προβολή ή ακτίνες Χ είναι μια εμπλουτισμένη σε επίπεδο μετρήσεων απεικόνιση των Οντολογιών λογισμικού και των σχέσεών τους [7]. Το κύριο πλεονέκτημά τους είναι ότι μπορούν να αποδώσουν οπτικά τους αριθμούς με απλό, αλλά αποτελεσματικό και εξαιρετικά συμπυκνωμένο τρόπο που είναι άμεσα ερμηνεύσιμος από τον μελετητή τους. Μια ελαφριά τεχνική μπορεί εύκολα να εφαρμοστεί σε οποιοδήποτε περιβάλλον ανάπτυξης και έχει ήδη υιοθετηθεί από ορισμένα ερευνητικά πρωτότυπα λόγω της απλότητάς της.

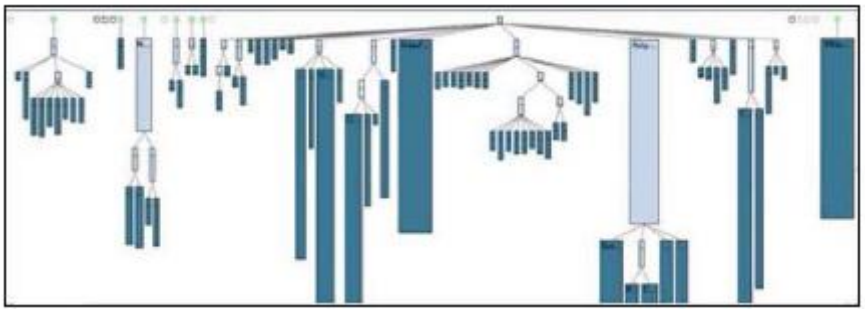


Εικόνα 15: *Κανόνες Πολυμετρικής Προβολής [28].*

Η Πολυμετρική Προβολή χρησιμοποιεί ορθογώνια για να εμφανίσει οντότητες λογισμικού ή αφαιρέσεις τους και χρησιμοποιεί ακμές για να αναπαραστήσει τις σχέσεις μεταξύ Οντολογιών. Αυτή είναι μια πρακτική που χρησιμοποιείται συνήθως στην οπτικοποίηση πληροφοριών και στα εργαλεία οπτικοποίησης λογισμικού.

Αυτή η βασική τεχνική οπτικοποίησης μπορεί να εμπλουτιστεί με την απόδοση τεσσάρων μετρικών μετρήσεων σε έναν μεμονωμένο κόμβο και δύο μετρήσεων σε ένα μόνο άκρο ταυτόχρονα, όπως βλέπουμε στην Εικόνα 15.

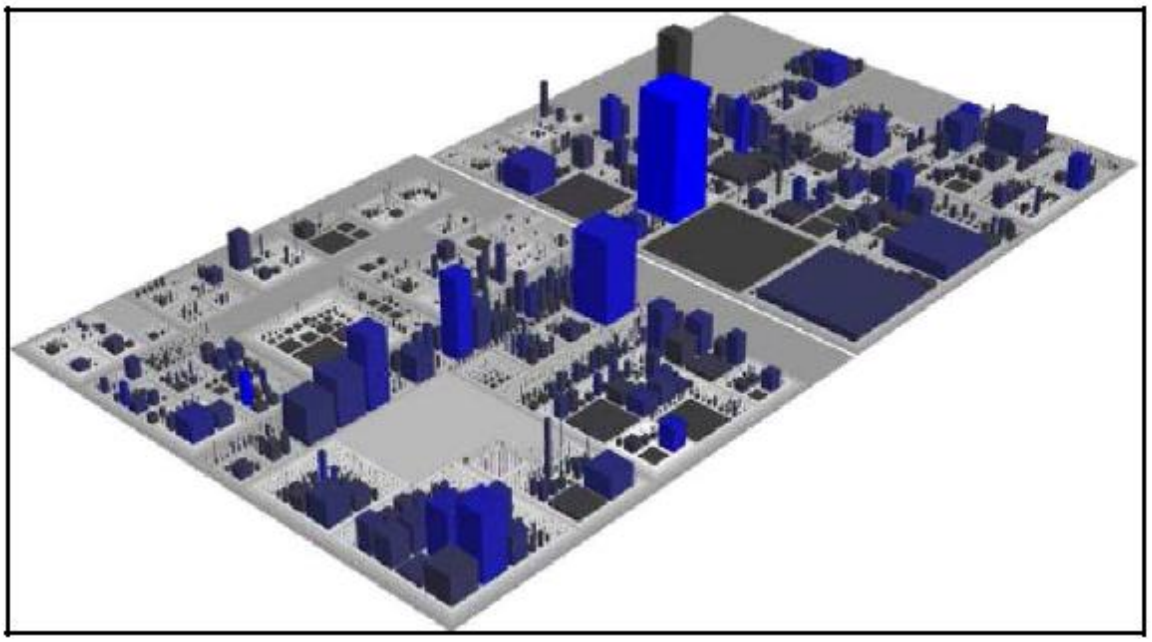
Το πιο δυνατό σημείο των Πολυμετρικών Προβολών είναι ότι μπορούν να συνδυάσουν πολλαπλές μετρήσεις και να παράγουν διαφορετικά χρωματιστά σχήματα που μπορούν να ερμηνευτούν από τον θεατή: Στη συνέχεια, ο θεατής πρέπει να κοιτάξει για ορισμένα οπτικά συμπτώματα (ανάλογα με την προβολή) και έτσι μπορεί οπτικά να εντοπίσει ενδιαφέροντα και δυσαρμονικά μέρη του συστήματος.



Εικόνα 16: *Αποτύπωση πολυπλοκότητας συστήματος.*

3.6.5. Το Εργαλείο CodeCity

Το CodeCity είναι ένα ολοκληρωμένο εργαλείο για ανάλυση λογισμικού, στο οποίο τα συστήματα λογισμικού οπτικοποιούνται ως διαδραστικές, πλοηγήσιμες τρισδιάστατες πόλεις [9]. Τα πακέτα κώδικα αντιπροσωπεύονται ως περιοχές, ενώ οι κλάσεις αντιπροσωπεύονται ως κτίρια στην πόλη. Τα ορατά χαρακτηριστικά των τεχνουργημάτων της πόλης απεικονίζουν ένα σύνολο επιλεγμένων μετρήσεων λογισμικού, όπως στις πολυμετρικές προβολές [7] του CodeCrawler [10].



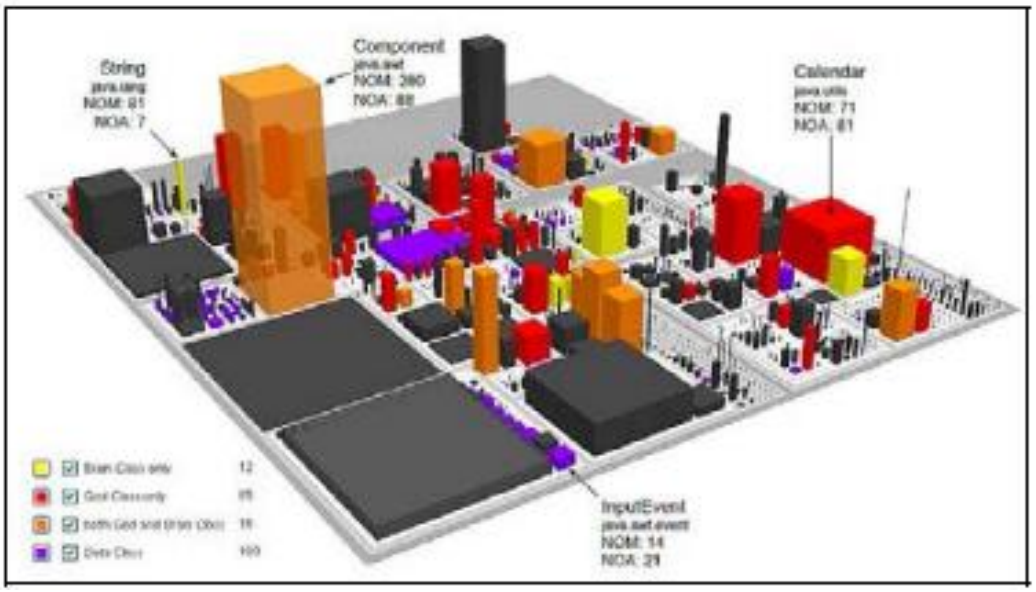
Εικόνα 17: *Code city παράδειγμα [28].*

Στην Εικόνα 17 παρουσιάζεται μια απεικόνιση ενός συστήματος που παράγεται από το CodeCity. Το CodeCity έχει αναπτύξει και διερευνήσει ορισμένες προσεγγίσεις για την κατανόηση συστημάτων λογισμικού, για την ανάλυση και την κατανόηση της εξέλιξής τους και την εμφάνιση προβλημάτων σχεδιασμού [9], [11]–[14].

Παρόμοια με την προσέγγιση της Πολυμετρικής προβολής, το CodeCity χαρτογραφεί ένα σύνολο μετρήσεων λογισμικού σχετικά με τις οπτικές ιδιότητες των διαγραμμάτων που αποτυπώνονται.

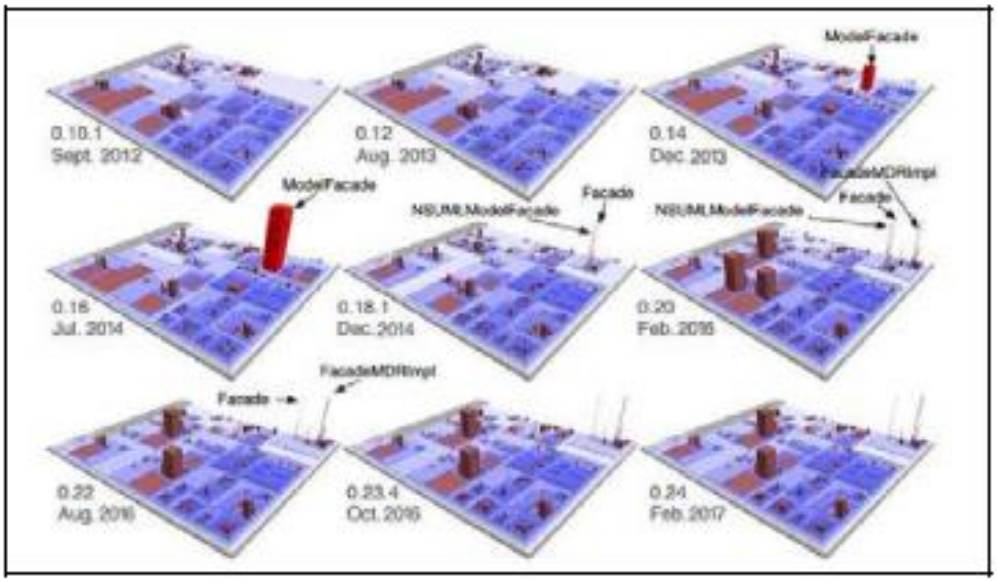
- Ο αριθμός των μεθόδων χαρτογραφείται ως το ύψος των κτιρίων.
- Ο αριθμός των ιδιοτήτων (attributes) ως τα τετραγωνικά της βάσης των κτιρίων.
- Ο αριθμός των γραμμών του κώδικα αποτυπώνεται με τον χρωματισμό των κτιρίων.

Το CodeCity χρησιμοποιεί χρώματα από σκούρο γκρι (μικρός αριθμός γραμμών κώδικα) έως έντονο μπλε (μεγάλος αριθμός γραμμών κώδικα) για να οπτικοποιήσει τις τιμές των καταχωρίσεων μετρήσεων. Μια νέα έκδοση του CodeCity παρέχει πιο έγχρωμες προβολές που ονομάζονται χάρτης δυσαρμονίας (Εικόνα 18).



Εικόνα 18: *Χάρτης δυσαρμονίας [28].*

Η οπτικοποίηση μας επιτρέπει να βλέπουμε εύκολα ορισμένα μοτίβα, όπως για παράδειγμα τα δύο ογκώδη κτίρια (οι κοινόχρηστες κλάσεις που είναι ένα κακό μοντέλο σχεδίασης), ορισμένες κατασκευές σε σχήμα κεραίας, ορισμένες κατηγορίες που μοιάζουν με χώρους στάθμευσης, καθώς και έναν μεγάλο αριθμό μικρά σπίτια. Η οπτικοποίηση είναι διαδραστική και πλοηγήσιμη χρησιμοποιώντας το πληκτρολόγιο. Είναι εύκολο οι χρήστες να μεγεθύνουν για να παρατηρήσουν λεπτομέρειες της πόλης ή να εστιάσουν σε μια συγκεκριμένη περιοχή. Ένα άλλο πλεονέκτημα είναι ότι χρησιμοποιώντας αυτήν την οπτικοποίηση, οι εκδόσεις του συστήματος μπορούν να συγκριθούν μεταξύ τους για να μάθουμε ακριβώς ποιο σημείο στον κώδικα αρχίζει να μεγεθύνεται και το σύστημα χάνει την απόδοσή του.



Εικόνα 19: *Εξέλιξη του συστήματος ανά τον χρόνο [28].*

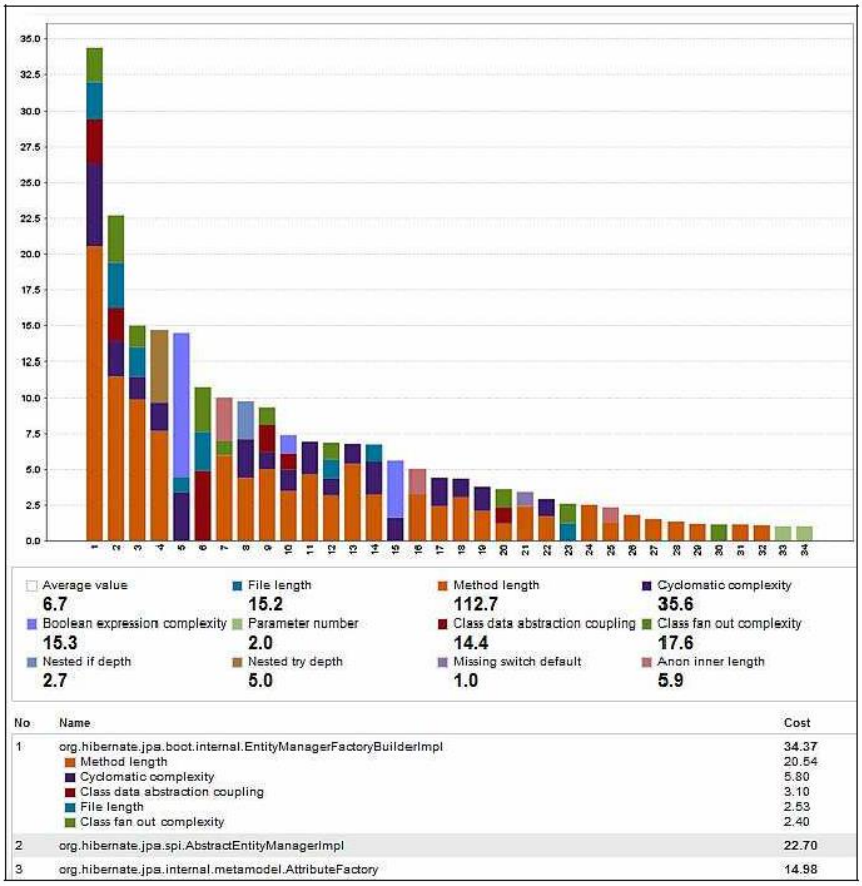
Η Εικόνα 19 εξηγεί πώς το σύστημα και στην περίπτωσή μας η οντολογία εξελίσσεται από καιρό σε καιρό. Επομένως, οποιοσδήποτε μηχανικός ή αρχιτέκτονας μπορεί με μία γρήγορη ματιά να αντιληφθεί και να χειριστεί πιθανά προβλήματα του συστήματος.

3.6.6. Διάγραμμα Τοξικότητας Κώδικα

Το διάγραμμα τοξικότητας είναι μια εξαιρετική τεχνική που βοηθάει τους διαχειριστές και τους μη προγραμματιστές να κατανοήσουν την εσωτερική ποιότητα του κώδικα [15].

Σε ένα διάγραμμα τοξικότητας, κάθε ράβδος αντικατοπτρίζει μια κατηγορία και το ύψος της ράβδου αντανακλά τη βαθμολογία τοξικότητας για αυτήν την κατηγορία. Η βαθμολογία βασίζεται σε πολλές μετρήσεις και όσο υψηλότερη είναι η βαθμολογία, τόσο πιο τοξική είναι η τάξη. Τα επιμέρους στοιχεία της κάθε κατηγορίας είναι χρωματιστά.

Αυτό καθιστά δυνατό να δούμε με μια ματιά όχι μόνο πόσο τοξικός είναι ένας κώδικας, αλλά και πώς κατανέμονται τα προβλήματα. Εάν υπάρχουν πολλά πορτοκαλί και μωβ στο γράφημα, αυτό σημαίνει μεγάλες και πολύπλοκες μεθόδους, γεγονός που δείχνει ότι ο κώδικας είναι πιθανώς δύσκολο να δοκιμαστεί ατομικά (unit testing).



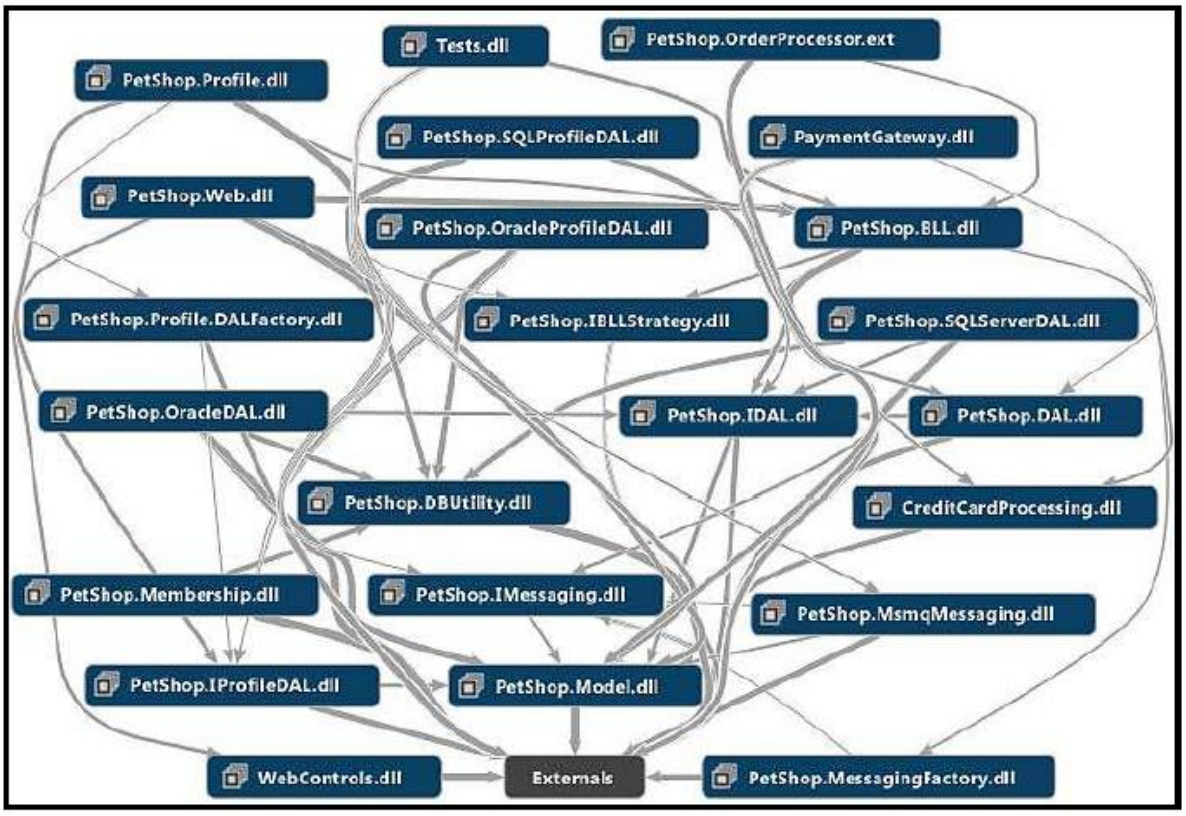
Εικόνα 20: *Διάγραμμα Τοξικότητας Κώδικα [28].*

Ο υπολογισμός της βαθμολογίας τοξικότητας βασίζεται σε ένα σύνολο από μετρικά και τα όρια για αυτά τα μετρικά.

3.6.7. Διάγραμμα Εξαρτήσεων

Όλα τα προαναφερθέντα εργαλεία εκτός από πολλά άλλα παραδοσιακά εργαλεία όπως (μέθοδοι Gantt, CPM, PERT και IDEF, π.χ. βλ. [16]) δεν αντιμετωπίζουν προβλήματα που προκύπτουν από την πολυπλοκότητα του συστήματος. Επιτρέπουν στους μηχανικούς λογισμικού να μοντελοποιούν διαδοχικά και παράλληλα στοιχεία, αλλά όχι αλληλοεξαρτώμενα στοιχεία, όπου ένα σύνολο στοιχείων εξαρτώνται το ένα από το άλλο. Το Dependency and Structure Modeling (DSM) (αναφέρεται επίσης ως διάγραμμα εξαρτήσεων) παρέχει αυτήν την ικανότητα αναπαράστασης με ευθύ και κομψό τρόπο.

Το DSM προσφέρει περιβάλλον μοντελοποίησης δικτύου που αντιπροσωπεύει τα στοιχεία ενός συστήματος και τις αλληλεπιδράσεις τους, τονίζοντας έτσι την αρχιτεκτονική (ή τη σχεδιασμένη δομή του συστήματος). Οι μέθοδοι DSM (βασισμένες σε πίνακες και γραφήματα) έχουν αποδειχθεί πολύ πολύτιμες για την κατανόηση, το σχεδιασμό και τη βελτιστοποίηση πολύπλοκων αρχιτεκτονικών συστημάτων [17].

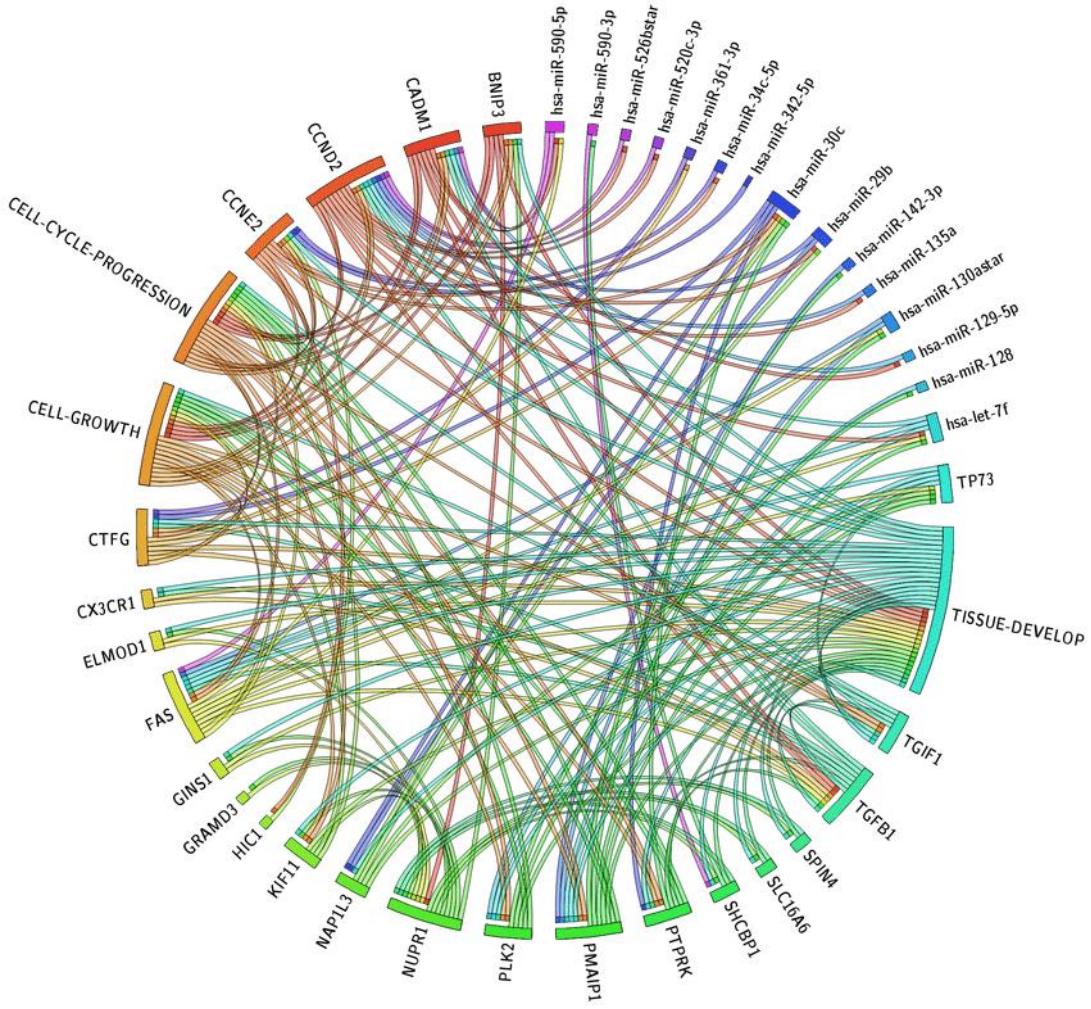


Εικόνα 21: *Κυκλικό γράφημα αποτύπωσης εξαρτήσεων [28].*

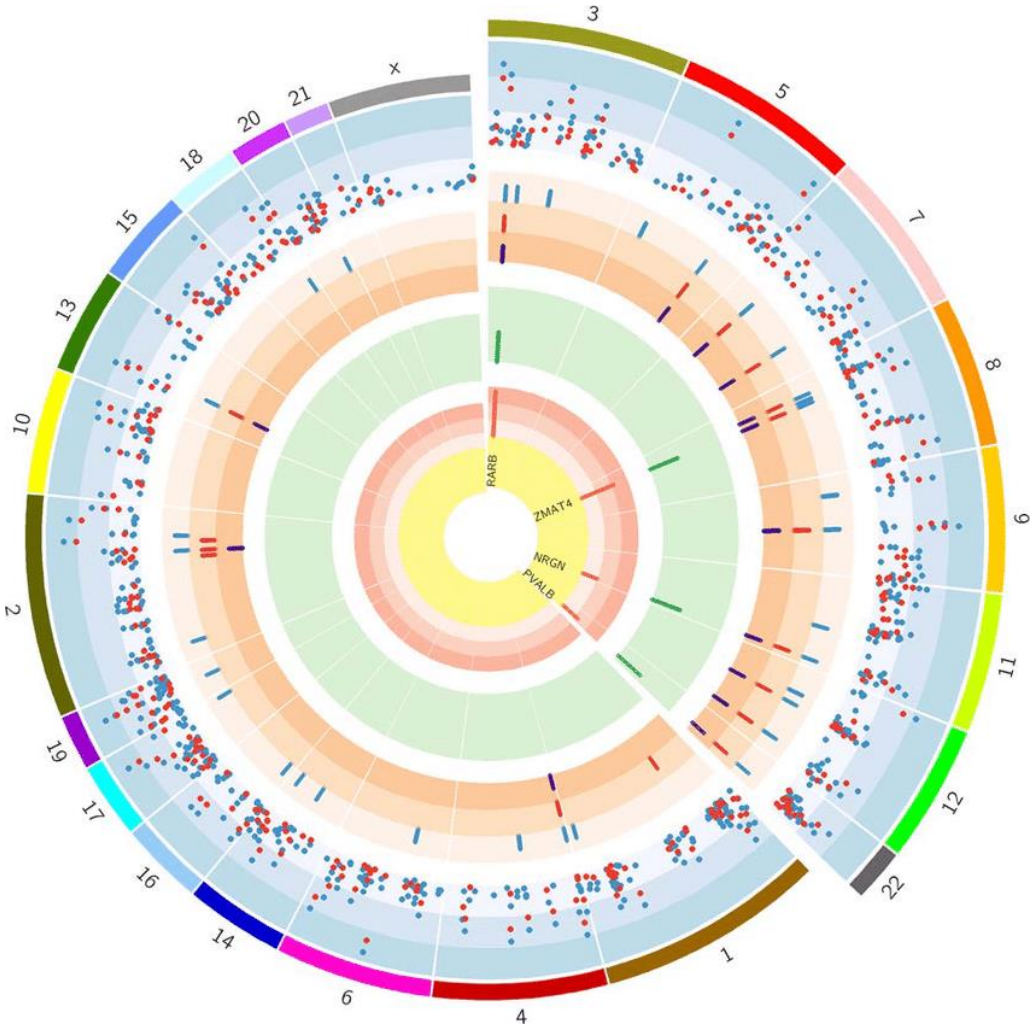
Μια άλλη ενδιαφέρουσα αναπαράσταση είναι η χρήση κυκλικών γραφημάτων. Τα κυκλικά γραφήματα οπτικοποιούν δεδομένα και πληροφορίες σε κυκλική διάταξη. Αυτό κάνει τα κυκλικά γραφήματα ιδανικά για την εξερεύνηση σχέσεων μεταξύ αντικειμένων ή θέσεων.

Παρόλο που το Circos σχεδιάστηκε αρχικά για την οπτικοποίηση γονιδιωματικών δεδομένων, χρησιμοποιείται για τη δημιουργία αριθμών από δεδομένα σε οποιονδήποτε άλλο τομέα. Ένα παράδειγμα κυκλικών γραφημάτων παράγονται από το Circos είναι οι εικόνες 23 και 24. Να

σημειωθεί πως το Circos μπορεί να χρησιμοποιηθεί με τον ίδιο τρόπο για να αποτυπώσει τις εξαρτήσεις των συστημάτων.



Εικόνα 22: *Κυκλικό γράφημα αποτύπωσης μέσω του Circos [29].*



Εικόνα 23: *Κυκλικό γράφημα αποτύπωσης μέσω του Circos [30].*

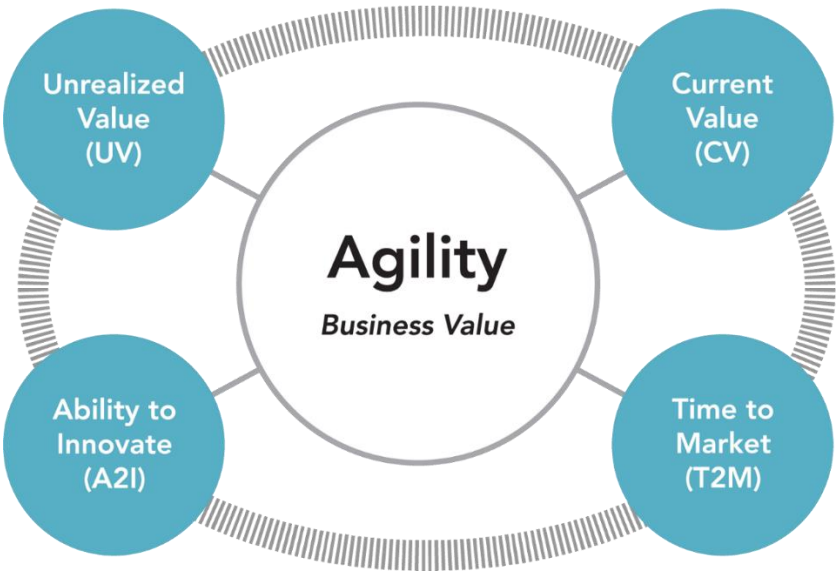
4. Διαχείριση Βάση Αποδείξεων (ΔΒΑ)

Στον σύγχρονο και γρήγορα μεταβαλλόμενο επιχειρηματικό κόσμο, οι επιχειρήσεις και οι οργανισμοί πρέπει να βασίζονται στην εμπειρική μάθηση πράγμα που τους βοηθάει να βελτιώνουν συνεχώς την εμπειρία των πελατών, τις οργανωτικές ικανότητες και τα επιχειρηματικά αποτελέσματα υπό συνθήκες αβεβαιότητας. Για αυτό το σκοπό έχει αναπτυχθεί η Διαχείριση Βάση Αποδείξεων (ΔΒΑ) (Empirical-Based Management (EMB)) [22] η οποία παρέχει ένα πλαίσιο για τους οργανισμούς με σκοπό της βελτίωσης της ικανότητά τους να προσφέρουν αξία σε έναν αβέβαιο κόσμο, αναζητώντας μια πορεία προς τους στρατηγικούς στόχους. Χρησιμοποιώντας σκόπιμα πειραματισμούς και αποδεικτικά στοιχεία (μετρικά), η ΔΒΑ επιτρέπει στους οργανισμούς να βελτιώνουν συστηματικά την απόδοσή τους με την πάροδο του χρόνου και να ορίζουν τους στόχους τους με βάση καλύτερες πληροφορίες.

Σκοπός αυτού του κεφαλαίου είναι να παρουσιάσει τις Περιοχές Βασικών Αξιών (ΠΒΑ) που προτείνονται από τη ΔΒΑ και να συγκρίνει τα μετρικά αυτών με αυτά που μελετήθηκαν και παρουσιάστηκαν στα προηγούμενο κεφάλαιο και πιο συγκεκριμένα στις ενότητες:

4.1. Περιοχές Βασικών Αξιών (ΠΒΑ)

Εκτός από τη χρήση υποθέσεων και πειραμάτων για την επίτευξη στόχων, η ΔΒΑ παρέχει ένα σύνολο προοπτικών αξίας και της ικανότητας του οργανισμού να προσφέρει αξία στους πελάτες του. Αυτές οι προοπτικές ονομάζονται Περιοχές Βασικών Αξιών (ΠΒΑ). Αυτοί οι τομείς εξετάζουν τους στόχους του οργανισμού (Μη Πραγματοποιημένη Αξία), την τρέχουσα κατάσταση του οργανισμού σε σχέση με αυτούς τους στόχους (Τρέχουσα Αξία), την ανταπόκριση του οργανισμού στην παροχή αξίας (Έγκαιρη Παροχή Αξίας) και την αποτελεσματικότητα του οργανισμού στην παροχή αξίας (Ικανότητα Καινοτομίας). Η εστίαση σε αυτές τις τέσσερις διαστάσεις δίνει τη δυνατότητα στους οργανισμούς να κατανοήσουν καλύτερα πού βρίσκονται και πού πρέπει να πάνε.



Εικόνα 24: *Περιοχές Βασικών Αξιών (ΠΒΑ) και οι στόχοι του οργανισμού [22].*

Κάθε ΠΒΑ εστιάζει σε μια διαφορετική πτυχή είτε της αξίας είτε της ικανότητας του οργανισμού να προσφέρει αξία. Η παροχή επιχειρηματικής αξίας (Τρέχουσα Αξία) είναι σημαντική, αλλά οι οργανισμοί πρέπει επίσης να δείξουν ότι μπορούν να ανταποκριθούν στην αλλαγή (Έγκαιρη Παροχή Αξίας) ενώ παράλληλα μπορούν να διατηρήσουν την καινοτομία με την πάροδο του χρόνου (Ικανότητα Καινοτομίας). Και πρέπει να είναι σε θέση να σημειώνουν συνεχώς πρόοδο προς τους μακροπρόθεσμους στόχους τους (Μη Πραγματοποιημένη Αξία) διαφορετικά κινδυνεύουν να υποκύψουν στη στασιμότητα και τον εφησυχασμό.

4.1.1. Τρέχουσα Αξία

Ο σκοπός της εξέτασης της Τρέχουσας Αξίας για τους οργανισμούς είναι να κατανοήσουν την αξία που προσφέρουν στους πελάτες και στα ενδιαφερόμενα μέρη αυτή τη χρονική στιγμή. Λαμβάνεται υπόψη μόνο αυτό που υπάρχει σήμερα και όχι την αξία που μπορεί να υπάρξει στο μέλλον. Τα ερωτήματα που οι οργανισμοί πρέπει να επαναξιολογούν συνεχώς για την τρέχουσα αξία είναι:

1. Πόσο ευχαριστημένοι είναι οι χρήστες και οι πελάτες σήμερα; Η ευτυχία τους βελτιώνεται ή μειώνεται;

2. Πόσο ευχαριστημένοι είναι οι υπάλληλοί του οργανισμού σήμερα; Η ευτυχία τους βελτιώνεται ή μειώνεται;

3. Πόσο ευχαριστημένοι είναι οι επενδυτές και οι άλλοι ενδιαφερόμενοι σήμερα; Η ευτυχία τους βελτιώνεται ή μειώνεται;

Η εξέταση της Τρέχουσας Αξίας βοηθά έναν οργανισμό να κατανοήσει την αξία που βιώνουν σήμερα οι πελάτες ή οι χρήστες του.

Τα μετρικά που αναδεικνύουν την Τρέχουσα Αξία είναι:

Πίνακας 3: *Μετρικά Τρέχουσας Αξίας.*

Τίτλος μετρικού	Περιγραφή μετρικού
Έσοδα ανά Εργαζόμενο	Η αναλογία (ακαθάριστα έσοδα / # εργαζομένων) είναι ένας βασικός ανταγωνιστικός δείκτης σε έναν κλάδο.
Αναλογία Κόστους Προϊόντος	Συνολικά έξοδα και κόστη για τα προϊόντα/συστήματα που μετρώνται, συμπεριλαμβανομένων των λειτουργικών δαπανών σε σύγκριση με τα έσοδα.
Ικανοποίηση εργαζομένων	Κάποια μορφή ανάλυσης συναισθήματος που βοηθά στη μέτρηση της αφοσίωσης, της ενέργειας και του ενθουσιασμού των εργαζομένων. Ικανοποίηση πελατών Κάποια μορφή ανάλυσης συναισθήματος που βοηθά στη μέτρηση της αφοσίωσης και της ευτυχίας των πελατών με το προϊόν.
Δείκτης χρήσης πελατών	Μέτρηση της χρήσης, ανά χαρακτηριστικό, για να συναχθεί ο βαθμός στον οποίο οι πελάτες βρίσκουν το προϊόν χρήσιμο και εάν η πραγματική χρήση ανταποκρίνεται στις προσδοκίες για το πόσο χρόνο θα πρέπει να κάνουν οι χρήστες με μια λειτουργία.

4.1.2. Μη Πραγματοποιημένη Αξία

Η Μη Πραγματοποιημένη Αξία εκφράζει την πιθανή μελλοντική αξία που θα μπορούσε να πραγματοποιηθεί εάν ο οργανισμός ικανοποιούσε τις ανάγκες όλων των πιθανών πελατών ή χρηστών. Η εξέταση της Μη Πραγματοποιημένης Αξίας βοηθά έναν οργανισμό να μεγιστοποιήσει την αξία που αντιλαμβάνεται από ένα προϊόν ή μια υπηρεσία με την πάροδο του χρόνου. Όταν οι πελάτες ή οι χρήστες βιώνουν ένα χάσμα μεταξύ της τρέχουσας εμπειρίας τους και της εμπειρίας που θα ήθελαν να έχουν, η διαφορά μεταξύ των δύο αντιπροσωπεύει μια ευκαιρία. Αυτή η ευκαιρία μετριέται με την Μη Πραγματοποιημένη Αξία. Τα ερωτήματα που οι οργανισμοί πρέπει να επαναξιολογούν συνεχώς για την Μη Πραγματοποιημένη Αξία είναι:

1. Μπορεί να δημιουργηθεί κάποια πρόσθετη αξία από τον οργανισμό μας σε αυτήν την αγορά ή σε άλλες αγορές;

2. Αξίζει τον κόπο και το ρίσκο για να επιδιώξουμε αυτές τις αναξιοποίητες ευκαιρίες;

3. Θα πρέπει να γίνουν περαιτέρω επενδύσεις για την απόκτηση πρόσθετης Μη Πραγματοποιημένης Αξίας;

Η εξέταση τόσο της Τρέχουσας όσο και της Μη Πραγματοποιημένης Αξίας παρέχει στους οργανισμούς έναν τρόπο να εξισορροπήσουν τα σημερινά και πιθανά μελλοντικά οφέλη. Οι στρατηγικοί στόχοι διαμορφώνονται από κάποιο κενό ικανοποίησης και μια ευκαιρία για έναν οργανισμό να μειώσει την Μη Πραγματοποιημένη Αξία αυξάνοντας την Τρέχουσα Αξία.

Τα μετρικά που αναδεικνύουν την Μη Πραγματοποιημένη Αξία είναι:

Πίνακας 4: *Μετρικά Μη Πραγματοποιημένης Αξίας.*

Τίτλος μετρικού	Περιγραφή μετρικού
Μερίδιο αγοράς προϊόντος	Το σχετικό ποσοστό της αγοράς που δεν ελέγχεται από το προϊόν. το δυνητικό μερίδιο αγοράς που θα μπορούσε να επιτύχει το προϊόν εάν κάλυπτε καλύτερα τις ανάγκες των πελατών.
Κενό ικανοποίησης πελατών ή χρηστών	Η διαφορά μεταξύ της επιθυμητής εμπειρίας ενός πελάτη ή χρήστη και της τρέχουσας εμπειρίας του.
Επιθυμητή εμπειρία πελάτη ή ικανοποίηση	Ένα μέτρο που υποδεικνύει την εμπειρία που θα ήθελε να έχει ο πελάτης.

4.1.3. Έγκαιρη Παροχή Αξίας

Η Έγκαιρη Παροχή Αξίας εκφράζει την ικανότητα του οργανισμού να παρέχει γρήγορα νέες δυνατότητες, υπηρεσίες ή προϊόντα. Ο λόγος για την εξέταση της Έγκαιρης Παροχής Αξίας είναι να ελαχιστοποιηθεί ο χρόνος που χρειάζεται ο οργανισμός για να προσφέρει αξία. Χωρίς την ενεργή διαχείριση της Έγκαιρης Παροχής Αξίας, η δυνατότητα βιώσιμης απόδοσης αξίας στο μέλλον είναι άγνωστη. Τα ερωτήματα που οι οργανισμοί πρέπει να επαναξιολογούν συνεχώς για την Έγκαιρη Παροχής Αξίας είναι:

1. Πόσο γρήγορα μπορεί ο οργανισμός να μάθει από νέα πειράματα και πληροφορίες;

2. Πόσο γρήγορα μπορεί να προσαρμοστεί με βάση τις πληροφορίες;

3. Πόσο γρήγορα μπορεί να δοκιμάσει νέες ιδέες με πελάτες;

Η βελτίωση της Έγκαιρης Παροχής Αξίας βοηθά στη βελτίωση της συχνότητας με την οποία ένας οργανισμός μπορεί ενδεχομένως να αλλάξει το επίπεδο της Τρέχουσα Αξίας του.

Τα μετρικά που αναδεικνύουν την Έγκαιρη Παροχή Αξίας είναι:

Πίνακας 5: *Μετρικά Έγκαιρης Παροχής Αξίας.*

Τίτλος μετρικού	Περιγραφή μετρικού
Συχνότητα υλοποίησης και ενσωμάτωσης	Ο αριθμός των ενσωματωμένων και δοκιμασμένων εκδόσεων ανά χρονική περίοδο. Για μια ομάδα που παραδίδει συχνά ή συνεχώς νέες εκδόσεις προϊόντος, αυτό το μέτρο αντικαθίσταται από τα Μετρικά Παράδοσης που αναφέρονται σε προηγούμενο κεφάλαιο.
Συχνότητα κυκλοφορίας	Ο αριθμός των κυκλοφοριών ανά χρονική περίοδο, π.χ. συνεχώς, καθημερινά, εβδομαδιαία, μηνιαία, τριμηνιαία, κ.λπ. Αυτό βοηθά να αντικατοπτρίζεται ο χρόνος που απαιτείται για την ικανοποίηση του πελάτη με νέα και ανταγωνιστικά προϊόντα.
Περίοδος σταθεροποίησης νέων εκδόσεων	Ο χρόνος που δαπανάται για τη διόρθωση προβλημάτων προϊόντος μεταξύ του σημείου που οι προγραμματιστές λένε ότι είναι έτοιμο να κυκλοφορήσει και του σημείου στο οποίο κυκλοφορεί πραγματικά στους πελάτες. Αυτό βοηθά στην αναπαράσταση του αντίκτυπου των ανεπαρκών

	πρακτικών ανάπτυξης και του υποκείμενου σχεδιασμού και της βάσης κώδικα.
Μέσος χρόνος επιδιόρθωσης	Ο μέσος χρόνος που απαιτείται από τη στιγμή που ανιχνεύεται ένα σφάλμα και όταν αυτό διορθώνεται. Αυτό βοηθά στην αποκάλυψη της αποτελεσματικότητας ενός οργανισμού να διορθώσει ένα σφάλμα.
Χρόνος Κύκλου Πελατών	Το χρονικό διάστημα από τη στιγμή που ξεκινά η εργασία σε μια έκδοση μέχρι το σημείο όπου κυκλοφορεί πραγματικά. Αυτό το μέτρο βοηθά στην αντανάκλαση της ικανότητας ενός οργανισμού να προσεγγίσει τον πελάτη του.
Χρόνος παράδοσης	Ο χρόνος από τη στιγμή που προτείνεται μια ιδέα ή σχηματίζεται μια υπόθεση έως ότου ένας πελάτης μπορεί να επωφεληθεί από αυτήν την ιδέα. Αυτό το μέτρο μπορεί να διαφέρει ανάλογα με τον πελάτη και το προϊόν. Είναι ένας παράγοντας που συμβάλλει στην ικανοποίηση των πελατών.
Χρόνος παράδοσης αλλαγών	Ο χρόνος που απαιτείται από την παράδοση του κώδικα από την ομάδα ανάπτυξης έως της εκτέλεσης αυτού με επιτυχία στην παραγωγή.
Συχνότητα ανάπτυξης	Ο αριθμός των φορών που ο οργανισμός ανέπτυξε (κυκλοφόρησε) μια νέα έκδοση του προϊόντος σε πελάτες/χρήστες.
Χρόνος επαναφοράς της υπηρεσίας	Το χρονικό διάστημα μεταξύ της έναρξης μιας διακοπής της υπηρεσίας και της αποκατάστασης της πλήρους διαθεσιμότητας της υπηρεσίας.
Χρόνος για μάθηση	Ο συνολικός χρόνος που απαιτείται για να σκιαγραφηθεί μια ιδέα ή μια βελτίωση, η υλοποίηση αυτής, η παράδοσή της στους τελικούς χρήστες και η μάθηση από τη χρήση τους.
Χρόνος εξάλειψης εξαρτήσεων και εμποδίων	Ο μέσος χρόνος από τη στιγμή που εγείρεται μία εξάρτηση ή ένα εμπόδιο μέχρι την επίλυσή τους. Είναι ένας παράγοντας που συμβάλλει στον χρόνο παράδοσης και στην ικανοποίηση των εργαζομένων.
Time to Pivot	Ένα μέτρο της πραγματικής επιχειρηματικής ευελιξίας που παρουσιάζει το χρόνο που μεσολάβησε από το πότε ένας οργανισμός λαμβάνει ανατροφοδότηση ή νέες πληροφορίες και όταν ανταποκρίνεται σε αυτά τα σχόλια. Για παράδειγμα, το χρονικό διάστημα που μεσολαβεί από τη στιγμή που ανακαλύπτει ότι ένας ανταγωνιστής έχει παραδώσει ένα νέο χαρακτηριστικό που κερδίζει την αγορά έως τη στιγμή που ο οργανισμός ανταποκρίνεται με αντιστοίχιση ή υπέρβαση νέων δυνατοτήτων που βελτιώνει την εμπειρία του πελάτη.

4.1.4. Ικανότητα Καινοτομίας

Η Ικανότητα Καινοτομίας περιγράφει την αποτελεσματικότητα ενός οργανισμού να παρέχει νέες δυνατότητες που θα μπορούσαν να ανταποκρίνονται καλύτερα στις ανάγκες των πελατών. Ο στόχος

της Ικανότητας Καινοτομίας είναι η μεγιστοποίηση της ικανότητας του οργανισμού να προσφέρει νέες δυνατότητες και καινοτόμες λύσεις. Οι οργανισμοί θα πρέπει να επαναξιολογούν συνεχώς την Ικανότητα Καινοτομίας τους ρωτώντας:

1. Τι εμποδίζει τον οργανισμό να προσφέρει νέα αξία;

2. Τι εμποδίζει τους πελάτες ή τους χρήστες να επωφεληθούν από αυτή την καινοτομία;

Η βελτίωση της Ικανότητας Καινοτομίας βοηθά έναν οργανισμό να γίνει πιο αποτελεσματικός διασφαλίζοντας ότι η εργασία που κάνει βελτιώνει την αξία που προσφέρουν τα προϊόντα ή οι υπηρεσίες του στους πελάτες ή τους χρήστες.

Τα μετρικά που αναδεικνύουν την Ικανότητα Καινοτομίας είναι:

Πίνακας 6: *Μετρικά Ικανότητας Καινοτομίας.*

Τίτλος μετρικού	Περιγραφή μετρικού
Ποσοστό Καινοτομίας	Το ποσοστό της προσπάθειας ή του κόστους που δαπανάται για νέες δυνατότητες προϊόντων, προς τη συνολική προσπάθεια ή κόστος προϊόντος. Αυτό παρέχει μια εικόνα για την ικανότητα του οργανισμού να παρέχει νέες δυνατότητες προϊόντων.
Τάσεις σφαλμάτων	Μέτρηση της αλλαγής των σφαλμάτων από την προηγούμενη έκδοση προϊόντος. Σφάλμα είναι οτιδήποτε μειώνει την αξία του προϊόντος σε έναν πελάτη, χρήστη ή στον ίδιο τον οργανισμό.
Ευρετήριο εγκατεστημένων εκδόσεων	Ο αριθμός των εκδόσεων ενός προϊόντος που υποστηρίζονται αυτήν τη στιγμή. Αυτό αντανακλά την προσπάθεια που καταβάλλει ο οργανισμός για την υποστήριξη και τη συντήρηση παλαιότερων εκδόσεων λογισμικού.
Τεχνικό χρέος	Μια έννοια στον προγραμματισμό που αντικατοπτρίζει την πρόσθετη εργασία ανάπτυξης και δοκιμών που προκύπτουν όταν οι «γρήγορες και βρώμικες» λύσεις χρησιμοποιούνται. Οι «γρήγορες και βρώμικες» λύσεις δημιουργούν ανεπιθύμητο αντίκτυπο στην παράδοση αξίας και η αύξηση τους μπορεί να αποφευχθεί μειώνοντας το τεχνικό χρέος.
Καταμέτρηση σφαλμάτων παραγωγής	Ο αριθμός των φορών σε μια δεδομένη περίοδο που η ομάδα ανάπτυξης διακόπηκε για να επιδιορθώσει ένα σφάλμα σε ένα εγκατεστημένο προϊόν. Ο αριθμός και η συχνότητα των σφαλμάτων παραγωγής μπορεί να βοηθήσει στην ένδειξη της σταθερότητας του προϊόντος.
Αριθμός Ενεργών Προϊόντων-Κωδικών Προϊόντος	Ο αριθμός των διαφορετικών εκδόσεων (ή παραλλαγών) ενός προϊόντος ή μιας υπηρεσίας. Παρέχει μια εικόνα για τον πιθανό αντίκτυπο της αλλαγής και την πολυπλοκότητα της εργασίας.
Χρόνος που δαπανάται για συγχώνευση κώδικα	Ο χρόνος που δαπανάται για την εφαρμογή αλλαγών σε διαφορετικές εκδόσεις ενός προϊόντος ή μιας υπηρεσίας.

	Παρέχει μια εικόνα για τον πιθανό αντίκτυπο της αλλαγής και την πολυπλοκότητα της εργασίας.
Χρόνος που δαπανάται για εναλλαγή τύπου εργασίας	Παραδείγματα περιλαμβάνουν χρόνο που χάνεται σε διακοπές που προκαλούνται από συσκέψεις ή κλήσεις, χρόνος που αφιερώνεται στην εναλλαγή μεταξύ εργασιών και χρόνος που χάνεται όταν τα μέλη της ομάδας διακόπτονται για να βοηθήσουν άτομα εκτός ομάδας να δώσουν απλή εικόνα για το μέγεθος του προβλήματος.
Αλλαγή ποσοστού αποτυχίας	Το ποσοστό των αλλαγών προϊόντων που κυκλοφόρησαν που οδηγούν σε υποβαθμισμένη υπηρεσία και απαιτούν αποκατάσταση (π.χ. επείγουσα επιδιόρθωση, επαναφορά, ενημέρωση κώδικα).

4.2. Σύγκριση Μετρικών της Διαχείρισης Βάση Αποδείξεων (ΔΒΑ) και των Προτεινόμενων Μετρικών της Agile Μεθοδολογίας

Όπως αναφέρθηκε στις προηγούμενες ενότητες, η ΔΒΑ έχει διακρίνει τις Περιοχές Βασικών Αξιών (ΠΒΑ) που βοηθούν ένα οργανισμό να διακρίνει εάν το προϊόν που χρησιμοποιούν οι πελάτες του παρέχει αξία την συγκεκριμένη χρονική στιγμή, να μετρά την αξία που πρόκειται να παρέχει στους πελάτες του, πόσο ευέλικτος είναι ο οργανισμός, και κατά πόσο μπορεί να καινοτομεί. Σε αυτό το κεφάλαιο θα μελετήσουμε και θα προτείνουμε επιπλέον μετρικά που μπορούν να ληφθούν υπόψιν σε κάθε μία από τις ΠΒΑ από τα Μετρικά της Agile Μεθοδολογίας που έχουν αναλυθεί σε προηγούμενο κεφάλαιο της διπλωματικής εργασίας.

Πιο συγκεκριμένα, εστιάζοντας στις βασικές ΠΒΑ τα εξής μετρικά μπορούν να υποστηρίξουν την κατηγοριοποίηση και την μελέτη του προϊόντος και της προστιθέμενης αξίας που παρέχει στον οργανισμό:

- **Τρέχουσα Αξία:** 3.2.2 Έλεγχος υγείας των ομάδων ανάπτυξης & 3.5.1 Παραδοθείσα Επιχειρηματική Αξία.
- **Μη Πραγματοποιημένη Αξία:** 3.5.1 Παραδοθείσα Επιχειρηματική Αξία & 3.4.3 Τυπική παράβαση και πιο συγκεκριμένα
  - ✓ Ο αριθμός μεγάλων εργασιών που βρίσκονται στο Product Backlog και
  - ✓ Ο αριθμός των εργασιών που εμφανίζονται σε πολλά Product Backlogs διαφορετικών ομάδων ανάπτυξης.
- **Έγκαιρη Παροχή Αξίας:** Εδώ μπορούμε να βασιστούμε σε πολλά μετρικά από αυτά έχουν αναφερθεί στα κεφάλαια που είναι σχετικά με την Παράδοση (3.3 Μετρικά Παράδοσης), την Συνεργατικότητα (3.2 Μετρικά Συνεργατικότητας).
- **Ικανότητα Καινοτομίας:** Εδώ μπορούμε να βασιστούμε σε πολλά μετρικά από αυτά έχουν αναφερθεί και είναι σχετικά με τα 3.4 Μετρικά Ποιότητας, τα 3.6 Μετρικά για Αρχιτεκτονικές Λογισμικών και Οντολογιών και τα 3.5 Μετρικά Προστιθέμενης Αξίας Προϊόντος.

5. Μελέτη Περίπτωσης 1 – Agile metrics

5.1. Περιγραφή του Προβλήματος

Σε μια εταιρία ανάπτυξης λογισμικού όπου όλος ο τομέας πληροφορικής λειτουργεί υπό Ευέλικτες πρακτικές αποφασίσαμε να εφαρμόσουμε και να οπτικοποιήσουμε από κοινού για όλες τις Ομάδες ανάπτυξης λογισμικού τα Ευέλικτα μετρικά. Η εταιρία αυτή αποτελείται από 38 αυτοτελείς ομάδες ανάπτυξης λογισμικού όπου αναπτύσσουν και παραδίδουν προς χρήση στους τελικούς χρήστες κομμάτια από το ίδιο τελικό προϊόν. Για να επιτευχθεί αυτό οι προκλήσεις που έπρεπε να αντιμετωπιστούν ήταν οι εξής:

- Ο τρόπος με τον οποίο θα γίνεται η συλλογή δεδομένων θα έπρεπε να ακολουθεί μια συνέπεια στον τρόπο καταγραφής αλλά και στον τρόπο συλλογής. Πράγμα που απαιτεί συγχρονισμό ανάμεσα σε αυτές τις 38 ομάδες ανάπτυξης λογισμικού.
- Η επιλογή των πιο απαραίτητων μετρικών για τον οργανισμό και η προτεραιοποίηση αυτών.
- Ο ορισμός ενός του ελάχιστου βιώσιμου προϊόντος (Minimum Viable Product MVP) και ο προγραμματισμός των επόμενων εκδόσεων αυτού.
- Θα έπρεπε να βρεθεί ένας αυτοματοποιημένος συλλογής των τελικών δεδομένων.
- Η οπτικοποιήση των δεδομένων θα έπρεπε να είναι εύκολη στην κατανόηση και στην ανάγνωση.
- Θα έπρεπε να δοθούν επιλογές για φιλτράρισμα των δεδομένων ακολουθώντας τους κανόνες ευχρηστίας για την εμπειρία των τελικών χρηστών (UX).

5.2. Συλλογή των δεδομένων

Για την ανάπτυξη λογισμικών σήμερα είναι απαραίτητη η χρήση Συστημάτων Καταγραφής Εργασιών τα οποία έχουν προσαρμοστεί στα Ευέλικτα Πλαίσια Ανάπτυξης Λογισμικού (π.χ JIRA, Azure DevOps, Youtrack, κλπ.). Πιο συγκεκριμένα, αυτά τα συστήματα, βοηθούν τις Ευέλικτες ομάδες ανάπτυξης Λογισμικού να καταγράφουν για κάθε εργασία τα Story Points που απαιτούνται για την υλοποίηση του κάθε User Story και παρέχουν λειτουργικότητες αναφορών για κάποια από τα Agile μετρικά. Όμως αυτές οι αναφορές δεν είναι παραμετροποιήσιμες και δεν βοηθούν πάντα τους οργανισμούς να βγάλουν κάποια συμπεράσματα απαραίτητα για την εξέλιξή τους. Για αυτό το λόγο, τα Συστήματα Καταγραφής Εργασιών υποστηρίζουν επιπλέον την αποστολή δεδομένων μέσω REST APIs[7][8] πράγμα που συμβάλει στην επεξεργασία αυτών και συνεπώς στην τελική οπτικοποίηση αυτών ανάλογα με τις ανάγκες του οργανισμού.

Απαραίτητη προϋπόθεση αποτελεί η ομοιομορφία στον τρόπο όπου οι ομάδες οργανώνουν το backlog τους, κάνουν αναφορά των παραγωγικών και μη δυσλειτουργιών, του effort για την υλοποίηση μιας νέας λειτουργικότητας κλπ. Για την επίτευξη της ομοιομορφίας απαιτείται η ευθυγραμμισμένη επικοινωνία και ο έλεγχος των δεδομένων τα οποία έχουν καταγραφεί στα Συστήματα Καταγραφής Εργασιών. Για την επίτευξη αυτού του στόχου, μπορούμε να ορίσουμε μια οριζόντια ομάδα η οποία συνεργάζεται για την οριστικοποίηση του κοινού τρόπου λειτουργίας του οργανισμού και του τελικού ελέγχου των δεδομένων που καταγράφονται από τις ομάδες. Επιπλέον,

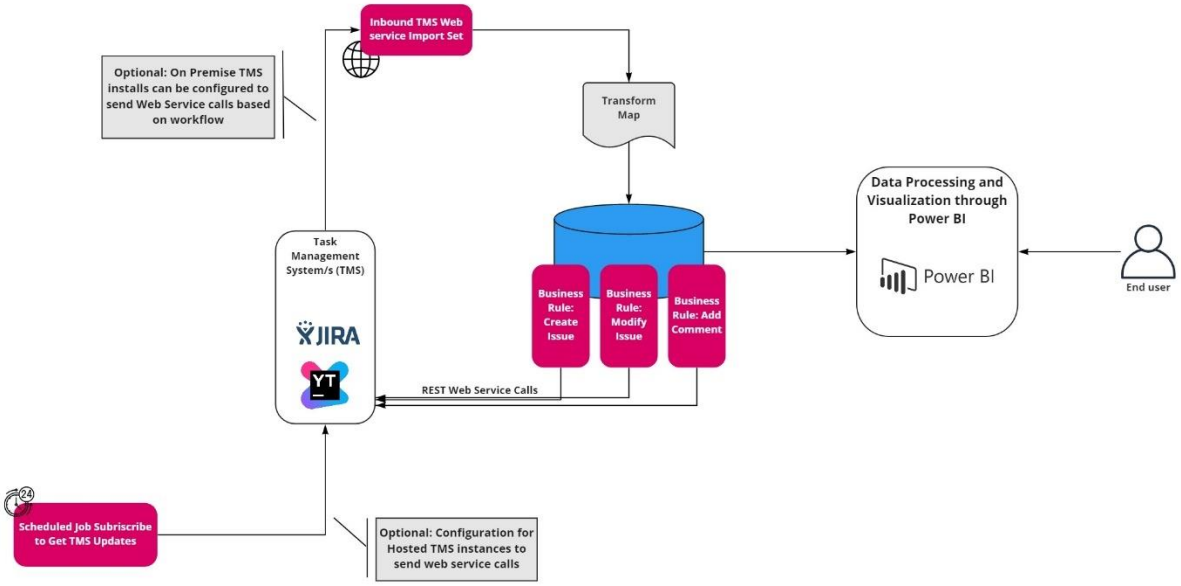
[7] Jira REST API documentation: https://developer.atlassian.com/server/jira/platform/rest-apis/
[8] YouTrack REST API documentation: https://www.jetbrains.com/help/youtrack/devportal/youtrack-rest-api.html#general-notes

τα Ευέλικτα Πλαίσια Ανάπτυξης Λογισμικού υποστηρίζουν την συνεχή επικοινωνία και την ευθυγράμμιση των ομάδων ανάπτυξης λογισμικού με διάφορους τρόπους (Scrum of Scrums) όσων αφορά τα τελικά παραδοτέα. Τέτοιες τεχνικές μπορούν να χρησιμοποιηθούν και για την οριζόντια επίτευξη της ομοιομορφίας των δεδομένων.

Αφού λοιπόν έχει επιτευχθεί αυτός ο στόχος, ο επόμενος είναι η αυτόματη συλλογή και αποτύπωση των δεδομένων, το οποίο αναλύεται στο επόμενο κεφάλαιο.

5.3. Αυτοματοποίηση συλλογής Δεδομένων από Συστήματα Καταγραφής Εργασιών

Τα Συστήματα Καταγραφής Εργασιών υποστηρίζουν την σύνδεση με Συστήματα Οπτικοποίησης Δεδομένων όπως για παράδειγμα το Power BI[9], της Microsoft. Όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο, τα Συστήματα Καταγραφής Εργασιών υποστηρίζουν την αποστολή δεδομένων μέσω REST APIs. Μετά την αποστολή αυτών των δεδομένων προς μια βάση, υπάρχει η δυνατότητα της οπτικοποίησης αυτών μέσω των Συστημάτων Οπτικοποίησης Δεδομένων. Αυτά τα συστήματα, Οπτικοποίησης, δίνουν την δυνατότητα στους χρήστες τους να φιλτράρουν ώστε να αποτυπώνονται μέσω ενός UI (User Interface) τα επιθυμητά δεδομένα σε διαγράμματα. ζω



Εικόνα 25: *Αρχιτεκτονική αυτόματης συλλογής δεδομένων από συστήματα καταγραφής εργασιών.*

Η αρχιτεκτονική του μοντέλου αυτού αποτυπώνεται στην Εικόνα 26. Ουσιαστικά, οι κανόνες που χρησιμοποιούνται από τα REST APIs των Συστημάτων Καταγραφής Εργασιών αποθηκεύονται στην Βάση Δεδομένων και πυροδοτούνται μέσω ενός προγραμματιστή (scheduler) είτε μετά από κάποιο προκαθορισμένο χρόνο (κάθε 24 ώρες για παράδειγμα) ή κάποιες φορές μέσα στην ημέρα. Εν συνεχεία, τα δεδομένα που συλλέγονται από τα Συστήματα Καταγραφής Εργασιών επεξεργάζονται και αποθηκεύονται στην Βάση Δεδομένων και τροφοδοτούνται στο Σύστημα Οπτικοποίησης Δεδομένων. Μέσω του Συστήματος Οπτικοποίησης Δεδομένων οι χρήστες μπορούν να αλληλοεπιδράσουν και να ενημερώνονται για τα μετρικά για τα οποία ενδιαφέρονται. Να σημειωθεί

[9] https://powerbi.microsoft.com/en-us/

πως τα συστήματα αυτά μπορούν να προσαρμοστούν στις ανάγκες των χρηστών (π.χ. φιλτράρισμα δεδομένων, επιλογή διαγράμματος οπτικοποίησης, κλπ.). Έτσι λοιπόν μπορεί να επιτευχθεί η δυναμική καταγραφή, τροφοδότηση και οπτικοποίηση των Ευέλικτων Μετρικών των πμάδων ανάπτυξης λογισμικού.

Στο επόμενο κεφάλαιο θα παρουσιάσουμε μια σειρά από προτάσεις για dashboards που μπορούν να χρησιμοποιηθούν για την αποτύπωση μετρικών όπως αυτά παρουσιάστηκαν στην διπλωματική εργασία. Να σημειωθεί πως κάποια από αυτές τις προτάσεις υλοποιήθηκαν αλλά δεν υπάρχει η δυνατότητα αποτύπωσης αυτών στην εργασία αλλά επίσης κάποια από αυτά αποτελούν προτάσεις βάση των μετρικών που προτείνονται παραπάνω έχοντας σαν οδηγό τις σωστές πρακτικές για την εμπειρία χρηστών (UX).

5.4. Παραδείγματα Διεπαφών Χρήστη από Συστήματα Οπτικοποίησης Δεδομένων

Σε αυτό το κεφάλαιο παρουσιάζουμε ένα σύνολο από παραδείγματα διεπαφών χρήστη (wireframes) που μπορούν να προκύψουν από την μέθοδο Αυτοματοποίησης Συλλογής Δεδομένων στα Συστήματα Οπτικοποίησης Δεδομένων.

5.4.1. Πίνακας Διαγραμμάτων Χρήσης των Ομάδων Ανάπτυξης Λογισμικού

Ο Πίνακας Διαγραμμάτων Χρήσης των ομάδων ανάπτυξης λογισμικού, αποτυπώνει τις βασικότερες πληροφορίες σχετικά με την απόδοση των ομάδων ανάπτυξης λογισμικού. Όπως φαίνεται και στην Εικόνα 27, το αριστερό κομμάτι του Πίνακα περιλαμβάνει ένα σύνολο από φίλτρα που επιτρέπουν στους χρήστες να αποτυπώσουν τα δεδομένα που επιθυμούν. Πιο συγκεκριμένα:

- **Year:** Το φίλτρο αυτό επιτρέπει στους χρήστες να επιλέξουν χρονιά για την οποία θέλουν να αποτυπωθούν δεδομένα στον Πίνακα.
- **Quarter:** Το φίλτρο αυτό επιτρέπει στους χρήστες να επιλέξουν τρίμηνο για το οποίο θέλουν να αποτυπωθούν δεδομένα στον Πίνακα.
- **Sprint – multiple selection:** Το φίλτρο αυτό επιτρέπει στους χρήστες να επιλέξουν Sprint ή και Sprints για τα οποία θέλουν να αποτυπωθούν δεδομένα στον Πίνακα. Να σημειωθεί πως αυτό το φίλτρο επιτρέπει την επιλογή περισσότερων από μια επιλογές.
- **Team:** Το φίλτρο αυτό επιτρέπει στους χρήστες να επιλέξουν την ομάδα ανάπτυξης λογισμικού για την οποία θέλουν να αποτυπωθούν δεδομένα στον Πίνακα.

Εικόνα 26: *Πίνακας Διαγραμμάτων Χρήσης των ομάδων ανάπτυξης λογισμικού*

Μετά την επιλογή των φίλτρων που επιθυμούν οι χρήστες, ο Πίνακας Διαγραμμάτων ανανεώνεται και αποτυπώνει τα δεδομένα όπως αναφέρονται παρακάτω:

1. Μπάρα στην κορυφή του Πίνακα Διαγραμμάτων Χρήσης των ομάδων ανάπτυξης λογισμικού, από αριστερά προς τα δεξιά:
    i. **Completed:** Το συνολικό (Total) και το μέσο όρο για τα έξη τελευταία Sprints (Median), των ολοκληρωμένων Story Points που παρέδωσε η ομάδα ανάπτυξης λογισμικού.
    ii. **Carry over:** Το συνολικό και το μέσο όρο για τα έξη τελευταία Sprints, των Story Points που η ομάδα ανάπτυξης λογισμικού δεν παρέδωσε αλλά μετέφερε στο αμέσως επόμενο Sprint.
    iii. **Committed:** Το συνολικό και το μέσο όρο για τα έξη τελευταία Sprints, των Story Points στα οποία δεσμεύτηκε η ομάδα ανάπτυξης λογισμικού προς παράδοση.
    iv. **Completed:** Το συνολικό και το μέσο όρο για τα έξη τελευταία Sprints, των Story Points τα οποία παρέδωσε η ομάδα ανάπτυξης λογισμικού.
    v. **Team Reliability %:** Αυτό το ποσοστό αποτελεί ο λόγος του μέσου όρου των Story Points που παραδόθηκαν προς τον μέσο όρο Story Points δέσμευσης της ομάδας ανάπτυξης λογισμικού για τα έξη τελευταία Sprints. Αυτός ο λόγος αντανακλά την αξιοπιστία της ομάδας.
2. Διαγράμματα στην κεντρική οθόνη:
    i. **Velocity Chart:** Το Velocity Chart παρουσιάζει τα Story Points που δεσμεύτηκε η ομάδα για να παραδώσει και ακριβώς δίπλα αυτά που παρέδωσε. Να σημειωθεί πως στο διάγραμμα αυτό παρουσιάζονται αυτά τα δεδομένα για τα έξι τελευταία Sprints. Για περισσότερες πληροφορίες σχετικά με τα μετρικά όπως και το διάγραμμα αυτό δείτε το κεφάλαιο 3.3.7 Velocity Rate/Chart & Velocity Deviation.
    ii. **Workload Categorisation:** Σε αυτό το διάγραμμα παρουσιάζεται το σύνολο των Story Points που παραδόθηκαν χωρισμένα σε κάποιες κατηγορίες. Οι κατηγορίες αυτές

μπορούν να τεθούν είτε από την ομάδα ανάπτυξης ή από την ίδια την εταιρία ανάλογα με τις ανάγκες. Εμείς έχουμε χρησιμοποιήσει για χάρη του παραδείγματος τις κατηγορίες Roadmap, Technical Depth και Ad-hoc.

iii. **Unplanned:** Το διάγραμμα αυτό παρουσιάζει ένα ιστόγραμμα που αποτυπώνει τα συνολικά Story Points που παραδόθηκαν από την ομάδα και δεν αποτελούσαν δουλεία που είχε προγραμματιστεί για αυτό το Sprint. Αυτό το διάγραμμα μπορεί να αποτελέσει ένας καλός δείκτης για να καταλάβουμε εάν η ομάδα ανάπτυξης έχει εξωτερικό θόρυβο ο οποίος την αποτρέπει από το να πετύχει τους στόχους του Sprint. Επίσης σε περιπτώσεις όπου ο Unplanned δείκτης είναι υψηλός, θα πρέπει η ομάδα ανάπτυξης να μελετήσει και την ποιότητα των παραδοτέων της μιας και έχει παρατηρηθεί ότι όταν η ποιότητα είναι χαμηλή, αυτός ο δείκτης τείνει να αυξάνεται.

iv. **Carry Over:** Το διάγραμμα αυτό παρουσιάζει ένα ιστόγραμμα που αποτυπώνει τα συνολικά Story Points που δεν παραδόθηκαν και μεταφέρθηκαν στο αμέσως επόμενο. Σε περιπτώσεις που ο δείκτης αυτός είναι υψηλός θα πρέπει να δούμε τα επίπεδα του Unplanned δείκτη και το Commitment της ομάδας για τα προηγούμενα Sprints. Όταν ο δείκτης Unplanned του ίδιου Sprint είναι υψηλός τότε αυτό επηρέασε και τον αυξημένο δείκτη Carry Over. Επίσης, εάν το Commitment της ομάδας έχει αυξηθεί απότομα από το ένα Sprint στο επόμενο και η δομή της ομάδας ανάπτυξης δεν έχει αλλάξει τότε θα πρέπει η ομάδα να δεσμευτεί σε λιγότερους πόντους (Story Points) για το επόμενο Sprint.

Με αυτό τον τρόπο ένας οργανισμός μπορεί να έχει μια συλλογική εικόνα για τις ομάδες ανάπτυξης Λογισμικού σε όλα τα επίπεδα, από την ίδια την ομάδα ανάπτυξης μέχρι τα στελέχη του οργανισμού. Αυτά τα μετρικά μπορούν να βοηθήσουν στην λήψη αποφάσεων, στην ανάδειξη κάποιας προβληματικής περιοχής, στη βελτίωση του τρόπου εργασίας κλπ. Επίσης μια σημαντική παρατήρηση είναι πως μέσω αυτών των μετρικών αλλά και μέσω του ορισμού νέων μετρικών μπορεί να επηρεαστεί η κουλτούρα όλου του οργανισμού και συνεπώς η συλλογική προσπάθεια για βελτίωση ή για εστίαση σε κάποιο σημείο του οργανισμού που χρειάζεται βελτίωση. Για παράδειγμα, μπορεί να έχει παρατηρηθεί πως σε έναν οργανισμό έχει μειωθεί η ποιότητα των τελικών παραδοτέων και έχουν αυξηθεί οι απρογραμμάτιστες εκδόσεις. Για την βελτίωση αυτών θα μπορούσαν να τεθούν νέα μετρικά τα οποία αποτυπώνουν τον αριθμό των σφαλμάτων που παρατηρήθηκαν στην παραγωγή κλπ. Με αυτό τον τρόπο επιτυγχάνεται η άμεση ενημέρωση και ευθυγράμμιση των ομάδων ανάπτυξης αλλά και όλου του οργανισμού και συνεπώς οι ομάδες θα μπορούν να εργάζονται επιπλέον για την άμεση επίλυση αυτών παρέχοντας ένα ποιοτικότερο τελικό προϊόν προς τους χρήστες του. Τέτοια μετρικά ποιότητας θα παρουσιάσουμε στο επόμενο κεφάλαιο.

5.4.2. Πίνακας Διαγραμμάτων Χρήσης Τεχνικών Σφαλμάτων

Ο Πίνακας Διαγραμμάτων Χρήσης Τεχνικών Σφαλμάτων (bugs) , αποτυπώνει τις βασικότερες πληροφορίες σχετικά με τεχνικά σφάλματα που παρουσιάζονται στα διαφορετικά περιβάλλοντα που χρησιμοποιούνται από τις ομάδες ανάπτυξης λογισμικού.

Στον Πίνακας Διαγραμμάτων Χρήσης Τεχνικών Σφαλμάτων όπως φαίνεται και στην Εικόνα 28 αποτυπώνονται τα tickets τα οποία δεν έχει ολοκληρωθεί η τεχνική τους ανάπτυξη. Έτσι λοιπόν στο

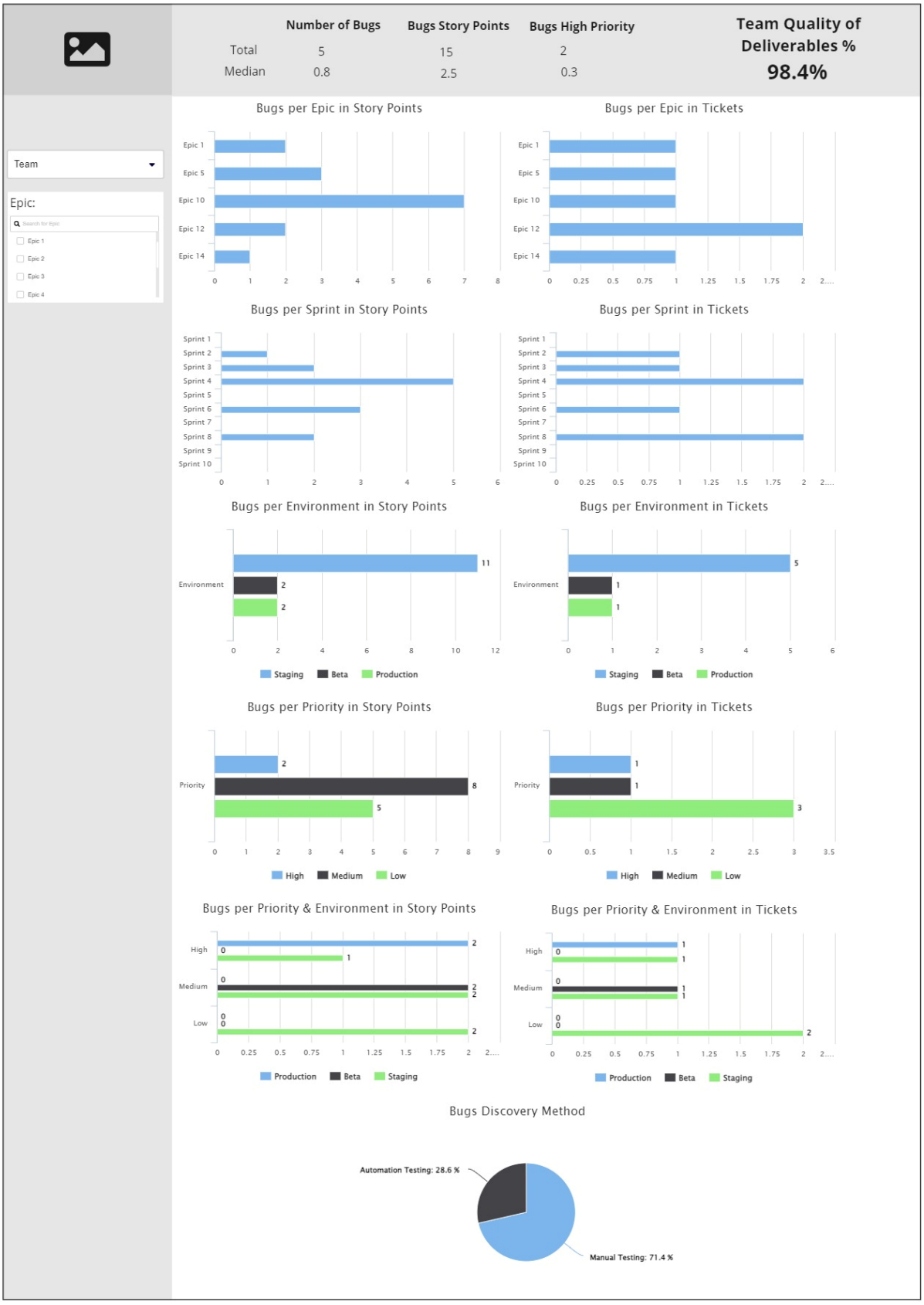
αριστερό κομμάτι του πίνακα υπάρχουν δύο φίλτρα ένα για να μπορέσουμε να ενημερωθούμε για τεχνικά σφάλματα ομάδων και ένα σχετικά με τα epics που μπορεί να ανήκουν αυτά.

Η κεντρική οθόνη, χωρίζεται σε δύο στήλες, αριστερά αποτυπώνονται οι πληροφορίες σε Story Points και δεξιά σε Tasks. Πιο συγκεκριμένα τα διαγράμματα που παρουσιάζονται σε αυτόν τον Πίνακα είναι τα παρακάτω:

1. Bugs per Epic: Σε αυτό το διάγραμμα παρουσιάζονται τα τεχνικά σφάλματα που βρέθηκαν κατά την διάρκεια των τεστ ανά epic. Σημείωση αυτό το διάγραμμα μπορεί να προσαρμοστεί και ανά έργο.
2. Bugs per Sprint: Σε αυτό το διάγραμμα παρουσιάζεται ο αριθμός των τεχνικών σφαλμάτων που αναγνωρίστηκαν σε κάθε Sprint.
3. Bugs per Environment: Σε αυτό το διάγραμμα παρουσιάζονται συλλογικά τα τεχνικά σφάλματα που έχουν αναγνωριστεί σε κάθε περιβάλλον που χρησιμοποιείται για την ανάπτυξη λογισμικού από τις ομάδες ανάπτυξης της εταιρίας.
   Να σημειωθεί πως τα περιβάλλοντα που χρησιμοποιήθηκαν για το δικό μας παράδειγμα είναι τα παρακάτω:
   - **Staging:** Το Staging περιβάλλον, δίνει την δυνατότητα στις ομάδες ανάπτυξης να μπορούν να δοκιμάσουν τον νέες υλοποιήσεις, το νέο design και τις δυνατότητες της ιστοσελίδας πριν οι νέες υλοποιήσεις περάσουν στο Beta περιβάλλον για να γίνουν τα τελικά τεστ και τέλος να γίνουν διαθέσιμες στους τελικούς πελάτες μέσω του Production.
   - **Beta:** Το Beta περιβάλλον είναι ένας κλώνος του Production και περιέχει επιπλέον τις νέες υλοποιήσεις της νέας εκδοχής του παραγωγικού περιβάλλοντος. Στο Beta περιβάλλον παίρνουν μέρος τα τελικά τεστ – αυτοματοποιημένα ή μη – όπως και τα τεστ αποδοχής χρηστών (User Acceptance Tests UAT) από τους πελάτες για την έγκριση του τελικού παραδοτέου.
   - **Production:** Παραγωγικό περιβάλλον – το περιβάλλον το οποίο χρησιμοποιούν οι τελικοί χρήστες και πελάτες.

   Κάποιες εταιρίες ανάπτυξης λογισμικού μπορεί να χρησιμοποιούν επιπλέον ή και λιγότερα περιβάλλοντα το οποίο δεν επηρεάζει το σκοπό του Πίνακας Διαγραμμάτων Χρήσης Τεχνικών Σφαλμάτων.

4. Bugs per Priority: Σε αυτό το διάγραμμα αποτυπώνεται ο αριθμός των τεχνικών σφαλμάτων ανά προτεραιότητα. Στο συγκεκριμένο παράδειγμα οι προτεραιότητες που έχουν επιλεγεί είναι Υψηλή (High), Μεσαία (Medium) και Χαμηλή (Low).
5. Bugs per Priority & Environment: Αυτό το διάγραμμα παρουσιάζει δεδομένα που έχουν προαναφερθεί αλλά βοηθάει τους χρήστες να απομονώσουν τα περιβάλλοντα και να αναγνωρίσουν εάν κάποιο από αυτά πρέπει να λυθεί με προτεραιότητα. Στο διάγραμμα που βλέπουμε στην Εικόνα 28, για παράδειγμα, είναι σαφές ότι υπάρχει ένα τεχνικό σφάλμα στην παραγωγή (Production) το οποίο έχει και Υψηλή προτεραιότητα (High Priority)
6. Bugs Discovery Method: Τέλος, αυτό το διάγραμμα αναδεικνύει τους τρόπους με τους οποίους το συγκεκριμένο τεχνικό σφάλμα αναγνωρίστηκε. Στο δικό μας παράδειγμα, έχουμε δύο πιθανούς τρόπους αναγνώρισης τεχνικών σφαλμάτων, αυτά που αναγνωρίστηκαν από ένα αυτοματοποιημένο σύστημα τεστ (automation testing) ή κάποιο χειροκίνητο τεστ (manual testing).



Εικόνα 27: *Πίνακας Διαγραμμάτων Χρήσης Τεχνικών Σφαλμάτων*

Οι παραπάνω πίνακες (boards) μπορούν να χρησιμοποιηθούν μεμονωμένα ή και συνδυαστικά από τις ομάδες ανάπτυξης για να τις βοηθήσουν να αυτό-οργανωθούν και να βελτιώσουν τις διαδικασίες αλλά και την αποτελεσματικότητά τους.

6. Μελέτη Περίπτωσης 2

6.1. Περιγραφή του Προβλήματος

Σε αυτό το σημείο θα θέλαμε να μελετήσουμε κατά πόσο θα μπορούσαν να χρησιμοποιηθούν τα Ευέλικτα μετρικά που έχουμε παρουσιάσει παραπάνω σε ομάδες ανάπτυξης λογισμικού που δεν ακολουθούν Ευέλικτες πρακτικές. Επιπλέον, σκοπός αυτής της άσκησης αποτελεί και η μελέτη και καταγραφή επιπλέον μετρικών που μπορεί να προκύψουν από την συλλογή δεδομένων από ένα Σύστημα Καταγραφής Εργασιών. Έτσι λοιπόν, για την δική μας ανάλυση χρησιμοποιήσαμε μια εργασία μαθήματος του Εθνικού Μετσόβιου Πολυτεχνείου (ΕΜΠ) προπτυχιακών φοιτητών. Στην εργασία αυτή λάβανε μέρος 197 φοιτητές που χωρίστηκαν σε 100 ομάδες – οι επιμέρους ομάδες εργασιών. Για την συλλογή των δεδομένων χρησιμοποιήσαμε ένα Σύστημα Καταγραφής Εργασιών, το YouTrack[10] το οποίο χρησιμοποιήθηκε για την καταγραφή των επιμέρους εργασιών που έλαβαν μέρος για την παράδοση της τελικής εργασίας που.

Μετά την αρχική ανάλυση, για την αποτελεσματικότερη μελέτη των δεδομένων μας, αναγκαστήκαμε να αποκλείσουμε τις ομάδες οι οποίες είχαν λιγότερα από 50 καταγεγραμμένες εργασίες για το έργο τους. Ένα δείγμα με συνολικά 51 καταγεγραμμένες εργασίες και παραπάνω θεωρήσαμε ότι είναι αποδεκτό για την μελέτη μας. Έτσι τελικά από 100 ομάδες αποκλείσαμε 89 και κρατήσαμε 11, επίσης αποκλείσαμε 176 φοιτητές μιας και οι ομάδες που απέμειναν αποτελούνταν από συνολικά 21 άτομα. Οι ομάδες που απέμειναν αποτελούνταν από ομάδες του ενός ή των δύο ατόμων. Πιο συγκεκριμένα είχαμε:

- Μία ομάδα του ενός ατόμου, και
- 10 ομάδες των δύο ατόμων.

Το εύρος τιμών των αριθμών tickets που είχαν ανοίξει από τις ομάδες ανάπτυξης κυμαίνονται από εώς 77 tickets ανά έργο.

Πίνακας 7: *Ανάλυση σχετικά με τις εργασίες που είχαν ανοιχτεί από τις ομάδες εργασίας στο YouTrack.*

Κωδικός έργου	Συνολικός αριθμός καταγεγραμμένων εργασιών	Ολοκληρωμένες εργασίες	Ποσοστό ολοκληρωμένων εργασιών	Μη ολοκληρωμένες εργασίες	Ποσοστό μη ολοκληρωμένων εργασιών	Δυναμική ομάδας
saas3	75	65	86.67%	10	13.33%	2
saas11	65	47	72.31%	18	27.69%	2
saas14	75	72	96.00%	3	4.00%	2
saas15	52	45	86.54%	7	13.46%	2
saas20	59	57	96.61%	2	3.39%	2
saas37	61	44	72.13%	17	27.87%	2

[10] https://www.jetbrains.com/youtrack/

saas40	71	71	100.00%	0	0.00%	2
saas46	59	46	77.97%	13	22.03%	2
saas47	52	50	96.15%	2	3.85%	2
saas52	77	66	85.71%	11	14.29%	1
saas64	79	79	100.00%	0	0.00%	2

6.2. Επιλογή Μετρικών

Για την επιλογή των μετρικών, όπως αναφέρθηκε και παραπάνω, αρχικά περιορίσαμε το δείγμα μας στις 11 ομάδες ανάπτυξης βάσει του συνολικού αριθμού εργασιών που έχουν καταγραφεί στο σύστημα που θα χρησιμοποιηθεί για την ανάλυση των δεδομένων. Μετά την επιλογή των ομάδων, το επόμενο βήμα είναι η επιλογή των μετρικών εστιάζοντας στις δυνατότητες της εφαρμογής και στα Agile μετρικά που έχουν αναφερθεί στα προηγούμενα κεφάλαια. Έτσι λοιπόν, για κάθε μια από τις κατηγορίες που έχουν αναφερθεί στην διπλωματική πολλά από τα μετρικά δεν μπορούν να χρησιμοποιηθούν για το λόγο ότι οι ομάδες ανάπτυξης δεν ακολουθούν Ευέλικτες πρακτικές ανάπτυξης λογισμικού και συνεπώς δεν χρησιμοποιούνται τα βασικά Agile μετρικά, όπως είναι τα Story Points η Ταχύτητα και η Δέσμευση της ομάδας, που αποτελούν η βάση για την ανάπτυξη επιπλέον μεθόδων και τρόπων ανάλυσης των δεδομένων μας. Επιπλέον τα μετρικά για Αρχιτεκτονικές Λογισμικών δεν μπορούσαν να χρησιμοποιηθούν επειδή δεν υπήρχε πρόσβαση στον πηγαίο κώδικα των επιμέρους ομάδων για επιπλέον ανάλυση και μελέτη.

Έτσι λοιπόν, στο επίπεδο των Agile μετρικών μπορούν να χρησιμοποιηθούν τα ακόλουθα:

- Αριθμός εξαρτήσεων μεταξύ των εργασιών
- Αριθμός εργασιών που παραδόθηκαν
- Ελαττώματα και σφάλματα & Χρόνος επίλυσης ελαττώματος/σφάλματος

Επιπλέον αυτών των μετρικών, θα χρησιμοποιήσουμε μέσω της πλατφόρμας του YouTrack τα δεδομένα που μας παρέχει για να μπορέσουμε να βγάλουμε κάποιο αποτέλεσμα σχετικά με την αποτελεσματικότητα της κάθε ομάδας. Τέλος, η προσέγγιση απεικόνισης σε κάποιο σύστημα απεικόνισης δεομένων που παρουσιάστηκε στο προηγούμενο κεφάλαιο, μπορεί να υλοποιηθεί και για αυτή την άσκηση αλλά δεν υλοποιήθηκε στα πλαίσια αυτής της διπλωματικής εργασίας.

6.2.1. Agile Μετρικά

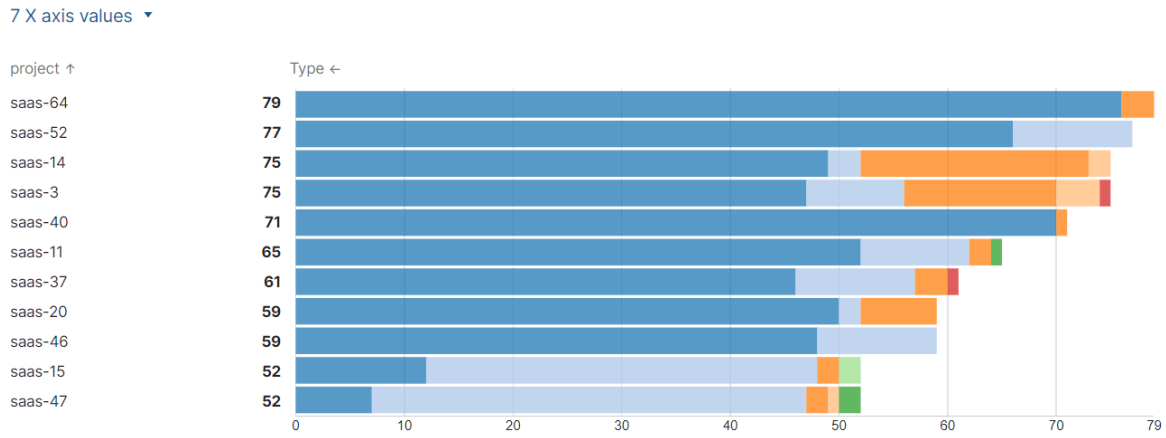
Για να μπορέσουμε να θέσουμε μια βάση σχετικά με τα δεδομένα που έχουμε από το σύστημα, παραθέτουμε τους Πίνακες 8, 9 και 10 που δίνουν τα παρακάτω δεδομένα:

1. Συνολικός τύπος εργασιών και αριθμός αυτών (Task, Feature, Bug, Epic, Cosmetics, Testing, Usability Problem).
2. Status των εργασιών (Verified, Done, Submitted, Open, In Progress, To be discussed, Won't fix, Obsolete).
3. Προτεραιότητα εργασιών και αριθμός αυτών (Normal, Critical, Major, Show-stopper, Minor).

Πίνακας 8: *Συνολικός τύπος εργασιών και αριθμός αυτών βάσει του YouTrack (Task, Feature, Bug, Epic, Cosmetics, Testing, Usability Problem).*

Ομάδα	Task	Feature	Bug	Epic	Cosmetics	Testing	Usability Problem
saas-64	76	0	3	0	0	0	0
saas-52	66	11	0	0	0	0	0
saas-14	49	3	21	2	0	0	0
saas-3	47	9	14	4	0	0	1
saas-40	70	0	1	0	0	0	0
saas-11	52	10	2	0	1	0	0
saas-37	46	11	3	0	0	0	1
saas-20	50	2	7	0	0	0	0
saas-46	48	11	0	0	0	0	0
saas-15	12	36	2	0	0	2	0
saas-47	7	40	2	1	2	0	0

Το οποίο μπορεί να αποτυπωθεί γραφικά:

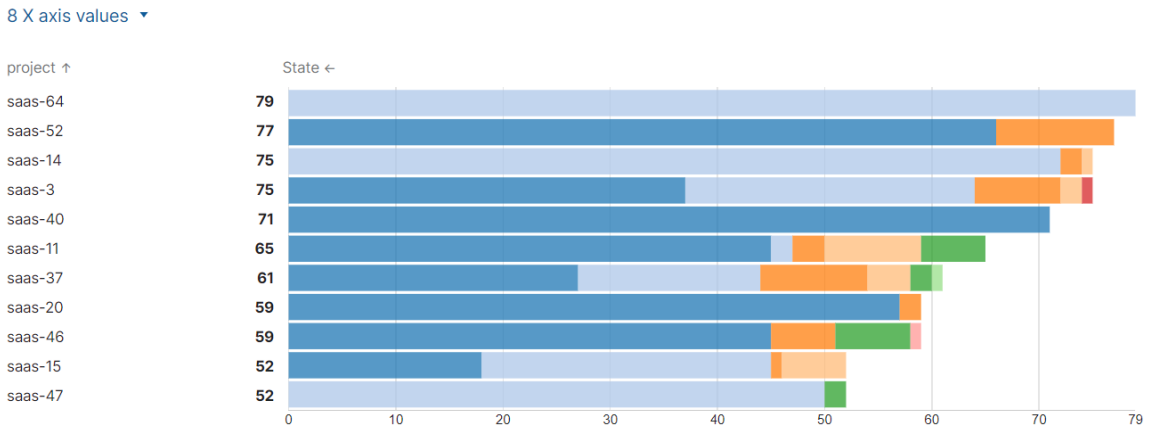


Εικόνα 28: *Γραφική αποτύπωση του Συνολικός τύπος εργασιών και αριθμός αυτών βάσει του YouTrack.*

Πίνακας 9: *Status των εργασιών βάσει του YouTrack (Verified, Done, Submitted, Open, In Progress, To be discussed, Won't fix, Obsolete).*

Ομάδα	Verified	Done	Submitted	Open	In Progress	To be discussed	Won't fix	Obsolete
saas-64	0	79	0	0	0	0	0	0
saas-52	66	0	11	0	0	0	0	0
saas-14	0	72	2	1	0	0	0	0
saas-3	37	27	8	2	0	0	1	0
saas-40	71	0	0	0	0	0	0	0
saas-11	45	2	3	9	6	0	0	0
saas-37	27	17	10	4	2	1	0	0
saas-20	57	0	2	0	0	0	0	0
saas-46	45	0	6	0	7	0	0	1
saas-15	18	27	1	6	0	0	0	0
saas-47	0	50	0	0	2	0	0	0

Το οποίο μπορεί να αποτυπωθεί και γραφικά:

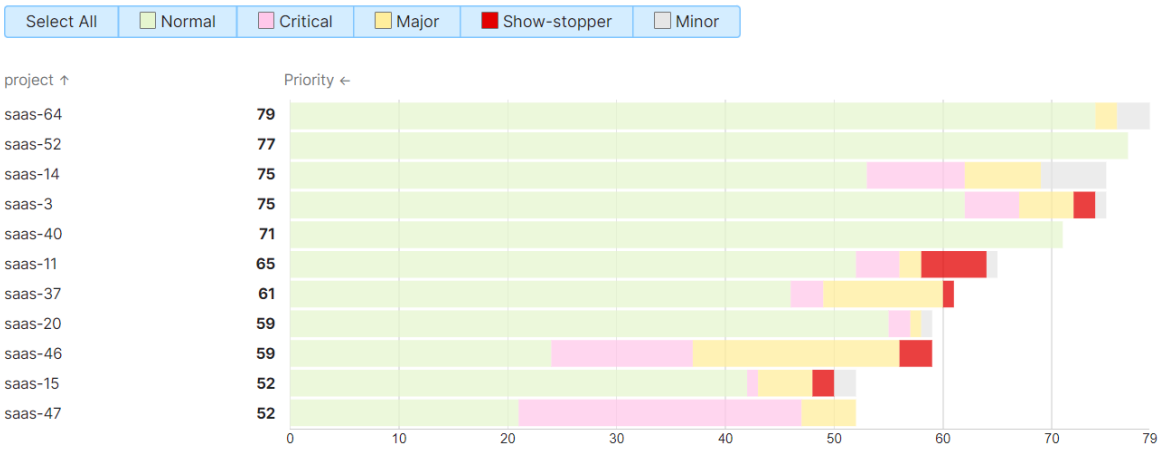


Εικόνα 29: *Γραφική αποτύπωση των Status των εργασιών βάσει του YouTrack.*

Πίνακας 10: *Προτεραιότητα εργασιών και αριθμός αυτών βάσει του YouTrack (Normal, Critical, Major, Show-stopper, Minor).*

Ομάδα	Normal	Critical	Major	Show-stopper	Minor
saas-64	74	0	2	0	3
saas-52	77	0	0	0	0
saas-14	53	9	7	0	6
saas-3	62	5	5	2	1
saas-40	71	0	0	0	0
saas-11	52	4	2	6	1
saas-37	46	3	11	1	0
saas-20	55	2	1	0	1
saas-46	24	13	19	3	0
saas-15	42	1	5	2	2
saas-47	21	26	5	0	0

Το οποίο μπορεί να αποτυπωθεί γραφικά:



Εικόνα 30: *Γραφική αποτύπωση της προτεραιότητας των εργασιών και του αριθμού αυτών βάσει του YouTrack.*

Ελαττώματα και σφάλματα & Χρόνος επίλυσης ελαττώματος/σφάλματος

Πίνακας 11: *Ανάλυση καταγεγραμμένων σφαλμάτων ανά ομάδα ανάπτυξης στο YouTrack.*

Ομάδα	Αριθμός Σφαλμάτων	Ανοιχτά Σφάλματα	Μέσος Χρόνος Επίλυσης Σφαλμάτων
saas3	14	1	6,2
saas11	2	2	
saas14	21	0	15,8
saas15	2	0	0,55
saas20	7	0	4,36
saas37	3	1	1,06
saas40	1	1	
saas46	0	0	0
saas47	2	0	0
saas52	0	0	0
saas64	3	0	15,62

Από τον παραπάνω πίνακα, θα εστιάσουμε στην ομάδα saas14η οποία είχε τον μεγαλύτερο αριθμό καταγεγραμμένων σφαλμάτων και μέσο χρόνο επίλυσης αυτών 15,8 ημέρες. Επίσης είναι εμφανές πως η ίδια ομάδα δεν έχει κανένα ανοιχτό σφάλμα. Επίσης αξίζει να παρατηρήσουμε την ομάδα saas20 που έχει καταγράψει 7 συνολικά σφάλματα και τα έχει επιλύσει όλα σε ένα μέσο χρόνο 4,36 ημέρες.

Έτσι λοιπόν μπορούμε να έχουμε μια συλλογική εικόνα για την ομάδα ανάπτυξης αλλά και για το προϊόν το οποίο έχουν καλεστεί να υποστηρίξουν. Επιπλέον, η χρήση των μετρικών αυτών στο πλαίσιο της ομάδας ανάπτυξης, μπορεί να χρησιμοποιηθεί από την ίδια την ομάδα για να γίνεται συνεχώς καλύτερη και να μπορεί γίνει πιο αποτελεσματική σε σημεία που θέλει να εστιάσει, στην

δική μας περίπτωση στα σφάλματα και στον χρόνο αντίδρασης από την στιγμή εύρεσης ενός σφάλματος μέχρι την επίλυσή του.

Επιπλέον μια ακόμα διάσταση που μπορεί να δοθεί σε αυτά τα μετρικά είναι η κρισιμότητα των σφαλμάτων. Έτσι λοιπόν, εάν εστιάσουμε στην ομάδα με τα περισσότερα καταγεγραμμένα σφάλματα (saas-14), έχουμε τα παρακάτω αποτελέσματα:

Πίνακας 12: *Σφάλματα και κρισιμότητα αυτών για την ομάδα saas-14 όπως εξήχθησαν από το YouTrack.*

Κρισιμότητα	**Αριθμός σφαλμάτων**	**Μέσος χρόνος επίλυσης**
Critical	4	1,37
Normal	15	19,84
Minor	2	17,1

Είναι εμφανές από τον πίνακα 12 πως η ομάδα ανάπτυξης έχει πολύ καλό χρόνο απόκρισης στην επίλυση κρίσιμων σφαλμάτων, αλλά μπορεί να βελτιωθεί πολύ στην επίλυση μεσαίας κρισιμότητας σφαλμάτων.

Τυπική παράβαση - Standard violation – Εργασίες χωρίς πληροφορία

Όπως αναφέραμε και σε προηγούμενο κεφάλαιο, σαν Τυπική παράβαση μπορεί να καταγραφεί η παράβαση προτύπων που παραβιάζονται ανά Sprint. Έτσι για το δικό μας παράδειγμα θέτουμε σαν Τυπική παράβαση όταν κάποια από τις εργασίες δεν περιέχει καθόλου λεπτομέρειες/περιγραφή. Αυτό το μετρικό επιλέχθηκε για την αποτελεσματική συνεργασία των μελών της ομάδας ανάπτυξης.

Πίνακας 13: *Τυπική παράβαση – αριθμός εργασιών που δεν έχουν περιγραφή, ανά ομάδα.*

Ομάδα	**Συνολικός αριθμός καταγεγραμμένων εργασιών**	**Αριθμός εργασιών χωρίς περιγραφή**	**Ποσοστό εργασιών χωρίς περιγραφή**	**Μέσος αριθμός χαρακτήρων περιγραφών**
saas3	75	55	73%	17,29
saas11	65	15	23%	118,78
saas14	75	67	89%	5,66
saas15	52	0	0%	9,6
saas20	59	59	100%	0
saas37	61	61	100%	0
saas40	71	71	100%	0
saas46	59	59	100%	0
saas47	52	39	75%	17,05
saas52	77	77	100%	0
saas64	79	5	6%	65,07

Από τα δεδομένα που συλλέχθηκαν, οι ομάδες saas20, saas37, saas40, saas46 και saas52 δεν είχαν σε κανένα από τις εργασίες που είχαν καταγράψει περιγραφή. Συνεπώς, εάν έχει παρατηρηθεί δυσκολία στην συνεργασία της ομάδας ανάπτυξης αυτός είναι πολύ πιθανό να είναι ένας από τους λόγους. Έτσι παρατηρούμε ότι τα μετρικά αυτά μπορούν να μας αναδείξουν ένα πρόβλημα που μπορεί να έχει η ομάδα ανάπτυξης και να βοηθήσει την ίδια την ομάδα να γίνει πιο παραγωγική και αποτελεσματική. Το πλεονέκτημα που έχουμε χρησιμοποιώντας Agile πρακτικές είναι πως η ομάδα

ανάπτυξης μπορεί να εστιάζει στα μετρικά και τα αποτελέσματά τους σε τακτά χρονικά διαστήματα (σε κάθε Sprint) και να μπορεί να εστιάζει σε ότι δεν πήγε καλά σε κάθε Sprint και να βάζει νέους στόχους για την αυτοβελτίωσή της. Από την άλλη σε ένα μη Agile περιβάλλον δεν έχουν οριστεί τα διαστήματα στα οποία η ομάδα μελετά τα αποτελέσματα των μετρικών και συνεπώς αυξάνεται η πολυπλοκότητα και μπορεί να μην ληφθούν ποτέ ενέργειες για την βελτίωσή της από τα ίδια τα μέλη της ομάδας.

Τυπική παράβαση - Standard violation – Reopened tickets

Θέτεται σαν Τυπική παράβαση το εκ νέου άνοιγμα κλειστών εργασιών ανά την διάρκεια του χρόνου όπου ένα έργο είναι ενεργό. Έτσι, από τα δεδομένα που ανακτήθηκαν, παρατηρήσαμε ανακτήσαμε τον αριθμό των εργασιών που είχαν κλείσει και ανοίξανε ξανά.

Πίνακας 14: *Τυπική παράβαση – αριθμός εργασιών που ενώ είχαν καταγραφεί σαν ολοκληρωμένες ανοίξανε εκ νέου, ανά ομάδα.*

Ομάδα	Reopened tickets
saas3	3
saas11	11
saas14	7
saas15	1
saas20	0
saas37	1
saas40	0
saas46	0
saas47	3
saas52	0
saas64	0

Αυτό το μετρικό μπορεί να επηρεάσει τον τρόπο λειτουργίας μιας ομάδας ανάπτυξης και την επικοινωνία εντός αυτής. Συνεπώς, οι ομάδες ανάπτυξης αφού έχουν ορίσει ένα Definition of Done (DoD), και αφότου μια εργασία τους θεωρείται ολοκληρωμένη, δεν θα πρέπει να ανοίγουν τις εργασίες που έχουν ολοκληρωθεί. Εάν αυτό συμβεί προτείνεται να παρακολουθείται και να λαμβάνονται τα απαραίτητα μέτρα για την αποφυγή αυτής της ενέργειας (αλλαγή του DoD, επιπλέον μέτρα για την εξασφάλιση της ποιότητας κλπ). Όπως φαίνεται και από τον πίνακα 14, η ομάδα saas-11 είχε ανοίξει 11 επιμέρους εργασίες αφού τις είχε κλείσει. Αναλύοντας σε βάθος αυτές τις εργασίες και το χρονικό πλαίσιο στο οποίο αυτές οι εργασίες άνοιξαν ξανά, βλέπουμε ότι υπήρχαν αρκετές καθυστερήσεις στην ανάπτυξη και στην επίλυση επιπλέον εργασιών την ίδια περίοδο.

Cycle Time

Όπως αναφέρθηκε και στο θεωρητικό κομμάτι της εργασίας, ο όρος Cycle Time αποτελεί ο χρόνος από την εκκίνηση ενός task ή user story έως την χρονική στιγμή που είναι έτοιμη για παράδοση. Στη δική μας περίπτωση, θα μετρήσουμε τον χρόνο από την δημιουργία ενός task μέχρι την ολοκλήρωση αυτού για μια ομάδα ανάπτυξης. Να σημειωθεί πως η άσκηση αυτή μπορεί να εφαρμοστεί για όλες τις ομάδες μιας και το YouTrack παρέχει δεδομένα για την δημιουργία αλλά και για την ολοκλήρωση ενός task. Για το δικό μας παράδειγμα θα χρησιμοποιήσουμε δεδομένα από την ομάδα saas-64 η

οποία είχε καταγράψει συνολικά 76 Tasks και 3 Bugs, τα οποία τα παρέδωσε όλα (Status == Done). Η ομάδα έχει ένα μέσο χρόνο παράδοσης 5.41 ημέρες. Οι τιμές κυμαίνονται από 0 (δηλαδή λύθηκαν την ίδια ημέρα από την ημέρα παράδοσης) έως και 31.8 ημέρες. Δυστυχώς το σύστημα δεν μας παρέχει την πληροφορία σε επίπεδο ώρας που θα είχαμε μια καλύτερη εικόνα για τον ακριβή χρόνο που απαιτείται από την ομάδα για την επίλυση μιας εργασίας από την στιγμή όπου αυτή καταγράφεται μέχρι την επίλυσή της.

6.3. Επιπλέον Μετρικά που θα μπορούσαν να αποτυπωθούν

Σε αυτό το κεφάλαιο θα αποτυπώσουμε τα επιπλέον μετρικά που θα μπορούσαν να χρησιμοποιηθούν σε ένα μη Agile πλαίσιο εργασίας από αυτά που έχουν παρουσιαστεί στο θεωρητικό κομμάτι της διπλωματικής εργασίας, σε περίπτωση όπου οι ομάδες εργασίες συνεργάζονταν ή ήταν διαθέσιμες να τρέξουν κάποιες ασκήσεις για την αποτύπωση αυτών των μετρικών.

Για παράδειγμα τα μετρικά που παρουσιάζονται στο κεφάλαιο 3.2 Μετρικά Συνεργατικότητας θα μπορούσαν να υποστηριχτούν και σε ένα μη Agile πλαίσιο εργασίας. Όπως επίσης και κάποια από τα μετρικά που παρουσιάζονται στο κεφάλαιο 3.3 Μετρικά Παράδοσης και πιο συγκεκριμένα:

1. Ο αριθμός απρογραμμάτιστων εργασιών που μπαίνουν σε κάποια περίοδο όπου η ομάδα ανάπτυξης έχει προγραμματίσει να υλοποιήσει συγκεκριμένα πράγματα. Να σημειωθεί πως αυτό μπορεί να αποτυπωθεί και γενικότερα ως μια Τυπική παράβαση – Standard violation του τρόπου που λειτουργεί η ομάδα ανάπτυξης.
2. Το Release Burnout Chart μπορεί να αποτυπώσει επίσης είτε τον αριθμό των εργασιών που απομένουν για να μπορέσει η ομάδα ανάπτυξης να παραδώσει μια νέα έκδοση ή και με την βοήθεια των Story Points σε ένα waterfall μοντέλο ανάπτυξης, εφόσον η ομάδα ανάπτυξης έχει μάθει να ποντοδοτεί τις εργασίες της με αυτό τον τρόπο.
3. Το Cycle και το Lead Time είναι ένα επιπλέον μετρικό που θα μπορούσε να μετρηθεί και να αξιολογηθεί και σε μη Agile περιβάλλοντα. Είναι κατανοητό όμως πως σε πιο δυσκίνητες πρακτικές ανάπτυξης πως αυτά τα μετρικά δεν έχουν καμία αξία μιας και εξυπηρετείται ένα συγκεκριμένο μεγάλο πλάνο με επιμέρους εργασίες και πακέτα εργασιών. Έτσι η σύλληψη μιας ιδέας μέχρι την τελική υλοποίησή της παίρνει αρκετό χρόνο και συνεπώς δεν υπάρχει καμία αξία στην μέτρησή της.

Επιπλέον όπως είδαμε και στην Μελέτη Περίπτωσης 2, η χρήση μερικών από τα 3.4 Μετρικά Ποιότητας είναι εφικτή.

Αυτό που παρατηρήθηκε έντονα είναι πως τα περισσότερα από τα Agile μετρικά δεν ήταν εφικτό να χρησιμοποιηθούν, επειδή οι ομάδες ανάπτυξης που μελετήθηκαν δεν ακολουθούσαν τεχνικές όπως η ποντοδότηση των εργασιών και επιπλέον δεν υπήρχε η διαρκής επανάληψη των Sprints που αποτελούν οι γερές βάσεις για την συνεχή τροφοδότηση των μετρικών και της αυτό-βελτίωσης των ομάδων εργασίας.

7. Συμπεράσματα & Μελλοντική εργασία

Η ταχεία εξέλιξη των ηλεκτρονικών συστημάτων και των χαρακτηριστικών αυτών, όπως και η αναγνώριση των χαρακτηριστικών όπου έχουν σημασία για τους οργανισμούς, έχουν αποτελέσει βασικό χαρακτηριστικό για την μελέτη και την παρακολούθηση δεδομένων. Βασική ανάγκη αποτελεί η παραγωγή και ο συνδυασμός δεδομένων από διαφορετικούς τομείς για να βοηθήσουν τους οργανισμούς και της ομάδες ανάπτυξης να βελτιώνονται συνεχώς. Οι βασικές περιοχές που αναφέρθηκαν σε αυτή τη διπλωματική εργασία αποτελούν οι εξής:

1. Πηγαίος κώδικας και τα συστήματα παραγωγής.
2. Ομάδες ανάπτυξης Λογισμικού και τα παραδοτέα αυτών.
3. Μετρικά που εξάγονται από τους πελάτες και τους χρήστες του τελικού προϊόντος.

Οι τρόποι αποτύπωσης της πολυπλοκότητας του πηγαίου κώδικα έχουν εξελιχθεί σε επίπεδο επεκτασιμότητας των εργαλείων και των μέσων μελέτης αλλά και ενσωμάτωσης αυτών σε πλατφόρμες. Θα ήταν άξιο μελέτης η χρήση τέτοιων μέσων για την υποστήριξη ομάδων ανάπτυξης λογισμικού σε δύο επίπεδα:

1. Στην υποστήριξη των ομάδων ανάπτυξης λογισμικού για την αποτελεσματικότερη εκτίμηση εργασιών. Με τη χρήση αυτών των μεθόδων οι ομάδες ανάπτυξης θα μπορούν να εκτιμούν αποτελεσματικότερα την εργασία που καλούνται να ολοκληρώσουν σε επόμενα Sprints μιας και θα μπορούν να έχουν άμεσα εικόνα της υγείας και της ευκολίας επέκτασης του πηγαίου κώδικα που καλούνται να αλλάξουν. Αξίζει να σημειωθεί εδώ ότι σε μελέτες που εφαρμόστηκαν 39 πηγαίους κώδικες παρατηρήθηκε πως οι περιοχές κώδικα που χρήζουν πηγαίος κώδικας με υψηλό τεχνικό χρέος αποτελούν αυτοί όπου δέχονται πολλές φορές αλλαγές από πολλά διαφορετικά άτομα [23]. Πράγμα που μπορεί να βοηθήσει τις ομάδες ανάπτυξης να λαμβάνουν αποφάσεις σχετικά με τον τρόπο διαμοίρασης του πηγαίου κώδικα κ.α.
2. Την προτεραιοποίηση του τεχνικού χρέους συναρτήσει του Product Backlog των ομάδων ανάπτυξης. Η προτεραιοποίηση του τεχνικού χρέους αποτελεί ολοένα και αναγκαιότερη όσο ένας οργανισμός και οι απαιτήσεις των χρηστών αυξάνονται. Για αυτό το λόγο θα ήταν καλό να μελετηθεί εάν προτού μια ομάδα ανάπτυξης δουλέψει για μια νέα λειτουργικότητα να καλύψει μέρος του τεχνικού χρέους που ανήκει σε μια νέα λειτουργικότητα θα πρέπει να προτεραιοποιηθεί προτού υλοποιήσει αυτή τη νέα λειτουργικότητα θα επιφέρει θετικά αποτελέσματα στη βιωσιμότητα του πηγαίου κώδικα, στην υγεία της ομάδας και των ατόμων αυτής κλπ.

Συνεπώς η χρήση μετρικών τόσο σε επίπεδο ομάδας αλλά και του πηγαίου κώδικα όπως και η αναζήτηση μετρικών που συνεπαγωγικά μπορεί να παράγονται από αυτές τις δύο γενικές μεταβλητές θα πρέπει να μελετηθούν ώστε να βοηθήσουν τους οργανισμούς και τα άτομα αυτών να εξελίσσονται και να παρέχουν ποιοτικές υπηρεσίες προς τους τελικούς χρήστες.

Ο τρόπος λειτουργίας των ομάδων ανάπτυξης αλλά και τα τελικά παραδοτέα που παρέχονται προς τους χρήστες είναι αλληλένδετα μιας και το ένα επηρεάζει το άλλο. Έτσι λοιπόν μια από κοινού μελέτη πως η αποδοτικότητα της ομάδας και η ικανοποίηση των τελικών χρηστών αποτελεί αναγκαία.

Μέσα από τα συμφραζόμενα αυτό που προτείνεται είναι η συνδυαστική χρήση μετρικών για την μελέτη του επιπέδου αλληλεπίδρασης των τριών περιοχών και ίσως και της πρότασης/δημιουργίας μετρικών που συλλέγουν δεδομένα από πολλές περιοχές (ίσως και από τις τρείς) και τελικά δίνουν ένα αποτέλεσμα που είναι κοινά κατανοητό και αποδεκτό από τους αναγνώστες του. Το αποτέλεσμα αυτής της διπλωματικής από την άλλη μεριά αναδεικνύει πως αξιοποιώντας τις Agile τεχνικές ανάπτυξης, ανοίγει παράλληλα και ένα μεγάλο μέρος μετρικών και τρόπων μελέτης των ομάδων ανάπτυξης που δεν υποστηρίζονται σε αντίθετες περιπτώσεις. Επιπλέον είναι εφικτό για τους μεγάλους οργανισμούς, με την χρήση και την παρουσίαση μετρικών, να ωθούν όλες τις ομάδες ανάπτυξης που υποστηρίζουν προϊόντα να εστιάζουν και να βελτιώνουν τις διαδικασίες και τις αποδόσεις τους. Από την άλλη, η έννοια των συχνών επαναλήψεων και των τρόπων με τους οποίους εκτιμάται η εργασία στο Agile, θέτουν γερές βάσεις για την υποστήριξη των επιπλέον μετρικών που χρησιμοποιούνται από τις ομάδες ανάπτυξης για την αυτοβελτίωσή τους. Αντίθετα με άλλες τεχνικές ανάπτυξης (όπως για παράδειγμα το Waterfall) το οποίο μπορεί να χρησιμοποιήσει κάποια από αυτά τα μετρικά όχι όμως όλα.

8. Βιβλιογραφία

[1] Ken Schwaber, Jeff Sutherland.. (November 2020). The Scrum Guide, The Definitive Guide to Scrum: The Rules of the Game. Creative Commons. Available at: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf#zoom=100.

[2] Diane Strode, Sid L. Huff. (2012). A Taxonomy of Dependencies in Agile Software Development. 23rd Australasian Conference on Information Systems. 23. [Online]. Available at: https://www.researchgate.net/publication/267706181_A_Taxonomy_of_Dependencies_in_Agile_Software_Deve.

[3] Alexander Scheerer, Saskia Bick, Tobias Schimmer, Armin Heinzl. (2015). The Effects of Team Backlog Dependencies on Agile Multiteam Systems: A Graph Theoretical Approach. 48th Hawaii International Conference on System Sciences. 48, [Online]. Available at: 10.1109/HICSS.2015.606.

[4] Christoph Matthies, Thomas Kowark, Keven Richly, Matthias Uflacker. (2016). ScrumLint: Identifying Violations of Agile Practices Using Development Artifacts. 9th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE'16)At: Austin, TX, USA. 9. [Online]. Available at: 10.1145/2897586.289760.

[5] Sunju Oh, Heon Yeom, Joongho Ahn. (2011). Cohesion and coupling metrics for ontology modules. Information Technology and Management. 12(2), pp.81-96. [Online]. Available at: 10.1007/s10799-011-0094-5.

[6] Rim Djedidi, Marie-Aude Aufaure. (2010). Article Ontology Evolution: State of the Art and Future Directions. Ontology Theory, Management and Design: Advanced Tools and Models. 7, p.179-207. [Online]. Available at: 10.4018/978-1-61520-859-3.ch007.

[7] Michele Lanza, Radu Marinescu. (2006). Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems: Springer. [Online]. Available at: 10.1007/3-540-39538-5.

[8] Charles Ashbacher. (2003). Test-Driven Development: By Example, by Kent Beck. Journal of Object Technology. 2. [Online]. Available at: 10.5381/jot.2003.2.2.r1.

[9] Richard Wettel, Michele Lanza, Romain Robbes. (2011). Software systems as cities: A controlled experiment. Proceedings of the 33rd International Conference on Software Engineering, ICSE 2: ACM. p.551– 560. [Online]. Available at: 10.1145/1985793.1985868.

[10] Michele Lanza. (2003). CodeCrawler-lessons learned in building a software visualization tool. Proceedings of the Euromicro Conference on Software Maintenance and Reengineering: IEEE. p. 409–418. [Online]. Available at: 10.1109/CSMR.2003.1192450

[11] Richard Wettel, Michele Lanza. (2007). Program Comprehension through Software Habitability. 15th IEEE International Conference on Program Comprehension (ICPC '07): IEEE. p.231–240. [Online]. Available at: 10.1109/ICPC.2007.30.

[12] Richard Wettel, Michele Lanza. (2008). Visual Exploration of Large-Scale System Evolution. Conference: WCRE 2008, Proceedings of the 15th Working Conference on Reverse Engineering, WCRE: IEEE. p.219–228. [Online]. Available at: 10.1109/WCRE.2008.55.

[13] Richard Wettel. (2009). Visual Exploration of Large-Scale Evolving Software. 2009 31st International Conference on Software Engineering - Companion Volume: IEEE. p.391–394. [Online]. Available at: 10.1109/ICSE-COMPANION.2009.5071029.

[14] Alam Sazzadul, Boccuzzo Sandro, Wettel Richard, Dugerdil Philippe, Gall Harald, Lanza Michele. (2009). EvoSpaces - Multi-dimensional Navigation Spaces for Software Evolution. Proceedings of the Euromicro Conference on Software Maintenance and Reengineering: Springer. p.167–192.

[15] Dean Leffingwell. (2007). Scaling Software Agility: Best Practices for Large Enterprises.

[16] Yassine Ali, Braha Dan. (2003). Complex Concurrent Engineering and the Design Structure Matrix Method. Concurrent Engineering Research and Applications. 11(3), p.165–176. [Online]. Available at: https://www.researchgate.net/publication/246228952_Complex_Concurrent_Engineering_and_the_Design_Structure_Matrix_Method.

[17] Eppinger Steven, Browning Tyson. (2012). Design Structure Matrix Methods and Applications: The MIT Press.

[18] Cheong Wei-Hien, Tan Yung-Chie, Yap Soon-Joo, Ng Kee Peng. (2015). ClicO FS: An interactive web-based service of Circos. Bioinformatics. 31(22), p.3685–3687. [Online]. Available at: 10.1093/bioinformatics/btv433.

[19] An Jiyuan, Lai John, Sajjanhar Atul, Batra Jyotsna, Wang Chenwei, Nelson Colleen. (2015). J-Circos: An Interactive Circos plotter. Bioinformatics. 31(9), p.1463–1465. [Online]. Available at: 10.1093/bioinformatics/btu842.

[20] Krzywinski Martin, Schein Jacqueline, Birol Inanç, Connors Joseph, Gascoyne Randy, Horsman Doug, Jones Steven, Marra Marco. (2009). CIRCOS: an information aesthetic for comparative genomics. Genome Research. 19(9), p.1639–1645. [Online]. Available at: 10.1101/gr.092759.109.

[21] Darzentas Nikos. (2010). Circoletto: Visualizing sequence similarity with Circos. Bioinformatics. 26(20), pp.2620-2621. [Online]. Available at: 10.1093/bioinformatics/btq484.

[22] Scrum.org (2020). The Evidence-Based Management Guide - Measuring Value to Enable Improvement and Agility. Creative Commons. [Online]. Available at: https://www.scrum.org/resources/evidence-based-management.

[23] Adam Tornhill, Markus Borg. (2022). Code Red: The Business Impact of Code Quality -- A Quantitative Study of 39 Prop. Sweden: RISE Research Institutes of Sweden. [Online]. Available at: https://www.researchgate.net/publication/359129462_Code_Red_The_Business_Impact_of_Code_Quality_--_A_Quantitative_Study_of_39_Proprietary_Production_Codebases

[24] ifulltest.github.io (2020). What are different frameworks within Agile methodology?. [Online]. Available at: https://ifulltest.github.io/en/posts/different-agile-methodology/

[25] kanbanize.com (2021). What Is a Kanban Board and How to Use It? Basics Explained. [Online]. Available at: https://kanbanize.com/kanban-resources/getting-started/what-is-kanban-board

[26] www.lucidchart.com (2016). What is extreme programming? An overview of XP rules and values. [Online]. Available at: https://www.lucidchart.com/blog/what-is-extreme-programming

[27] www.nimblework.com (2020). Lead Time & Cycle Time Metrics: What Do They Reveal?. [Online]. Available at: https://www.nimblework.com/agile/lead-time-cycle-time/

[28] Wa'el Mohsen, Mostafa Aref, Khaled ElBahnasy. (2017) Software Metrics for Cooperative Scrum Based Ontology Analysis: 2017 2nd International Conference on Knowledge Engineering and Applications. IEEE. p.60-79. [Online]. Available at: 10.1109/ICKEA.2017.8169903

[29] Maria Teresa Di Martino, Pietro Hiram Guzzi, Daniele Caracciolo, Luca Agnelli. (2015) Integrated analysis of microRNAs, transcription factors and target genes expression discloses a specific molecular architecture of hyperdiploid multiple myeloma. [Online]. Available at: 10.18632/oncotarget.4302

[30] Huaigui Liu, Lixue Xu, Jilian Fu, Qian Su, Nana Liu, Jiayuan Xu, Jie Tang, Wei Li, Fangshi Zhao, Hao Ding, Feng Liu, Wen Qin, Chunshui Yu (2020) Prefrontal Granule Cell-Related Genes and Schizophrenia. [Online]. Available at: 10.1093/cercor/bhaa360