Τμήμα Μηχανικών Πληροφορικής
Τεχνικό Εκπαιδευτικό Ίδρυμα Κρήτης

Διπλωματική Εργασία

Ανάπτυξη εφαρμογής δοκιμών αυτοματισμού, που προσαρμόζεται σε ηλεκτρονικό κατάστημα, με απώτερο σκοπό τη βελτιστοποίηση της ποιότητας του λογισμικού.

Ιωάννης Τζανάκης (ΑΜ 2452)

Επιβλέποντες Καθηγητές:
Τριανταφυλλίδης Γεώργιος- Νικόλαος Βιδάκης

Επιτροπή αξιολόγησης:
Τριανταφυλλίδης Γεώργιος- Νικόλαος Βιδάκης- Μανιφάβας Χαράλαμπος

Κοπεγχάγη, 11 Απριλίου 2014

Ευχαριστίες

Ο τίτλος της παρούσας διπλωματικής είναι: «Ανάπτυξη εφαρμογής δοκιμών αυτοματισμού, που προσαρμόζεται σε ηλεκτρονικό κατάστημα, με σκοπό τη βελτιστοποίηση της ποιότητας του λογισμικού». Αυτή πραγματοποιήθηκε, στο πλαίσιο της πτυχιακής εργασίας του τμήματος Μηχανικών πληροφορικής του Τεχνολογικού Εκπαιδευτικού Ιδρύματος Κρήτης.

Στο σημείο αυτό, αισθάνομαι την ανάγκη να εκφράσω τις ειλικρινείς και θερμές ευχαριστίες μου σε όσους συνέβαλαν στην ολοκλήρωση αυτής της προσπάθειας :

Και πρώτα απ' όλα, στους επιβλέποντες καθηγητές μου, Τριανταφυλλίδη Γεώργιο και Νικόλαο Βιδάκη. Αυτοί έπαιξαν καθοριστικότατο ρόλο στη πτυχιακή αυτή, καθώς έδωσαν ώθηση στο κίνητρό μου, για την ολοκλήρωση των σπουδών μου. Αυτό βέβαια δεν πρέπει να θεωρηθεί δεδομένο, καθώς τα τελευταία χρόνια είμαι κάτοικος εξωτερικού και απέχω από τα κοινά της σχολής.

Παράλληλα, στον ίδιο κύκλο εντάσσεται ο φίλος και κουμπάρος μου Κατσικά Δημήτρη. Τον ευχαριστώ για την αμέριστη υποστήριξη και βοήθεια ιδιαίτερα σε στιγμές μεγάλων διλλημάτων. Παράλληλα να τονίσω την μεταφορά θετικής ενέργειάς που ήταν πάντοτε κρίσιμη σε στάδια αυξημένης πίεσης.

Φυσικά, από τις ευχαριστίες αυτές, δεν θα μπορούσαν να λείπουν οι γονείς μου. Η μητέρα μου Παναγιώτα και ο πατέρας μου Παύλος, οι οποίοι με τεράστιους κόπους και προσωπικές στερήσεις μου παρείχαν τα απαραίτητα εφόδια ώστε να κάνω τα πρώτα μου βήματα σε επαγγελματικό αλλά κυρίως προσωπικό επίπεδο. Σε αυτούς αφιερώνω και την εργασία αυτή.

Θέλω επίσης, να ευχαριστήσω την υπόλοιπη οικογένεια μου, τους φίλους μου και τους συναδέλφους μου. Σε αυτούς, που με την καθημερινή τους υποστήριξη, συμπαράσταση και υπομονή, συνέβαλαν στην τόνωση της αυτοπεποίθησης και ηθικού, ούτος ώστε να πετύχω το στόχο μου. Η πτυχιακή αυτή αποτέλεσε σημαντικό αλλά και καίριο μάθημα για το πως να προσπερνώ τις δυσκολίες που μου παρουσιάζονται.

Τέλος, θα ήθελα να ευχαριστήσω θερμά τους συναδέλφους μου για την αμέριστη συμπαράστασή και βοήθεια που απλόχερα μου προσέφεραν.
I would like to thank through the bottom of my heart Juan Manuel Lago Dominguez, Suchitra Mogili, Shamika Vijay Deshmukh and Vijay Kumar Eswaravaka, colleagues of mine in Bestseller. The motivation, solidarity and assistance that I received were far greater than I could have ever imagined; in accomplishing this extremely challenging and demanding task.

Abstract

A significant aspect of the E-commerce software development is Quality that includes testing various applications, verification to ensure stable applications with minimal defects. It is not easy for big E-commerce sites to maintain defect free systems which involves various functionalities, markets, integration to external systems, combined with functionality updates as part of Agile model.

Consequently, the main objective of this thesis was to develop a tool that would help reduce manual effort to create redundant tasks and to perform better functional tests on Bestseller web shop, available across 14 different countries, 12 brands, supports various payment methods. This tool performs, end to end test, i.e. searching specific products, adding to basket, delivery, shipping and checkout.

Ultimately, the automation tool maintains and optimizes the quality of the software, by verifying these essential functionalities. Meanwhile reducing test effort, disposes to some other crucial functions and systems, and hence increasing return on investment.

The Automation Tool was developed in Visual Studio using C# with Selenium.
During the development, it was found that in order for the automation tool to be effective and efficient, some essential requirements have to be followed such as scalability, user Friendliness and the dynamism.

Alongside the development of the application, the aim of the final project was theoretically, to describe the Software Testing Life Cycle among Software Developing Life Cycle, since those two go parallelly. Furthermore, one more task of this paper was to list the benefits of automation testing in a software development process. Each of these theoretical positions make's an important contribution to our understanding of the Automation tool.

To summarize, the Automation tool is build and already in use from the quality assurance team of Bestseller. All the main requirements that it had to follow are developed and running as expected. At the moment, the tool is still under development, however on a new phase, in order to apply some improvements.

Ultimately, this tool has a lot of capabilities but there is also space for improvements; it can be expanded and get used for extra purposes, such as monitoring in order to verify the availability to users at any time

Περίληψη

Μια σημαντική πτυχή της ανάπτυξης λογισμικού για οποιαδήποτε σελίδα ηλεκτρονικού εμπορίου, είναι οι δοκιμές διαδικτύου και η πιστοποίηση της ποιότητας από την ομάδα Δοκιμών. Λόγω του μεγάλου όγκου λειτουργιών, πεδίων και συστημάτων, σε συνδυασμό με τις διαδοχικές τροποποιήσεις λόγο των απαιτούμενων βελτιώσεων; είναι σπάνια η ύπαρξη ηλεκτρονικών καταστημάτων μεγάλου μεγέθους τα οποία ακολουθούν διαδικασίες παραγωγής λογισμικού (SDLC [1]) όπως η Agile, και ταυτόχρονα να διατηρούν τις σελίδες καθαρή από λάθη και ατέλειες.

Κατά συνέπεια, ο κύριος στόχος της πτυχιακής μου είναι η ανάπτυξη μιας εφαρμογής η οποία θα δέχεται πληροφορίες από το χρήστη και στη συνέχεια, θα εκτελέσει σενάρια δοκιμών. Η εφαρμογή αυτή θα είναι ουσιαστικά ένα εργαλείο το οποίο θα κάνει αυτοέλεγχο, χωρίς να έχει την ανάγκη της περεταίρω ανθρώπινης παρέμβασης (Automated Testing), στο ηλεκτρονικό κατάστημα ρούχων της Bestseller. Αυτό το Web Shop δραστηριοποιείται σε 14 διαφορετικές χώρες, έχοντας 12 διαφορετικές μάρκες, ενώ υποστηρίζει αρκετές διαφορετικές μεθόδους πληρωμής και κάποιες άλλες βασικές λειτουργίες για την ύπαρξη του. Ο απώτερος στόχος είναι η δοκιμή από άκρη σε άκρη (end to end), δηλαδή από τη στιγμή που οι χρήστες εισέρχονται στο ηλεκτρονικό κατάστημα μέχρι την ολοκλήρωση της περιήγησης και τελικά την παραγγελία.

Σε τελική ανάλυση, το εργαλείο αυτοματισμού θα πρέπει να διατηρεί και να βελτιστοποιεί την ποιότητα του λογισμικού, επαληθεύοντας ορισμένες βασικές λειτουργίες. Εν τω μεταξύ, ο χρόνος του δοκιμαστή που έχει εξοικονομηθεί από τον αυτοματισμό, μπορεί να διατεθεί σε ορισμένες άλλες κρίσιμες λειτουργίες και τα συστήματα. Έμμεσα, θα μειωθούν οι απαιτούμενοι πόροι της δοκιμής.

Το εργαλείο αυτοματισμού αναπτύχθηκε μέσω του Visual Studio σε C # γλώσσα με την υποστήριξητου εργαλείου αυτοματισμού Selenium.
Κατά τη διάρκεια της ανάπτυξης, διαπιστώθηκε ότι για να είναι το εργαλείο αυτοματισμού αποτελεσματικό και αποδοτικό, ορισμένες βασικές απαιτήσεις θα πρέπει να ακολουθηθούν, όπως η επεκτασιμότητα, η δυναμικότητα και η φιλικότητα του UI προς τον χρήστη.

Παράλληλα με την ανάπτυξη της εφαρμογής, η πτυχιακή αυτή περιλαμβάνει μια θεωρητική παρουσίαση ορισμένων μεθόδων, διαδικασιών και λειτουργιών. Οπως για παράδειγμα οι διαδικασίες παραγωγής λογισμικού σε σχέση με τις διαδικασίες δοκιμών του λογισμικού, δεδομένου ότι αυτές οι δύο δουλεύουν παράλληλα. Επιπλέον, ένα ακόμη θεωρητικό στοιχείο που αναλύεται στην εργασία αυτή, είναι τα οφέλη που προκύπτουν από τις δοκιμές αυτοματισμού κατά τη διαδικασία ανάπτυξης του λογισμικού. Κάθε μία από αυτές τις θέσεις είναι σημαντικές για τη συμβολή στην κατανόηση του εργαλείου αυτοματισμού.

Για να συνοψίσουμε, το εργαλείο αυτοματισμού έχει ολοκληρωθεί ενώ βρίσκεται ήδη σε χρήση από την ομάδα διασφάλισης ποιότητας της Bestseller. Όλες οι απαιτήσεις που θα έπρεπε να ακολουθηθούν έχουν εφαρμοστεί και λειτουργούν όπως ήταν αναμενόμενο. Αυτή τη στιγμή, το εργαλείο είναι υπό ανάπτυξη, προκειμένου να εφαρμοστούν κάποιες βελτιώσεις.

Σε τελική ανάλυση, το εργαλείο αυτό έχει πολλές δυνατότητες, παρόλαυτά υπάρχει χώρος για βελτιώσεις. Ένας επιπλέον ρεαλιστικός σκοπός για τον οποίο θα μπορούσε να χρησιμοποιηθεί, είναι η παρακολούθηση (monitoring), προκειμένου να ελέγξει ότι το κατάστημα είναι online οποιαδήποτε χρονική στιγμή.

[1] SDLC: Software Development Life Cycle

Πίνακας περιεχομένων

Ευχαριστίες 2
Abstract 3
Περίληψη 4
Πίνακας περιεχομένων 5
1. Εισαγωγή 7
2. Διαδικασία Παραγωγής Λογισμικού (SDLC) 9
2.1. Συλλογή απαιτήσεων / Ανάλυση 9
2.2. Σχεδιασμός 9
2.3. Ανάπτυξη κώδικα 10
2.4. Δοκιμές 11
2.5. Εφαρμογή του λογισμικού σε νέο περιβάλλον (Implementation/Deployment) 11
3. Μοντέλα Παραγωγής Λογισμικού 12
3.1. Μοντέλο Καταρράκτη (Waterfall Model): 13
3.2. Μοντέλο Πρωτοτύπου (Prototype model) 15
3.3. Σπειροειδές Μοντέλο (Spiral Model) 16
3.4. Μοντέλο επαλήθευσης και επικύρωσης (V- Model) 19
3.5. Agile 21
4. Διαδικασία δοκιμών Λογισμικού (STLC) 24
4.1. Στάδια και Φάσεις δοκιμών Λογισμικού 24
4.2 Κύριος στόχος των δοκιμών λογισμικού 26
4.3 Μεθοδολογίες δοκιμών Λογισμικού 26
4.4 Μοντέλα Δοκιμών Λογισμικού 29
5. Δοκιμές αυτοματισμού (Automated Testing) 30
5.1 Πως προέκυψε η ανάγκη της αυτόματης δοκιμής 30
5.2 Πλεονεκτήματα Αυτομάτων δοκιμών 31
5.3 Μειονεκτήματα αυτομάτου έλεγχου. 33
5.4 Εργαλεία διαθέσιμα στην αγορά 33
6. Περιγραφή του προϊόντος 36
6.1. Οι μάρκες του ηλεκτρονικού καταστήματος 36
6.2 Υπηρεσίες που προσφέρει η ιστοσελίδα 37
6.3 Περιγραφή της παραγωγής του ηλεκτρονικού καταστήματος 39
7. Περιγραφή της διαδικασίας Δοκιμών της εταιρίας Bestseller. 42
7.1 Δοκιμή καπνού (Smoke Testing) 43

7.2 Δοκιμές παλινδρόμησης (Regression test) 43
8. Ανάπτυξη του εργαλείου δοκιμών αυτοματισμού 45
8.1 Προκλήσεις της ανάπτυξης του εργαλείου 45
8.2 Σενάρια δοκιμών του εργαλείου αυτοματισμού 46
8.3 Περιβάλλον ανάπτυξης λογισμικού: Visual Studio 46
8.4 Γλώσσα προγραμματισμού: C# 47
8.5 Εισαγωγή στο εργαλείο Selenium 47
8.6 Εφαρμογή τριών επιπέδων (three-tier application) 48
9. Η αρχιτεκτονική του κώδικα 52
9.1 UML Class diagram 52
9.2 Activity Diagram 52
9.3 Βιβλιοθήκες 54
9.4 Ανάλυση βασικών εντολών του κώδικα 56
10. Αποτελέσματα 63
Συμπεράσματα 64
Βιβλιογραφία 65

Πίνακας περιεχομένων εικόνων

Εικόνα 2.1 SDLC 9
Εικόνα 3.2 Prototype 15
Εικόνα 3.3 piral Model 2 17
Εικόνα 3.4 V model 1 20
Εικόνα 3.5 Agile model 22
Εικόνα 6.1 Logo της εταιρείας 36
Εικόνα 7.1 Διαδικασία κυκλοφορίας λογισμικού 40
Εικόνα 8.1 Regression Test 44
Εικόνα 9.1 Class diagram UML 52
Εικόνα 9.10 Activity Diagram UML 53
Εικόνα 10.1 Time Test Results 64

1. Εισαγωγή

Δοκιμή είναι η διαδικασία που σκοπό έχει το προσδιορισμό της ποιότητας, της απόδοσης και της αξιοπιστίας σε κάτι προτού αυτό τεθεί σε ευρεία χρήση. Η εργασία αυτή θα ασχοληθεί με τις δοκιμές του λογισμικού και πώς αυτές μπορούν να γίνουν πιο αποτελεσματικά, σύντομα και αποδοτικά. Το θέμα της πτυχιακής σε πρακτικό επίπεδο, είναι η ανάπτυξη ενός εργαλείου το οποίο εφαρμόζεται σε μία ιστοσελίδα εμπορίου με σκοπό την εκτέλεση δοκιμών αυτοματισμού.

Στις μέρες μας, η τεχνολογία είναι αυτή που έχει αναπτυχθεί περισσότερο από οποιαδήποτε άλλη επιστήμη. Οι χρήστες είναι αρκετά προχωρημένοι επιζητώντας συνεχώς λογισμικά υψηλότερης ποιότητας, ενώ παρουσιάζουν και αρκετά προηγμένη αίσθηση αξιολόγησης αυτών. Οι δοκιμές αυτοματισμού αποτελούν έργο μείζονος σημασίας για εταιρίες, οι οποίες ακολουθούν μοντέλα με αυστηρούς κύκλους ανάπτυξης λογισμικού. Αυτό γιατί παρέχεται η δυνατότητα βελτίωσης της ποιότητας του λογισμικού που αναπτύσουν, με τρόπο δυναμικό.

Σε θεωρητικό επίπεδο, ο στόχος είναι να φέρει τον αναγνώστη πιο κοντά στις διαδικασίες παραγωγής και δοκιμών λογισμικού και τελικώς να αναλύσει τη σημασία του αυτοματισμού κατά τη διαδικασία των δοκιμών.

Πιο συγκεκριμένα στο πρώτο κεφάλαιο, γίνεται μια αναδρομή στις διαδικασίες παραγωγής λογισμικού (SDLC[2]). Ο σκοπός είναι κυρίως η ανάλυση τους ενώ αργότερα η περιγραφή των πιο σημαντικών μοντέλων που προκύπτουν εξ αυτών.

Τα μοντέλα στα οποία γίνεται αναφορά στο δεύτερο κεφάλαιο, είναι μερικά από τα πιο σημαντικά και χρησιμοποιούνται ως επί το πλείστον κατά τη διάρκεια του SDLC. Πέρα από την γενική περιγραφή των φάσεων τους, αναφέρονται επίσης τα πλεονεκτήματα και μειονεκτήματα τους. Η αναφορά αυτή είναι εξίσου σημαντική, καθώς με βάση αυτής γίνεται συνήθως και η τελική επιλογή του μοντέλου που θα εκτελεστεί, από το διευθυντή παραγωγής.

Στη συνέχεια, ξεκινάει και περιγραφή των διαδικασιών δοκιμών του λογισμικού (STLC[3]). Βέβαια, η ένταξη του αναγνώστη σε αυτή είναι αρκετά ομαλή, καθώς έχουν ήδη γίνει ορισμένες αναφορές και περιγραφές στις διαδικασίες δοκιμών από τα δυο προηγούμενα κεφάλαια. Εδώ ουσιαστικά γίνεται η ανάλυση τους και η περιγραφή των διαδικασιών, των φάσεων και ορισμένων από τα μοντέλα δοκιμών.

Στο πλαίσιο αυτό, ο αναγνώστης κατανοεί τις διαδικασίες δοκιμών, γεγονός που κάνει πιο εύκολη και κατανοητή την αναφορά στον αυτοματισμό. Εκείνο λοιπόν που έχει ιδιαίτερη σημασία στο πέμπτο κεφάλαιο, δεν είναι μόνο η αναφορά αλλά και η παρουσίαση όλων των πλεονεκτημάτων του σε σχέση με τα μειονεκτήματα. Εύκολα λοιπόν, μπορεί ο καθένας να συμπεράνει τους λόγους που οδήγησαν στην αρχική σκέψη του εργαλείου αυτοματισμού, για το ηλεκτρονικό κατάστημα της Bestseller.

Θα αποτελούσε παράλειψη σε αυτό το σημείο να μην γίνει αναφορά της ιστοσελίδας στην οποία εφαρμόζεται το εργαλείο και τους λόγους που προέκυψε η ανάγκη αυτή. Αυτό γίνεται στο έκτο και έβδομο κεφάλαιο. Πέρα από τη περιγραφή του ηλεκτρονικού καταστήματος, υπάρχει και αναφορά στο κύκλο

[2] SDLC: Software Development Life Cycle
[3] STLC: Software Testing Life Cycle

παραγωγής λογισμικού όπως και στο κύκλο δοκιμών της εταιρίας. Αυτό είναι σημαντικό για τη περιγραφή της δομής του τελικού προϊόντος που αναπτύχθηκε.

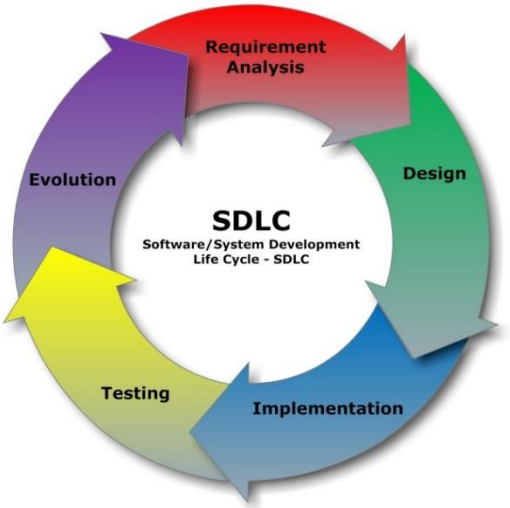
Δεν υπάρχει αμφιβολία, πως δύο από τα πιο σημαντικά κεφάλαια, είναι το όγδοο και ένατο καθώς αποτελούν το πρακτικό μέρος της πτυχιακής. Το πρώτο περιλαμβάνει μια γενικότερη περιγραφή του εργαλείου και των βασικών συντελεστών του, όπως είναι το εργαλείου αυτοματισμού της Selenium, το περιβάλλοντος και η γλώσσας ανάπτυξης της εφαρμογής. Παράλληλα παρουσιάζεται και η εφαρμογή τριών επιπέδων, που χρησιμοποιείται για την πιο αναλυτική περιγραφή της εφαρμογής.

Η ουσιαστική περιγραφή του πρακτικού μέρους γίνεται στο ένατο κεφάλαιο. Εκεί βρίσκονται τα βασικά διαγράμματα UML και Activity που αποτελούν τεχνικές εικονοποίησης της εφαρμογής. Επιπλέον, παραθέτονται πληροφορίες για την αρχιτεκτονική του κώδικα, τις βιβλιοθήκες που χρησιμοποιούνται ενώ κλείνει με τη σύνοψη και ανάλυση των βασικών εντολών του κώδικα.

Τέλος, η πτυχιακή άσκηση ολοκληρώνεται με την παράθεση των αποτελεσμάτων και συμπερασμάτων που προέκυψαν από τη χρήση του εργαλείου.

2. Διαδικασία Παραγωγής Λογισμικού (SDLC[4])

Η διαδικασία παραγωγής λογισμικού ή Κύκλος Ανάπτυξης Λογισμικού (Software Development Life Cycle), είναι μια διαδικασία που εξασφαλίζει πως το παραγόμενο λογισμικό ικανοποιεί τις απαιτήσεις. Κάθε φάση του κύκλου αυτού, έχει τη δική της διαδικασία και παραδίδει φόρτο εργασίας στην επόμενη φάση. Υπάρχουν συνήθως 5 στάδια, αρχής γενομένης με την ανάλυση και τη συλλογή των απαιτήσεων και τελειώνει με την εκτέλεση. Ας δούμε με μεγαλύτερη λεπτομέρεια τη κάθε φάση.



Εικόνα 2.1 SDLC

2.1. Συλλογή απαιτήσεων / Ανάλυση

Αυτή η φάση είναι κρίσιμη για την επιτυχία του σχεδίου.
Οι προσδοκίες του πελάτη πρέπει να καταγραφούν με μεγάλη λεπτομέρεια και τεκμηρίωση. Αυτή είναι μια επαναληπτική διαδικασία που απαιτεί συνεχείς επικοινωνία μεταξύ των ενδιαφερομένων , τους τελικούς χρήστες και την ομάδα σχεδιασμού του έργου. Για τη συγκέντρωση απαιτήσεων χρησιμοποιούνται οι ακόλουθες τεχνικές:

- Εντοπισμός και καταγραφή απαιτήσεων των ενδιαφερομένων φορέων, χρησιμοποιώντας συνεντεύξεις των πελατών και έρευνες .
- Δημιουργία πολλαπλών περιπτώσεων χρήσης, για να περιγραφεί κάθε ενέργεια που ο χρήστης θα μπορεί να εκχωρήσει στο νέο σύστημα .
- Κατασκευή πρωτοτύπων που μπορούν που θα παρουσιαστούν στο πελάτη πως θα μοιάσει τελικό προϊόν.

Τα προαναφερθέντα είναι μείζονος σημασίας στο περιβάλλον της εταιρίας, καθώς έχουν άμεση σχέση με την ικανοποίηση του πελάτη. Αρχικά για να κάνει την εκτίμηση και στην συνέχεια να ερμηνεύσει την επιτυχία στο νέο κομμάτι του λογισμικού .

2.2. Σχεδιασμός

Οι απαιτήσεις του τεχνικού σχεδιασμού παρασκευάζονται σε αυτή τη φάση από τους επικεφαλείς του προσωπικού ανάπτυξης λογισμικού. Αρκετές φορές, σε αυτή τη φάση περιλαμβάνονται οι αρχιτέκτονες και προγραμματιστές του έργου. Οι απαιτήσεις που προέκυψαν από τη ομάδα σχεδιασμού, χρησιμοποιούνται

[4] SDLC: Software Development Life Cycle

για να καθορίσουν το πώς θα γίνει η ανάπτυξη της εφαρμογής. Οι τεχνικές απαιτήσεις θα περιγράψουν λεπτομερώς τους πίνακες της βάσης δεδομένων που πρέπει να προστεθούν, τις νέες συναλλαγές που θα καθοριστούν, τις διαδικασίες ασφάλειας, τα φυσικά εξαρτήματα και τις απαιτήσεις του συστήματος .

Ας δούμε πιο αναλυτικά ορισμένες από τις δραστηριότητες που εμπλέκονται σε αυτό το στάδιο :

Η ανάλυση κινδύνου

- Οι απειλές και τα τρωτά σημεία που μπορεί να προκύψουν από τις αλληλεπιδράσεις με άλλα συστήματα .
- Ο κώδικας πρέπει να αναλυθεί για να διαπιστωθεί εάν υπάρχουν κενά στην ασφάλεια του.
- Έργα υψηλού κινδύνου που αφορούν προσωπικά δεδομένα πελατών, απαιτούν την αναθεώρηση από κάποιο νομικό τμήμα . Αυτή η διεργασία θα πρέπει να ερευνήσει τον τρόπο και το είδος των δεδομένων που μπορεί να συλλέξει η εταιρία, καθώς παράλληλα τα δικαιώματα και τις άδειες για να γίνουν αλλαγές . Αυτή η επανεξέταση είναι ιδιαίτερα αναγκαία στα έργα των επιχειρήσεων.

Λειτουργικές Προδιαγραφές

- Περιλαμβάνει μια περιγραφή της διασύνδεσης των απαιτήσεων όπως είναι ο καθορισμός των πεδίων δεδομένων εισόδου (επιτρέπει μόνο αλφαριθμητικά, μπορεί να μείνει κενό, κτλ)
- Σημαντικές λεπτομέρειες όπως: μπορεί η εισαγόμενη ημερομηνία να αποτελεί παρελθόν; Σε ποια ζώνη ώρας θα βρίσκονται οι χρήστες;
- Ροή εργασιών – έπειτα από κλικ σε κάποιο κουμπί, ποια σελίδα θα ακολουθήσει;
- Διαδικασία δοκιμών για κάθε ενημέρωση της βάσης δεδομένων.

Μη λειτουργικές προδιαγραφές

- Επεκτασιμότητα του συστήματος– Την ευκολία με την οποία το σύστημα θα επιτρέπει νέες βελτιώσεις και λειτουργίες σε κάθε μελλοντική εγκατάσταση και ενημέρωση. Αυτό είναι κρίσιμης σημασίας για κάθε εφαρμογή η οποία θα ενημερώνετε συχνά και θα προστίθενται νέα χαρακτηριστικά.
- Επιδόσεις και χρόνο απόκρισης- Έχει καθοριστεί ο αναμενόμενος χρόνος απόκρισης;
- Περιορισμός πόρων- υπάρχουν περιορισμοί που πρέπει να ληφθούν υπόψη σε αυτή τη φάση; Κοινοί περιορισμοί παρουσιάζονται στο χώρο του δίσκου , το εύρος ζώνης , κλπ.

2.3. Ανάπτυξη κώδικα

Η φάση αυτή αποτελείται από την από την ανάπτυξη του κώδικα και τον έλεγχο αυτού από την ομάδα της ανάπτυξης λογισμικού. Μετά από κάθε στάδιο ο προγραμματιστής παρουσιάζει τη δουλειά που έχει ολοκληρώσει στην ομάδα σχεδιασμού του έργου, με σκοπό τη βελτίωση τυχών ατελειών. Σημαντικό είναι βέβαια σε αυτό το στάδιο οι προγραμματιστές να είναι ανοιχτοί στις διάφορες βελτιώσεις που πρόκειται να συζητηθούν. Συνήθως αυτή είναι η μεγαλύτερη φάση του κύκλου ανάπτυξης λογισμικού SDLC.
Στη συνέχεια το τελικό προϊόν περνάει στην ομάδα δοκιμών.

2.4. Δοκιμές

Από τη στιγμή που η εφαρμογή θα περάσει στο περιβάλλον δοκιμών (Test Environment), θα πρέπει να εφαρμοστούν διαφορετικοί τύποι δοκιμών. Από τους πιο καίριους είναι αυτός της ένταξης του στο νέο περιβάλλον που ονομάζεται δοκιμή καπνού (Smoke Test) και αναλύεται αργότερα.

Η τελευταία φάση του ελέγχου, είναι οι δοκιμές αποδοχής (User acceptance testing) που εκτελούνται από την ομάδα δοκιμών, για την εξασφάλιση πως το σύστημα ανταποκρίνεται στις προσδοκίες τους. Σε αυτό το σημείο, υπάρχει μεγάλη πιθανότητα να βρεθούν ελαττώματα , που οδηγούν σε περεταίρω ανάλυση, σχεδιασμό ή ανάπτυξη του κώδικα.

Έπειτα ακολουθεί η πιστοποίηση, πως η εφαρμογή πληρεί τις κατάλληλες προϋποθέσεις(sign-off). Αυτό σηματοδοτεί την ένταξη του συστήματος στο περιβάλλον της παραγωγής (Implementation/Deployment).

2.5. Εφαρμογή του λογισμικού σε νέο περιβάλλον (Implementation/Deployment)

Το μέγεθος της εφαρμογής θα καθορίσει τη πολυπλοκότητα της ανάπτυξης (Deployment).
Απαιτείται η εκπαίδευση του προσωπικού που αναλαμβάνει τη δουλειά αυτή, ενώ πολλές φορές και η υποστήριξη από την ομάδα υποστήριξης του εργασιών (Operations Support).
Η εξαγωγή του συστήματος από το περιβάλλον δοκιμών στην παραγωγή (Roll-out) μπορεί να γίνει ολοκληρωμένα (Full Implementation) είτε σε στάδια αρχής γενομένης με ένα κλάδο, και έπειτα προσθέτοντας τα υπόλοιπα κομμάτια. Κατά τη διάρκεια των κύκλων ανάπτυξης λογισμικού(SDLC) χρησιμοποιείται μία από αυτές τις δύο μεθόδους.

Καταρράκτης (Waterfall) είναι το πιο παραδοσιακό από τα μοντέλα ανάπτυξης λογισμικού. Διαθέτει ένα καλά δομημένο σχέδιο, ενώ παράλληλα και απαιτήσεις οι οποίες πρέπει να ακολουθούνται κατά γράμμα. Αυτή η μέθοδος είναι ιδανική για μεγάλα έργα που προαπαιτούν πολλούς μήνες δουλειάς για να αναπτυχθεί η εφαρμογή.

Η Agile Μεθοδολογία είναι πιο ευέλικτη στη διαδικασία των απαιτήσεων , του σχεδιασμό και ανάπτυξης της εφαρμογής ενώ μεγάλο πλεονέκτημα είναι και η αντιληπτικότητας που διαθέτει στο χρήστη . Αυτή η διαδικασία λειτουργεί καλύτερα για μικρότερα έργα, με προσδοκίες τη συνεχή βελτίωσης της εφαρμογής.

Στο επόμενο κεφάλαιο, θα ακολουθήσει η εκτενέστερη περιγραφή των δύο μοντέλων.

3. Μοντέλα Παραγωγής Λογισμικού

Είναι πολύ σημαντικό για την επιστήμη της πληροφορικής, η εύρεση και θεμελίωση μεθόδων για τη περιγραφή, κατασκευή και συντήρηση λογισμικού υψηλής ποιότητας, παρέχοντας επίσης και οδηγίες που αφορούν στις ενέργειες που θα πρέπει να πραγματοποιηθούν.
Πρέπει να γίνει κατανόηση της σειράς με την οποία θα πρέπει να γίνουν αυτές οι ενέργειες, τις προϋποθέσεις καθώς και τα κριτήρια με τα οποία θα ολοκληρωθεί μια ενέργεια, για να γίνει μετάβαση στην επόμενη.

Τα μοντέλα διαδικασιών παραγωγής λογισμικού είναι αφηρημένες αναπαραστάσεις κάποιας διαδικασίας παραγωγής λογισμικού. Επομένως κάθε μοντέλο παρέχει πληροφορίες για ορισμένες μόνο πλευρές της διαδικασίας. Δεν υπάρχει μοναδιαίος τρόπος για τον προσδιορισμό των ενεργειών ανάπτυξης λογισμικού αν και σε γενικές γραμμές θα πρέπει να περιλαμβάνονται οι διαδικασίες του σχεδιασμού, της ανάπτυξης του κώδικα, του ελέγχου και της εφαρμογής του (Implementation).

Τα μοντέλα που χρησιμοποιούνται επί το πλείστων στις ημέρες μας για τη διαδικασία παραγωγής λογισμικού, είναι τα παρακάτω:

- Μοντέλο Καταρράκτη (Waterfall)
- Μοντέλο Πρωτοτύπου (Prototype model)
- Σπειροειδές Μοντέλο (Spiral)
- V Model
- Agile Development

3.1. Μοντέλο Καταρράκτη (Waterfall Model):

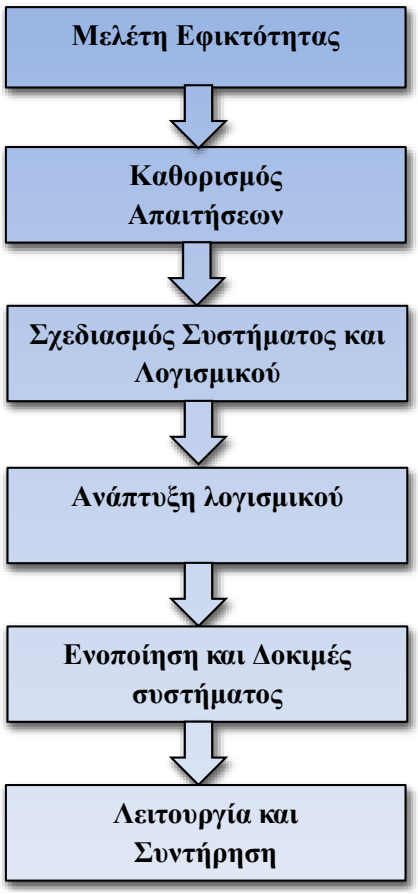
Το μοντέλο του Καταρράκτη (waterfall) αναπτύχθηκε μόλις το 1970 και περιλαμβάνει 6 διακριτές φάσεις. Πρέπει να τονίσουμε πώς ήταν το πρώτο μοντέλο που δημιουργήθηκε και έγινε ευρέως αποδεκτό. Παράλληλα, παραμένει δημοφιλές αφού συμβάλλει στην επιτυχή κατασκευή αξιόπιστων προϊόντων σε μικρό χρονικό διάστημα.

Στο μοντέλο αυτό οι διάφορες φάσεις διαχωρίζονται και ακολουθούνται σειριακά. Η κάθε φάση παράγει κάποια αποτελέσματα τα οποία χρησιμοποιούνται από τις επόμενες φάσεις που ακολουθούν. Η κορύφωση γίνεται μέσω της διαδικασίας επικύρωσης ή επαλήθευσης των παραγόμενων προϊόντων, με σκοπό την απαλοιφή τυχόν σφαλμάτων.

Το μοντέλο αυτό διαφοροποιείται περισσότερο στον τρόπο που γίνεται ο έλεγχος, η διόρθωση, η επαλήθευση, η επικύρωση καθώς και ο τρόπος που θα πραγματοποιηθούν αυτές. Δηλαδή σε ποια φάση θα πρέπει να επιστρέψουμε αν χρειαστεί.

Οι φάσεις του μοντέλου Καταρράκτη

- **Μελέτη Εφικτότητας (Feasibility):** Εδώ ο διαχειριστής του έργου, μαζί με τη διοίκηση θα εξετάσει το κόστος, τη διάρκεια, τους πόρους , τον προϋπολογισμό για να καταλήξει τελικά στο πόσο εφικτό είναι το έργο.
- **Ανάλυση και καθορισμός απαιτήσεων (Analysis):** Η ομάδα σχεδιασμού μαζί με τον διαχειριστή του έργου, συλλέγουν τις απαιτήσεις από τους πελάτες και αναλύουν το πεδίο εφαρμογής του σχεδίου.
- **Σχεδιασμός συστήματος και λογισμικού (Design):** Σε αυτή τη φάση, ο αρχιτέκτονας του συστήματος και οι προγραμματιστές διαδραματίζουν καθοριστικό ρόλο στη δημιουργία και τον καθορισμό της συνολικής δομής. Σχεδιάζουν το περιβάλλον διεπαφής του χρήστη, τη βάση δεδομένων και την επιχειρηματική λογική που περιγράφεται αναλυτικότερα στο κεφάλαιο 8.6
- Τα κύρια ζητήματα αυτής της φάσης του σχεδιασμού είναι: η Επεκτασιμότητα, η Αντοχή, η Αξιοπιστία, τα Λάθη/Ανοχή σε λάθη, η Ασφάλεια, η Συντηρησιμότητα, η Συμβατότητα, και η Επαναχρησιμοποίηση.
- **Ανάπτυξη λογισμικού (Coding/ Implementation):** Βασιζόμενοι στο σχεδιασμό και τις απαιτήσεις που παρέλαβαν από την ομάδα σχεδιασμού, οι προγραμματιστές παράγουν το λογισμικό.
- **Ενοποίηση και Δοκιμές συστήματος (System Integration and Testing):** Σε αυτή τη φάση, το προϊόν που προκύπτει από το development ενσωματώνεται στο περιβάλλον των δοκιμών. Ο έλεγχος είναι η διαδικασία της δοκιμής και στη συνέχεια της αξιολόγησης ενός



Εικόνα 3.1 Waterfall

συστήματος ή μέρος αυτού, για την εξακρίβωση ότι πληροί τις συγκεκριμένες απαιτήσεις που δόθηκαν από την ομάδα των σχεδιαστών.

- **Λειτουργία και συντήρηση (Release & Maintenance):** Αυτό είναι το τελευταίο στάδιο του μοντέλου, όπου η εφαρμογή του λογισμικού παραδίδεται στους πελάτες. Τυχόν προβλήματα ή βελτιώσεις θεωρούνται ως μέρος της συντήρησης.

Πλεονεκτήματα

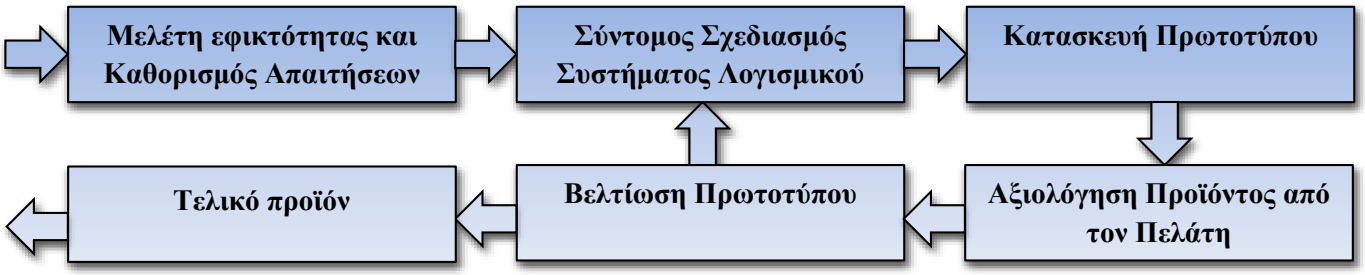
- Απλή και εύκολη στη χρήση
- Εύκολη διαχείριση λόγω της ανελαστικότητας του μοντέλου - Κάθε φάση έχει συγκεκριμένα παραδοτέα και διαδικασία αξιολόγησης.
- Για να προχωρήσουμε στην επόμενη φάση, θα πρέπει να έχει ολοκληρωθεί η προηγούμενη.
- Λειτουργεί καλά για μικρά έργα, όπου οι απαιτήσεις είναι πολύ συγκεκριμένες και κατανοητές.

Μειονεκτήματα

- Η αναπροσαρμογή του στόχου κατά της διαδικασίας παραγωγής, σκοτώνει το έργο.
- Μεγάλο ποσοστό κινδύνου και αβεβαιότητας για το αποτέλεσμα.
- Αδυναμία εκτέλεσης πολύπλοκου και αντικειμενοστραφή έργου.
- Αδυναμία εκτέλεσης, μακροχρόνιου και εν εξελίξει έργου.
- Αδυναμία εκτέλεσης έργου όπου υπάρχει υψηλός κίνδυνος αλλαγής των απαιτήσεων.

3.2. Μοντέλο Πρωτοτύπου (Prototype model)

Η βασική ιδέα του μοντέλου πρωτοτύπου, είναι ότι η υλοποίηση του συστήματος λογισμικού μπορεί να ξεκινήσει χωρίς να έχουν οριστικοποιηθεί οι απαιτήσεις του πελάτη. Οι προγραμματιστές στηρίζονται σε ένα πρόχειρο πρωτότυπο με τις υπάρχουσες βασικές απαιτήσεις.
Χρησιμοποιώντας αυτό το πρωτότυπο, ο πελάτης μπορεί να πάρει μια "αίσθηση" του συστήματος, δεδομένου ότι οι αλληλεπιδράσεις με το πρωτότυπο μπορεί να επιτρέπουν στον πελάτη να κατανοήσει καλύτερα τις απαιτήσεις του επιθυμητού συστήματος.
Τα πρωτότυπα είναι μια ελκυστική ιδέα για πολύπλοκα και μεγάλα συστήματα, για τα οποία δεν υπάρχει αυτόματη διαδικασία ή υπάρχον σύστημα για να βοηθήσει στον καθορισμό των απαιτήσεων.



Εικόνα 3.2 Prototype

Πλεονεκτήματα

- Οι χρήστες συμμετέχουν ενεργά στην ανάπτυξη.
- Οι χρήστες έχουν καλύτερη κατανόηση του συστήματος που αναπτύχθηκε, στο τέλος της παραγωγής.
- Λάθη μπορούν να ανιχνευθούν πολύ νωρίτερα.
- Ταχύτερη ανάδραση (feedback) από το χρήστη είναι διαθέσιμη, που οδηγεί σε καλύτερες λύσεις.
- Η έλλειψη λειτουργικότητας μπορεί να προσδιοριστεί εύκολα.
- Μπερδεμένες ή δύσκολες λειτουργίες μπορούν να προσδιοριστούν πιο εύκολα.
- Επικύρωση απαιτήσεων, γρήγορη εφαρμογή, Ελλιπής αλλά Λειτουργική.

Μειονεκτήματα

- Αποτελεί σύστημα που ουσιαστικά εφαρμόζει και στη συνέχεια επιδιορθώνει, αυτό μεταφράζεται σε χαμένο χρόνο και χρήμα.
- Πρακτικά, αυτή η μέθοδος μπορεί να αυξήσει την πολυπλοκότητα του συστήματος, καθώς και το πεδίο εφαρμογής του συστήματος μπορεί να επεκταθεί πέρα από τα αρχικά σχέδια.
- Ελλιπής ή ανεπαρκής ανάλυση του προβλήματος.
- Η Ελλιπής εφαρμογή, μπορεί να αποτρέψει το πλήρες σύστημα να λειτουργήσει σωστά.

3.3. Σπειροειδές Μοντέλο (Spiral Model)

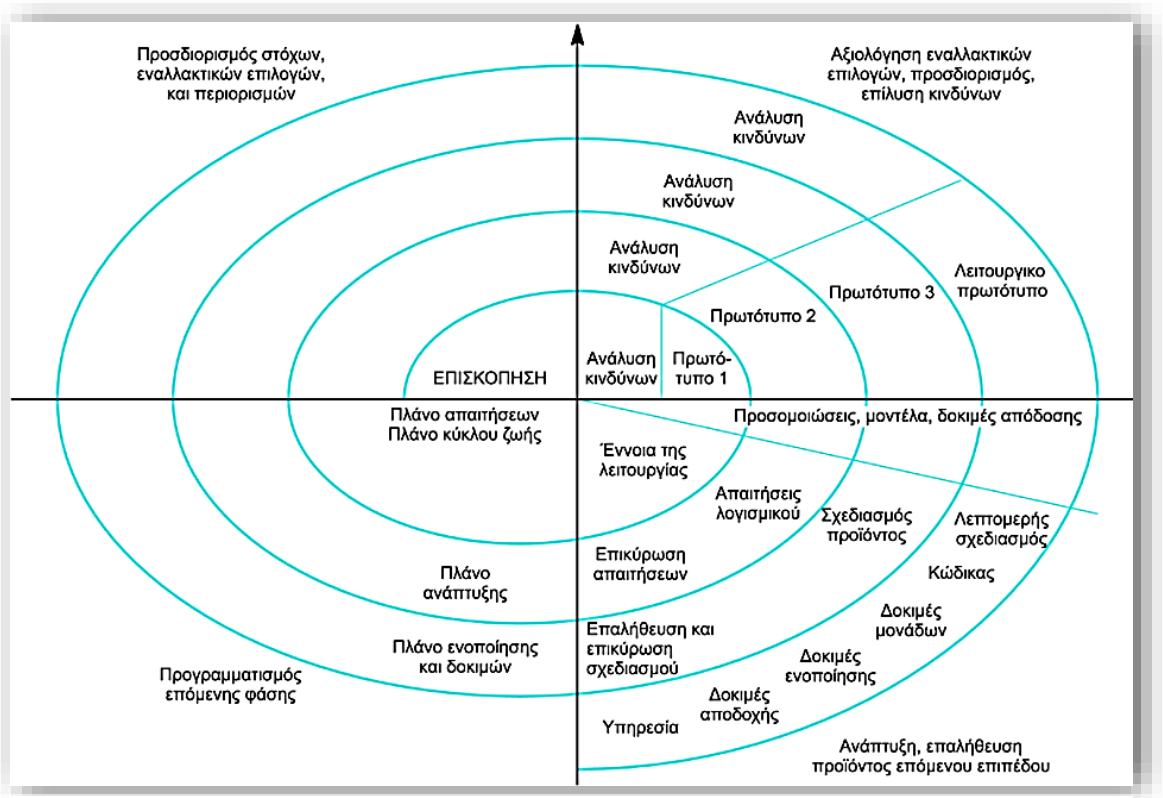
Η σπειροειδής διαδικασία παραγωγής, είναι ένα είδος μοντέλου ανάπτυξης λογισμικού με επαναληπτική διαδικασία, το οποίο γενικά εφαρμόζεται σε σχέδια υψηλού κινδύνου. Σε αυτή τη μέθοδο, που συνδυάζουν τα χαρακτηριστικά των δύο προ αναφερθέντων μοντέλων, δηλαδή του καταρράκτη και του πρωτοτύπου.

Αποτελείται από τέσσερις φάσεις: Σχεδιασμός, Ανάλυση Κινδύνου, κατασκευή και Αξιολόγηση. Κάθε βρόχος σε μια σπείρα αντιπροσωπεύει μια φάση ανάπτυξης, ενώ τα προγράμματα λογισμικού περνάνε επανειλημμένα και διαδοχικά μέσα από αυτές τις φάσεις.

Οι Φάσεις του Σπειροειδούς μοντέλου

- **Ορισμός αντικειμενικών στόχων:** Προσπάθεια κατανόησης των στόχων του προϊόντος, τις εναλλακτικές του σχεδιασμού και τους περιορισμούς που επιβάλλονται λόγω του κόστους, της τεχνολογίας, του χρονοδιαγράμματος, κ.λπ.
- **Αξιολόγηση και περιορισμός κινδύνων:** Στη φάση αυτή γίνεται η προσπάθεια εύρεσης εναλλακτικών προσεγγίσεων που μπορούν να εφαρμοστούν, προκειμένου το λογισμικό να πληρεί όλους τους περιορισμούς. Πιο συγκεκριμένα εδώ εξετάζονται τα λειτουργικά και τεχνικά ζητήματα, με σκοπό το περιορισμό κινδύνων που ελλοχεύουν. Παράλληλα γίνεται και αξιολόγηση όλων αυτών των παραγόντων που καθορίζουν τη μελλοντική δράση.
- **Ανάπτυξη και επικύρωση:** Σε αυτή τη φάση έχουμε την προγραμματισμένη ανάπτυξη του προϊόντος από τους προγραμματιστές, ενώ παράλληλα γίνονται και οι δοκιμές του από την ομάδα δοκιμών. Για την ανάπτυξη μπορούν να εφαρμοστούν το μοντέλο καταρράκτη ή της σταδιακής προσέγγισης.
- **Προγραμματισμός:** Γίνεται η επανεξέταση και επισκόπηση του έργου, λαμβάνοντας υπόψη όλες τις παραμέτρους. Παράλληλα προσδιορίζονται τα ζητήματα που πρέπει να επιλυθούν και λαμβάνονται τα απαραίτητα μέτρα, ενώ γίνεται και ο προγραμματισμός της επόμενης φάσης της σπειροειδούς ανάπτυξη ανάπτυξης

Η ανάλυση είναι το βασικό γνώρισμα που εφαρμόζονται σε αυτό το μοντέλο. Μεγάλα, ακριβά ή πολύπλοκα έργα χρησιμοποιούν τη διαδικασία παραγωγής λογισμικού. Εάν κάποιος, σε οποιοδήποτε σημείο του χρόνου, αισθάνεται το κίνδυνος που ενέχει στο έργο είναι πολύ μεγαλύτερος από τον αναμενόμενο, μπορεί να το εγκαταλείψει. Η αξιολόγηση των διαφορετικών φάσεων μπορεί να γίνουν από κάποιον που ανήκει στο περιβάλλον του έργου ή και από κάποιον πελάτη εκτός αυτού.



Εικόνα 3.3 piral Model 2

Πλεονεκτήματα

- Οι Φάσεις Ανάπτυξης μπορούν να καθοριστούν από τον επικεφαλή του έργου, ανάλογα με την πολυπλοκότητα του έργου καθώς, σπειροειδές μοντέλο είναι ένα από τα πιο ευέλικτα SDLC.
- Εύκολος και αποτελεσματικός έλεγχος στο έργο. Κάθε φάση, καθώς και κάθε βρόγχος, απαιτεί την επανεξέταση από τους υπεύθυνους. Αυτό καθιστά το μοντέλο πιο διαφανές. Παράλληλα, η διαχείριση κινδύνων είναι μία από τις ενσωματωμένες δυνατότητες του μοντέλου.
- Αλλαγές μπορούν να εισαχθούν και να αφομοιωθούν από το μοντέλο κατά τη διάρκεια της παραγωγής του λογισμικού. Παράλληλα η αντιμετώπιση αυτών των αλλαγών δεν αποτελεί μεγάλο πονοκέφαλο για τον επικεφαλή του έργου, όπως θα συνέβαινε με άλλα μοντέλα.
- Είναι κατάλληλο για σχέδια υψηλού κινδύνου, όπου οι ανάγκες των πελατών ενδέχεται να είναι ασταθείς.

Μειονεκτήματα

- Το κόστος που προκύπτει από αυτό το μοντέλο είναι συνήθως υψηλό.
- Πρόκειται για μια πολύπλοκη προσέγγιση, ειδικά για έργα με σαφείς και πλήρεις απαιτήσεις.
- Οι κανόνες και τα πρωτόκολλα πρέπει να ακολουθούνται σωστά για την αποτελεσματική εφαρμογή αυτού του μοντέλου. Απαιτεί σκληρή και συστηματική εργασία.
- Δεν είναι κατάλληλο για έργα που ελλοχεύουν κίνδυνοι.
- Είναι δύσκολη και σκληρή η σύνταξη του προϋπολογισμού και η συγκέντρωση των τρεχόντων απαιτήσεων ανά πάσα στιγμή, όσο η διαδικασία παραγωγής λογισμικού προχωράει.

- Ποσοστό της τεκμηρίωσης που απαιτείται στα ενδιάμεσα στάδια, καθιστά τη διαχείριση των έργων πολύπλοκη.

3.4. Μοντέλο επαλήθευσης και επικύρωσης (V- Model)

Ένα από τα πιο σημαντικά SDLC μοντέλα είναι και το V-Model, στο οποίο η εκτέλεση των διαδικασιών συμβαίνει διαδοχικά σε σχήμα V. Είναι επίσης γνωστό ως μοντέλο επαλήθευσης και επικύρωσης (Verification and Validation). Αποτελεί μια επέκταση του μοντέλου Καταρράκτη και βασίζεται στην επικόλληση μιας φάσης δοκιμών ελέγχου για κάθε αντίστοιχο στάδιο ανάπτυξης. Αυτό σημαίνει ότι για κάθε φάση της διαδικασίας παραγωγής, υπάρχει μια άμεση σχέση φάσης των δοκιμών. Είναι ένα πειθαρχημένο μοντέλο και κάθε φάση ξεκινά μόνο μετά την ολοκλήρωση της προηγούμενης.

Οι Φάσεις του V-model

- **Ανάλυση και Καθορισμός απαιτήσεων** (Requirement Analysis): Απαιτήσεις όπως BRS [5]και SRS[6] βρίσκονται στην αρχή της διαδικασίας παραγωγής αυτής, όπως το μοντέλο καταρράκτη. Όμως , σε αυτό το μοντέλο πριν ξεκινήσει η ανάπτυξη του λογισμικού, δημιουργείται ένα σχέδιο δοκιμής του συστήματος. Το σχέδιο δοκιμών εστιάζει στη λειτουργικότητα και μελετά κατά πόσο ανταποκρίνεται στις απαιτήσεις .
- **Σχεδιασμός υψηλού επιπέδου** (High-level design)**:** Η φάση αυτή εστιάζει στην αρχιτεκτονική του συστήματος και το σχεδιασμό. Παρέχει επισκόπηση της λύσης , της πλατφόρμας , του συστήματος, του προϊόντος και των υπηρεσιών / διαδικασιών . Ένα σχέδιο δοκιμής της ενσωμάτωσης (Integration Test) δημιουργείται σε αυτή τη φάση, προκειμένου να δοκιμαστεί η δυνατότητα συνεργασίας των κομματιών του συστήματος λογισμικού.
- **Σχεδιασμός των προδιαγραφών** (Low-level design)**:** φάση όπου σχεδιάζονται τα πραγματικά στοιχεία του λογισμικού. Ορίζεται η λογική για κάθε συνιστώσα του συστήματος . Διάγραμμα κλάσης με όλες τις μεθόδους και τη σχέση μεταξύ των τάξεων υπάγεται στο σχεδιασμό αυτό. Παράλληλα, σε αυτή τη φάση δημιουργούνται και οι δοκιμές τον διαφόρων εξαρτημάτων του συστήματος.
- **Η φάση της Κωδικοποίησης** (Coding): Η πραγματική κωδικοποίηση των ενοτήτων του συστήματος που αποφασίστηκαν κατά τη φάση του σχεδιασμού, παραλαμβάνονται στη φάση κωδικοποίησης. Η κατάλληλη γλώσσα προγραμματισμού αποφασίζεται ανάλογα με το σύστημα και τις αρχιτεκτονικές απαιτήσεις. Η κωδικοποίηση εκτελείται με βάση των οδηγιών και των προτύπων κωδικοποίησης. Ο κώδικας περνά μέσα από πολλές αξιολογήσεις και τροποποιείται με στόχο την καλύτερη απόδοση.

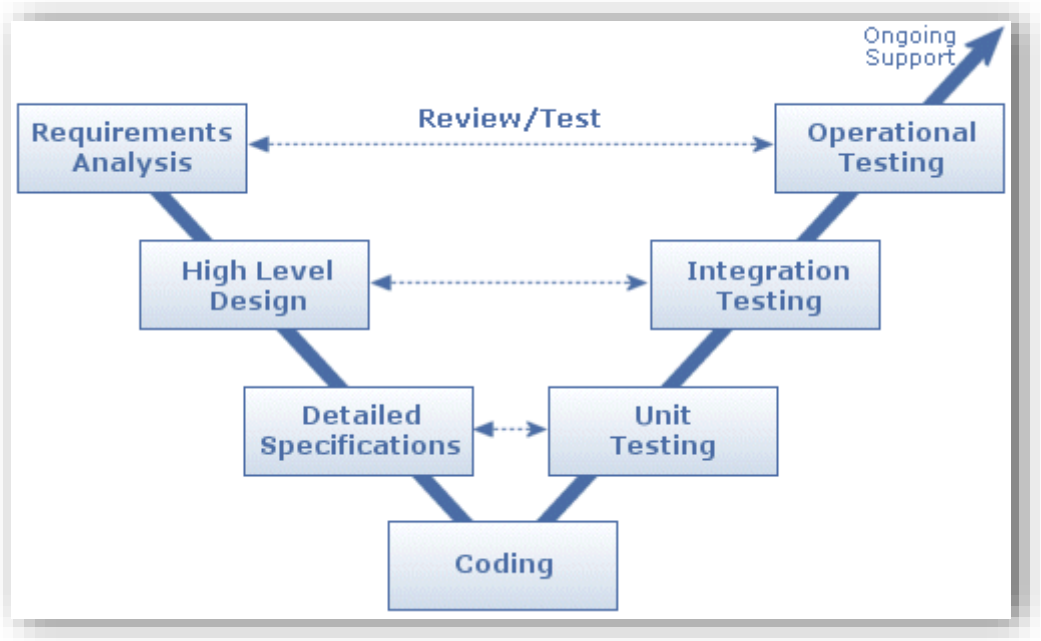
[5] BRS: Business Required Specifications
[6] SRS: Software Required Specifications

Πλεονεκτήματα

- Για να προχωρήσουμε στην επόμενη φάση, θα πρέπει να έχει ολοκληρωθεί η προηγούμενη.
- Λειτουργεί πολύ καλά με μικρότερα έργα, όπου οι απαιτήσεις είναι κατανοητές.
- Απλό και εύκολο στην κατανόηση και τη χρησιμοποίηση.
- Εύκολη διαχείριση λόγω της ακαμψίας του μοντέλου. Κάθε φάση έχει συγκεκριμένα παραδοτέα και μια διαδικασία αναθεώρησης.

Μειονεκτήματα

- Έλλειψη ευελιξίας.
- Το λογισμικό αναπτύσσεται σχετικά αργά- κατά τη φάση της κωδικοποίησης.
- Εάν συμβαίνουν τυχόν αλλαγές κατά τη διάρκεια της παραγωγής του λογισμικού, τότε θα πρέπει να ενημερωθούν όλα τα έγγραφα με τις απαιτήσεις, τις προδιαγραφές και τις δοκιμές.



Εικόνα 3.4 V model 1

3.5. Agile

Η Agile είναι μια ευέλικτη επαναληπτική μέθοδος (Interactive Method) για να καθορίσουμε τις απαιτήσεις του προϊόντος με τρόπο ευέλικτο και διακρατικό. Η ανάπτυξη του λογισμικού γίνεται σταδιακά, με την ομάδα σχεδιασμού να χωρίζει το έργο σε μικρούς κύκλους διάρκειας λίγων εβδομάδων, έως και μήνα. Μέσα σε αυτούς τους κύκλους η ομάδα υλοποιεί και παραδίδει σταδιακά τα χαρακτηριστικά του προϊόντος του έργου. Εκτός από την ομάδα που αφοσιώνεται στο έργο (απαγορεύεται το Multi-Tasking), υπάρχει ο ιδιοκτήτης του προϊόντος (Product Owner) που είναι υπεύθυνος για την επιχειρηματική αξία (Business Value) του έργου και ο Scrum Master ρόλος που εξασφαλίζει ότι η ομάδα τεχνολογίας (Προγραμματιστές, Γραφίστες και Δοκιμαστές) είναι αποδοτική και παραγωγική.

Οι φάσεις του Agile μοντέλου

Το μοντέλο αποτελείται από τέσσερεις φάσεις:

- **Φάση Προετοιμασίας (Preparation):**

αντιπροσωπεύει την επιλογή των βασικών διαδικασιών και την εκτέλεση μιας γενικότερης αξιολόγησης. Η φάση τελειώνει με τον καθορισμό του Solution Backlog, που αντιπροσωπεύει την αρχειοθέτηση των απαιτήσεων που πρέπει να εκτελεστούν στην επόμενη φάση.

- **Φάση Εκτέλεσης (Execution):**

Η φάση αυτή καλύπτει λεπτομερή ανάλυση, σχεδιασμό και ανάπτυξη της εφαρμογής σε κοινή φάση, ενώ αξίζει να σημειώσουμε πως η εκτέλεση χωρίζεται σε έναν προκαθορισμένο αριθμό Sprint Cycles . Ο στόχος σε αυτή τη φάση είναι η παραγωγή του λογισμικού με βάση των απαιτήσεων που συγκεντρώθηκαν κατά τη διάρκεια της φάσης προετοιμασίας. Αυτό εκτελείται μέσα από μια σειρά Κύκλων Σπρίντ (Sprint Cycles), με συνήθης διάρκεια 2-8 εβδομάδων. Απαιτεί:

- **Στενή συνεργασία μεταξύ των ομάδων:** Στενή και άρρηκτη συνεργασία μεταξύ της ομάδας σχεδιασμού με την ομάδα τεχνολογίας (Technology team[7]). Ο σκοπός είναι κυρίως η ελαχιστοποίηση του κινδύνου μέσω συχνών κύκλων ανατροφοδότησης και η βελτίωση της επικοινωνίας μέσω στενότερης συνεργασίας.
- **Εφαρμογή λειτουργιών κατά σειρά προτεραιότητας**: Η Agile δίνει την δυνατότητα στους Stakeholders[8] να αλλάξουν τις απαιτήσεις κατά τη διάρκεια του κύκλου της παραγωγής. Έχουν τον πλήρη έλεγχο πάνω από το πεδίο εφαρμογής, τον προϋπολογισμό και το χρονοδιάγραμμα, με συνέπεια να παίρνουν αυτό που θέλουν. Έτσι λοιπόν και η ομάδα σχεδιασμού έχει την ελαστικότητα να κάνει αλλαγές είτε στις προδιαγραφές είτε στη σειρά προτεραιότητας της εφαρμογής για να καλύψει τις ακριβείς ανάγκες των ενδιαφερομένων.
- **Ανάλυση και σχεδιασμός:** Σε αυτή τη φάση γίνεται ξεχωριστή ανάλυση και ιεράρχηση κάθε απαίτησης, όπου με βάση τη προτεραιότητα εντάσσεται σε μία βάση (Backlog). Ουσιαστικά μέσω αυτής της βάσης γίνεται και η εφαρμογή και υλοποίηση του έργου. Καθοδηγούμενοι από τα διάφορα μοντέλα αρχιτεκτονικής, γίνεται εφαρμογή μιας εξαιρετικά συνεργατικής ανάπτυξης

[7] **Technology team:** Development, Creative Development (Graphic Designing), Testing (Quality Assurance)
[8] **Stakeholders:** ενδιαφερόμενοι κύκλοι συμφερόντων

λογισμικού με γνώμονα τις δοκιμές (TDD[9]). Μερικές φορές, ιδίως σε έργα με πολύπλοκες απαιτήσεις, διαμορφώνεται ένα υπόδειγμα που περιλαμβάνει μικρά κομμάτια του έργου, για να εξασφαλιστεί ότι οι προγραμματιστές δεν χρειάζεται να περιμένουν για πληροφορίες.

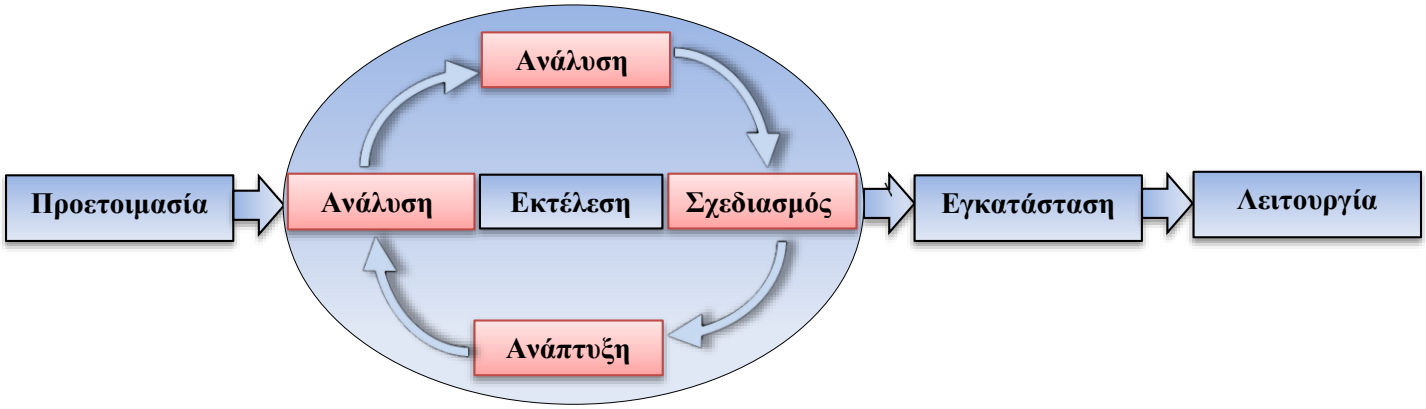
- **Διασφάλιση ποιότητας:** Στην μέθοδο Agile υπάρχει καθοδήγηση που αφορά τη κωδικοποίηση και το στυλ μοντελοποίησης, η οποία θα πρέπει να τηρείται. Επιπλέον, γίνεται αναδιοργάνωση της δομής του κώδικα της εφαρμογής και του σχήματος της βάσης δεδομένων όπως απαιτείται, για την εξασφάλιση του καλύτερου δυνατού σχεδιασμού.
- **Συνεχής παράδοση λογισμικού:** Στο τέλος κάθε κύκλου ανάπτυξης θα πρέπει να υπάρχει μέρος του συστήματος, το οποίο θα λειτουργεί. Το ιδανικό βέβαια θα είναι το μέρος του συστήματος να αναπτυχθεί, στο περιβάλλον δοκιμών (QA sandbox) για να δοκιμαστεί και η ενσωμάτωση του. Όσο πιο γρήγορη και συχνή η δοκιμή αυτή τόσο το καλύτερο.
- **Δοκιμές:** Σε αυτό το μοντέλο γίνονται πολλές δοκιμές κατά τη διάρκεια της ανάπτυξης του λογισμικού. Μέρος της κατασκευής αποτελεί η δοκιμή επιβεβαίωσης, η οποία είναι ουσιαστικά ένας συνδυασμός δοκιμών από τους προγραμματιστές σε επίπεδο σχεδιασμού και από την ομάδα σχεδιασμού του έργου σε επίπεδο απαιτήσεων. Αυτή βέβαια δεν είναι η πλήρης εικόνα της δοκιμής. Αυτό γιατί η παραγωγή λογισμικού είναι συνεχής, τουλάχιστον στο τέλος κάθε επανάληψης, το λογισμικό θα πρέπει να είναι σε θέση να περάσει σε μια ανεξάρτητη ομάδα για διεξοδική δοκιμή. Αυτές οι δοκιμές γίνονται από επαγγελματίες δοκιμαστές που ειδικεύονται στο να βρίσκουν ελαττώματα και ελλείψεις του κώδικα. Τα ελαττώματα αυτά μπορεί να αφορούν τη χρηστικότητα, την ενσωμάτωση, απαιτήσεις που χάθηκαν ή απλά δεν έχουν ακόμη εφαρμοστεί, και μερικές φορές αφορούν σε πράγματα που απλά δεν σκέφτηκαν να δοκιμάσουν οι προγραμματιστές και οι σχεδιαστές των απαιτήσεων στα προηγούμενα στάδια των δοκιμών.

➢ Φάση Εγκατάστασης (Deployment):

Βασικές δραστηριότητες στη φάση αυτή περιλαμβάνουν την εκπαίδευση χρηστών, δοκιμές αποδοχής από τους χρήστες , και την μετάβαση της εφαρμογής στη παραγωγή.

➢ Φάση Λειτουργίας (Operation):

Σε αυτή τη φάση καθορίζονται οι δραστηριότητες που απαιτούνται για το τερματισμό του έργου, παροχή υποστήριξης για τη λειτουργία της εφαρμογής στη παραγωγή ενώ παράλληλα έχουμε τη παρουσίαση της λύση και με την μεταφορά της γνώση προς τον πελάτη .



Εικόνα 3.5 Agile model

[9] **TDD:** test-driven design

Πλεονεκτήματα

- Η ικανοποίηση των πελατών από την ταχεία και συνεχή παροχή λογισμικού.
- Δίνεται έμφαση στους ανθρώπους και τις μεταξύ τους αλληλεπιδράσεις αντί σε μεθόδους και εργαλεία. Οι πελάτες, οι προγραμματιστές και οι δοκιμαστές συνεχώς αλληλεπιδρούν μεταξύ τους. Η προσωπική επικοινωνία, ειδικά πρόσωπο με πρόσωπο είναι η καλύτερη μορφή.
- Λογισμικό παραδίδεται συχνά (εβδομάδες και όχι μήνες).
- Στενή και καθημερινή συνεργασία μεταξύ των ομάδων του έργου (business, developers, etc).
- Συνεχής προσοχή στην τεχνολογική υπεροχή και καλό σχεδιασμό.
- Η τακτική προσαρμογή στις μεταβαλλόμενες συνθήκες.
- Αλλαγές στις απαιτήσεις είναι πάντα ευπρόσδεκτες.

Μειονεκτήματα

- Είναι δύσκολο να εκτιμηθεί η προσπάθεια που απαιτείται κατά την έναρξη της ανάπτυξης του κύκλου ζωής του λογισμικού, ειδικότερα στη περίπτωση ορισμένων μεγάλων παραδοτέων.
- Υπάρχει έλλειψη έμφασης σε απαραίτητο σχεδιασμό και τεκμηριώσεις.
- Το έργο μπορεί εύκολα να παρεκκλίνει από το προσδοκώμενο αποτέλεσμα, εάν ο πελάτη δεν είναι σαφής στο τελικό αποτέλεσμα που επιθυμεί.
- Μόνο οι έμπειροι προγραμματιστές είναι ικανοί να αναλάβουν τις αποφάσεις που απαιτούνται κατά τη διάρκεια της διαδικασίας ανάπτυξης. Ως εκ τούτου, δεν υπάρχει θέση για νέους προγραμματιστές, παρά μόνο σε συνδυασμό με έμπειρους πόρους.

Πότε Ενδείκνυται η χρησιμοποίηση του μοντέλου Agile

- Όταν υπάρχει απαίτηση συχνής εφαρμογής νέων απαιτήσεων, που οδηγούν σε αλλαγές του κώδικα. Η ελευθερία που δίνει η Agile σε αλλαγές είναι πολύ σημαντική . Παράλληλα, αυτές οι αλλαγές μπορούν να εφαρμοστούν με πολύ μικρό κόστος λόγω της συχνότητας των νέων κομματιών του λογισμικού που παράγονται .
- Για την εφαρμογή κάποιου νέου χαρακτηριστικού οι προγραμματιστές χρειάζεται να αφιερώσουν μόνο λίγο χρόνο.
- Σε αντίθεση με το μοντέλο καταρράκτη η Agile χρειάζεται πολύ περιορισμένο σχεδιασμό, για να ξεκινήσει το έργο. Το μοντέλο Agile υποθέτει ότι οι ανάγκες των τελικών χρηστών είναι συνεχώς μεταβαλλόμενες. Αλλαγές που έγιναν συζητούνται και τα χαρακτηριστικά που πρόσφατα πραγματοποιήθηκαν μπορούν να αφαιρεθεί με βάση τα σχόλια της αξιολόγησης. Αυτό δίνει αποτελεσματικά στον πελάτη το τελικό σύστημα που θέλει ή χρειάζεται.
- Όλες οι ομάδες (Σχεδιαστές, προγραμματιστές, κτλ) της παραγωγής λογισμικού, αισθάνονται μεγαλύτερη ελευθερία σε επιλογές, ενώ πολλές φορές και χρόνο, παρά αν το λογισμικό είχε αναπτυχτεί με διαδοχικά πιο άκαμπτο τρόπο. Η ύπαρξη επιλογών, τους δίνει τη δυνατότητα της αποφυγής σημαντικών αποφάσεων μέχρι περισσότερα ή καλύτερα δεδομένα είναι διαθέσιμα. Αυτό σημαίνει κυρίως ότι το έργο μπορεί να συνεχίσει να κινείται προς τα εμπρός χωρίς το φόβο κάποιας ξαφνικής στασιμότητας.

(versionone) (Habib, 2013)

4. Διαδικασία δοκιμών Λογισμικού (STLC[10])

Ένα από τα πιο σημαντικά μέρη του όλου κύκλου ανάπτυξης λογισμικού είναι η δοκιμή του λογισμικού, η οποία επικεντρώνεται στην βέλτιστη παραγωγή του τελικού προϊόντος. Πολλοί πιστεύουν πως οι δοκιμές του λογισμικού είναι μια στατική εργασία και θεωρείται βαρετή, αλλά σίγουρα αυτή η άποψη είναι λάθος.

Η δοκιμή είναι ένα από τα ζωτικά μέρη οποιασδήποτε ανάπτυξη του κύκλου ζωής του προϊόντος σε κάθε τομέα της βιομηχανίας. Ο έλεγχος μίας εφαρμογής λογισμικού είναι ένα από τα πιο κρίσιμα βήματα στη ζωή του προϊόντος, καθώς δημιουργεί ένα 'τελικό προϊόν' έτοιμο για την εφαρμογή του στον πραγματικό κόσμο και παρέχει τη διασφάλιση της ποιότητας του.

Κάθε προϊόν πρέπει να ελεγχθεί με ακρίβεια για την απόδοση και τη λειτουργικότητα, πριν αυτό γίνει διαθέσιμο στους χρήστες, για τον έλεγχο κατά πόσον ανταποκρίνεται στις πραγματικές ανάγκες του κόσμου. Ο έλεγχος του προϊόντος γίνεται κυρίως δοκιμάζοντας τα διάφορα σενάρια του λογισμικού με σκοπό την επαλήθευση αυτών και περνάει από πολλές φάσεις.

Στις μέρες μας ορισμένες εταιρείες ειδικεύονται σε συγκεκριμένες υπηρεσίες που παρέχουν δοκιμές. Οι μεγάλες εταιρείες λογισμικού, έχουν ένα τμήμα δοκιμών αποκλειστικά για το σκοπό αυτό ακλουθώντας τη δική τους διαδικασία δοκιμών. Αυτό καθώς η διαδικασία δοκιμών επηρεάζεται από τη Διαδικασία ανάπτυξης του λογισμικού που εφαρμόζει η κάθε εταιρεία, καθώς και τις απόψεις της διοίκησης για τη διασφάλιση της ποιότητας και δραστηριότητες δοκιμών.

4.1. Στάδια και Φάσεις δοκιμών Λογισμικού

Ανάλυση Απαιτήσεων / Επανεξέταση

- Αυτή είναι μια πολύ σημαντική φάση στην STLC. Εδώ η έμφαση δίνεται στην κατανόηση των απαιτήσεων του συστήματος από τη σκοπιά των δοκιμών.
- Σε αυτή τη φάση η ομάδα δοκιμών (Testing Team- Quality Assurance) αλληλοεπιδρά με τους Αναλυτές της επιχείρησης και των συστημάτων, τον διευθυντή ανάπτυξης, κλπ. Κάποιες φορές, για τη πλήρη κατανόηση των απαιτήσεων του συστήματος και τη διασφάλιση της ποιότητας, απαιτείται η αλληλεπίδραση με τον πελάτη.
- Κατά τη διάρκεια αυτής της φάσης, η ομάδα δοκιμών παίρνει πολλές σημαντικές αποφάσεις, όπως για τα είδη δοκιμών και τεχνικών που πρέπει να εκτελεστούν, τη δυνατότητα εφαρμογής αυτόματης δοκιμής, κλπ.

Προγραμματισμός Δοκιμών

- Σε αυτή τη φάση η ο ηγέτης της ομάδας δοκιμών προγραμματίζει την πλήρη διαδικασία των δοκιμών. Τα σημαντικά έγγραφα όπως η στρατηγική και το σχέδιο των δοκιμών ενώ παράλληλα η εκτίμησης του χρόνου και των πόρων προέρχονται από αυτή τη φάση.

[10] Software Testing Life Cycle

- Τα πάντα σχετικά με τις δοκιμές όπως η επιλογή των εργαλείων αξιολόγησης, ο προγραμματισμός των πόρων, ο καθορισμός των ρόλων και των ευθυνών του προσωπικού που συμμετέχει στη διαδικασία, ο σχεδιασμός για την απαιτούμενη κατάρτιση, κλπ. αποφασίζονται σε αυτή τη φάση.
- Αυτή η φάση είναι πολύ σημαντική καθώς κάθε μικρό λάθος μπορεί να οδηγήσει σε σημαντικά ζητήματα στο πλαίσιο του σχεδίου όσον αφορά το χρόνο, τα χρήματα, τις προσπάθειες, κλπ.

Σχεδιασμός Δοκιμών

- Δημιουργία, αναθεώρηση και ενημέρωση των διαφορετικών σεναρίων και περιπτώσεων δοκιμής γίνεται σε αυτή τη φάση. Τα σενάρια αυτά προετοιμάζονται από την ομάδα δοκιμών και πρέπει να ελεγχθούν και εγκριθούν.
- Τα δεδομένα των δοκιμών μπορούν επίσης να δημιουργηθούν σε αυτή τη φάση από την ομάδα δοκιμών εάν το περιβάλλον δοκιμών είναι στη διάθεσή τους.

Εγκατάσταση Περιβάλλοντος Δοκιμών

- Περιβάλλον Δοκιμών είναι το ίδιο το σύστημα / περιβάλλον, στο οποίο η ομάδα δοκιμών θα δοκιμάσει την εφαρμογή. Το περιβάλλον δοκιμής παρασκευάζεται με την κατανόηση της απαιτούμενης αρχιτεκτονικής του συστήματος, των απαιτήσεων του λογισμικού κλπ.
- Πολλές φορές η ομάδα δοκιμών δεν εμπλέκεται στη δημιουργία του περιβάλλοντος δοκιμής. Σε αυτά τα σενάρια, η ομάδα δοκιμών θα πρέπει να εφαρμόσει Δοκιμές καπνού (Smoke Test) για να ελέγξει την ετοιμότητα του περιβάλλοντος δοκιμής, πριν από την έναρξη της πραγματικής δοκιμής του παραδοτέου λογισμικού.

Εκτέλεση δοκιμών

- Τα διάφορα σενάρια δοκιμών που παρασκευάσθηκαν νωρίτερα εκτελούνται σε αυτή τη φάση. Οι διαφορετικές τεχνικές δοκιμών, καθώς και οι μέθοδοι εφαρμόζονται και εκτελούνται στο λογισμικό/ εφαρμογή με σκοπό τον εντοπισμό των σφαλμάτων του συστήματος.
- Τα σφάλματα που εντοπίζει η ομάδα δοκιμών, αναφέρονται στην ομάδα ανάπτυξης του λογισμικού. Η ομάδα ανάπτυξης από τη μεριά της θα επιλύσει τα σφάλματα και το σύστημα θα ξαναδοκιμαστεί από τους δοκιμαστές για να διασφαλιστεί ότι το σφάλμα έχει εξαλειφτεί.

Περάτωση Δοκιμών

- Όταν η ομάδα δοκιμής του λογισμικού είναι σίγουρη ότι όλα τα σφάλματα που αναφέρθηκαν έχουν επιλυθεί και το σύστημα είναι έτοιμο σύμφωνα με τις απαιτήσεις του πελάτη, η διαδικασία Δοκιμών του Λογισμικού(STLC) εισέρχεται στο τελευταίο στάδιο. Το στάδιο της τελικής δοκιμής.

- Σε αυτό το στάδιο, γίνεται η αξιολόγηση για τον πλήρη κύκλο δοκιμών, παρασκευάζεται η έκθεση που αποτελεί το τέλος των δοκιμών, η σωστή ανάλυση και τεκμηρίωση γίνεται για τα μεγάλα ή κρίσιμα σφάλματα, έτσι ώστε τέτοιες καταστάσεις μπορούν να αντιμετωπιστούν αποτελεσματικά και αποδοτικά σε μελλοντικά έργα.

(STLC - Software Testing Life Cycle, 2013)

4.2 Κύριος στόχος των δοκιμών λογισμικού

Το ολοκληρωμένο προϊόν λογισμικού για παράδειγμα ένα παιχνίδι στον υπολογιστή, μία νέα εφαρμογή για τα κινητά ή μια ιστοσελίδα περνά μέσα από πολλά στάδια δοκιμών, πριν κερδίσει τον τελικό χρήστη. Όλα αυτά τα επίπεδα δοκιμών έχουν ως στόχο τον καθορισμό των προβλημάτων απόδοσης, σφαλμάτων, αποκλίσεων από την αναμενόμενη λειτουργικότητα.

Το πρόγραμμα θα πρέπει να πληρεί όλες τις αναμενόμενες απαιτήσεις λειτουργίας και ο υπεύθυνος δοκιμής εφαρμογών (Software Tester), θα πρέπει να το πιστοποιήσει. Υπάρχουν μια σειρά από διαφορετικές μεθόδους δοκιμών που χρησιμοποιούνται για τον καθορισμό σφαλμάτων που απομένουν μετά την ανάπτυξη του λογισμικού, λόγω του ότι υπάρχουν διάφοροι τύποι προγραμμάτων λογισμικού και απουσία κοινής μεθοδολογίας δοκιμών.

4.3 Μεθοδολογίες δοκιμών Λογισμικού

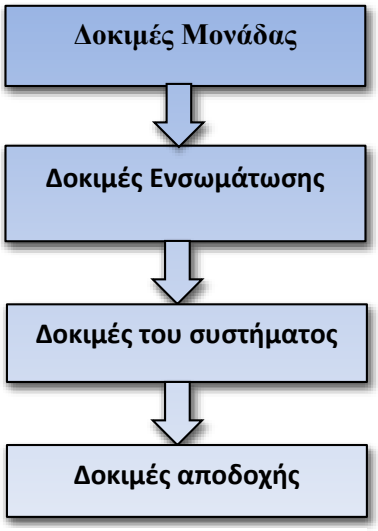
Έλεγχος Λογισμικού είναι η αξιολόγηση ενός λογισμικού/ προϊόντος που έχει αναπτυχτεί, για τον έλεγχο της ικανότητάς και της δυνατότητας του να παραδώσει τα επιδιωκόμενα αποτελέσματα. Υπάρχουν διάφορες μεθοδολογίες που χρησιμοποιούνται στον τομέα των δοκιμών του λογισμικού και τη διασφάλιση της ποιότητας.

Η δοκιμή λογισμικού αποτελεί αναπόσπαστο μέρος της ανάπτυξης του κύκλου ζωής του λογισμικού (SDLC[i]). Η δοκιμή ενός κομματιού του κώδικα αποτελεσματικά και αποδοτικά είναι εξίσου σημαντικός, με το γράψιμο. Η δοκιμή λογισμικού δεν είναι τίποτα άλλο παρά η υποβολή ενός κομματιού του κώδικα, σε ελεγχόμενη και μη ελεγχόμενη συνθήκη λειτουργίας. Όλη αυτή η προσπάθεια μας οδηγεί στη παρατήρηση του αποτελέσματος, και στη συνέχεια την εξέταση του, με βάση τις προκαθορισμένες προϋποθέσεις.

Οι υπεύθυνοι των δοκιμών του λογισμικού, παρασκευάζουν διαφορετικά σενάρια υποθέσεων(Test Cases) και στρατηγικών δοκιμών(testing strategies). Οι δοκιμές αυτές έχουν ως κοινό στόχο την εύρεση των σφαλμάτων του κώδικα, με σκοπό την εξάλειψη τους. Έμμεσα στόχο έχει την παροχή ακριβής και βέλτιστης παραγωγή. Υπάρχουν διάφοροι τύποι τεχνικών και μεθοδολογιών που εμπλέκονται σε αυτή τη δοκιμή.

Οι μέθοδοι δοκιμών που χρησιμοποιούνται συνήθως είναι οι δοκιμές μονάδας, δοκιμές ολοκλήρωσης, δοκιμών αποδοχής και τη δοκιμή του συστήματος. Ένα λογισμικό υποβάλλεται σε αυτές τις δοκιμές σε μια συγκεκριμένη σειρά.

Δοκιμές μονάδας (Unit Testning)- Το πρώτο που πρέπει να πραγματοποιηθεί είναι οι δοκιμές μονάδων.
Όπως υποδηλώνει και το όνομα, αυτή η μέθοδος δοκιμών γίνεται σε επίπεδο αντικειμένου. Μεμονωμένα στοιχεία λογισμικού ελέγχονται για τυχόν λάθη. Η ακριβής γνώση του προγράμματος είναι απαραίτητη για τη δοκιμή αυτή, καθώς κάθε μονάδα δοκιμάζεται ξεχωριστά. Έτσι, αυτή η δοκιμή γίνεται από τους προγραμματιστές. Διάφοροι κώδικες ελέγχου δημιουργήθηκαν, για τη δοκιμή της αναμενόμενης συμπεριφοράς του λογισμικού.



Εικόνα 4.1 STLC

Δοκιμές Ενσωμάτωσης (Integration Testing) - Οι επιμέρους ενότητες που έχουν ήδη υποβληθεί σε δοκιμές μονάδας έχουν ενσωματωθεί μεταξύ τους, και δοκιμάζονται για τη διασφάλιση της έλλειψης σφαλμάτων. Ένα τέτοιο είδος δοκιμών αναδεικνύει τα λάθη διασύνδεσης . Η από πάνω προς τα κάτω «Top-Down» προσέγγιση της δοκιμής ολοκλήρωσης, ακολουθεί την αρχιτεκτονική του συστήματος. Μια άλλη προσέγγιση είναι η προσέγγιση του «bottom-up», η οποία διεξάγεται από το κάτω μέρος του ελέγχου ροής.

Δοκιμές του συστήματος (System Testing) - Σε αυτή τη δοκιμή, ολόκληρο το σύστημα δοκιμάζεται για τα λάθη και σφάλματα. Η δοκιμή αυτή διεξάγεται με τη διασύνδεση όλων των στοιχείων του συστήματος(hardware & software), και στη συνέχεια τον έλεγχό τους. Αυτή η δοκιμή είναι εισηγμένη κάτω από τη μέθοδο δοκιμής Μαύρου Κουτιού (Black Box Testing), όπου ο χρήστης παρέχει εισόδους στο λογισμικό και ελέγχει τα αποτελέσματα. Στο Μαύρο Κουτί ο δοκιμαστής αγνοεί την αρχιτεκτονική του συστήματος και δεν έχει πρόσβαση στον κώδικα.

Δοκιμές αποδοχής (Acceptance Testing) - Αυτή είναι η τελευταία δοκιμή που διεξάγεται πριν από το λογισμικό παραδοθεί στον πελάτη. Διενεργείται για να διασφαλιστεί ότι το λογισμικό που έχει αναπτυχθεί πληρεί όλες τις απαιτήσεις του πελάτη. Υπάρχουν δύο τύποι:
Μία που διεξάγεται από τα μέλη της ομάδας ανάπτυξης λογισμικού(Developers & Testers), γνωστή ως εσωτερικές δοκιμές αποδοχής (Alpha testing), και την άλλη που πραγματοποιείται από τον πελάτη, γνωστή ως εξωτερική δοκιμή αποδοχής. Το στάδιο που ο πελάτης εγκρίνει τα αποτελέσματα από τον έλεγχο που διεξάγει, έχει ονομαστεί ως δοκιμή αποδοχής του πελάτη (Client Acceptance Testing). Η περίπτωση οπού ο έλεγχος γίνει από τον τελικό χρήστη του λογισμικού, ονομάζεται Δοκιμή αποδοχής Χρήστη (Beta testing).

Δοκιμές λειτουργίας (Functional Tests)

Υπάρχουν μερικές βασικοί μέθοδοι δοκιμών, που αποτελούν μέρος του συστήματος δοκιμών του λογισμικού.

Έλεγχος Μαύρου Κουτιού (Black-Box): πραγματοποιείται χωρίς καμία γνώση της εσωτερικής λειτουργίας του συστήματος. Ο Ελεγκτής του συστήματος θα παρέχει διαφορετικές εισόδους στο λογισμικό και θα ελέγξει τα παραγόμενα αποτελέσματα του. Η δοκιμή αυτή είναι επίσης γνωστή ως κλειστό κουτί δοκιμών (Closed-Box Testing) ή λειτουργικές δοκιμές(Functional Testing).

Έλεγχος Λευκού Κουτιού (White-Box): σε αντίθεση με τον έλεγχο του μαύρου κουτιού, ο έλεγχος αυτός λαμβάνει υπόψη την εσωτερική λειτουργία και τη λογική του κώδικα. Για την πραγματοποίηση αυτής της δοκιμής, ο δοκιμαστής του συστήματος θα πρέπει να έχει γνώση του κώδικα, έτσι ώστε να αντιληφθεί το ακριβές τμήμα του κώδικα που έχει λάθη. Η δοκιμή αυτή είναι επίσης γνωστή ως Ανοιχτό Κουτί (Open-Box) ή Δοκιμή Γυαλιού (Glass Testing).

Έλεγχος Γκρίζου Κουτιού (Grey-Box): Η δοκιμή όπου η γνώση τμημάτων του κώδικα είναι αναγκαία για την εκτέλεση της, ονομάζεται Έλεγχος Γκρίζου Κουτιού. Ο έλεγχος αυτός αφορά κυρίως τα έγγραφα του συστήματος και τα διαγράμματα ροής δεδομένων. Η δοκιμή διεξάγεται από τους τελικούς χρήστες ή χρήστες που λαμβάνουν τη στάση τελικών χρηστών.

Δοκιμές άσχετες με την λειτουργία (Non-Functional Tests)

Δοκιμές Ασφαλείας (Security Testing)- Η ασφάλεια μιας εφαρμογής είναι ένα από τα κύρια μελήματα του έργου. Η δοκιμή της ασφάλειας δοκιμάζει το λογισμικό για την εμπιστευτικότητα, την ακεραιότητα, αυθεντικότητα και τη διαθεσιμότητα. Μεμονωμένες δοκιμές διενεργούνται για να αποφευχθεί οποιαδήποτε μη εξουσιοδοτημένη πρόσβαση στον κώδικα του λογισμικού.

Δοκιμές κατά ακραίων καταστάσεων (Stress Testing) - είναι μια μέθοδος όπου το λογισμικό υπόκειται και δοκιμάζεται σε καταστάσεις που είναι πέρα από τις κανονικές συνθήκες εργασίας του. Μόλις το σημείο στο οποίο το πρόγραμμα παρουσιάσει ελαττώματα ή λάθη (break-point) έχει καθοριστεί, εξετάζονται τα αποτελέσματα που προέκυψαν. Η δοκιμή αυτή προσδιορίζει τη σταθερότητα ολόκληρου του συστήματος.

Δοκιμές συμβατότητας (Compatibility Testing) - Το λογισμικό έχει ελεγχθεί για συμβατότητα με ένα εξωτερικό περιβάλλον, όπως τα λειτουργικά συστήματα, πλατφόρμες, προγράμματα περιήγησης στο Web, κ.λπ. Οι Δοκιμές που δεν σχετίζονται με την λειτουργία της συμβατότητας (non-functional compatibility test) ελέγχει εάν το προϊόν έχει κατασκευαστεί για να ταιριάζει σε κάθε πλατφόρμα λογισμικού.

Δοκιμές Απόδοσης (Efficiency Testing)- Όπως υποδηλώνει το όνομα, αυτή η τεχνική δοκιμή ελέγχει την ποσότητα του κώδικα ή τους πόρους που χρησιμοποιούνται από το λογισμικό ενώ εκτελεί μία και μοναδική λειτουργία. Έχει ελεγχθεί ως προς τον αριθμό των διαφόρων σεναρίων δοκιμών (Test Cases), που εκτελούνται σε ένα δεδομένο χρονικό διάστημα.

Δοκιμές ευχρηστίας (Usability Testing) - Αυτή η δοκιμή εξετάζει την πτυχή της ευχρηστίας του λογισμικού. Η ευκολία με την οποία ένας χρήστης μπορεί να έχει πρόσβαση στο προϊόν σχηματίζει το κύριο σημείο hw dokimæhw. Οι δοκιμές ευχρηστίας εξετάζουν πέντε πτυχές των δοκιμών, -την δυνατότητα εκμάθησης, την αποτελεσματικότητα, την ικανοποίηση, τις δυνατότητες της μνήμης, και τα λάθη.

(Kekare, 2009)

4.4 Μοντέλα Δοκιμών Λογισμικού

Μοντέλο Καταρράκτη (waterfall model)- Το μοντέλο καταρράκτη υιοθετεί μια προσέγγιση «από πάνω προς τα κάτω», ανεξάρτητα από το αν χρησιμοποιείται για την ανάπτυξη λογισμικού ή δοκιμή. Τα βασικά στάδια που εμπλέκονται σε αυτή τη μεθοδολογία δοκιμών του λογισμικού είναι οι εξής:

- Ανάλυση Απαιτήσεων
- Σχεδιασμός υποθέσεων δοκιμής
- Εφαρμογή και εκτέλεση των υποθέσεων δοκιμής
- Έλεγχος, Αποσφαλμάτωση, και η επικύρωση του κώδικα
- Ανάπτυξη και συντήρηση

Στη μεθοδολογία αυτή, προχωράμε στο επόμενο βήμα μόνο αφού έχει ολοκληρωθεί το παρόν στάδιο. Το μοντέλο ακολουθεί μια μη επαναληπτική προσέγγιση. Το κύριο πλεονέκτημα αυτής της μεθόδου είναι απλοϊκή, συστηματική και ορθόδοξη προσέγγιση της. Ωστόσο, έχει πολλές ελλείψεις, δεδομένου ότι τα λάθη και οι ελλείψεις στον κώδικα δεν θα εντοπιστούν μέχρι να επιτευχθεί το στάδιο της δοκιμής. Αυτό μπορεί συχνά να οδηγήσει σε σπατάλη χρόνου, χρήματος, και άλλων πολύτιμων πόρων.

Agile Μοντέλο - Η μεθοδολογία αυτή δεν ακολουθεί ούτε αμιγώς διαδοχική αλλά ούτε αμιγώς επαναληπτική προσέγγιση. Είναι μια εκλεκτική μίξη και των δύο προσεγγίσεων. Η γρήγορη και σταδιακή ανάπτυξη είναι μία από τις βασικές αρχές αυτής της μεθοδολογίας. Η έμφαση δίνεται στην γρήγορη ανάκτηση, πρακτικά και ορατά αποτελέσματα, αντί απλώς να ακολουθεί τις θεωρητικές διαδικασίες. Η συνεχής αλληλεπίδραση των πελατών και η συμμετοχή τους, αποτελεί αναπόσπαστο μέρος της όλης διαδικασίας ανάπτυξης.

Μοντέλο Σπιράλ (Spiral Model)- Όπως υποδηλώνει και το όνομα, το μοντέλο σπιράλ ακολουθεί μια προσέγγιση στην οποία υπάρχει ένας αριθμός κύκλων, με όλα τα διαδοχικά στάδια του μοντέλου καταρράκτη. Μετά την ολοκλήρωση του αρχικού κύκλου, γίνεται μια διεξοδική ανάλυση και αναθεώρηση του αποτελέσματος. Αν δεν είναι σύμφωνα με τις συγκεκριμένες απαιτήσεις ή αναμενόμενα πρότυπα, ένας δεύτερος κύκλος ακολουθεί, και ούτω καθεξής. Η μεθοδολογία ακολουθεί μια επαναληπτική προσέγγιση, και είναι γενικά κατάλληλη για τα μεγάλα πολύπλοκα έργα με διαρκώς μεταβαλλόμενες απαιτήσεις.

5. Δοκιμές αυτοματισμού (Automated Testing)

Κάθε ομάδα ανάπτυξης λογισμικού ελέγχει το προϊόν της, το οποίο τις περισσότερες φορές, ακόμη και αν έχει παραδοθεί έχει κάποια μικρά ή μεγάλα ελαττώματα. Οι δοκιμές προσπαθούν να εντοπίσουν τα λάθη αυτά πριν από την παράδοση του προϊόντος. Παρόλα αυτά τα σφάλματα καταφέρνουν να παρεισφρήσουν και συχνά επανεμφανίζονται ακόμη και ύστερα από εστιασμένες χειροκίνητες διαδικασίες δοκιμής. Η αυτοματοποιημένη δοκιμή του λογισμικού είναι ο καλύτερος τρόπος για την αύξηση της αποτελεσματικότητας, της αποδοτικότητας και κάλυψης των δοκιμών του λογισμικού.

Χειροκίνητη δοκιμή του λογισμικού γίνεται από έναν άνθρωπο μπροστά από έναν υπολογιστή, ο οποίος προσεκτικά ελέγχει τη διεπαφή χρήστη των εφαρμογών, δοκιμάζει διάφορες χρήσεις και συνδυασμούς των εισροών, συγκρίνει τα αποτελέσματα με την αναμενόμενη συμπεριφορά και τελικά καταγράφει τις παρατηρήσεις του. Οι χειροκίνητες δοκιμές επαναλαμβάνονται συχνά κατά τη διάρκεια των κύκλων ανάπτυξης για τις αλλαγές του πηγαίου κώδικα και άλλες καταστάσεις όπως, πολλαπλά λειτουργικά περιβάλλοντα και διαμορφώσεις υλικού.

Ένα αυτοματοποιημένο εργαλείο δοκιμών είναι σε θέση να αναπαράγει προ-καταγεγραμμένες και προκαθορισμένες ενέργειες, να συγκρίνει τα αποτελέσματα με την αναμενόμενη συμπεριφορά και να αναφέρει την επιτυχία ή την αποτυχία των εν λόγω εγχειριδίων δοκιμών.

Από τη στιγμή που οι αυτοματοποιημένες δοκιμές δημιουργούνται, μπορούν εύκολα να επαναληφθούν και να επεκταθούν για την εκτέλεση εργασιών που είναι χρονικά αδύνατον να γίνουν χειροκίνητα. Εξαιτίας αυτού, πολλά στελέχη εταιριών έχουν διαπιστώσει ότι οι αυτοματοποιημένες δοκιμές λογισμικού, είναι ένα βασικό συστατικό των επιτυχημένων έργων ανάπτυξης λογισμικού και όχι μόνο.

5.1 Πως προέκυψε η ανάγκη της αυτόματης δοκιμής

Στο σύγχρονο επιχειρηματικό περιβάλλον, οι ομάδες που εργάζονται επάνω στην ανάπτυξη λογισμικού, αναμένονται να κάνουν περισσότερα παραδίνοντας συστήματα ανώτερης ποιότητας σε μικρότερο χρονικό διάστημα και με λιγότερους πόρους. Όταν στις εταιρίες μειώνεται ο προϋπολογισμός, οι δοκιμές λογισμικού είναι συχνά μια από τις πρώτες διαδικασίες οι οποίες ελαττώνονται.
Έτσι λοιπόν, Η ομάδα δοκιμών λογισμικού είναι συνήθως από τις πρώτες που βιώνουν τις διάφορες περικοπές.

Τα πληροφοριακά συστήματα τα οποία δεν επιλύουν τα πραγματικά προβλήματα της επιχείρησης ή απλά δεν αποδίδουν τα αναμενόμενα, επιβάλλουν ένα παρόμοιο οικονομικό τίμημα για το κόστος των επιχειρήσεων και τα αποτελέσματα. Περισσότερα από τα μισά έργα λογισμικού αποτυγχάνουν να επιτύχουν τους στόχους τους ή να υποστούν σημαντικές αποκλίσεις στο χρονοδιάγραμμα είτε στον προϋπολογισμό, επειδή κάποια ελαττώματα ανακαλύπτονται πολύ αργά. Όλοι αυτοί οι παράγοντες όταν συνδυαστούν έχουν ως αποτέλεσμα σε ένα υψηλό ποσοστό διαρροής ελαττωμάτων στη παραγωγή.
Αποτέλεσμα των ελαττωμάτων αυτών είναι η μείωση ικανοποίησης του πελάτη, ενώ παράλληλα είναι και επιβλαβής στην απόδοση της επένδυσης .

Οι κινήσεις που θα πρέπει να κάνει η επιχείρηση έτσι ώστε να διασφαλίσει την επένδυση της είναι:

- Αποκλειστική εστίαση - Εύρεση λύσης για τα προβλήματα των δοκιμών.
- Εύρεση μιας μακροπρόθεσμης και οικονομικά αποδοτικής λύσης.
- Ολοκληρωμένη κάλυψη έναντι των απαιτήσεων.
- Ακολουθία κοινού προτύπου ανάμεσα στις ομάδες ανάπτυξης λογισμικού

Όλα τα προαναφερθέντα στοιχεία μπορούν να αντιμετωπιστούν αποδοτικότερα και πιο εστιασμένα με την αυτοματοποίηση κομματιών της δοκιμής.

(Sethu, 2012)

5.2 Πλεονεκτήματα Αυτομάτων δοκιμών

Αύξηση της κάλυψης των δοκιμών

Η αυτοματοποιημένη δοκιμή του λογισμικού, αυξάνει το βάθος και την έκταση των δοκιμών ενώ έμμεσα συμβάλει στη βελτίωση της ποιότητας του λογισμικού. Χρονοβόρες δοκιμές που συχνά αποφεύγονται κατά τη διάρκεια της χειροκίνητης δοκιμής, μπορούν να λειτουργούν χωρίς επίβλεψη.

Αυτοματοποιημένη δοκιμή του λογισμικού μπορεί να κοιτάξει μέσα από μια εφαρμογή και να δει τα περιεχόμενα της μνήμης, πίνακες δεδομένων, τα περιεχόμενα του αρχείου, καθώς και τις εσωτερικές καταστάσεις του προγράμματος για να διαπιστώσει αν το προϊόν συμπεριφέρεται όπως ήταν αναμενόμενο. Παράλληλα οι αυτοματοποιημένες δοκιμές του λογισμικού, μπορούν εύκολα να εκτελέσουν πολλές διαφορετικές και πολύπλοκες περιπτώσεις δοκιμών κατά τη διάρκεια των σπριντ δοκιμών, παρέχοντας κάλυψη που είναι πρακτικά αδύνατο με τις χειροκίνητες δοκιμές.
Οι δοκιμαστές κερδίζοντας χρόνο από τις επαναλαμβανόμενες δοκιμές, έχουν τη δυνατότητα να δοκιμάσουν άλλα πιο πολύπλοκα μέρη του συστήματος, είτε και να δημιουργήσουν νέες αυτοματοποιημένες δοκιμές του λογισμικού οι οποίες θα ασχολούνται με διαφορετικές λειτουργίες.

Αυτοματοποιημένο Λογισμικό δοκιμών εξοικονομεί χρόνο και χρήμα

Ο Χρόνος είναι το βασικότερο πλεονέκτημα του αυτοματισμού κατά πολλούς, ειδικά όταν χρησιμοποιείται για δοκιμές παλινδρόμησης (Regression Testing). Όπως αναφέρθηκε προηγουμένως οι δοκιμές παλινδρόμησης είναι η δοκιμή μιας εφαρμογής που επαναλαμβάνεται σε κάθε κύκλο δοκιμών.
Ο στόχος των δοκιμών παλινδρόμησης είναι να πιστοποιήσει πως η εφαρμογή έχει ακόμα την αναμενόμενη λειτουργία. Για να αποδειχτεί αυτό, ο χρήστης θα πρέπει να ελέγξει όλα τα σενάρια (test cases) που έχουν σχέση με την αλλαγή που έκανε. Στην περίπτωση που, λόγο της έλλειψης χρόνου, ο χρήστης δεν καταφέρει να ελέγξει όλα τα σενάρια, υπάρχει μεγάλο ρίσκο η εφαρμογή να έχει προβλήματα και κενά.
Αυτά τα προβλήματα μπορούν να αποκαλυφθούν από τον χρήστη, με το να θέσει σε λειτουργία το εργαλείο και να ελέγξει αυτόματα όλα τα διαφορετικά σενάρια. Αυτό θα δώσει στον χρήστη το χρόνο που χρειάζεται ώστε να εκτελέσει δοκιμές σε μέρη τα οποία δεν μπορούν να αυτοματοποιηθούν και άλλα καθήκοντα. Μόλις δημιουργηθούν οι αυτοματοποιημένες δοκιμές, μπορούν να τρέξουν επαναλαμβανόμενα χωρίς κανένα επιπλέον κόστος, ενώ παράλληλα θα είναι πολύ πιο γρήγορες από ότι οι χειροκίνητες δοκιμές.

Το εργαλείο αυτοματισμού, είναι ωφέλιμο όταν ο μειώνεται αριθμός των πόρων που απαιτούνται για τις χρονοβόρες δοκιμές παλινδρόμησης. Φυσικά, με την αυτοματοποίηση των δοκιμών δεν σημαίνει ότι εξαλείφουμε το ρόλο των δοκιμαστών. Πάντα θα υπάρχει μια θέση για έναν άνθρωπο δοκιμαστή μέσα σε μια ομάδα έργου, καθώς δεν είναι κάθε δοκιμή σχετική με τη λειτουργία. Όπως επίσης, δεν είναι κάθε πρόγραμμα κατάλληλο για αυτοματοποιημένες δοκιμές.

Βελτιώνει την ακρίβεια της δοκιμής

Το εργαλείο μπορεί να ελέγξει επανειλημμένα ακριβώς τα ίδια σενάρια, ώστε να εξαλείψει το ρίσκο του ανθρώπινου λάθους. Είναι γεγονός, πως ακόμα και οι πιο ευσυνείδητοι και έμπειροι δοκιμαστές, κάποια στιγμή θα κάνουν λάθος κατά τη διάρκεια του χειροκίνητου έλεγχου. Αυτό γιατί, πολλές φορές οι δοκιμαστές ξεχνούν τις ακριβείς κινήσεις τους παραλείποντας βήματα από τα σενάρια που θα πρέπει να ελέγξουν. Αυτό μπορεί να έχει ως αποτέλεσμα την παρουσία προβλημάτων που δεν έχουν προσδιοριστεί, είτε την αναφορά μη έγκυρων λαθών. Αυτοματοποιημένα τεστ εκτελούν τα ίδια ακριβώς βήματα κάθε φορά που εκτελούνται, ενώ συνήθως καταγράφουν τα αποτελέσματα, προς την αποφυγή λαθών.

Συμβάλει στη συντήρηση της κασετίνας δοκιμών

Όταν οι δοκιμές είναι αυτοματοποιημένες και τρέχουν μετά από κάθε κατασκευή, αυτές που έχουν ξεπεράσει την ημερομηνία λήξης τους θα αποτύχουν. Ως εκ τούτου, θα αναγκάσουν τον δοκιμαστή να επιστρέψει και να ενημερώσει το σενάριο δοκιμής. Η διαδικασία αυτή διασφαλίζει πως τα σενάρια δοκιμών συμβαδίζουν με τα σημερινά δεδομένα και η ποιότητα του λογισμικού διατηρείται.

Η αυτοματοποιημένη δοκιμή βοηθάει δοκιμαστές αλλά και προγραμματιστές

Τα κοινόχρηστα εργαλεία αυτοματισμού δοκιμών μπορούν να χρησιμοποιηθούν από τους προγραμματιστές για να πιάσουν γρήγορα τα προβλήματα πριν από την αποστολή στην ομάδα δοκιμών. Οι δοκιμές μπορούν να εκτελεστούν αυτόματα κάθε φορά που οι αλλάζει ο πηγαίος κώδικας και ενημερώνει σχετικά την ομάδα ή τον υπεύθυνο του έργου, εάν αποτύχουν. Χαρακτηριστικά όπως αυτά εξοικονομούμε χρόνο και να αυξήσει την εμπιστοσύνη τους.

Το ηθικό της ομάδας βελτιώνεται

Αυτό είναι δύσκολο να υπολογιστεί, η αυτοματοποιημένη δοκιμή του λογισμικού μπορεί να βελτιώσει το ηθικό της ομάδας. Αυτοματοποιώντας επαναλαμβανόμενες εργασίες δίνει το χρόνο στην ομάδα να περάσει σε πιο προκλητικά και αποδοτικά έργα. Τα μέλη της ομάδας με αυτό τον τρόπο, βελτιώνουν την οργάνωση των δοκιμών τους και μακροπρόθεσμα τις δεξιότητές τους αφού ασχολούνται με πιο κρίσιμες δοκιμές συστημάτων.

5.3 Μειονεκτήματα αυτομάτου έλεγχου.

Αν και η δοκιμή αυτοματοποίησης έχει πολλά πλεονεκτήματα, έχει παράλληλα και τα μειονεκτήματα της. Μερικά από αυτά είναι:

• Απαιτείται εμπειρία και άτομα με δεξιότητες για την ανάπτυξη των σεναρίων δοκιμών αυτοματισμού.

• Το Debugging στα σενάρια δοκιμών είναι σημαντικό θέμα. Εάν οποιοδήποτε σφάλμα είναι παρών στο σενάριο δοκιμής και δεν εντοπιστεί, μπορεί να οδηγήσει σε δαπανηρές συνέπειες για την επιχείρηση.

• Η συντήρηση των δοκιμών είναι δαπανηρή, ειδικά στις περιπτώσεις μεθόδων αναπαραγωγής. Ακόμα κι αν συμβεί μια μικρή αλλαγή στο GUI, το σενάριο δοκιμής πρέπει να ανακαταγραφεί ή να αντικατασταθεί.

Μερικά από τα παραπάνω μειονεκτήματα συχνά ζημιώνουν το όφελος που δημιουργείται από τα αυτοματοποιημένα σενάρια. Αν και η δοκιμή αυτοματισμού έχει πλεονεκτήματα και μειονεκτήματα, εφαρμόζεται ευρέως σε όλο τον κόσμο.

5.4 Εργαλεία διαθέσιμα στην αγορά

Αν ρίξει κανείς μια ματιά στο διαδίκτυο θα καταλάβει πως ο αυτόματος έλεγχος είναι πολύ ανεπτυγμένος και όλο και περισσότερες εταιρίες ειδικεύονται σε αυτό. Στις μέρες μας, κυκλοφορούν στην αγορά πολλά και διάφορα εργαλεία αυτοματισμού, που υποστηρίζουν τις περισσότερες μεθοδολογίες σε για διάφορες εφαρμογές και περιβάλλοντα.

Το QTP και το Selenium είναι οι δύο πιο δημοφιλείς επιλογές για δοκιμές λειτουργίας που διατίθενται σήμερα. Το QTP είναι το καλύτερο εργαλείο δοκιμών λειτουργίας και υποστηρίζεται από τις περισσότερες γλώσσες κωδικοποίησης και πλατφόρμες. Το Selenium είναι το καλύτερο open source λειτουργικό εργαλείο για δοκιμές στο web. Η σύγκριση μεταξύ των δύο είναι τόσο απλή όσο η αγορά ενός επώνυμου αυτοκινήτου και η συναρμολόγηση ενός αυτοκινήτου σύμφωνα με τις προσωπικές προτιμήσεις του καθενός. Το επώνυμο αυτοκίνητο έχει κόστος και υπηρεσίες που επισυνάπτονται, ενώ το συναρμολογούμενο αυτοκίνητο, θα πρέπει πάντα να φροντίζεται από το χρήστη. Βέβαια, υπάρχει και η υβριδική λύση, ο χρήστης μπορεί να πάει σε ένα εργαστήριο και να αγοράσει ένα συναρμολογούμενο αυτοκίνητο, πληρώνοντας για τις υπηρεσίες.

Η ίδια ιδέα είναι διαθέσιμη στον τομέα της πληροφορικής, οπού οι χρήστες υποστηρίζονται από φορείς παροχής υπηρεσιών που υποστηρίζουν όλες τους τις απαιτήσεις. Συνήθως οι φορείς αυτοί αναπτύσσουν πλαίσια αυτοματοποίησης, χρησιμοποιώντας διαφορετικά κομμάτια ανοικτού κώδικα. Οι προγραμματιστές αυτών των παρόχων, ενοποιούν μεταξύ τους είτε ενσωματώνουν τα κομμάτια αυτά στον έτοιμο κώδικα και παραδίνουν το τελικό προϊόν ή υπηρεσία στους πελάτες. Η έννοια αυτή είναι η νέα τάση στις ημέρες μας και πολλές εταιρείες έχουν αλλάξει σε πλαίσια open source κώδικα, που χτίστηκε από τους παρόχους, μειώνοντας έτσι το συνολικό κόστος του προϊόντος τους.

Παρακάτω ακολουθεί ένας πίνακας σύγκρισης και παράθεσης των χαρακτηριστικών, των δύο προαναφερθέντων εργαλείων, QTP και Selenium:

Feature	QTP(UFT)	Selenium
Language Support	VB Script	Java, C#, Ruby, Python, Perl PHP , Javascript
Windows (Non-browser) based Application support	Yes	No
Browser support	Google Chrome (uptill ver 23) Internet Explorer , Firefox (ver 21)	Google Chrome , Internet Explorer , Firefox , Opera , HtmlUnit
Environment Support	Only Windows	Windows , Linux , Solaris OS X , Others (If brower & JVM or Javascript support exists)
Mobile (Phones & Tablets) support	Different commercial product i.e. HP UFT Mobile (formerly known as MobileCloud for QTP)	Android , iPhone & iPad , Blackberry , Headless WebKit
Framework	Easily integrated with HP Quality Center or HP ALM (separate commercial products)	Selenium + Eclipse + Maven / ANT + Jenkins / Hudson & its plugins / Cruise Control + TestNG + SVN
Continuous Integration	Possible through Quality Center / ALM or Jenkins	Possible through Jenkins / Hudson / Cruise Control
Object Recognition / Storage	Inbuilt Object Repository (storing Element Id, multiple attributes) along with weightage that gives flexibility on deviation acceptance in control recognition	UI Maps and different object location strategy such as -XPath Element ID or attribute DOM
Image based Tests	Easily possible	Possible but not easy
Reports	Quality Center has in-built awesome dashboards	Integration with Jenkins can give good reporting & dashboard capabilities
Software Cost	License & Annual maintenance fees	Zero
Coding Experience of Engineer	Not Much	Should be very good along with technical capabilities of integrating different pieces of framework
Script Creation Time	Less	High
Hardware resource (CPU + RAM) consumption during script execution	High	Low

Εικόνα 5.1 QTP vs Selenium

Παράλληλα με τα προαναφερθέντα εργαλεία GTP και Selenium, διαθέσιμα στην αγορά υπάρχουν και διάφορα άλλα. Η ταξινόμηση τους γίνεται με βάση διαφορετικές προϋποθέσεις που ένας χρήστης μπορεί να απαιτήσει για τον έλεγχο εφαρμογών του web, λειτουργιών GUI (Graphical user interface) και τη συμβατότητα των προγραμμάτων περιήγησης.

Ο πίνακας που ακολουθεί, περιέχει τα διαθέσιμα εργαλεία αυτοματισμού μέχρι και το 2011.
Στις ημέρες μας υπάρχουν πολλές αλλαγές και νέες εισαγωγές, παρόλα αυτά, τα πιο σημαντικά εργαλεία παραμένουν αυτά του πίνακα.

Πίνακας 1Διαθέσιμα εργαλεία αυτοματισμού

Web Testing Tools	Web browser based (model)	Scriptable	Scripting Language	Recorder	Multiple domain	Frames	HTTPS
iMacros	Yes (Firefox)	Yes	iMacro Script	Yes	Yes	Yes	Yes
QF-Test	Yes (IE, Firefox)	Yes	visual scripting, Jython, Groovy	Yes	Yes	Yes	Yes
Ranorex Studio	Yes (Chrome, Firefox, Safari, IE)	Yes	C#, VB.NET	Yes	Yes	Yes	Yes
Sahi	Yes (IE/Firefox)	Yes	Sahi Script	Yes	Yes	Yes	Yes
Selenium	Yes	Yes	Ruby, Java, PHP, Perl, Python, C#, Groovy	only Firefox	Yes		
SOAtest	Yes	Yes	Python, JavaScript, Java	Yes	Yes	Yes	Yes
TestComplete	Yes (IE, Firefox, Chrome)	Yes	VBScript, JScript, C++Script, C#Script, DelphiScript	Yes	Yes	Yes	Yes
TOSCA Testsuite	Yes (IE, Firefox)	Yes	Java, C#, VB6	Yes	Yes	Yes	Yes
WatiN	Yes (IE)	Yes	C#, ability to run JavaScript from C# calls	IE/FF		Yes	Yes
Watir	Yes (IE, Firefox)	Yes	Ruby	Yes	Yes	Yes	Yes

(http://opensourcetesting.org/) (2011, 2011) (Ford, 2011)
(softwaretestinghelp, softwaretestinghelp, 2013)

Εικόνα 6.1 Logo της εταιρείας

6. Περιγραφή του προϊόντος

Το εργαλείο αυτομάτων δοκιμών θα εφαρμοστεί πάνω στην ιστοσελίδα/ ηλεκτρονικό κατάστημα της Bestseller. Η εταιρεία αυτή, είναι μια οικογενειακή επιχείρηση που ιδρύθηκε το 1975 στη Δανία. Αρχικά στόχος της ήταν για τη μόδα των γυναικών, ενώ στη συνέχεια εισήγαγε ανδρικά, εφηβικά και παιδικά ενδύματα. Αυτή τη στιγμή η Bestseller διαθέτει καταστήματα στις περισσότερες ευρωπαϊκές χώρες, τη Μέση Ανατολή, την Ινδία, την Κίνα και τον Καναδά (πάνω από 45 χώρες συνολικά), ενώ απασχολεί περισσότερους από 45.000 ανθρώπους. Τα προϊόντα της εταιρίας είναι διαθέσιμα στο διαδίκτυο, σε επώνυμες αλυσίδες καταστημάτων και πολυκαταστήματα. Από τη περιγραφή αυτή, εύκολα κανείς συμπεραίνει πως η εταιρία είναι τύπου B2C (Business-to-Consumer).

6.1. Οι μάρκες του ηλεκτρονικού καταστήματος

Τα πρώτα βήματα της εταιρίας στο χώρο του ηλεκτρονικού εμπορίου (E-Commerce) έγιναν το 2010 με 10 μάρκες κάτω από την ίδια στέγη. Στη συνέχεια (2013) δύο ακόμα ενσωματώθηκαν στην ιστοσελίδα. Παρακάτω βλέπουμε τις παρούσες αυτή τη στιγμή μάρκες :

Εικόνα 6.2 Logo των φιλοξενούμενων εταιρειών

6.2 Υπηρεσίες που προσφέρει η ιστοσελίδα

Η εκκίνηση του εκάστοτε ηλεκτρονικού καταστήματος δεν σηματοδοτεί αυτομάτως την δυνατότητα του να προσφέρει όλες τις υπηρεσίες της καθολικά. Είναι προφανές πως οι ιστοσελίδες ηλεκτρονικού εμπορίου πέρα από το κεντρικό προϊόν που στη περίπτωση μας είναι τα ρούχα, πρέπει να παρέχουν και κάποιες υπηρεσίες με στόχο την εξυπηρέτηση και ικανοποίηση του πελάτη. Σε κάποιες από αυτές τις υπηρεσίες, όπως για παράδειγμα οι πληρωμές και οι μεταφορές του προϊόντος, η εταιρία στηρίζει την ύπαρξη της.

Μεταφορές

Γίνεται, επομένως, εύκολα αντιληπτό πώς κάθε ιστοσελίδα ηλεκτρονικού εμπορίου πριν βγει στη παραγωγή πρέπει να διευθετήσει και παρέλθει σε συμφωνία με τις διάφορες εταιρίες μεταφορών. Αυτή η συμφωνία διαφέρει και ποικίλει ανά τις χώρες που δραστηριοποιείται, καθώς πλέον υπάρχουν πολλοί και διαφορετικοί μέθοδοι μεταφοράς.

Στη περίπτωση μας, οι χώρες που ήταν διαθέσιμες για μεταφορές στα πρώτα στάδια του E-Commerce της Bestseller ήταν 6, με βάση τη κεντρική και βόρεια Ευρώπη (DE, DK ,NO, SE, NL & UK). Η εταιρία πλέον είναι διαθέσιμη στις περισσότερες χώρες της Ευρώπης, ενώ η δυνατότητα αποστολής ρούχων έχει σχεδόν υπερδιπλασιαστεί. Η εταιρία να έχει πλέον λάβει θέση στην αγορά των παρακάτω χωρών:

Εικόνα 6,3 Χώρες Συνεργασίας

Για τη μεταφορά των προϊόντων της στις παραπάνω ευρωπαϊκές χώρες (εκτός Αμερικής), η Bestseller συνεργάζεται με διάφορες εταιρίες ανάλογα με τη χώρα. Οι συνεργαζόμενες εταιρίες είναι οι παρακάτω:

- **DENMARK**: GLS Parcel Shop, Post Denmark, GLS Parcel Shop
- **SWEDEN**: BRING, DHL Post Sweden
- **GERMANY**: HERMES, UPS
- **AUSTRIA**: HERMES, Austrian post
- **SWITZERLAND**: Heine, Swiss post
- **NORWAY**, **FINLAND**: BRING
- **NETHERLANDS**, **BELGIUM**, **FRANCE**, **SPAIN**: KIALA, UPS
- **IRELAND**, **ITALY**: UPS
- **UNITED KINGDOM**: HLG, HLG/UPS

Εάν βασιστούμε στις προβλέψεις, μελλοντικά θα υπάρχει η δυνατότητα της πώληση προϊόντων σε ακόμα περισσότερες χώρες, ενώ παράλληλα θα αυξηθεί και ο αριθμός με τις συνεργαζόμενες εταιρίες μεταφορών.

Μέθοδοι πληρωμής

Ένα σύστημα πληρωμής ηλεκτρονικού εμπορίου, διευκολύνει την αποδοχή των πληρωμών για τις ηλεκτρονικές συναλλαγές. Τα συστήματα πληρωμών ηλεκτρονικού εμπορίου έχουν γίνει ολοένα και πιο δημοφιλή λόγω της ευρείας χρήσης του διαδικτύου, με βάση το εμπορικό και τραπεζικό τομέα. Με τη πάροδο των χρόνων, οι πιστωτικές κάρτες έχουν γίνει μια από τις πιο κοινές μορφές πληρωμής για τις συναλλαγές ηλεκτρονικού εμπορίου. Βέβαια, τελευταία έχουν κάνει αισθητή τη παρουσία τους και άλλες μορφές ηλεκτρονικών πληρωμών με σκοπό την αμεσότητα, την ασφάλεια και την διαδραστικότητα με το χρήστη.

Κάθε εταιρία δίνει βάση στο να υποστηρίζει και να παρέχει στους πελάτες της όσο το δυνατόν περισσότερες από τις μεθόδους πληρωμής. Με αυτό το τρόπο κερδίζει την εμπιστοσύνη των πελατών και παράλληλα αυξάνει την ικανοποίηση τους. Εντούτοις, η Bestseller παρέχει τις παρακάτω μεθόδους πληρωμής ανάλογα με τη χώρα του ηλεκτρονικού καταστήματος:

	DK	NO	FI	SE	DE	NL	FR	IR	UK	BE	AT	IT	ES	CH
Credit & Debit Cards														
VISA & Master Card (CC)														
Mobile / Tablet orders (CC)														
Maestro (DC)														
CarteBleue														
PayPal														
Credit Card or Instant Transfer														
Mobile / Tablet orders (PayPal)														
RTBT (RealTimeBankTransfer)														
I-Deal														
GiroPay														
Nordea														
KLARNA														
Invoicing (no credit card)														
Klarna Account														

Εικόνα 6.2 Επιτρεπόμενοι τρόποι αποπληρωμής ***(Με πράσινο χρώμα είναι μαρκαρισμένες οι μέθοδοι πληρωμής που υποστηρίζονται*** **(Bestseller).)**

Λοιπές υπηρεσίες

Παράλληλα με τις μεταφορές και τις πληρωμές, υπάρχουν και άλλες υπηρεσίες αρκετά σημαντικές στις οποίες η επιχείρηση στηρίζει ένα μέρος των πωλήσεων. Θα αποτελούσε σοβαρή παράλειψη να μην αναφέρουμε την Εξυπηρέτηση πελατών, την λέσχη πελατών (customer Club), την αυτοματοποιημένη συλλογή δεδομένων (όταν χρησιμοποιείται προς όφελος του πελάτη- Προτεινόμενα προϊόντα), το μάρκετινγκ ηλεκτρονικού ταχυδρομείου κτλ.

Αυτά βέβαια δεν αποτελούν αντικείμενο ενδιαφέροντος της συγκεκριμένης εργασίας και γι αυτό δεν θα αναφερθούν εκτενέστερα.

6.3 Περιγραφή της παραγωγής του ηλεκτρονικού καταστήματος

Η ομάδα του ηλεκτρονικού καταστήματος της Bestseller εργάζεται σε 3 ευρείς τομείς:

- **Έργο παραγωγής λογισμικού:** Αυτός ο τομέας συμπεριλαμβάνει την υποστήριξη αλλά και την ανάπτυξη του λογισμικού, που στηρίζει τις διάφορες μάρκες της ιστοσελίδας στην ανάπτυξη και τη βελτίωση της ηλεκτρονικής επιχείρησης.
- **Η θέση της επιχείρησης στις αγορές:** Είναι μια αναπτυσσόμενη λειτουργία που είναι υπεύθυνη για την εξέλιξη της θέσης της επιχείρησης στις αγορές (Marketplace development) και την μελλοντική επέκταση (future expansion)
- **Λέσχη Πελατών**: Η λέσχη πελατών είναι μια υπηρεσία που υποστηρίζουν ορισμένες εταιρίες, με σκοπό την ενημέρωση των πελατών μέσω E-mail και SMS για καμπάνιες και προσφορές.

Όλα τα παραπάνω είναι σημαντικά για την επιχείρηση, σίγουρα όμως το πιο βασικό είναι η παραγωγή λογισμικού. Εκεί λοιπόν το κείμενο θα εστιάσει στις επόμενες παραγράφους και θα επέλθει μία πιο εκτενή αναφορά στη παραγωγή αυτή.

Ο κύκλος παραγωγής λογισμικού

Το τμήμα παραγωγής λογισμικού της Bestseller, απαρτίζεται από μια έμπειρη ομάδα ανάπτυξης, με έντονη την αίσθηση της υπευθυνότητας, και εργασίας σε ένα γρήγορο ρυθμό. Η παραγωγή του λογισμικού ακολουθεί το ευέλικτο περιβάλλον ανάπτυξης (Agile).

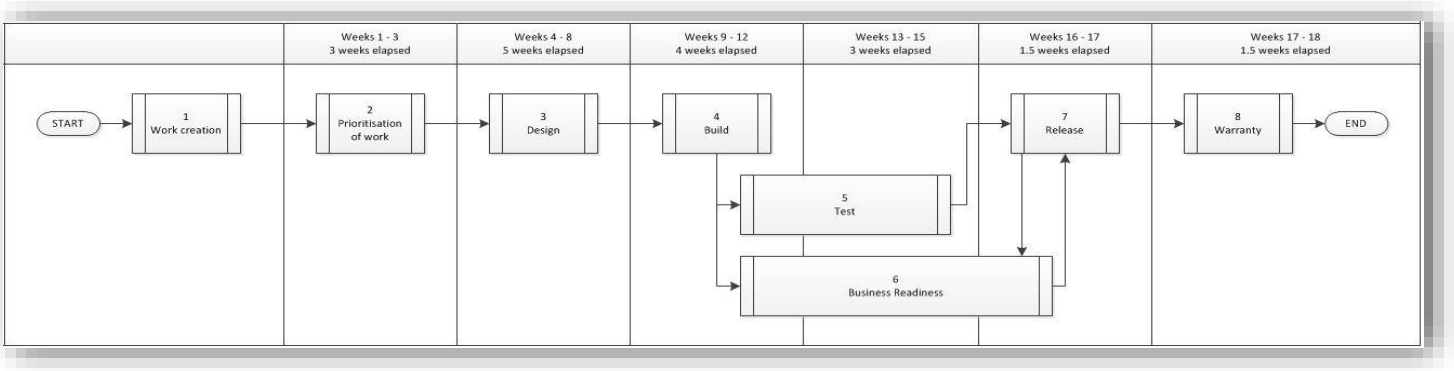
Η ομάδα αυτή λοιπόν είναι υπεύθυνη για την εφαρμογή, τον έλεγχο και υποστήριξη των συνεχιζόμενων βελτιώσεων, αλλαγών και σφαλμάτων που προκύπτουν ευρύτερα από την επιχείρηση (εταιρίες) και το οικοσύστημα του ηλεκτρονικού εμπορίου (E-Commerce Ecosystem).

Η πλειοψηφία των εργασιών συντονίζεται από την ομάδα σχεδιασμού (Projects Team). Η ομάδα αυτή είναι συνυφασμένη με την παροχή νέων ή την αλλαγή τρεχόντων χαρακτηριστικών. Τα καθήκοντα αναθέτονται και τελικά να εκτελούνται από τις τρεις ομάδες ανάπτυξης που αποτελούνται από, τους γραφίστες (Creative), τους προγραμματιστές (Build) και τους Δοκιμαστές (Test).

Η Ομάδα υποστήριξης εξασφαλίζει πως το οικοσύστημα ηλεκτρονικού εμπορίου είναι σε πλήρη λειτουργία ανά πάσα στιγμή. Παράλληλα διασφαλίζει τις συναλλαγές μέσω παρακολούθησης, διατήρησης και υποβολής εκθέσεων σχετικά με το πλήρες οικοσύστημα εμπορίου.

Αφότου τα σχέδια ολοκληρωθούν από την ομάδα σχεδιασμού, τότε θα ξεκινήσει η διαδικασία παραγωγής. Η διαδικασία της κυκλοφορίας του λογισμικού περιγράφεται στην παρακάτω ενότητα.

Διαδικασία κυκλοφορίας λογισμικού (Project Release Process)



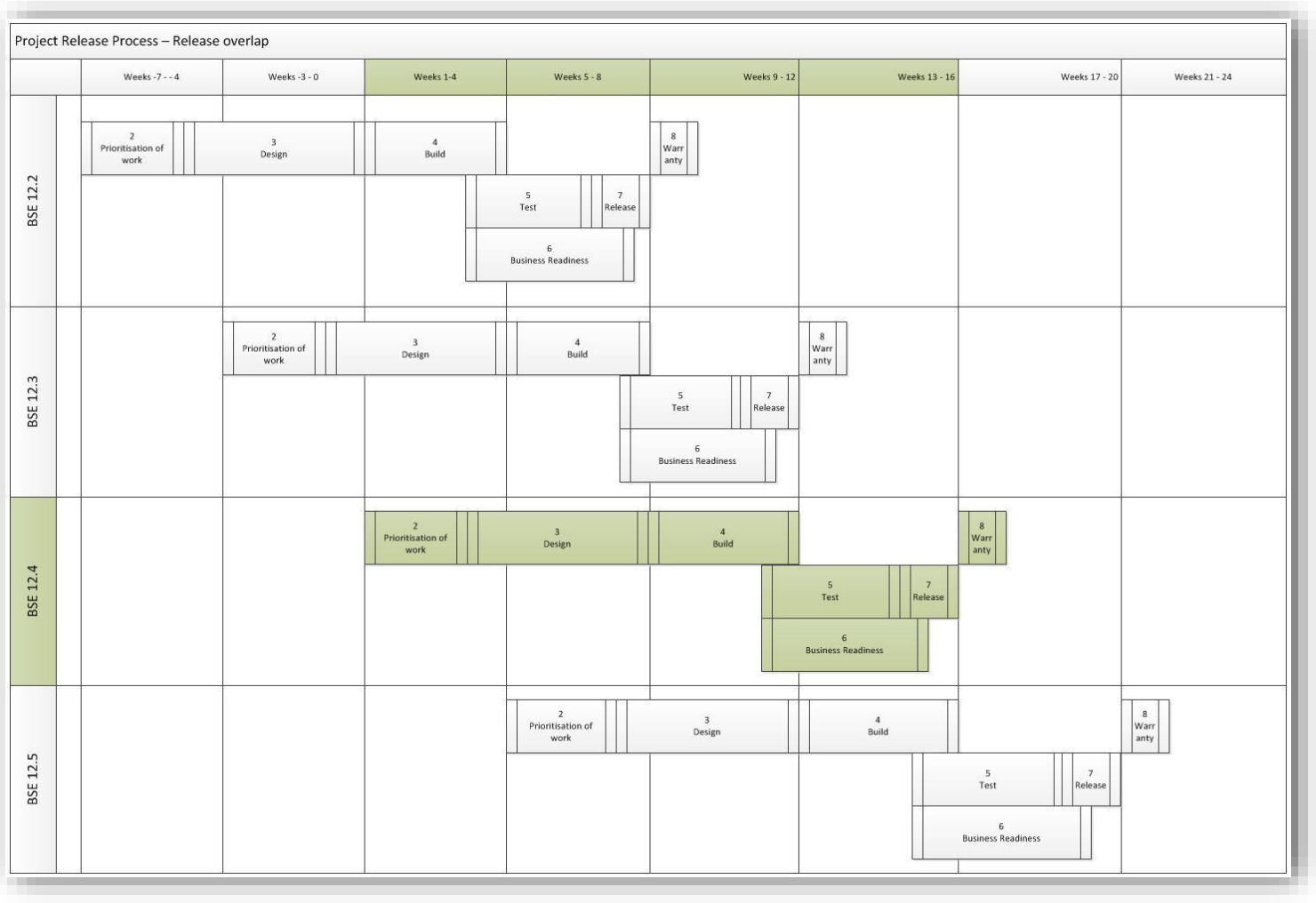
Εικόνα 7.1 Διαδικασία κυκλοφορίας λογισμικού

Σε αυτή την ενότητα περιγράφονται με περισσότερες λεπτομέρειες οι παραπάνω φάσεις:

1. **Δημιουργία εργασίας(Work Creation):** Σε αυτή τη φάση έχουμε τις λεπτομέρειες για το έργο και την ανάθεση των αρμοδιοτήτων της κάθε ομάδας.
2. **Ιεράρχηση των εργασιών(Prioritization of work):** Μετά από την ανάθεση του έργου, πρέπει να προσδιοριστεί η προτεραιότητα των ενεργειών της κάθε φάσης.
3. **σχέδιο(Design):** Η φάση του σχεδιασμού είναι σημαντική, δεδομένου ότι διασφαλίζει την ύπαρξη τεκμηρίων με βάση τις ανάγκες για βελτιώσεις που έχει η επιχείρηση.
   Μια σειρά από εργαστήρια πραγματοποιούνται για να γίνει παράδοση τόσο των λειτουργικών όσο και των τεχνικών περιγραφών των προγραμματισμένων αλλαγών.
4. **Κατασκευή (Build):** Τυχόν βελτιώσεις ή σφάλματα που είναι φτιαγμένα(built)/επιδιορθωμένα είτε από τη τεχνική σκοπιά του συστήματος είτε με τη μορφή νέων επιχειρησιακών διαδικασιών.
5. **Δοκιμή(Test):** Η δοκιμαστική φάση είναι αφιερωμένη στην εξασφάλιση ότι αυτό που παραδίδεται από τις ομάδες κατασκευής (Developers/Creative), υποστηρίζει τα αναμενόμενα πρότυπα, όπως αυτά δόθηκαν από την ομάδα σχεδιασμού ή τους πελάτες.
   Εάν μια φάση προερχόμενη από την κατασκευή δεν πληρεί τα απαιτούμενα πρότυπα, συνήθως μετακινείτε σε μια νεότερη έκδοση; ενώ σε ορισμένες περιπτώσεις, ανάλογα με τη προτεραιότητα της βελτίωσης/επιδιόρθωσης ο κύκλος ανάπτυξης μπορεί και να καθυστερήσει.
6. **Ετοιμότητα της επιχείρησης(Business Readiness):** Τα βήματα σε αυτή την φάση εξασφαλίζουν ότι η επιχείρηση είναι έτοιμη για τυχόν βελτιώσεις που έχουν εισαχθεί μέσα σε κάθε κυκλοφορία λογισμικού. Αυτή η φάση αυτή θα μπορούσε να περιλαμβάνει την εκπαίδευση των χρηστών για τα νέα εργαλεία, προγράμματα επίγνωσης για την ομάδα Εξυπηρέτησης Πελατών, ή τη δημιουργία νέου περιεχομένου από τις εταιρίες της ιστοσελίδας.
7. **Κυκλοφορία λογισμικού (Release):** Η φάση αυτή περιλαμβάνει το συντονισμό των ομάδων πριν από την κυκλοφορία του λογισμικού.
8. **Εγγύηση(Warranty):** Αυτή η φάση έχει σχεδιαστεί για να επικυρώσει ότι το έργο που επιτελέσθηκε, κυκλοφόρησε τόσο για τους πελάτες όσο και την επιχείρηση είναι:
   - κατάλληλου επιπέδου
   - αποτελεσματικά παραδοτέο στις ομάδες υποστήριξης
   - αντανακλάται σε όσον αφορά τη διαδικασία της παράδοση από τις ομάδες εργασίας
   - ανατροφοδοτούμενο για το ευρύτερο κοινωνικό σύνολο, τόσο από την μεριά της διαδικασίας όσο και από τη μεριά των "πρόσθετων βελτιώσεων"

Επικάλυψη της κυκλοφορίας του λογισμικού ανάμεσα στις ομάδες

Η μέθοδος ορίζεται κυρίως γύρω από ένα σταθερό χρόνο, τη φάση κατασκευής 4 εβδομάδων. Για να εξασφαλιστεί ότι η ομάδα εργάζεται συνεχώς υπάρχουν 4 κυκλοφορίες που τρέχουν παράλληλα, με κάθε κυκλοφορία να αποτελεί διαφορετική φάση του κύκλου:



Εικόνα 7.2 Κυκλοφορία λογισμικού

7. Περιγραφή της διαδικασίας Δοκιμών της εταιρίας Bestseller.

Για την ανοικοδόμηση των δυνατοτήτων της ομάδας δοκιμών, ανατίθεται σε ένα μέλος αυτής οι αρμοδιότητες του υπεύθυνου δοκιμών (test lead). Παράλληλα αναλαμβάνει πρόσθετες ευθύνες για τις κυκλοφορίες λογισμικού οι οποίες περιλαμβάνουν.

- Συμμετοχή στη συνεδρία για τη γενική αναθεώρηση της φάσης του σχεδιασμού.
  Παρέχει την εκτίμηση για το μέγεθος της δοκιμής (ουσιαστικά η ομάδα σχεδιασμού κάνει κάποια πρόβλεψη και ο υπεύθυνος δοκιμών ενημερώνει εάν η πρόβλεψη της είναι υλοποιήσιμη).
  Μοιράζεται πληροφορίες με την ομάδα των δοκιμών, για επερχόμενες κυκλοφορίες λογισμικού κατά τη διάρκεια εσωτερικών συνεδριάσεων.

- Συμμετοχή στη συνεδρία για τη λεπτομερή αναθεώρηση της φάσης του σχεδιασμού.
  Παρέχει λεπτομερή εκτίμηση και με πλάνο για το μέγεθος της δοκιμής.
  Μοιράζεται τις λεπτομέρειες με την ομάδα των δοκιμών για το χρονικό περιθώριο τη προτεραιότητα κάθε λειτουργίας ενώ κάνει και την ανάθεση εργασιών.

- Αφήνει τις δοκιμές του λογισμικού που του έχουν ανατεθεί μία εβδομάδα πριν από κάθε κυκλοφορία, ενώ η ομάδα οφείλει να συνεχίσει μέχρι και την ημέρα της κυκλοφορίας. Ο υπεύθυνος δοκιμών χρειάζεται την εβδομάδα αυτή για να διαβάσει και να επανεξετάσει τα τεκμήρια του σχεδιασμού της επόμενης κυκλοφορίας; Έτσι ώστε να είναι έτοιμος να παραθέσει τις απόψεις του για την επερχόμενη συνεδρία αναθεώρησης της φάσης του σχεδιασμού ως εμπειρογνώμων του αντικειμένου (Subject Matter Expert). Παράλληλα θα πρέπει να ξεκινήσει τη δημιουργία του σχεδίου δοκιμών της επόμενης κυκλοφορίας για την ανάθεση της εργασίας.

Οι δοκιμές πραγματοποιούνται σε μια χρονική περίοδο 3 εβδομάδων, ενώ αξίζει να σημειώσουμε τα στάδια των δοκιμών που πρέπει να εκτελεστούν.

1. Δοκιμή καπνού (Smoke Testing)
2. Δοκιμές παλινδρόμησης (Regression test)
3. Δοκιμές αποδοχής χρήστη (User acceptance testing)
4. Δοκιμή καπνού (Smoke Testing) στο περιβάλλον ενσωμάτωσης (Staging [11])
5. Δοκιμή καπνού (Smoke Testing) στο περιβάλλον της παραγωγής.

[11] Staging: Περιβάλλον που αποτελεί ομοίωμα της παραγωγής και χρησιμοποιείται για τον έλεγχο της ενσωμάτωσης από το περιβάλλον δοκιμών στο περιβάλλον της παραγωγής

7.1 Δοκιμή καπνού (Smoke Testing)

Η δοκιμή καπνού είναι μία μη διεξοδική μέθοδος της δοκιμής του λογισμικού. Κατά κοινή ομολογία, σε αυτή τη δοκιμή Θα πρέπει να δοκιμαστούν τα πιο κρίσιμα τμήματα του συστήματος, ωστόσο δεν πρέπει να ελέγξουν διεξοδικά όλες τις διαφορετικές λειτουργικές περιοχές του συστήματος, αρκεί μόνο η επιβεβαίωση της λειτουργίας τους. Παρακάτω βλέπετε μερικά γενικά σενάρια δοκιμών καπνού, Ο χρήστης λοιπόν θα πρέπει να:

- έχει πρόσβαση σε όλες τις μάρκες όλων των διαφορετικών χωρών.
- είναι σε θέση να αναζητήσει κάποιο προϊόν.
- είναι σε θέση να προσθέτει ένα αντικείμενο στο καλάθι.
- είναι σε θέση να επιλέξει μέθοδο μεταφοράς.
- είναι σε θέση να ολοκληρώσει την παραγγελία μέσω όλων των διαθέσιμων μεθόδων πληρωμής
- έχει πρόσβαση στο λογαριασμό του.
- είναι σε θέση να εξάγει τις παραγγελίες στα διάφορα συστήματα όπως αυτό των παραγγελιών (OMS), της αποθήκης (Hermes), των πληρωμών (Global Collect, PayPal κτλ.), της λέσχης πελατών (customer Club) κτλ .

Αξίζει να αναφερθεί πως το εργαλείο αυτοματοποιημένων δοκιμών που θα αναπτύχτηκε βασίστηκε πάνω στις δοκιμές καπνού και σκοπό είχε να καλύψει τα πρώτα 5 από τα παραπάνω σενάρια!

Γενικότερα, η δοκιμή καπνού εφαρμόζεται για να εξακριβώσει ότι η ομάδα είναι ευχαριστημένη με την ενσωμάτωση και πως το περιβάλλον είναι έτοιμο για διεξοδικές δοκιμές. Επίσης επιτρέπει στην ομάδα να πιάσει σε πρώιμο στάδιο, οτιδήποτε το οποίο μπορεί να επηρεάσει ή να καθυστερήσει τις δοκιμές όταν θα χρειαστεί να προχωρήσει στην εκτέλεση λεπτομερή σεναρίων. Η καθυστέρηση οφείλετε στην επιστροφή του κώδικα στους προγραμματιστές για την αποσφαλμάτωση.

7.2 Δοκιμές παλινδρόμησης (Regression test)

Ο σκοπός των δοκιμών παλινδρόμησης είναι να εξασφαλιστεί ότι η αποσφαλμάτωση του κώδικα από τους προγραμματιστές, κατά τη διάρκεια της περιόδου κατά την οποία επιδιορθώνουν το κώδικα, δεν παρουσιάζει νέα ελαττώματα. Οι δοκιμές αυτές γίνονται σε κύκλους δοκιμής 1, 2 κλπ.

Οι δοκιμές παλινδρόμησης είναι σημαντικές για τη συνέπεια την οποία παρουσιάζουν, καθώς επιτρέπουν στους δοκιμαστές να εντοπίσουν ελαττώματα τα οποία δεν παραβρίσκονταν στον τελευταίο κύκλο δοκιμών σε σχέση με τον επόμενο κύκλο. Τη στιγμή που οι νέες λειτουργίες έχουν δοκιμαστεί μέσα στον τρέχοντα κύκλο, στη συνέχεια αυτά τα ”νέα” σενάρια μεταφέρονται στην κασετίνα δοκιμών παλινδρόμησης (regression test suite) έτοιμα να αποτελέσουν μέρος του επόμενου κύκλου.
Οι νέες λειτουργίες, θα πρέπει πάντα να ελέγχονται πριν η ομάδα δοκιμών προχωρήσει στη κασετίνα

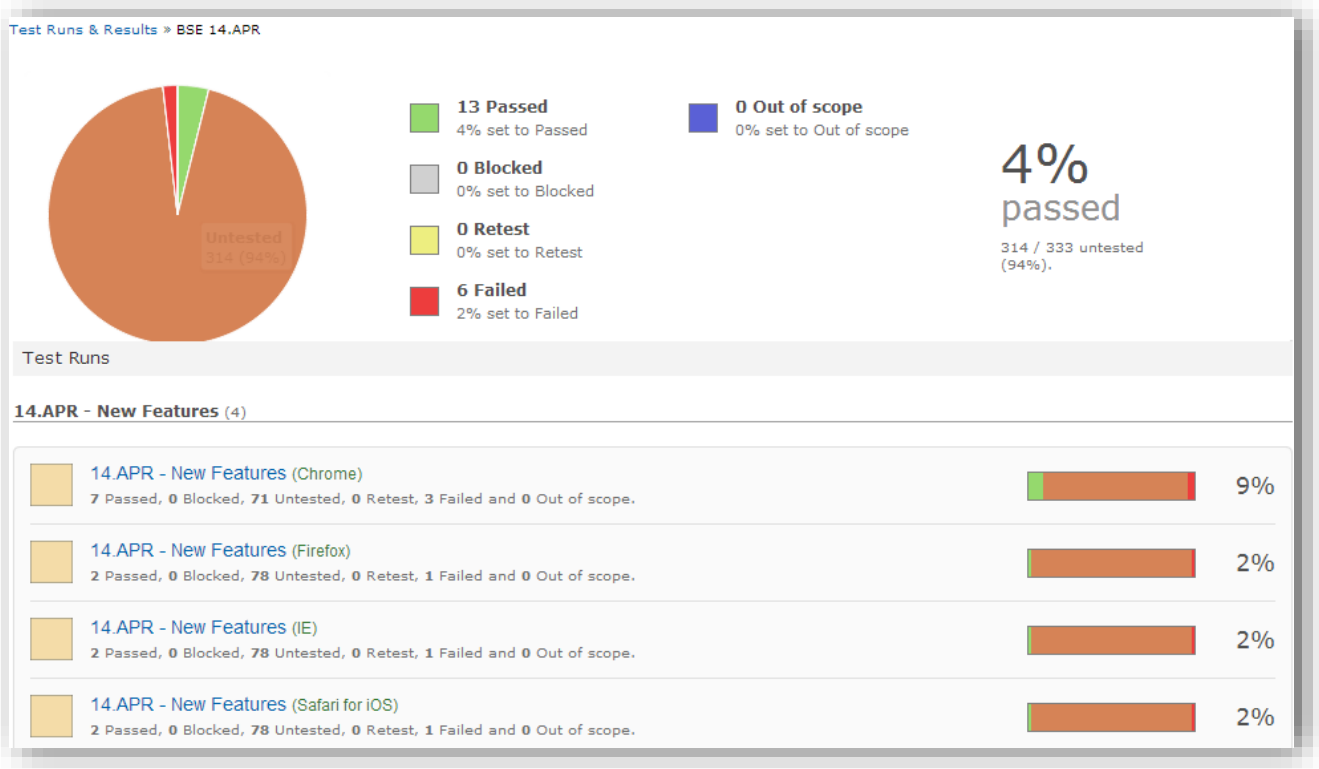
δοκιμών παλινδρόμησης. Συνοψίζοντας μπορούμε να επισημάνουμε, πως οι δοκιμές παλινδρόμησης όσο περνάει ο καιρός δοκιμάζουν εκτενέστερα το βάθος του συστήματος.

Οφείλουμε να τονίσουμε πως δεν είναι αναγκαίο η ομάδα δοκιμών να τρέχει όλα τα σενάρια των ελέγχων παλινδρόμησης σε κάθε κύκλο δοκιμής; Εκτός βεβαία εάν το επιτρέπει ο χρόνος. Αντ 'αυτού, ως μέρος της δοκιμής του σχεδιασμού, τομείς που επιλέγονται με βάση τις αλλαγές του κώδικα κατά τη διάρκεια της κατασκευής θεωρούνται ως Υψηλής, Μέσης και Χαμηλής επίπτωσης. Μόλις λοιπόν προσδιοριστούν τα κατάλληλα σενάρια δοκιμών, θα πρέπει να προστεθούν στους κύκλους δοκιμών με σειρά προτεραιότητας.

Παραδείγματος χάριν: Εάν κατά τη διάρκεια του προγραμματιστικού κύκλου, η δουλειά επηρέασε το καλάθι και το ταμείο (Checkout), τότε οι περιπτώσεις δοκιμών παλινδρόμησης του καλαθιού και του ταμείου θα είναι υψηλής προτεραιότητας. Ενώ παράλληλα, οι δοκιμές άλλων τομέων όπως ο λογαριασμός των χρηστών (My account) ή λίστα με τα επιθυμητά προϊόντα (Wishlist) θα χαρακτηρίζονταν ως χαμηλού κινδύνου και η δοκιμές θα γίνονταν προς το τέλος ή ίσως και καθόλου.

Παράλληλα, για να υπάρχει μεγαλύτερη κάλυψη από τη σκοπιά των προγραμμάτων περιήγησης, πρέπει να προσδιορίζονται τα προγράμματα περιήγησης που θα χρησιμοποιηθούν κατά τη διάρκεια του κύκλου δοκιμών και να διανέμονται στους δοκιμαστές. Αυτά είναι που πρέπει να αλλάζουν συστηματικά οι δοκιμαστές στους κύκλους δοκιμών, με συνέπεια την να εξασφαλίσουν ένα υψηλότερο επίπεδο εμπιστοσύνης στη διεπαφή χρήστη (UI).

Παρακάτω ακολουθεί μια βιβλιοθήκη η οποία περιέχει δοκιμές παλινδρόμησης και αποτελέσματα αυτών για μια συγκεκριμένη έκδοση λογισμικού (Test Runs & Results of a regression suite)



Εικόνα 8.1 Regression Test

8. Ανάπτυξη του εργαλείου δοκιμών αυτοματισμού

Η διαπίστωση από την αναφορά στο προηγούμενο κεφάλαιο είναι πώς η ομάδα δοκιμών της Bestseller, επαναλαμβάνει πολλά σενάρια δοκιμών κατά τη διάρκεια κάθε κυκλοφορίας λογισμικού. Αυτός ο λόγος ωθεί στην αναζήτηση ενός εργαλείου το οποίο θα μπορεί αυτόματα να εκτελεί κάποια από τα σενάρια αυτά. Ο στόχος είναι το εργαλείο αυτό να λύσει τα χέρια των δοκιμαστών γλυτώνοντας τους χρόνο ο οποίος είναι σημαντικός για τη μετέπειτα βελτίωση τη ποιότητας των δοκιμών. Στο κεφάλαιο 5, παρατίθενται αναλυτικά τα χαρακτηριστικά της αυτόματης δοκιμής. Σε αυτό το σημείο, αξίζει να σημειωθεί, πως το εργαλείο αυτό δεν αποσκοπεί σε καμία περίπτωση στην αντικατάσταση των μελών της ομάδας δοκιμών, αλλά στη βελτιστοποίηση του αποτελέσματος.

Εύκολα, λοιπόν, μπορεί ο καθένας να συμπεράνει πως ο αυτόματος έλεγχος, εάν εφαρμοστεί σωστά μπορεί να δώσει πολλές λύσεις στην ομάδα δοκιμών και στην επιχείρηση γενικότερα με την αποφυγή σφαλμάτων. Ακολουθώντας λοιπόν τις απαιτήσεις της αγοράς, τα τελευταία χρόνια, πολλές εταιρίες έχουν εντρυφήσει στο χώρο των δοκιμών είτε παρέχοντας υπηρεσίες συμβουλευτικές είτε παράγοντας εργαλεία αυτοματισμού. Η διάθεση των εργαλείων αυτών δεν μπορούμε να πούμε πως είναι ευρέως διαδεδομένη, καθώς δεν έχουν καταφέρει να κατακτήσουν την αγορά.

Ένας σημαντικός λόγος της περιορισμένης διάδοσης τους, είναι πως στην αγορά υπάρχουν αρκετά Open Source λογισμικά και εργαλεία τα οποία εξυπηρετούν αυτούς τους σκοπούς. Παράλληλα, τα εργαλεία αυτομάτων δοκιμών μπορούν εύκολα να αναπτυχθούν μέσα στο περιβάλλον της κάθε ενδιαφερόμενης εταιρίας από τις ομάδα ανάπτυξης λογισμικού.

8.1 Προκλήσεις της ανάπτυξης του εργαλείου

Η κύριες προκλήσεις του εγχειρήματος αυτού, ήταν η δημιουργία ενός εργαλείου το οποίο θα εκτελεί διάφορα σενάρια δοκιμών, στα ευρέως διαδεδομένα προγράμματα περιήγησης του Web. Παράλληλα θα πρέπει να κάνει ανάκτηση δεδομένων από διαφορετικές βάσεις δεδομένων, που εξυπηρετούν τη δυναμικότητα του εργαλείου. Το UI θα πρέπει να είναι φιλικό για το χρήστη, ενώ επίσης να του δίνει τη δυνατότητα να επιλέξει οποιοδήποτε από τα διαθέσιμα σενάρια για τη δοκιμή.

Εκείνο που έχει ιδιαίτερη σημασία, αποτελεί και μία από τις μεγαλύτερες προκλήσεις κατά τη δημιουργία οποιοδήποτε εργαλείου τέτοιου τύπου, είναι η αξιοπιστία και η ασφάλεια του. Εδώ θα σταθούμε καθώς καμία εταιρία ηλεκτρονικού εμπορίου δεν επιτρέπεται να εφαρμόσει οποιοδήποτε μη αξιόπιστο η ασφαλή εργαλείο αυτοματοποιημένων δοκιμών. Ο λόγος είναι πως οι ζημιές που μπορούν να δημιουργηθούν για κάποιες εταιρίες θα είναι ανεπανόρθωτες.

Για παράδειγμα, φανταστείτε το εν λόγω εργαλείο σε κάποια σελίδα κρατήσεων αεροπορικών εισιτηρίων διαφόρων εταιριών. Το εν λόγο παράδειγμα, είναι καίριο καθώς το προϊόν που η εταιρίες αυτές προωθούν δεν τους ανήκει ουσιαστικά, είναι απλά μεσάζοντες που προσφέρουν υπηρεσίες. Ένα πιθανό ενδεχόμενο είναι εργαλείο αυτό, λόγω κακής ανάπτυξης είτε διαχείρισης του λογισμικού, κάνει κρατήσεις σε διάφορες αεροπορικές εταιρίες και για διαφορετικές πτήσεις. Το αποτέλεσμα αυτής της πράξης για οποιαδήποτε εταιρία τέτοιου τύπου, θα ήταν κατά πολύ ζημιογόνο έως και καταστροφικό.

8.2 Σενάρια δοκιμών του εργαλείου αυτοματισμού

Το εργαλείο αυτό λοιπόν, αναπτύχθηκε με σκοπό τον αυτόματο έλεγχο κυρίως κατά τη διάρκεια των δοκιμών καπνού (Smoke test) και κάποιων σεναρίων δοκιμών παλινδρόμησης (Regression test). Παράλληλα, εξυπηρετεί τους δοκιμαστές και στην δημιουργία παραγγελιών, όταν αυτό επιβάλλεται από άλλα συστήματα που πρέπει να δοκιμαστούν.

Πιο συγκεκριμένα, το εργαλείο θα πραγματοποιεί έλεγχο από άκρη σε άκρη (End-to-End). Αυτό σημαίνει από τη στιγμή που ο χρήστης εισέρθει στο κατάστημα, μέχρι την ολοκλήρωση της παραγγελίας. Από τη σκοπιά του χρήστη, μεταφράζεται ως:

- Δυνατότητα πρόσβασης στο κατάστημα με τη χρήση κάποιων εκ των κυρίων διαθέσιμων προγραμμάτων περιήγησης (Google Chrome, Mozilla Firefox, Internet Explorer, Safari κτλ.)
- Δυνατότητα πρόσβασης σε όλες τις μάρκες όλων των χωρών που είναι διαθέσιμο το ηλεκτρονικό κατάστημα. Με πρόχειρους υπολογισμούς, τουλάχιστον 170 σελίδες αφού γίνεται αναφορά σε 14 εταιρίες και 14 διαθέσιμες χώρες διάθεσης. (επί του παρόντος υπάρχουν 56 σελίδες – καθώς οι χώρες δοκιμής είναι 4)
- Δυνατότητα αναζήτησης οποιουδήποτε προϊόντος και εισαγωγής του στο καλάθι (ανάλογα με τη μάρκα, καθώς τα προϊόντα είναι διαφορετικά ανάμεσά τους).
- Δυνατότητα επιλογής μεθόδου μεταφοράς. (τουλάχιστον 17 διαθέσιμες μέθοδοι μεταφοράς οι οποίες μοιράζονται ανάμεσα στις διάφορες χώρες)
- Δυνατότητα ολοκλήρωσης της παραγγελίας- Checkout, μέσω όλων των διαθέσιμων μεθόδων πληρωμής (10 διαθέσιμες μέθοδοι πληρωμής: Global Collect, Klarna, iDeal, Nordea, PayPal κτλ.)

8.3 Περιβάλλον ανάπτυξης λογισμικού: Visual Studio

Γενικότερα, υπάρχουν διάφορα Ολοκληρωμένα Περιβάλλοντα Ανάπτυξης (IDE[12]). Στις ήμερες μας, τα πιο σημαντικά και ευρέως διαδεδομένα από αυτά είναι το Visual Studio Express, το Eclipse και το Netbeans.

Η τελική επιλογή έγινε τελικά με γνώμονα τις προσωπικές επιρροές του χρήστη. Έτσι η ανάπτυξη της εφαρμογής πραγματοποιήθηκε στο ολοκληρωμένο περιβάλλον ανάπτυξης **Visual Studio**. Είναι μια ολοκληρωμένη συλλογή εργαλείων και υπηρεσιών, που συνθέτουν τη δημιουργία μια μεγάλης ποικιλίας από εφαρμογές, για διάφορες πλατφόρμες. Χρησιμοποιείται κυρίως από την προγραμματιστές που παράγουν .NET κώδικα.

Ένα από τα σημαντικά του πλεονεκτήματα, είναι πως παρέχεται σε εκδώσεις express ή ultimate. Το γεγονός αυτό παρέχει τη δυνατότητα επιλογής στο χρήστη, ανάλογα με την χρήση που προορίζει το παραγόμενο λογισμικό. Αυτό το καθιστά ιδανικό για τους περιστασιακούς χρήστες, που θέλουν να παράγουν κάποιες γραμμές κώδικα, όπως στη προκειμένη περίπτωση μια πτυχιακής εργασίας. Παράλληλα, εξυπηρετεί και προγραμματιστές που το χρειάζονται για να αναπτύξουν λογισμικό σε κώδικα σε C#.

[12] IDE: Integrated development environment (Ολοκληρωμένο περιβάλλον ανάπτυξης)

8.4 Γλώσσα προγραμματισμού: C#

Η επιλογή της γλώσσας προγραμματισμού έγινε όπως και στο IDE, με βάση τη προσωπική εμπειρία του χρήστη σε γλώσσες προγραμματισμού.

Είναι μια γλώσσα που εξυπηρετεί και επιτρέπει την πραγματοποίηση όλων των στόχων για την ανάπτυξη του συγκεκριμένου λογισμικού, καθώς είναι ιδιαίτερα διακρατική και δυναμική. Παράλληλα πρέπει να αναφέρουμε, πως μια από τις βασικές προϋποθέσεις της γλώσσας προγραμματισμού, είναι και η δυνατότητα εφαρμογής του Selenium. Το εργαλείο αυτό αποτελεί θεμέλιο του αυτοματισμού.

8.5 Εισαγωγή στο εργαλείο Selenium

Είναι ένα από τα πιο διαδεδομένα εργαλεία για δοκιμές αυτοματισμού στο Web. Πιο συγκεκριμένα, το εργαλείο ειδικεύεται στην αυτοματοποίηση προγραμμάτων περιήγησης. Είναι ένα σύνολο από διαφορετικά εργαλεία λογισμικού, που το καθένα έχει διαφορετική προσέγγιση, υποστηρίζοντας πάντα την αυτοματοποίηση της δοκιμής.

Το σύνολο των εργαλείων της, οδηγεί σε μια πλούσια σειρά από λειτουργίες δοκιμών, ειδικά διαμορφωμένες για τις ανάγκες των δοκιμών web εφαρμογών όλων των τύπων. Οι λειτουργίες αυτές είναι εξαιρετικά ευέλικτες, προσφέροντας πολλές επιλογές για τον εντοπισμό στοιχείων της διεπαφής του χρήστη (UI). Παράλληλα, συγκρίνει τα αναμενόμενα αποτελέσματα των δοκιμών, με βάση την αναμενόμενη συμπεριφορά της εφαρμογής.

Ένα από τα βασικά χαρακτηριστικά του Selenium είναι η υποστήριξη για τη διενέργεια δοκιμών ενός ατόμου σε πολλαπλές πλατφόρμες περιήγησης. Το εργαλείο αυτό έχει τρείς εκδόσεις.

- Selenium RC
- Selenium WebDriver
- Selenium IDE

Οι τρείς παραπάνω εκδόσεις έχουν διαφορετικά χαρακτηριστικά. Στο εργαλείο αυτοματισμού έχει εφαρμοστεί η έκδοση του WebDriver. Οι λόγοι της επιλογής αυτής είναι κυρίως γιατί δεν απαιτείται κάποιος Server για να ξεκινήσει , αλληλεπιδρά με το πρόγραμμα περιήγησης ενώ παράλληλα δεν έχει αμιγώς αντικειμενοστραφή διεπαφή προγραμματισμού εφαρμογών (API[13]). Το τελευταίο οδηγεί στη δυνατότητα μετακίνησης του ποντικιού αυτόματα, από τα προγράμματα περιήγησης.

[13] Application programming interface

8.6 Εφαρμογή τριών επιπέδων (three-tier application)

Για τη δημιουργία της εφαρμογής της δοκιμής αυτοματισμού, δόθηκε βάση στο μοντέλο τριών επιπέδων (three-tier application). Στον τομέα της ανάπτυξης του WEB, η εφαρμογή τριών επιπέδων συχνά αναφέρεται σε ιστοσελίδες. Οι σελίδες αυτές είναι συνήθως ηλεκτρονικού εμπορίου και κατασκευάζονται σε τρία επίπεδα: στο περιβάλλον εργασίας ή παρουσίασης (Presentation layer / UI), στο περιβάλλον της επιχειρηματικής λογικής (Logic tier) και στη βάση δεδομένων και προγραμματισμού (Data tier).

Ένα από τα βασικά πλεονεκτήματα του μοντέλου, είναι το γεγονός του διαχωρισμού των στρωμάτων μίας εφαρμογής σε διαφορετικές φυσικές τοποθεσίες. Αυτό σημαίνει αυτόματα πως το εργαλείο γίνεται πιο επεκτάσιμο και συντηρήσιμο του. Το συμπέρασμα αυτό προκύπτει από το γεγονός, πως το debug είναι πολύ πιο εύκολο όταν για παράδειγμα, υπάρχουν κάποια προβλήματα στο στρώμα πρόσβασης δεδομένων και όλος ο κώδικας πρόσβασης δεδομένων είναι συγκεντρωμένος εκεί.

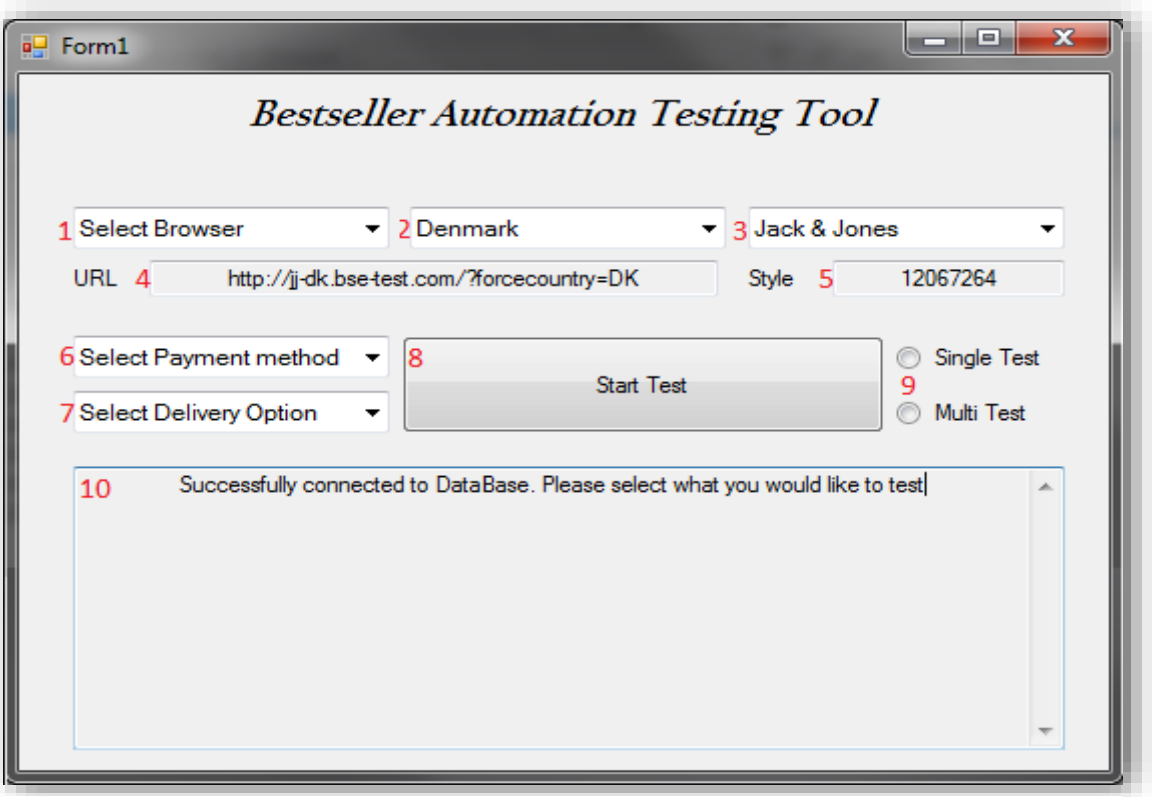
Περιβάλλον εργασίας ή παρουσίασης (Presentation layer / UI)

Αυτό είναι το κορυφαίο επίπεδο της εφαρμογής. Η βαθμίδα παρουσίασης εμφανίζει πληροφορίες που σχετίζονται με υπηρεσίες όπως την περιήγηση των εμπορευμάτων, την αγορά και το περιεχόμενο του καλαθιού αγορών του πελάτη. Επικοινωνεί με από κάτω βαθμίδες (επιχειρηματικής λογικής και βάσης δεδομένων) με τις οποίες παρουσιάζει τα αποτελέσματα στον περιηγητή του χρήστη. Με απλούστερα λόγια, είναι ένα στρώμα στο οποίο οι χρήστες μπορούν να έχουν άμεση πρόσβαση, όπως μια ιστοσελίδα ή ένα γραφικό σύστημα διεπαφής χρήστη (GUI).

Στη προκειμένη περίπτωση, είναι μια εφαρμογή GUI η οποία δέχεται εντολές από το χρήστη και με βάση την επιχειρηματική λογική και τις πληροφορίες που ανακτά από την Βάση δεδομένων εκτελεί τις λειτουργίες που της έχουν ανατεθεί.

Το UI του εργαλείου βρίσκεται στη κλάση MainForm.cs η οποία θα περιγραφεί αναλυτικότερα, στο ένατο κεφάλαιο της αρχιτεκτονικής του κώδικα.

Παρακάτω ακολουθεί μια σύντομη περιγραφή για την γραφική διεπαφή χρήστη της εφαρμογής.



Εικόνα 8.2 Software User Interface

Περιγραφή του UI

1. Επιλογή Browser: διαθέσιμοι περιηγητές είναι οι Chrome, Firefox και Internet Explorer.
2. Επιλογή χώρας: Παρόλο που το κατάστημα υποστηρίζει 14 διαφορετικές χώρες, το εργαλείο του αυτοματισμού εφαρμόζεται για αρχή σε 4 χώρες (Denmark, Netherlands, Germany και UK).
3. Επιλογή μάρκας: Όλες οι μάρκες του καταστήματος είναι διαθέσιμες.
4. Παρουσίαση URL: Κατόπιν ο χρήστης θα επιλέξει χώρα και μάρκα, το πεδίο του URL κάνει ανάκτηση από τη βάση δεδομένων και επιδεικνύει το URL πάνω στο οποίο ο χρήστης θα κάνει τη δοκιμή.
5. Παρουσίαση Style: Όπως και στο προηγούμενο πεδίο, γίνεται ανάκτηση από τη βάση δεδομένων και επίδειξη της τιμής αυτής. Προαπαιτούμενα είναι πάλι η επιλογή χώρας και μάρκας από το χρήστη.
6. Επιλογή μεθόδου πληρωμής: Υπάρχουν διάφορες μέθοδοι πληρωμής αυτή τη στιγμή στο ηλεκτρονικό κατάστημα όπως Credit Card, PayPal, RTBT (RealTimeBankTransfer) και Klarna.
7. Επιλογή μεθόδου παράδοσης: Όπως και με τις μεθόδους πληρωμής, έτσι και οι μέθοδοι παράδοσης ποικίλουν. Βέβαια από τη στιγμή που το εργαλείο εφαρμόζεται σε 4 χώρες και όχι σε όλες ο αριθμός των μεθόδων παράδοσης είναι πιο περιορισμένος. Πιο συγκεκριμένα έχουμε: GLS Parcel Shop, Post Denmark, HERMES, UPS, KIALA, HLG και HLG/UPS τα οποία μοιράζονται ανάλογα με τη χώρα που βρίσκεται το ηλεκτρονικό κατάστημα.
8. Έναρξη δοκιμής, με βάση τις προηγούμενες εισαγωγές του χρήστη.
9. Επιλογή δοκιμής ανάμεσα
    - σε απλή δομική συγκεκριμένης χώρας-εταιρίας-μεθόδου πληρωμής-μεθόδου μεταφοράς
    - και σε πολλαπλή, που περιλαμβάνει συνδυασμό των παραπάνω
10. Απεικόνιση των σημείων που παρουσιάστηκαν λάθη κατά τη διάρκεια της δοκιμής.

Επιχειρηματική λογική (Logic tier)

Η λογική βαθμίδα προέρχεται από την βαθμίδα παρουσίασης και ελέγχει τη λειτουργικότητα μιας εφαρμογής, εκτελώντας λεπτομερή επεξεργασία.
Πιο συγκεκριμένα, αυτό το επίπεδο συντονίζει την εφαρμογή, εκτελεί εντολές, παίρνει λογικές αποφάσεις και εκτιμήσεις ενώ έχει τη δυνατότητα να κάνει υπολογισμούς. Παράλληλα μετακινεί και επεξεργάζεται δεδομένα ανάμεσα στα δύο επίπεδα που το περιβάλλουν.

Στη περίπτωση της εφαρμογής μας, η επιχειρηματική λογική του εργαλείου αυτοματισμού είναι ορισμένες βασικές λειτουργίες όπως:

1. Επιλογή οδηγού ανάλογα με το πρόγραμμα περιήγησης που θα επιλέξει ο χρήστης
2. Αναζήτηση συγκεκριμένου προϊόντος από το πεδίο της αναζήτησης της σελίδας- SearchBox
3. Μετάβαση στα αποτελέσματα της αναζήτησης
4. Τοποθέτηση του προϊόντος στο καλάθι
5. Μετάβαση στο καλάθι και επιλογή μεθόδου μεταφοράς
6. Μετάβαση στη σελίδα του Checkout οπού τα στοιχεία του χρήστη θα πρέπει να συμπληρωθούν ανάλογα με τη χώρα που βρίσκεται και επιλογή καταστήματος παραλαβής του πακέτου
7. Μετάβαση στη σελίδα πληρωμής και συμπλήρωση των στοιχείων για την ολοκλήρωση της πληρωμής
8. Έναρξη επικοινωνίας με τη βάση δεδομένων, ούτως ώστε το εργαλείο να κάνει ανάκτηση δεδομένων που χρειάζεται σε κάθε βήμα.

Οι λειτουργίες αυτές βρίσκονται στη κλάση Automation.cs η οποία θα περιγραφεί πιο αναλυτικά στο επόμενο κεφάλαιο, στη αρχιτεκτονική του κώδικα.

```
33  namespace AutomationTool2
34  {
35      class Automation
36      {
37
38          private string parentWindowHandle;
39          private RemoteWebDriver driver;
```

Επίπεδο Βάσης Δεδομένων (Data tier)

Αυτό το επίπεδο αποτελείται από εξυπηρετητές βάσεων δεδομένων, όπου τα δεδομένα αποθηκεύονται και ανακτώνται. Αυτό το επίπεδο διατηρεί τα δεδομένα ουδέτερα και ανεξάρτητα από τους διακομιστές εφαρμογών και την επιχειρηματική λογική. Αυτό δίνει στα δεδομένα το δικό τους επίπεδο, βελτιώνοντας την επεκτασιμότητα και την απόδοση του συστήματος.

Από τη σκοπιά του Web development: είναι μια βάση δεδομένων, η οποία περιλαμβάνει όσο το σύνολο των δεδομένων όσο και το λογισμικό του συστήματος διαχείρισης των βάσεων δεδομένων. Αυτό με την κατάλληλη διαχείριση, παρέχει πρόσβαση στα δεδομένα.

Στη προκειμένη περίπτωσή του εργαλείου αυτοματισμού, οι βάσεις δεδομένων είναι δύο ενώ η επιλογή τους έγινε με βάση τις ανάγκες της εφαρμογής.

1. SQL: Σε αυτή τη βάση δεδομένων, βρίσκονται τα δεδομένα τα οποία χρειάζεται το εργαλείο ούτως ώστε να φτάσει μέχρι και το Checkout. Πιο συγκεκριμένα, περιέχει δεδομένα και παρέχει συσχετίσεις για τα διάφορα URL, Styles και τα μεγέθη των ρούχων (size, length και quantity)
2. EXCEL: Το αρχείο αυτό αποτελεί βάση δεδομένων, καθώς αποθηκεύονται και ανακτώνται δεδομένα χρήσιμα για την ολοκλήρωση της δοκιμής.
   Η συγκεκριμένη βάση παρέχει δεδομένα/πληροφορίες τα οποία χρησιμοποιούνται από το checkout μέχρι και την ολοκλήρωση της δοκιμής. Ουσιαστικά, πληροφορίες δοκιμών (Testing Data) που αφορούν το χρήστη. Τα δεδομένα αυτά είναι προαπαιτούμενα για την συμπλήρωση της φόρμας του checkout (ονοματεπώνυμο, διεύθυνση, Email, τηλέφωνο, κτλ) όπως και πληροφορίες για τις μεθόδους πληρωμής (αριθμός πιστωτικής, ημερομηνία λήξης, CSV, κτλ).

Οι λειτουργίες αυτές βρίσκονται στη κλάση SQL.cs η οποία θα περιγραφεί πιο αναλυτικά στο επόμενο κεφάλαιο, στη αρχιτεκτονική του κώδικα.

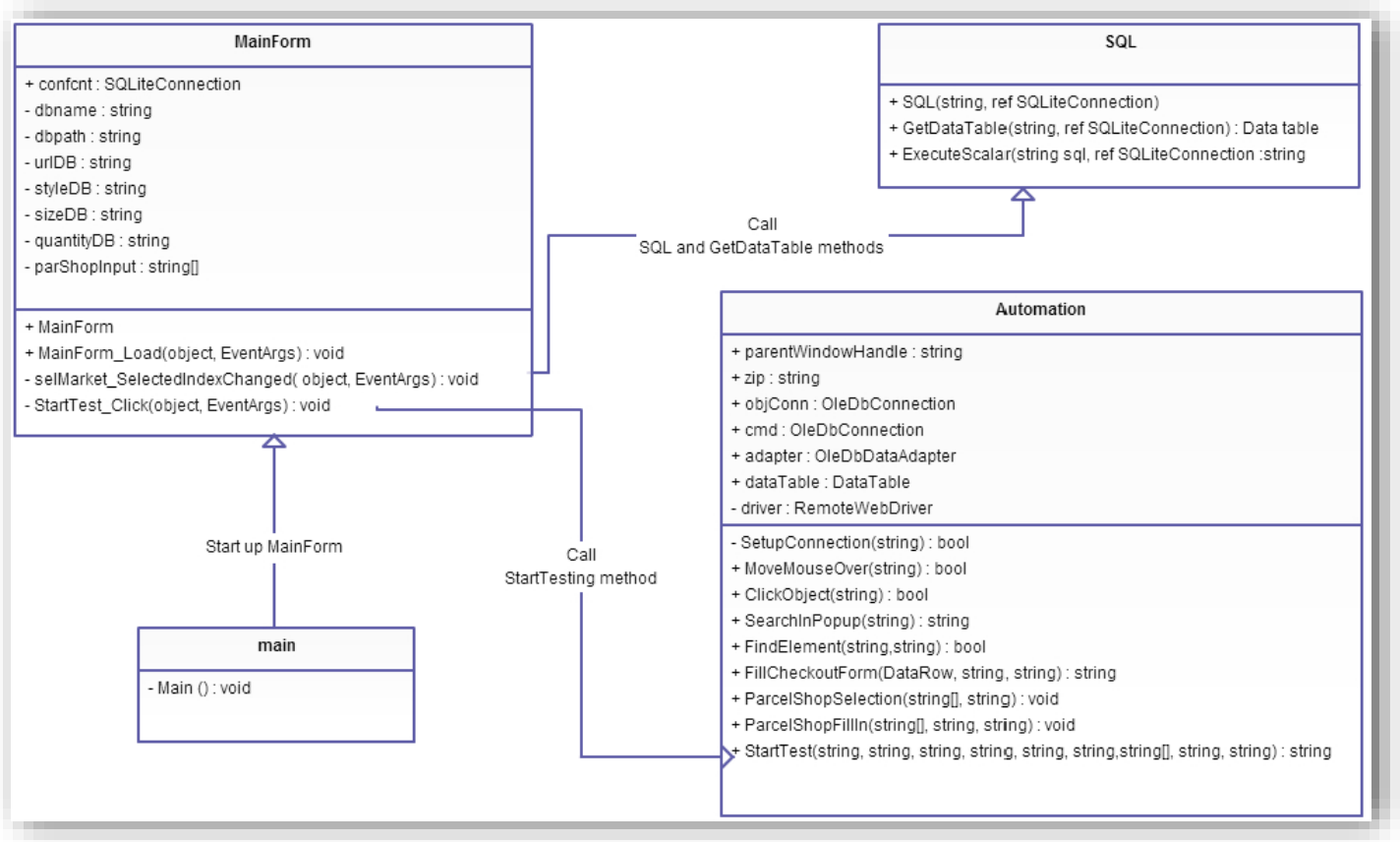
(Wright, 2013) (Howitz, 2012) (Rouse, 2007)

9. Η αρχιτεκτονική του κώδικα

Αφού τελείωσε η περιγραφή του εργαλείου, σε αυτό το κεφάλαιο ακολουθεί η περιγραφή της δόμησης και των βασικών εντολών του κώδικα. Η περιγραφή θα ξεκινήσει με τη γραφική απεικόνιση των κλάσεων στο διάγραμμα UML.

Στη συνέχεια θα ακολουθήσει το διάγραμμα ενεργειών. Αυτό θα περιγράψει τη διαδικασία που ακολουθεί το εργαλείο αυτοματισμού, παρουσιάζοντας τις διάφορες αλληλεπιδράσεις του ιστού. Οι εργασίες αυτές γίνονται στο περιηγητή μέσω του webDriver της Selenium.

9.1 UML Class diagram

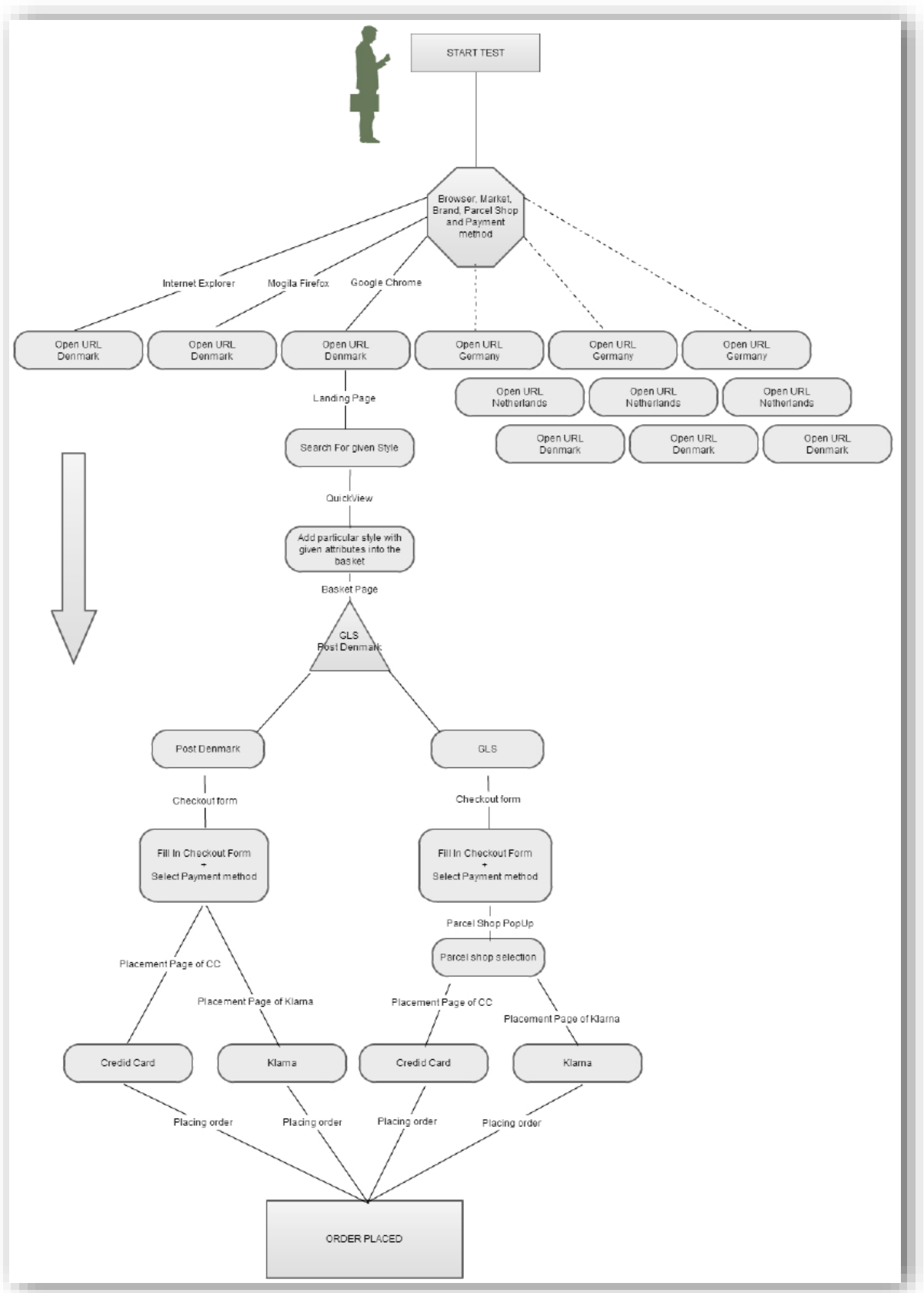


Εικόνα 9.1 Class diagram UML

(tutorialspoint.com)

9.2 Activity Diagram

Το διάγραμμα ενεργειών που ακολουθεί, περιγράφει τη διαδικασία που ακολουθεί το εργαλείο αυτοματισμού από τη στιγμή που ο χρήστης ξεκινάει τη δοκιμή. Η περιγραφή που ακολουθεί εστιάζει στη δοκιμή του ηλεκτρονικού καταστήματος της Δανίας, επιλέγοντας παράλληλα το Google Chrome ως πρόγραμμα περιήγησης. Εδώ αξίζει να σημειωθεί, πως δεν υπάρχει περιγραφή των υπολοίπων σεναρίων για λόγους συντομίας, καθώς όλα τα υπόλοιπα σενάρια δοκιμών, έχουν παρόμοια δομή.



Εικόνα 9.10 Activity Diagram UML

9.3 Βιβλιοθήκες

Το εργαλείο καλεί ως επί το πλείστων δύο βιβλιοθήκες. Η μια είναι η γενική, αυτή της Microsoft (System Namespace) και παρέχει πρόσβαση σε κλάσεις που περιλαμβάνουν βάσεις δεδομένων. Η δεύτερη ονομάζεται Selenium, είναι πιο εξειδικευμένη καθώς αυτοματοποιεί προγράμματα περιήγησης. Και οι δύο αναλύονται παρακάτω.

System Namespace

```
//Generic System libraries

            using System;
            using System.Text;
            using System.Collections.Generic;
            using System.ComponentModel;
            using System.Windows.Forms;
            using System.Drawing;
            using System.Linq;
            using System.Threading.Tasks;
```

Η *System.Text* περιλαμβάνει μαθήματα που αντιπροσωπεύουν ASCII και Unicode κωδικοποίηση χαρακτήρων.

Η *System.Collections* περιέχει διεπαφές και κλάσεις που καθορίζουν διάφορες συλλογές αντικειμένων, όπως λίστες, ουρές, πίνακες κατακερματισμού και λεξικά.

Η *System.ComponentModel* παρέχει κλάσεις που χρησιμοποιούνται για την εφαρμογή του χρόνου εκτέλεσης και σχεδίασης της συμπεριφοράς των εξαρτημάτων και του ελέγχου τους. Παράλληλα περιλαμβάνει τις θεμελιώδεις κλάσεις και διεπαφές για την εφαρμογή των χαρακτηριστικών και μετατροπείς τύπων, σύνδεση με τους παρόχους δεδομένων και τα συστατικά για τις χορηγήσεις αδειών.

Η *System.Windows.Forms* περιέχει κλάσεις για τη δημιουργία εφαρμογών που βασίζονται στα Windows. Αυτές επωφελούνται πλήρως από τα πλούσια χαρακτηριστικά της διεπαφής του χρήστη, που διατίθενται στο λειτουργικό σύστημα των Windows.

Η *System.Drawing* παρέχει πρόσβαση σε GDI (Graphics Device Interface) και βασικές λειτουργίες γραφικών.

Η *System.Linq* παρέχει κλάσεις και διεπαφές που υποστηρίζουν ερωτήματα, τα οποία χρησιμοποιούν Language-Integrated Queries (LINQ).

Η *System.Threading* παρέχει κλάσεις και διασυνδέσεις που επιτρέπουν τον προγραμματισμό πολλαπλών νημάτων. Είναι σημαντική, καθώς περιλαμβάνει μια κλάση Timer που εκτελεί μεθόδους επανάκλησης σε θέματα συγκέντρωσης νημάτων.

```
// Database access libraries
            using System.Data;
            using System.Data.SQLite;
            using System.Data.OleDb;
```

Το System.Data παρέχει πρόσβαση σε κλάσεις που αντιπροσωπεύουν την αρχιτεκτονική ADO.NET. Η αρχιτεκτονική αυτή, επιτρέπει την οικοδόμηση εξαρτημάτων που διαχειρίζονται αποτελεσματικά δεδομένα από πολλαπλές πηγές.

Στο σενάριο του Διαδικτύου, η ADO.NET παρέχει τα εργαλεία για την αίτηση, ενημέρωση και το συμβιβασμό των δεδομένων σε συστήματα πολλαπλών επιπέδων. Η αρχιτεκτονική ADO.NET εφαρμόζεται επίσης σε εφαρμογές πελάτη, όπως οι διάφορες Windows Forms , ή σελίδες HTML που δημιουργήθηκαν μέσω ASP.NET.

Το επίκεντρο της αρχιτεκτονικής ADO.NET είναι η κλάση DataSet. Κάθε DataSet μπορεί να περιέχει πολλά DataTable αντικείμενα. Κάθε DataTable περιέχει δεδομένα από μια μόνο πηγή δεδομένων, όπως είναι ένας SQL Server ή κάποιο αρχείο δεδομένων που στη περίπτωση μας είναι το Excel.

Selenium

```
// Selenium libs
using OpenQA.Selenium;
using OpenQA.Selenium.Remote;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.IE;
using OpenQA.Selenium.Support;
using OpenQA.Selenium.Support.UI;
using OpenQA.Selenium.Support.Events;
using OpenQA.Selenium.Interactions;
```

Η Selenium είναι ένα σύνολο από διαφορετικά εργαλεία λογισμικού, με το καθένα από αυτά να έχει μια διαφορετική προσέγγιση στην υποστήριξη της αυτοματοποιημένης δοκιμής.
Το σύνολο των εργαλείων της βιβλιοθήκης αυτής, οδηγεί σε μια σειρά από πλούσιες λειτουργίες ελέγχου. Καλείται ειδικά για τις ανάγκες των δοκιμών, web εφαρμογών διαφορετικών τύπων. Οι λειτουργίες αυτές είναι εξαιρετικά ευέλικτες, επιτρέποντας τον εντοπισμό στοιχείων UI. Παράλληλα συγκρίνει τα αναμενόμενα αποτελέσματα των δοκιμών με βάση την πραγματική συμπεριφορά της εφαρμογής. Ένα από τα βασικά χαρακτηριστικά του Selenium είναι η υποστήριξη της διενέργειας δοκιμών, σε πολλαπλές πλατφόρμες περιήγησης.

Ο οδηγός του Web driver της Selenium, υποστηρίζει ένα μεγάλο εύρος προγραμμάτων περιήγησης με τα βασικά να είναι τα παρακάτω:

- Google Chrome,
- Internet Explorer 6, 7, 8, 9, 10 - 32 και 64-bit (όταν χρειάζεται)
- Mozilla Firefox
- Safari
- Opera
- HtmlUnit
- Android (with Selendroid or appium)
- iOS (with ios-driver or appium)

(Singh, 2012)

9.4 Ανάλυση βασικών εντολών του κώδικα

- `public partial class MainForm : Form`

//Δήλωση του ονόματος της βάσης δεδομένων
```
protected const string dbname = @"\db\automated.s3db";
```

//Αναζήτηση του μονοπατιού για την βάση δεδομένων SQLite.
```
dbpath = AppDomain.CurrentDomain.BaseDirectory + dbname;
```

//Εγκαθίδρυση της σύνδεσης με τη βάση δεδομένων SQLite
```
new SQL(dbpath, ref confcnt);
```

// Γέμισμα των Comboboxes της χώρας.
Τα δεδομένα εισόδου προέρχονται από τη βάση δεδομένων μέσω ενός Query.
```
selMarket.DataSource = SQL.GetDataTable("SELECT DISTINCT(market) FROM inputDB", ref confcnt);
selMarket.DisplayMember = "market";
selMarket.ValueMember = "market";
```

Το εργαλείο είναι Δυναμικό και ο χρήστης μέσω του UI, έχει τη δυνατότητα ενημέρωσης των Comboboxes. Η βάση δεδομένων αυτή τη στιγμή αποτελείται από 4 χώρες και 40 διαφορετικούς συνδυασμούς ανάλογα με τη χώρα και την εταιρία που ο χρήστης επιλέγει. (Κλειδί είναι το πεδίο URL)

ID	URL	market	brand	style	size	quantity
1	http://dwd-dk.bestseller.com/jack-jones	Denmark	Jack & Jones	12069181	L	1
2	http://dwd-dk.bestseller.com/jlindeberg	Denmark	J.Lindeberg	11111111	L	1
3	http://jj-roe.bse-test.com/?forcecountry=GB	ROE-UK	Jack & Jones	12069181	L	1
4	http://jj-de.bse-test.com/?forcecountry=DE	Germany	Jack & Jones	12069181	L	1
13	http://dwd-nl.bestseller.com/jack-jones	Netherlands	Jack & Jones	12069181	L	1

// Έτσι λοιπόν, οι τιμές θα πρέπει να ενημερώνονται. Στη περίπτωση μας όμως, κάθε αλλαγή επηρεάζει παραπάνω από ένα πεδίο. Παράδειγμα είναι μία αλλαγή στη εταιρία(Brand) η οποία θα επηρεάσει τα παρακάτω πεδία: `InUrl, InStyle, InSize, και InQuantity.` Είναι σημαντική η ενημέρωση των πεδίων αυτών, καθώς στη συνέχεια θα χρησιμοποιηθούν ως δεδομένα εισόδου για τη δοκιμή.
```
private void selMarket_SelectedIndexChanged(object sender, EventArgs e) {
DataTable brand = SQL.GetDataTable("SELECT * FROM inputDB WHERE market = '" + selMarket.Text + "'
AND brand = '" + selBrand.Text + "' LIMIT 1", ref confcnt);
```

// Ανάκτηση δεδομένων από τη βάση δεδομένων.
```
DataRow row = brand.Rows[0];
row["url"].ToString();
```

// Κλήση της μεθόδου StartTest. Με τη κλήση γίνεται πέρασμα στη κλάση Automation, στην οποία πραγματοποιείται η δοκιμή. Αξίζει να σημειώθεί πως η διεπαφή του χρήστη τελειώνει σε αυτό το σημείο.

```
Automation test = new Automation();
String input = test.StartTest(selBrowser.Text, urlDB, styleDB, sizeDB, quantityDB,
selMarket.Text, parShopInput, selPayment.Text, selParShop.Text);
outputBox.Text = input;
```

- **class Automation**

// Δήλωση της βάσης δεδομένων και των χαρακτηριστικών της (Excel)

```
public String sConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source='C:\\HPM
Client\\TestIoannis.xlsx';Extended Properties= \"Excel 12.0 Xml;HDR=YES\";";
```

// Εγκαθίδρυση σύνδεσης με τη βάση δεδομένων (Excel)

```
SetupConnection("InputSheet");
```

// Εισαγωγή ολόκληρης γραμμής από το Excel

```
foreach (DataRow dr in dataTable.Rows)
{
```

// Ελέγχει τις γραμμές της βάσης. Η χώρα που επέλεξε ο χρήστης μέσω του UI, θα πρέπει να αντιστοιχεί σε αυτή της βάσης. Αυτό γιατί κάθε χώρα έχει διαφορετικά στοιχεία χρηστών, τα οποία είναι υποχρεωτικά για την ολοκλήρωση της δοκιμής.

```
<dr["Country"].ToString() != market>
 if (dr["Country"].ToString() != market) {
      continue;
 }
```

Country	Title	FirstName	LastName	Street	Housenumber	Address2	Zipcode	City
Denmark	Fr.	Jonas	Tzanakis	Amaliegade	22	Wellhausen	8000	Aarhus
Germany	Hr.	Ioannis	Tzanakis	Alexandersplatz	22	test	10115	Berlin
Netherlands	Hr.	Ioannis	Tzanakis	Rijksmuseum	11	test2	1199ZZ	Amsterdam
ROE-UK	Hr.	Ioannis	Tzanakis	Oxford	33st	Test3	A99AA	London

// Επιλογή προγράμματος περιήγησης.

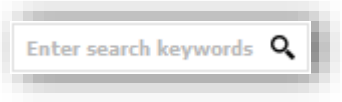
```
try{
     switch (InBrowser) {
          case "Google Chrome":
               driver = new ChromeDriver(@"C:\HPM Client\automation_drivers");
               break;
          case "Mogila Firefox":
```

- **<u>Ο χρήστης αυτή τη στιγμή βρίσκεται στην κεντρική σελίδα κάποιας από τις εταιρίες.</u>**

// κλήση της FindElement. Αυτή είναι βασική μέθοδος, μέσω της οποίας γίνεται αναζήτηση κάποιου πεδίου και εκτέλεση συγκεκριμένων ενεργών. Στη προκειμένη περίπτωση, βρίσκει το πεδίο και το συμπληρώνει με την κατάλληλη γραμματοσειρά <InStyle>

```
FindElement("searchinput", InStyle);
```

```
if (driver.FindElementById(id).Displayed == true){
     driver.FindElement(By.Id(id)).Clear();
     driver.FindElement(By.Id(id)).SendKeys(dr);
     return true;
}
```

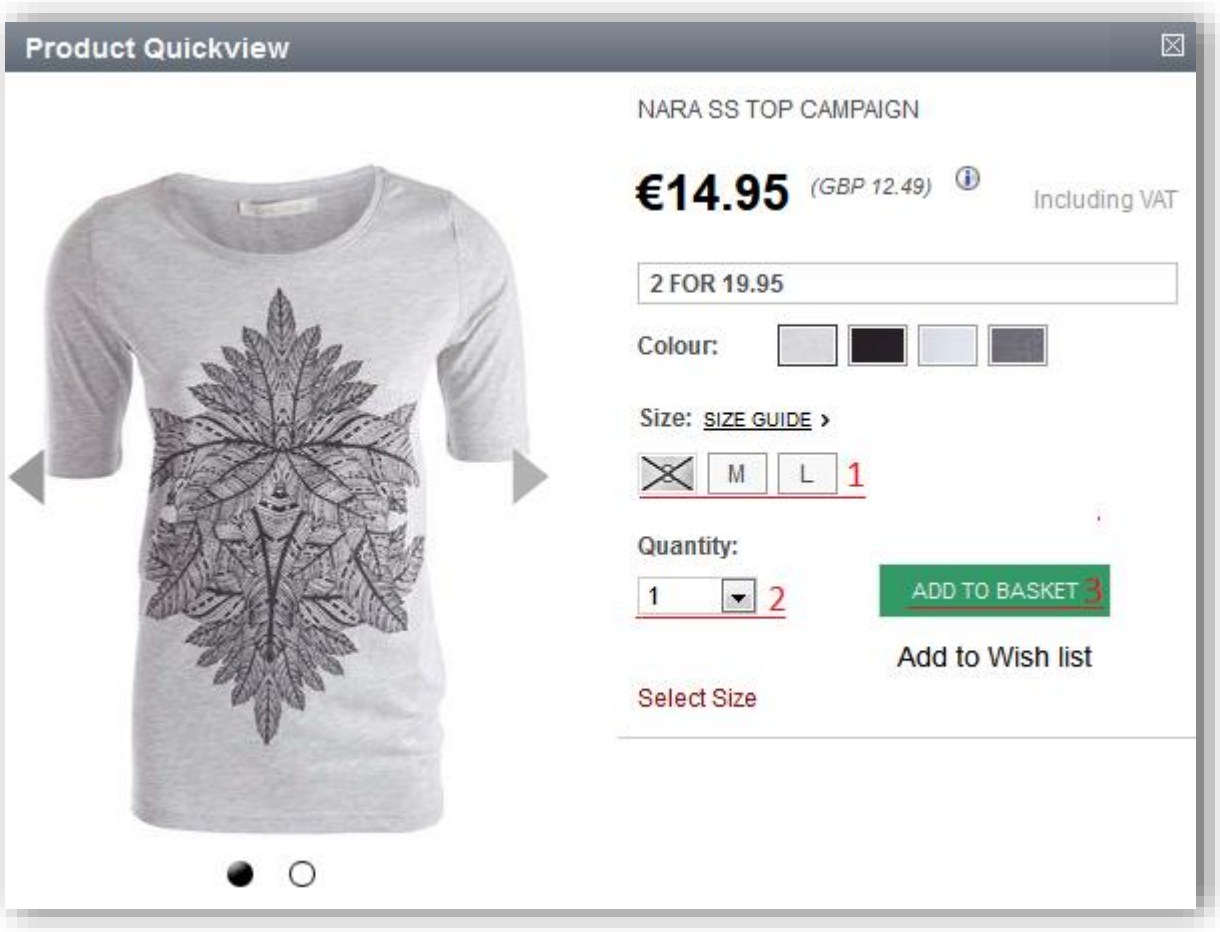


// Με την παρακάτω εντολή εκτελούμε την αναζήτηση

```
driver.FindElement(By.CssSelector("button[type=\"submit\"]")).Click();
```

- **<u>Ο χρήστης αυτή τη στιγμή βρίσκεται στα αποτελέσματα της αναζήτησης.</u>**

// Είναι σημαντικό σε αυτό το βήμα, το εργαλείο να ανοίξει το Quickview μέσω του οποίου θα γίνει και η επιλογή του προϊόντος. Για το άνοιγμα του Quickview, καλείται η μέθοδος hover-Over του mouse.




```
MoveMouseOver(".//*[@class='imagelink']");
ClickObject(".//*[@class='quickviewbutton']/a");
```

// πέρασμα της τιμής size (1)

```
driver.FindElement(By.XPath(".//*[@rel='" + InSize + "']")).Click();
```

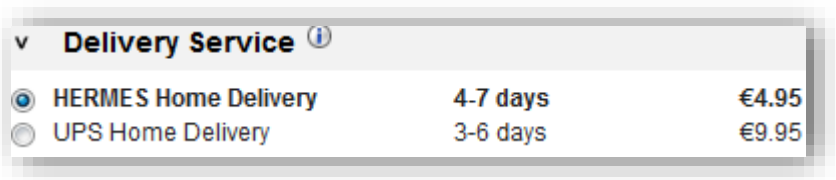
// επιλογή του κουμπιού για τη εισαγωγή του προϊόντος στο καλάθι (3)

```
driver.FindElement(By.XPath(".//*[@id='actionbuttons']/button")).Click();
```

- **<u>Ο χρήστης αυτή τη στιγμή βρίσκεται στο καλάθι.</u>**

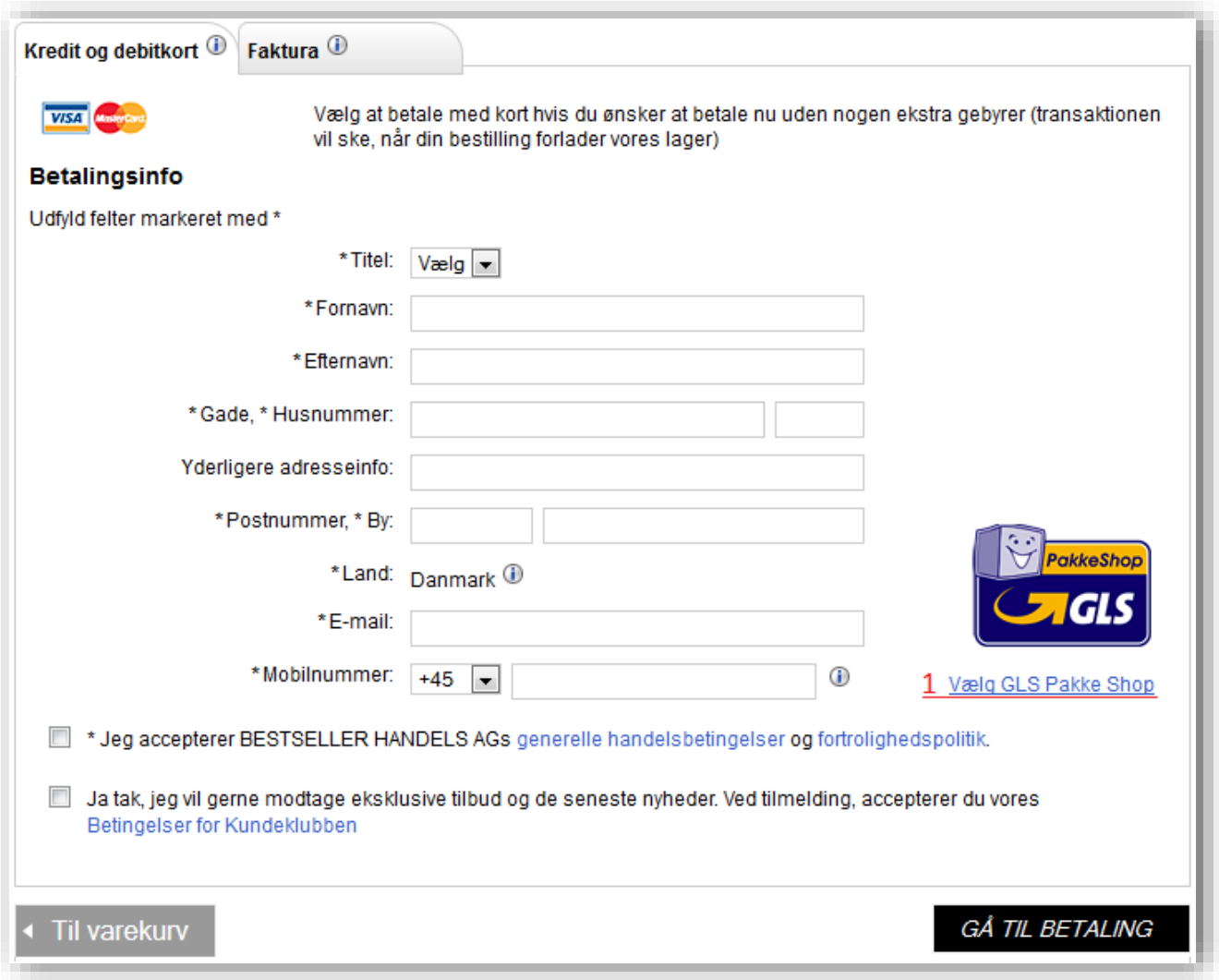
// Κάλεσμα της μεθόδου για την επιλογή μεθόδου μεταφοράς

```
ParcelShopSelection(InParcelShop, ParcelShop);
```



Στην εικόνα αυτή ο χρήστης βρίσκεται στο κατάστημα της UK, με τις μεθόδους παράδοσης να αποτελούνται από την Hermes και UPS. Σε κάθε χώρα οι επιλογές αυτές είναι διαφορετικές.

- **<u>Ο χρήστης αυτή τη στιγμή</u>**



<u>βρίσκεται στο CheckOut.</u>

// Κάλεσμα της μεθόδου για την συμπλήρωση των πεδίων του Checkout.

```
String Zip= FillCheckoutForm(dr, market, InPaymentMethod);
```

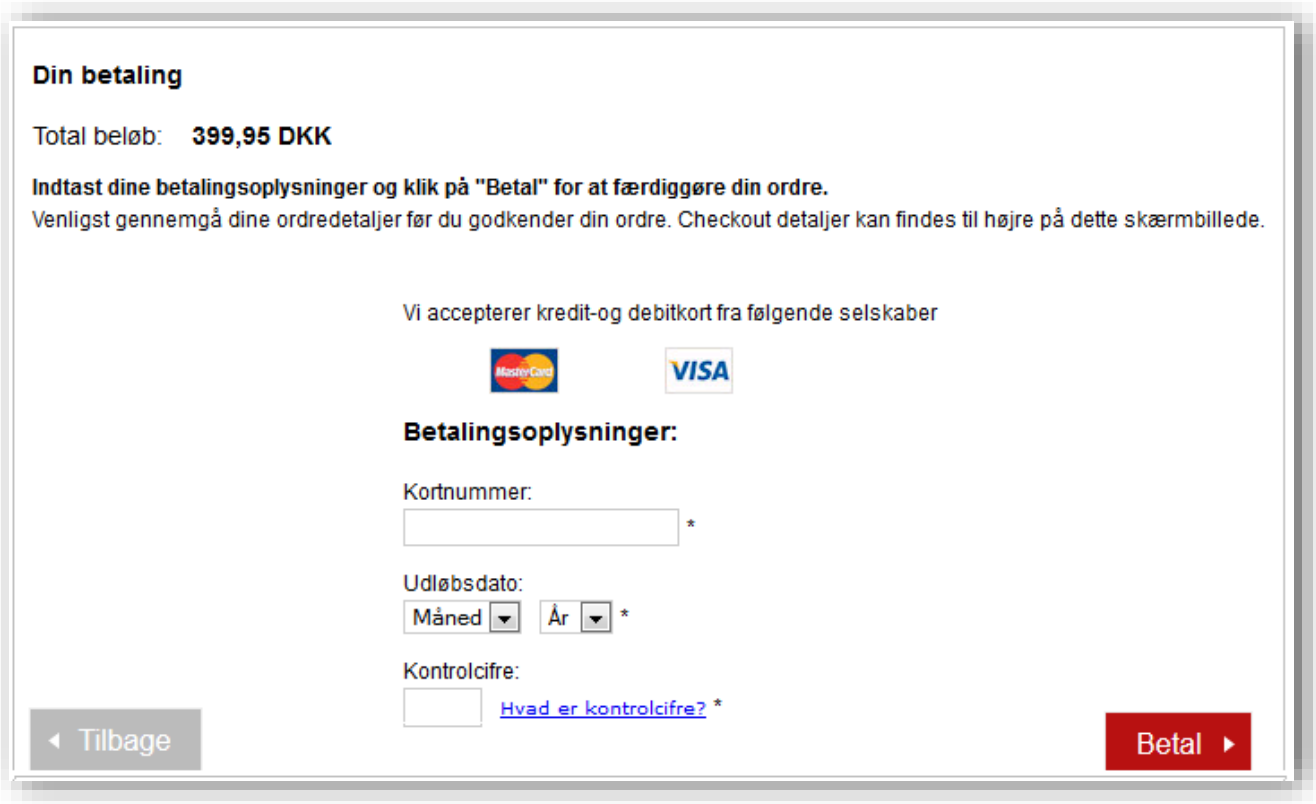
// Σε κάποιες από τις μεθόδους μεταφοράς ο χρήστης έχει την επιλογή καταστήματος παραλαβής.
Η επιλογή του καταστήματος αυτού γίνεται μέσω ενός νέου Pop-up παραθύρου. (1)

```
if (ParcelShop == "GLS Pakke Shop" || ParcelShop == "KIALA" || ParcelShop == "HERMES"){
ParcelShopFillIn(InParcelShop, ParcelShop, Zip);}
```

// Είναι σημαντικό να αναφέρουμε πως τα στοιχεία με τα οποία συμπληρώνονται οι φόρμες, θα πρέπει να προσαρμόζονται για κάθε χώρα. Παράλληλα, κάποιοι μέθοδοι πληρωμής απαιτούν παραπάνω πληροφορίες από άλλες. Για παράδειγμα, για την πληρωμή μέσω Klarna, η ηλικία του χρήστη είναι υποχρεωτικό πεδίο.

```
new SelectElement(driver.FindElement(By.Id("dwfrm_billing_billingAddress_addressFields_year"))).
SelectByIndex(1);
```

- **<u>Ο χρήστης αυτή τη στιγμή βρίσκεται στην σελίδα πληρωμής.</u>**



// Κάθε μέθοδος πληρωμής είναι διαφορετική, αλλά οι διαδικασίες είναι ίδιες. Συμπλήρωση των φορμών και ολοκλήρωση πληρωμής

```
case "Credit Card":
```

```
driver.SwitchTo().Frame(0);
driver.FindElement(By.Name("CREDITCARDNUMBER")).SendKeys(dr["CreditcardNumber"].ToString());
newSelectElement(driver.FindElement(By.Id("F1010_MM"))).SelectByText(dr["CCM"]. ToString());
new SelectElement(driver.FindElement(By.Id("F1010_YY"))).SelectByText(dr["CCY"]. ToString());
driver.FindElement(By.Id("btnSubmit")).Click();
break;
```

10. Αποτελέσματα

Για να συνοψίσουμε, το εργαλείο αυτοματισμού έχει ολοκληρωθεί και βρίσκεται ήδη σε χρήση, από την ομάδα διασφάλισης ποιότητας της Bestseller. Η εποπτεία του για τη συλλογή αποτελεσμάτων, θα διαρκέσει αρκετούς μήνες, ενώ τα πρώτα αποτελέσματα προέκυψαν κατά τη διάρκεια του κύκλου δοκιμών λογισμικού του μήνα Μάρτη (14_March). Όλες οι απαιτήσεις που θα έπρεπε να ακολουθήσει έχουν εφαρμοστεί και το εργαλείο λειτουργεί όσο αποδοτικά ήταν αναμενόμενο. Για να γίνουμε πιο συγκεκριμένοι, θα παρατεθούν τα αποτελέσματα τα οποία έχουν συγκεντρωθεί κατά τη διάρκεια της χρήσης του:

Το βασικό όφελος λοιπόν είναι ο χρόνος που αποκομίσθηκε. Η αυτοματοποιημένη δοκιμή μέσω του εργαλείου εφαρμόστηκε σε:

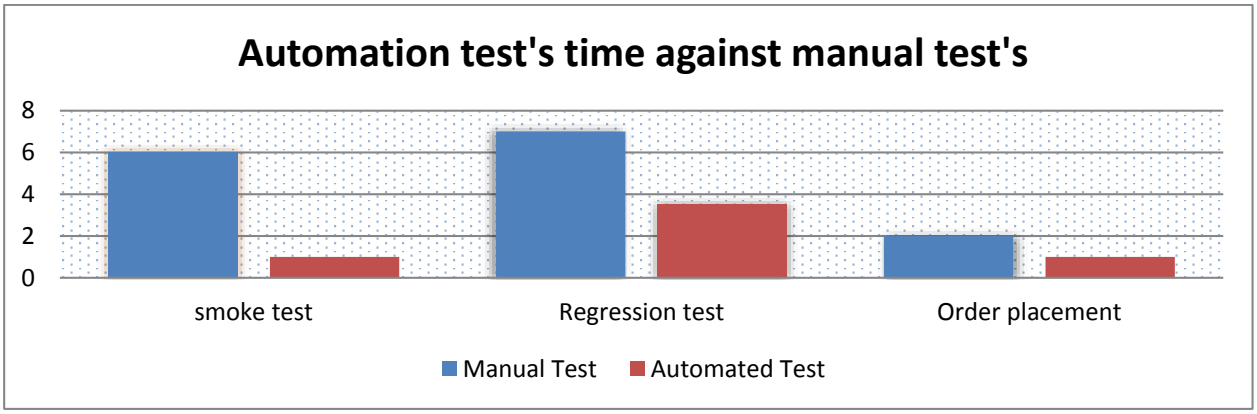
Smoke test: Δοκιμή της ενσωμάτωσης του συστήματος, από το περιβάλλον ανάπτυξης εφαρμογών στο περιβάλλον δοκιμών, από εκεί στο περιβάλλον του staging[14] και τέλος στο περιβάλλον της παραγωγής. Ο χρόνος που διαθέτουμε στην έλλογο δοκιμή χειροκίνητα είναι κατά μέσο όρο μία ώρα, ενώ το εργαλείο αυτοματισμού εκτελεί τη δοκιμή σε δέκα λεπτά. Κατά τη διάρκεια του κύκλου δοκιμών, το εργαλείο εκτελέστηκε έξι φορές με συνολικό χρόνο εκτέλεσης μίας ώρας. Συνολικό κέρδος 5 ώρες.

Regression test: Χρήση κατά τη διάρκεια των δοκιμών παλινδρόμησης (Regression). Ειδικότερα σε σενάρια που βεβαιώνουν την δυνατότητα του χρήστη να μεταβεί στο Checkout. Παράλληλα, θα πρέπει να ολοκληρώσει τη πληρωμή χρησιμοποιώντας όλες τις διαφορετικούς μεθόδους πληρωμής, όλων των χωρών στους τέσσερις βασικούς browsers. Η διαδικασία αυτή απασχολεί συνήθως 2 άτομα για μία ολόκληρη εργάσιμη ημέρα (7 ώρες), ενώ με τη μερική χρήση του αυτοματισμού, η δοκιμή εκτελέστηκε από ένα άτομο στη διάρκεια μίας ημέρας. Συνολικό κέρδος 7 ώρες.

Order placement: Το εργαλείο εκτελέστηκε κατά τη διάρκεια δοκιμών ενσωμάτωσης των παραγγελιών. Ο αυτοματισμός σε αυτές τις δοκιμές εφαρμόστηκε δύο φορές με συνολικό χρόνο εκτέλεσης μίας ώρας. Συνολικό κέρδος 1 ώρα.

Παρακάτω ακολουθεί ένα γράφημα το οποίο παρουσιάζει το συνολικό χρόνο που εξοικονομήθηκε μέσω του αυτοματισμού σε σχέση με το χειροκίνητο έλεγχο κατά τη διάρκεια διαφόρων μεθόδων δοκιμής.

[14] Staging: Περιβάλλον που αποτελεί ομοίωμα της παραγωγής και χρησιμοποιείται για τον έλεγχο της ενσωμάτωσης από το περιβάλλον δοκιμών στο περιβάλλον της παραγωγής



Εικόνα 11 Time Test Results

Σύμφωνα με τα αποτελέσματα του αυτοματισμού, κατά την εφαρμογή του στο κύκλο δοκιμών λογισμικού της bestseller; Το εργαλείο εξοικονομεί το μεγαλύτερο μέρος του Smoke test ενώ επίσης σημαντικό χρόνο κατά τη διάρκεια του Regression test.

Παράλληλα, Αξίζει να σημειωθεί πως λόγο του χρόνου που εξοικονομήθηκε, καλύφθηκαν και αρκετά περισσότερα σενάρια δοκιμών από ότι συνήθως. Αυτό είχε ως συνέπεια την δοκιμή της ποιότητας του λογισμικού, με μεγαλύτερη ακρίβεια και περισσότερη σιγουριά κατά τη πιστοποίηση.

Εν κατακλείδι, πρέπει να γίνει αναφορά στο κόστος του εργαλείου. Σύμφωνα με πρόχειρους υπολογισμούς, για να γίνει η απόσβεση του εργαλείου, θα πρέπει να περάσουν τουλάχιστον πέντε κύκλοι δοκιμών. Αυτό δεν αποτελεί υπερβολή, καθώς για την ανάπτυξη του εργαλείου αφιερώθηκε αρκετός χρόνος, που αγγίζει τις 60 ώρες.

Συμπεράσματα

Το συμπέρασμα είναι, πως με τη ορθή και αποτελεσματική χρήση από την ομάδα δοκιμών, το εργαλείο αυτοματισμού μπορεί να βελτιώσει ριζικά την ποιότητα του λογισμικού. Αυτό είναι σημαντικό να αναφερθεί, καθώς ο βασικός στόχος του αυτοματισμού δεν είναι η αντικατάσταση των μελών της ομάδας δοκιμών. Η σωστή χρήση βέβαια, προϋποθέτει τη διανομή του χρόνου που αντλείται, στον έλεγχο διαφορετικών λειτουργιών και συστημάτων είτε στην βελτίωση και εξέλιξη του εργαλείου.

Σε τελική ανάλυση, το εργαλείο αυτό έχει πολλές δυνατότητες. Είναι σημαντικό σε αυτό το σημείο να τονίσουμε, πως λόγω της μεγάλης αποτελεσματικότητας του στο περιβάλλον δοκιμών, λήφθηκε η απόφαση της εφαρμογής ορισμένων βελτιώσεων.

Ο επόμενος στόχος του εργαλείου, είναι η παρακολούθηση (monitoring), προκειμένου να κάνει δοκιμές στο περιβάλλον της παραγωγής για το εάν το κατάστημα είναι online οποιαδήποτε χρονική στιγμή. Επίσης, θα γίνεται και δοκιμή της επίδοσης (performance) με σκοπό την αποφυγή καθυστερήσεων και μελλοντικά τη βελτιστοποίηση της απόδοσης του καταστήματος.

Βιβλιογραφία

2011, S. J. (2011, 6 09). *www.toolsjournal.com/articles/item/195-10-best-tools-for-test-automation*. Ανάκτηση 3 23, 2014, από toolsjournal.com: http://www.toolsjournal.com/articles/item/195-10-best-tools-for-test-automation

Dickinson. (n.d.). *Software Engineering Basics*. Ανάκτηση από users.dickinson.edu/~wahlst/491/lectures/lec02/se.html: http://users.dickinson.edu/~wahlst/491/lectures/lec02/se.html

Ford, B. D. (2011). *www.starbase.co.uk/attachments/article/171/White%20Paper.%20Selenium%20vs.%20HP%20QuickTest%20Professional.pdf*. Ανάκτηση 03 25, 2014, από www.starbase.co.uk: http://www.starbase.co.uk/attachments/article/171/White%20Paper.%20Selenium%20vs.%20HP%20QuickTest%20Professional.pdf

Group, S. (2011). *http://scales-group.com/*. Retrieved October 18, 2013, from Scales Group: http://scales-group.com/index.php?page=Agile-project-phases

Habib, M. (2013, September 12). *Agile software development methodologies and how to apply them.* Ανάκτηση October 20, 2013, από Codeproject: http://www.codeproject.com/Articles/604417/Agile-software-development-methodologies-and-how-t

Howitz, C. (2012, 06 1). *blog.simcrest.com/what-is-3-tier-architecture-and-why-do-you-need-it/*. Ανάκτηση 3 12, 2014, από blog.simcrest.com: http://blog.simcrest.com/what-is-3-tier-architecture-and-why-do-you-need-it/

http://opensourcetesting.org/. (n.d.). Ανάκτηση από http://opensourcetesting.org/: http://opensourcetesting.org/

ISTQB. (2013). *ISTQB Exam Certification*. Retrieved October 11, 2013, from http://istqbexamcertification.com/: http://istqbexamcertification.com/what-is-prototype-model-advantages-disadvantages-and-when-to-use-it/

Justin. (9. July 2013). *http://airbrake.io/*. Hentede 4. October 2013 fra Airbrake: http://airbrake.io/blog/insight/what-is-the-software-development-life-cycle/

Kekare, H. (2009, September 20). *Types of Software Testing*. Ανάκτηση October 21, 21, από buzzle: http://www.buzzle.com/articles/types-of-software-testing.html

Malhotra, P. (n.d.). *www.aspiresys.com/WhitePapers/QTPvsSelenium.pdf*. Ανάκτηση από www.aspiresys.com: http://www.aspiresys.com/WhitePapers/QTPvsSelenium.pdf

Rouse, M. (2007, 04). *searchsoftwarequality.techtarget.com/definition/3-tier-application*. Ανάκτηση 03 12, 2014, από searchsoftwarequality.techtarget.com: http://searchsoftwarequality.techtarget.com/definition/3-tier-application

Sethu, S. (2012, 3 19). */www.thoughtworks.com/insights/articles/guide-test-automation*. Ανάκτηση 3 23, 2014, από /www.thoughtworks.com: http://www.thoughtworks.com/insights/articles/guide-test-automation

Singh, S. (2012, 05 05). *webdriverselenium.blogspot.dk/2012/05/difference-between-selenium-ide.html*. Ανάκτηση 03 25, 2014, από webdriverselenium.blogspot.dk: http://webdriverselenium.blogspot.dk/2012/05/difference-between-selenium-ide.html

softwareandme.wordpress.com. (n.d.). Ανάκτηση από sdlc_v_model: http://softwareandme.wordpress.com/2009/10/20/software-development-life-cycle/sdlc_v_model/

software-development-life-cycle-sdlc-system-development. (n.d.). Retrieved from www.mks.com: http://www.mks.com/solutions/discipline/software-development-life-cycle-sdlc-system-development

softwaretestinghelp. (2013). *softwaretestinghelp*. Ανάκτηση 03 25, 2014, από softwaretestinghelp.com: http://www.softwaretestinghelp.com/choosing-automation-tool-for-your-organization/

softwaretestinghelp. (n.d.). *softwaretestinghelp.com/choosing-automation-tool-for-your-organization/*. Ανάκτηση από softwaretestinghelp.com: http://www.softwaretestinghelp.com/choosing-automation-tool-for-your-organization/

STLC - Software Testing Life Cycle. (2013). Ανάκτηση october 22, 2013, από Quality Testified: http://qualitytestified.blogspot.dk/2013/02/software-testing-life-cycle.html

Tintin. (2013, May 20). *buzzle.com*. Retrieved September 29, 2013, from buzzle.com/Software Testing Methodologies: http://www.buzzle.com/articles/software-testing-methodologies.html

tutorialspoint. (2013). *SDLC V-Model*. Retrieved october 12, 12, from tutorialspoint: http://www.tutorialspoint.com/sdlc/sdlc_v_model.htm

tutorialspoint.com. (n.d.). *tutorialspoint.com/uml/uml_class_diagram.htm*. Ανάκτηση από tutorialspoint.com: http://www.tutorialspoint.com/uml/uml_class_diagram.htm

V Model. (2012, May 29). Retrieved October 12, 2013, from softwaretestingclass: http://www.softwaretestingclass.com/v-model/

versionone. (n.d.). *versionone*. Ανάκτηση October 20, 2013, από Agile Methodologies for Software Development: http://www.versionone.com/Agile101/Agile-Development-Methodologies-Scrum-Kanban-Lean-XP/

Wright, M. (2013, 9 20). *nitrosphere.com/2013/09/20/2-tier-vs-3-tier-application-architecture-could-the-winner-be-2-tier-2/*. Ανάκτηση 03 12, 2014, από nitrosphere.com: http://www.nitrosphere.com/2013/09/20/2-tier-vs-3-tier-application-architecture-could-the-winner-be-2-tier-2/