ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ Μ/Υ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΣΧΟΛΗ ΝΑΥΤΙΛΙΑΣ ΚΑΙ ΒΙΟΜΗΧΑΝΙΑΣ
ΤΜΗΜΑΤΟΣ ΒΙΟΜΗΧΑΝΙΚΗΣ ΔΙΟΙΚΗΣΗΣ & ΤΕΧΝΟΛΟΓΙΑΣ
ΔΙΑΠΑΝΕΠΙΣΤΗΜΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΤΕΧΝΟ-ΟΙΚΟΝΟΜΙΚΑ ΣΥΣΤΗΜΑΤΑ»

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Διασφάλιση ποιότητας στον τομέα της μηχανικής λογισμικού

ΧΑΝΤΖΑΚΟΥ ΦΡΑΝΤΖΕΣΚΑ ΕΙΡΗΝΗ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:
ΔΟΥΛΑΜΗΣ ΝΙΚΟΛΑΟΣ Καθηγητής Τομέα Τοπογραφίας

ΟΚΤΩΒΡΙΟΣ 2022

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου Δουλάμη Νικόλαο, Καθηγητή Τομέα Τοπογραφίας, για τη βοήθειά του, τις χρήσιμες συμβουλές του και για την άψογη συνεργασία που είχαμε καθόλη την διάρκεια της διπλωματικής εργασίας. Εν συνεχεία, θα ήθελα να ευχαριστήσω από καρδιάς την οικογένειά μου και τον σύντροφό μου για την στήριξη που μου παρείχαν καθ' όλη την διάρκεια των σπουδών μου.

Περίληψη

Σήμερα ο κλάδος της μηχανικής λογισμικού εξελίσεται ραγδαία. Οι τεχνολογίες αναπτύσσονται εκθετικά και η αγορά είναι γεμάτη από επιμέρους ειδικότητες που συνεισφέρουν στην τελική παράδοση ενός προιόντος λογισμικού. Για ποικίλους λόγους, όπως η ανάγκη για γρήγορη παράδοση τμημάτων του προιόντος, το ανθρώπινο λάθος, η έλλειψη αποτελεσματικών δεξιοτήτων, καθίσταται απαραίτητη η διασφάλιση της ποιότητας των υπηρεσιών που παρέχονται. Στο σημείο αυτό λοιπόν κατά συνέπεια επαφίεται η ομάδα υλοποίησης στις τεχνικές που διέπουν τις αρχές δοκιμών λογισμικών, με απώτερο στόχο την παράδοση υπηρεσιών λογισμικού που αντιστοιχούν στις απαιτήσεις του εκάστοτε πελάτη. Χρησιμοποιώντας την τεχνολογία στον κλάδο της διασφάλισης ποιότητας στη μηχανική λογισμικού, εκτελούνται πλέον αυτόματες προγραμματισμένες δοκιμές καθημερινά, ελαχιστοποιώντας έτσι τα ελαττώματα ενός προϊόντος, σχετικά νωρίς στον κύκλο ζωής ενός προϊόντος.

Στην παρούσα διπλωματική που εκπονήθηκε, στο πρώτο κεφάλαιο αναφέρεται το σχετικό θεωρητικό υπόβαθρο. Διαδικασίες, μεθοδολογίες, τύπου και επίπεδα δοκιμών εξετάζονται, και εν τέλει αποτυπώνονται με τέτοιο τρόπο με στόχο να αναδειχθεί η εν γένει συνεισφορά της διασφάλισης ποιότητας στην παράδοση των προϊόντος λογισμικού. Στο δεύτερο κεφάλαιο αναφέρεται η μεθοδολογία που έχει ακολουθηθεί, η εφαρμογή που αναλύθηκε και επεξεργάστηκε, το λεγόμενο σύστημα δηλαδή που υπόκειται σε δοκιμή. Στο τρίτο κεφάλαιο παρουσιάζονται τα τελικά αποτελέσματα των δοκιμών που εκμαιεύτηκαν, και κατά συνέπεια τα συμπεράσματα που προκύπτουν μέσα από αυτά, επαληθεύοντας έτσι το γεγονός ότι η διασφάλιση ποιότητας στη μηχανική λογισμικού καθίσταται απαραίτητη για την επιτυχή παράδοση προϊόντος σύμφωνα με τις απαιτούμενες προδιαγραφές.

Λέξεις - Κλειδιά: Δοκιμή Λογισμικού, Αυτοματοποιημένη Δοκιμή Λογισμικού, Χειροκίνητη Δοκιμή Λογισμικού, Επίπεδα δοκιμής, Σφάλματα λογισμικού

Abstract

Today, the field of software engineering is being developed rapidly. Technologies are being developed exponentially and current market is full of software engineering subspecialties that contribute to the final delivery of a software product. For various reasons, such as the need for quick delivery of product parts, human error, lack of effective skills, it becomes necessary to assure the services provide quality. At this point, therefore, the development team is left to the techniques that govern the principles of software testing, with the ultimate goal of delivering software services that correspond to client requirements. Using technology into the quality assurance sector of software engineering, automated scheduled tests are now performed daily, minimizing thus the defects in a product development lifecycle.

In the present thesis, the first chapter mentions the relevant theoretical background. Processes, methodologies, types and levels of testing are examined, and ultimately captured in such a way as to highlight the overall contribution of quality assurance to the delivery of software products. The second chapter mentions the methodology followed, the application that has been analyzed and processed and the system under test. In the third chapter, the final results of the elicited tests are presented, and consequently the conclusions derived from them, verifying thus the fact that quality assurance in software engineering becomes necessary for the successful delivery of a product according to the required specifications.

Keywords: Software Testing, Automated Testing, Manual Testing, Test Levels, Defects

Περιεχόμενα

Πίνακας Εικόνων

Πίνακας Πινάκων

Πίνακας Διαγραμμάτων

Εισαγωγή

Καθημερινά οι περισσότερες ενέργειες ενός ανθρώπου που λαμβάνουν χώρα εξαρτώνται από λογισμικά ή εφαρμογές, που μπορεί να σχετίζονται για παράδειγμα με τις online τραπεζικές συναλλαγές, με την online αγορά προϊόντων, με την online επικοινωνία, με την online πλοήγηση, με τη online διασκέδαση όπως social media, online betting και gaming και πολλά άλλα ακόμη. Η καθημερινή χρήση πολλών εφαρμογών, έχει οδηγήσει τους ανθρώπους που τις χρησιμοποιούν να εμπιστεύονται τα αποτελέσματα που θα τους επιστραφούν χωρίς δεύτερη σκέψη. Τίθεται το ερώτημα, τι θα συμβεί όμως εάν μια συναλλαγή μιας τράπεζας αποτύχει, εάν ένα πολύ σημαντικό email χαθεί ή εάν το απόθεμα της αποθήκης ενός ηλεκτρονικού καταστήματος δεν μειωθεί μετά από μια αγορά; Σε αυτό το ερώτημα έρχεται να απαντήσει ο έλεγχος των εφαρμογών και η διασφάλιση της ποιότητας αυτών (software testing και quality assurance). Καθώς πρέπει να ενσωματώνεται σε κάθε ανάπτυξη λογισμικού για να απαντηθεί όσο το δυνατόν νωρίτερα το παραπάνω ερώτημα και στην περίπτωση που δεν υπάρχει η αναμενόμενη συμπεριφορά στην εφαρμογή να διορθωθεί πριν να καταλήξει στα χέρια των τελικών χρηστών. Εν ολίγοις, δεν μπορεί να υπάρξει μια εφαρμογή στην αγορά χωρίς αυτή να έχει δοκιμαστεί από τους κατάλληλους ανθρώπους και τα κατάλληλα εργαλεία. Ο χώρος του software testing και quality assurance αποτελεί τα τελευταία χρόνια απαραίτητο μέρος των εταιρειών ανάπτυξης λογισμικού (software development).

Η δομή της παρούσας διπλωματικής εργασίας είναι η ακόλουθη, στο πρώτο κεφάλαιο αναφέρεται το σχετικό θεωρητικό υπόβαθρο. Διαδικασίες, μεθοδολογίες, τύπου και επίπεδα δοκιμών εξετάζονται, και εν τέλει αποτυπώνονται με τέτοιο τρόπο με στόχο να αναδειχθεί η εν γένει συνεισφορά της διασφάλισης ποιότητας στην παράδοση των προϊόντος λογισμικού. Στο δεύτερο κεφάλαιο αναφέρεται η μεθοδολογία που έχει ακολουθηθεί, η εφαρμογή που αναλύθηκε και επεξεργάστηκε, το λεγόμενο σύστημα δηλαδή που υπόκειται σε δοκιμή. Στο τρίτο κεφάλαιο παρουσιάζονται τα τελικά αποτελέσματα των δοκιμών που εκμαιεύτηκαν, και κατά συνέπεια τα συμπεράσματα που προκύπτουν μέσα από αυτά, επαληθεύοντας έτσι το γεγονός ότι η διασφάλιση ποιότητας στη μηχανική λογισμικού καθίσταται απαραίτητη για την επιτυχή παράδοση προϊόντος σύμφωνα με τις απαιτούμενες προδιαγραφές.

ΚΕΦΑΛΑΙΟ 1ο

1. Βασικά Στοιχεία της Δοκιμής Λογισμικού και της Διασφάλισης Ποιότητας

1.1 Ορισμοί

Στην περίπτωση που η συμπεριφορά ενός προϊόντος λογισμικού δεν είναι η επιθυμητή, δηλαδή όταν δεν πληρούνται οι απαιτήσεις (requirements) και οι προδιαγραφές (specifications), τότε έχουμε ένα σφάλμα (error) ή μια αποτυχία (failure). Το σφάλμα αυτό δημιουργείται από ένα ελάττωμα στο λογισμικό (defect), το οποίο συνήθως συμβαίνει είτε από λάθος του προγραμματιστή, όπως στην περίπτωση λάθος εντολών, ξεχασμένου κώδικα κτλ. είτε από λάθος του συστήματος.

Οι δοκιμές λογισμικού δηλαδή το software testing είναι η διαδικασία εκτέλεσης του λογισμικού χρησιμοποιώντας ένα επιλεγμένο σύνολο από δεδομένα με σκοπό την ανίχνευση των ελαττωμάτων, την ανάλυση του κώδικα για την πρόληψη ελαττωμάτων και την αξιολόγηση της ποιότητας του λογισμικού (software quality) με σκοπό την διασφάλιση της ποιότητας (quality assurance) του software προϊόντος-συστήματος (system under test).

Συνεπώς βασικός στόχος των δοκιμών λογισμικού (software testing) είναι επίσης η βελτίωση της ποιότητας του λογισμικού (software quality), όχι μόνο ανιχνεύοντας τα σφάλματα και ελαχιστοποιώντας τα ελαττώματα του, αλλά αποκτώντας γνώση πάνω στα ποιοτικά χαρακτηριστικά του [1]. Όπως, το σύνολο γνωρισμάτων και χαρακτηριστικών ενός προϊόντος που εξυπηρετούν δεδομένες ανάγκες και προδιαγραφές, ο βαθμός που το λογισμικό κατέχει ένα επιθυμητό συνδυασμό από ιδιότητες, ο βαθμός στον οποίο ο πελάτης ή ο χρήστης αντιλαμβάνεται ότι το προϊόν ικανοποιεί τις προσδοκίες του και τα σύνθετα χαρακτηριστικά του λογισμικού τα οποία καθορίζουν το βαθμό που το λογισμικό υπό χρήση ικανοποιεί τις προσδοκίες του πελάτη.

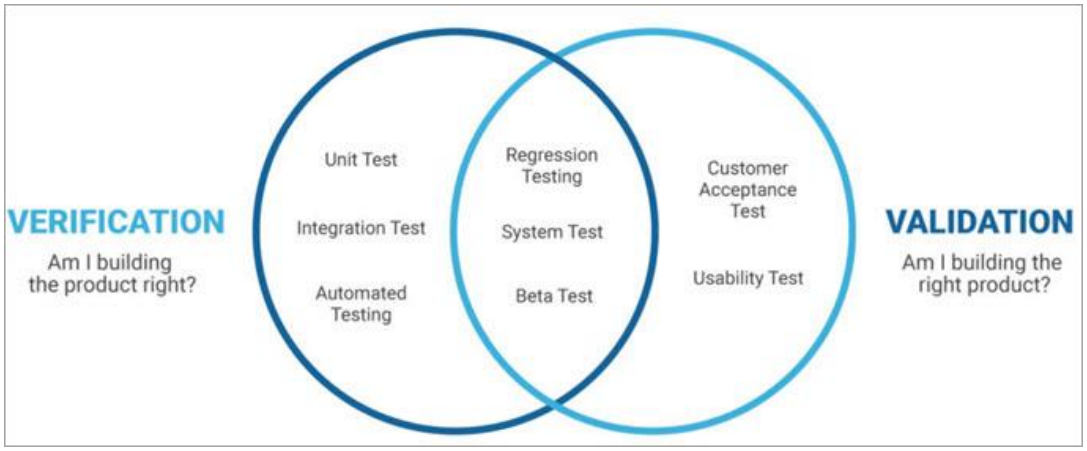
Οι περιπτώσεις δοκιμών (test cases) ορίζονται ως ένα σύνολο δεδομένων εισόδου (input), προϋποθέσεων, αναμενόμενων αποτελεσμάτων, δεδομένων εξόδου (output) και κριτηρίων εξόδου (exit criteria) πάνω στο αντικείμενο υπό έλεγχο λογισμικού. Μπορούν να διαχωριστούν σε δύο κατηγορίες σε αυτές που ελέγχουν την αναμενόμενη συμπεριφορά του λογισμικού, έγκυρες (expected test cases), και σε αυτές που εξετάζουν τη συμπεριφορά του αντικειμένου για μη αναμενόμενα δεδομένα εισόδου, μη έγκυρες (unexpected test case) περιπτώσεις δοκιμής.

Εκτέλεση δοκιμής (test run) ονομάζεται η εκτέλεση ενός ή περισσότερων περιπτώσεων δοκιμής, ενώ πακέτο σεναρίων δοκιμής ή απλά σενάρια δοκιμής (test suite) είναι ένα σύνολο περιπτώσεων δοκιμής, όπου το εξαγόμενο αποτέλεσμα δοκιμής (output) αποτελεί το δεδομένο εισόδου (input) της επόμενης περίπτωσης δοκιμής. Η διαδικασία εκτέλεσης ενός συγκεκριμένου συνόλου περιπτώσεων δοκιμής πάνω στο αντικείμενο υπό δοκιμή με σκοπό την ανίχνευση σφαλμάτων, ονομάζεται δοκιμή (test). Η οργάνωση, ο σχεδιασμός, η υλοποίηση και η ανάλυση μιας δοκιμής είναι βασικό κομμάτι της δοκιμής λογισμού (software testing) [2]. Ο εντοπισμός κα η διόρθωση ελαττωμάτων (debugging) συνιστά δουλειά του προγραμματιστή και είναι κάτι εντελώς διαφορετικό από την δοκιμή λογισμικού, γι’αυτό οι δύο αυτές έννοιες δεν πρέπει να συγχέονται.

Η δοκιμή λογισμικού (software testing) μπορεί να εξετάζει λοιπόν τη συμπεριφορά εφαρμογών μέσα από επικύρωση (validation) και επαλήθευση (verification) [3]. Ενώ η επαλήθευση πραγματοποιείται ενώ το λογισμικό είναι ακόμη υπό ανάπτυξη, η επικύρωση εκτελείται με την ολοκλήρωση μιας δεδομένης ενότητας ή ακόμη και την ολοκλήρωση ολόκληρου του λογισμικού.

Η επικύρωση (validation) είναι μια δραστηριότητα που διασφαλίζει την κάλυψη των πραγματικών αναγκών και προσδοκιών ενός ενδιαφερόμενου μέρους του τελικού προϊόντος-λογισμικού. Τα validation tests έχουν να κάνουν με το κατά πόσο σωστά έχουν διευθετηθεί οι προδιαγραφές του λογισμικού.
Η επαλήθευση (verification) αναφέρεται στον έλεγχο του λογισμικού που βρίσκεται ακόμη υπό ανάπτυξη για να διασφαλιστεί ότι συμμορφώνεται με τις προδιαγραφές (specifications) και απαιτήσεις (requirements) που έχουν οριστεί για το συγκεκριμένο λογισμικό [14]. Τα verification tests έχουν να κάνουν με το κατά πόσο το λογισμικό

συναντά τις προδιαγραφές και απαιτήσεις. Αυτοί οι έλεγχοι θα μπορούσαν να είναι κάτι τόσο απλό όσο η ανάγνωση των προδιαγραφών και η σύγκρισή τους με τη λογική του κώδικα για να επιβεβαιωθεί ότι ευθυγραμμίζονται. Η διαδικασία επαλήθευσης περιλαμβάνει δραστηριότητες όπως αναθεωρήσεις κώδικα (code reviews), περιηγήσεις (walkthroughs), επιθεωρήσεις (inspections) αλλά ελάχιστες, εάν υπάρχουν, πραγματικές δοκιμές (testing).



Εικόνα 1 Διαφορές μεταξύ των validation και verification tests.

Βασικός σκοπός της δοκιμής λογισμικού (software testing) είναι η εύρεση ελαττωματικών (defected) λειτουργιών του συστήματος το οποίο υπόκειται σε δοκιμή (testing). Η εύρεση των λεγόμενων αυτών ελαττωμάτων (defects ή αλλιώς bugs) έρχεται μέσα από την ανάλυση, σχεδιασμό, υλοποίηση και εκτέλεση σεναρίων που αντιστοιχούν σε επιχειρηματικές ροές (business flows) του συστήματος. Καθώς κάθε σύστημα έχει ορισμένα στάδια προόδου της κατάστασης (state progressions), η μετάβαση από κάποιες καταστάσεις σε κάποιες άλλες άλλοτε είναι επιτρεπτή και άλλοτε όχι (state transition table), σύμφωνα με τις προκαθορισμένες απαιτήσεις (requirements). Μέσα λοιπόν από τη δημιουργία θετικών (positive) και αρνητικών (negative) σεναρίων μπορούμε να επιβεβαιώνουμε πως μία εφαρμογή, ένα λογισμικό έχει την εκάστοτε λειτουργικότητα όταν και μόνο όταν αυτή είναι θεμιτή. Για την δοκιμή λογισμικού είναι απαραίτητη η ανάπτυξη ενός συνόλου δοκιμών (tests) τα οποία θα διαχωρίζονται σε θετικές δοκιμές (positive tests) και αρνητικές δοκιμές (negative test) ανάλογα με το επιθυμητό αποτέλεσμα της συμπεριφοράς της εφαρμογής σε κάθε δοκιμή [4].

1.2 Σημαντικότητα Δοκιμής Λογισμικού

Χρησιμοποιώντας την διαδικασία της δοκιμής λογισμικού ένα σύνολο από οφέλη προκύπτουν και αναφέρονται στη συνέχεια:

- Οικονομικά αποδοτικό: Στις περιπτώσεις όπου βρίσκονται ελαττώματα (defects - bugs) σχετικά νωρίς εντός του κύκλου ζωής ανάπτυξης λογισμικού (software development lifecycle), το να διορθωθούν έγκαιρα κοστίζει λιγότερο από ότι εάν τα ελαττώματα (defects) αυτά ανακαλυφθούν αργότερα.
- Ασφάλεια: Ένα σύστημα που υπόκειται σε δοκιμή (testing) είναι λιγότερο ευάλωτο (vulnerable). Επίσης, βοηθά στην απομάκρυνση των κινδύνων και των προβλημάτων σε πρώιμο στάδιο.
- Αποφέρει ποιότητα προϊόντος και ικανοποίηση πελατών: Επαληθεύεται η συμβατότητα με διάφορα λειτουργικά συστήματα (operating systems) και συσκευές, όπως ακριβώς ορίζουν τα προαπαιτούμενα. Καλύτερη εμπειρία χρηστών (UI/UX), πιο δυνατή φήμη του προϊόντος.
- Ενίσχυση της διαδικασίας ανάπτυξης (development process): Χρησιμοποιώντας δοκιμή λογισμικού, βρίσκονται λάθη (errors) τα οποία δεν πιάνονται κατά το στάδιο της δοκιμής (testing) από developer (unit testing). Επίσης εντοπίζονται ελαττώματα (defects) τα οποία είναι “ακραίες” περιπτώσεις (edge cases) και θα μπορούσαν να ανακαλυφθούν μόνο σε περίπτωση που η εφαρμογή- το λογισμικό έχει βγει στην παραγωγή.
- Ενισχύει την επίδοση (performance) της εφαρμογής: Η δοκιμασία λογισμικού (software testing) μπορεί να βοηθήσει στην αποδοτική χρήση μιας εφαρμογής (π.χ. από πολλούς χρήστες ταυτόχρονα), κάτι που απαιτείται ειδικά σε κρίσιμες ημέρες/περιόδους εκτεταμένης χρήσης της εφαρμογής αυτής [1].

1.3 Επιπτώσεις Έλλειψης Δοκιμής Λογισμικού

Οι επιπτώσεις από παράλειψη της διαδικασίας της δοκιμής λογισμικού μπορεί να οδηγήσουν σε:

- Αύξηση του κόστους παραγωγής του λογισμικού
- Κίνδυνο διακοπής λειτουργίας του λογισμικού σε απρόβλεπτο σημείο και με απρόβλεπτες συνέπειες

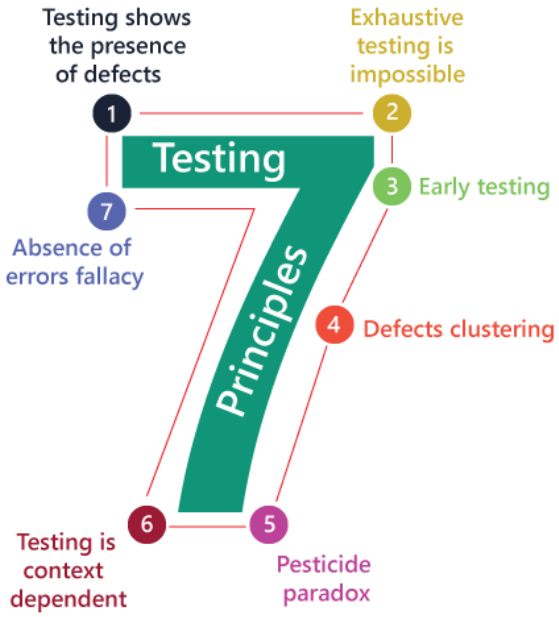
- Μείωση της αξιοπιστίας του λογισμικού
- Μείωση της αξιοπιστίας της εταιρείας [1].

1.4 Βασικές Αρχές Δοκιμής Λογισμικού

Οι βασικές αρχές της δοκιμής λογισμικού (software testing) είναι επτά:

- Η δοκιμή δείχνει την παρουσία ελαττωμάτων (defects) λογισμικού και όχι την απουσία τους. Μπορεί να καταδείξει ελαττώματα, αλλά δε μπορεί να αποδείξει ότι δεν υπάρχει κανένα άλλο ελάττωμα [33]. Δηλαδή, η δοκιμή μειώνει την πιθανότητα ύπαρξης κρυφών ελαττωμάτων στο λογισμικό, αλλά ακόμα και αν δεν βρεθούν ελαττώματα, αυτό δε σημαίνει ότι δεν υπάρχουν.
- Η διεξοδική δοκιμή είναι αδύνατη. Η δοκιμή όλων (όλοι οι συνδυασμοί εισροών και προϋποθέσεων) δεν είναι εφικτή. Πρέπει να χρησιμοποιείται ανάλυση κινδύνου, τεχνικές δοκιμών και προτεραιοτήτων για την σωστή εστίαση των δοκιμών.
- Οι διαδικασίες δοκιμής πρέπει να ξεκινάνε το νωρίτερα δυνατό [34]. Οι δραστηριότητες της δοκιμής πρέπει να ενσωματώνονται στον κύκλο ζωής του λογισμικού (software development lifecycle).
- Τα ελαττώματα (defects) λογισμικού συγκεντρώνονται μαζεμένα (cluster). Η πιθανότητα ύπαρξης πρόσθετων σφαλμάτων σε ένα κομμάτι λογισμικού είναι συνήθως ανάλογη του συνόλου των σφαλμάτων που έχουν ανιχνευθεί εκεί [35].
- Αν οι ίδιες περιπτώσεις δοκιμής τρέξουν επανειλημμένα στο αντικείμενο υπό δοκιμή, κάποια στιγμή δε βρίσκονται νέα ελαττώματα (παράδοξο του παρασιτοκτόνου - pesticide paradox). Για το λόγο αυτό, οι περιπτώσεις δοκιμής πρέπει να επανεξετάζονται και να διορθώνονται συστηματικά, και νέες περιπτώσεις δοκιμής πρέπει να γράφονται για τον δοκιμή διαφορετικών κομματιών του κώδικα ή του συστήματος.
- Η δοκιμή λογισμικού εξαρτάται από το περιεχόμενο και το περιβάλλον. Για κάθε προϊόν λογισμικού διαφορετικού περιεχομένου ακολουθείται διαφορετική δοκιμή. Για παράδειγμα, το λογισμικό που χρησιμοποιείται για μία τράπεζα μιας χώρας δοκιμάζεται διαφορετικά από μια σελίδα ηλεκτρονικού εμπορίου.
- Η «απουσία σφαλμάτων» είναι πλάνη. Είναι μια πλάνη (δηλαδή, μια εσφαλμένη πεποίθηση) να περιμένουμε ότι μόνο η εύρεση και η διόρθωση μεγάλου αριθμού

ελαττωμάτων θα εξασφαλίσει την επιτυχία ενός συστήματος. Η εφαρμογή των δοκιμών σε μια εφαρμογή απλά μειώνει την πιθανότητα ύπαρξης ελαττωμάτων δεν την εξαλείφει. Σημασία λοιπόν δεν έχει η εξάλειψη των ελαττωμάτων αλλά η εφαρμογή να καλύπτει τις απαιτήσεις του πελάτη [1].



Εικόνα 2 Οι επτά βασικές αρχές της δοκιμής λογισμικού.

1.5 Προσεγγίσεις Δοκιμής Λογισμικού

Η αρχιτεκτονική τριών επιπέδων (3-Tier Architecture) είναι μια αρχιτεκτονική Client-Server η οποία έχει ως χαρακτηριστικό της την διαμόρφωση των μερών εκείνων που αποτελούν την συνολική εφαρμογή σε επίπεδα (layers ή αλλιώς tiers).
Τα επίπεδα αυτά είναι τα εξής:

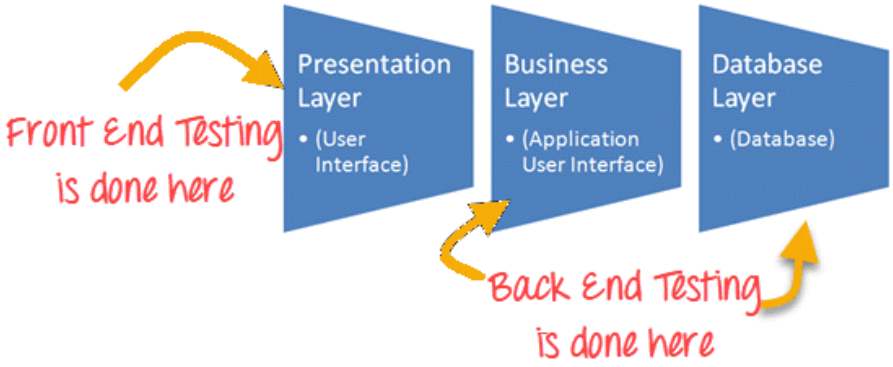
- 1ο Επίπεδο – Data Layer – Database Server
- 2ο Επίπεδο – Business Logic Layer – Application Server
- 3ο Επίπεδο – Presentation Layer – User Interface

Μπορούμε να έχουμε δύο κατηγορίες προσεγγίσεων της δοκιμής λογισμικού (software testing) :

- Front end testing: είναι μια τεχνική δοκιμής λογισμικού στην οποία ελέγχεται το γραφικό περιβάλλον διεπαφής χρήστη (graphical user interface - GUI), η

λειτουργικότητα και η χρηστικότητα ενός λογισμικού. Ο στόχος της Front end δοκιμής λογισμικού είναι η δοκιμή των συνολικών λειτουργιών για να διασφαλιστεί ότι το επίπεδο παρουσίασης (presentation layer) ενός λογισμικού είναι απαλλαγμένο από ελαττώματα (defect free).Για παράδειγμα, η Front end δοκιμή λογισμικού ελέγχει εάν τα πεδία εισαγωγής δέχονται τους σωστούς χαρακτήρες, εάν οι φόρμες υποβάλλονται μόνο αφού συμπληρωθούν τα απαιτούμενα πεδία, εάν η πλοήγηση είναι αρκετά εύκολη, εάν η σελίδα φορτώνει αρκετά γρήγορα κ.λπ.

- Back end testing: η Back end δοκιμή λογισμικού είναι μια τεχνική δοκιμής (testing) για το επίπεδο εφαρμογής (application layer) και το επίπεδο βάσης δεδομένων (database layer) της αρχιτεκτονικής τριών επιπέδων σε ένα λογισμικό. Ουσιαστικά η Back end δοκιμή λογισμικού πραγματοποιείται στο application interface (API) και στη βάση δεδομένων (database).



Εικόνα 3 Frontend Testing Vs. Backend Testing.

Κύριες διαφορές: το Front end testing πραγματοποιείται στο presentation layer και το γραφικό περιβάλλον διεπαφής χρήστη (GUI) είναι απαραίτητο [5] ενώ στο Back end testing δεν είναι καθώς πραγματοποιείται στο application layer και στο database layer. Στο Front end testing δεν χρειάζεται να αποθηκευτούν πληροφορίες σε μια βάση δεδομένων, ενώ στο Back end testing οι πληροφορίες αποθηκεύονται στη βάση δεδομένων.

1.6 Αυτοματοποιημένη και Χειροκίνητη Δοκιμή Λογισμικού

Η δοκιμή από άκρο σε άκρο (end to end testing) είναι η διαδικασία δοκιμής ενός λογισμικού από την αρχή ως το τέλος. Για παράδειγμα σε μια εφαρμογή ιστού (web application), θα χρειαστεί να γίνει εκκίνηση ενός προγράμματος περιήγησης, έπειτα μετάβαση στη σωστή διεύθυνση URL, και να χρησιμοποιηθεί η εφαρμογή όπως προορίζεται να χρησιμοποιηθεί από τους χρήστες και να γίνει επαλήθευση της συμπεριφοράς της. Υπάρχουν δύο διαφορετικοί τρόποι εκτέλεσης δοκιμών λογισμικού, ο χειροκίνητος και ο αυτοματοποιημένος [31]. Η χειροκίνητη είναι μια διαδικασία κατά την οποία οι υπεύθυνοι δοκιμών (testers) εξετάζουν διαφορετικά χαρακτηριστικά μιας εφαρμογής. Οι υπεύθυνοι δοκιμών (testers) εκτελούν διαφορετικές περιπτώσεις δοκιμών χωρίς τη χρήση αυτοματοποιημένων εργαλείων. Τέλος, δημιουργούν μια έκθεση δοκιμής. Στις αυτοματοποιημένες δοκιμές, οι υπεύθυνοι δοκιμών (testers) χρησιμοποιούν σενάρια για δοκιμές (αυτοματοποιώντας έτσι τη διαδικασία). Οι δοκιμές με προκαθορισμένο σενάριο εκτελούνται αυτόματα για να συγκρίνουν πραγματικά και αναμενόμενα αποτελέσματα. Με την αυτοματοποίηση αυτών, όταν η συνεχής ανθρώπινη παρέμβαση δεν είναι απαραίτητη και η εκτέλεση επαναλαμβανόμενων σεναρίων δεν απαιτούν μεγάλη προσπάθεια.

Εικόνα 4 Η ποιότητα του λογισμικού εξαρτάται από τη αυτοματοποιημένη και από τη χειροκίνητη δοκιμή λογισμικού.

Η αυτοματοποιημένη δοκιμή (automation testing) είναι μια τεχνική δοκιμής λογισμικού που εκτελείται χρησιμοποιώντας ειδικά εργαλεία λογισμικού αυτοματοποιημένης δοκιμής για την εκτέλεση μιας σειράς σεναρίων. Το λογισμικό αυτοματοποιημένης δοκιμής μπορεί επίσης να εισαγάγει δεδομένα δοκιμών στο υπό δοκιμή σύστημα, να συγκρίνει τα αναμενόμενα και πραγματικά αποτελέσματα και να δημιουργήσει λεπτομερείς αναφορές δοκιμών με σκοπό τη βελτίωση της ποιότητας του λογισμικού. Η αυτοματοποίηση δοκιμών λογισμικού απαιτεί σημαντικές επενδύσεις χρημάτων και πόρων. Μια δοκιμή πρέπει να πληρεί ορισμένα κριτήρια για να αυτοματοποιηθεί, διαφορετικά, ενδέχεται να κοστίσει περισσότερο από ό,τι εξοικονομεί. Εξάλλου, ένας κύριος στόχος του αυτοματισμού είναι να εξοικονομήσει χρόνο, προσπάθεια και χρήμα.

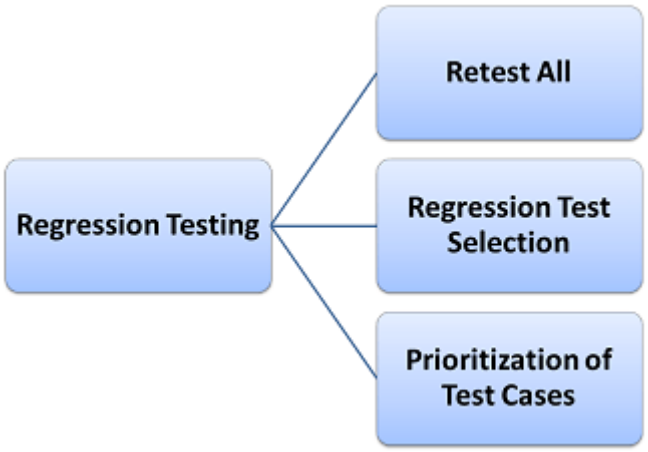
Οποιοδήποτε λογισμικό, πρέπει να δοκιμάζεται ώστε να είμαστε σίγουροι ότι θα λειτουργήσει με τον τρόπο που προορίζεται. Η δοκιμή λογισμικού πρέπει να είναι αποτελεσματική στην εύρεση τυχόν σφαλμάτων που υπάρχουν, εκτελώντας τις δοκιμές όσο πιο γρήγορα και φθηνά γίνεται. Η αυτοματοποίηση, μπορεί να μειώσει σημαντικά την απαιτούμενη προσπάθεια για επαρκή δοκιμή, και αυξάνει σημαντικά τις δοκιμές που μπορούν να γίνουν σε περιορισμένο χρόνο. Οι δοκιμές μπορούν να εξαχθούν σε λίγα λεπτά εν αντιθέσει των χειροκίνητων δοκιμών που μπορεί να διαρκέσουν ώρες. Οι αυτοματοποιημένες δοκιμές έχουν οικονομικά οφέλη, βοηθούν στην παραγωγή λογισμικού καλύτερης ποιότητας, πιο γρήγορα, μειώνουν την ανθρώπινη προσπάθεια που απαιτείται, και επιτρέπουν να δοκιμάζονται πλήρως οι μικρότερες αλλαγές συντήρησης του λογισμικού με ελάχιστη προσπάθεια.

Στη συνέχεια παρουσιάζονται επιπλέον οφέλη της αυτοματοποιημένης δοκιμής:

- Εκτέλεση περισσότερων δοκιμών. Ένα σαφές όφελος του αυτοματισμού είναι η ικανότητα εκτέλεσης περισσότερων δοκιμών σε λιγότερο χρόνο. Αυτό θα οδηγήσει σε μεγαλύτερη εμπιστοσύνη στο σύστημα.
- Εκτέλεση υπαρχουσών δοκιμών (regression) σε μια νέα έκδοση ενός λογισμικού. Η δοκιμή παλινδρόμησης (regression testing) καθίσταται απαραίτητη και μπορεί να πραγματοποιηθεί χρησιμοποιώντας τις ακόλουθες τεχνικές:
    - Επανάληψη όλων των δοκιμών, όλες οι δοκιμές στην υπάρχουσα σουίτα δοκιμής θα πρέπει να εκτελεστούν ξανά.

- Επιλογή δοκιμής παλινδρόμησης, είναι μια τεχνική στην οποία εκτελούνται ορισμένες επιλεγμένες περιπτώσεις δοκιμής από τη σουίτα δοκιμών για να ελεγχθεί εάν ο τροποποιημένος κώδικας επηρεάζει την εφαρμογή λογισμικού ή όχι.
- Προτεραιοποίηση των Δοκιμαστικών Περιπτώσεων, δίνεται προτεραιότητα στις δοκιμαστικές περιπτώσεις ανάλογα με τον επιχειρηματικό αντίκτυπο, τις κρίσιμες και τις συχνά χρησιμοποιούμενες λειτουργίες (functionalities) της εφαρμογής που δοκιμάζεται.

Η προσπάθεια που απαιτείται για την εκτέλεση ενός συνόλου regression δοκιμών συνήθως είναι ελάχιστη. Δεδομένου ότι οι δοκιμές υπάρχουν ήδη και έχουν αυτοματοποιηθεί για εκτέλεση σε παλαιότερη έκδοση του λογισμικού, θα πρέπει να είναι δυνατή η επιλογή των δοκιμών και η έναρξη της εκτέλεση τους με λίγα λεπτά χειροκίνητης προσπάθειας.



Εικόνα 5 Τεχνικές του regression testing.

- Εκτέλεση δοκιμών που θα ήταν δύσκολο ή αδύνατο να πραγματοποιηθούν χειροκίνητα. Κατά τη μη αυτόματη δοκιμή, τα αναμενόμενα αποτελέσματα συνήθως περιλαμβάνουν τα προφανή σενάρια που είναι ορατά στον υπεύθυνο δοκιμών. Ωστόσο, υπάρχουν σενάρια που πρέπει να δοκιμαστούν και δεν είναι εύκολο να επαληθευτούν χειροκίνητα.
- Καλύτερη χρήση των πόρων. Η αυτοματοποίηση χειροκίνητων και βαρετών εργασιών, όπως η επανειλημμένη εισαγωγή [7] των ίδιων εισόδων δοκιμής, δίνει μεγαλύτερη ακρίβεια και δεν χρειάζεται οι υπεύθυνοι των δοκιμών να καταβάλουν μεγάλη προσπάθεια.

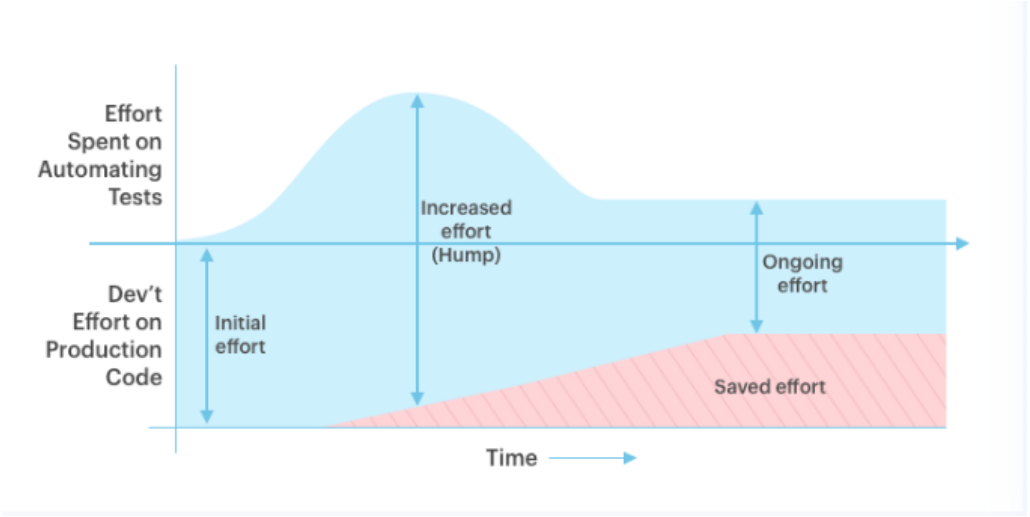
- Επαναχρησιμοποίηση των δοκιμών.
- Συνοχή και επαναληψιμότητα των δοκιμών. Οι ίδιες δοκιμές μπορούν να εκτελεστούν σε διαφορετικές ρυθμίσεις παραμέτρων υλικού (hardware configurations), χρησιμοποιώντας διαφορετικά λειτουργικά συστήματα ή χρησιμοποιώντας διαφορετικές βάσεις δεδομένων. Αυτό παρέχει συνοχή ποιότητας μεταξύ πλατφορμών για προϊόντα πολλαπλών πλατφορμών που είναι σχεδόν αδύνατο να επιτευχθεί με χειροκίνητη δοκιμή.
- Αυξημένη εμπιστοσύνη. Γνωρίζοντας ότι ένα εκτεταμένο σύνολο αυτοματοποιημένων δοκιμών έχει τρέξει με επιτυχία, μπορεί να υπάρχει μεγαλύτερη εμπιστοσύνη ότι δεν θα υπάρξουν τυχόν σφάλματα στο λογισμικό που κυκλοφόρησε (released software) δηλαδή που είναι έτοιμο προς χρήση από τον τελικό πελάτη.
- Εξοικονόμηση χρόνου. Μόλις ένα σύνολο δοκιμών αυτοματοποιηθεί, μπορεί να επαναληφθεί γρηγορότερα απ' ότι ένα χειροκίνητο σύνολο δοκιμών, έτσι ο χρόνος δοκιμής του συστήματος μειώνεται.

Συνεπώς, πιο εμπεριστατωμένες δοκιμές μπορούν να επιτευχθούν με λιγότερη προσπάθεια, και μεγαλύτερη ποιότητα και παραγωγικότητα [8].

Στη συνέχεια, οι στρατηγικές για την επιτυχία της αυτοματοποιημένης δοκιμής (testing) απεικονίζονται παρακάτω :

Εικόνα 6 Οι στρατηγικές για την επιτυχία της αυτοματοποιημένης δοκιμής.

Στην παρακάτω εικόνα απεικονίζεται η γραφική παράσταση που δείχνει τον αρχικό χρόνο και τις προσπάθειες που απαιτούνται για την εκμάθηση των βέλτιστων πρακτικών αυτοματισμού δοκιμών. Μπορούμε να δούμε πώς εξοικονομεί τη συνολική προσπάθεια και τον χρόνο που δαπανάται.



Εικόνα 7 Γραφική παράσταση για την εκμάθηση των βέλτιστων πρακτικών αυτοματισμού δοκιμών στην πάροδο του χρόνου.

Η κλιμάκωση του αυτοματισμού δοκιμών και η διαχείρισή του με την πάροδο του χρόνου παραμένει μια πρόκληση για τις ομάδες ανάπτυξης λογισμικού (δηλαδή προγραμματιστώ, υπεύθυνων δοκιμών κτλ.) [20]. Οι ομάδες ανάπτυξης (development team) μπορούν να χρησιμοποιήσουν τη μηχανική μάθηση (machine learning- ML) τόσο στις φάσεις σύνταξης και εκτέλεσης αυτοματισμού δοκιμών της εφαρμογής – λογισμικού, καθώς και στην ανάλυση δοκιμής μετά την εκτέλεση που περιλαμβάνει την εξέταση τάσεων, προτύπων, μοτίβων και επιπτώσεων στην επιχείρηση. Είναι σημαντικό να κατανοήσουμε τις βασικές αιτίες του γιατί η αυτοματοποίηση δοκιμών είναι τόσο ασταθής όταν δεν χρησιμοποιούνται τεχνολογίες μηχανικής εκμάθησης (machine learning- ML) [21] [22]. Η σταθερότητα δοκιμών τόσο των εφαρμογών για κινητά όσο και των εφαρμογών ιστού συχνά επηρεάζεται από στοιχεία εντός τους που είτε είναι εξ ορισμού δυναμικά είτε έχουν αλλάξει από τους προγραμματιστές. Η σταθερότητα των δοκιμών μπορεί επίσης να επηρεαστεί όταν γίνονται αλλαγές στα δεδομένα από τα οποία εξαρτάται η δοκιμή ή συνηθέστερα γίνονται αλλαγές απευθείας στην εφαρμογή (δηλαδή προστίθενται νέες οθόνες, κουμπιά, ροές χρήστη ή είσοδοι χρήστη). Τα σενάρια δοκιμής που δεν είναι μηχανικής εκμάθησης (machine learning- ML) είναι στατικά, επομένως δεν μπορούν

να προσαρμοστούν αυτόματα και να ξεπεράσουν τις παραπάνω αλλαγές [23]. Αυτή η αδυναμία προσαρμογής έχει ως αποτέλεσμα αστοχίες δοκιμών, εύθραυστες δοκιμές, αστοχίες κατασκευής, ασυνεπή δεδομένα δοκιμών και πολλά άλλα. Παρουσιάζονται στη συνέχεια μερικοί συγκεκριμένοι τρόποι με τους οποίους η μηχανική εκμάθηση μπορεί να είναι πολύτιμη για τις ομάδες τις ομάδες ανάπτυξης λογισμικού. Σε μερικές περιπτώσεις απαιτούνται εξαιρετικά υψηλές ποσότητες δεδομένων δοκιμής [24].

Οι οργανισμοί που εφαρμόζουν συνεχείς δοκιμές σύμφωνα με την agile μεθοδολογία, εκτελούν μια μεγάλη ποικιλία τύπων δοκιμών πολλές φορές την ημέρα. Αυτό περιλαμβάνει unit tests, API tests, functional tests, accessibility tests, integration tests και άλλους τύπους δοκιμών. Με κάθε εκτέλεση δοκιμής, ο όγκος των δεδομένων δοκιμής που δημιουργείται αυξάνεται σημαντικά, καθιστώντας τη διαδικασία λήψης αποφάσεων πιο δύσκολη. Τα βασικά ζητήματα του προϊόντος βρίσκονται μέσω της οπτικοποίησης των πιο ασταθών περιπτώσεων δοκιμών και άλλων τομέων στους οποίους πρέπει να γίνεται η περισσότερη συγκέντρωση, η μηχανική μάθηση (machine learning- ML) στην αναφορά και ανάλυση δοκιμών διευκολύνει τη ζωή των στελεχών [25] [26]. Με τα συστήματα AI/ML, τα στελέχη θα πρέπει να μπορούν να τεμαχίζουν καλύτερα τα δεδομένα δοκιμών, να κατανοούν τις τάσεις και τα μοτίβα, να ποσοτικοποιούν τους επιχειρηματικούς κινδύνους και να λαμβάνουν αποφάσεις πιο γρήγορα και συνεχώς. Για παράδειγμα, μαθαίνοντας ποιες εργασίες CI είναι πιο πολύτιμες ή χρονοβόρες ή ποιες πλατφόρμες υπό δοκιμή (κινητά, web, επιτραπέζιοι υπολογιστές) είναι πιο ελαττωματικές από άλλες. Με τα συστήματα AI/ML δηλαδή τεχνητή νοημοσύνη (artificial -intelligence -AI) ή /και μηχανική μάθηση (machine learning- ML), τα στελέχη θα πρέπει να μπορούν να κατηγοριοποιούν καλύτερα τα δεδομένα δοκιμών, να κατανοούν τις τάσεις και τα μοτίβα, να ποσοτικοποιούν τους επιχειρηματικούς κινδύνους και να λαμβάνουν αποφάσεις πιο γρήγορα και συνεχώς. Χωρίς τη βοήθεια της τεχνητής νοημοσύνης ή της μηχανικής μάθησης [27], η εργασία είναι επιρρεπής σε σφάλματα, χειροκίνητη και μερικές φορές αδύνατη. Με το AI/ML οι επαγγελματίες της ανάλυσης δεδομένων δοκιμών έχουν την ευκαιρία να προσθέσουν χαρακτηριστικά γύρω από την δοκιμαστική ανάλυση επιπτώσεων, τα κενά ασφαλείας, τα ελαττώματα ειδικά για την πλατφόρμα, τη δοκιμή σταθερότητας περιβάλλοντος, τα επαναλαμβανόμενα μοτίβα σε αποτυχίες δοκιμών, και την ευθραυστότητα στον εντοπισμό στοιχείων εφαρμογής.

Η χειροκίνητη δοκιμή (manual testing) είναι ένας τύπος δοκιμής λογισμικού κατά τον οποίο οι δοκιμές εκτελούνται χειροκίνητα από έναν υπεύθυνο δοκιμών (tester) χωρίς τη χρήση αυτοματοποιημένων εργαλείων. Ο σκοπός της χειροκίνητης δοκιμής είναι να εντοπίσει σφάλματα, ζητήματα και ελαττώματα στην εφαρμογή λογισμικού [28]. Η διαδικασία συγκρίνει τη συμπεριφορά μιας εφαρμογής λογισμικού (ή ενός από τα συστατικά ή χαρακτηριστικά της) με την αναμενόμενη συμπεριφορά που ορίστηκε στις αρχικές φάσεις του κύκλου ζωής ανάπτυξης λογισμικού. Η χειροκίνητη δοκιμή λογισμικού είναι η πιο πρωτόγονη τεχνική όλων των τύπων δοκιμών και βοηθά στην εύρεση κρίσιμων σφαλμάτων στην εφαρμογή λογισμικού. Κάθε νέα εφαρμογή πρέπει να δοκιμαστεί χειροκίνητα προτού αυτοματοποιηθεί η δοκιμή της.

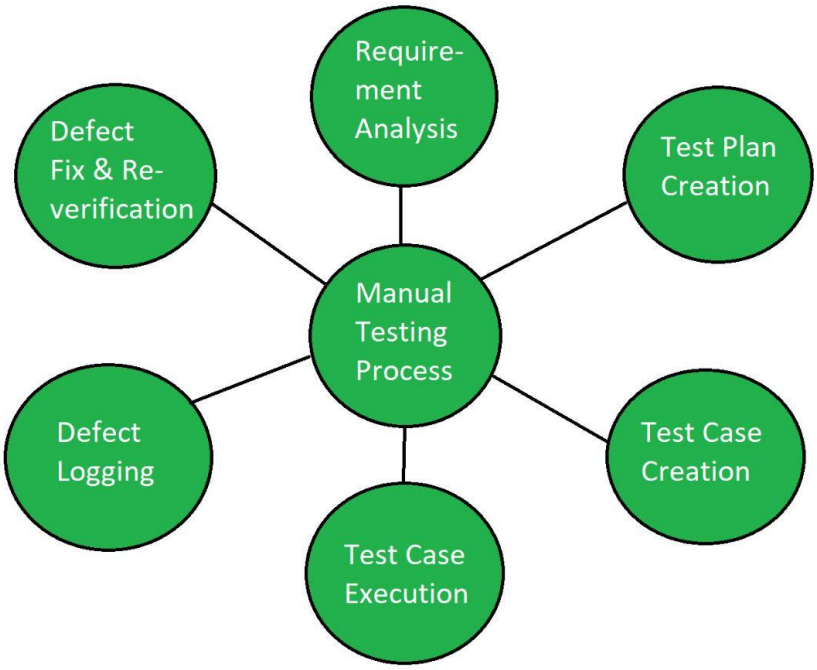
Η χειροκίνητη δοκιμή απαιτεί περισσότερη προσπάθεια αλλά είναι απαραίτητη για τον έλεγχο της σκοπιμότητας αυτοματισμού. Για τον χειροκίνητο έλεγχο δεν απαιτείται γνώση οποιουδήποτε εργαλείου δοκιμής. Οι χειροκίνητες δοκιμές παραμένουν ουσιαστικό συστατικό των δοκιμών για τους εξής λόγους:

- Τα σφάλματα εντοπίζονται στα λιγότερο εμφανή μέρη. Οι υπεύθυνοι δοκιμών (testers) ενθαρρύνονται να ακολουθήσουν την περιέργεια και την πρωτοβουλία τους και να εργαστούν σε αυθόρμητες διαδρομές εξερευνώντας περιοχές που δεν δοκιμάζονται συνήθως.
- Ευχρηστία και δοκιμές ελέγχου χρηστικότητας, είναι ο τρόπος με τον οποίο οι υπεύθυνοι δοκιμών (testers) ανακαλύπτουν εάν ένας ιστότοπος ή άλλο λογισμικό συμπεριφέρεται όπως αναμένεται από τον σχεδιαστή όταν βρίσκεται στα χέρια ενός πραγματικού χρήστη.
- Οικονομικότερη τεχνική δοκιμής για σύντομα έργα. Για σύντομα έργα, οι επιπλέον ώρες και οι διαδικασίες που απαιτούνται για τη δημιουργία μιας ακολουθίας αυτοματοποιημένων δοκιμών, πριν από τη δοκιμή του ίδιου του σεναρίου, μπορεί να αφαιρέσουν κάθε πιθανή εξοικονόμηση. Μόλις αυτοματοποιηθεί και τεθεί σε λειτουργία μια δοκιμαστική εκτέλεση, θα υπάρξει μείωση του χρόνου που απαιτείται για τον κύκλο ανάπτυξης, αλλά εάν μια διεπαφή, για παράδειγμα, λάβει ενδελεχή επανασχεδιασμό, αυτές οι αλλαγές μπορεί να προκαλέσουν προβλήματα και απαιτούνται πολλές επιπλέον ώρες για την ενημέρωση των αυτοματοποιημένων δοκιμών [9]. Επίσης, εάν οι αλλαγές αποτελούν μέρος μιας τρέχουσας διαδικασίας, που σημαίνει ότι η διεπαφή

υποβάλλεται σε συνεχή αναθεώρηση με σημαντικές βελτιώσεις, ενδέχεται να μην είναι πρακτικό να αυτοματοποιήσουμε τις δοκιμές στο γραφικό περιβάλλον χρήστη έως ότου η κατασκευή είναι πιο σταθερή. Μερικές φορές, όμως, οι αυτοματοποιημένες δοκιμές περιέχουν σφάλματα, γεγονός που καθιστά τη χειροκίνητη δοκιμή απαραίτητη για να μάθουμε πού βρίσκονται τα προβλήματα, ώστε να διορθωθεί η κατάσταση. Λόγω της απαραίτητης επένδυσης σε χρόνο και χρήμα προτού ξεκινήσουν οι αυτοματοποιημένες δοκιμές, λίγα είναι τα οφέλη από τη διαδικασία αυτή για σύντομες δοκιμές.

- Στις δοκιμές, τίποτα δεν συγκρίνεται με έναν υπεύθυνο δοκιμής (tester) που έχει τη δυνατότητα να πραγματοποιήσει μια λεπτομερή επισκόπηση ενός προϊόντος ή έργου απλά αξιοποιώντας την εκπαίδευσή του, και την εμπειρία του. Καμία αυτοματοποιημένη ρύθμιση δεν μπορεί να επαναλάβει αυτό το είδος χειροκίνητης δοκιμής που δεν χρειάζεται δοκιμαστικά σχέδια και οι εκτιμήσεις βασίζονται μόνο στην αίσθηση ότι κάτι δεν πάει καλά. Τα αυτοματοποιημένα εργαλεία είναι έξυπνα, αλλά δεν είναι τόσο έξυπνα όσο οι άνθρωποι.
- Οι αυτοματοποιημένες δοκιμές μπορούν να περιέχουν σφάλματα. Παρά τα θετικά που προσφέρει η αυτοματοποιημένη δοκιμή, τα αυτοματοποιημένα σενάρια μπορούν να δοκιμάσουν μόνο όσα έχουν γραφτεί για δοκιμή. Αυτό σημαίνει ότι υπάρχει πάντα η πιθανότητα το άτομο που γράφει το σενάριο να χάσει ένα πιθανό ζήτημα που πρέπει να συμπεριληφθεί. Χωρίς την εισαγωγή ανθρώπινης συμβολής σε αυτό το σημείο, είναι πιθανό κάποιο ζήτημα να μη δοκιμαστεί.
- Δοκιμή σε διαφορετικά προγράμματα περιήγησης και συσκευές. Οι δοκιμές μεταξύ των προγραμμάτων περιήγησης παίζουν ουσιαστικό ρόλο στην πλήρη διαδικασία δοκιμών. Ο έλεγχος για σφάλματα στη λειτουργικότητα είναι συνήθως μια αυτοματοποιημένη διαδικασία, επομένως χρησιμοποιείται ένα εργαλείο δοκιμής του προγράμματος περιήγησης, το οποίο εκτελεί επανειλημμένα το ίδιο σενάριο σε δυνητικά τεράστιους αριθμούς συνδυασμών προγράμματος περιήγησης, συσκευής και έκδοσης. Ωστόσο, κατά τη δοκιμή των πιο φυσικών, οπτικών και απτών στοιχείων, η χειροκίνητη δοκιμή είναι η καλύτερη επιλογή, επειδή οι δοκιμαστές μπορούν να εξερευνήσουν χρησιμοποιώντας τις αισθήσεις της όρασης και της αφής [10]. Ένας ιστότοπος μπορεί να μην φαίνεται τέλειος σε ένα συγκεκριμένο πρόγραμμα περιήγησης, οπότε θα πρέπει να αποφασιστεί χρησιμοποιώντας την ανθρώπινη κρίση, τι είναι αποδεκτό.

- Το πλεονέκτημα των παραλλαγών στη χειροκίνητη δοκιμή. Υπό κανονικές συνθήκες, η αυτοματοποιημένη δοκιμή δεν θα αποκλίνει από τη διαδρομή του σεναρίου δοκιμής που ακολουθεί. Όταν ένα άτομο δοκιμάζει κάτι, ακόμα και αν ακολουθεί δοκιμαστικές περιπτώσεις, ο τρόπος που δοκιμάζει θα διαφέρει ελαφρώς κάθε φορά λόγω ανθρώπινων ενστίκτων και επειδή μπορεί να χρειαστεί να επεξεργαστεί διαφορετικές εισόδους. Αυτό προκαλεί την πιθανότητα οι υπεύθυνοι δοκιμών (testers) να υποπέσουν σε σφάλματα σε μέρη που προηγουμένως δεν είχαν ληφθεί υπόψη.



Εικόνα 8 Χαρακτηριστικά της χειροκίνητης δοκιμής.

- Συντήρηση αυτοματοποιημένων δοκιμών. Όταν αλλάζει το λογισμικό είναι συχνά απαραίτητο να ενημερωθούν ορισμένα, ή ακόμη και όλα, τα τεστ, ώστε να μπορούν να εκτελεστούν ξανά επιτυχώς. Όταν η προσπάθεια συντήρησης αυτοματοποιημένων δοκιμών είναι μεγαλύτερη από την προσπάθεια για την εκ νέου εκτέλεση αυτών των δοκιμών χειροκίνητα, ο αυτοματισμός δοκιμής θα εγκαταλειφθεί.
- Τεχνικά προβλήματα στα εργαλεία αυτοματοποιημένων ελέγχων. Τα εργαλεία εκτέλεσης δοκιμών είναι προϊόντα λογισμικού που πωλούνται από εταιρείες. Ως προϊόντα λογισμικού τρίτων, δεν είναι απαλλαγμένα από ελαττώματα ή προβλήματα υποστήριξης. Η διαλειτουργικότητα του εργαλείου με άλλο

λογισμικό, είτε με δικές μας εφαρμογές είτε με προϊόντα τρίτων, μπορεί να είναι ένα σοβαρό πρόβλημα [8] [9].

Τέλος γίνεται μια αναφορά για τις διερευνητικές δοκιμές (exploratory testing), όπου είναι μια προσέγγιση στη δοκιμή λογισμικού που περιγράφεται συνοπτικά ως ταυτόχρονη μάθηση, σχεδιασμός και εκτέλεση δοκιμών [10]. Ορίζονται οι διερευνητικές δοκιμές ως «ένα στυλ δοκιμής λογισμικού που δίνει έμφαση στην προσωπική ελευθερία και ευθύνη του υπεύθυνου δοκιμών να βελτιστοποιεί συνεχώς την ποιότητα της εργασίας του αντιμετωπίζοντας τη μάθηση που σχετίζεται με την συγκεκριμένη δοκιμή(τεστ), σχεδιασμό δοκιμής, εκτέλεση δοκιμής και ερμηνεία αποτελεσμάτων δοκιμής ως αμοιβαία υποστηρικτικές δραστηριότητες που εκτελούνται παράλληλα σε όλο το έργο". Ενώ το λογισμικό δοκιμάζεται, ο υπεύθυνος δοκιμής μαθαίνει πράγματα που μαζί με την εμπειρία και τη δημιουργικότητα δημιουργούν νέες καλές δοκιμές για εκτέλεση [11]. Στο σημείο αυτό, παρουσιάζεται ένας συγκεντρωτικός πίνακας με τις κύριες διαφορές μεταξύ αυτοματοποιημένης δοκιμής (automation testing) και χειροκίνητης δοκιμής (manual testing).

Αυτοματοποιημένη δοκιμή (automation testing)	Χειροκίνητη δοκιμή (manual testing)
Η αυτοματοποιημένη δοκιμή είναι μια διαδικασία που γίνεται με την βοήθεια αυτοματοποιημένων εργαλείων.	Η χειροκίνητη δοκιμή είναι μια διαδικασία που γίνεται χειροκίνητα.
Η αυτοματοποιημένη δοκιμή εκτελεί την ίδια λειτουργία κάθε φορά.	Η χειροκίνητη δοκιμή δεν είναι εφικτό να εκτελεί την ίδια λειτουργία κάθε φορά.
Η αυτοματοποιημένη δοκιμή είναι χρήσιμη όταν χρειάζεται να επαναλάβουμε τα ίδια σενάρια.	Η χειροκίνητη δοκιμή είναι χρήσιμη όταν ένα σενάριο χρειάζεται να επαναληφθεί λίγες φορές ή μόνο μία φορά.
Η αυτοματοποίηση ενός σεναρίου δοκιμής είναι συνέχεια της χειροκίνητης δοκιμής αυτού του σεναρίου.	Πάντα ένα σενάριο δοκιμής γίνεται πρώτα χειροκίνητα.
Η αυτοματοποιημένη δοκιμή για να υλοποιηθεί απαιτεί χρήση κώδικα.	Η χειροκίνητη δοκιμή δεν απαιτεί χρήση κώδικα.

Η αυτοματοποιημένη δοκιμή είναι ακριβή συνήθως.	Η χειροκίνητη δοκιμή δεν είναι συνήθως τόσο ακριβή.

Πίνακας *1 Διαφορές μεταξύ αυτοματοποιημένης δοκιμής και χειροκίνητης δοκιμής.*

1.7 Επίπεδα Δοκιμής Λογισμικού

Υπάρχουν διάφορα επίπεδα (test levels) στα οποία μπορεί να εξελίσσεται η δοκιμή λογισμικού (software testing) [12]. Τα διάφορα επίπεδα δοκιμών λογισμικού κυρίως πραγματοποιούνται από τα άτομα που ασχολούνται με την διασφάλιση ποιότητας της εφαρμογής και πραγματοποιούν τις σχετικές δοκιμές και ονομάζονται testers, κάποιες όμως δοκιμές γίνονται και από τα ίδια άτομα που έχουν αναπτύξει τον κώδικα της εφαρμογής και ονομάζονται developers.

Συγκεκριμένα υπάρχουν τέσσερα επίπεδα (test levels) [1] [6] στα οποία μπορεί να εξελίσσεται η δοκιμή λογισμικού (software testing) τα οποία αναλύονται στη συνέχεια:

- Δοκιμές Μονάδας (Unit testing): Το Unit testing είναι μια μέθοδος δοκιμής (testing) λογισμικού με την οποία ελέγχονται μεμονωμένες μονάδες πηγαίου κώδικα (source code) για να καθοριστεί εάν είναι κατάλληλες για χρήση. Κατά κύριο λόγο το Unit testing πραγματοποιείται από τον developer, δηλαδή το άτομο που αναπτύσσει και «γράφει» τον κώδικα της εφαρμογής, το εκάστοτε λογισμικό.

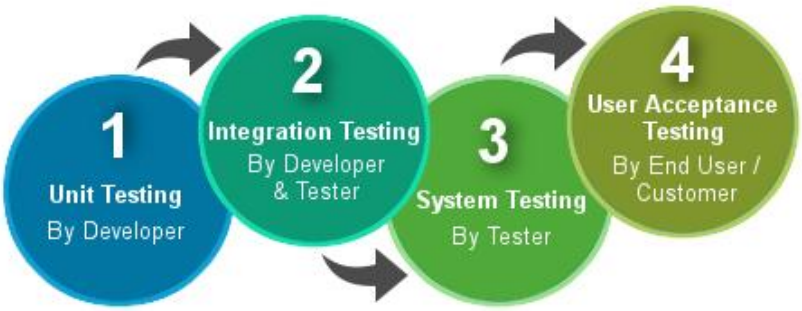
- Δοκιμές Συνένωσης (Integration testing): Το Integration testing είναι μια μέθοδος δοκιμής (testing) λογισμικού κατά την οποία μεμονωμένες ενότητες λογισμικού συνδυάζονται και δοκιμάζονται ως ομάδα, και διεξάγεται για την αξιολόγηση της συμμόρφωσης ενός συστήματος ή ενός στοιχείου με συγκεκριμένες λειτουργικές απαιτήσεις [12]. Συνήθως το Integration testing πραγματοποιείται από τον tester, δηλαδή το άτομο που είναι υπεύθυνο για την ποιότητα του κώδικα της εφαρμογής του εκάστοτε λογισμικού, σε κάποιες περιπτώσεις όμως πραγματοποιείται και από τον developer.

- Δοκιμές Συστήματος (System testing): Το System testing είναι μια μέθοδος δοκιμής (testing) λογισμικού όπου αξιολογεί τον τρόπο με τον οποίο τα διάφορα στοιχεία μιας εφαρμογής αλληλοεπιδρούν μαζί στο πλήρες ολοκληρωμένο

σύστημα ή εφαρμογή. Ως επί το πλείστον, το System testing πραγματοποιείται από τον tester.

- Δοκιμές Αποδοχής (Acceptance testing): Το Acceptance testing είναι η πιο σημαντική μέθοδος δοκιμής (testing) λογισμικού, καθώς σε αυτήν αποφασίζει ο πελάτης εάν εγκρίνει την εφαρμογή/λογισμικό ή όχι. Το Acceptance testing πραγματοποιείται αποκλειστικά από τον τελικό χρήστη, πελάτη.

Η δοκιμή λογισμικού σε διαφορετικά επίπεδα του κύκλου ζωής του λογισμικού αποφέρει και διαφορετικά είδη ελαττωμάτων. Για παράδειγμα, οι δοκιμές μονάδας είναι πιο πιθανό να βρουν λογικά ελαττώματα του κώδικα παρά ελαττώματα σχεδίου συστήματος (system design defects), τα οποία βρίσκονται κυρίως στις δοκιμές συστήματος.



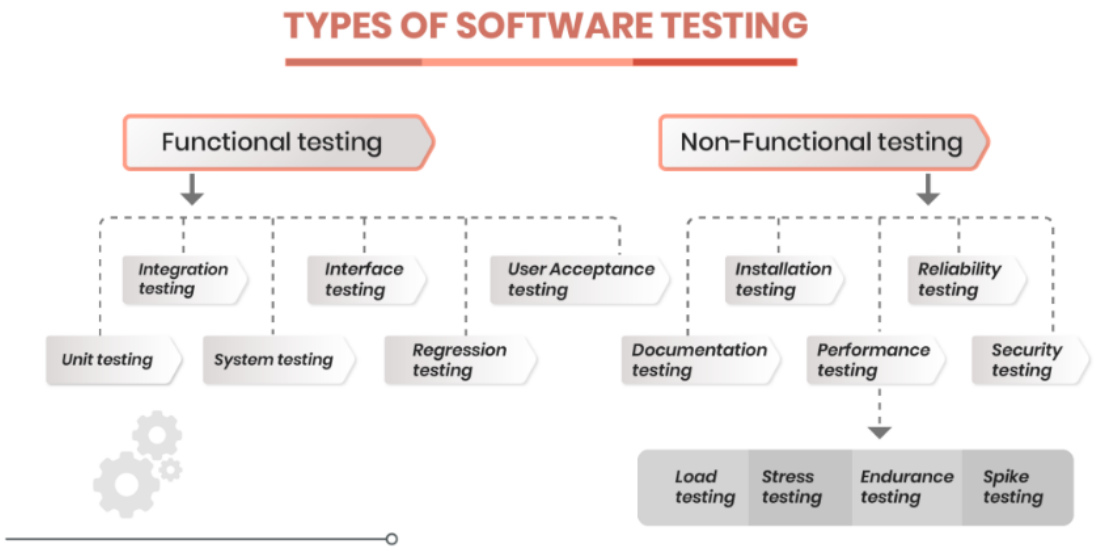
Εικόνα 9 Επίπεδα δοκιμής λογισμικού.

1.8 Γενικοί Τύποι Δοκιμής Λογισμικού

Υπάρχουν δύο τύποι δοκιμής λογισμικού (software testing):

- Λειτουργική δοκιμή (Functional testing): Έχει να κάνει με το εάν η εφαρμογή λειτουργεί σύμφωνα με τα προαπαιτούμενα (requirements) [13]. Λειτουργική αποκαλείται κάθε δοκιμή που επαληθεύει την αναμενόμενη συμπεριφορά των δεδομένων εισόδου-εξόδου.
- Μη λειτουργική δοκιμή (Non functional testing): Έχει να κάνει με το εάν η εφαρμογή λειτουργεί σωστά γενικότερα υπό διαφορετικές συνθήκες όπως η αυξημένη χρήση εφαρμογής (performance testing), διαφορετικό λειτουργικό

σύστημα (cross platform testing) και η χρήση για άτομα με ειδικές ανάγκες (accessibility testing). Οι μη λειτουργικές απαιτήσεις περιγράφουν τα γνωρίσματα ενός συστήματος ως ενιαίου συνόλου, καθώς και το γνώρισμα της λειτουργικής συμπεριφοράς [1] [2].



Εικόνα 10 Κατηγορίες λειτουργικών και μη λειτουργικών δοκιμών.

Στο σημείο αυτό, παρουσιάζεται ένας συγκεντρωτικός πίνακας με τις κύριες διαφορές μεταξύ λειτουργικής δοκιμής (functional testing) και μη λειτουργικής δοκιμής (non functional testing).

Λειτουργική δοκιμή (Functional testing)	**Μη Λειτουργική δοκιμή (Non Functional testing)**
Δοκιμάζει χαρακτηριστικά/λειτουργίες (Tests features/functions)	Δοκιμάζει μη λειτουργικές πτυχές
Αξιολογεί εάν κάτι είναι παρόν ή δεν υπάρχει	Αξιολογείται σε κλίμακα
Συνήθως εκτελείται χειροκίνητα	Συνήθως εκτελείται αυτοματοποιημένα
Ελέγχει τη συμμόρφωση με τις απαιτήσεις του πελάτη	Ελέγχει τη συμβατότητα με τις προσδοκίες του τελικού χρήστη
Δοκιμάζει τη λειτουργία του προϊόντος	Δοκιμάζει πώς λειτουργεί το προϊόν
Οι απαιτήσεις (requirements) είναι	Οι απαιτήσεις (requirements) είναι

εύκολο να καθοριστούν	δύσκολο να καθοριστούν

Πίνακας 2 Κύριες διαφορές μεταξύ λειτουργικής δοκιμής και μη λειτουργικής δοκιμής.

1.9 Δραστηριότητες Δοκιμής Λογισμικού

Μια διαδικασία δοκιμής αποτελείται από τις ακόλουθες κύριες ομάδες δραστηριοτήτων:

- Test Planning: Είναι η διαδικασία κατά την οποία παράγεται το σχέδιο δοκιμών (Test Plan). Το σχέδιο δοκιμών είναι συνήθως ένα έγγραφο στο οποίο καταγράφονται και αναλύονται οι στρατηγικές που θα ακολουθηθούν, οι στόχοι που καλούνται να επιτύχουν οι δοκιμές, οι καταληκτικές ημερομηνίες και οι διαθέσιμοι πόροι που έχει το εκάστοτε προϊόν στην διάθεσή του για τις δοκιμές. Το έγγραφο αυτό ανανεώνεται καθ' όλη την διάρκεια της ανάπτυξης του λογισμικού καθώς προκύπτουν νέες λειτουργίες και πρέπει εκτός από την ομάδα που είναι υπεύθυνη για τη δοκιμή της εφαρμογής να κοινοποιηθεί στους Επιχειρηματικούς Αναλυτές (Business Analysts), Project Managers και τις ομάδες προγραμματιστών (developers, testers).
- Test Monitor: Είναι η διαδικασία κατά την οποία ενημερώνονται όλοι οι εμπλεκόμενοι στο εκάστοτε προϊόν για την εξέλιξη της δοκιμαστικής διαδικασίας παρέχοντας τους τα αποτελέσματα των δοκιμών καθώς και τις μετρήσεις που έχουν υπολογιστεί (π.χ. το ποσοστό κάλυψης, πόσα σενάρια δοκιμής έχουν αποτύχει, τους πόρους που χρειάστηκαν). Βάση αυτής της επικοινωνίας γίνονται εκτιμήσεις και παίρνονται αποφάσεις για τις επόμενες ενέργειες που πρέπει να γίνουν.
- Test Control: Προκύπτει ως αποτέλεσμα του Test Monitor και περιλαμβάνει την περιγραφή των προαναφερόμενων ενεργειών μερικές εκ των οποίων είναι αλλαγές στις προτεραιότητες που πρέπει να τεθούν, επανεκτίμηση των προθεσμιών, πιθανές αλλαγές στο περιβάλλον στο οποίο αναπτύσσονται οι δοκιμές.
- Test Analysis: Σε αυτή τη διαδικασία αναλύονται οι απαιτήσεις που έχει μια εφαρμογή ή ένα κομμάτι της και ορίζεται το τι πρέπει να δοκιμαστεί.

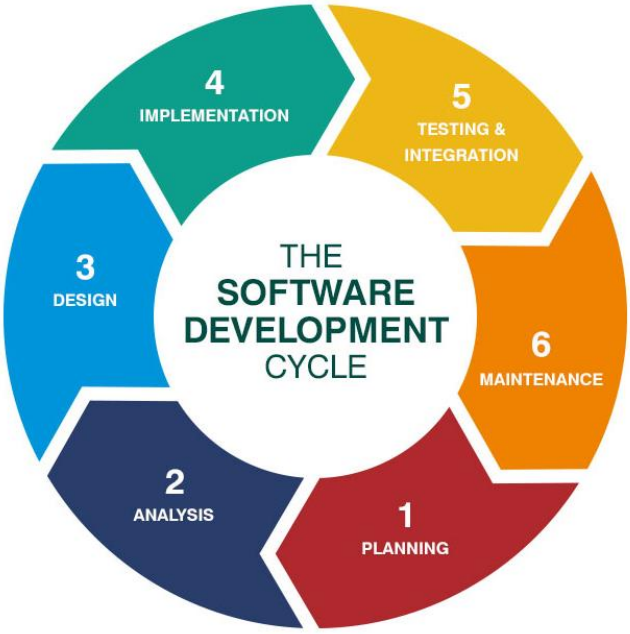
- Test Design: Σε αυτό το στάδιο αναπτύσσονται οι πιθανές περιπτώσεις και τα σενάρια δοκιμών από τις απαιτήσεις που προέκυψαν από το στάδιο του test analysis.
- Test Implementation: Σε αυτή τη διαδικασία υλοποιούνται οι δοκιμές και παράγονται όλα τα δεδομένα δοκιμών που θα χρειαστούν.

- Test Execution: Ακολουθεί το Test Implementation και εδώ εκτελούνται όλες οι δοκιμές που έχουν αναπτυχθεί από το στάδιο αυτό για την εφαρμογή η τμήμα αυτής.
- Test Completion: Είναι το τελευταίο στάδιο και σε αυτό ελέγχονται αν τα κριτήρια που έχουν οριστεί για να σταματήσει η δοκιμαστική διαδικασία (π.χ. ότι το ορισμένο ποσοστό κάλυψης έχει επιτευχθεί, ότι δεν υπάρχουν ελαττώματα) πληρούνται [1] [2].

1.10 Μοντέλο Κύκλου Ζωής Ανάπτυξης Λογισμικού

Ένα μοντέλο κύκλου ζωής ανάπτυξης λογισμικού (software development lifecycle model) περιγράφει τους τύπους δραστηριότητας που εκτελούνται σε κάθε στάδιο του έργο ανάπτυξης λογισμικού και παρουσιάζεται πώς οι δραστηριότητες σχετίζονται μεταξύ τους λογικά και χρονολογικά. Υπάρχουν πολλά διαφορετικά μοντέλα κύκλου ζωής ανάπτυξης λογισμικού, καθένα από τα οποία απαιτεί διαφορετικές προσεγγίσεις όσον αφορά την δοκιμή του λογισμικού [14]. Τις περισσότερες φορές χωρίζουμε τον κύκλο ζωής ανάπτυξης λογισμικού στα έξι παρακάτω βήματα:

- Προγραμματισμός
- Ανάλυση
- Σχεδιασμός
- Υλοποίηση
- Έλεγχος Λογισμικού
- Διατήρηση



Εικόνα 11 Μοντέλο κύκλου ζωής ανάπτυξης λογισμικού.

Ένα μοντέλο κύκλου ζωής ανάπτυξης λογισμικού διαχωρίζει τις δραστηριότητες που θα πρέπει να εκτελεστούν σε στάδια και τις συσχετίζει μεταξύ τους λογικά και χρονολογικά. Σε κάθε μοντέλο κύκλου ζωής ανάπτυξης λογισμικού η δοκιμή λογισμικού έχει διαφορετικές προσεγγίσεις [15]. Για την επιλογή ενός μοντέλου κύκλου ζωής λογισμικού πρέπει να ελέγχονται τα χαρακτηριστικά, ο στόχος και ο τύπος του έργου καθώς και το πόσο γρήγορα πρέπει να βγει στην αγορά. Σε αυτή την ενότητα θα παρουσιαστούν δυο μοντέλα κύκλου ζωής λογισμικού :

- Διαδοχικά μοντέλα ανάπτυξης (Sequential development models): Η ροή των δραστηριοτήτων είναι γραμμική και διαδοχική. Όταν ολοκληρώνεται μια φάση ξεκινάει η επόμενη. Σε αυτό το είδος μοντέλο ανάπτυξης λογισμικού οι δοκιμές γίνονται όταν ολοκληρωθεί η υλοποίηση όλων των λειτουργιών. Στην ουσία τα στάδια που αναπτύσσονται το ένα μετά το άλλο είναι :
    - Ανάλυση Απαιτήσεων
    - Σχεδιασμός
    - Ανάπτυξη Κώδικα
    - Ανάπτυξη Δοκιμών.

Με αυτό τον τρόπο μια εφαρμογή που αναπτύσσεται με ένα διαδοχικό μοντέλο ανάπτυξης παραδίδεται ολοκληρωμένη αλλά απαιτείται πολύς χρόνος για αυτό. Ένα γνωστό διαδοχικό μοντέλο ανάπτυξης λογισμικού είναι το Waterfall.

- Επαναληπτικά μοντέλα ανάπτυξης (Iterative and incremental development models): Η ροή των δραστηριοτήτων είναι διαδοχική αλλά είναι αλληλοκαλυπτόμενη. Δεν απαιτείται η ολοκλήρωση μιας φάσης για να ξεκινήσει η επόμενη. Οι δοκιμές αναπτύσσονται παράλληλα με την υλοποίηση του κώδικα. Υπάρχουν επαναλαμβανόμενες εκδόσεις λογισμικού προς τον πελάτη για να υπάρχει άμεση ανατροφοδότηση με σχόλια πάνω στο κομμάτι που παραδόθηκε. Ένα γνωστό επαναληπτικό μοντέλο ανάπτυξης λογισμικού είναι το Agile [1] [2].

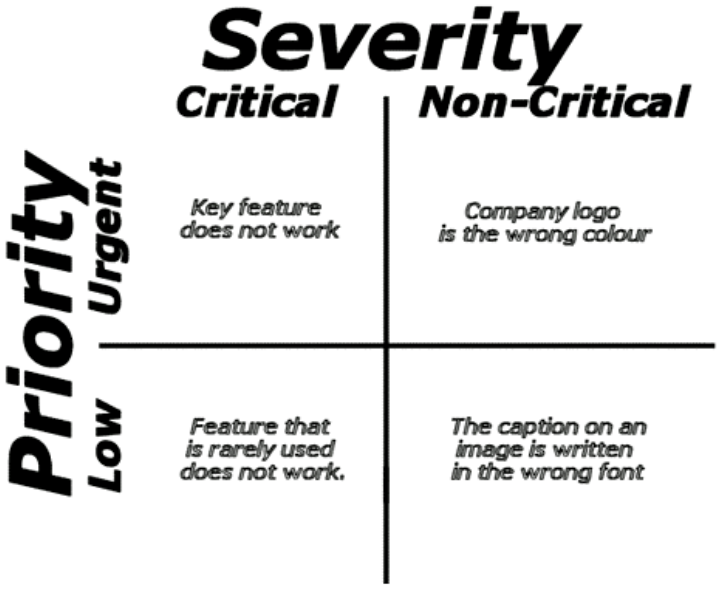
1.11 Συμπληρωματικές Έννοιες

Όπως αναφέρθηκε στο παραπάνω κεφάλαιο ανάλογα με το σχετικό μοντέλο που έχει επιλεχθεί για τον κύκλο ζωής ανάπτυξης λογισμικού, καθορίζεται εάν η υλοποίηση της εφαρμογής θα γίνει τμηματικά και θα παραδίδεται μέρος της εφαρμογής σε τακτά χρονικά διαστήματα στον τελικό πελάτη ή θα παραδίδεται η εφαρμογή ολοκληρωμένη μετά από αρκετά μεγάλο χρονικό διάστημα [29]. Στην περίπτωση που πραγματοποιείται η ανάπτυξη λογισμικού ανά τμήματα και σε συγκεκριμένο χρονικό πλαίσιο (η μια χρονική περίοδος ονομάζεται sprint), το agile είναι ένα δημοφιλές και πολύ επιτυχημένο μοντέλο ανάπτυξης λογισμικού για να παραδίδονται τα προϊόντα γρήγορα στα χέρια των χρηστών με καλή ποιότητα.

Σε μια ευέλικτη προσέγγιση οι απαιτήσεις ή οι επιχειρηματικές ανάγκες αναλύονται σε μικρότερες ενότητες (tickets) που ονομάζονται stories. Κάθε story έχει καθορισμένα κριτήρια αποδοχής [32], στις περιπτώσεις που κάποιο καινούργιο μέρος «κώδικα» της αναπτυσσόμενης εφαρμογής δεν ανταποκρίνεται στις προκαθορισμένες απαιτήσεις δημιουργείται αναλόγως ένα νέο είδος ticket που ονομάζεται defect. Το σύνολο των stories και defects καταγράφονται στο εκάστοτε Ticket Management Tool για παράδειγμα ένα ευρέως γνωστό agile project management tool είναι το Jira. Η κατανομή του συνόλου των tickets στα sprints γίνεται με βάση τα story points, την προτεραιότητα (priority) και την σοβαρότητα (severity).

Για τα διάφορα είδη tickets δίνονται εκτιμήσεις σε μορφή χρόνου: ημέρες, εβδομάδες, μήνες για την υλοποίηση αυτών. Πολλές agile ομάδες χρησιμοποιούν την

τεχνική του προγραμματισμού (planning) poker technique για την εκτίμηση των story points. Τα story points είναι μονάδες μέτρησης για την έκφραση μιας εκτίμησης της συνολικής προσπάθειας που απαιτείται για την πλήρη υλοποίηση ενός story σε σχέση με την πολυπλοκότητα της εργασίας, τον όγκο της εργασίας και τον κίνδυνο ή την αβεβαιότητα. Αυτή η τεχνική προγραμματισμού περιλαμβάνει για την εκτίμηση των story points τους αριθμούς που προκύπτουν από την ακολουθία Fibonacci. Η σειρά Fibonacci αποτελείται από μια ακολουθία αριθμών όπου κάθε αριθμός είναι ένα άθροισμα των δύο προηγούμενων αριθμών. Η παραδοσιακή ακολουθία Fibonacci είναι 1, 2, 3, 5, 8, 13, 21, 34 και ούτω καθεξής. Στα έργα Agile, αυτή η σειρά τροποποιείται. Η τροποποιημένη σειρά Fibonacci είναι 0, 1, 2, 3, 5, 8, 13, 20, 40, 100 [16].



Εικόνα 12 Διαφορά μεταξύ προτεραιότητας (priority) και σοβαρότητας (severity).

Η σειρά με την οποία πρέπει να επιδιορθωθεί ένα defect ή να υλοποιηθεί ένα story αναφέρεται ως προτεραιότητα (priority). Όσο υψηλότερη είναι η προτεραιότητα, τόσο πιο γρήγορα θα πρέπει να επιλυθεί το πρόβλημα ή να υλοποιηθεί ένα story. Τα defects που καθιστούν το σύστημα λογισμικού μη λειτουργικό έχουν προτεραιότητα πάνω από τα defects που επηρεάζουν μόνο ένα μικρό μέρος της λειτουργίας του λογισμικού [17]. Η προτεραιότητα (priority) αναφέρεται στη σειρά με την οποία ένας προγραμματιστής πρέπει για παράδειγμα να αντιμετωπίσει ένα defect, ενώ η σοβαρότητα (severity) αναφέρεται στον βαθμό επιρροής που έχει ένα ελάττωμα στη

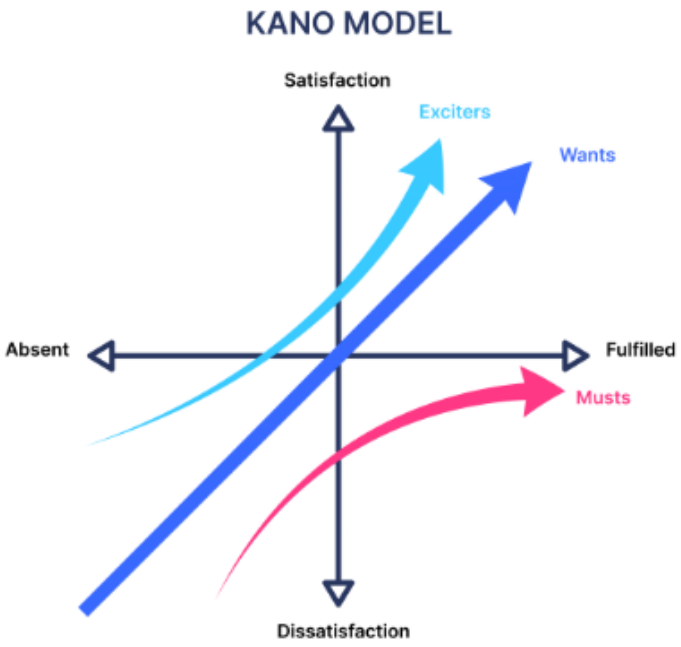
λειτουργία του προϊόντος Τέλος η προτεραιότητα (priority) χωρίζεται σε τρεις κατηγορίες: χαμηλή (low), μεσαία (medium) και υψηλή (high), ενώ η σοβαρότητα (severity) χωρίζεται στις εξής κατηγορίες: κρίσιμη (critical), μείζων (major), μέτρια (moderate) , μικρής σημασίας (minor) και κοσμητικής σημασίας (cosmetic).

Η agile ευέλικτη ανάπτυξη προϊόντων (λογισμικό) μπορεί να είναι πολύ πιο περίπλοκη από τις κανονικές διαδικασίες ανάπτυξης, κυρίως επειδή περιλαμβάνει πολλές ομάδες με αυτοοργάνωση [30]. Αυτό παρουσιάζει μερικά άνευ προηγουμένου πιθανά προβλήματα. Αυτός είναι ο λόγος για τον οποίο οι agile ευέλικτες ομάδες ανάπτυξης προϊόντων προτιμούν να εργάζονται με μια τεχνική ιεράρχησης στην αρχή, για παράδειγμα το agile Kano μοντέλο προτεραιοτήτων ανάπτυξης προϊόντων - λογισμικού.

Το μοντέλο Kano ενθαρρύνει τη σκέψη πώς τα προϊόντα (λογισμικό) σχετίζονται με τις ανάγκες των πελατών και βοηθάει στο να επιλεχθούν τα χαρακτηριστικά εκείνα που αυξάνουν την ικανοποίηση των πελατών χωρίς να ανεβάζουν πολύ το κόστος [18]. Το μοντέλο Kano διακρίνει τις ιδιότητες ενός προϊόντος σε τρεις τύπους:

- Βασικά γνωρίσματα: Αυτά είναι τα βασικά χαρακτηριστικά που οι πελάτες αναμένουν να έχει ένα προϊόν.
- Αποδοτικά γνωρίσματα: Αυτά τα στοιχεία δεν είναι απολύτως απαραίτητα, ωστόσο αυξάνουν την ικανοποίηση ενός πελάτη.
- Γνωρίσματα ενθουσιασμού: Αυτά είναι χαρακτηριστικά που έχουν το στοιχείο της έκπληξης και μπορούν να δώσουν πραγματικό ανταγωνιστικό πλεονέκτημα. Πρόκειται για χαρακτηριστικά που οι πελάτες δεν ξέρουν καν ότι θέλουν, γι’ αυτό ενθουσιάζονται όταν τα βρίσκουν.

Όπως γίνεται εύκολα αντιληπτό, αν τα χαρακτηριστικά ενός προϊόντος δεν διαθέτουν τα βασικά γνωρίσματα, τότε τα επίπεδα ικανοποίησης είναι πολύ χαμηλά.



Εικόνα 13 Το μοντέλο Kano.

Μια από τις επικρατέστερες μεθοδολογίες ανάπτυξης λογισμικού είναι η scrum. Η scrum είναι πλαίσιο επαναληπτικής και επαυξητικής ευέλικτης μεθοδολογίας ανάπτυξης λογισμικού για τη διαχείριση της ανάπτυξης ενός προϊόντος. Ορίζει μια ευέλικτη, ολιστική στρατηγική για την ανάπτυξη του προϊόντος, όπου μια ομάδα ανάπτυξης λειτουργεί ως μονάδα για την επίτευξη ενός κοινού στόχου και επιτρέπει στις ομάδες να αυτοοργανώνονται με την ενθάρρυνση της φυσικής συνεργασίας ή διαδικτυακής συνεργασίας όλων των μελών της ομάδας, καθώς και την καθημερινή πρόσωπο-με-πρόσωπο επικοινωνία μεταξύ όλων των μελών της ομάδας και κλάδων στο σχεδιασμό. Η αρχή-κλειδί του scrum είναι η αναγνώριση του ότι κατά τη διάρκεια της διαδικασίας παραγωγής, οι πελάτες μπορούν να αλλάξουν γνώμη σχετικά με το τι θέλουν και χρειάζονται (συχνά αυτό αναφέρεται ως μεταβλητότητα αναγκών, καθώς και ότι μη αναμενόμενες προκλήσεις δεν είναι εύκολα διαχειρίσιμες με έναν παραδοσιακό προγνωστικό ή σχεδιασμένο τρόπο [19]. Το scrum υιοθετεί μία εμπειρική προσέγγιση αποδεχόμενο ότι το πρόβλημα δεν μπορεί να κατανοηθεί πλήρως ή να οριστεί, εστιάζοντας αντ' αυτού στην μεγιστοποίηση της ικανότητας της ομάδας να διανέμει γρήγορα, να αποκρίνεται σε αναδυόμενες απαιτήσεις και να προσαρμόζεται σε εξελισσόμενες τεχνολογίες και αλλαγές στην κατάσταση της αγοράς.

Σχετικά με τη ροή εργασιών στη scrum μεθοδολογία, Το sprint είναι το βασικό στοιχείο ανάπτυξης στη scrum μεθοδολογία. Το sprint είναι μία χρονοπεριορισμένη προσπάθεια σε μία ακριβή διάρκεια [33]. Η διάρκεια καθορίζεται εκ των προτέρων για κάθε sprint και είναι συνήθως μεταξύ μίας εβδομάδας και ενός μήνα, με τις δύο εβδομάδες να αποτελούν την πιο κοινή. Κάθε sprint ξεκινάει με μία δραστηριότητα σχεδιασμού sprint που έχει ως στόχο τον καθορισμό των εκκρεμοτήτων του sprint, τον προσδιορισμό της εργασίας για το sprint και τον ορισμό [34] μίας εκτιμώμενης χρονικής δέσμευσης για την επίτευξη του στόχου του sprint. Κάθε sprint τελειώνει με μία επανεξέταση και αναδρομή του sprint, ώστε οι συμμετέχοντες να αξιολογούν την πρόοδό τους που θα παρουσιάσουν στους ενδιαφερόμενους και να αναγνωρίσουν οι ίδιοι όσα έμαθαν ώστε να τα αξιοποιήσουν στα επόμενα sprint.

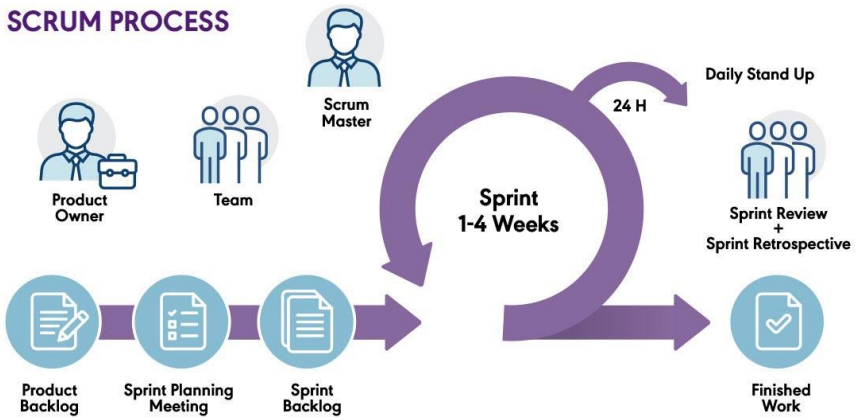
Η scrum μεθοδολογία δίνει έμφαση στο προϊόν εργασίας στο τέλος του sprint που είναι όντως έτοιμο. Στην περίπτωση του λογισμικού, αυτό συνήθως περιλαμβάνει ότι το λογισμικό έχει ολοκληρωθεί, έχει ελεγχθεί πλήρως [19], έχει τεκμηριωθεί για τον τελικό χρήστη και είναι ενδεχομένως έτοιμο για παράδοση.

Σχετικά με τον σχεδιασμό στη scrum μεθοδολογία, Στην αρχή κάθε sprint, η ομάδα οργανώνει μία εκδήλωση σχεδιασμού sprint (sprint planning event) στην οποία:

- Γνωστοποιείται το πιθανό εύρος δουλειάς για το sprint.
- Επιλέγονται στοιχεία από τις εκκρεμότητες του προϊόντος που είναι πιθανό να ολοκληρωθούν.
- Ετοιμάζονται οι εκκρεμότητες του sprint που περιγράφουν την απαιτούμενη δουλειά για την ολοκλήρωση των επιλεγόμενων εκκρεμοτήτων του προϊόντος.
- Οργανώνεται μία εκδήλωση σχεδιασμού, με χρονική διάρκεια έως τέσσερις ώρες, για ένα sprint των δύο εβδομάδων (το χρονικό όριο ρυθμίζεται αναλόγως για σπριντ διαφορετικής διάρκειας).
    - Στο πρώτο μισό της εκδήλωσης, ολόκληρη η ομάδα (ομάδα ανάπτυξης, ο αρχηγός του scrum και ο ιδιοκτήτης προϊόντος) συμφωνεί στο ποιά στοιχεία από τις εκκρεμότητες του προϊόντος θα ληφθούν υπόψιν σε αυτό το sprint.

- Στο δεύτερο μισό της εκδήλωσης, η ομάδα ανάπτυξης αναλύει τη δουλειά (εργασία) που απαιτείται για να ολοκληρώσουν αυτές τις εκκρεμότητες, οι οποίες και καταλήγουν στις εκκρεμότητες του sprint.

Όταν τα μέλη της ομάδας ανάπτυξης ετοιμάσουν την λίστα με τις εκκρεμότητες του sprint, δεσμεύονται (συνήθως ψηφίζοντας) να φέρουν εις πέρας τις εργασίες μέσα στο όριο του sprint.



Εικόνα 14 Η μέθοδος του scrum.

Σχετικά με το καθημερινό scrum, Κάθε μέρα κατά τη διάρκεια του sprint, η ομάδα κάνει ένα καθημερινό scrum (ή stand-up) με συγκεκριμένες οδηγίες:

- Όλα τα μέλη της ομάδας ανάπτυξης έρχονται προετοιμασμένα. Το καθημερινό scrum:
  - ξεκινάει ακριβώς στην ώρα του ακόμα και αν κάποια μέλη της ομάδας ανάπτυξης απουσιάζουν .
  - θα πρέπει να πραγματοποιείται την ίδια ώρα και στο ίδιο μέρος κάθε μέρα.
  - είναι περιορισμένο (χρονοπρογραμματισμένο) στα δεκαπέντε λεπτά.
- Ο καθένας είναι ευπρόσδεκτος, παρόλο που κανονικά μόνο οι ρόλοι των ομάδων scrum συνεισφέρουν.
- Κατά τη διάρκεια του καθημερινού scrum, κάθε μέλος της ομάδας απαντάει τρεις ερωτήσεις:
  - Τι έκανα χθες που βοήθησε την ομάδα ανάπτυξης να πλησιάσει την επίτευξη του στόχου του sprint;

- Τι θα κάνω σήμερα ώστε να βοηθήσω την ομάδα ανάπτυξης να πλησιάσει την επίτευξη του στόχου του sprint;
- Βλέπω κάποιο εμπόδιο που εμποδίζει εμένα ή την ομάδα στο να πλησιάσει την επίτευξη του στόχου του sprint;

Οποιοδήποτε εμπόδιο (στο οποίο μπορεί να "σκοντάψει" κάποιος, το οποίο μπορεί να προκαλέσει κίνδυνο ή πρόβλημα) αναγνωρίζεται στο καθημερινό scrum, πρέπει να συλλαμβάνεται από τον αρχηγό του scrum και να αναρτάται στον πίνακα scrum της ομάδας, με ένα άτομο [30] που επιλέγεται και έπειτα προορίζεται να δουλέψει στην εύρεση μίας λύσης πάνω στο θέμα (εκτός του καθημερινού scrum). Καμία λεπτομερής συζήτηση περί του θέματος δεν θα πρέπει να συμβαίνει κατά τη διάρκεια του καθημερινού scrum.

Υπάρχει επίσης η έννοια της επανεξέτασης – ανασκόπησης του sprint (sprint review) και της αναδρομής (sprint retrospective) στη scrum μεθοδολογία. Στο τέλος κάθε sprint η ομάδα διεξάγει δύο εκδηλώσεις: την επανεξέταση – ανασκόπηση του sprint και την αναδρομή του sprint. Η ομάδα στην επανεξέταση – ανασκόπηση του sprint (sprint review):

- Παρουσιάζει την ολοκληρωμένη δουλειά στους ενδιαφερόμενους (γνωστό και ως το demo).
- Συνεργάζεται με τους ενδιαφερόμενους φορείς σε θέματα όπως τον αντίκτυπο της ημιτελούς εργασίας (προγραμματισμένης ή μη), τη λήψη προτάσεων για επερχόμενη εργασία (καθοδήγηση για το τι θα πρέπει να υλοποιηθεί στη συνέχεια).

Στην αναδρομή του sprint (sprint retrospective), η ομάδα:

- Κάνει απολογισμό του περασμένου sprint [17]
- Επιθεωρεί πώς πήγε το τελευταίο sprint (άτομα, αλληλεπιδράσεις, διαδικασίες, εργαλεία)
- Αναγνωρίζει μεθόδους συνεχούς βελτίωσης και συμφωνεί στην χρήση τους.

ΚΕΦΑΛΑΙΟ 2ο

2. Σενάριο Χρήσης – Δοκιμή Λογισμικού Συστήματος

Τα τυχερά παιχνίδια είναι παιχνίδια που στηρίζονται στον στοιχηματισμό χρηματικού ποσού ή άλλου αντικειμένου αξίας. Σε τέτοιου είδους παιχνίδια το κέρδος του εκάστοτε παίκτη εξαρτάται αποκλειστικά ή κατά κύριο λόγο από την τύχη. Σε αυτά προϋπόθεση είναι δηλαδή η ύπαρξη ενός στοιχήματος, ενός ρίσκου και κάποιας ανταμοιβής (συνήθως χρηματικής). Σχετικά με τις αποδόσεις στις εφαρμογές στον κλάδο του στοιχηματισμού, ο όρος απόδοση αντιπροσωπεύει την μαθηματικά ή με άλλους τρόπους ορισμένη πιθανότητα επιβεβαίωσης κάποιου γεγονότος.

Στο κεφάλαιο αυτό θα παρουσιαστεί αρχικά η εφαρμογή η οποία υπόκειται σε δοκιμή (system under test - SUT). Πρόκειται για μια διαδικτυακή εφαρμογή (web application), η οποία αναπτύσσεται και συντηρείται από εταιρεία λογισμικού, με βασικό αντικείμενο το διαδικτυακό στοίχημα. Στο πλαίσιο της διαδικασίας αυτής, ακολουθείται ένα σύνολο από ορισμένα βήματα με σκοπό τη διαδικασία διασφάλισης ποιότητας της εφαρμογής αυτής.

Στη συνέχεια αναφέρεται ο τρόπος που εφαρμόζεται το στάδιο της ανάλυσης των δοκιμών (test analysis). Στο βήμα αυτό επεξεργάζονται και αναλύονται διεξοδικά οι απαιτήσεις του πελάτη του προϊόντος, με σκοπό να οριστεί το πλαίσιο στο οποίο θα εφαρμοστεί η δοκιμή. Μετά το βήμα της ανάλυσης, η διασφάλιση ποιότητας στην στοιχηματική αυτή εφαρμογή προχωράει με τη προετοιμασία της δοκιμής (test preparation). Στην πράξη ορίζεται ένα σύνολο από tests, τα οποία οφείλουν να καλύπτουν επαρκώς το σύνολο των απαιτούμενων λειτουργικοτήτων (test implementation). Με άλλα λόγια, διασφαλίζεται το ότι η εκάστοτε λειτουργικότητα ισχύει όταν (εξαγωγή positive tests) και μόνο όταν (εξαγωγή negative tests) οφείλει να ισχύει, σύμφωνα με τις απαιτήσεις του πελάτη.

Έπεται η διαδικασία εκτέλεσης των tests (test execution), η οποία μπορεί να πραγματοποιηθεί χειροκίνητα δηλαδή manually, είτε με εφαρμογή αυτοματοποιημένης δοκιμής (test automation application). Στην πρώτη περίπτωση, ο

ίδιος ο χρήστης δηλαδή ο tester χρησιμοποιεί το γραφικό περιβάλλον χρήστη (user interface), ακολουθώντας ορισμένα σύνολα βημάτων (steps). Ένα σύνολο από steps πρακτικά αποτελεί ένα σενάριο δοκιμής, μία ροή σύμφωνα με τις προδιαγραφές (business flow). Αντίστοιχα, ένα σύνολο από σενάρια, πρακτικά προσφέρει την εκάστοτε κάλυψη (test coverage), η οποία αντιστοιχεί στην επαρκή διασφάλιση ποιότητας του προϊόντος.

Η αξιολόγηση κριτηρίων εξόδου (test exit criteria evaluation) ορίζει τη διαδικασία μετά την εκτέλεση, όπου ο tester κάνει εκτίμηση των αποτελεσμάτων και ως εκ τούτου αποφασίζει εάν αυτά είναι επαρκή ώστε να τελειώσει η διαδικασία δοκιμής ή να επαναληφθεί.

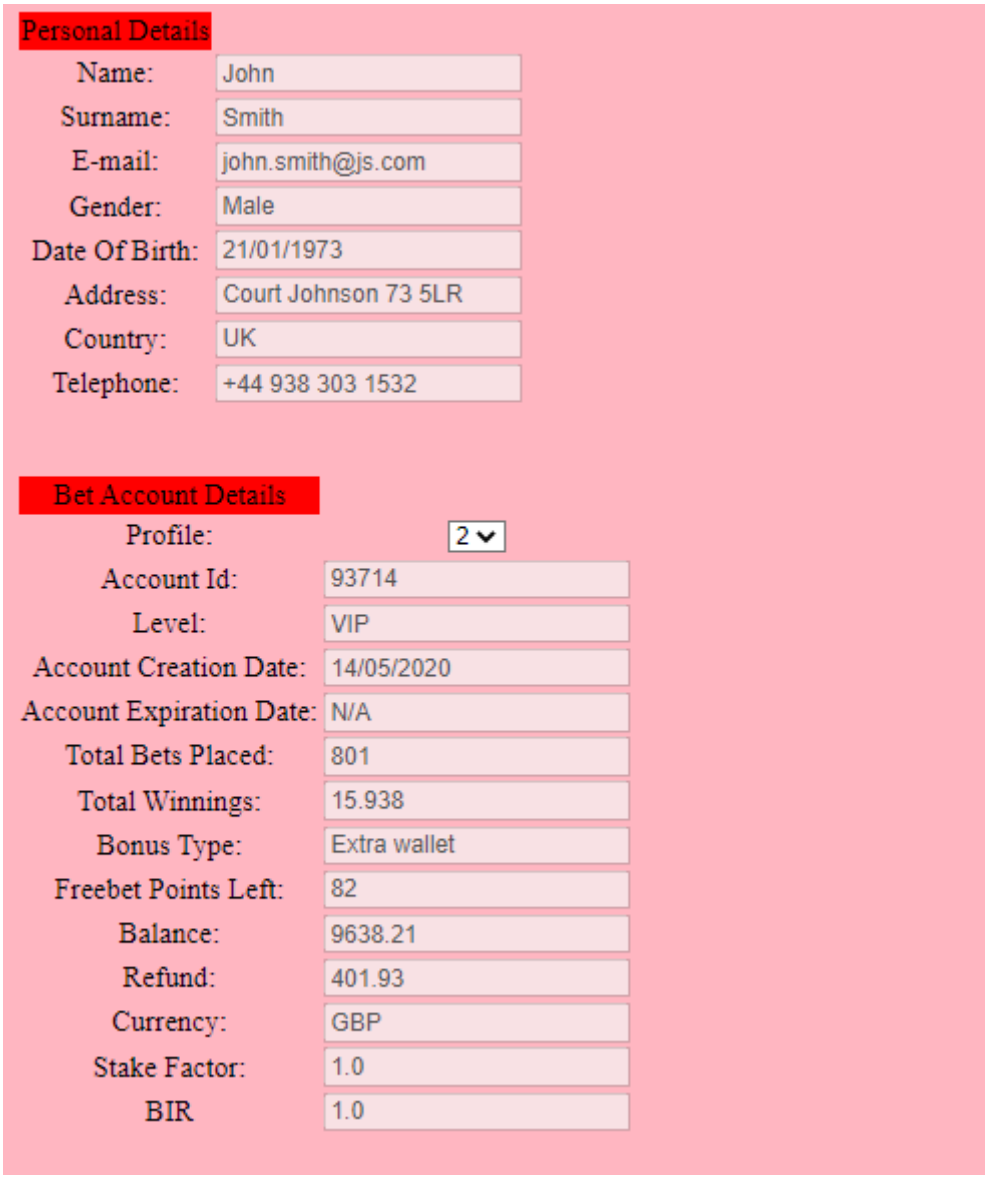
2.1 Εφαρμογή που Υπόκειται σε Δοκιμή

Η εφαρμογή που υπόκειται σε δοκιμή δηλαδή testing, αποτελείται από ένα πλήθος διαδικτυακών σελίδων. Υπάρχουν διαφορετικοί τύπου χρηστών, απλοί χρήστες με περιορισμένα δικαιώματα (permissions) και διαχειριστές με τα περισσότερα έαν όχι όλα τα δικαιώματα (permissions) της εφαρμογής, οι χρήστες μπορεί να είναι ενεργοί χρήστες, ανενεργοί ή μπλοκαρισμένοι, επίσης υπάρχουν και διαφορετικές εκδόσεις του προϊόντος, οι οποίες αντιστοιχούν στα περιβάλλοντα όπου υπάρχει η εφαρμογή (deployed versions). Οι κύριες σελίδες της εφαρμογής είναι η αρχική οθόνη χρήστη στοιχηματικής εφαρμογής (welcome customer page), η κύρια σελίδα του πελάτη (customer page), η σελίδα της τοποθέτησης ενός στοιχήματος (bet placement page) και η σελίδα της εξαργύρωσης ενός στοιχήματος (bet cashout page). Το σύστημα υπό δοκιμή (SUT) που αποτελείται από ένα σύνολο από λειτουργικότητες αναφέρεται σε ένα σύστημα που ελέγχεται για σωστή λειτουργία. Κάθε μία από αυτές τις λειτουργικότητες μπορεί να έχει τη δική του διαμόρφωση (όνομα και έκδοση), καθιστώντας το επεκτάσιμο για μια σειρά δοκιμών ώστε να γίνεται όλο και πιο ακριβής, ανάλογα με την ποσότητα ποιότητας του συστήματος που δοκιμάζεται.

Ακολουθεί μια εικόνα με την αρχική οθόνη χρήστη (welcome customer page) της στοιχηματικής εφαρμογής που υπόκειται σε δοκιμή.

Εικόνα 15 Αρχική οθόνη χρήστη στοιχηματικής εφαρμογής.

Στην εικόνα που ακολουθεί φαίνεται η σελίδα του πελάτη (customer page) της στοιχηματικής εφαρμογής που υπόκειται σε δοκιμή. Η εφαρμογή είναι σχεδιασμένη έτσι ώστε ο εκάστοτε χρήστης να παραμετροποιεί διάφορα στοιχεία του γραφικού περιβάλλοντος (χρώμα, γραμματοσειρά, κ.ά). Τα βασικά πεδία είναι χωρισμένα σε δύο λειτουργικές περιοχές. Τις προσωπικές πληροφορίες (Personal Details) καθώς και τις πληροφορίες που αφορούν τον λογαριασμό (Bet Account Details). Στην πρώτη λειτουργική περιοχή, τα βασικά πεδία είναι το ονοματεπώνυμο (Name, Surname), η χώρα (Country) και το τηλέφωνο επικοινωνίας (Telephone). Στην δεύτερη λειτουργική περιοχή, τα βασικά πεδία είναι το αναγνωριστικό (id) του λογαριασμού του χρήστη (Account Id), το επίπεδο (Level) στο οποίο ανήκει, το υπόλοιπο (Balance) που διαθέτει για να μπορεί να πραγματοποιήσει ένα στοίχημα ή πολλά στοιχήματα, και οι πόντοι που αφορούν την δυνατότητα δωρεάν στοιχήματος (Freebet Points Left).

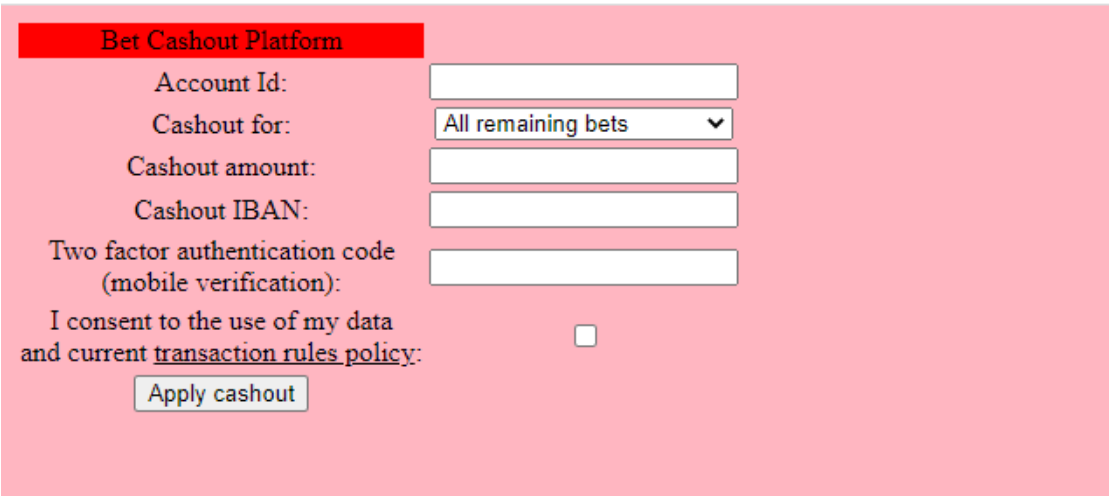


Εικόνα 16 Στιγμιότυπο της σελίδας του πελάτη της στοιχηματικής εφαρμογής που υπόκειται σε δοκιμή.

Η παρακάτω εικόνα αντιστοιχεί στη σελίδα τοποθέτησης στοιχήματος (bet placement page). Τα βασικά πεδία της σελίδας αυτής είναι το αναγνωριστικό (id) του λογαριασμού του χρήστη (Account Id), η κατηγορία του αθλήματος που θα αφορά το συγκεκριμένο στοίχημα (Category Type), το αναγνωριστικό (id) του συμβάντος στο οποίο πρόκειται να στοιχηματίσει (Event Id), και άλλων αντίστοιχων ιεραρχικών στοιχηματικών εννοιών (Market Type Id, Selection Id). Ο χρήστης επιλέγει το ποσό που στοιχηματίζει (Stake), το νόμισμα που αντιστοιχεί (EUR, GBP, USD, DKK, RON κ.ά), καθώς και εάν στοιχηματίζει πριν το παιχνίδι (Pre- Play) ή/και εν ώρα παιχνιδιού (BIR, Bet In Running). Πατώντας το “Place bet”, ο χρήστης υποβάλλει στο σύστημα την τοποθέτηση του στοιχήματος.

Εικόνα 17 Στιγμιότυπο της σελίδας της τοποθέτησης ενός στοιχήματος της εφαρμογής που υπόκειται σε δοκιμή.

Στη συνέχεια ακολουθεί η εικόνα που εμφανίζει τη σελίδα που αντιστοιχεί στη διαδικασία της εξαργύρωσης ενός στοιχήματος (bet cashout page). Βασικά πεδία είναι το ποσό εξαργύρωσης (Cashout Amount) καθώς και το πεδίο ασφαλείας που αντιστοιχεί στην επαλήθευση σε δύο βήματα. Η εξαργύρωση μπορεί να είναι ολική (full cashout) ή μερική (partial cashout).



Εικόνα 18 Στιγμιότυπο της σελίδας της εξαργύρωσης ενός στοιχήματος της εφαρμογής που υπόκειται σε δοκιμή.

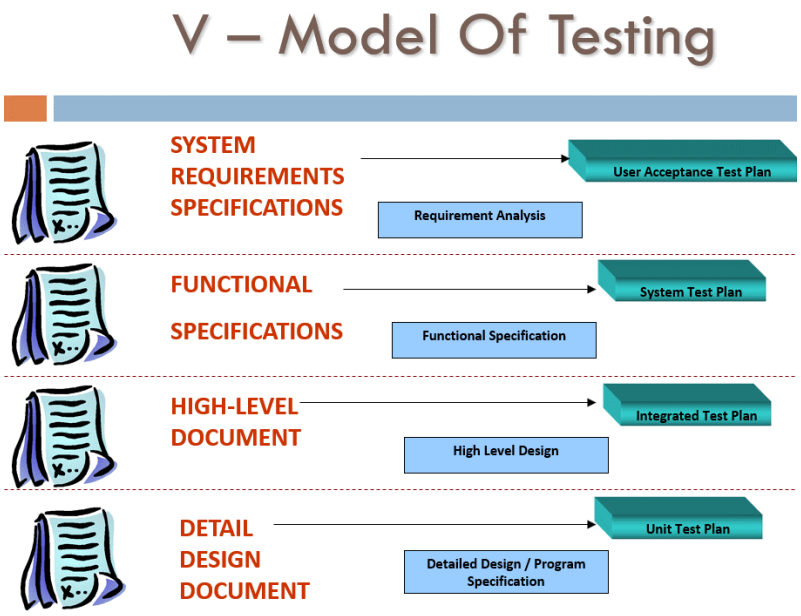
2.2 Ανάλυση Δοκιμής

Στο στάδιο της ανάλυσης δοκιμής δηλαδή test analysis, η ανάλυση δοκιμής στη δοκιμή λογισμικού είναι μια διαδικασία ελέγχου και ανάλυσης των τεχνουργημάτων δοκιμής προκειμένου να βασιστούν οι συνθήκες δοκιμής ή οι περιπτώσεις δοκιμής [36] [37]. Ο στόχος της ανάλυσης δοκιμής είναι να συγκεντρώσει απαιτήσεις και να καθορίσει τους στόχους της δοκιμής για να δημιουργήσει τη βάση των συνθηκών δοκιμής. Ως εκ τούτου, ονομάζεται επίσης βάση δοκιμής δηλαδή test basis.

Οι πηγές από τις οποίες αντλούνται πληροφορίες δοκιμής [38] θα μπορούσαν να είναι:

- Προδιαγραφές Απαιτήσεων Λογισμικού (Software Requirement Specification - SRS)
- Προδιαγραφές Επιχειρηματικών Απαιτήσεων (Business Requirement Specification - BRS)
- Έγγραφα Λειτουργικού Σχεδιασμού (Functional Design Documents)

Οι υπεύθυνοι δοκιμών (testers) μπορούν να δημιουργήσουν συνθήκες δοκιμής εξετάζοντας και διερευνώντας την εφαρμογή υπό δοκιμή ή χρησιμοποιώντας την εμπειρία τους. Αλλά ως επί το πλείστον, οι περιπτώσεις δοκιμής προέρχονται από τεχνουργήματα δοκιμής (test artifacts). Ένα σύνηθες και προτεινόμενο μοντέλο είναι το λεγόμενο V-Model, όπου διερευνάται και αναλύεται η βάση δοκιμής (test basis) και προκύπτουν οι συνθήκες δοκιμής (test conditions). Αυτό συμβαίνει κατά τις διαφορετικές φάσεις του V- Model. Το σχέδιο δοκιμής (test plan) και οι περιπτώσεις δοκιμής (test cases) δημιουργούνται χρησιμοποιώντας τα αντίστοιχα έγγραφα που είναι διαθέσιμα στις διαφορετικές φάσεις [39]. Η παρακάτω εικόνα αντιπροσωπεύει τη διαδικασία του μοντέλου V-Model για την δοκιμή (testing).



Εικόνα 19 Η διαδικασία του μοντέλου V-Model για την δοκιμή (testing).

Στο στάδιο αυτό, η συγκεκριμένη ανάλυση δοκιμής έχει ως είσοδο τις προδιαγραφές του πελάτη καθώς και τις λειτουργικές απαιτήσεις και τις απαιτήσεις συστήματος. Διακρίνονται οι λειτουργικές υποπεριοχές και ο τρόπος με τον οποίο η εφαρμογή πρέπει να λειτουργεί. Ορίζονται τα πεδία που ενδιαφέρουν τα σενάρια που πρόκειται να υλοποιηθούν, και πως αυτά μπορούν να συνδυαστούν [40] [41]. Με άλλα λόγια, η ανάλυση δοκιμών καθορίζει «τι πρέπει να δοκιμαστεί» από την άποψη της μετρήσιμης κάλυψης των κριτηρίων (που προκύπτουν από τις προδιαγραφές του πελάτη για την εφαρμογή που αναπτύσσεται και παράλληλα υπόκειται σε δοκιμή). Πραγματοποιείται και αξιολόγηση της βάσης δοκιμής (test basis) και των στοιχείων δοκιμής για τον εντοπισμό ελαττωμάτων (defects) διαφόρων τύπων, όπως: ασάφειες (ambiguities), παραλείψεις (omissions), ασυνέπειες (inconsistencies), ανακρίβειες (inaccuracies), αντιφάσεις (contradictions) και και περιττές δηλώσεις (superfluous statements). Γίνεται προσδιορισμός χαρακτηριστικών (features) και συνόλων χαρακτηριστικών προς δοκιμή και καθορίζεται η ιεράρχηση των συνθηκών δοκιμής για κάθε χαρακτηριστικό με βάση την ανάλυση της βάσης δοκιμής λαμβάνοντας υπόψη λειτουργικά, μη λειτουργικά και δομικά χαρακτηριστικά, άλλους επιχειρηματικούς και τεχνικούς παράγοντες και επίπεδα κινδύνου [1] [13].

2.3 Προετοιμασία και Σχεδιασμός Δοκιμής

Στο στάδιο αυτό δημιουργούνται τα σενάρια, τα οποία εξάγονται σύμφωνα με τις προδιαγραφές μέσα από την κατάλληλη προετοιμασία και τον σχετικό σχεδιασμό αυτών. Καθώς η τεχνική συσχετίζεται με τον αποτελεσματικό τρόπο να πετυχαίνεται κάτι, η τεχνική σχεδιασμού δοκιμής (test design) χρησιμοποιείται για την επιλογή ενός καλού συνόλου δοκιμών από όλες τις πιθανές δοκιμές για ένα δεδομένο σύστημα [42] [43]. Η ανάγκη για τη χρήση της τεχνικής σχεδιασμού δοκιμής (test design) έρχεται καθώς:

- Δεν είναι δυνατή η εξαντλητική δοκιμή, επομένως πρέπει να χρησιμοποιήσουμε τεχνικές σχεδιασμού δοκιμών για να μειώσουμε το μέγεθος της εισόδου.
- Η εξαντλητική δοκιμή είναι μια δοκιμαστική προσέγγιση στην οποία η σουίτα δοκιμών περιλαμβάνει όλους τους συνδυασμούς τιμών εισόδου και προϋποθέσεων.
- Οι εξαντλητικές δοκιμές δεν προτείνονται γιατί απαιτούν πολύ χρόνο και μεγάλο κόστος.

Υπάρχουν δύο κύριες κατηγορίες της τεχνικής σχεδιασμού δοκιμής (test design):

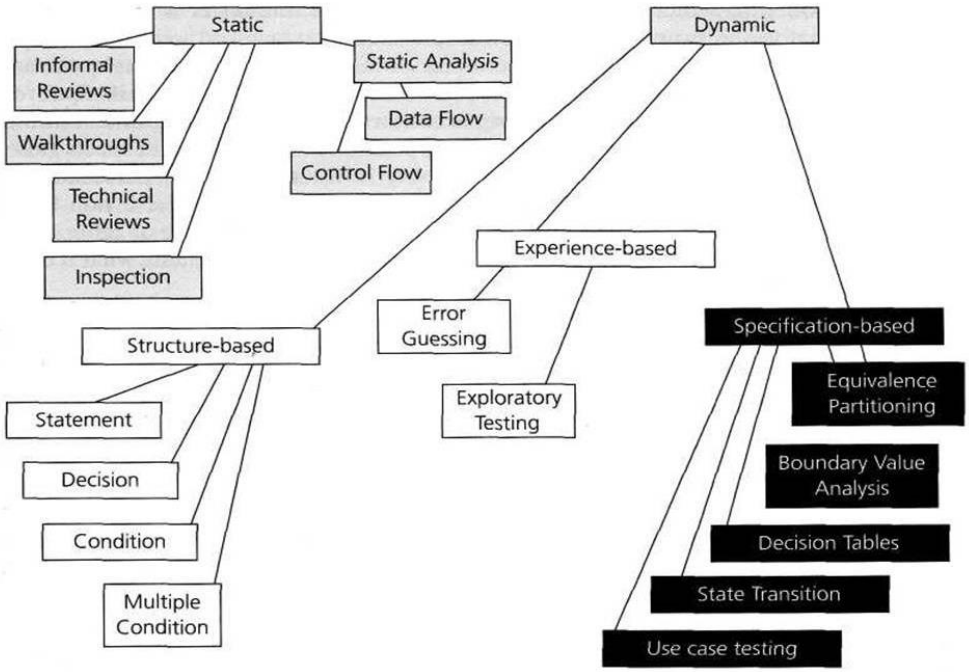
- Στατική τεχνική: Δοκιμή και έλεγχος των εγγράφων λογισμικού χειροκίνητα ή με ένα σύνολο εργαλείων αλλά χωρίς εκτέλεση του Λογισμικού. Στην τεχνική αυτή ανήκουν δύο βασικές υποκατηγορίες [44]:
  - Κριτικές (Reviews) -Χειροκίνητη εξέταση
  - Στατική Ανάλυση (Static Analysis) - Αυτοματοποιημένη Ανάλυση

- Δυναμική τεχνική: Είναι η τεχνική που το λογισμικό δοκιμάζεται εκτελώντας το σε υπολογιστή. Ο κύριες υποκατηγορίες αντιστοιχούν σε μία από τις παρακάτω περιπτώσεις που:
  - βασίζονται σε προδιαγραφές (Specification-based - Black-box Techniques) [30] [31]
  - βασίζονται στη δομή (Structure-based or White-box Techniques)
  - βασίζονται στην εμπειρία (Experience-based Techniques) [14].

Αναφορικά οι πιο δημοφιλείς και επικρατέστερες τεχνικές που βασίζονται σε προδιαγραφές (Black-box τεχνικές) [30] [31] για το σχεδιασμό δοκιμής (test design) είναι οι παρακάτω:

- Κατάτμηση ισοδυναμίας (Equivalence Partitioning)
- Ανάλυση οριακής αξίας (Boundary Value Analysis)
- Πίνακας Αποφάσεων (Decision Table Testing)
- Δοκιμές μετάβασης κατάστασης (State Transition Testing)
- Χρήση Δοκιμής περίπτωσης (Use Case Testing)

Οι στατικές και δυναμικές τεχνικές σχεδιασμού δοκιμής (test design) μπορούν να συνοψιστούν στην παρακάτω εικόνα:



Εικόνα 20 Οι στατικές και δυναμικές τεχνικές σχεδιασμού δοκιμής (test design).

Ένας λειτουργικός τρόπος γραφής σεναρίων (tests) είναι η γλώσσα Gherkin. Η γλώσσα αυτή υπάγεται στις BDD γλώσσες (Behaviour Driven Development) [45] [46] και προσφέρεται από το test framework που ονομάζεται Cucumber .

```
Feature: Set alarm
  As a light alarm user
  I must be able to update the alarm time
  So I can decide what time to wake up

  Background:
    Given I am on the homescreen

  Scenario: Set alarm time
    When I set the alarm to "9:03 AM"
    Then I should see the homePage alarm is set to "9:03 AM"
```

Εικόνα 21 Βασικό παράδειγμα Cucumber Gherkin γραφής σεναρίων.

Ο BDD τρόπος γραφής των σεναρίων δίνει αρχικά τη δυνατότητα στην ομάδα εσωτερικά που αναπτύσσει το προϊόν να έχει έναν κοινό κώδικα επικοινωνίας. Αναλυτές, προγραμματιστές και υπεύθυνοι δοκιμών (testers), καθώς και εξωτερικά μέλη που δεν ανήκουν στις προηγούμενες κατηγορίες (π.χ. πελάτης) χρησιμοποιούν έναν κοινό κώδικα επικοινωνίας την Gherkin γλώσσα. Καθώς τα BDD σενάρια μπορούν να αποτελούν παραδοτέο τμήμα του προϊόντος προς τον πελάτη αυτού, με στόχο να αξιολογεί την εταιρεία που αναπτύσσει το προϊόν αξιολογώντας τα σενάρια και διασφαλίζοντας πως οι απαιτήσεις είναι ορθώς αντιληπτές [47]. Έπειτα, η γλώσσα Gherkin δίνει τη δυνατότητα για την αυτοματοποίηση σεναρίων, χρησιμοποιώντας γλώσσα προγραμματισμού, με σκοπό να εκτελούνται τα tests συχνά και αποδοτικά [48], μειώνοντας το τελικό κόστος των εργατοωρών που απαιτούνται στην ανάπτυξη του προϊόντος, καθώς και μειώνει τα ελαττώματα (defects) που υπάρχουν σε αυτό, αναγνωρίζοντάς τα σχετικά έγκαιρα.

```
Scenario Outline: A user withdraws money from an ATM
    Given <Name> has a valid Credit or Debit card
    And their account balance is <OriginalBalance>
    When they insert their card
    And withdraw <WithdrawalAmount>
    Then the ATM returns <WithdrawalAmount>
    And their account balance is <NewBalance>

    Examples:
      | Name   | OriginalBalance | WithdrawalAmount | NewBalance |
      | Eric   | 100             | 45               | 55         |
      | Gaurav | 100             | 40               | 60         |
      | Ed     | 1000            | 200              | 800        |
```

Εικόνα 22 Παράδειγμα συνοπτικής αναπαράστασης πολλών Cucumber Gherkin σεναρίων.

Παρακάτω αναφέρονται σενάρια για τις τρεις βασικές σελίδες της στοιχηματικής εφαρμογής (σελίδα του πελάτη - customer page, σελίδα τοποθέτησης ενός στοιχήματος - bet placement page, σελίδα εξαργύρωσης ενός στοιχήματος- bet cashout page), σε γλώσσα Gherkin. Χρησιμοποιούνται οι βασικότερες από τις δεσμευμένες λέξεις της γλώσσας Gherkin: Σενάριο (Scenario), Δεδομένου (Given), Όταν (When), Τότε (Then), Και (And).

Στη σελίδα του πελάτη - customer page, τα βασικά σενάρια είναι τα ακόλουθα:

Σενάριο: Ο πελάτης Α μπορεί να δει τις πληροφορίες του
Δεδομένου Ο πελάτης Α είναι ενεργός
Όταν Ο πελάτης Α εισέρχεται στη σελίδα του λογαριασμού του
Τότε Ο πελάτης Α μπορεί να δει την περιοχή Personal Details
Και Ο πελάτης Α μπορεί να δει την περιοχή Bet Account Details

Το παραπάνω σενάριο είναι θετικό (positive scenario) και επαληθεύει την ορατότητα των πληροφοριών του πελάτη από τον ίδιο.

Σενάριο: Ο χρήστης Α όταν είναι διαχειριστής μπορεί να δει τις πληροφορίες άλλου χρήστη
Δεδομένου Ο χρήστης Α έχει λογαριασμό διαχειριστή
Όταν Ο χρήστης Α αναζητά τις πληροφορίες του χρήστης Β
Τότε Ο πελάτης Α μπορεί να δει τις πληροφορίες του χρήστη Β

Το παραπάνω σενάριο είναι θετικό (positive scenario) και διασφαλίζει την ποιότητα του προϊόντος επαληθεύοντας πως αυτό έχει αναπτυχθεί έτσι ώστε ένας διαχειριστής του συστήματος μπορεί να έχει πρόσβαση στις σελίδες άλλων χρηστών.

Στη σελίδα της τοποθέτησης στοιχήματος - bet placement page, αναλύονται τρία βασικά σενάρια:

Σενάριο: Ο πελάτης μπορεί να τοποθετήσει ένα στοίχημα
Δεδομένου Ο πελάτης Α έχει υπόλοιπο 1000 EUR
Όταν Ο πελάτης Α θέτει 500 EUR σε selectionA με απόδοση 2
Και Ο πελάτης πατάει "Place Bet"
Τότε Εμφανίζεται μήνυμα "Το στοίχημα τοποθετήθηκε επιτυχώς"
Και Ο πελάτης πετυχαίνει το στοίχημα
Και το Υπόλοιπο του πελάτη Α είναι 500 EUR

Το παραπάνω σενάριο είναι θετικό (positive scenario). Αυτό σημαίνει ότι αντιστοιχεί σε απαίτηση του πελάτη σχετικά με τη ροή του να είναι δυνατή η τοποθέτηση στοιχήματος κάτω από τις κατάλληλες προϋποθέσεις, δηλαδή να είναι ενεργός ο πελάτης, να έχει την άδεια (permission) να τοποθετεί ένα στοίχημα καθώς και να έχει υπόλοιπο χρηματικό που να είναι αρκετό έτσι ώστε να μπορεί να τοποθετεί ένα στοίχημα.

Σενάριο: Ο πελάτης δεν μπορεί να τοποθετήσει ένα στοίχημα όταν δεν έχει υπόλοιπο - negative edge case σενάριο
Δεδομένου Ο πελάτης Α έχει υπόλοιπο 499.99 EUR
Όταν Ο πελάτης Α θέτει 500 EUR σε selectionA με απόδοση 2
Και Ο πελάτης πατάει "Place Bet"
Τότε το Υπόλοιπο του πελάτη Α είναι 499.99 EUR
Και Εμφανίζεται μήνυμα "Το στοίχημα δεν τοποθετήθηκε επιτυχώς"

Το παραπάνω σενάριο είναι αρνητικό (negative scenario). Αποτελεί μια ακραία περίπτωση, ένα edge case, το οποίο αντιστοιχεί σε απαιτήσεις του πελάτη του προϊόντος τέτοιες, ώστε να μην είναι δυνατή η τοποθέτηση στοιχήματος όταν το υπόλοιπο δεν επαρκεί. Στο σενάριο αυτό υπάρχει η προϋπόθεση πως η ακρίβεια

δεκαδικών ψηφίων είναι δύο. Έτσι, το edge case δημιουργήθηκε με τέτοια ακρίβεια δεκαδικών (499.99 έναντι 500).

Σενάριο: Ο πελάτης δεν μπορεί να τοποθετήσει ένα στοίχημα σε GBP όταν το υπόλοιπό του είναι σε EUR - negative σενάριο
Δεδομένου Ο πελάτης Α έχει υπόλοιπο 1000 EUR
Όταν Ο πελάτης Α θέτει 500 GBP σε selectionA με απόδοση 2
Και Ο πελάτης πατάει "Place Bet"
Τότε το Υπόλοιπο του πελάτη Α είναι 1000 EUR
Και Εμφανίζεται μήνυμα "Το στοίχημα δεν τοποθετήθηκε επιτυχώς"

Το παραπάνω σενάριο είναι αρνητικό (negative scenario). Αντιστοιχεί σε προδιαγραφή (specification) όπου δεν είναι δυνατή η τοποθέτηση στοιχήματος σε διαφορετικό νόμισμα από το νόμισμα που έχει ορισθεί στην σελίδα του χρήστη.

Στη σελίδα της εξαργύρωσης ενός στοιχήματος- bet cashout page, τα βασικά σενάρια είναι τα παρακάτω:

Σενάριο: Ο πελάτης μπορεί να εξαργυρώσει το πλήρες ποσό επιτυχώς

Δεδομένου Ο πελάτης Α έχει υπόλοιπο 1000 EUR

Όταν Ο πελάτης Α θέτει 500 EUR σε selectionA με απόδοση 2

Και Ο πελάτης πατάει "Place Bet"

Τότε Εμφανίζεται μήνυμα "Το στοίχημα τοποθετήθηκε επιτυχώς"

Και Ο πελάτης κερδίζει το στοίχημα

Και το Υπόλοιπο του πελάτη Α είναι 1500 EUR

Όταν Ο πελάτης θέτει ποσό εξαργύρωσης 1500 EUR

Και Ο Πελάτης πατάει "Apply cashout"

Τότε Η εξαργύρωση γίνεται επιτυχώς

Και το Υπόλοιπο του πελάτη Α είναι 0 EUR

Το παραπάνω σενάριο είναι θετικό (positive scenario) και αντιστοιχεί σε πλήρη εξαργύρωση δηλαδή το ποσό εξαργύρωσης ισούται με το συνολικό διαθέσιμο

υπόλοιπο του πελάτη που βρίσκεται στον λογαριασμό του, κατά συνέπεια μετά την επιτυχή εξαργύρωση το διαθέσιμο υπόλοιπο του πελάτη θα είναι μηδενικό.

Σενάριο: Ο πελάτης μπορεί να εξαργυρώσει μερικό ποσό επιτυχώς

Δεδομένου Ο πελάτης Α έχει υπόλοιπο 1000 EUR

Όταν Ο πελάτης Α θέτει 500 EUR σε selectionA με απόδοση 2

Και Ο πελάτης πατάει "Place Bet"

Τότε Εμφανίζεται μήνυμα "Το στοίχημα τοποθετήθηκε επιτυχώς"

Και Ο πελάτης κερδίζει το στοίχημα

Και το Υπόλοιπο του πελάτη Α είναι 1500 EUR

Όταν Ο πελάτης θέτει ποσό εξαργύρωσης 1000 EUR

Και Ο Πελάτης πατάει "Apply cashout"

Τότε Η εξαργύρωση γίνεται επιτυχώς

Και το Υπόλοιπο του πελάτη Α είναι 500 EUR

Το παραπάνω σενάριο είναι θετικό (positive scenario) και αντιστοιχεί σε μερική εξαργύρωση, δηλαδή το ποσό εξαργύρωσης αποτελεί μέρος του συνολικού υπολοίπου του πελάτη που διαθέτει στον λογαριασμό του κατά συνέπεια μετά την επιτυχή μερική εξαργύρωση, ο πελάτης θα έχει διαθέσιμο υπόλοιπο στο λογαριασμό του.

Σενάριο: Ο πελάτης δεν μπορεί να εξαργυρώσει ποσό τιμής μεγαλύτερης του υπόλοιπού του

Δεδομένου Ο πελάτης Α έχει υπόλοιπο 1000 EUR

Όταν Ο πελάτης Α θέτει 500 EUR σε selectionA με απόδοση 2

Και Ο πελάτης πατάει "Place Bet"

Τότε Εμφανίζεται μήνυμα "Το στοίχημα τοποθετήθηκε επιτυχώς"

Και Ο πελάτης κερδίζει το στοίχημα

Και το Υπόλοιπο του πελάτη Α είναι 1500 EUR

Όταν Ο πελάτης θέτει ποσό εξαργύρωσης 3000 EUR

Και Ο Πελάτης πατάει “Apply cashout”
Τότε Η εξαργύρωση δεν γίνεται επιτυχώς
Και εμφανίζεται μήνυμα “Η εξαργύρωση δεν ολοκληρώθηκε”
Και το Υπόλοιπο του πελάτη Α είναι 1500 EUR

Το παραπάνω σενάριο είναι αρνητικό (negative scenario). Επαληθεύει πως δεν είναι δυνατή η εξαργύρωση ποσού μεγαλύτερου του επιτρεπτού, δηλαδή μεγαλύτερο το ποσό εξαργύρωσης από το συνολικό διαθέσιμο υπόλοιπο που διαθέτει ο πελάτης στον λογαριασμό του, κατα συνέπεια μετά την ανεπιτυχή εξαργύρωση το διαθέσιμο συνολικό υπόλοιπο του πελάτη θα είναι παραμείνει ίδιο.

Στην περίπτωση αυτοματοποιημένης δοκιμής (automation testing), ο υπεύθυνος δοκιμής (tester) υλοποιεί τα σενάρια σε κάποια γλώσσα προγραμματισμού όπως Java, C++, Python, Ruby, TypeScript και άλλες.

2.4 Εκτέλεση δοκιμής

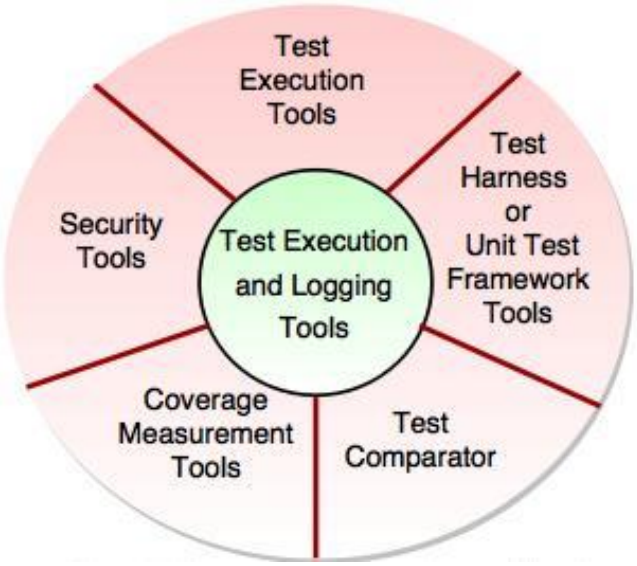
Στο βήμα αυτό, ο υπεύθυνος δοκιμής (tester) εκτελεί τα σενάρια (tests) που έχουν δημιουργηθεί κατά το βήμα της προετοιμασίας. Η εκτέλεση των σεναρίων (tests) υλοποιείται είτε χειροκίνητα (manually), με πράξεις δηλαδή χρησιμοποιείται κυρίως το ποντίκι του υπολογιστή όπως θα το έκανε ο χρήστης της τελικής έκδοσης της εφαρμογής. Απαραίτητη προϋπόθεση είναι η ύπαρξη - δημιουργία των δεδομένων δοκιμής (test data) από τον υπεύθυνο δοκιμή (tester). Αυτό σημαίνει πως απαιτείται να έχει ήδη δημιουργηθεί ένας χρήστης, ένας λογαριασμός, μία ιεραρχία από γεγονότα, αποδόσεις, στοιχήματα επιτρεπτά, και γενικότερα να έχουν τεθεί όρια, κανόνες και παράμετροι που απαιτούνται. Για παράδειγμα, για ένα απλό στοίχημα που αφορά ένα event αγώνα ποδοσφαίρου με συγκεκριμένη ημερομηνία και ώρα έναρξης του αγώνα, η τοποθέτηση του στοιχήματος για να είναι επιτυχής θα πρέπει να πραγματοποιηθεί πριν από την έναρξη του αγώνα. Στην περίπτωση του αυτοματοποιημένου testing, ο χρήστης εκτελεί τα σενάρια που έχει υλοποιήσει σε κάποια γλώσσα προγραμματισμού όπως Java, C++, Python, Ruby, TypeScript και άλλες.

Υπάρχει μία πληθώρα από εργαλεία εκτέλεσης δοκιμών (test execution) όπου ισχύουν τα παρακάτω:

- Τα εργαλεία εκτέλεσης δοκιμών (test execution tools) ονομάζονται και εργαλεία test running [49] καθώς βοηθούν στην αποτελεσματική εκτέλεση των δοκιμών.
- Τα εργαλεία εκτέλεσης δοκιμών (test execution tools) ξεκινούν με την εγγραφή (αποτύπωση) ή τη λήψη χειροκίνητων δοκιμών (manual tests), επομένως ονομάζονται εργαλεία λήψης ή αναπαραγωγής (capture - playback tools).
- Τα εργαλεία αυτά απαιτούν μια γλώσσα προγραμματισμού για την εκτέλεση τους. Συνεπώς είναι απαραίτητη η γνώση της εκάστοτε γλώσσας προγραμματισμού του εκάστοτε εργαλείου εκτέλεσης δοκιμών από τον υπεύθυνο δοκιμής (tester) που θα χρησιμοποιήσει το εκάστοτε εργαλείο καθώς και για τις περιπτώσεις που θα απαιτηθεί η δημιουργία νέων σεναρίων δοκιμής ή τροποποίηση των υπαρχόντων σεναρίων.

Τα κύρια χαρακτηριστικά των εργαλείων εκτέλεσης δοκιμών (test execution tools) είναι τα ακόλουθα:

- Καταγράφουν τις εισόδους δοκιμής (test inputs) κατά τη χειροκίνητη εκτέλεση των δοκιμών.
- Η αναμενόμενη έξοδος αποθηκεύεται με τη μορφή του αντικειμένου και αυτή η έξοδος συγκρίνεται με την επόμενη έξοδο.
- Αποθηκεύουν τα αποτελέσματα των δοκιμών [50] και τα κατηγοριοποιούν σε επιτυχή και μη επιτυχή αποτελέσματα (pass ή fail), και παρέχουν τις διαφορές μεταξύ αναμενόμενου (expected result) και πραγματικού αποτελέσματος (actual result).
- Υπολογίζουν το χρόνο που χρειάζονται για να γίνουν οι δοκιμές (δηλαδή πόσο χρόνο χρειάζεται για να τρέξει μια δοκιμή).
- Στέλνουν τη σύνοψη των αποτελεσμάτων στο εργαλείο διαχείρισης δοκιμών (test management tool) [51].



Εικόνα 23 Εργαλεία για την εκτέλεση δοκιμών και την καταγραφή των αποτελεσμάτων αυτών.

2.5 Αξιολόγηση κριτηρίων εξόδου και αναφορά

Κάθε ένα στάδιο δοκιμής, έχει καθορισμένα κριτήρια εισόδου (entry criteria) και εξόδου (exit criteria) καθώς και παραδοτέα (deliverables) που σχετίζονται με αυτό. Τα κριτήρια εισόδου (entry criteria) παρέχουν τα προαπαιτούμενα στοιχεία που πρέπει να συμπληρωθούν πριν ξεκινήσει η δοκιμή και τα κριτήρια εξόδου (exit criteria) ορίζουν τα στοιχεία που πρέπει να συμπληρωθούν [52] πριν ολοκληρωθεί η δοκιμή.

Τα κριτήρια εξόδου (exit criteria) είναι ένα σύνολο συμφωνημένων όρων με τα ενδιαφερόμενα μέρη βάσει των οποίων ορίζεται επίσημα η ολοκλήρωση της διαδικασία δοκιμής για ένα συγκεκριμένο επίπεδο δοκιμής. Είναι απαραίτητο για κάθε επίπεδο δοκιμής να ορίζονται τα κριτήρια εξόδου. Στην αξιολόγηση κριτηρίων εξόδου (exit criteria) αξιολογούμε την εκτέλεση της δοκιμής σε σχέση με τα καθορισμένα και συμφωνημένα κριτήρια εξόδου (exit criteria) για ένα συγκεκριμένο επίπεδο δοκιμής. Με βάση αυτήν την αξιολόγηση, μπορούμε να αποφασίσουμε εάν έχουμε κάνει όντως αρκετές δοκιμές για αυτό το επίπεδο για να το επισημάνουμε ως ολοκληρωμένο. Μερικά κοινά σημεία που υπάρχουν κυρίως στα κριτήρια εξόδου (exit criteria) είναι τα εξής:

- Εκτελούνται όλες οι δοκιμαστικές περιπτώσεις κρίσιμης, υψηλής και μέσης προτεραιότητας.
- Δεν υπάρχουν εκκρεμή ελαττώματα (defects) αποκλεισμού (blocker defects), κρίσιμης σημασίας ή υψηλής προτεραιότητας.
- Όλα τα ελαττώματα (defects) μέσης και χαμηλής προτεραιότητας (medium - low priority) ελέγχονται και συμφωνείται ότι θα διαφοροποιηθούν ή θα επιδιορθωθούν.
- Οι συμφωνημένες διορθώσεις για ελαττώματα (defects) μέσης και χαμηλής προτεραιότητας (medium - low priority) ολοκληρώθηκαν και δοκιμάστηκαν ξανά.
- Όλη η τεκμηρίωση δοκιμών (test documentation), όπως το σχέδιο δοκιμών (test plan), οι περιπτώσεις δοκιμών (test cases) είναι καταγεγραμμένα και πάντα ενημερωμένα στο εργαλείο διαχείρισης δοκιμών (test management tool).

Οι κύριες εργασίες για την αξιολόγηση των κριτηρίων εξόδου (exit criteria) είναι οι εξής:

- Έλεγχος των αρχείων καταγραφής δοκιμών σε σχέση με τα καθορισμένα κριτήρια εξόδου (exit criteria). Για παράδειγμα, έλεγχος της προόδου εκτέλεσης της δοκιμής για όλες τις κρίσιμες, μεγάλες και μεσαίες περιπτώσεις δοκιμών. Έλεγχος της κατάστασης όλων των εκκρεμών ελαττωμάτων (defects) και πόσων πρέπει να επιδιορθωθούν και να επανεξεταστούν για να πληρούνται τα κριτήρια εξόδου (exit criteria).
- Δεύτερον, πρέπει να γίνεται μια αξιολόγηση στο εάν υπάρχουν περισσότερες δοκιμές που πρέπει να εκτελεστούν λόγω των ανησυχιών για την ποιότητα του προϊόντος. Ίσως να δημιουργείται η ανάγκη για προσθήκη περισσότερων δοκιμών για εκτέλεση με σκοπό να μειωθεί ο κίνδυνος για μη ικανοποιητική ποιότητα που καθορίζεται στα κριτήρια εξόδου (exit criteria). Μερικές φορές υπάρχει το ενδεχόμενο να χρειαστεί να γίνει τροποποίηση των κριτηρίων εξόδου (exit criteria) με τη σύμφωνη γνώμη των ενδιαφερομένων μερών, εάν είχαν καθοριστεί αρχικά πολύ αυστηρά κριτήρια, αλλά στο τρέχον στάδιο του έργου αυτά τα κριτήρια ενέχουν ελάχιστο κίνδυνο, έτσι ώστε να πληρούνται τα κριτήρια εξόδου (exit criteria).

- Συγγραφή της συνοπτικής έκθεσης δοκιμής (test summary report). Μετά την ολοκλήρωση της αξιολόγησης των κριτηρίων εξόδου (exit criteria) πρέπει να γνωρίζουν το αποτέλεσμα όχι μόνο η ομάδα ανάπτυξης (development team) και η ομάδα δοκιμών (test team), αλλά και το ευρύτερο σύνολο ενδιαφερομένων πρέπει να γνωρίζουν τι συνέβη στα επίπεδα δοκιμών, ώστε να μπορούν να λαμβάνουν αποφάσεις σχετικά με το λογισμικό. Για παράδειγμα, είναι καλό να ξεκινήσει το επόμενο επίπεδο δοκιμής ή εάν έχουν ολοκληρωθεί όλα τα επίπεδα δοκιμής μπορεί το λογισμικό να ξεκινήσει να λειτουργεί στην παραγωγή.

ΚΕΦΑΛΑΙΟ 3ο

3. Αποτελέσματα και Συμπεράσματα

3.1 Αποτελέσματα

Τα αποτελέσματα αποτελούν γραφικές παραστάσεις σύμφωνα με τα δεδομένα που αντιστοιχούν σε συνιστώσες σχετικά με τον κύκλο ζωής του στοιχηματικού προϊόντος - λογισμικού, τις παραδοτέες προς τον πελάτη λειτουργικότητες, τις ημερομηνίες εκκίνησης και τερματισμού της εκάστοτε διαδικασίας καθώς και άλλες μονάδες όπως αυτή της έκδοσης εφαρμογής (application version), των story points, της προτεραιότητας (priority) και της σοβαρότητας (severity).

Τέλος, σημαντικό ρόλο παίζουν οι παράμετροι του είδους της δοκιμής (testing) σε κάθε ticket (όπως manual testing, automation testing, not tested), το δυναμικό της ομάδας (ο αριθμός των ατόμων που αντιστοιχούν στην εκάστοτε υλοποίησης εργασιών), το εάν ο πελάτης ήταν ικανοποιημένος με την αντίστοιχη εργασία, οι ώρες που έχουν κερδηθεί έπειτα από δοκιμή (testing), και εν τέλει εάν ο πελάτης μπόρεσε να χρησιμοποιήσει επιτυχώς την εκάστοτε λειτουργικότητα.

Το σύνολο των δεδομένων που έχουν εξαχθεί έχουν αποθηκευτεί σε ένα αρχείο excel, στην συνέχεια έχουν αναλυθεί, επεξεργαστεί και αποκωδικοποιηθεί και μπορούν να απεικονιστούν στους πίνακες που ακολουθούν:

Sprint Number	Ticket Id	Functional Area	Ticket title	Ticket type Story: 1 Defect: 2	Start date	End date	Dates diff	Application version	Story points	Priority	Severity	Testing Default manual testing: 1 Automation testing: 2 Exploratory manual testing: 3 Not tested: 4	Team capacity power (incl. developers, testers, business analysts team members)	Customer satisfied with delivery Yes: 1 No: 2	Human hours spent (actually when testing activities do not exists)	Customer can use bet application process successfully Yes: 1 No: 2
201	CT-9381	Customer page	Add "Name" field in section "Personal Details" in Customer page	1	01/11/2021	11/11/2021	10	CB v.1.040	1	High	Moderate	1	4	1	1	1
201	CT-9382	Customer page	Add "Surname" field in section "Personal Details" in Customer page	1	02/11/2021	10/11/2021	8	CB v.1.041	2	Medium	Moderate	4	6	2	15	2
201	CT-9383	Customer page	Remove "Nickname" field from section "Personal Details" in Customer page	1	05/11/2021	12/11/2021	7	CB v.1.042	3	Medium	Minor	2	5	1	1	1
201	CT-9384	Customer page	The field "Nickname" should not be visible in section "Personal Details" in Customer page	2	09/11/2021	10/11/2021	1	CB v.1.043	1	Medium	Minor	4	1	2	12	2
202	BT-3950	Bet placement page	Add "Account Id" field in section Bet placement platform	1	15/11/2021	25/11/2021	10	BE v.3.201	1	Medium	Medium	1	4	1	0	1
202	BT-3951	Bet placement page	Remove"Customer Id" field from section Bet placement platform	1	24/11/2021	01/12/2021	7	BE v.3.202	2	Medium	Medium	1	8	1	2	1
202	BT-3952	Bet placement page	The "Customer Id" field should not be visible in section Bet placement platform	2	25/11/2021	28/11/2021	1	BE v.3.203	1	Medium	Medium	4	2	2	11	1
203	CT-9385	Customer page	Add "E-mail" field in section "Personal Details" in Customer page	1	02/12/2021	03/12/2021	1	CB v.1.044	5	Medium	Critical	3	7	1	0	1
203	CT-9386	Customer page	Add "Gender" field in section "Personal Details" in Customer page	1	06/12/2021	11/12/2021	5	CB v.1.045	1	Low	Moderate	4	4	2	14	2
203	CT-9387	Customer page	Add "Address" field in section "Personal Details" in Customer page	1	08/12/2021	17/12/2021	9	CB v.1.046	2	Low	Moderate	1	8	1	0	1
204	BT-3953	Bet placement page	Add the dropdown list with name "Category Type" in section Bet placement platform	1	03/01/2022	07/01/2022	4	BE v.3.204	1	Medium	Medium	1	6	1	0	1
204	BC-1078	Bet cashout page	Add field "Account id" in section Bet cashout platform	1	03/01/2022	12/01/2022	9	CA v.2.199	8	Medium	Medium	2	6	1	1	1
204	BC-1079	Bet cashout page	Remove field "Customer Id" from section Bet cashout platform	1	04/01/2022	05/01/2022	1	CA v.2.200	5	High	Medium	4	2	2	13	1
204	BC-1080	Bet cashout page	Add field "Cashout for" in section Bet cashout platform	1	04/01/2022	11/01/2022	7	CA v.2.201	5	Medium	Medium	2	6	1	1	1
204	BC-1081	Bet cashout page	Add the dropdown list with name "All remaining bets" next to field "Cashout for" in section Bet cashout platform	1	10/01/2022	12/01/2022	2	CA v.2.202	5	High	High	4	3	2	9	1
205	CT-9388	Customer page	Address is not displayed properly in section "Personal Details" in Customer page	2	13/01/2022	18/01/2022	5	CB v.1.047	2	High	Moderate	4	3	2	8	2
205	CT-9389	Customer page	Add "Country" field in section "Personal Details" in Customer page	1	17/01/2022	28/01/2022	11	CB v.1.048	2	Medium	Moderate	1	7	1	2	1
205	CT-9390	Customer page	Add "Telephone" field in section "Personal Details" in Customer page	1	26/01/2022	31/01/2022	5	CB v.1.049	3	Medium	Moderate	2	5	1	0	1
206	CT-9391	Customer page	Remove "Telephone 2" field from section "Personal Details" in Customer page	1	03/02/2022	11/02/2022	8	CB v.1.050	3	Medium	Moderate	2	8	1	0	1
206	BC-1082	Bet cashout page	Add field "Cashout ammount" in section Bet cashout platform	1	03/02/2022	11/02/2022	8	CA v.2.203	5	Medium	Medium	2	7	1	0	1

Πίνακας 3 Δεδομένα της στοιχηματικής εφαρμογής που υπόκειται σε δοκιμές - μέρος 1ο.

Sprint Number	Ticket Id	Functional Area	Ticket title	Ticket type Story: 1 Defect: 2	Start date	End date	Dates diff	Application version	Story points	Priority	Severity	Testing Default manual testing: 1 Automation testing: 2 Exploratory manual testing: 3 Not tested: 4	Team capacity power (incl. developers, testers, business analysts team members)	Customer satisfied with delivery Yes: 1 No: 2	Human hours spent (actually when testing activities do not exists)	Customer can use bet application process successfully Yes: 1 No: 2
206	BC-1083	Bet cashout page	The value in "Cashout ammount" field is not displayed properly in section Bet cashout platform	2	07/02/2022	09/02/2022	2	CA v.2.204	5	Medium	High	3	2	1	2	1
206	CT-9392	Customer page	The "Telephone 2" field should not be visivle in section "Personal Details" in Customer page	1	04/02/2022	05/02/2022	1	CB v.1.051	2	High	Moderate	3	7	1	0	1
206	BT-3954	Bet placement page	Add "Event Id" field in section Bet placement platform	1	01/02/2022	14/02/2022	13	BE v.3.205	5	Medium	Medium	2	6	1	0	1
206	BT-3955	Bet placement page	The "Event Id" field does not exist in section Bet placement platform	2	04/02/2022	11/02/2022	7	BE v.3.206	3	Medium	Medium	4	2	2	15	1
207	BT-3956	Bet placement page	Add "Market Type Id" field in section Bet placement platform	1	14/02/2022	21/02/2022	7	BE v.3.207	2	Low	Medium	1	7	1	0	1
207	CT-9393	Customer page	Add the dropdown list with name "Profile" in section "Bet Account Details" in Customer page	1	23/02/2022	28/02/2022	5	CB v.1.052	2	Medium	Moderate	1	9	1	0	2
207	BC-1084	Bet cashout page	Add field "Cashout IBAN" in section Bet cashout platform	1	15/02/2022	01/03/2022	14	CA v.2.205	5	Medium	Medium	2	5	1	0	1
207	BC-1085	Bet cashout page	The value in "Cashout IBAN" field is not displayed properly in section Bet cashout platform	2	16/02/2022	02/03/2022	14	CA v.2.206	3	Medium	Medium	1	2	1	0	1
208	CT-9394	Customer page	Add "Account Id" field in section "Bet Account Details" in Customer page	1	07/03/2022	10/03/2022	3	CB v.1.053	1	Medium	Critical	3	5	1	1	1
209	CT-9395	Customer page	Remove "Customer Id" field from section "Bet Account Details" in Customer page	1	11/03/2022	23/03/2022	12	CB v.1.054	3	Medium	Minor	2	4	1	0	1
209	CT-9396	Customer page	Add "Level" field in section "Bet Account Details" in Customer page	1	14/03/2022	31/03/2022	17	CB v.1.055	1	Medium	Moderate	4	6	2	21	2
209	CT-9397	Customer page	The "Level" field should exist in section "Bet Account Details" in Customer page	2	17/03/2022	19/03/2022	2	CB v.1.056	3	Medium	Moderate	4	2	2	16	2
210	BT-3957	Bet placement page	Add "Selection Id" field in section Bet placement platform	1	04/04/2022	13/04/2022	9	BE v.3.208	2	Low	Medium	1	9	1	0	1
210	BT-3958	Bet placement page	Add "Stake" field in section Bet placement platform	1	04/04/2022	15/04/2022	11	BE v.3.209	2	Medium	Medium	1	6	1	0	1
211	BC-1086	Bet cashout page	Add new field "Two factor authentication code (mobile verification)" in section Bet cashout platform	1	18/04/2022	22/04/2022	4	CA v.2.207	5	Medium	Medium	2	7	1	0	1
211	CT-9398	Customer page	Add "Account Creation Date" field in section "Bet Account Details" in Customer page	1	25/04/2022	05/05/2022	10	CB v.1.057	1	Medium	Moderate	1	7	1	0	1
212	CT-9399	Customer page	Add "Account Expiration Date" field in section "Bet Account Details" in Customer page	1	06/05/2022	18/05/2022	12	CB v.1.058	1	Medium	Moderate	1	6	1	2	1
213	CT-9400	Customer page	Add "Total bets placed" field in section "Bet Account Details" in Customer page	1	20/05/2022	25/05/2022	5	CB v.1.059	2	Medium	Moderate	1	8	1	2	1
213	BT-3959	Bet placement page	Add the dropdown list with name "Currency" in section Bet placement platform	1	19/05/2022	30/05/2022	11	BE v.3.210	2	Low	Medium	1	5	1	0	1

Πίνακας 4 Δεδομένα της στοιχηματικής εφαρμογής που υπόκειται σε δοκιμές – μέρος 2ο.

Sprint Number	Ticket Id	Functional Area	Ticket title	Ticket type Story: 1 Defect: 2	Start date	End date	Dates diff	Application version	Story points	Priority	Severity	Testing Default manual testing: 1 Automation testing: 2 Exploratory manual testing: 3 Not tested: 4	Team capacity power (incl. developers, testers, business analysts team members)	Customer satisfied with delivery Yes: 1 No: 2	Human hours spent (actually when testing activities do not exists)	Customer can use bet application process successfully Yes: 1 No: 2
213	CT-9401	Customer page	Add "Total Winnings" field in section "Bet Account Details" in Customer page	1	20/05/2022	27/05/2022	7	CB v.1.060	2	Low	Moderate	1	8	1	2	1
214	BC-1087	Bet cashout page	Add new field name "I consent to the use of my data and current transcation rules policy" with checkbox in section Bet cashout platform	1	30/05/2022	10/06/2022	11	CA v.2.208	1	Medium	Medium	1	8	1	0	1
214	BT-3960	Bet placement page	Add the dropdown list with name "Repeat" in section Bet placement platform	1	30/05/2022	11/06/2022	12	BE v.3.211	1	Low	Medium	1	7	1	0	1
214	CT-9402	Customer page	Add "Bonus Type" field in section "Bet Account Details" in Customer page	1	08/06/2022	11/06/2022	5	CB v.1.061	5	High	Minor	2	6	1	2	1
214	CT-9403	Customer page	The "Bonus Type" field does not exist in section "Bet Account Details" in Customer page	2	08/06/2022	10/06/2022	2	CB v.1.062	1	High	Cosmetic	4	3	2	11	2
215	CT-9404	Customer page	Add "Freebet Points Left" field in section "Bet Account Details" in Customer page	1	13/06/2022	20/06/2022	7	CB v.1.063	2	High	Moderate	1	8	1	0	1
216	CT-9405	Customer page	Add "Balance" field in section "Bet Account Details" in Customer page	1	20/06/2022	30/06/2022	10	CB v.1.064	3	High	Minor	2	9	1	2	1
216	CT-9406	Customer page	Balance is not displayed properly in section "Bet Account Details" in Customer page	2	22/06/2022	27/06/2022	5	CB v.1.065	2	High	Minor	4	2	2	14	1
216	BC-1088	Bet cashout page	Add "Apply cashout" button in section Bet placement platform	1	22/06/2022	24/06/2022	2	CA v.2.209	5	High	Medium	4	3	2	11	1
216	BC-1089	Bet cashout page	The "Apply cashout" button is not clickable in section Bet placement platform	2	24/06/2022	30/06/2022	6	CA v.2.210	1	Medium	Medium	1	3	1	0	1
217	CT-9407	Customer page	Add "Refund" field in section "Bet Account Details" in Customer page	1	01/07/2022	11/07/2022	10	CB v.1.066	1	Low	Moderate	1	6	1	2	1
217	CT-9408	Customer page	Add "Currency" field in section "Bet Account Details" in Customer page	1	11/07/2022	14/07/2022	3	CB v.1.067	2	High	Moderate	3	8	1	1	1
217	CT-9409	Customer page	The "Currency" field does not exist insection "Bet Account Details" in Customer page	2	12/07/2022	13/07/2022	1	CB v.1.068	2	Medium	Moderate	1	3	1	1	1
218	BT-3961	Bet placement page	Add "In Running" field in section Bet placement platform	1	15/07/2022	22/07/2022	7	BE v.3.212	2	Medium	Medium	1	7	1	0	1
218	BT-3962	Bet placement page	Add "Load and Add Selection" field in section Bet placement platform	1	25/07/2022	29/07/2022	4	BE v.3.213	2	Medium	Medium	4	4	2	12	1
218	BT-3963	Bet placement page	The "Load and Add Selection" field does not exist in section Bet placement platform	2	26/07/2022	28/07/2022	2	BE v.3.214	1	Medium	Medium	1	1	1	0	1
219	CT-9410	Customer page	Add "Stake factor" field in section "Bet Account Details" in Customer page	1	01/08/2022	08/08/2022	7	CB v.1.069	1	Low	Moderate	1	8	1	2	2
219	CT-9411	Customer page	Add "BIR" field in section "Bet Account Details" in Customer page	1	08/08/2022	12/08/2022	4	CB v.1.070	2	Medium	Moderate	1	5	1	0	1
220	CT-9412	Customer page	Remove "Stake percentage" field from section "Bet Account Details" in Customer page	1	16/08/2022	19/08/2022	3	CB v.1.071	2	Medium	Major	3	7	1	0	1

Πίνακας 5 Δεδομένα της στοιχηματικής εφαρμογής που υπόκειται σε δοκιμές – μέρος 3ο.

Sprint Number	Ticket Id	Functional Area	Ticket title	Ticket type Story: 1 Defect: 2	Start date	End date	Dates diff	Application version	Story points	Priority	Severity	Testing Default manual testing: 1 Automation testing: 2 Exploratory manual testing: 3 Not tested: 4	Team capacity power (incl. developers, testers, business analysts team members)	Customer satisfied with delivery Yes: 1 No: 2	Human hours spent (actually when testing activities do not exists)	Customer can use bet application process successfully Yes: 1 No: 2
220	CT-9413	Customer page	The "Stake percentage" field sould not be visible in section "Bet Account Details" in Customer page	2	17/08/2022	18/08/2022	1	CB v.1.072	1	Medium	Moderate	1	1	1	2	1
220	BT-3964	Bet placement page	Add "Place bet" button in section Bet placement platform	1	22/08/2022	25/08/2022	3	BE v.3.215	2	High	Medium	3	6	1	0	1
221	BT-3965	Bet placement page	The "Place bet" button is not clickable in section Bet placement platform	2	29/08/2022	31/08/2022	5	BE v.3.216	2	Medium	Medium	1	2	1	0	1

Πίνακας 6 Δεδομένα της στοιχηματικής εφαρμογής που υπόκειται σε δοκιμές – μέρος 4ο.

Τα παραπάνω δεδομένα είναι αποτέλεσμα, κυρίως από το πρόγραμμα Jira. Τα στοιχεία καταγράφονται στο πρόγραμμα αυτό από τα άτομα που ανήκουν στις ομάδες ανάπτυξης (development) και δοκιμής (testing) της εφαρμογής αυτής. Στο πλαίσιο της διπλωματικής εργασίας αυτής έγινε επιπλέον επεξεργασία προσθαφαιρώντας στήλες και σχετική τροποποίηση με σκοπό την αποφυγή διαρροής στοιχείων που αποτελούν ιδιωτική ιδιοκτησία. Το Jira αποτελεί σήμερα το δημοφιλέστερο ticket management tool, ένα εργαλείο διαχείρισης των λειτουργικοτήτων δηλαδή που αφορούν το σύστημα που υποβάλλεται σε δοκιμή (testing), καθώς και το ίδιο το testing σύστημα. Ο παραπάνω πίνακας, που λόγω μεγάλου μεγέθους έχει αποτυπωθεί σε τμήματα στις παραπάνω εικόνες, αποτελεί ένα υποσύνολο των tickets στο πλαίσιο της στοιχηματικής εφαρμογής, καθώς τα tickets είναι της τάξεων ορισμένων εκατοντάδων.

Στο σημείο αυτό παρουσιάζονται αναλυτικά οι στήλες του πίνακα Δεδομένων. Συγκεκριμένα, η στήλη Sprint Number, αντιστοιχεί στον αριθμό του sprint, όπως έχουμε αναφέρει για το sprint στο θεωρητικό μέρος της διπλωματικής αυτής, αφορά την προκαθορισμένη χρονική περίοδο (στην συγκεκριμένη περίπτωση είναι δύο βδομάδες) που έχει ορισθεί σύμφωνα με την agile μεθοδολογία που έχει ακολουθηθεί. Κάθε sprint περιλαμβάνει μια διαλειτουργική ομάδα που εργάζεται σε όλες τις λειτουργίες: προγραμματισμό (planning), ανάλυση απαιτήσεων (requirements analysis), σχεδιασμό (design), κωδικοποίηση (coding), δοκιμή μονάδας (testing) και δοκιμή αποδοχής (acceptance testing). Στο τέλος του sprint το προϊόν εργασίας (μέρος ή τμήμα του λογισμικού - εφαρμογής) επιδεικνύεται στα ενδιαφερόμενα μέρη. Η αρίθμηση το sprint αυξάνεται κατά μία μονάδα κάθε φορά που ολοκληρώνεται η περίοδος του sprint και ξεκινάει το επόμενο sprint. Η στήλη Sprint Number, είναι συνδεδεμένη με τις στήλες Start date και End date που έχουν την μορφή ημερομηνίας (π.χ. για το sprint 201 έχουμε τις τιμές Start date = 01/11/2021, End date = 12/11/2021 και τις ενδιάμεσες τιμές αυτής της περιόδου- sprint). Οι τιμές που θα έχουν οι στήλες Start date και End date πρέπει να ανήκουν στην καθορισμένη περίοδο του εκάστοτε sprint, για παράδειγμα στην παρακάτω εικόνα το sprint 201 είναι τις δύο πρώτες εβδομάδες του Νοεμβρίου 2021 και το sprint 202 είναι τις δύο τελευταίες εβδομάδες του Νοεμβρίου 2021.

Sprint Number	Ticket Id	Functional Area	Ticket title	Ticket type Story: 1 Defect: 2	Start date	End date
201	CT-9381	Customer page	Add "Name" field in section "Personal Details" in Customer page	1	01/11/2021	11/11/2021
201	CT-9382	Customer page	Add "Surname" field in section "Personal Details" in Customer page	1	02/11/2021	10/11/2021
201	CT-9383	Customer page	Remove "Nickname" field from section "Personal Details" in Customer page	1	05/11/2021	12/11/2021
201	CT-9384	Customer page	The field "Nickname" should not be visible in section "Personal Details" in Customer page	2	09/11/2021	10/11/2021
202	BT-3950	Bet placement page	Add "Account Id" field in section Bet placement platform	1	15/11/2021	25/11/2021
202	BT-3951	Bet placement page	Remove"Customer Id" field from section Bet placement platform	1	24/11/2021	01/12/2021
202	BT-3952	Bet placement page	The "Customer Id" field should not be visible in section Bet placement platform	2	25/11/2021	26/11/2021

Πίνακας 7 Οι στήλες Sprint Number, Start date και End date του πίνακα Δεδομένων.

Για κάθε sprint ορίζονται ένα σύνολο από tickets που πρόκειται να υλοποιηθούν κατά την διάρκεια του συγκεκριμένου sprint, αυτή η διαδικασία (καθορισμός των tickets) γίνεται πριν την έναρξη της χρονικής περιόδου του sprint, για παράδειγμα στην παραπάνω εικόνα τα tickets για το sprint 201 έχουν οριστεί πριν την χρονική περίοδο έναρξης του sprint 201 δηλαδή πριν από την 01/11/2021. Η στήλη Ticket Id αποτελείται από μοναδικές τιμές (δηλαδή δεν υπάρχουν και δεν θα πρέπει να υπάρχουν δύο ίδιες Ticket Id τιμές) και είναι άρρηκτα συνδεδεμένη με την στήλη Ticket title. Η στήλη Ticket Id έχει τιμές με συγκεκριμένη μορφή που ακολουθούν την εκάστοτε πολιτική που έχει οριστεί, στην προκειμένη περίπτωση είναι μια ακολουθία αλφαριθμητικών χαρακτήρων. Η στήλη Ticket title αντιστοιχεί στον τίτλο του εκάστοτε ticket, όπως αυτό αναγράφεται στην εφαρμογή Jira. Η στήλη Ticket title είναι στην ουσία μια συνοπτική περιγραφή με την μορφή τίτλου για το τι πρόκειται να υλοποιηθεί στο συγκεκριμένο ticket. Όπως αναφέρθηκε οι τιμές Ticket Id είναι μοναδικές, αυτό συμβαίνει με σκοπό να είναι εύκολη η αναζήτηση των tickets σε περίπτωση που κάποιος επιθυμεί να ανατρέξει στο αντίστοιχο ticket και να πληροφορηθεί σχετικά με το τι υλοποιήθηκε ή τροποποιήθηκε στο συγκεκριμένο ticket για την εκάστοτε λειτουργικότητα. Η λειτουργικότητα αυτή, στο πλαίσιο της διπλωματικής εργασίας, είναι είτε μία νέα παραδοτέα, είναι μία προβληματική που προέκυψε, και έχει αναφερθεί είτε από εσωτερική πηγή (εντός της ομάδας ανάπτυξης), είτε από τον πελάτη που την ανακάλυψε.

Το κάθε ticket με μοναδικό Ticket Id αναφέρεται σε μια και μόνο μια μοναδική λειτουργική περιοχή, συνεπώς η στήλη Functional area αποτελεί τη λειτουργική περιοχή που θα αναπτυχθεί και τροποποιηθεί σύμφωνα με την περιγραφή και τις απαιτήσεις που αναλύονται στο συγκεκριμένο ticket. Στη διπλωματική εργασία εξετάστηκαν οι λειτουργικές περιοχές της σελίδας του πελάτη (customer page), της σελίδας τοποθέτησης στοιχήματος (bet placement page) καθώς και της σελίδας εξαργύρωσης στοιχήματος (bet cashout page). Συνεπώς οι τιμές που μπορεί να πάρει η στήλη Functional area είναι Customer page, Bet placement page, Bet cashout page.

Ανάλογα με την περιγραφή και τις απαιτήσεις που αναλύονται στο συγκεκριμένο ticket, τα tickets για την διεκπεραίωση της διπλωματικής εργασίας κατηγοριοποιούνται σε δύο κύριες κατηγορίες, Story και Defect. Συνεπώς η στήλη Ticket Type αφορά τον τύπο του ticket (Story: 1, Defect: 2). Οι τιμές που μπορεί να

πάρει η στήλη Ticket Type είναι 1,2. Το Story είναι συχνά μια νέα δυνατότητα (feature) ή μια πρόσθετη λειτουργία (functionality). Δημιουργούνται τα stories για τις νέες απαιτήσεις και ανάγκες του πελάτη σχετικά με την εφαρμογή που αναπτύσσεται και δοκιμάζεται. Γενικά είναι κάτι που θεωρείται ότι προσθέτει αξία στο προϊόν δηλαδή στην στοιχηματική εφαρμογή. Το Defect είναι ένα ελάττωμα- σφάλμα σε μια λειτουργικότητα που δεν επιτρέπει την ολοκλήρωση της επιδιωκόμενης ενέργειας-λειτουργίας της εκάστοτε λειτουργικότητας. Πολλά Defects δημιουργούνται κατά την υλοποίηση ενός Story, γι αυτό και υπάρχει η σχέση ένα Story σε πολλά Defects. Στην παρακάτω εικόνα παρουσιάζονται οι σχέσεις των στηλών Ticket Id, Functional Area, Ticket title και Ticket type, και έχουν επισημανθεί τα δύο διαφορετικά Ticket Type, με το πορτοκαλί πλαίσιο είναι για το Story και με το κόκκινο πλαίσιο είναι για το Defect.

Sprint Number	Ticket Id	Functional Area	Ticket title	Ticket type Story: 1 Defect: 2
204	BT-3953	Bet placement page	Add the dropdown list with name "Category Type" in section Bet placement platform	1
204	BC-1078	Bet cashout page	Add field "Account Id" in section Bet cashout platform	1
204	BC-1079	Bet cashout page	Remove field "Customer Id" from section Bet cashout platform	1
204	BC-1080	Bet cashout page	Add field "Cashout for" in section Bet cashout platform	1
204	BC-1081	Bet cashout page	Add the dropdown list with name "All remaining bets" next to field "Cashout for" in section Bet cashout platform	1
205	CT-9388	Customer page	Address is not displayed properly in section "Personal Details" in Customer page	2

Πίνακας 8 Οι στήλες Ticket Id, Functional Area, Ticket title και Ticket type του πίνακα Δεδομένων.

Όπως έχουμε εξηγήσει και παραπάνω, σε κάθε sprint έχουν οριστεί ένα σύνολο από tickets που πρόκειται να υλοποιηθούν κατά την διάρκεια του συγκεκριμένου sprint, αυτό συνήθως (η επιλογή των tickets) γίνεται πριν την έναρξη του εκάστοτε sprint. Η επιλογή όμως των tickets που θα επιλεχθούν για το sprint, γίνεται με βάση τα Story points, το Priority και το Severity του κάθε ticket. Τα story points έχουν τιμές που προκύπτουν από την ακολουθία Fibonacci. Η σειρά Fibonacci αποτελείται από μια

ακολουθία αριθμών όπου κάθε αριθμός είναι ένα άθροισμα των δύο προηγούμενων αριθμών. Η παραδοσιακή ακολουθία Fibonacci είναι 1, 2, 3, 5, 8, 13, 21, 34 και ούτω καθεξής. Στα agile έργα, αυτή η σειρά τροποποιείται. Η τροποποιημένη σειρά Fibonacci είναι 0, 1, 2, 3, 5, 8, 13, 20, 40, 100 είναι η ακολουθία που χρησιμοποιείται για την εκτίμηση του σχετικού μεγέθους των tickets σε story points. Ένα βασικό πλεονέκτημα της εφαρμογής της κλίμακας Fibonacci σε περιβάλλοντα agile είναι ο τρόπος με τον οποίο δημιουργεί χώρο για τα μέλη της ομάδας και τους διαχειριστές έργων να δουν ρεαλιστικά την προσπάθεια που απαιτείται για την ολοκλήρωση κάθε εργασίας σε έναν κύκλο sprint. Αυτό οδηγεί σε πιο ακριβείς εκτιμήσεις στη διαδικασία σχεδιασμού του έργου, και την καλή εκτίμηση των tickets σε story points που να ανταποκρίνονται σε κοντινά προσεγγιστικά αποτελέσματα από άποψη χρόνο ολοκλήρωσης ενός ticket. Συνεπώς οι τιμές που μπορεί να πάρει η στήλη Story points είναι 0, 1, 2, 3, 5, 8, 13, 20, 40, 100. Η σειρά με την οποία πρέπει να επιδιορθωθεί ένα defect ή να υλοποιηθεί ένα story αναφέρεται ως προτεραιότητα (priority). Όσο υψηλότερη είναι η προτεραιότητα, τόσο πιο γρήγορα θα πρέπει να επιλυθεί το πρόβλημα ή να υλοποιηθεί ένα story. Τα defects που καθιστούν το σύστημα λογισμικού μη λειτουργικό έχουν προτεραιότητα πάνω από τα defects που επηρεάζουν μόνο ένα μικρό μέρος της λειτουργίας του λογισμικού. Η προτεραιότητα (priority) αναφέρεται στη σειρά με την οποία ένας προγραμματιστής πρέπει για παράδειγμα να αντιμετωπίσει ένα defect, ενώ η σοβαρότητα (severity) αναφέρεται στον βαθμό επιρροής που έχει ένα ελάττωμα στη λειτουργία του προϊόντος Η προτεραιότητα (priority) χωρίζεται σε τρεις κατηγορίες: χαμηλή (low), μεσαία (medium) και υψηλή (high), ενώ η σοβαρότητα (severity) χωρίζεται στις εξής κατηγορίες: κρίσιμη (critical), μείζων (major), μέτρια (moderate), μικρής σημασίας (minor) και κοσμητικής σημασίας (cosmetic). Οι τιμές που μπορεί να πάρει η στήλη Priority είναι Low, Medium, High και οι τιμές που μπορεί να πάρει η στήλη Severity είναι Critical, Major, Moderate, Minor, Cosmetic. Συνεπώς η στήλη Ticket Id, συνδέεται άμεσα με την στήλη Story points και τις στήλες Priority και Severity. Για παράδειγμα στην παρακάτω εικόνα βλέπουμε ότι στο sprint 203 το Ticket type Story με Ticket Id CT-9385 έχοντας πολύ υψηλό Severity δηλαδή Critical, έχει ξεκινήσει στις αρχές του sprint, ενώ το Ticket type Story με Ticket Id CT-9387 έχοντας πολύ χαμηλό Priority δηλαδή Low, υλοποιείται προς το τέλος του sprint.

Sprint Number	Ticket Id	Functional Area	Ticket title	Ticket type Story: 1 Defect: 2	Start date	End date	Dates diff	Application version	Story points	Priority	Severity
202	BT-3952	Bet placement page	The "Customer Id" field should not be visible in section Bet placement platform	2	25/11/2021	26/11/2021	1	BE v.3.203	1	Medium	Medium
203	CT-9385	Customer page	Add "E-mail" field in section "Personal Details" in Customer page	1	02/12/2021	03/12/2021	1	CB v.1.044	5	Medium	Critical
203	CT-9386	Customer page	Add "Gender" field in section "Personal Details" in Customer page	1	06/12/2021	11/12/2021	5	CB v.1.045	1	Low	Moderate
203	CT-9387	Customer page	Add "Address" field in section "Personal Details" in Customer page	1	08/12/2021	17/12/2021	9	CB v.1.046	2	Low	Moderate

Πίνακας 9 Οι στήλες Story points, Priority και Severity του πίνακα Δεδομένων.

Με την ολοκλήρωση ενός ticket, δημιουργείται μία νέα έκδοση της συγκεκριμένης λειτουργικότητας της εφαρμογής που τροποποιήθηκε σύμφωνα με τα απαιτούμενα που αναλύονταν στο συγκεκριμένο ticket, οι διαφορετικές εκδόσεις φαίνονται στην στήλη Application version. Η στήλη Application version έχει τιμές με συγκεκριμένη μορφή που ακολουθούν την εκάστοτε πολιτική που έχει οριστεί, στην προκειμένη περίπτωση η τιμή είναι χωρισμένη σε δύο μέρη, το πρώτο μέρος αφορά την εκάστοτε λειτουργικότητα και έχει κάποια σταθερά αρχικά γράμματα και το δεύτερο μέρος είναι αριθμοί με αύξουσα σειρά κατά μια μονάδα αυξανόμενα, για παράδειγμα BE v.3.203 και BE v.3.204 (τιμές από sprint 202 και 203). Επίσης υπάρχει και η στήλη Dates diff που προκύπτει από την διαφορά μεταξύ των στηλών End date και Start date, και παρουσιάζει τον χρόνο που χρειάστηκε για να υλοποιηθεί το εκάστοτε ticket. Οι τιμές που μπορεί να πάρει η στήλη Dates diff είναι θετικοί ακέραιοι αριθμοί και δεν γίνεται σε καμία περίπτωση να είναι αρνητικοί αριθμοί, καθώς δεν είναι εφικτό το Start date να είναι μεταγενέστερο δηλαδή μετά από το End date. Για παράδειγμα για το ticket με Ticket Id BC-1085 και με ημερομηνία ολοκλήρωσής του στις 02/03/2022, δημιουργήθηκε μια νέα έκδοση της Bet cashout page, η έκδοση CA v.2.206, και χρειάστηκε 14 ημέρες για να ολοκληρωθεί.

Sprint Number	Ticket Id	Functional Area	Ticket title	Ticket type Story: 1 Defect: 2	Start date	End date	Dates diff	Application version	Story points
207	CT-9393	Customer page	Add the dropdown list with name "Profile" in section "Bet Account Details" in Customer page	1	23/02/2022	28/02/2022	5	CB v.1.052	2
207	BC-1084	Bet cashout page	Add field "Cashout IBAN" in section Bet cashout platform	1	15/02/2022	01/03/2022	14	CA v.2.205	5
207	BC-1085	Bet cashout page	The value in "Cashout IBAN" field is not displayed properly in section Bet cashout platform	2	16/02/2022	02/03/2022	14	CA v.2.206	3
208	CT-9394	Customer page	Add "Account Id" field in section "Bet Account Details" in Customer page	1	07/03/2022	10/03/2022	3	CB v.1.053	1
209	CT-9395	Customer page	Remove "Customer Id" field from section "Bet Account Details" in Customer page	1	11/03/2022	23/03/2022	12	CB v.1.054	3
209	CT-9396	Customer page	Add "Level" field in section "Bet Account Details" in Customer page	1	14/03/2022	31/03/2022	17	CB v.1.055	1

Πίνακας 10 Οι στήλες Dates diff και Application version του πίνακα Δεδομένων.

Κάθε ticket έχει συγκεκριμένο είδος δοκιμής, η στήλη Testing μας δείχνει με ποιον τρόπο έχει δοκιμαστεί το εκάστοτε ticket, οι κατηγορίες είναι οι εξής: Default manual testing: 1 δηλαδή η χειροκίνητη δοκιμή, Automation testing: 2 δηλαδή η αυτοματοποιημένη δοκιμή, Exploratory manual testing: 3 δηλαδή η διερευνητική δοκιμή και τέλος η κατηγορία Not tested: 4 όπου σε αυτήν ανήκουν οι περιπτώσεις που δεν πραγματοποιείται κάποια δοκιμή. Συνεπώς οι τιμές που μπορεί να πάρει η στήλη Testing είναι 1,2,3,4. Επίσης η στήλη Team capacity power αφορά το σύνολο των ατόμων που δούλεψαν στο συγκεκριμένο ticket, δηλαδή προγραμματιστές (developers), υπεύθυνοι δοκιμών (testers), επιχειρηματικοί αναλυτές (business analysts) και άλλα μέλη της ομάδας. Οι τιμές που μπορεί να πάρει η στήλη Team capacity power είναι 1,2,3,4,5,6,7,8,9 (9μελή ομάδα). Η στήλη Customer satisfied with delivery που αφορά το εάν ο πελάτης είναι ικανοποιημένος με την παράδοση του εκάστοτε ticket και μπορεί να είναι Yes: 1 και No: 2. Η συγκεκριμένη στήλη συμπληρώνεται μετά από την σχετική ανατροφοδότηση που λαμβάνεται από τον πελάτη. Οι τιμές που μπορεί να πάρει η στήλη Customer satisfied with delivery είναι 1,2. Επίσης υπάρχει η στήλη Human hours spent που μας δείχνει τις ώρες που ξοδεύτηκαν (εάν υπάρχουν) στο εκάστοτε ticket χωρίς να υπάρχουν ώρες που δαπανήθηκαν για δοκιμή. Οι τιμές που μπορεί να πάρει η στήλη Human hours spent είναι από 0 δηλαδή καθόλου έως 21. Τέλος υπάρχει και η στήλη Customer can use bet application process successfully, με τιμές Yes: 1 και No: 2, όπου μας δείχνει εάν ο πελάτης μπορεί να χρησιμοποιήσει με την νέα προσθήκη ή αλλαγή που απορρέει από το αντίστοιχο ticket την εκάστοτε λειτουργικότητα με επιτυχία, για παράδειγμα ο τελικός πελάτης που στοιχηματίζει επιτυγχάνει την τοποθέτηση του στοιχήματος, ή την εξαργύρωση του στοιχήματος. Και αυτή η στήλη συμπληρώνεται μετά από την σχετική ανατροφοδότηση που λαμβάνεται από τον πελάτη. Οι τιμές που μπορεί να πάρει η στήλη Customer can use bet application process successfully είναι 1,2. Στην παρακάτω εικόνα έχουν επισημανθεί με τέσσερα διαφορετικά χρώματα (πορτοκαλί, ρόζ, μωβ και κόκκινο) τέσσερα παραδείγματα με διαφορετικό είδος δοκιμής του εκάστοτε ticket. Με πορτοκαλί είναι η αυτοματοποιημένη δοκιμή (Automation testing), με ρόζ είναι η χειροκίνητη δοκιμή (Default manual testing), με μωβ είναι η διερευνητική δοκιμή (Exploratory manual testing) και με κόκκινο είναι η περίπτωση που δεν πραγματοποιείται καμία δοκιμή (Not tested).

Sprint Number	Ticket Id	Ticket type Story: 1 Defect: 2	Story points	Priority	Severity	Testing Default manual testing: 1 Automation testing: 2 Exploratory manual testing: 3 Not tested: 4	Team capacity power (incl. developers, testers, business analysts team members)	Customer satisfied with delivery Yes: 1 No: 2	Human hours spent (actually when testing activities do not exists)	Customer can use bet application process successfully Yes: 1 No: 2
201	CT-9381	1	1	High	Moderate	1	4	1	1	1
201	CT-9382	1	2	Medium	Moderate	4	6	2	15	2
201	CT-9383	1	3	Medium	Minor	2	5	1	1	1
201	CT-9384	2	1	Medium	Minor	4	1	2	12	2
202	BT-3950	1	1	Medium	Medium	1	4	1	0	1
202	BT-3951	1	2	Medium	Medium	1	8	1	2	1
202	BT-3952	2	1	Medium	Medium	4	2	2	11	1
203	CT-9385	1	5	Medium	Critical	3	7	1	0	1
203	CT-9386	1	1	Low	Moderate	4	4	2	14	2
203	CT-9387	1	2	Low	Moderate	1	8	1	0	1
204	BT-3953	1	1	Medium	Medium	1	6	1	0	1
204	BC-1078	1	8	Medium	Medium	2	6	1	1	1
204	BC-1079	1	5	High	Medium	4	2	2	13	1

Πίνακας 11 Οι στήλες Testing, Team capacity power, Customer satisfied with delivery, Human hours spent και Customer can use bet application process successfully του πίνακα Δεδομένων.

3.2 Διαγράμματα και Συμπεράσματα

Κατανοώντας, χρησιμοποιώντας και αξιοποιώντας τα στοιχεία του Πίνακα Δεδομένων που αναλύθηκαν στο προηγούμενο υποκεφάλαιο, κατά κύριο λόγο γίνεται ανάλυση αυτών και προκύπτουν τα συμπεράσματα που αφορούν τον τομέα της διασφάλισης ποιότητας της στοιχηματικής εφαρμογής. Καθώς διάφορα στατιστικά στοιχεία μπορούν να προκύψουν έχοντας ως σημείο αναφοράς τα άτομα της ομάδας που ασχολήθηκαν με ένα ticket, την ταχύτητα υλοποίησης του εκάστοτε ticket, την προτεραιότητα των tickets, την κρισιμότητα των tickets, την διαθεσιμότητα δυναμικού για να υλοποιήσει τα tickets, τη λειτουργική περιοχή που θα αναπτυχθεί και θα τροποποιηθεί σύμφωνα με την περιγραφή και τις απαιτήσεις που αναλύονται στο συγκεκριμένο ticket και άλλα. Στο πλαίσιο όμως της συγκεκριμένης διπλωματικής εργασίας θα εστιάσουμε στα στατιστικά στοιχεία που αφορούν την δοκιμή (testing) της στοιχηματικής εφαρμογής, θα επικεντρωθούμε στα είδη των tickets δηλαδή εάν είναι Story ή Defect, στο είδος της δοκιμή (testing) που υλοποιήθηκε κάθε ticket, δηλαδή αυτοματοποιημένη δοκιμή (Automation testing), χειροκίνητη δοκιμή (Default manual testing), διερευνητική δοκιμή (Exploratory manual testing) τις περιπτώσεις που δεν πραγματοποιείται καμία δοκιμή (Not tested), στην διάρκεια της υλοποίησης κάθε ticket, εάν είναι ευχαριστημένος ο πελάτης για κάθε ticket και εάν μπορεί να το αξιοποιήσει ανάλογα.

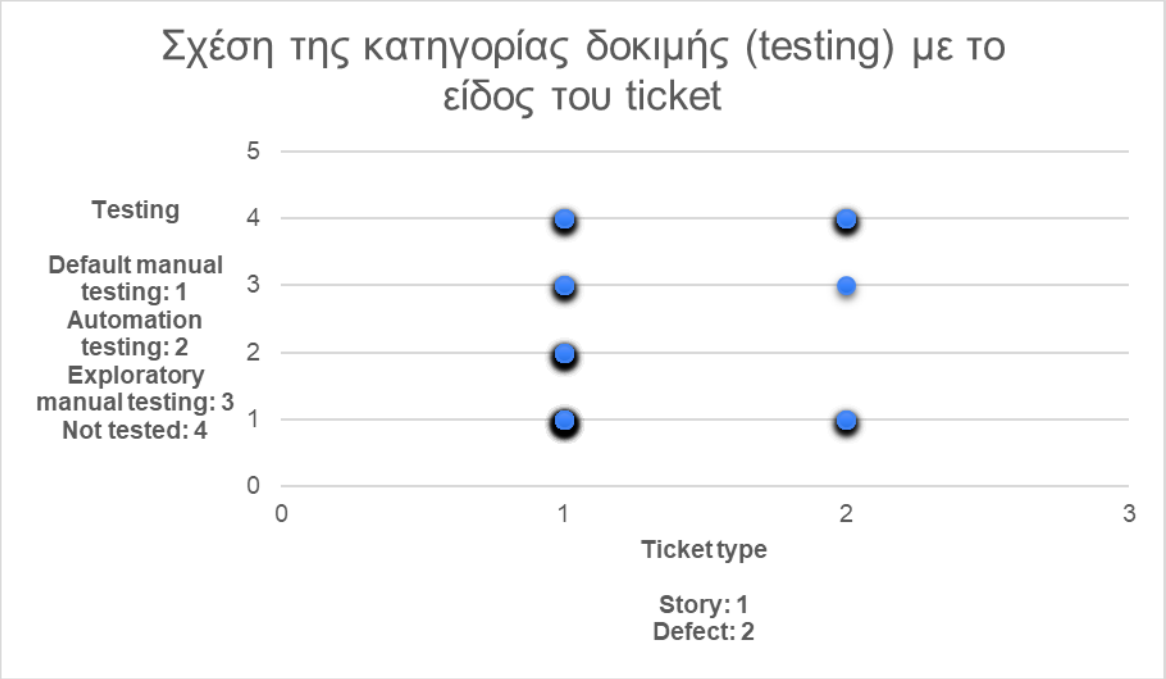
Στο σημείο αυτό θα γίνει μια σύντομη αναφορά στη θεωρία της στατιστικής, υπάρχουν και κάποια αριθμητικά μέτρα με τα οποία μπορούμε να περιγράψουμε μια κατανομή. Αυτά διακρίνονται σε μέτρα θέσης, τα οποία δείχνουν τη θέση του

κέντρου των παρατηρήσεων πάνω στον οριζόντιο άξονα, τα μέτρα διασποράς (ή μεταβλητότητας) που εκφράζουν τις αποκλίσεις των τιμών της μεταβλητής γύρω από τα μέτρα κεντρικής τάσης όπως είναι η μέση τιμή. Πρακτικά το πόσο οι παρατηρήσεις εκτείνονται από το κέντρο και τα μέτρα ασυμμετρίας που καθορίζουν τη μορφή της καμπύλης συχνοτήτων και το αν αυτή παρουσιάζει συμμετρία. Συγκεκριμένα τα μέτρα διασποράς είναι: το Εύρος (ή κύμανση) R, η Διακύμανση (ή διασπορά) s2, η Τυπική απόκλιση s και ο Συντελεστής Μεταβολής (ή Μεταβλητότητας. Ένας απλός ορισμός της Διακύμανσης s2 ακολουθεί, Διακύμανση ενός συνόλου παρατηρήσεων χ1, χ2,…χν, ονομάζουμε τη μέση τιμή των τετραγώνων των αποκλίσεων των χi από τη μέση τιμή x.

Πραγματοποιήθηκαν ένα σύνολο συνδυασμών των στηλών του Πίνακα δεδομένων και δημιουργήθηκαν τα αντίστοιχα διαγράμματα διακύμανσης - διασποράς των συνδυασμών αυτών. Τα διαγράμματα διακύμανσης – διασποράς είναι ιδιαίτερα χρήσιμα στην απεικόνιση ανεξαρτησίας ή συσχέτισης ανάμεσα σε δύο μεγέθη. Στη συνέχεια αξιοποιώντας κάποιους από αυτούς του συνδυασμούς που κρίθηκαν χρήσιμοι, τους αναλύσαμε περαιτέρω και προέκυψαν τα συμπεράσματα που αναλύονται στη συνέχεια:

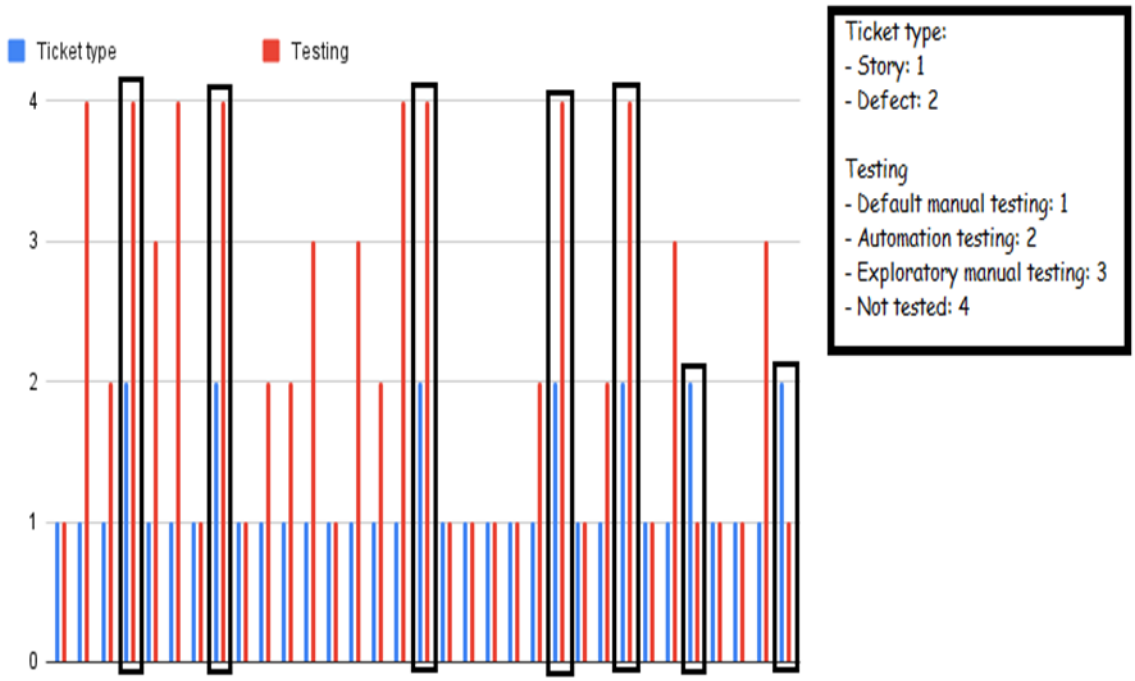
Συμπέρασμα 1: Στο παρακάτω διάγραμμα διακύμανσης – διασποράς παρουσιάζεται η σχέση της κατηγορίας δοκιμής (testing) με το είδος του ticket. Έχουμε τέσσερις κατηγορίες δοκιμής (testing) που εμφανίζονται στο άξονα y του διαγράμματος και είναι οι εξής: το νούμερο 1 αντιστοιχεί στο Default manual testing (χειροκίνητη δοκιμή), το νούμερο 2 αντιστοιχεί στο Automation testing (αυτοματοποιημένη δοκιμή), το νούμερο 3 αντιστοιχεί στο Exploratory manual testing (διερευνητική δοκιμή) και τέλος το νούμερο 4 αντιστοιχεί στο Not tested (καμία δοκιμή). Στον άξονα x, έχουμε το είδος του ticket, το νούμερο 1 αντιστοιχεί στο Story και το νούμερο 2 αντιστοιχεί στο Defect. Μέσα από το διάγραμμα που ακολουθεί, καθώς το διάγραμμα διασποράς – διακύμανσης απεικονίζει την συσχέτιση της ανεξάρτητης μεταβλητής (άξονας x) με την εξαρτημένη μεταβλητή (άξονας y), για παράδειγμα φαίνεται καθαρά στην περίπτωση άξονας x για τιμή =2 και για άξονας y τιμή =2 ότι δεν υπάρχει καμία συσχέτιση μεταξύ των δύο αυτών τιμών. Συνεπώς προκύπτει το συμπέρασμα ότι στο σύνολο των tickets που χρησιμοποιήθηκε αυτοματοποιημένη δοκιμή ήταν όλα Stories και δεν βρέθηκε κανένα defect από τα tickets. Αυτό δείχνει

το καλό επίπεδο του automation testing στην ομάδα, το καλό test coverage που καλύφθηκε, πέρα από τα happy paths (τις πιο απλές δοκιμές), καθώς και διάφορα edge case σενάρια. Έτσι, πρακτικά τα tickets όπου δοκιμάστηκαν με τη χρήση automation, δεν μας δημιουργούν έπειτα κάποιο defect.



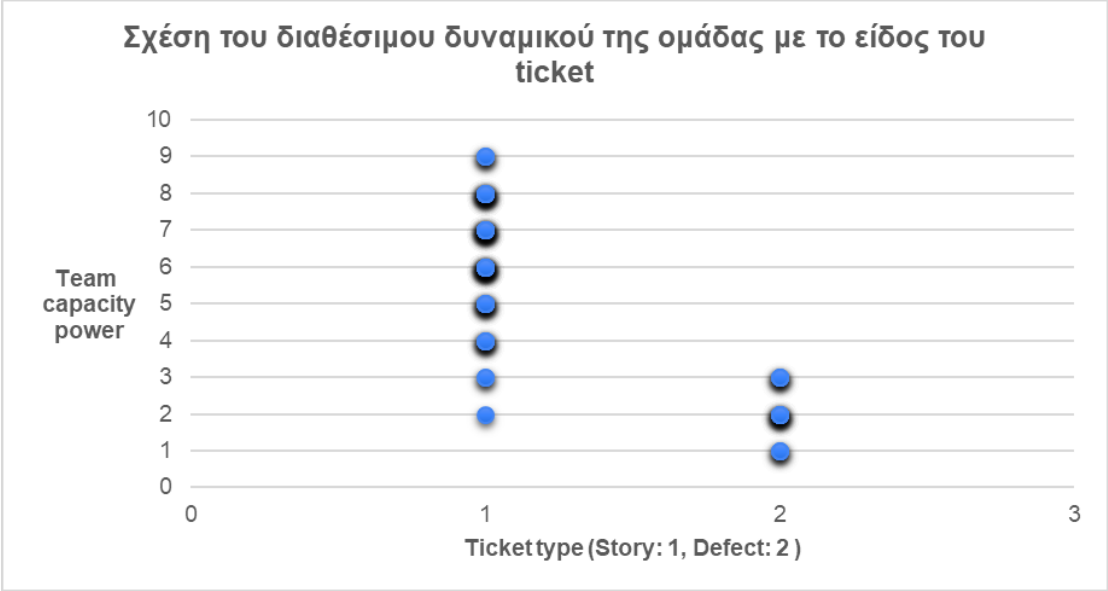
Διάγραμμα 1 Η σχέση της κατηγορίας δοκιμής (testing) με το είδος του ticket.

Στη συνέχεια παρουσιάζεται μια άλλη γραφική παράσταση σχετικά με τη σχέση της κατηγορίας δοκιμής (testing) με το είδος του ticket, και καταλήγουμε στο ίδιο συμπέρασμα δηλαδή ότι στο σύνολο των tickets που χρησιμοποιήθηκε αυτοματοποιημένη δοκιμή ήταν όλα Stories και δεν βρέθηκε κανένα defect από τα tickets.



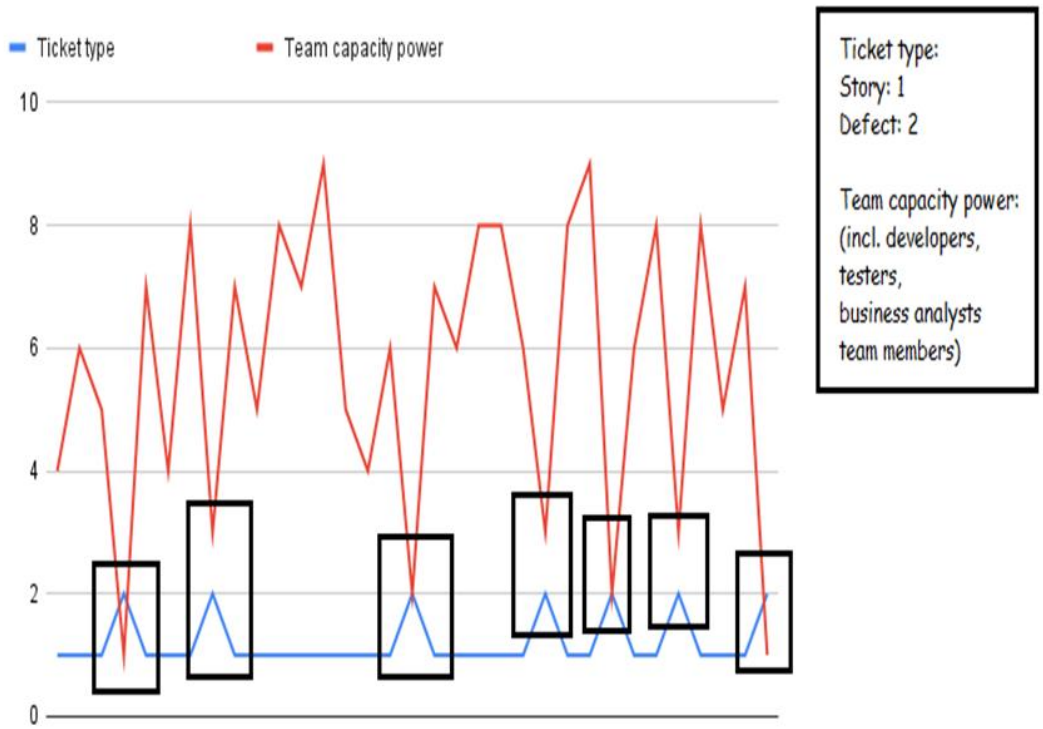
Διάγραμμα 2 Η σχέση της κατηγορίας δοκιμής (testing) με το είδος του ticket.

Συμπέρασμα 2: Στο παρακάτω διάγραμμα διακύμανσης – διασποράς παρουσιάζεται η σχέση του διαθέσιμου δυναμικού της ομάδας με το είδος του ticket. Το διαθέσιμο δυναμικό της ομάδας (Team capacity power) δηλαδή το σύνολο των ατόμων που δούλεψαν στο συγκεκριμένο ticket, δηλαδή προγραμματιστές (developers), υπεύθυνοι δοκιμών (testers), επιχειρηματικοί αναλυτές (business analysts) και άλλα μέλη της ομάδας, παρουσιάζονται στον άξονα y και οι τιμές που μπορεί να έχουν ειναι από 1 έως 9 (9μελή ομάδα). Στον άξονα x, έχουμε το είδος του ticket, το νούμερο 1 αντιστοιχεί στο Story και το νούμερο 2 αντιστοιχεί στο Defect. Μέσα από το παρακάτω διάγραμμα διασποράς – διακύμανσης που απεικονίζεται η συσχέτιση της ανεξάρτητης μεταβλητής (άξονας x) με την εξαρτημένη μεταβλητή (άξονας y), φαίνεται καθαρά ότι τα defects αντιστοιχούν στις περιόδους με μικρή δυναμική ομάδας (1-3 άτομα). Αυτό μπορεί να συμβαίνει για παράδειγμα λόγω αργιών, διακοπών, ειδικών αδειών, κλπ. ενώ τα Stories υλοποιούνται όταν υπάρχουν αρκετά άτομα της ομάδας δηλαδή 4-9 άτομα της ομάδας. Συνεπώς αυτό μας δείχνει πώς όταν η ομάδα είναι μικρή, υπάρχει μειωμένη εσωτερική επικοινωνία, μετάδοση γνώσης, και γενικότερα μειωμένη ποιότητα παράδοσης, που οδηγεί και σε περισσότερα defects.



Διάγραμμα 3 Η σχέση του διαθέσιμου δυναμικού της ομάδας με το είδος του ticket.

Στη συνέχεια παρουσιάζεται μια άλλη γραφική παράσταση σχετικά με τη σχέση του διαθέσιμου δυναμικού της ομάδας με το είδος του ticket με τα ίδια δεδομένα που χρησιμοποιήθηκαν και αναλύθηκαν παραπάνω και παρατηρούμε ότι καταλήγουμε στο ίδιο συμπέρασμα, τα defects αντιστοιχούν στις περιόδους με μικρή δυναμική ομάδας (1-3 άτομα).

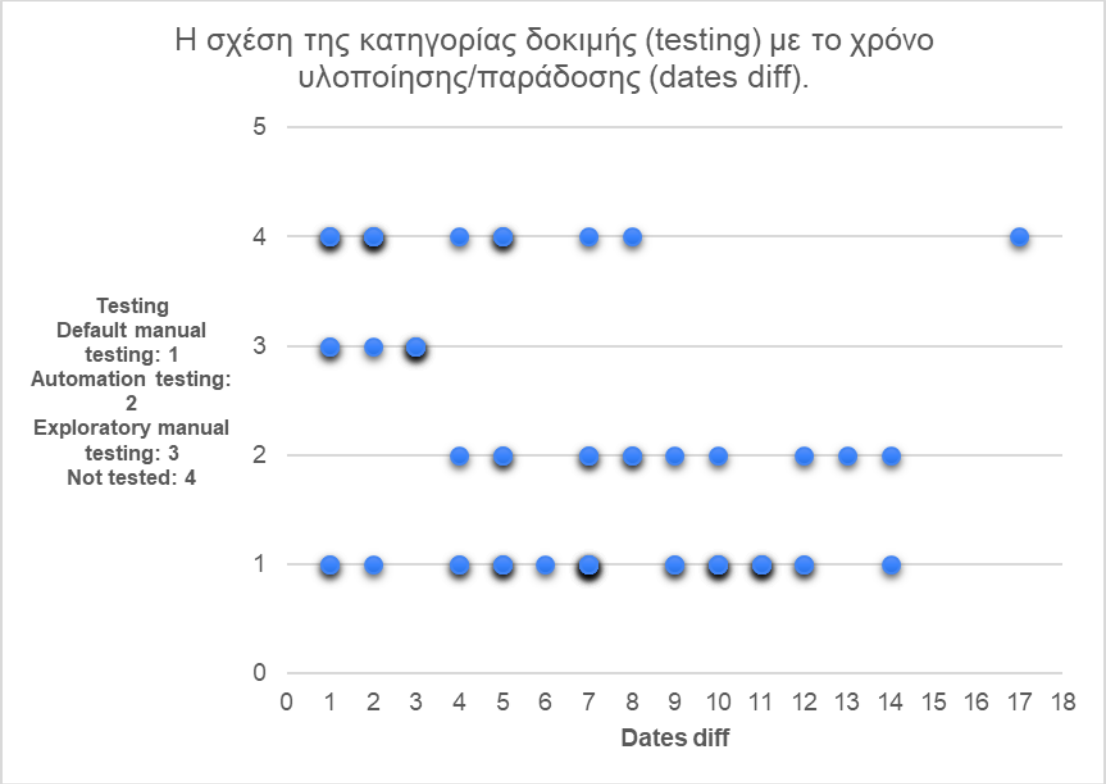


Διάγραμμα 4 Η σχέση του διαθέσιμου δυναμικού της ομάδας με το είδος του ticket.

Συμπέρασμα 3: Στο παρακάτω διάγραμμα διακύμανσης – διασποράς παρουσιάζεται h σχέση της κατηγορίας δοκιμής (testing) με τα story points. Όπως αναφέραμε και προηγουμένως, έχουμε τέσσερις κατηγορίες δοκιμής (testing) που εμφανίζονται στο άξονα y του διαγράμματος και είναι οι εξής: το νούμερο 1 αντιστοιχεί στο Default manual testing (χειροκίνητη δοκιμή), το νούμερο 2 αντιστοιχεί στο Automation testing (αυτοματοποιημένη δοκιμή), το νούμερο 3 αντιστοιχεί στο Exploratory manual testing (διερευνητική δοκιμή) και τέλος το νούμερο 4 αντιστοιχεί στο Not tested (καμία δοκιμή). Στον άξονα x, έχουμε τα Story points, οι τιμές προκύπτουν από την τροποποιημένη σειρά Fibonacci (0, 1, 2, 3, 5, 8, 13, 20, 40, 100) που χρησιμοποιείται για την εκτίμηση του σχετικού μεγέθους των tickets σε story points, και στην προκειμένη παίρνει τις τιμές 1,2,3,5,και 8. Μέσα από το διάγραμμα που ακολουθεί, καθώς το διάγραμμα διασποράς – διακύμανσης απεικονίζει την συσχέτιση της ανεξάρτητης μεταβλητής (άξονας x) με την εξαρτημένη μεταβλητή (άξονας y), παρατηρείται ότι στα tickets με μικρά Story points (1-2) δεν έχουμε test automation, ενώ στα μεγάλα Story points (δηλαδή 8), αυτό αναδεικνύει την αξία του automation σε σημαντικές λειτουργικότητες και παράλληλα πως αυτό δεν απαιτείται (το automation) για μικρές αλλαγές, μιας και η συντήρηση (maintenance) του test automation κοστίζει.

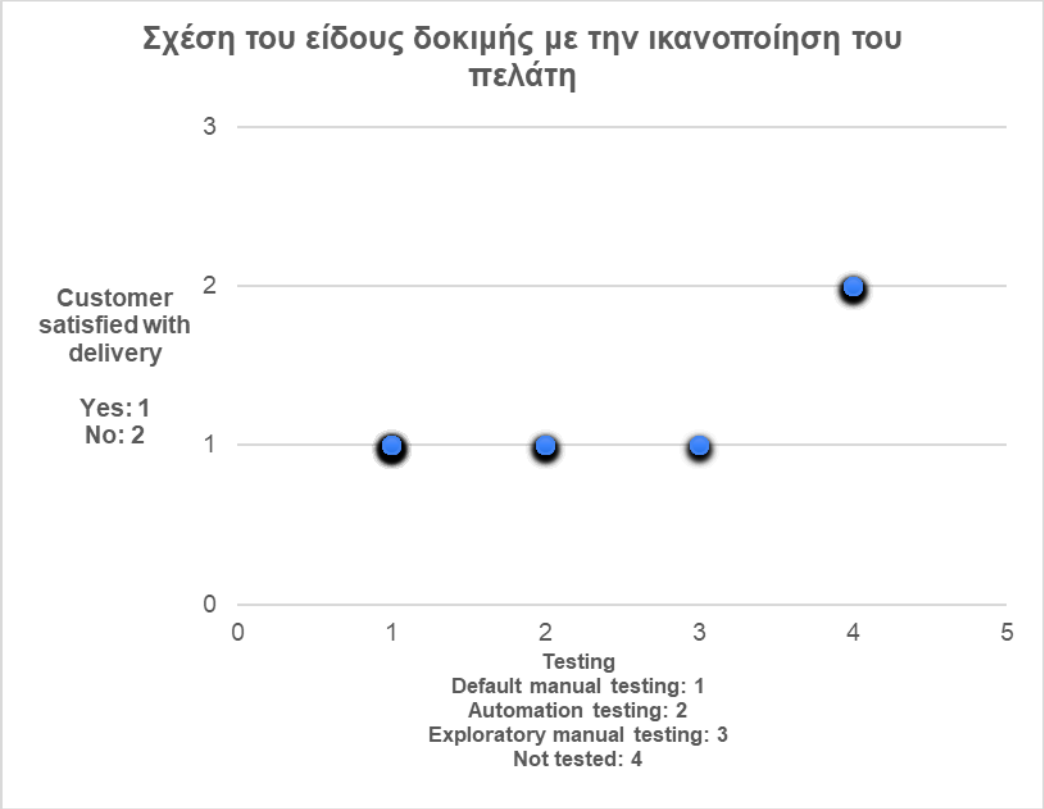
Διάγραμμα 5 Η σχέση της κατηγορίας δοκιμής (testing) με τα story points.

Συμπέρασμα 4: Στο παρακάτω διάγραμμα διακύμανσης – διασποράς παρουσιάζεται η σχέση της κατηγορίας δοκιμής (testing) με το χρόνο υλοποίησης/παράδοσης (dates diff). Όπως αναφέραμε και προηγουμένως, έχουμε τέσσερις κατηγορίες δοκιμής (testing) που εμφανίζονται στο άξονα y του διαγράμματος και είναι οι εξής: το νούμερο 1 αντιστοιχεί στο Default manual testing (χειροκίνητη δοκιμή), το νούμερο 2 αντιστοιχεί στο Automation testing (αυτοματοποιημένη δοκιμή), το νούμερο 3 αντιστοιχεί στο Exploratory manual testing (διερευνητική δοκιμή) και τέλος το νούμερο 4 αντιστοιχεί στο Not tested (καμία δοκιμή). Στον άξονα x, έχουμε το χρόνο υλοποίησης/παράδοσης (dates diff) που είναι σε ημέρες με τιμές από 1 έως 18. Μέσα από το διάγραμμα που ακολουθεί, καθώς το διάγραμμα διασποράς – διακύμανσης απεικονίζει την συσχέτιση της ανεξάρτητης μεταβλητής (άξονας x) με την εξαρτημένη μεταβλητή (άξονας y), παρατηρείται ότι στα tickets πραγματοποιείται exploratory testing μόνο στις περιπτώσεις που έχουμε μικρό dates diff δηλαδή με τιμές 1 έως 3, επίσης παράλληλα από το πίνακα δεδομένων αυτό συμβαίνεις στις περιπτώσεις που υπάρχει δεν υπάρχει Low priority (δηλαδή υπάρχει Medium και High priority) και παράλληλα δεν υπάρχει Low severity. Αυτό πρακτικά συμβαίνει στις περιπτώσεις περιορισμένου διαθέσιμου χρόνου υλοποίησης των tickets που δεν είναι εφικτό να αυτοματοποιήσουμε τα σενάρια δοκιμής.



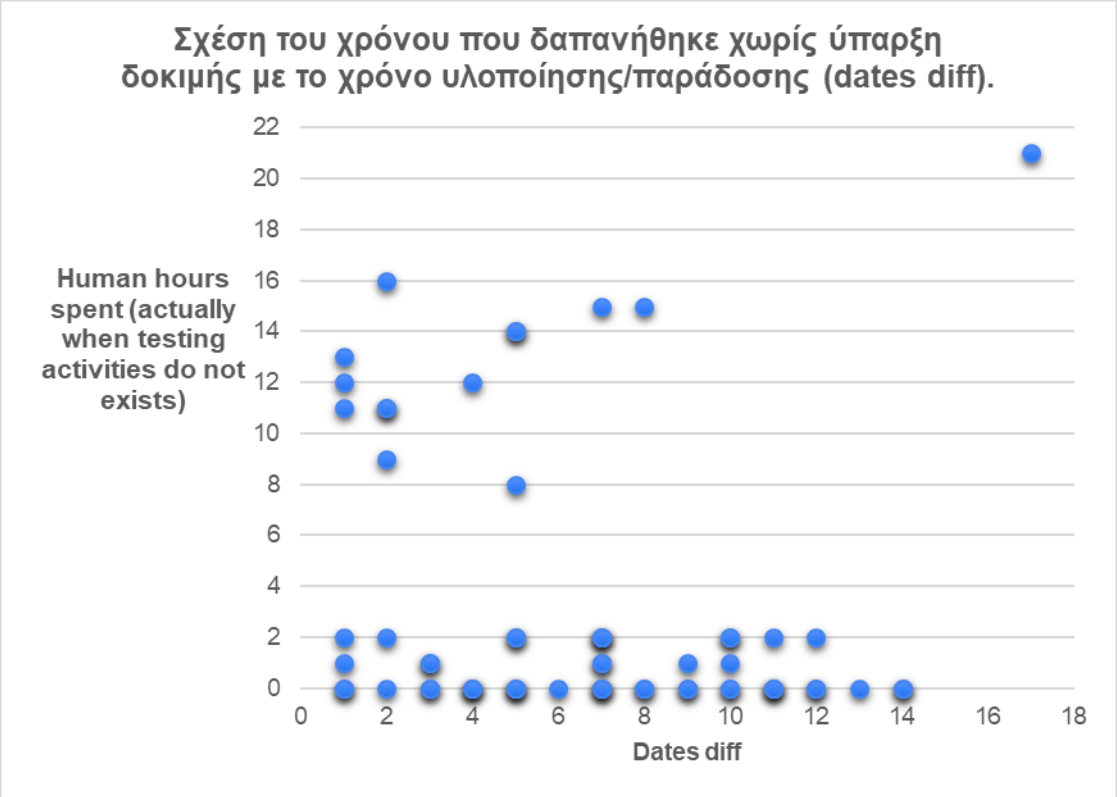
Διάγραμμα 6 Η σχέση της κατηγορίας δοκιμής (testing) με το χρόνο υλοποίησης/παράδοσης (dates diff).

Συμπέρασμα 5: Στο παρακάτω διάγραμμα διακύμανσης – διασποράς παρουσιάζεται η σχέση της κατηγορίας δοκιμής (testing) με το χρόνο υλοποίησης/παράδοσης (dates diff). Ο άξονας y, αφορά το εάν ο πελάτης είναι ικανοποιημένος με την παράδοση του εκάστοτε ticket (Customer satisfied with delivery) και μπορεί να είναι Yes: 1 και No: 2. Και έχουμε τις τέσσερις κατηγορίες δοκιμής (testing) που εμφανίζονται στο άξονα x του διαγράμματος και είναι οι εξής: το νούμερο 1 αντιστοιχεί στο Default manual testing (χειροκίνητη δοκιμή), το νούμερο 2 αντιστοιχεί στο Automation testing (αυτοματοποιημένη δοκιμή), το νούμερο 3 αντιστοιχεί στο Exploratory manual testing (διερευνητική δοκιμή) και τέλος το νούμερο 4 αντιστοιχεί στο Not tested (καμία δοκιμή). Μέσα από το διάγραμμα που ακολουθεί, καθώς το διάγραμμα διασποράς – διακύμανσης απεικονίζει την συσχέτιση της ανεξάρτητης μεταβλητής (άξονας x) με την εξαρτημένη μεταβλητή (άξονας y), παρατηρείται ότι όταν τα tickets δεν έχουμε δοκιμαστεί (δεν έχει πραγματοποιηθεί testing), υπάρχει μια δυσαρέσκεια από την πλευρά του πελάτη.



Διάγραμμα 7 Η σχέση του είδους δοκιμής με την ικανοποίηση του πελάτη.

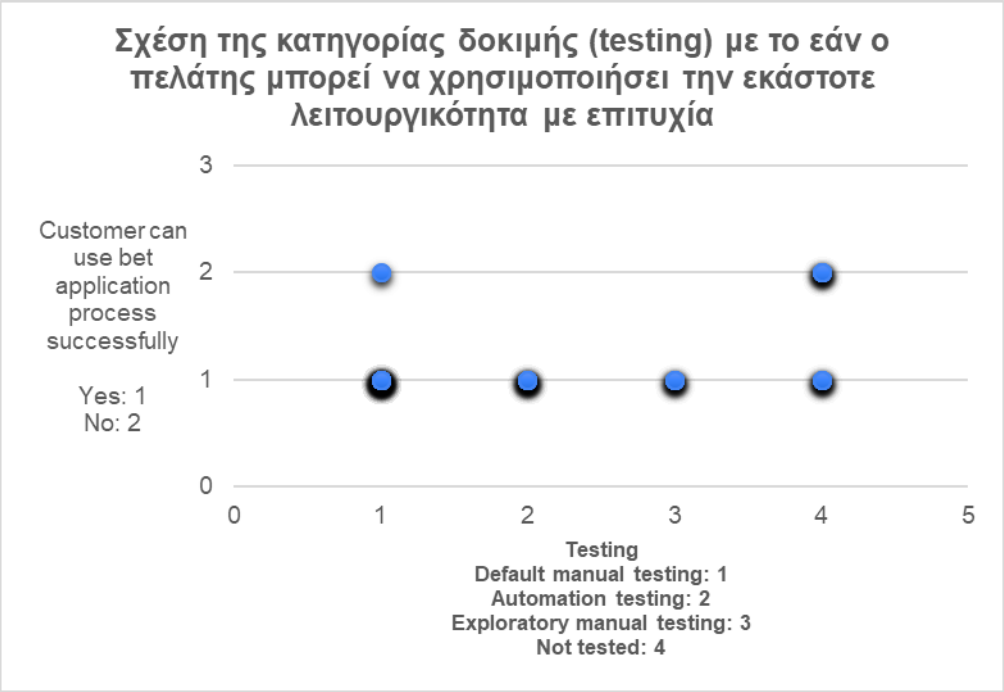
Συμπέρασμα 6: Στο παρακάτω διάγραμμα διακύμανσης – διασποράς παρουσιάζεται η σχέση του χρόνου που δαπανήθηκε χωρίς ύπαρξη δοκιμής με το χρόνο υλοποίησης/παράδοσης (dates diff). Ο άξονας y, μας δείχνει τις ώρες που ξοδεύτηκαν (Human hours spent) στο εκάστοτε ticket χωρίς την ύπαρξη δοκιμής. Στον άξονα x, έχουμε το χρόνο υλοποίησης/παράδοσης (dates diff) που είναι σε ημέρες με τιμές από 1 έως 18. Μέσα από το διάγραμμα που ακολουθεί, καθώς το διάγραμμα διασποράς – διακύμανσης απεικονίζει την συσχέτιση της ανεξάρτητης μεταβλητής (άξονας x) με την εξαρτημένη μεταβλητή (άξονας y), παρατηρείται ότι όταν στα tickets δεν πραγματοποιείται testing, χρειάζεται παραπάνω χρόνος στο να διορθωθούν τα defects. Ο χρόνος που απαιτείται από την ομάδα (devolopers και testers) για να παραδώσουν το εκάστοτε ticket ορθώς σύμφωνα με τις καθορισμένες απαιτήσεις του πελάτη είναι μικρότερος όταν πραγματοποιείται testing.



Διάγραμμα 8 Η σχέση του χρόνου που δαπανήθηκε χωρίς ύπαρξη δοκιμής με το χρόνο υλοποίησης/παράδοσης (dates diff).

Συμπέρασμα 7: Στο παρακάτω διάγραμμα διακύμανσης – διασποράς παρουσιάζεται η σχέση της κατηγορίας δοκιμής (testing) με το εάν ο πελάτης μπορεί να χρησιμοποιήσει με την νέα προσθήκη ή αλλαγή που απορρέει από το αντίστοιχο ticket την εκάστοτε λειτουργικότητα με επιτυχία. Συνεπώς ο άξονας y αναφέρεται στο εάν ο πελάτης μπορεί να χρησιμοποιήσει την εκάστοτε λειτουργικότητα με επιτυχία και έχει τιμή ίση με 1 και στις ανεπιτυχείς περιπτώσεις έχει τιμή ίση με 2. Και στον άξονα y του διαγράμματος έχουμε τα εξής: το νούμερο 1 αντιστοιχεί στο Default manual testing (χειροκίνητη δοκιμή), το νούμερο 2 αντιστοιχεί στο Automation testing (αυτοματοποιημένη δοκιμή), το νούμερο 3 αντιστοιχεί στο Exploratory manual testing (διερευνητική δοκιμή) και τέλος το νούμερο 4 αντιστοιχεί στο Not tested (καμία δοκιμή). Μέσα από το διάγραμμα που ακολουθεί, καθώς το διάγραμμα διασποράς – διακύμανσης απεικονίζει την συσχέτιση της ανεξάρτητης μεταβλητής (άξονας x) με την εξαρτημένη μεταβλητή (άξονας y), παρατηρείται ότι όταν στις περιπτώσεις που έχουμε δοκιμή αυτοματοποιημένη (Automation testing) ή διερευνητική (Exploratory manual testing), ο πελάτης μπορεί να χρησιμοποιήσει με

την νέα προσθήκη ή αλλαγή που απορρέει απο το αντίστοιχο ticket την εκάστοτε λειτουργικότητα με επιτυχία.



Διάγραμμα 9 Η σχέση της κατηγορίας δοκιμής (testing) με το εάν ο πελάτης μπορεί να χρησιμοποιήσει την εκάστοτε λειτουργικότητα με επιτυχία.

ΠΗΓΕΣ

[1] D. Graham, E. Veenendaal, I. Evans, R. Black, 2018, Foundations of Software Testing, International Software Testing Qualifications Board (ISTQB) Certification.

[2] E. Veenendaal & «Glossary Working Party», 2018, Standard glossary of terms used in Software Testing, International Software Testing Qualifications Board (ISTQB).

[3] Mishra, D., Hacaloglu, T., & Mishra, A., 2014, Teaching Software Verification and Validation Course: A Case Study. International Journal of Engineering Education, 6 (vol. 30), pp. 1473–1483.

[4] Fernando B. Abreu and Walcelio Melo, 1996, Evaluating the impact of object-oriented design on software quality. In Proceedings of the 3rd International Software Metrics Symposium. 90–99. https://doi.org/10.1109/METRIC.1996.492446.

[5] Kunte, P., & Mane, D., 2017, Automation Testing of Web Based Application with Selenium and HP UFT (QTP). International Research Journal of Engineering and Technology (IRJET), 6 (vol.4), pp. 2577–2583.

[6] Harald Altinger, Sebastian Siegl, Yanja Dajsuren, and Franz Wotawa, 2015, A Novel Industry Grade Dataset for Fault Prediction Based on Model-driven Developed Automotive Embedded Software. In Proceedings of the 12th Working Conference on Mining Software Repositories. IEEE Press, Piscataway, NJ, USA, 493–496.

[7] Singh, I., & Tarika, B., 2014, Comparative Analysis of Open Source Automated Software Testing Tools: Selenium, Sikuli and Watir. International Journal of Information & Computation Technology, 4 (vol. 15), pp. 1500–1513.

[8] Victor Basili and D. Rombach, 1988, The TAME project: towards improvement-oriented software environments. IEEE Transactions on Software Engineering 14, 6 (June 1988), 760–773.

[9] Shaukat, K., Shaukat, U., Feroz, F., Kayani, S., & Akbar, A., 2015, Taxonomy of Automated Software Testing Tools. International Journal of Computer Science and Innovation, 1, pp. 3–13.

[10] Islam, N. A, 2016, Comparative Study of Automated Software Testing Tools. Culminating Projects in Computer Science and Information Technology.

[11] Freeman, R. B. , K. Shaw, 2009, "International differences in the business practices and productivity of firms", Chicago ; London, University of Chicago Press.
[12] A. Dennis, B. H. Wixom, and D. Tegarden, 2009, Systems Analysis and Design with OOP Approach with UML 2.0, 4th Editio. USA: John Wiley & Sons, Inc.
[13] L. Luo, 'A Report on Software Testing Techniques', Pittsburgh, USA.
[14] D. R. Graham, 1979, 'TESTING, VERIFICATION AND VALIDATION', Int. J., vol. XVI, pp. 1069–1101.
[15] G. J. Myers, C. Sandler, and T. Badgett, 2012, The Art of Software Testing 3rd Edition, Third Edit. Canada.: John Wiley & Sons, Inc.
[16] Falls, M. , Books24x7 Inc., 2004, "Inside the minds the software business : how top companies design, develop & sell successful products & applications", Inside the minds. Boston, Mass., Aspatore.
[17] Ebert, C., SpringerLink (Online service), 2005, "Best practices in software measurement how to use metrics to improve project and process performance", Berlin, Springer-Verlag,: xi, p.295.
[18] C. Padmini, 2013, '1- Beginners Guide To Software Testing', pp. 1–41.
[19] E. Miller, Software testing & validation techniques. [Washington D.C.]: IEEE Computer Society Press, 1981.
[20] A Voulodimos, N Doulamis, A Doulamis, Eftychios Protopapadakis, 2018, Deep learning for computer vision: A brief review, Computational intelligence and neuroscience, Hindawi, Online: 01 February 2018.
[21] K Makantasis, K Karantzalos, A Doulamis, N Doulamis, 2015, Deep supervised learning for hyperspectral data classification through convolutional neural networks, 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), IEEE, 4959-4962.
[22] Nikolaos D Doulamis, Anastasios D Doulamis, Yannis S Avrithis, Klimis S Ntalianis, Stefanos D Kollias, 2000, Efficient summarization of stereoscopic video sequences, IEEE Transactions on Circuits and Systems for Video Technology, IEEE, 10.4, 501-517.
[23] Vassilios Vescoukis, Nikolaos Doulamis, Sofia Karagiorgou, 2012, A service oriented architecture for decision support systems in environmental crisis management, Future generation computer systems, North-Holland, 28.3, 593-604.
[24] Eftychios Protopapadakis, Athanasios Voulodimos, Anastasios Doulamis, Nikolaos Doulamis, Tania Stathaki, 2019, Automatic crack detection for tunnel

inspection using deep learning and heuristic image post-processing, Springer US, 49.7, 2793-2806.

[25] N Mastorakis, 1999, Computational Intelligence and Applications, World Scientific Engineering Society.

[26] George Kopsiaftis, Ioannis Georgoulas, Ioannis Rallis, Ioannis Markoulidakis, Kostis Tzanettis, Michael Sfakianos, Nikolaos Doulamis, 2021, Application Programming Interface for a Customer Experience Analysis Tool.

[27] P Kokkinos, K Christodoulopoulos, N Doulamis, E Varvarigos, 2008, Grid Computing Research Progress, 91-150.

[28] R. V. Binder, 1999, Testing Object-Oriented Systems: Objects, Patterns, and Tools. Addison-Wesley Professional.

[29] I. Jovanovic, 2009, '21 - Software Testing Methods and Techniques', IPSI BgD Trans. Internet Res., vol. 5, no. 1, pp. 30–40.

[30] S. Nidhra and J. Dondeti, 2012, '14 - Black Box and White Box Testing Techniques', Int. J. Embed. Syst. Appl., vol. 2, no. 2, pp. 28–49.

[31] Mohd. Ehmer Khan and Farmeena Khan, 2012,'19 - A Comparative Study of White Box, Black Box and Grey Box Testing Techniques', Int. J. Adv. Comput. Sci. Appl., vol. 3, no. 6, pp. 12–15.

[32] J. Badlaney, R. Ghatol, and R. Jadhwani, 2006 '10 - An Introduction to Data-Flow Testing', Control, pp. 1–8.

[33] L. Rajamanickam, N. Azlia, B. Mat, S. Norbaya, and B. Daud, 2019,Software Testing : The Generation Tools,‖ Int. J. Adv. Trends Comput. Sci. Eng., vol. 8, no. 2, pp. 231–234.

[34] S.-T. Lai, 2017, Test Case Quality Management Procedure for Enhancing the Efficiency of IID Continuous Testing,‖ J. Softw., vol. 12, no. 10, pp. 794–806, , doi: 10.17706/jsw.12.10.794-806.

[35] N. Chauhan, 2010, Software Testing: Principles and Practices. Oxford university press.

[36] Appelrouth, J.I., & Zabrucky, K.M., 2015, Preparing for the SAT: A review. College and University, 92, p. 2-17.

[37] Briggs, D. C., 2001, The effect of admissions test preparation: evidence from NELS:88. Chance 14(1): 10-18.

[38] P. Ron, 2001, Software testing. Vol. 2. Indianapolis: Sam‟s.

[39] Trivedi, S. H., 2012, Software Testing Techniques, International Journal of Advanced Research in Computer Science and Software Engineering ,vol. 2 no.10, pp.433–439.

[40] Bharati, C., & Verma, 2013, Analysis of Different Regression Testing Approaches. International Journal of Advanced Research in Computer and Communication Engineering,vol. 2 no.5, 2150–2155.

[41] Chen, S., Chen, Z., Zhao, Z., Xu, B., & Feng, Y, 2011, Using semi-supervised clustering to improve regression test selection techniques. Fourth IEEE International Conference on Software Testing, Verification and Validation , pp. 1–10.

[42] Ur, S., Khan, R., Lee, S., Parizi, R. M., & Elahi, M., An Analysis of the Code Coverage-based Greedy Algorithms for Test Suite Reduction, The Second International Conference on Informatics Engineering & Information Science (ICIEIS2013), pp.370–377,2-13.

[43] Kichigin, D., 2010, Test Suite Reduction for Regression Testing of Simple Interactions between Two Software Modules, Perspectives of Systems Informatics ,pp. 107–123.

[44] Parsa, S., & Khalilian, A., 2010, On the Optimization Approach towards Test Suite Minimization. International Journal of Software Engineering and Its Applications,vol.4 ,no. 1, pp.15–28.

[45] Mahajan, P., Shedge, H., & Patkar, U., 2016, Automation Testing In Software Organization. International Journal of Computer Applications Technology and Research, 4 (vol. 5), pp.198–201.

[46] Dingsøyr, T., et al, 2010, "Agile software development : current research and future directions", Berlin, Springer.

[47] Fernandes, J. D., & Fonzo, A. D., 2013, When to Automate Your Testing (and When Not To), March 2013.

[48] Yoo, S., & Harman, M, 2012, Regression Testing Minimisation, Selection and Prioritisation - A Survey, 2009,Software Testing Verification and Reliability, 22(2) , pp. 67 – 120.

[49] Kim Herzig. 2014. Using Pre-Release Test Failures to Build Early Post-Release Defect Prediction Models. In Proceedings of the 2014 IEEE 25th International Symposium on Software Reliability Engineering (ISSRE ’14). IEEE Computer Society,Washington, DC, USA, 300–311.

[50] Ramler, R., & Wolfmaier, K. Economic perspectives in test automation. Proceedings of the 2006 International Workshop on Automation of Software Test - AST 06, 2006, pp.85-91.

[51] Bharti, T., & Dutt, E. V., 2014, Relative Review of Automated Testing Tools: (QTP) Quick Test Professional, Selenium and Test Complete. International Journal of Computer Science Trends and Technology (IJCST), 6 (vol. 2), pp.110–114.

[52] W. E. Lewis, 2009, Software Testing and Continuous Quality Improvement Third Edithion. Boca Raton London New York: Taylor & Francis Group, LLC.