Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ιστοσελίδα καταγραφής δεδομένων χρηστών και παραγωγής συστάσεων ιατρικής φύσεως.** **Website for user data registration and production of medical recommendations.**
Ονοματεπώνυμο Φοιτητή	**Χαζιζάι Φλαντίτα**
Πατρώνυμο	**Αντέμ**
Αριθμός Μητρώου	**ΜΠΠΛ/ 11026**
Επιβλέπων	**Ευθύμιος Αλέπης, Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **Νοέμβριος 2015**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα
Ευθύμιος Αλέπης **Επίκουρος Καθηγητής**	**Μαρία Βίρβου** **Καθηγήτρια**	**Γεώργιος Τσιχριντζής** **Καθηγητής**

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΕΡΙΛΗΨΗ

Είναι πλέον κοινά αποδεκτό ότι ο όγκος της πληροφορίας που σχετίζεται με την ιατρική φροντίδα ενός ασθενούς έχει αυξηθεί κατά πολύ τα τελευταία χρόνια. Οι κλασικοί χειρόγραφοι φάκελοι ασθενών, που βασίζονται στην καταγραφή των δεδομένων του ασθενούς σε χαρτί, αδυνατούν να συγκρατήσουν τον μεγάλο όγκο των πληροφοριών, με αποτέλεσμα την απώλεια ιατρικών δεδομένων και την αδυναμία διατήρησης της χρονικής συσχέτισης των διαφόρων εξετάσεων με το ιστορικό του ασθενούς. Έτσι προκύπτει η ανάγκη δημιουργίας και διατήρησης ηλεκτρονικών φακέλων, οι οποίοι θα έχουν συγκεντρωμένες όλες τις ιατρικές πληροφορίες του ασθενούς, με σκοπό την παροχή καλύτερης και ποιοτικότερης ιατροφαρμακευτικής περίθαλψης.

Ο ηλεκτρονικός ιατρικός φάκελος του ασθενούς είναι ένας ψηφιακά αποθηκευμένος φάκελος με σκοπό να υποστηριχτεί η φροντίδα υγείας του ατόμου εφ' όρου ζωής, προωθεί την έρευνα και την εκπαίδευση των επαγγελματιών υγείας και βοηθά στην πρόσβαση και στο διαμοιρασμό πληροφοριών στους επαγγελματίες υγείας με φιλικό τρόπο καθώς ελέγχεται και η ασφάλεια των δεδομένων.

Τα δεδομένα είναι τα κλασσικά δεδομένα που εισάγονται σε ένα χειρόγραφο ιατρικό φάκελο. Έτσι, υπάρχουν δημογραφικά δεδομένα του ασθενούς, τα αποτελέσματα των εξετάσεων, οι ιατρικές οδηγίες και η φαρμακευτική αγωγή καθώς επίσης αποθηκεύονται και ακτινογραφίες ή αξονικές ή άλλες αντίστοιχες εξετάσεις. Το τελευταίο συστατικό στοιχείο του, το ανθρώπινο δυναμικό είναι το ιατρικό, το παραϊατρικό, το διοικητικό και το γραμματειακό προσωπικό. Ακόμα σε αυτήν την κατηγορία εντάσσονται οι υπεύθυνοι του συστήματος, δηλαδή αυτοί που το υλοποιούν το και το συντηρούν όπως αναλυτές, προγραμματιστές και τεχνικοί υλικού.

Στην σύγχρονη εποχή θα έπρεπε να δίνεται η δυνατότητα στον ίδιο τον ασθενή, εφόσον επιθυμεί, να μπορεί να έχει πρόσβαση σε κάποιο λογισμικό για να δημιουργήσει τον δικό του προσωπικό ιατρικό φάκελο. Στον φάκελο αυτό θα μπορεί να κρατάει κάποιο ιστορικό των εξετάσεων του για να μπορεί οποιαδήποτε στιγμή να κάνει σύγκριση τιμών των παλαιοτέρων τιμών με τις καινούριες για να μπορεί να ελέγξει ανά πάσα στιγμή εάν έχει καλυτέρευση ή χειροτερεύσει η υγεία του.

Θα μπορεί επίσης να πάρει κάποιες πρώτες συμβουλές για το τι θα πρέπει να τρώει για να μπορέσει να καλυτερεύσει την υγεία του, πράγματα που πρέπει να κάνει στην καθημερινότητα του και πράγματα που θα πρέπει να αποφύγει. Θα παρέχεται επίσης λεπτομερή ανάλυση υγείας που θα ενημερώνει τον χρήστη για πιθανά προβλήματα υγείας που μπορεί να δημιουργηθούν αναλόγως με τα αποτελέσματα των εξετάσεων, όπως επίσης και για ποιόν λόγο προέκυψαν οι συγκεκριμένες τιμές.

ΕΙΣΑΓΩΓΗ

Η διαδικασία ανάπτυξης του πληροφοριακού συστήματος στη παρούσα εργασία θα πραγματοποιηθεί χρησιμοποιώντας την μεθοδολογία Rational Unified Process. Συγκεκριμένα, η δομή της εργασίας έχει ως εξής:

Στο **πρώτο κεφάλαιο** θα ασχοληθούμε με την ανάλυση των απαιτήσεων. Θα αναφερθούμε στην περιοχή γύρω από την οποία πραγματοποιήθηκε η έρευνα για την υλοποίηση της ιστοσελίδας. Θα αναφερθούμε στο πρόβλημα το οποίο έχει προκύψει και θα πρέπει να αντιμετωπίσουμε με την υλοποίηση του συγκεκριμένου λογισμικού. Τέλος, θα αναφερθούμε στο σκοπό τον οποίο καλείται να πετύχει η ιστοσελίδα καθώς και τους αντικειμενικούς στόχους.

Στο **δεύτερο κεφάλαιο** θα ασχοληθούμε με την ανάπτυξη των φάσεων του αντικειμενοστρεφούς μοντέλου ανάπτυξης λογισμικού (Rational Unified Process) χρησιμοποιώντας διαγράμματα της UML. Θα γίνει χρήση των εξής διαγραμμάτων: Διαγράμματα Περίπτωσης Χρήσης, Διαγράμματα Τάξεων, Διαγράμματα συνεργασίας, Διαγράμματα Σειράς και Διαγράμματα Δραστηριοτήτων.

Στο **τρίτο κεφάλαιο** της εργασίας θα παρουσιάσουμε την υλοποίηση του πληροφοριακού συστήματος, την υλοποίηση της βάσης δεδομένων, των πινάκων, της σχεσής μεταξύ τους και των πεδίων που θα περιέχουν. Θα παρουσιάσουμε επίσης και το Εγχειρίδιο Χρήστη με όλες τις σελίδες της ιστοσελίδας. Για κάθε σελίδα θα γίνεται μια περιγραφή για τις ενέργειες που μπορεί να πραγματοποιήσει ο χρήστης.

Και τέλος στο τέταρτο και **τελευταίο κεφάλαιο** θα ασχοληθούμε με τον έλεγχο της εφαρμογής. Θα αναπτυχθεί αναλυτικά η διαδικασία ελέγχου, οι μέθοδοι ελέγχου, θα γραφτούν τα σενάρια ελέγχου για το συγκεκριμένο λογισμικό και θα τρεχτούν όλα τα σενάρια είτε χειρωνακτικά, είτε αυτοματοποιημένα αναλόγως με την περίπτωση. Θα παρουσιάσουμε αναλυτικά τον τρόπο με τον οποίο θα πραγματοποιηθεί ο αυτοματοποιημένος έλεγχος, θα παρουσιάσουμε τον τρόπο με τον οποίο θα πρέπει να στηθεί το περιβάλλον και ποιά προγράμματα πρέπει να εγκατασταθούν. Στη συνέχεια θα παρουσιάσουμε όλα τα scripts, όπως αυτά προκύψανε για την κάθε αυτοματοποιημένη περίπτωση ελέγχου. Τέλος θα παρουσιάσουμε τα αποτελέσματα της πρώτης και της δεύτερης εκτέλεσης.

Κεφάλαιο 1ο - Σύλληψη απαιτήσεων

1.1 Ερευνητική περιοχή

Η Ιατρική Πληροφορική (Healthcare Informatics) είναι το σημείο τομής της πληροφορικής, της επιστήμης των υπολογιστών, και της υγειονομικής περίθαλψης. Συνδυάζει τους τομείς της πληροφορικής και της υγείας για την ανάπτυξη των συστημάτων που απαιτούνται για την διαχείριση των αυξανόμενων πληροφοριών, των πολύπλοκων κλινικών διαδικασιών και την βελτίωση της ασφάλειας του συστήματος υγείας. Ασχολείται με τους πόρους, τις συσκευές και τις μεθόδους που απαιτούνται για τη βελτίωση της αποθήκευσης, της ανάκτησης και της χρήσης των πληροφοριών στον τομέα της υγείας και της βιοϊατρικής.

Τα εργαλεία της Ιατρικής Πληροφορικής δεν περιλαμβάνουν μόνο υπολογιστές αλλά και κλινικές κατευθυντήριες γραμμές, τυποποιημένες ιατρικές ορολογίες, καθώς και συστήματα επικοινωνίας. Εφαρμόζεται στους τομείς της νοσηλευτικής, της κλινικής φροντίδας, της οδοντιατρικής, της φαρμακευτικής, της δημόσιας υγείας και της (βιο) ιατρικής έρευνας.

Η ιατρική πληροφορική χρησιμοποιεί υπολογιστές, εξειδικευμένο λογισμικό και συσκευές επικοινωνίας διαμορφώνοντας ένα πολύπλοκο δίκτυο, με σκοπό την συλλογή, την ανάλυση και τη διαβίβαση ιατρικών διαδικασιών. Τα εργαλεία για τη δημιουργία συστημάτων πληροφορικής για την υγεία δεν περιορίζονται μόνο στην τεχνολογία της πληροφορικής. Τα συστήματα αυτά θα πρέπει, επίσης, να επιτρέπουν την αφομοίωση των κλινικών οδηγιών, την κατανόηση της επίσημης ιατρικής ορολογίας, την αποθήκευση των δεδομένων και την σαφή επικοινωνία. Η ιατρική πληροφορική μπορεί να εφαρμοστεί σε όλα τα είδη των περιβαλλόντων της υγείας, συμπεριλαμβανομένης της πρωτοβάθμιας φροντίδας, τη γενική ιατρική, τη νοσοκομειακή περίθαλψη και την αποκατάσταση.

Ένας από τους βασικούς στόχους της ιατρικής πληροφορικής είναι να διαμορφώσει μια τυποποιημένη προσέγγιση για την υγειονομική περίθαλψη σε διεθνές επίπεδο. Η ιδέα είναι να επωφεληθούν οι ερευνητές, οι πάροχοι, και οι ασθενείς από τα εργαλεία πληροφορικής, τις τεχνικές, τις ιδέες και τα πρωτόκολλα που αναδιαμορφώνουν την παροχή της υγειονομικής περίθαλψης και να προωθηθούν οι βέλτιστες πρακτικές σε αυτόν τον τομέα. Πολλοί ειδικοί συμφωνούν ότι μία από τις κύριες προκλήσεις της ιατρικής πληροφορικής είναι να πεισθούν οι πάροχοι της ιατρικής φροντίδας, ώστε να αποδεχθούν και να χρησιμοποιήσουν την συγκεκριμένη τεχνολογία. Μόνο τότε μπορεί να δημιουργηθεί ένα ομοιογενές και ολοκληρωμένο σύστημα υγειονομικής περίθαλψης, το οποίο θα βοηθήσει τους επαγγελματίες του κλάδου στην επίλυση προβλημάτων, τη λήψη αποφάσεων, την παροχή καλύτερων υπηρεσιών στους ασθενείς και την εκτέλεση των καθηκόντων τους πιο αποτελεσματικά και αποδοτικά.

1.2 Πρόβλημα προς αντιμετώπιση

Με τη ραγδαία ανάπτυξη της τεχνολογίας της πληροφορικής τις τελευταίες δεκαετίες και ειδικότερα των τεχνολογιών για την ιατροφαρμακευτική περίθαλψη, έχουν προταθεί πολλά και διαφορετικά μοντέλα Ηλεκτρονικού Φακέλου Υγείας. Σε πολλές χώρες, μάλιστα, έχει προγραμματιστεί η εισαγωγή ενός ηλεκτρονικού αρχείου υγείας σε εθνικό επίπεδο, ενώ σε άλλες έχει εφαρμοστεί σε κάποια μορφή του. Παρ' όλα αυτά, υπάρχει ποικιλομορφία ως προς τον τύπο, τη δομή και την υλοποίηση των αρχείων αυτών. Σε μερικές περιπτώσεις μπορεί να είναι ένας φάκελος διαθέσιμος ευρέως σε διάφορα ιδρύματα, ενώ σε άλλες μπορεί να αποτελεί ένα περιορισμένο σύστημα, το οποίο είναι διαθέσιμο μόνο σε μια συγκεκριμένη μονάδα ή τμήμα.

Η δημιουργία ενός ιατρικού φακέλου στον οποίο να έχει πρόσβαση άμεσα ο ασθενείς είναι πλέον μια ανάγκη η οποία θα διευκολύνει κατά πολύ την καθημερινότητα των ασθενών. Ο ασθενής έχει την ανάγκη να μπορεί να έχει έναν ηλεκτρονικό ιατρικό φάκελο για να μπορεί να καταχωρεί τις εξετάσεις, τις οποίες κάνει κατά καιρούς και να μπορεί να να κάνει σύγκριση των παλαιοτέρων τιμών με τις καινούριες για να μπορεί να ελέγξει ανά πάσα στιγμή εάν έχει καλυτερεύσει ή χειροτερεύσει η υγεία του.

1.3 Σκοπός και αντικειμενικοί στόχοι

Στην παρούσα διπλωματική εργασία θα ασχοληθούμε με τα Πληροφοριακά Συστήματα Υγείας και συγκεκριμένα με την ανάπτυξη μιας ιστοσελίδας - ιατρικού φακέλου ασθενούς, η οποία θα δέχεται τα αποτελέσματα των εξετάσεων από τους χρήστες και στη συνέχεια θα τους δίνει κάποιες συμβουλές για το πως μπορούν να βελτιώσουν τη φυσική τους κατάσταση ή/και την υγεία τους.

Σκοπός είναι ο κάθε χρήστης να μπορεί ανά πάσα στιγμή να μπορεί να ενημερωθεί αναλυτικά για την κατάσταση της υγείας του σύμφωνα με τα αποτελέσματα των εξετάσεων του, οι οποίες μπορεί να είναι: αιματολογικές, βιοχημικές και ορμονολογικές.

Ο ασθενής θα έχει πρόσβαση σε κάποιο λογισμικό για να δημιουργήσει τον δικό του προσωπικό ιατρικό φάκελο. Στον φάκελο αυτό θα μπορεί να κρατάει κάποιο ιστορικό των εξετάσεων του για να μπορεί οποιαδήποτε στιγμή να κάνει σύγκριση των παλαιοτέρων τιμών με τις καινούριες για να μπορεί να ελέγξει ανά πάσα στιγμή εάν έχει καλυτερεύσει ή χειροτερεύσει η υγεία του.

Ο ασθενής θα μπορεί να πάρει κάποιες πρώτες συμβουλές για το τι θα πρέπει να τρώει για να μπορέσει να καλυτερεύσει την υγεία του, πράγματα που πρέπει να κάνει στην καθημερινότητα του και πράγματα που θα πρέπει να αποφύγει. Στη συνέχεια θα παρέχεται και λεπτομερή ανάλυση υγείας που θα ενημερώνει τον χρήστη για πιθανά προβλήματα.

1.4 Ανάπτυξη με τη χρήση της Rational Unified Process

Η ανάπτυξη λογισμικού για ένα πληροφοριακό σύστημα μπορεί να γίνει με πολλούς τρόπους και μεθοδολογίες ωστόσο τα τελευταία χρόνια η μεθοδολογία Rational Unified Process (RUP) της IBM έχει γίνει πολύ δημοφιλής. Αυτό συνέβη επειδή η RUP αναιρεί την παραδοσιακή προσέγγιση μεθόδων όπως το μοντέλο καταρράκτη ή το V-model και δίνει μία οπτική παράλληλης ανάλυσης, σχεδίασης, ανάπτυξης και δοκιμών ώστε οποιοδήποτε προβλήματα και παρανοήσεις να διαπιστώνονται έγκαιρα και να ενσωματώνονται με το λιγότερο δυνατό κόστος. Συνοπτικά οι βασικές ροές εργασίες αυτού του μοντέλου είναι οι ακόλουθες:

- Μοντελοποίηση επιχειρησιακού περιβάλλοντος (Business Modeling)
- Συγγραφή προδιαγραφών (Requirements)
- Ανάλυση και σχεδίαση (Analysis and Design)
- Υλοποίηση (Implementation)
- Έλεγχος (Test)
- Εγκατάσταση

Η διαδικασία RUP είναι δομημένη σε δύο διαστάσεις : τον χρόνο, όπου ο κύκλος ζωής χωρίζεται σε φάσεις και επαναλήψεις, και τα τμήματα διαδικασίας, όπου γίνεται ο ορισμός των ενεργειών του συστήματος.

Η δόμηση ενός έργου σε σχέση με το χρόνο ακολουθεί τις εξής φάσεις:

1. Έναρξη: Καθορίζεται η προοπτική του έργου.
2. Εκπόνηση μελέτης: Σε αυτή τη φάση σχεδιάζονται οι απαιτούμενες δραστηριότητες και πόροι του συστήματος ενώ επιπλέον καθορίζονται τα χαρακτηριστικά του συστήματος και σχεδιάζεται η αρχιτεκτονική του.

3. Κατασκευή: Σε αυτή τη φάση γίνεται η ανάπτυξη του προϊόντος σε μια σειρά βηματικών επαναλήψεων.

4. Μετάβαση: Στη τελική φάση η κοινότητα των χρηστών (παραγωγή, διανομή, εκπαίδευση) προμηθεύεται το προϊόν.

Η δόμηση έργου σύμφωνα με τη διάσταση των τμημάτων διαδικασίας περιλαμβάνει τις ακόλουθες δραστηριότητες :

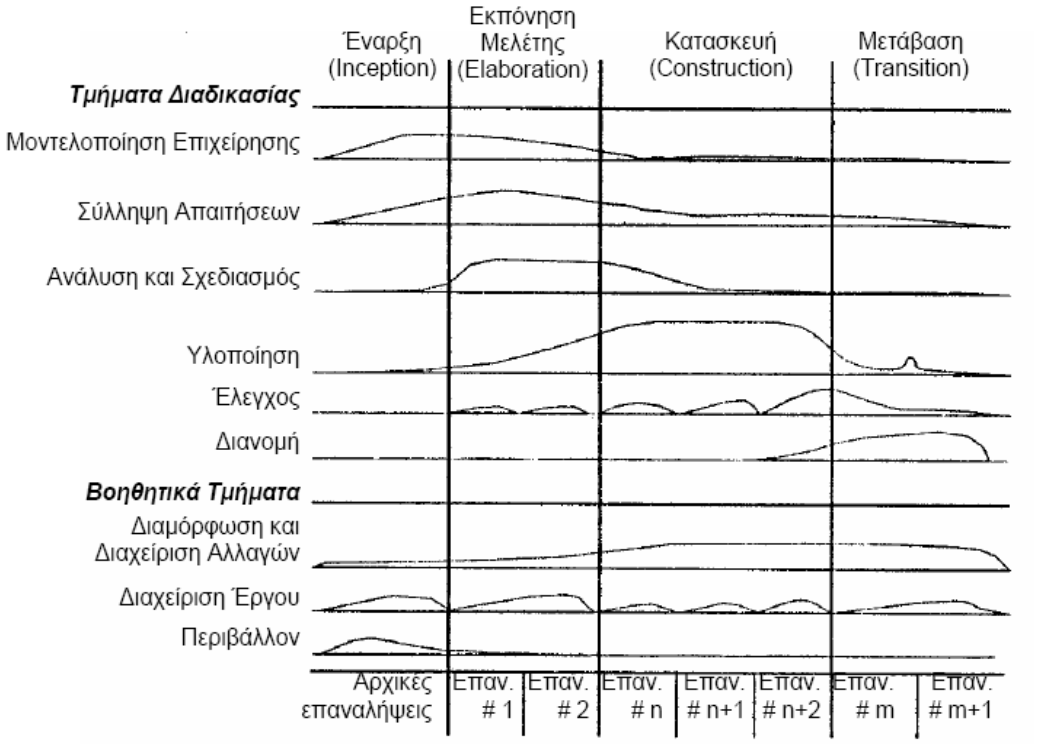
1. Σύλληψη απαιτήσεων (Requirements capture) : Μια αφήγηση του τι πρέπει να κάνει το σύστημα.

2. Ανάλυση και σχεδιασμός (Analysis and design) : Μια περιγραφή του πώς θα υλοποιηθεί το σύστημα.

3. Υλοποίηση (Implementation) : Η παραγωγή του κώδικα.

4. Έλεγχος (Test) : Η επαλήθευση του συστήματος.

Οι παραπάνω φάσεις του κύκλου ζωής ανάπτυξης του λογισμικού απεικονίζονται στην Εικόνα 1.



Εικόνα 1: Κύκλος Ζωής Ανάπτυξης Λογισμικού

Η Unified modeling language (UML) είναι μια είναι γλώσσα μοντελοποίησης της μεθοδολογίας RUP. Συγκεκριμένα είναι μία γραφική γλώσσα για την οπτική παράσταση, τη διαμόρφωση προδιαγραφών και την τεκμηρίωση συστημάτων που βασίζονται σε λογισμικό. Η UML

στοχεύει στο σχεδιασμό αντικειμενοστραφών συστημάτων. Το σχέδιο είναι μια απλοποιημένη παράσταση της πραγματικότητας.

Η UML είναι γλώσσα μοντελοποίησης και όχι μεθοδολογία, δηλ. η UML ορίζει μόνο τη γλώσσα μοντελοποίησης και όχι τη διαδικασία. Η UML ορίζει δύο εργαλεία: έναν συμβολισμό και ένα μεταμοντέλο. Ο συμβολισμός ορίζει τα 9 διαγράμματα της UML που θα χρησιμοποιηθούν για την ανάπτυξη των 4 φάσεων του αντικειμενοστραφούς μοντέλου κύκλου ζωής λογισμικού του νοσοκομείου. Τα διαγράμματα αυτά είναι τα ακόλουθα:

- Διαγράμματα τάξεων (Class Diagrams)
- Διαγράμματα περιπτώσεων χρήσης (Use case diagrams)
- Διαγράμματα αντικειμένων (Object Diagrams)
- Διαγράμματα αλληλεπίδρασης, τα οποία αποτελούνται από:
    - Διαγράμματα συνεργασίας (Collaboration Diagrams)
    - Διαγράμματα σειράς (Sequence diagrams)
- Διαγράμματα καταστάσεων (Statechart diagrams)
- Διαγράμματα δραστηριοτήτων (Activity diagrams)
- Διαγράμματα εξαρτημάτων (Component diagrams)
- Διαγράμματα διανομής (Deployment diagrams)

1.4.1 Φάση έναρξης

Η φάση έναρξης είναι αυτή όπου καθορίζεται η προοπτική του έργου. Εδώ γίνεται η σύλληψη των απαιτήσεων για το τι θα κάνει το σύστημα που θα αναπτυχθεί. Όλα τα έργα ανάπτυξης λογισμικού ξεκινούν από τον ορισμό τους. Ανεξάρτητα από τον τρόπο με τον οποίο ο ορισμός αυτός αναφέρεται στις διάφορες μεθοδολογίες, στα μοντέλα κύκλου ζωής και στα πρότυπα που χρησιμοποιούνται στην ανάπτυξη του λογισμικού, το βασικό στοιχείο είναι να γίνει μία καταγραφή των αναγκών των χρηστών σχετικά με την λειτουργικότητα που θα εξυπηρετεί το σύστημα.

1.4.2 Σύλληψη απαιτήσεων

Κάνοντας μια έρευνα σχετικά με τις ανάγκες του χρήστη για την συγκεκριμένη ιστοσελίδα, καταγράψαμε πληροφορίες οι οποίες μας βοήθησαν να καταλάβουμε σε βάθος την δομή που θα πρέπει να έχει η εφαρμογή, σε ποιούς χρήστες θα αναφέρεται και ποιές θα είναι οι ενέργειες οι οποίες θα μπορούν να πραγματοποιούν μέσα από την σελίδα αυτή. Τις ενέργειες αυτές θα καταγράψουμε παρακάτω.

Χρήστης Ιστοσελίδας:

	Actor	**Use Case**
1	Χρήστης	Επισκέπτεται την ιστοσελίδα
2	Χρήστης	Εγγραφή στο σύστημα

3	Χρήστης	Σύνδεση στο σύστημα
4	Χρήστης	Καταχώρηση Εξετάσεων
5	Χρήστης	Αναζήτηση Εξετάσεων
6	Χρήστης	Ενημέρωση Αποτελεσμάτων Εξετάσεων
7	Χρήστης	Ενημέρωση Ιατρικών Συμβουλών

- Ο Χρήστης θα μπορεί να επισκεφτεί την ιστοσελίδα και να κάνει απλά ανάγνωση των όρων χρήσης, χωρίς να προβεί σε κάποια άλλη ενέργεια.
- Ο Χρήστης θα μπορεί να χρησιμοποιεί την ιστοσελίδα προκειμένου να εγγραφεί στο σύστημα δίνοντας τα προσωπικά του στοιχεία, τα οποία είναι το όνομά του, το επώνυμό του, το φύλλο του, το email του και το password, για να μπορεί να έχει έναν λογαριασμό με τον οποίο θα είναι σε θέση να καταχωρεί εξετάσεις.
- Ο Χρήστης για να μπορέσει να προχωρήσει σε καταχώρηση εξετάσεων θα πρέπει πρώτα να συνδεθεί στο σύστημα καταχωρώντας το e-mail και το password.
- Ο Χρήστης αφού έχει εγγραφεί και συνδεθεί στο σύστημα θα μπορεί πλέον να κάνει καταχώρηση των εξετάσεων του. Οι εξετάσεις είναι χωρισμένες σε τρεις κατηγορίες, σε βιοχημικές, αιματολογικές και ορμονολογικές.
- Ο Χρήστης αφού κάνει επιτυχής καταγραφή των εξετάσεων του, θα του δίνεται η δυνατότητα να κάνει αναζήτηση αυτών, μέσω δύο τρόπων αναζήτησης, Ο ένας τρόπος θα είναι βάσει της ημερομηνίας καταχώρησης των εξετάσεων και ο δεύτερος τρόπος θα είναι μέσω της κατηγορίας των εξετάσεων.
- Ο Χρήστης στη συνέχεια, αφού έχει επιλέξει ένα κριτήριο αναζήτησης, θα του εμφανίζεται μια σελίδα με τα αποτελέσματα των εξετάσεων του σε έναν πίνακα με τα εξής πεδία:

  - Ένα πεδίο στο οποίο θα εμφανίζεται η ημερομηνία καταχώρησης της εξέτασης.
  - Ένα πεδίο στο οποίο θα εμφανίζεται η ονομασία της εξέτασης.
  - Ένα πεδίο στο οποίο θα εμφανίζεται η τιμή της εξέτασης, την οποία έχει καταχωρήσει ο χρήστης.
  - Και τέλος ο χαρακτηρισμός των ορίων, όπου το σύστημα θα ενημερώνει τον χρήστη εάν οι τιμές των εξετάσεων του είναι φυσιολογικές, άνω του φυσιολογικού ή κάτω του φυσιολογικού. Στην περίπτωση που είναι άνω του φυσιολογικού ή κάτω του φυσιολογικού, θα παρέχεται δίπλα και ένας σύνδεσμος ο οποίος θα παραπέμπει τον χρήστη σε μία σελίδα όπου θα του δίνονται κάποιες συμβουλές για την βελτίωση της υγείας του.

- Τέλος ο χρήστης αφού έχει μεταβεί στην σελίδα για να μπορέσει να πάρει κάποιες πρώτες συμβουλές για την βελτίωση της υγείας του, θα βλέπει ένα πεδίο στο οποίο θα δίνεται μια περιγραφή της κατάστασης και από κάτω θα υπάρχει ένας πίνακας με τα προτεινόμενα είδη τροφίμων, με τους τρόπους αντιμετώπισης της ασθένειας και πράγματα που θα πρέπει να αποφύγει να κάνει.

Διαχειριστής Ιστοσελίδας:

	Actor	**Use Case**
1	Διαχειριστής	Εισάγει/ενημερώνει είδη τροφίμων
2	Διαχειριστής	Εισάγει/ενημερώνει πράγματα που πρέπει να αποφευχθούν.
3	Διαχειριστής	Συνδέει εξετάσεις με τρόφιμα, κτλ.
4	Διαχειριστής	Εισάγει/ενημερώνει Εξετάσεις
5	Διαχειριστής	Εισάγει/ενημερώνει τρόπους αντιμετώπισης ασθενειών.

- Ο διαχειριστής της ιστοσελίδας θα έχει πρόσβαση στην βάση δεδομένων της ιστοσελίδας ώστε να εισάγει/ενημερώνει είδη τροφίμων. Αυτό σημαίνει ότι θα μπορεί να εισάγει καινούρια τρόφιμα, τα οποία στη συνέχεια θα τα συνδέει με συγκεκριμένες εξετάσεις. Θα μπορεί επίσης να ενημερώνει τα ήδη υπάρχοντα τρόφιμα. Στα τρόφιμα προσθέτει τον κωδικό, την ονομασία και την κατηγορία τροφίμων.
- Ο διαχειριστής της ιστοσελίδας θα έχει πρόσβαση στην βάση δεδομένων της ιστοσελίδας ώστε να εισάγει/ενημερώνει πράγματα που πρέπει να αποφευχθούν από τους ασθενείς για συγκεκριμένες ασθένειες. Αυτό σημαίνει ότι θα μπορεί να εισάγει καινούρια πράγματα που πρέπει να αποφευχθούν, τα οποία στη συνέχεια θα τα συνδέει με συγκεκριμένες εξετάσεις. Θα μπορεί επίσης να ενημερώνει τα ήδη υπάρχοντα πράγματα που πρέπει να αποφευχθούν. Στα πράγματα που πρέπει να αποφευχθούν προσθέτει τον κωδικό και την ονομασία. Στη σύνδεση προσθέτει τον κωδικό της εξέτασης, τον κωδικό των τροφίμων, την στήλη για το εάν οι τιμές της εξέτασης είναι χαμηλότερες ή υψηλότερες του φυσιολογικού, τα πράγματα που πρέπει να αποφευχθούν, καθώς και τους τρόπους αντιμετώπισης ασθενειών.
- Ο διαχειριστής της ιστοσελίδας θα έχει πρόσβαση στην βάση δεδομένων της ιστοσελίδας ώστε να συνδέει εξετάσεις με είδη τροφίμων, με πράγματα που πρέπει να αποφευχθούν καθώς επίσης και με τρόπους αντιμετώπισης ασθενειών. Αυτό σημαίνει ότι θα μπορεί να εισάγει καινούριες συνδέσεις μεταξύ εξετάσεων και τροφίμων, πραγμάτων που πρέπει να αποφευχθούν και με τρόπων αντιμετώπισης για εγγραφές που υπάρχουν ήδη μέσα στο σύστημα ή για καινούριες εγγραφές.
- Ο διαχειριστής της ιστοσελίδας θα έχει πρόσβαση στην βάση δεδομένων της ιστοσελίδας ώστε να εισάγει/ενημερώνει είδη τροφίμων. Αυτό σημαίνει ότι θα μπορεί να εισάγει καινούρια τρόφιμα, τα οποία στη συνέχεια θα τα συνδέει με συγκεκριμένες εξετάσεις. Θα μπορεί επίσης να ενημερώνει τα ήδη υπάρχοντα τρόφιμα. Στις εξετάσεις προσθέτει τον κωδικό, την ονομασία και την κατηγορία εξετάσεων, την χαμηλότερη τιμή εντός ορίων, την υψηλότερη τιμή εντός ορίων, την περιγραφή για όταν οι τιμές είναι χαμηλότερες του φυσιολογικού και την περιγραφή για όταν οι τιμές είναι υψηλότερες του φυσιολογικού.
- Ο διαχειριστής της ιστοσελίδας θα έχει πρόσβαση στην βάση δεδομένων της ιστοσελίδας ώστε να εισάγει/ενημερώνει τους τρόπους αντιμετώπισης για συγκεκριμένες ασθένειες. Αυτό σημαίνει ότι θα μπορεί να εισάγει καινούριους τρόπους αντιμετώπισης ασθενειών, τους οποίους στη συνέχεια θα τους συνδέει με συγκεκριμένες εξετάσεις. Θα μπορεί επίσης να ενημερώνει τους ήδη υπάρχον τρόπους αντιμετώπισης ασθενειών. Στους τρόπους αντιμετώπισης ασθενειών προσθέτει τον κωδικό και την ονομασία.

Κεφάλαιο 2ο - Ανάλυση και Σχεδιασμός

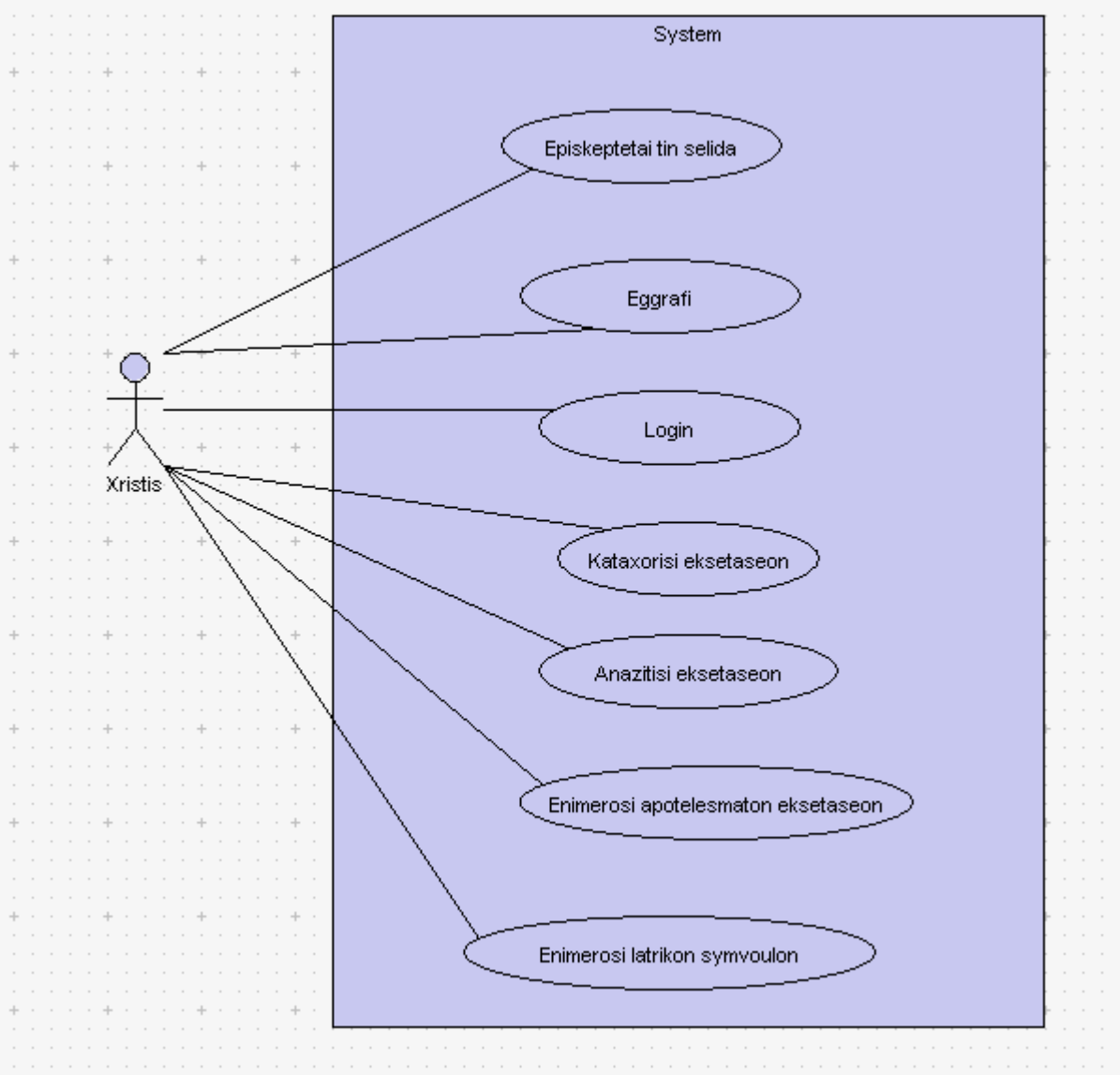
2.1 Διαγράμματα περιπτώσεων χρήσης

Η ικανοποίηση κάθε λειτουργικής απαίτησης από μία εφαρμογή λογισμικού υλοποιείται ως μια αλληλουχία ενεργειών που εκτελούνται από το λογισμικό, αλληλεπιδρώντας είτε με κάποιον χρήστη (φυσικό πρόσωπο), είτε με άλλα συστήματα (π.χ. άλλες εφαρμογές λογισμικού, εξωτερικές συσκευές, εξωτερικές πηγές δεδομένων). Μια τέτοια αλληλεπίδραση παράγει ένα αποτέλεσμα επιθυμητό για το χρήστη της εφαρμογής λογισμικού, δηλαδή ικανοποιεί μια λειτουργική απαίτησή του και ονομάζεται **περίπτωση χρήσης**.

Μια περίπτωση χρήσης χαρακτηρίζεται τόσο από την αλληλουχία των ενεργειών που εκτελεί το λογισμικό, όσο και από το μέρος εκείνο με το οποίο αλληλεπιδρά, δηλαδή ένα χρήστη – φυσικό πρόσωπο ή ένα εξωτερικό σύστημα. Το μέρος αυτό ονομάζεται **ενεργοποιός (actor)**. Κάθε περίπτωση χρήσης ενεργοποιείται από ένα χειριστή. Όταν εκτελούνται οι ενέργειες που περιλαμβάνονται στην περίπτωση χρήσης, τότε μπορούμε να λέμε ότι «εκτελείται η περίπτωση χρήσης».

Οι περιπτώσεις χρήσης (Use cases) του συστήματος αναπαρίστανται γραφικά παρακάτω.

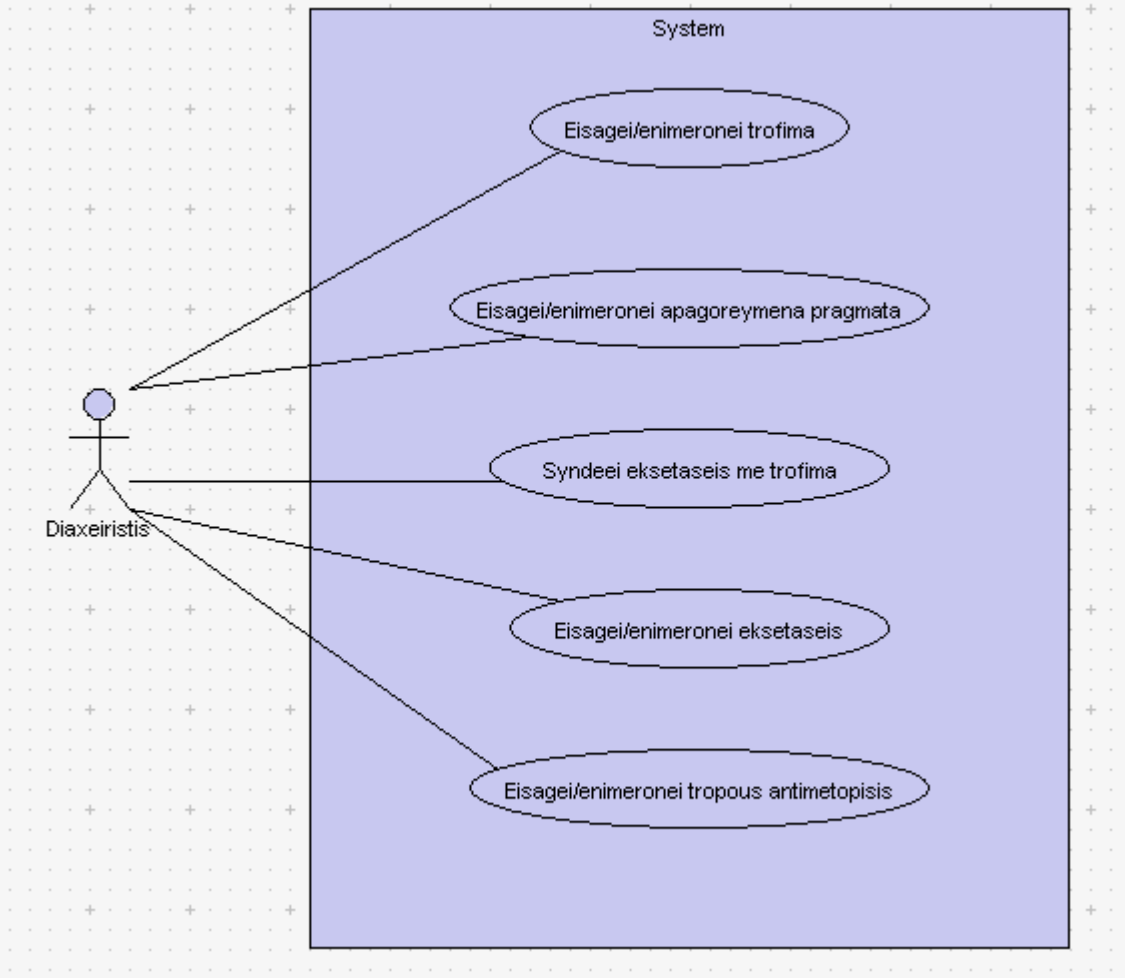
2.1.1 Use case Χρήστη



Διάγραμμα 1 - Διάγραμμα Περίπτωσης Χρήσης Χρήστη 1ης Έκδοσης

Στο παραπάνω διάγραμμα απεικονίζονται οι ενέργειες τις οποίες μπορεί να πραγματοποιήσει ο χρήστης που επισκέπτεται την ιστοσελίδα. Η πρώτη ενέργεια είναι απλώς να επιστεφτεί την ιστοσελίδα και να διαβάσει τους όρους χρήσης. Έπειτα, ο χρήστης μπορεί να επιλέξει να εγγραφεί στο σύστημα. Αφού έχει κάνει την εγγραφή του, ο χρήστης πλέον μπορεί να συνδεθεί στο σύστημα. Όταν πλέον είναι συνδεδεμένος, η ενέργεια η οποία μπορεί να πραγματοποιήσει είναι να καταχωρήσει τις εξετάσεις του. Στη συνέχεια ο χρήστης μπορεί να αναζητήσει τις εξετάσεις τις οποίες καταχώρησε προηγουμένως. Το σύστημα θα τον ενημερώσει σχετικά με τα αποτελέσματα των εξετάσεων του και τέλος ο χρήστης μπορεί να επιλέξει να λάβει ιατρικές συμβουλές σχετικά με τα αποτελέσματα των εξετάσεων του.

2.1.2 Use Case Διαχειριστή



Διάγραμμα 2 - Διάγραμμα Περίπτωσης Χρήσης Διαχειριστή 1ης Έκδοσης

Στο παραπάνω διάγραμμα απεικονίζονται οι ενέργειες τις οποίες μπορεί να πραγματοποιήσει ο διαχειριστή που διαχειρίζεται την ιστοσελίδα. Οι ενέργειες τις οποίες μπορεί να πραγματοποιήσει ο διαχειριστής είναι να ενημερώνει την βάση δεδομένων. Στη βάση δεδομένων μπορεί να εισάγει ή να ενημερώνει τα εξής: εισάγει ή ενημερώνει τα είδη τροφίμων, εισάγει ή ενημερώνει τα πράγματα τα οποία απαγορεύεται να κάνει ο ασθενής για κάποια συγκεκριμένη εξέταση, συνδέει τα είδη τροφίμων με τις αντίστοιχες εξετάσεις, εισάγει ή ενημερώνει τα είδη των εξετάσεων και τέλος εισάγει ή ενημερώνει τους τρόπους με τους οποίους μπορεί να αντιμετωπιστεί μια ασθένεια.

2.2 Διαγράμματα Τάξεων

Τα διαγράμματα τάξεων UML είναι το στήριγμα της αντικειμενοστραφούς ανάλυσης και σχεδιασμού. Τα διαγράμματα τάξεων UML παρουσιάζουν τις τάξεις του συστήματος, αλληλεξαρτήσεών τους (συμπεριλαμβανομένης της κληρονομιάς, της συνάθροισης, και της ένωσης), και των διαδικασιών και των ιδιοτήτων των τάξεων. Τα διαγράμματα τάξεων χρησιμοποιούνται για πολλούς σκοπούς, αλλά κυρίως για να διαμορφώνουν έναν λεπτομερή σχεδιασμό του συστήματος. Ένα διάγραμμα τάξης περιγράφει τους τύπους των αντικειμένων στο σύστημα και τα διάφορα είδη στατικών σχέσεων που υπάρχουν μεταξύ τους (αμφίδρομες και μονόδρομες συσχετίσεις, υπότυποι, πολλαπλότητα, κληρονομικότητα, κλπ.). Στα διαγράμματα αυτά αποτυπώνονται ακόμα οι ιδιότητες των τάξεων και οι λειτουργίες αυτών.

Οι τάξεις (classes) αποτελούν τη βάση της κατασκευής οποιουδήποτε αντικειμενοστραφούς συστήματος. Σκοπός του διαγράμματος τάξεων (class diagram) δεν είναι η απεικόνιση αυτών των αλληλεπιδράσεων, αλλά η απεικόνιση της στατικής δομής του συστήματος.

Μια τάξη σχεδιάζεται ως ένα ορθογώνιο με τρία τμήματα. Το πρώτο τμήμα περιέχει το όνομα της τάξης, το δεύτερο περιέχει τα χαρακτηριστικά της τάξης και το τρίτο της λειτουργίες της. Στο πληροφοριακού σύστημα της ιστοσελίδας μας, με βάση τις απαιτήσεις που συλλέξαμε νωρίτερα, έχουμε θεωρήσει τις ακόλουθες τάξεις:

Users : Είναι η τάξη που αναφέρεται στους χρήστες της ιστοσελίδας.

Exams: Είναι η τάξη που αναφέρεται στις εξετάσεις που είναι καταχωρημένες στο σύστημα και τις οποίες μπορεί να χρησιμοποιεί ο χρήστης για να εισάγει τις εξετάσεις του.

Make exams: Είναι η τάξη που αναφέρεται στην καταχωρημένες εξετάσεις από τον χρήστη.

Suggested: Είναι η τάξη που αναφέρεται στα προτεινόμενα τρόφιμα και άλλα προτεινόμενα πράγματα στον χρήστη, ανάλογα με τα αποτελέσματα των εξετάσεων του.

Foods: Είναι η τάξη που αναφέρεται στα τρόφιμα, τα οποία είναι αποθηκευμένα στη βάση δεδομένων και προτείνονται στον χρήστη ανάλογα με τα αποτελέσματα των εξετάσεων του.

Things to do : Είναι η τάξη που αναφέρεται στα πράγματα τα οποία πρέπει να κάνει ο χρήστης, τα οποία είναι αποθηκευμένα στη βάση δεδομένων και προτείνονται στον χρήστη ανάλογα με τα αποτελέσματα των εξετάσεων του.

Prohibited: Είναι η τάξη που αναφέρεται στα πράγματα τα οποία πρέπει να αποφύγει ο χρήστης, τα οποία είναι αποθηκευμένα στη βάση δεδομένων και προτείνονται στον χρήστη ανάλογα με τα αποτελέσματα των εξετάσεων του.

Οι παραπάνω τάξεις δεν λειτουργούν ανεξάρτητα η μία από την άλλη αλλά αντίθετα αλληλεπιδρούν μέσω σχέσεων. Ακολούθως περιγράφουμε ποιες είναι οι συσχετίσεις μεταξύ των τάξεων.

Ο χρήστης σχετίζεται με της εξετάσεις με σχέση 1-N (ένα προς πολλά), καθώς ένας χρήστης μπορεί να καταχωρήσει πολλές εξετάσεις

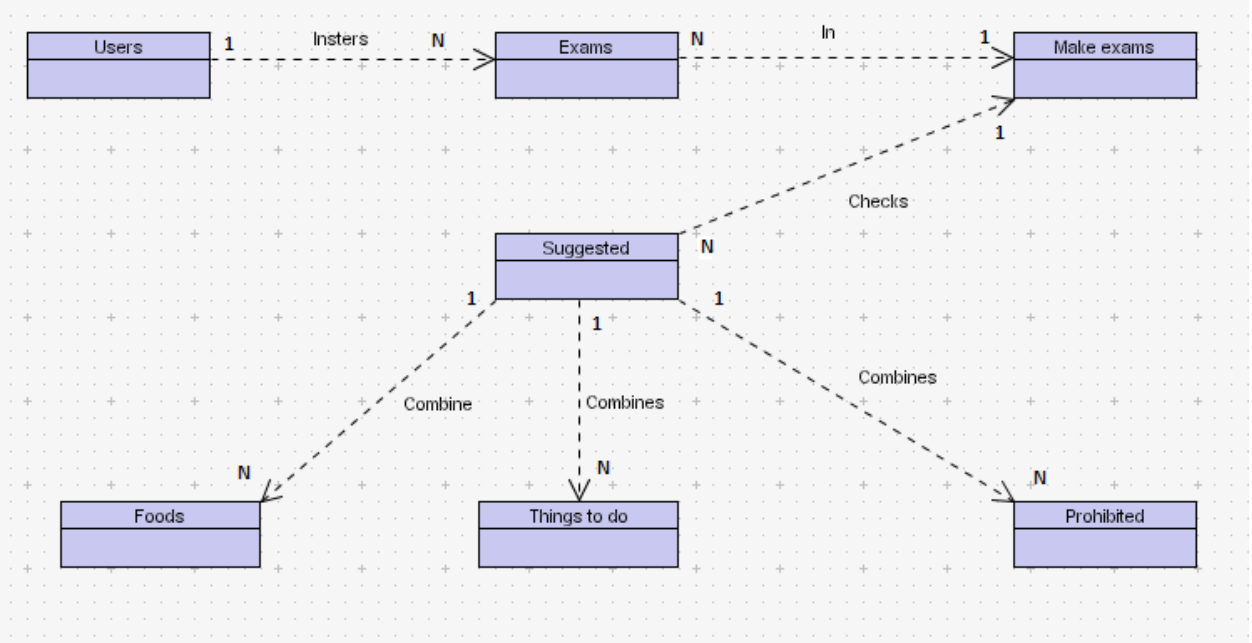
Οι εξετάσεις με το make exams σχετίζεται με σχέση N-N (πολλά προς πολλά), καθώς πολλές εξετάσεις μπορεί να λαμβάνουν χώρα σε μια καταχώρηση αλλά και μια καταχώρηση μπορεί να περιλαμβάνει πολλές εξετάσεις.

Το make exams σχετίζεται με το suggested με σχέση 1-1 (ένα προς πολλά), καθώς μια καταχώρηση εξετάσεων μπορεί να περιλαμβάνει πολλές ιατρικές συμβουλές ανάλογα με τα αποτελέσματα των εξετάσεων.

Το suggested σχετίζεται με το foods με σχέση 1-N (ένα προς πολλά), καθώς μια συμβουλή μπορεί να περιλαμβάνει πολλά τρόφιμα.

Το suggested σχετίζεται με το things to do με σχέση 1-N (ένα προς πολλά), καθώς μια συμβουλή μπορεί να περιλαμβάνει πολλά πράγματα που πρέπει να κάνει ο ασθενείς για να αντιμετωπίσει την ασθένεια.

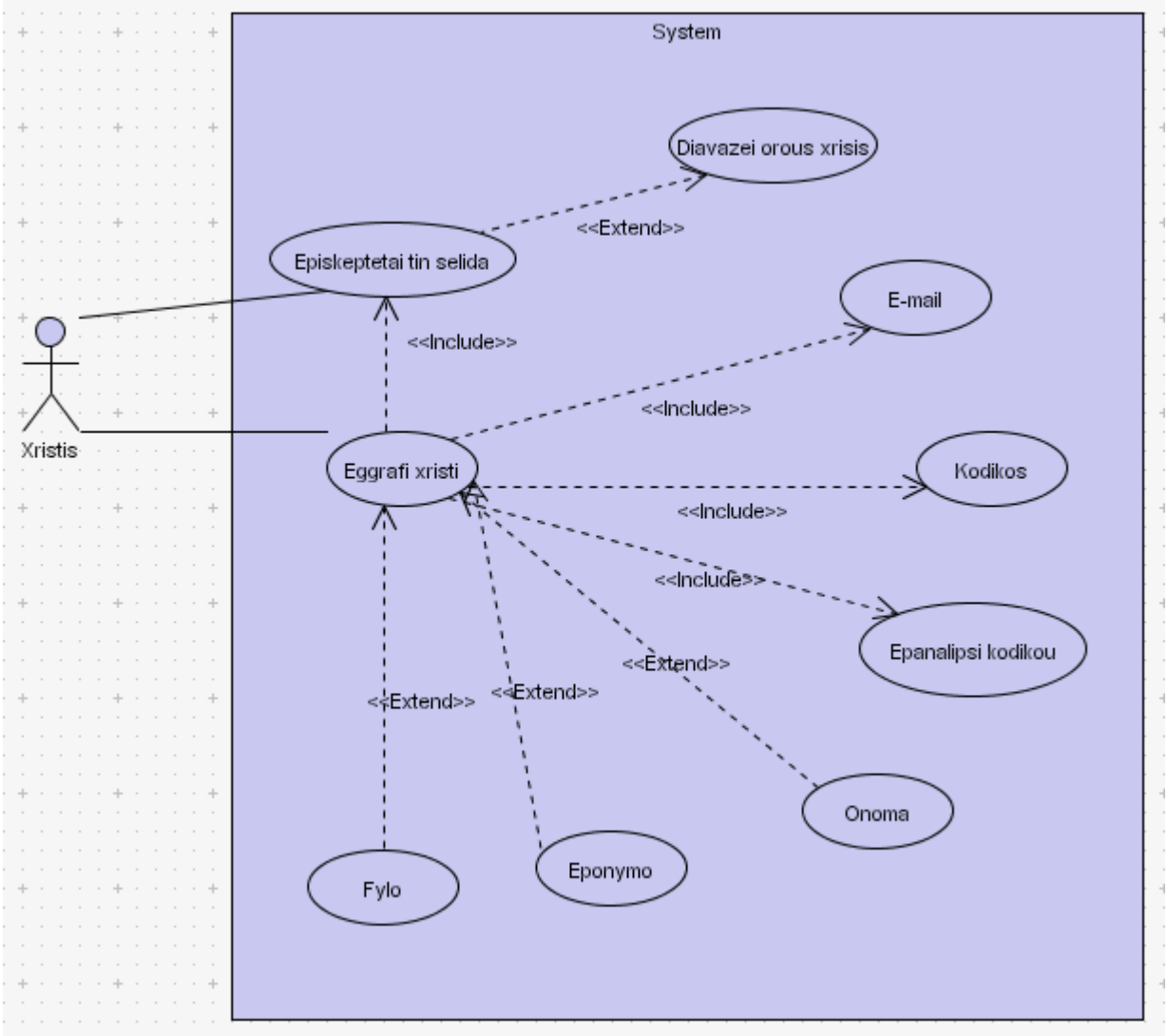
Το suggested σχετίζεται με το prohibited με σχέση 1-N (ένα προς πολλά), καθώς μια συμβουλή μπορεί να περιλαμβάνει πολλά πράγματα που πρέπει να αποφύγει ο ασθενείς για να αντιμετωπίσει την ασθένεια.



Διάγραμμα 3 - Διάγραμμα Τάξεων 1ης Έκδοσης

2.3 Διαγράμματα Περιπτώσεων Χρήσης (2η Έκδοση)

2.3.1 Use case χρήστη - Προβολή σελίδας / Εγγραφή



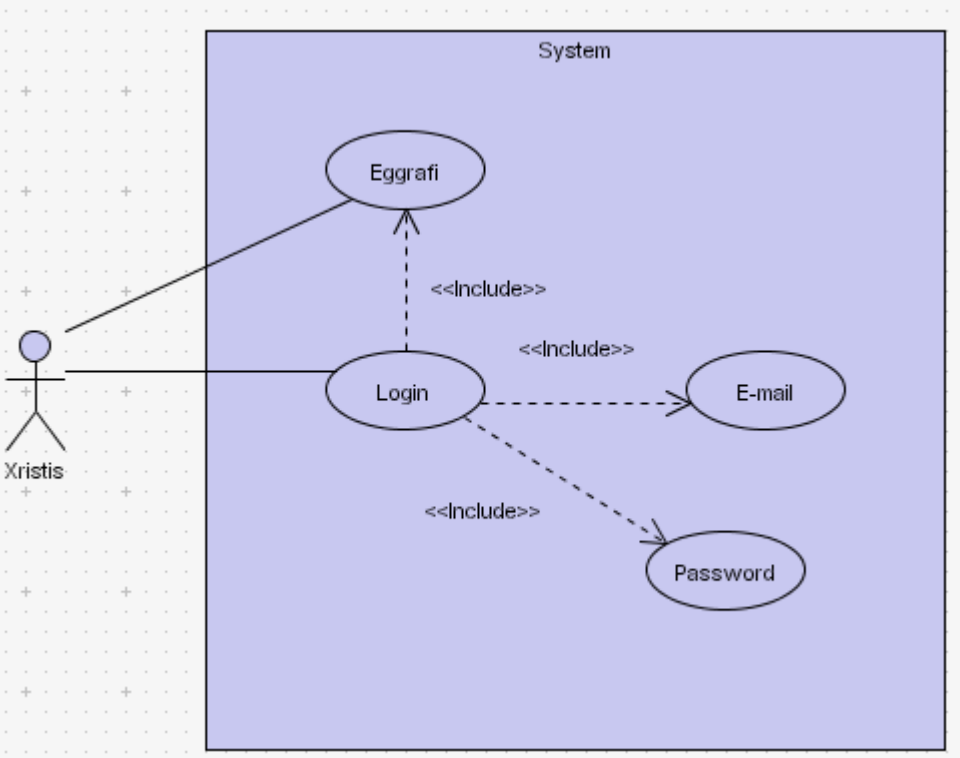
Διάγραμμα 4 - Επίσκεψη ιστοσελίδας / Εγγραφή Χρήστη 2ης Έκδοσης

Στο παραπάνω διάγραμμα απεικονίζεται το Διάγραμμα Περίπτωσης Χρήσης του χρήστη για την περίπτωση που ο χρήστης απλά επισκέπτεται την ιστοσελίδα. Σε αυτή την περίπτωση ο χρήστης μπορεί να διαβάσει και τους όρους χρήσης.

Παρακάτω απεικονίζεται η περίπτωση που ο χρήστης επιλέγει να εγγραφεί στο σύστημα. Σε αυτή την περίπτωση ο χρήστης θα πρέπει υποχρεωτικά να εισάγει το e-mail του, τον κωδικό πρόσβασης καθώς και την επανάληψη του κωδικού πρόσβασης.

Προαιρετικά ο χρήστης εισάγει το όνομα του, το επώνυμο του και το φύλο του.

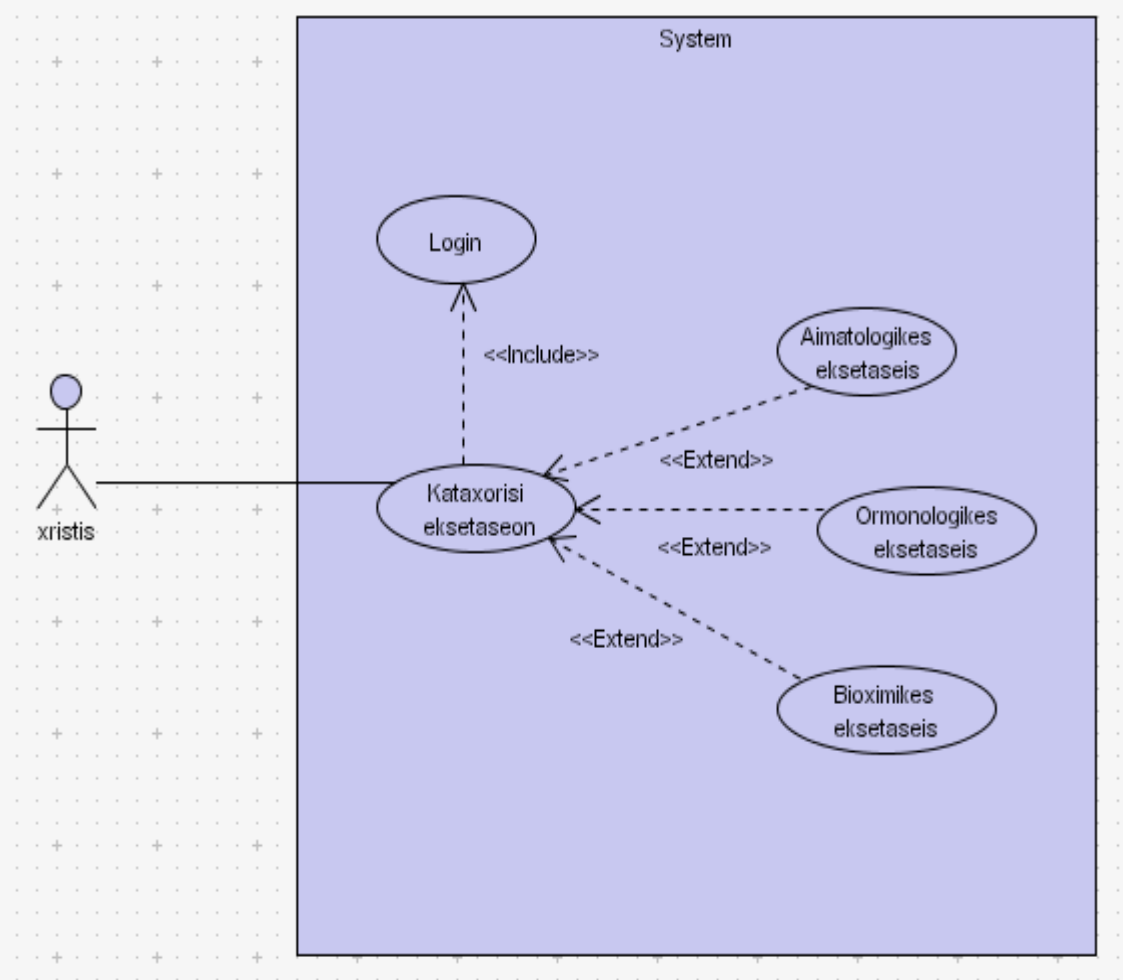
2.3.2 Use case χρήστη – Σύνδεση



Διάγραμμα 5 - Σύνδεση χρήστη 2ης Έκδοσης

Στο παραπάνω διάγραμμα απεικονίζεται το Διάγραμμα Περίπτωσης Χρήσης του χρήστη για την περίπτωση που ο χρήστης επιλέγει να συνδεθεί στο σύστημα. Σε αυτή την περίπτωση ο χρήστης θα πρέπει υποχρεωτικά να εισάγει το e-mail του και τον κωδικό πρόσβασης καθώς θα πρέπει προηγουμένως να έχει εγγραφεί στο σύστημα.

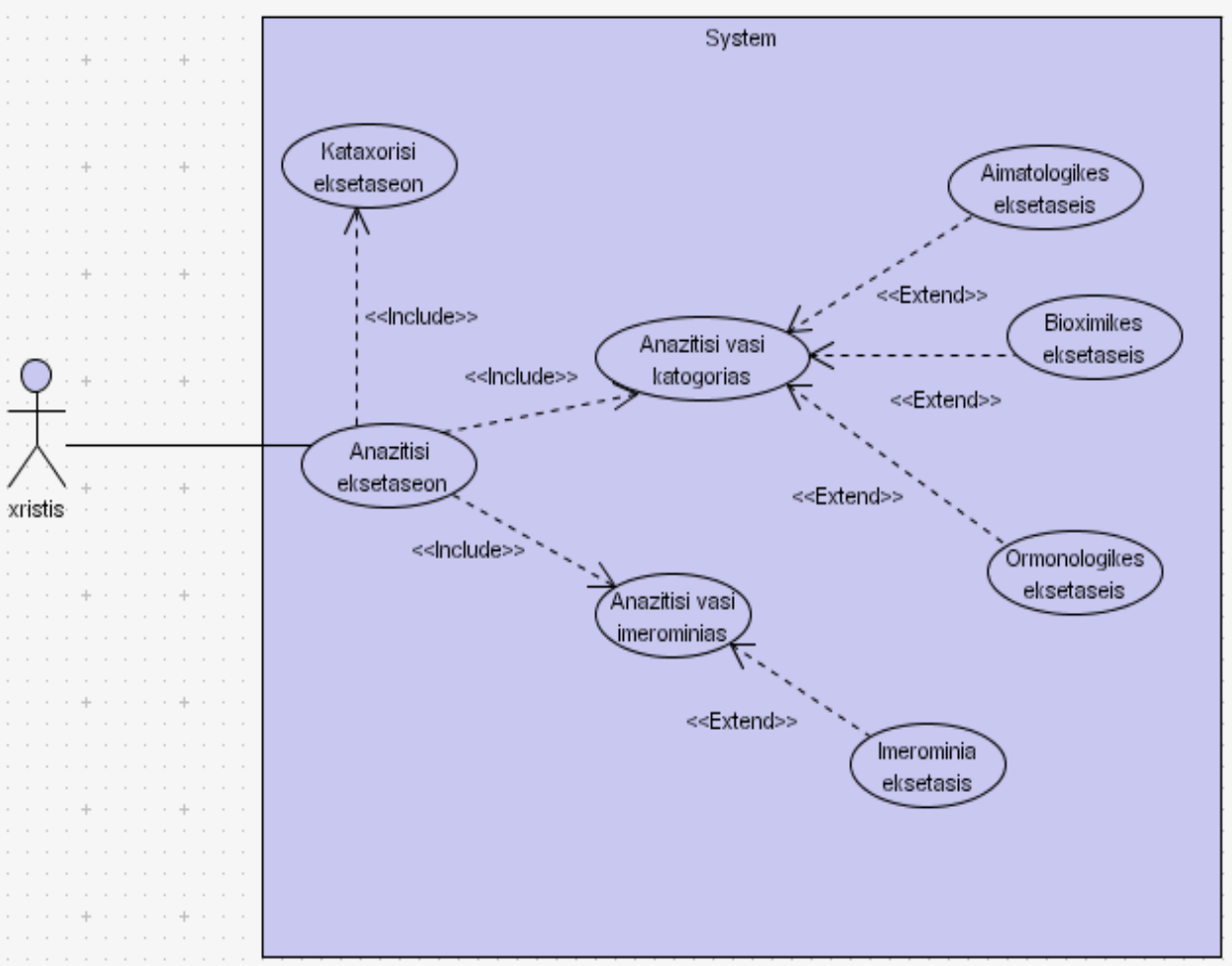
2.3.3 Use case χρήστη - Καταχώρηση εξετάσεων



Διάγραμμα 6 - Καταχώρηση Εξετάσεων 2ης Έκδοσης

Στο παραπάνω διάγραμμα απεικονίζεται το Διάγραμμα Περίπτωσης Χρήσης του χρήστη για την περίπτωση που ο χρήστης επιλέγει να καταχωρήσει εξετάσεις. Σε αυτή την περίπτωση ο χρήστης θα πρέπει υποχρεωτικά να έχει συνδεθεί στο σύστημα και μπορεί να καταχωρήσει από μία έως και όλες τις εξετάσεις και για τις τρείς κατηγορίες: "Αιματολογικές Εξετάσεις", "Ορμονολογικές Εξετάσεις", "Βιοχημικές Εξετάσεις".

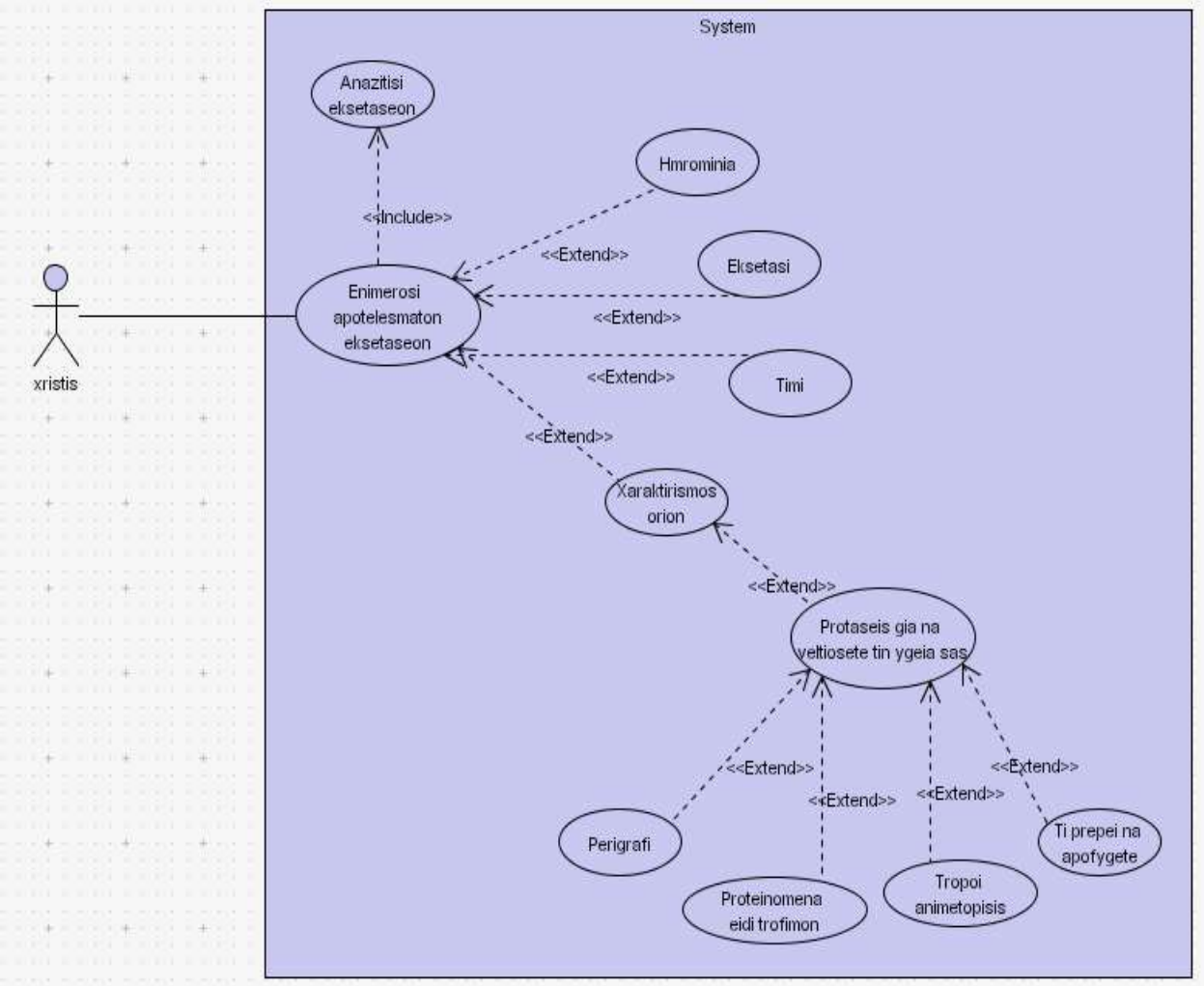
2.3.4 Use case χρήστη - Αναζήτηση εξετάσεων



Διάγραμμα 7 - Αναζήτηση Εξετάσεων 2ης Έκδοσης

Στο παραπάνω διάγραμμα απεικονίζεται το Διάγραμμα Περίπτωσης Χρήσης του χρήστη για την περίπτωση που ο χρήστης επιλέγει να αναζητήσει εξετάσεις. Σε αυτή την περίπτωση ο χρήστης θα πρέπει υποχρεωτικά να έχει καταχωρήσει προηγουμένως εξετάσεις στο σύστημα, μπορεί να καταχωρήσει από μία έως και όλες τις εξετάσεις και για τις τρείς κατηγορίες: "Αιματολογικές Εξετάσεις", "Ορμονολογικές Εξετάσεις", "Βιοχημικές Εξετάσεις".

2.3.5 Use case χρήστη - Ενημέρωση Αποτελεσμάτων εξετάσεων

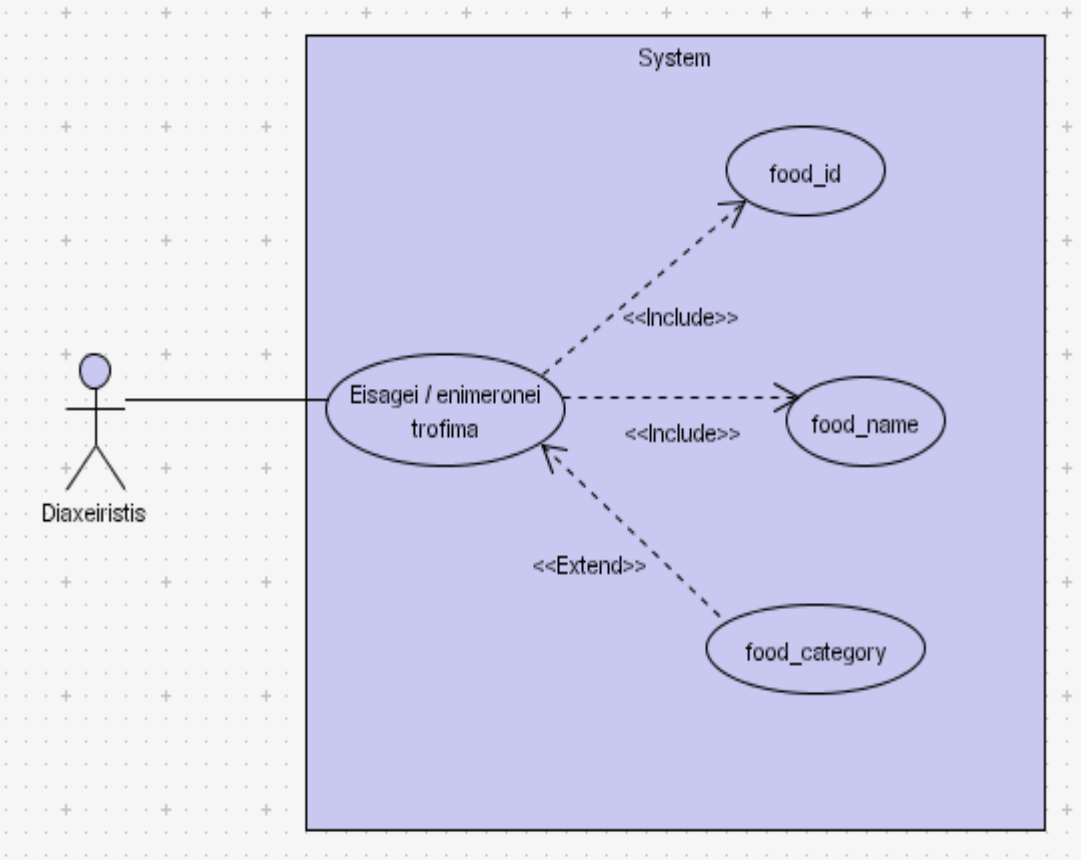


Διάγραμμα 8 - Ενημέρωση Αποτελεσμάτων Εξετάσεων 2ης Έκδοσης

Στο παραπάνω διάγραμμα απεικονίζεται το Διάγραμμα Περίπτωσης Χρήσης του χρήστη για την περίπτωση που ο χρήστης επιλέγει να ενημερωθεί για τα αποτελέσματα των εξετάσεων του. Σε αυτή την περίπτωση ο χρήστης θα πρέπει υποχρεωτικά να έχει επιλέξει να κάνει αναζήτηση των εξετάσεων του. Τα αποτελέσματα των εξετάσεων εμφανίζονται σε έναν πίνακα με τα εξής πεδία: "Ημερομηνία", "Εξέταση", "Τιμή", "Χαρακτηρισμός Ορίων".

Στη στήλη "Χαρακτηρισμός Ορίων" ανάλογα με το εάν οι εξετάσεις είναι εντός η εκτός ορίων εμφανίζεται ένας σύνδεσμος ο οποίος παραπέμπει σε μια σελίδα, η οποία ενημερώνει τον χρήστη σχετικά με τις συνέπειες , τα προτεινόμενα είδη τροφίμων, τους τρόπους αντιμετώπισης καθώς επίσης και τι θα πρέπει να αποφευχθεί.

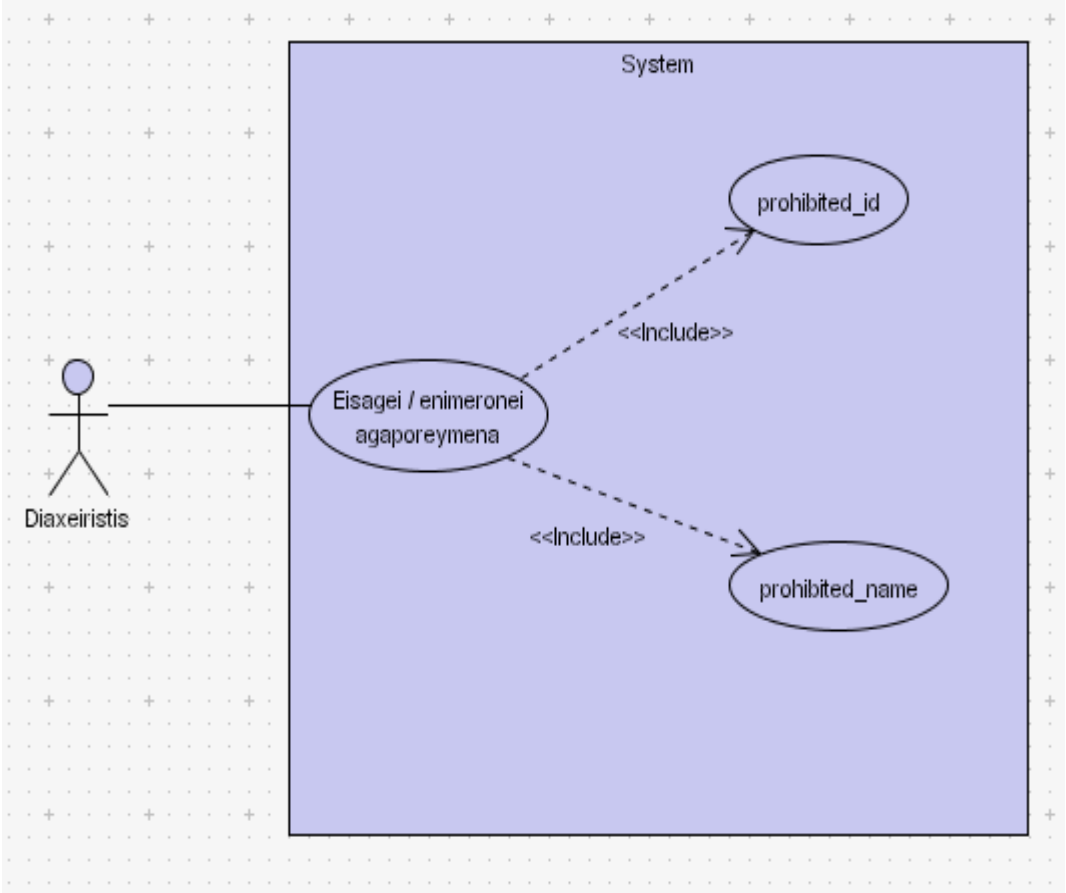
2.3.6 Use case Διαχειριστή - Εισάγει / Ενημερώνει τρόφιμα



Διάγραμμα 9 - Διαχειριστής Εισάγει / Ενημερώνει τρόφιμα 2ης Έκδοσης

Στο παραπάνω διάγραμμα απεικονίζεται το Διάγραμμα Περίπτωσης Χρήσης του διαχειριστή για την περίπτωση που ο διαχειριστής επιλέγει να εισάγει ή να ενημερώσει τα είδη τροφίμων. Σε αυτή την περίπτωση ο διαχειριστής θα πρέπει υποχρεωτικά να εισάγει τον κωδικό των τροφίμων (food_id) και το όνομα των τροφίμων (food_name), προαιρετικά εισάγει την κατηγορία των τροφίμων (food_category).

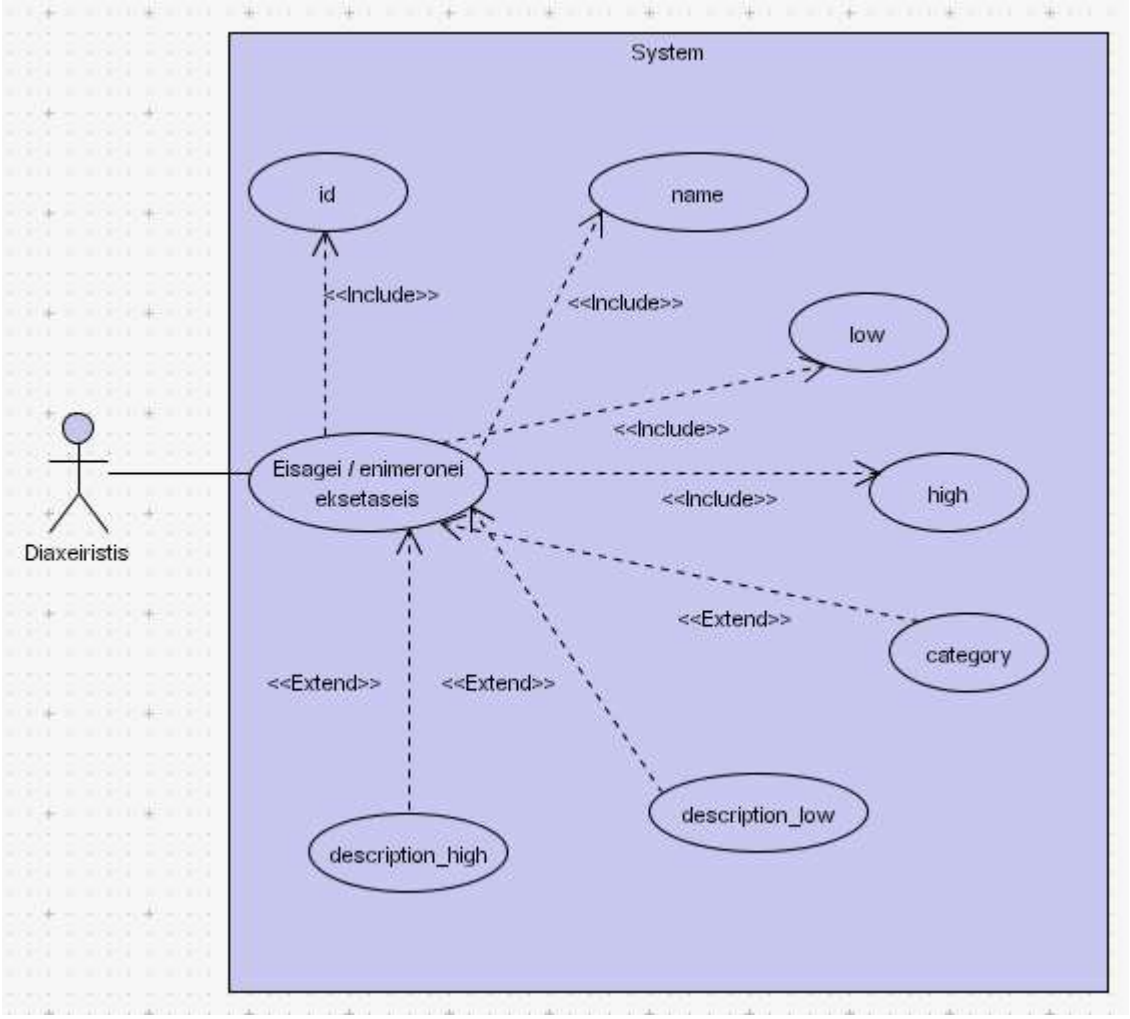
2.3.7 Use case Διαχειριστή - Εισάγει / Ενημερώνει απαγορευμένα πράγματα



Διάγραμμα 10 - Διαχειριστής Εισάγει / Ενημερώνει απαγορευμένα 2ης Έκδοσης

Στο παραπάνω διάγραμμα απεικονίζεται το Διάγραμμα Περίπτωσης Χρήσης του διαχειριστή για την περίπτωση που ο διαχειριστής επιλέγει να εισάγει ή να ενημερώσει τα απαγορευμένα πράγματα. Σε αυτή την περίπτωση ο διαχειριστής θα πρέπει υποχρεωτικά να εισάγει τον κωδικό των απαγορευμένων πραγμάτων (prohibited_id) και το όνομα των απαγορευμένων πραγμάτων (prohibited_name).

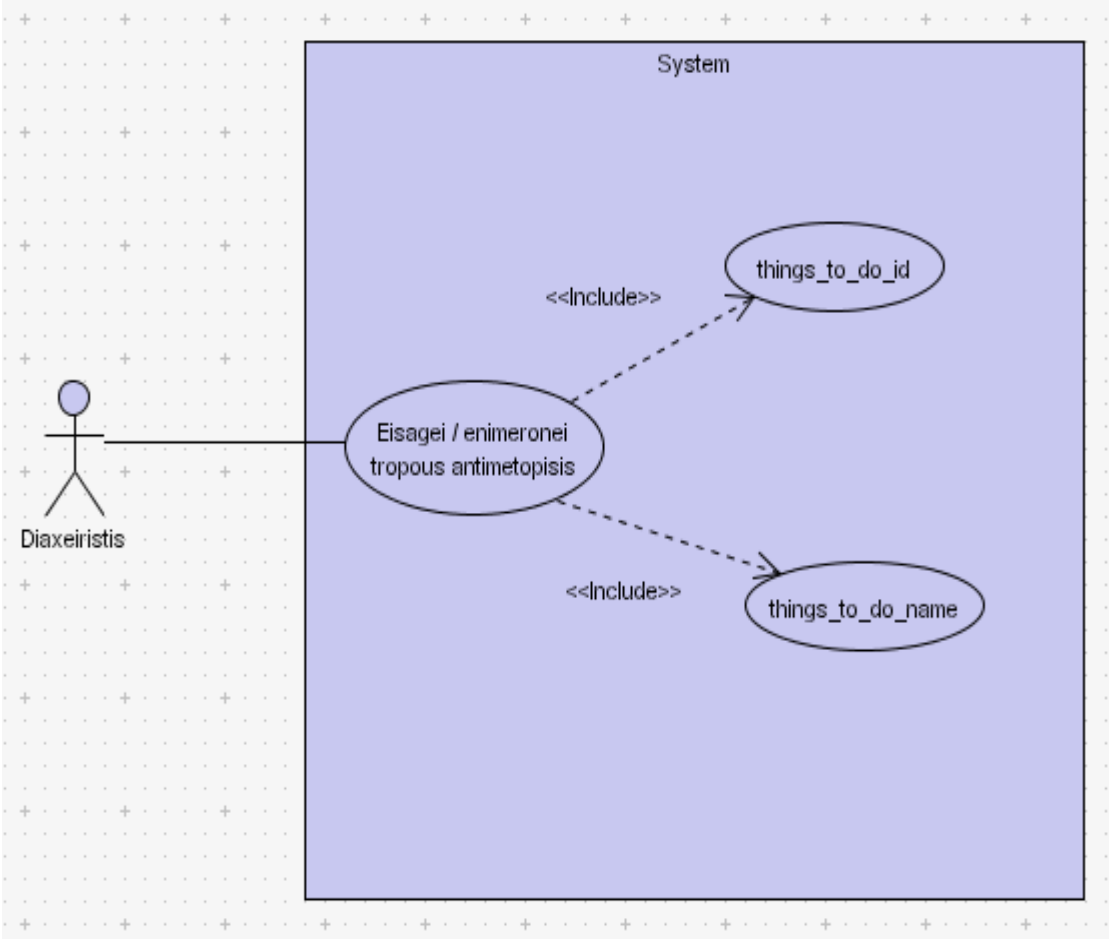
2.3.8 Use case Διαχειριστή - Εισάγει / Ενημερώνει εξετάσεις



Διάγραμμα 11 - Διαχειριστής Εισάγει / Ενημερώνει εξετάσεις 2ης Έκδοσης

Στο παραπάνω διάγραμμα απεικονίζεται το Διάγραμμα Περίπτωσης Χρήσης του διαχειριστή για την περίπτωση που ο διαχειριστής επιλέγει να εισάγει ή να ενημερώσει τις εξετάσεις. Σε αυτή την περίπτωση ο διαχειριστής θα πρέπει υποχρεωτικά να εισάγει τον κωδικό των εξετάσεων (id) , το όνομα των εξετάσεων (name), το κατώτατο όριο τιμών (low), τον ανώτατο όριο τιμών (high), προαιρετικά ο διαχειριστής εισάγει την κατηγορία των εξετάσεων (category), την περιγραφή όταν οι τιμές είναι χαμηλότερες του φυσιολογικού (description_low) και την περιγραφή όταν οι τιμές είναι υψηλότερες του φυσιολογικού (description_high).

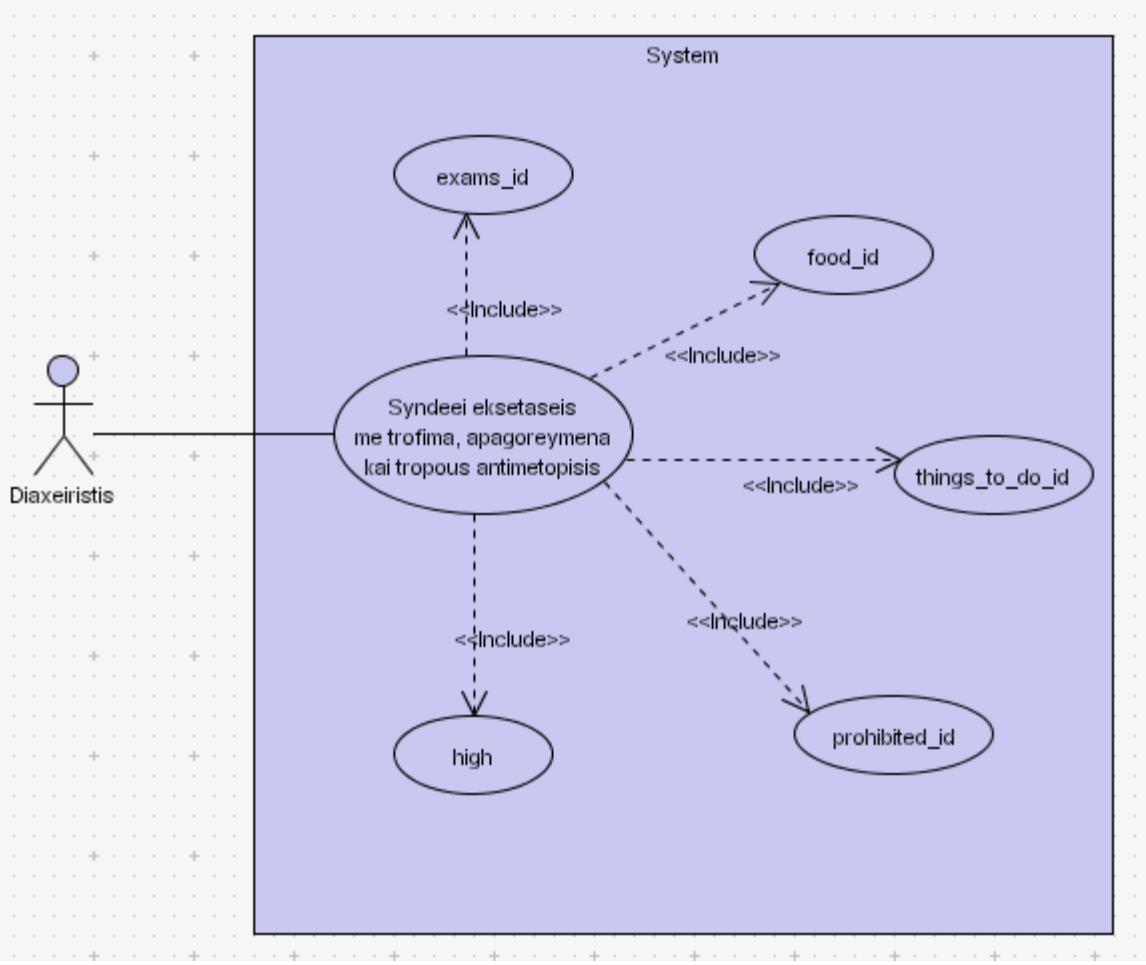
2.3.9 Use case Διαχειριστή - Εισάγει / Ενημερώνει εξετάσεις



Διάγραμμα 12 - Διαχειριστής Εισάγει / Ενημερώνει τρόπους αντιμετώπισης 2ης Έκδοσης

Στο παραπάνω διάγραμμα απεικονίζεται το Διάγραμμα Περίπτωσης Χρήσης του διαχειριστή για την περίπτωση που ο διαχειριστής επιλέγει να εισάγει ή να ενημερώσει τους τρόπους αντιμετώπισης. Σε αυτή την περίπτωση ο διαχειριστής θα πρέπει υποχρεωτικά να εισάγει τον κωδικό (things_to_do_id) , το όνομα (things_to_do_name).

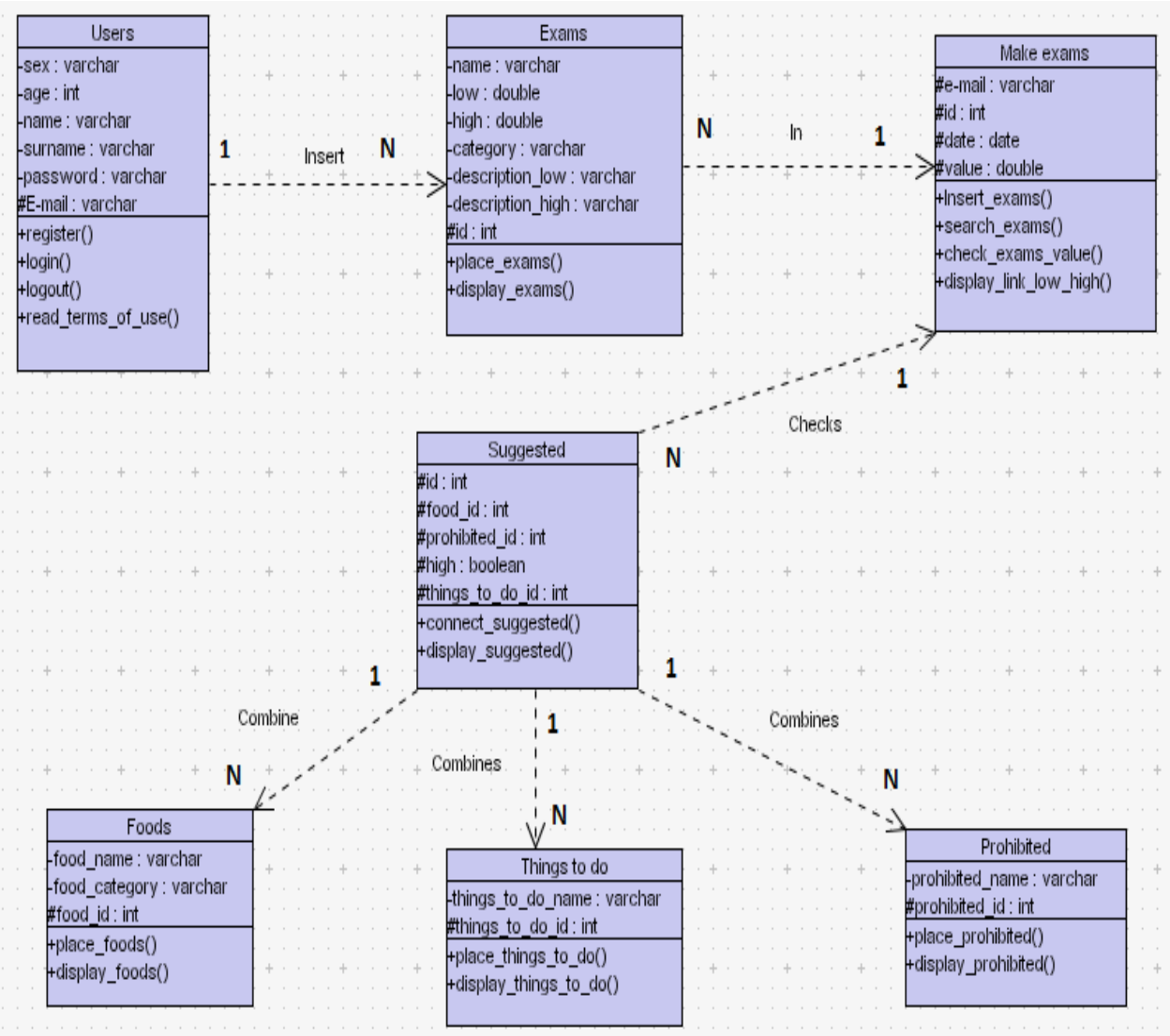
2.3.10 Use case Διαχειριστή - Συνδέει εξετάσεις με τρόφιμα, τρόπους αντιμετώπισης και απαγορευμένα.



Διάγραμμα 12 - Διαχειριστής Συνδέει εξετάσεις με τρόφιμα, τρόπους αντιμετώπισης και απαγορευμένα 2ης Έκδοσης

Στο παραπάνω διάγραμμα απεικονίζεται το Διάγραμμα Περίπτωσης Χρήσης του διαχειριστή για την περίπτωση που ο διαχειριστής επιλέγει να συνδέσει εξετάσεις με τρόφιμα, τρόπους αντιμετώπισης και απαγορευμένα. Σε αυτή την περίπτωση ο διαχειριστής θα πρέπει υποχρεωτικά να εισάγει τον κωδικό των εξετάσεων (exams_id) , το κωδικό των τροφίμων (food_id), τον κωδικό των τρόπων αντιμετώπισης (things_to_do_id), τον κωδικό των απαγορευμένων πραγμάτων (prohibited_id) καθώς και την ένδειξη εάν η τιμή είναι υψηλότερη ή χαμηλότερη του φυσιολογικού (high).

2.4 Διάγραμμα Τάξεων (2η Έκδοση)

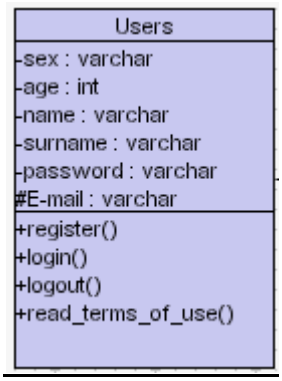


Διάγραμμα 13 - Διάγραμμα τάξεων 2ης Έκδοσης

Το τροποποιημένο διάγραμμα τάξεων για το πληροφοριακό σύστημα της ιστοσελίδας μας φαίνεται παραπάνω. Υπάρχουν επτά τάξεις, η τάξη των χρηστών (users), η τάξη εξετάσεων (exams), η τάξη στην οποία καταχωρούνται οι εξετάσεις (make exams), η τάξη με τα προτεινόμενα (suggested), η τάξη με τα τρόφιμα (foods), η τάξη με τους τρόπους αντιμετώπισης (things_to_do) και τέλος η τάξη με τα πράγματα που πρέπει να αποφεύγει ο χρήστης (prohibited).

Ακολούθως θα περιγράψουμε αναλυτικά τις λειτουργίες της κάθε τάξης:

2.4.1 Τάξη Users



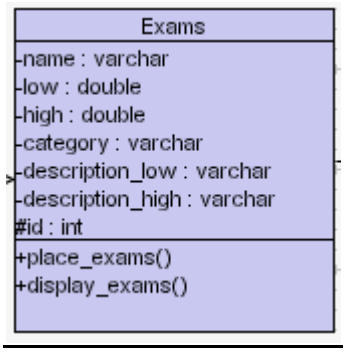
Στη τάξη Users παρατηρούμε τα εξής αντικείμενα:

- sex : string
- age : int
- name : varchar
- surname : varchar
- password : varchar
- e-mail : varchar

Όπως και τις παρακάτω λειτουργίες:

- **register ()** : Μέσω αυτής της λειτουργίας ο χρήστης εγγράφεται στο σύστημα.
- **login ()** : Μέσω αυτής της λειτουργίας ο χρήστης συνδέεται στο σύστημα.
- **logout ()** : Μέσω αυτής της λειτουργίας ο χρήστης αποσυνδέεται από το σύστημα.
- **read_terms_of_use ()** : Μέσω αυτής της λειτουργίας ο χρήστης διαβάζει τους όρους χρήσης.

2.4.2 Τάξη Exams



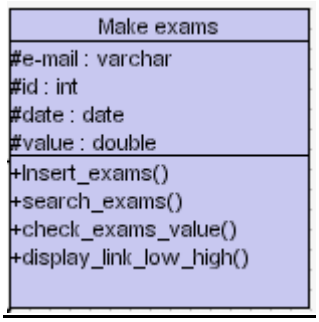
Στη τάξη αυτή παρατηρούμε τα εξής αντικείμενα:

- name : varchar
- low : double
- high : double
- category : varchar
- description_low : varchar
- description_high : varchar
- id : int

Όπως και τις παρακάτω λειτουργίες:

- **place_exams ()** : Μέσω αυτής της λειτουργίας ο διαχειριστής καταχωρεί εξετάσεις στο σύστημα.
- **display_exams ()** : Μέσω αυτής της λειτουργίας ο χρήστης βλέπει τις υπάρχουσες εξετάσεις.

2.4.3 Τάξη Make exams



Στη τάξη αυτή παρατηρούμε τα εξής αντικείμενα:

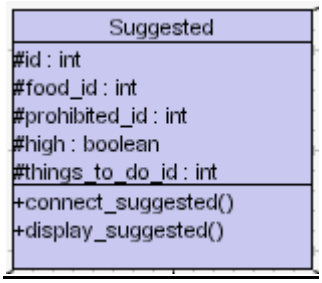
- e-mail : varchar
- id : int
- date : date
- value: double

Όπως και τις παρακάτω λειτουργίες:

- **insert_exams ()** : Μέσω αυτής της λειτουργίας ο χρήστης καταχωρεί τις εξετάσεις.
- **search_exams ()** : Μέσω αυτής της λειτουργίας ο χρήστης κάνει αναζήτηση εξετάσεων.
- **check_exams_value ()** : Μέσω αυτής της λειτουργίας το σύστημα ελέγχει την τιμή που δόθηκε από τον χρήστη και ταξινομεί την εξέταση είτε εντός ορίων, είτε εκτός ορίων.

- **display_link_low_high ()** : Μέσω αυτής της λειτουργίας το σύστημα εμφανίζει τον σύνδεσμο στον χρήστη, ανάλογα με εάν τα όρια της εξέτασης ήταν χαμηλότερα ή υψηλότερα του φυσιολογικού.

2.4.4 Τάξη Suggested



Στη τάξη αυτή παρατηρούμε τα εξής αντικείμενα:

- id : int
- food_id : int
- prohibited_id : int
- high : boolean
- things_to_do_id : int

Όπως και τις παρακάτω λειτουργίες:

- **connect_suggetsed () :** Μέσω αυτής της λειτουργίας το σύστημα συνδέει τις εξετάσεις με τα τρόφιμα, με τους τρόπους αντιμετώπισης και τα απαγορευμένα ανάλογα με το αν η τιμή high είναι true ή false, δηλαδή εάν οι τιμές των εξετάσεων είναι χαμηλότερες ή υψηλότερες του φυσιολογικού.

- **display_suggested ()** : Μέσω αυτής της λειτουργίας το σύστημα εμφανίζει προτάσεις στον χρήστη, ανάλογα με εάν τα όρια της εξέτασης ήταν χαμηλότερα ή υψηλότερα του φυσιολογικού.

2.4.5 Τάξη Foods

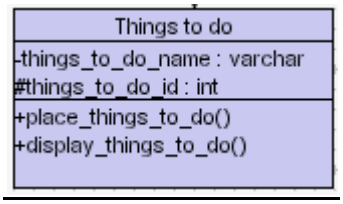
Στη τάξη αυτή παρατηρούμε τα εξής αντικείμενα:

- food_id : int
- food_name : varchar
- food_category : varchar

Όπως και τις παρακάτω λειτουργίες:

- **place_foods () :** Μέσω αυτής της λειτουργίας ο διαχειριστής καταχωρεί τρόφιμα στη βάση δεδομένων.
- **display_foods ()** : Μέσω αυτής της λειτουργίας το σύστημα εμφανίζει τα τρόφιμα στο χρήστη.

2.4.6 Τάξη Things_to_do



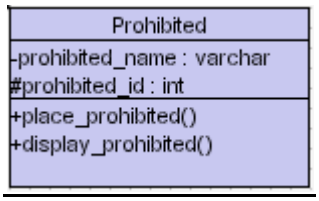
Στη τάξη αυτή παρατηρούμε τα εξής αντικείμενα:

- things_to_do_name : varchar
- things_to_do_id : int

Όπως και τις παρακάτω λειτουργίες:

- **place_things_to_do () :** Μέσω αυτής της λειτουργίας ο διαχειριστής καταχωρεί τρόπους αντιμετώπισης στη βάση δεδομένων.
- **display_ things_to_do ()** : Μέσω αυτής της λειτουργίας το σύστημα εμφανίζει τους τρόπους αντιμετώπισης στο χρήστη.

2.4.7 Τάξη Prohibited



Στη τάξη αυτή παρατηρούμε τα εξής αντικείμενα:

- prohibited_name : varchar
- prohibited_id : int

Όπως και τις παρακάτω λειτουργίες:

- **place_prohibited () :** Μέσω αυτής της λειτουργίας ο διαχειριστής καταχωρεί τα απαγορευμένα πράγματα στη βάση δεδομένων.
- **display_ prohibited ()** : Μέσω αυτής της λειτουργίας το σύστημα εμφανίζει τα απαγορευμένα πράγματα στο χρήστη.

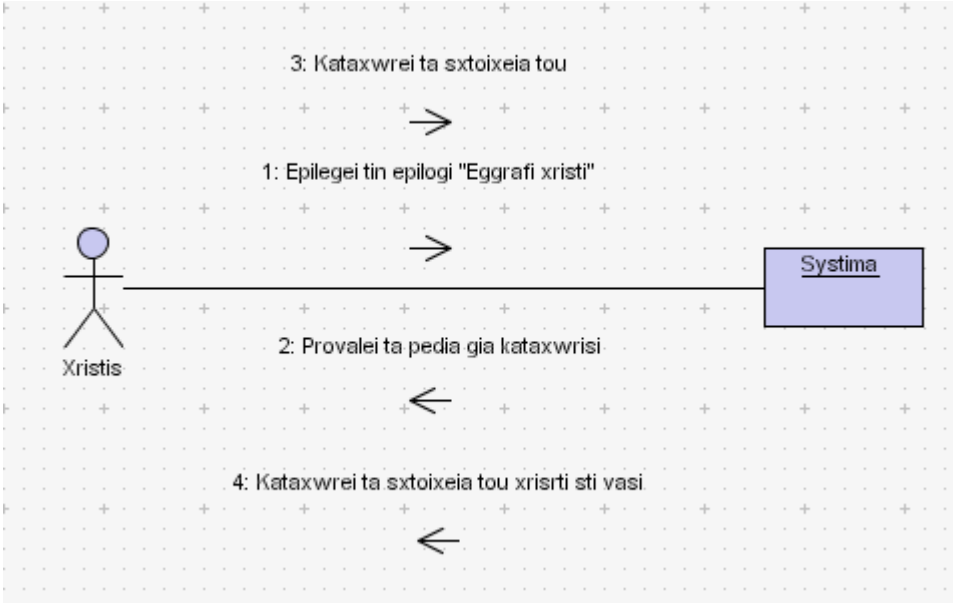
2.5 Διαγράμματα συνεργασίας (1η Έκδοση)

Τα διαγράμματα συνεργασίας είναι σημαντικά κατά την αντικειμενοστραφή ανάλυση και σχεδιασμό διότι απεικονίζουν τις αλληλεπιδράσεις ανάμεσα στα αντικείμενα. Τα διαγράμματα συνεργασίας εκφράζουν το περιβάλλον μεταξύ των αντικειμένων μέσω των συνδέσμων, αλλά και την αλληλεπίδραση μεταξύ των αντικειμένων μέσω μηνυμάτων που εμφανίζονται κατά μήκος των συνδέσμων. Στα διαγράμματα συνεργασίας ο χρόνος δεν αναπαρίσταται, με αποτέλεσμα τα διάφορα μηνύματα να αριθμούνται για να δηλώσουν τη σειρά αποστολής.

Η αναπαράσταση των αλληλεπιδράσεων γίνεται ως εξής. Το περιεχόμενο μιας αλληλεπίδρασης περιλαμβάνει τα ορίσματα, τις τοπικές μεταβλητές που δημιουργήθηκαν κατά την διάρκεια της εκτέλεσης και τους συνδέσμους ανάμεσα στα αντικείμενα που συμμετέχουν στην αλληλεπίδραση. Μια αλληλεπίδραση εκτελείται από ένα σύνολο αντικειμένων που συνεργάζονται ανταλλάσσοντας μηνύματα. Αυτά τα μηνύματα εμφανίζονται κατά μήκος των συνδέσμων που συνδέουν τα αντικείμενα χρησιμοποιώντας βέλη που δείχνουν προς τον παραλήπτη του μηνύματος. Αυτό θα το διαπιστώσουμε στα διαγράμματα παρακάτω.

Αντίθετα με τα διαγράμματα σειράς ο χρόνος δεν αναπαρίσταται σαφώς σε ένα διάγραμμα συνεργασίας και αυτό έχει σαν αποτέλεσμα τα διάφορα μηνύματα να αριθμούνται για να δηλώσουν τη σειρά αποστολής. Τέλος, τα διαγράμματα αυτά δείχνουν τις αλληλεπιδράσεις ανάμεσα στα αντικείμενα και ταυτόχρονα τις δομικές σχέσεις που διευκολύνουν αυτές τις αλληλεπιδράσεις.

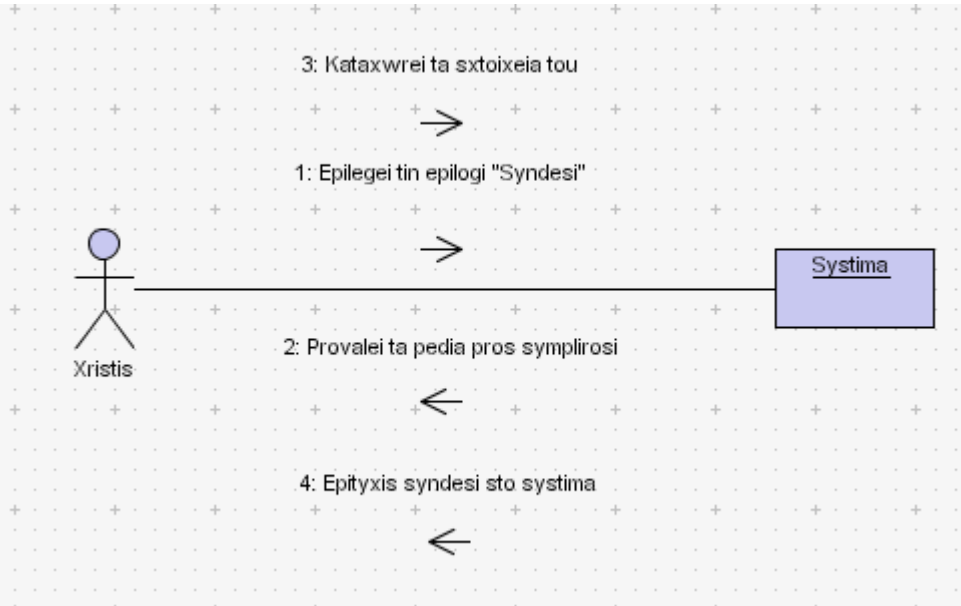
2.5.1 Διαδικασία εγγραφής χρήστη



Διάγραμμα 14 - Διάγραμμα συνεργασίας - Εγγραφή χρήστη 1ης Έκδοσης

Στο παραπάνω διάγραμμα απεικονίζεται το διάγραμμα συνεργασίας για την εγγραφή χρήστη. Όπως παρατηρούμε, ο χρήστης επιλέγει την επιλογή "Εγγραφή χρήστη", στη συνέχεια το σύστημα προβάλλει τα πεδία προς καταχώρηση, ο χρήστης καταχωρεί τα στοιχεία του και το σύστημα τα αποθηκεύει στη βάση δεδομένων.

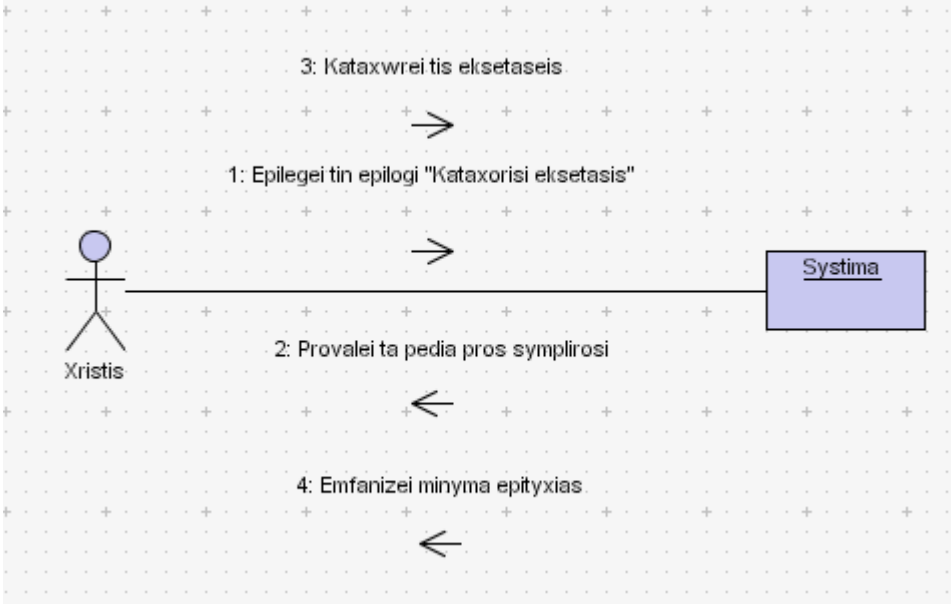
2.5.2 Διαδικασία σύνδεσης χρήστη



Διάγραμμα 15 - Διάγραμμα συνεργασίας - Σύνδεση χρήστη 1ης Έκδοσης

Στο παραπάνω διάγραμμα απεικονίζεται το διάγραμμα συνεργασίας για την σύνδεση χρήστη. Όπως παρατηρούμε, ο χρήστης επιλέγει την επιλογή " Σύνδεση ", στη συνέχεια το σύστημα προβάλλει τα πεδία προς καταχώρηση, ο χρήστης καταχωρεί τα στοιχεία του και το σύστημα προχωράει σε επιτυχής σύνδεση στο σύστημα.

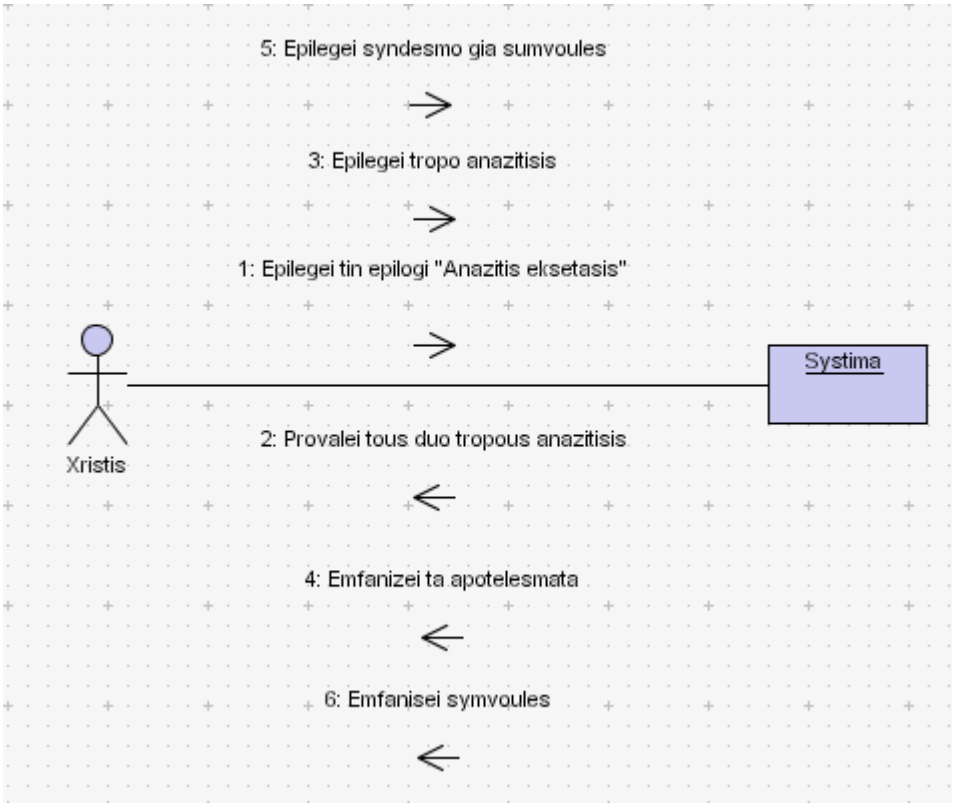
2.5.3 Διαδικασία καταχώρησης εξετάσεων από τον χρήστη



Διάγραμμα 16 - Διάγραμμα συνεργασίας - Καταχώρηση εξετάσεων χρήστη 1ης Έκδοσης

Στο παραπάνω διάγραμμα απεικονίζεται το διάγραμμα συνεργασίας για την καταχώρηση εξετάσεων χρήστη. Όπως παρατηρούμε, ο χρήστης επιλέγει την επιλογή "Καταχώρηση εξέτασης ", στη συνέχεια το σύστημα προβάλλει τα πεδία προς καταχώρηση, ο χρήστης καταχωρεί τις εξετάσεις και το σύστημα εμφανίζει μήνυμα επιτυχούς καταχώρησης εξετάσεων.

2.5.4 Διαδικασία αναζήτησης εξετάσεων και προβολή συμβουλών



Διάγραμμα 17 - Διάγραμμα συνεργασίας - Αναζήτηση εξετάσεων χρήστη 1ης Έκδοσης

Στο παραπάνω διάγραμμα απεικονίζεται το διάγραμμα συνεργασίας για την αναζήτηση εξετάσεων χρήστη. Όπως παρατηρούμε, ο χρήστης επιλέγει την επιλογή "Αναζήτηση εξέτασης ", στη συνέχεια το σύστημα προβάλλει τους δύο τρόπους αναζήτησης, ο χρήστης επιλέγει τρόπο αναζήτησης και το σύστημα εμφανίζει τα αποτελέσματα.

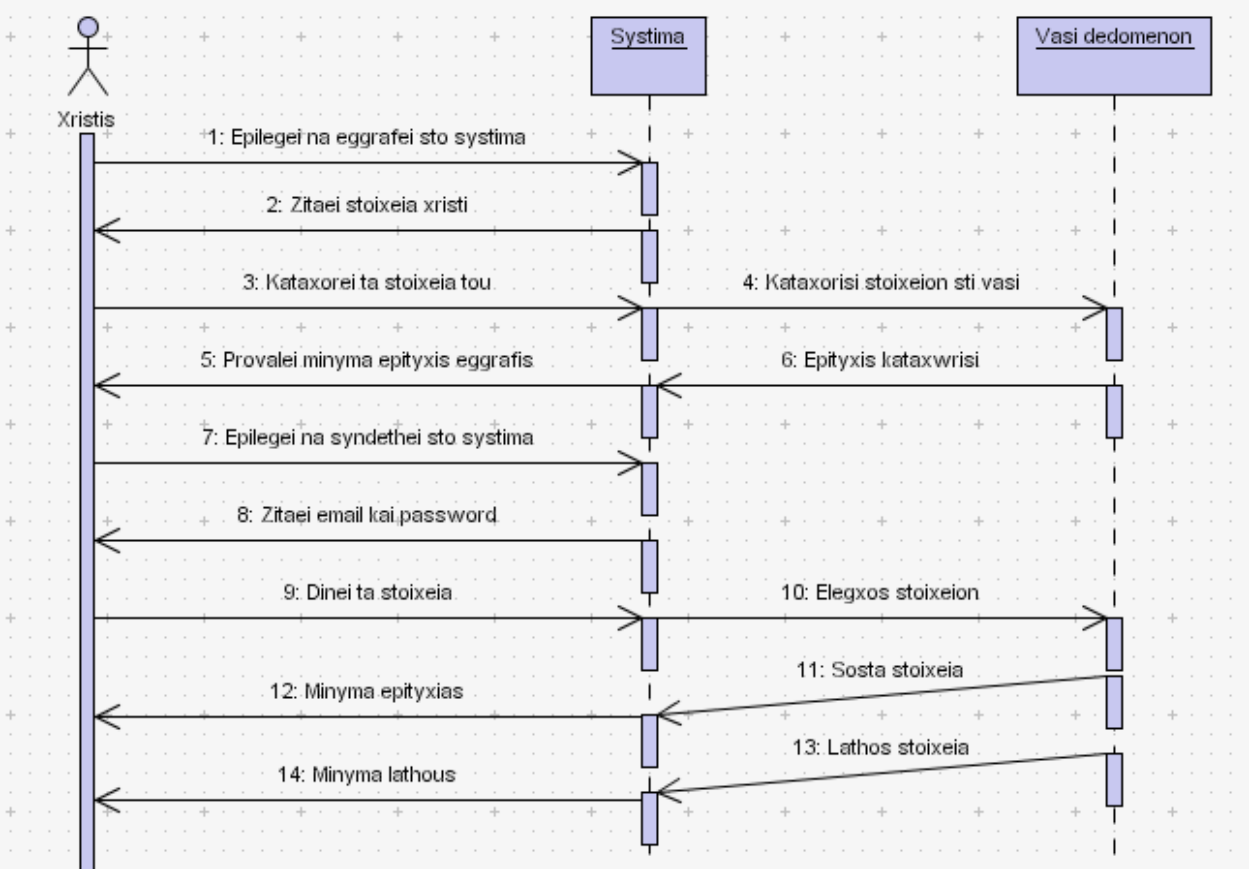
Στη συνέχεια ο χρήστης επιλέγει κάποιο σύνδεσμο για να ενημερωθεί για τις συμβουλές για την συγκεκριμένη εξέταση και το σύστημα εμφανίζει τις συμβουλές για την εξέταση εκείνη.

2.6 Διαγράμματα Σειράς (1η Έκδοση)

Τα διαγράμματα σειράς αναπαριστούν αλληλεπιδράσεις ανάμεσα στα αντικείμενα από μια χρονική άποψη. Σε αντίθεση με τα διαγράμματα συνεργασίας, το περιβάλλον των αντικειμένων δεν αναπαρίσταται σαφώς. Η αναπαράσταση επικεντρώνεται στην έκφραση των αλληλεπιδράσεων.

Ένα διάγραμμα σειράς αναπαριστά μια αλληλεπίδραση ανάμεσα σε αντικείμενο αναπαρίσταται με ένα ορθογώνιο και μία κάθετη γραμμή, που καλείται γραμμή ζωής του αντικειμένου.

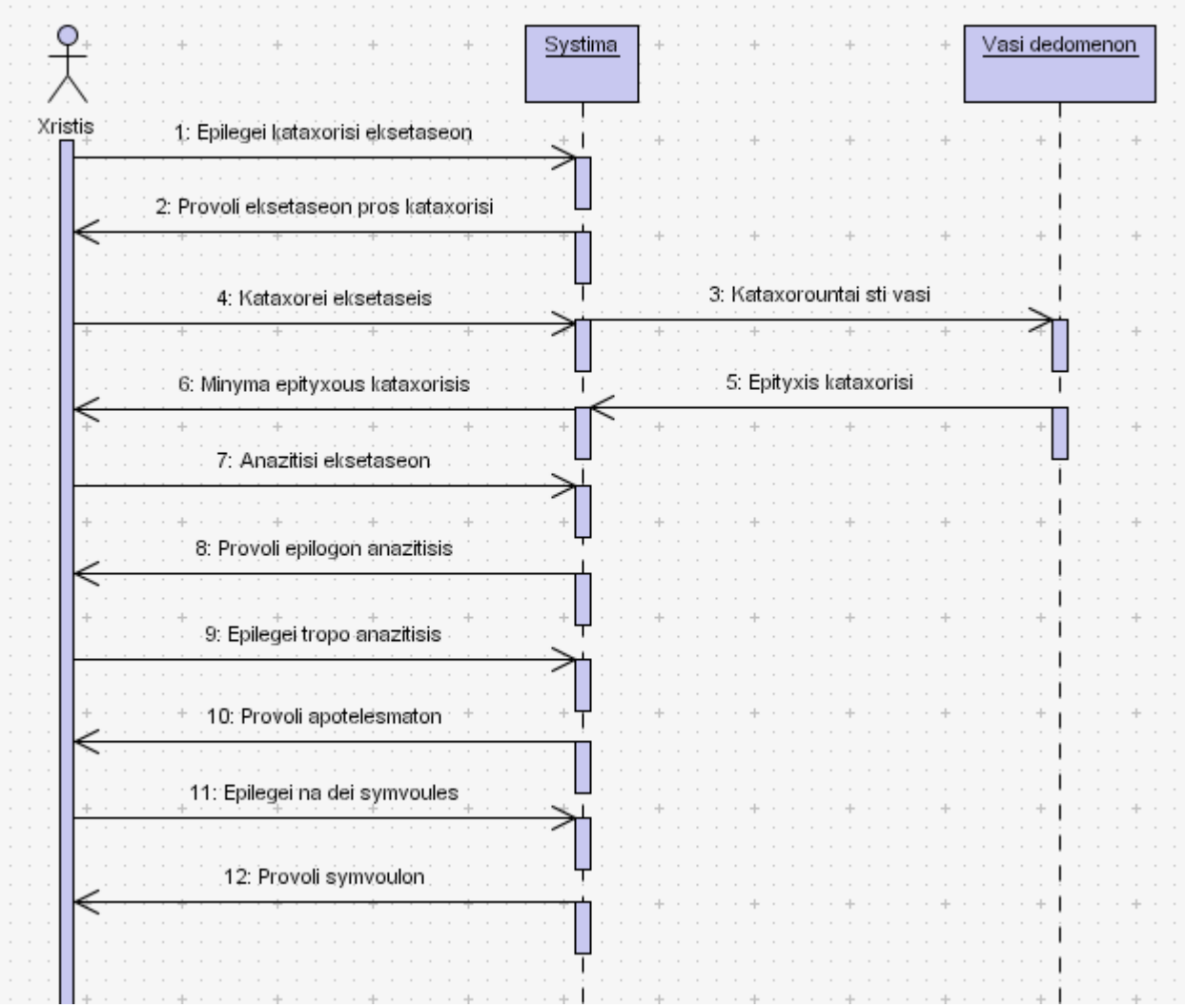
2.6.1 Διαδικασία εγγραφής και σύνδεσης χρήστη



Διάγραμμα 18 - Διάγραμμα σειράς - Εγγραφή / Σύνδεση χρήστη 1ης Έκδοσης

Στο παραπάνω διάγραμμα απεικονίζεται το διάγραμμα σειράς για την εγγραφή και σύνδεση χρήστη. Όπως παρατηρούμε, ο χρήστης επιλέγει να εγγραφεί στο σύστημα. Στη συνέχεια το σύστημα προβάλλει τα πεδία προς καταχώρηση, ο χρήστης καταχωρεί τα στοιχεία του και το σύστημα τα αποθηκεύει στη βάση δεδομένων. Έπειτα ο χρήστης επιλέγει να συνδεθεί στο σύστημα, το σύστημα προβάλλει τα πεδία προς καταχώρηση, ο χρήστης καταχωρεί τα στοιχεία του και το σύστημα προχωράει σε επιτυχής σύνδεση αν τα στοιχεία είναι σωστά ή σε μήνυμα λάθους αν τα στοιχεία είναι λάθος.

2.6.2 Διαδικασία αναζήτησης εξετάσεων και προβολή συμβουλών



Διάγραμμα 19 - Διάγραμμα σειράς - Εξετάσεις 1ης Έκδοσης

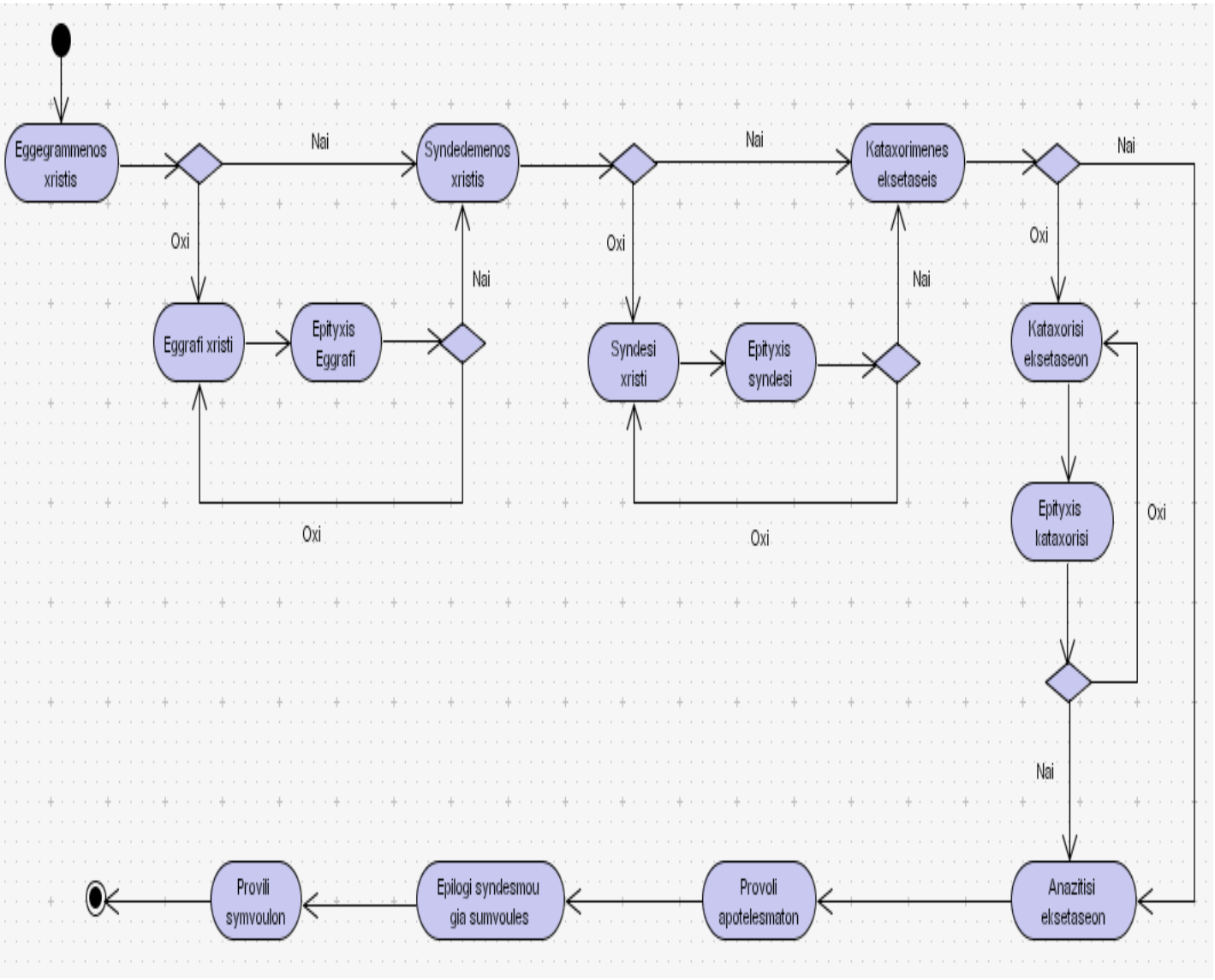
Στο παραπάνω διάγραμμα απεικονίζεται το διάγραμμα σειράς για την καταχώρηση εξετάσεων χρήστη. Όπως παρατηρούμε, ο χρήστης επιλέγει να καταχωρήσει τις εξετάσεις, στη συνέχεια το σύστημα προβάλλει τα πεδία προς καταχώρηση, ο χρήστης καταχωρεί τις εξετάσεις και το σύστημα εμφανίζει μήνυμα επιτυχούς καταχώρησης εξετάσεων.

Στη συνέχεια ο χρήστης επιλέγει να αναζητήσει τις εξετάσεις, το σύστημα προβάλλει τους δύο τρόπους αναζήτησης, ο χρήστης επιλέγει τρόπο αναζήτησης και το σύστημα εμφανίζει τα αποτελέσματα. Έπειτα ο χρήστης επιλέγει κάποιο σύνδεσμο για να ενημερωθεί για τις συμβουλές για την συγκεκριμένη εξέταση και το σύστημα εμφανίζει τις συμβουλές για την εξέταση εκείνη.

2.7 Διαγράμματα Δραστηριοτήτων

Ένα διάγραμμα δραστηριοτήτων είναι μια παραλλαγή των διαγραμμάτων καταστάσεων οργανωμένο σύμφωνα με ενέργειες. Στόχος τους είναι κυρίως η αναπαράσταση της εσωτερικής συμπεριφοράς μιας μεθόδου ή μιας περίπτωσης χρήσης σαν μια ακολουθία βημάτων.

Τα σύμβολα που χρησιμοποιούνται για τον σχεδιασμό αυτών των διαγραμμάτων είναι τα ίδια με τα διαγράμματα καταστάσεων απλά η σημασία τους αλλάζει. Ποιο συγκεκριμένα οι αρχική και τελική κατάσταση θεωρείται η αρχή και το τέλος της λειτουργίας ενώ οι λεγόμενες ενδιάμεσες καταστάσεις αποτελούν της δραστηριότητες της λειτουργίας . Τέλος οι δραστηριότητες συνδέονται με αυτόματες μεταβάσεις που αναπαριστώνται με βέλη.



Διάγραμμα 20 - Διάγραμμα δραστηριοτήτων

Κεφάλαιο 3° - Υλοποίηση

3.1 Υλοποίηση βάσης δεδομένων

Συμφώνα με τις παραπάνω αναλύσεις απαιτήσεων, προκύπτει ότι η βάση δεδομένων θα πρέπει να περιλαμβάνει επτά πίνακες, όπως φαίνονται παρακάτω:

Table	Rows	Type	Size	Comments
exams	12	InnoDB	64 KiB	Creation: Aug 30, 2015 at 03:41 PM
foods	49	InnoDB	16 KiB	Creation: Aug 30, 2015 at 04:32 PM
make_exams	24	InnoDB	32 KiB	Creation: Aug 30, 2015 at 03:41 PM
prohibited	22	InnoDB	16 KiB	Creation: Aug 30, 2015 at 03:45 PM
suggested	74	InnoDB	64 KiB	Creation: Sep 12, 2015 at 07:22 PM
things_to_do	9	InnoDB	16 KiB	Creation: Aug 30, 2015 at 06:17 PM
users	19	InnoDB	16 KiB	Creation: Aug 30, 2015 at 03:41 PM
7 tables	209	--	224 KiB	

Ο πίνακας **exams,** θα περιλαμβάνει πληροφορίες σχετικά με τις εξετάσεις, όπως ονομασία εξετάσεων, περιγραφή, τα όρια των αποτελεσμάτων.

Ο πίνακας **foods,** θα περιλαμβάνει πληροφορίες σχετικά με τα τρόφιμα, όπως ονομασία τροφίμων, κωδικός κτλ.

Ο πίνακας **make_exams,** θα περιλαμβάνει πληροφορίες σχετικά με τις εξετάσεις που θα καταχωρεί ο χρήστης. Τέτοιες πληροφορίες θα είναι, το είδος εξέτασης, η ημερομηνία, οι τιμές κτλ.

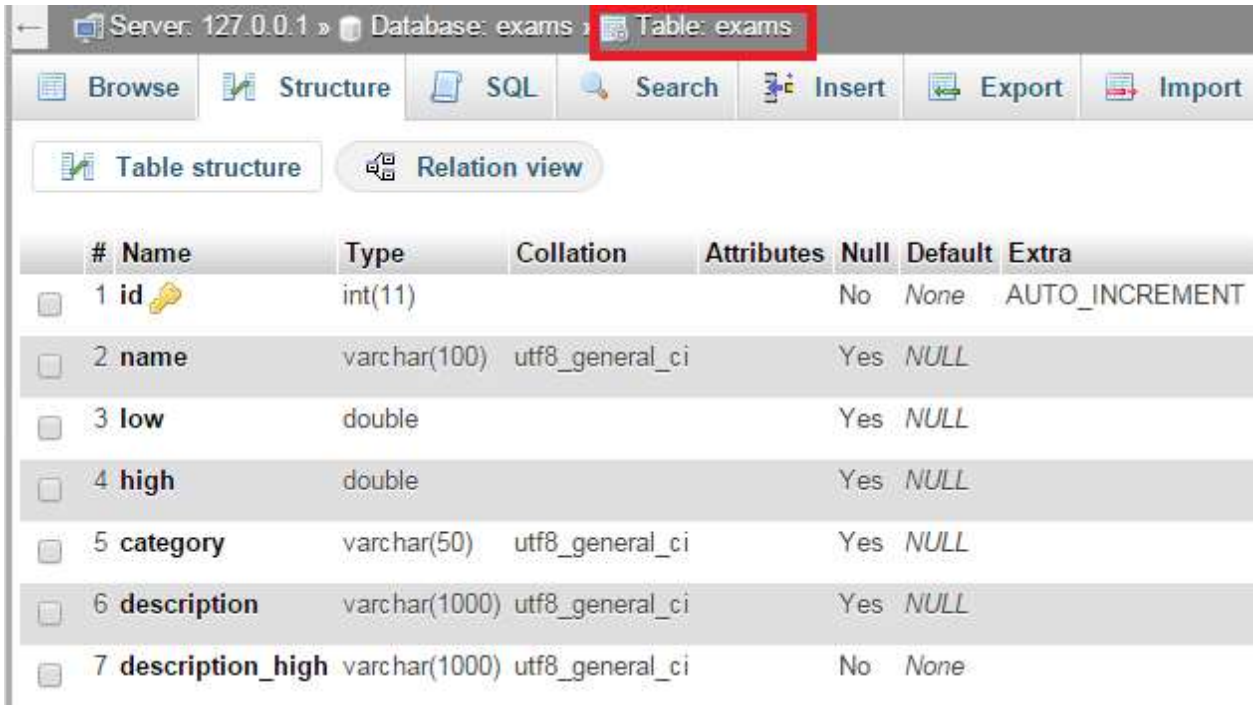
Ο πίνακας **prohibited,** θα περιλαμβάνει πληροφορίες σχετικά με τα πράγματα που πρέπει να αποφύγει ο χρήστης, όπως ονομασία, κωδικός κτλ.

Ο πίνακας **suggested** θα περιλαμβάνει πληροφορίες σχετικά με τα πράγματα που θα προτείνονται στο χρήστη, όπου θα είναι ένας συνδυασμός από τα foods, prohibited και things to do.

Ο πίνακας **things_to_do,** θα περιλαμβάνει πληροφορίες σχετικά με τους τρόπους αντιμετώπισης, όπως ονομασία, κωδικός κτλ.

Τέλος, ο πίνακας **users,** θα περιλαμβάνει πληροφορίες σχετικά με τους χρήστες, όπως όνομα, κωδικός, e-mail, password κτλ.

3.1.1 Πίνακας exams



#	Name	Type	Collation	Attributes	Null	Default	Extra
1	id	int(11)			No	None	AUTO_INCREMENT
2	name	varchar(100)	utf8_general_ci		Yes	NULL	
3	low	double			Yes	NULL	
4	high	double			Yes	NULL	
5	category	varchar(50)	utf8_general_ci		Yes	NULL	
6	description	varchar(1000)	utf8_general_ci		Yes	NULL	
7	description_high	varchar(1000)	utf8_general_ci		No	None	

Ο πίνακας **exams** περιέχει τις εξής στήλες:

1. Κωδικός εξετάσεων (id)
2. Το όνομα των εξετάσεων (name)
3. Το κατώτατο όριο τιμών (low)
4. Τον ανώτατο όριο τιμών (high)
5. Την κατηγορία των εξετάσεων (category)
6. Την περιγραφή όταν οι τιμές είναι χαμηλότερες του φυσιολογικού (description)
7. Την περιγραφή όταν οι τιμές είναι υψηλότερες του φυσιολογικού (description_high)

Το **primary key** του πίνακα είναι η στήλη (id).

3.1.2 Πίνακας foods

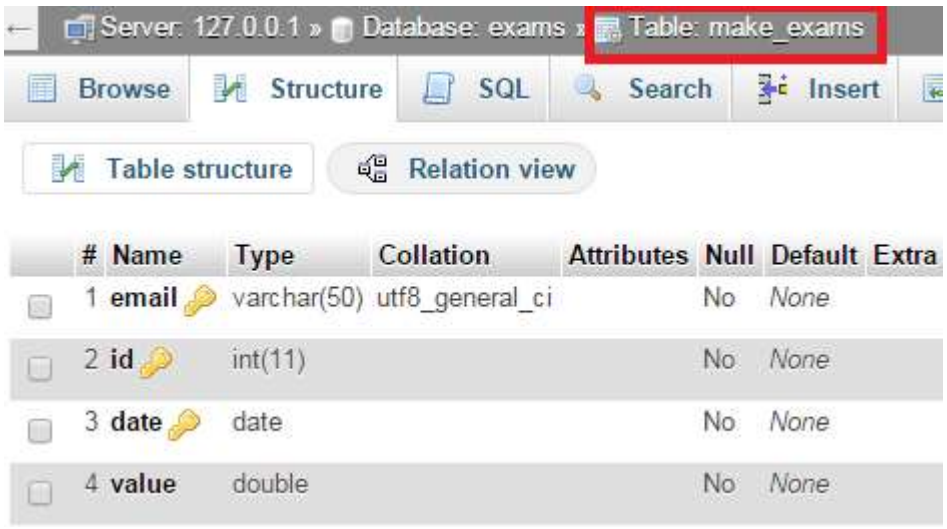
#	Name	Type	Collation	Attributes	Null	Default	Extra
1	foodid	int(11)			No	None	AUTO_INCREMENT
2	foodname	varchar(50)	utf8_general_ci		No	None	
3	food_category	varchar(100)	utf8_general_ci		No	None	

Ο πίνακας **foods** περιέχει τις εξής στήλες:

1. Κωδικός τροφίμων (foodid)
2. Την ονομασία των τροφίμων (foodname)
3. Την κατηγορία των τροφίμων (food_category)

Το **primary key** του πίνακα είναι η στήλη (foodid).

3.1.3 Πίνακας make_exams



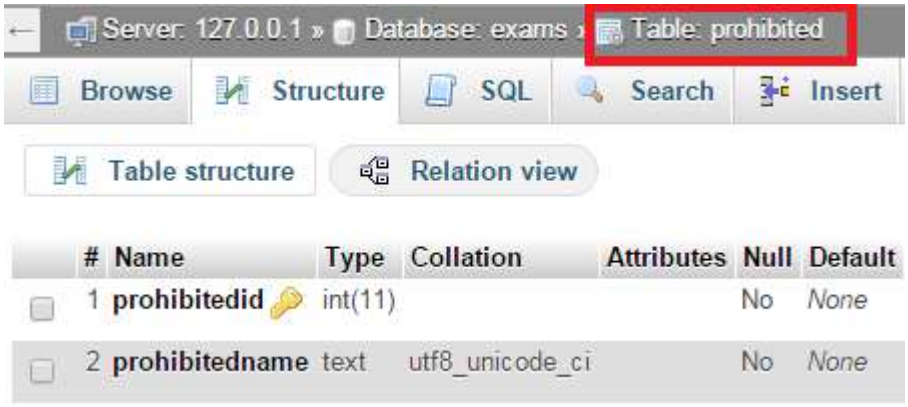
#	Name	Type	Collation	Attributes	Null	Default	Extra
1	email	varchar(50)	utf8_general_ci		No	None	
2	id	int(11)			No	None	
3	date	date			No	None	
4	value	double			No	None	

Ο πίνακας **make_exams** περιέχει τις εξής στήλες:

1. E-mail (email)
2. Κωδικός εξετάσεων (id)
3. Ημερομηνία εξετάσεων (date)
4. Τιμή εξετάσεων (value)

Το **primary key** του πίνακα είναι ο συνδυασμός των στηλών (email), (id), (date).

3.1.4 Πίνακας prohibited

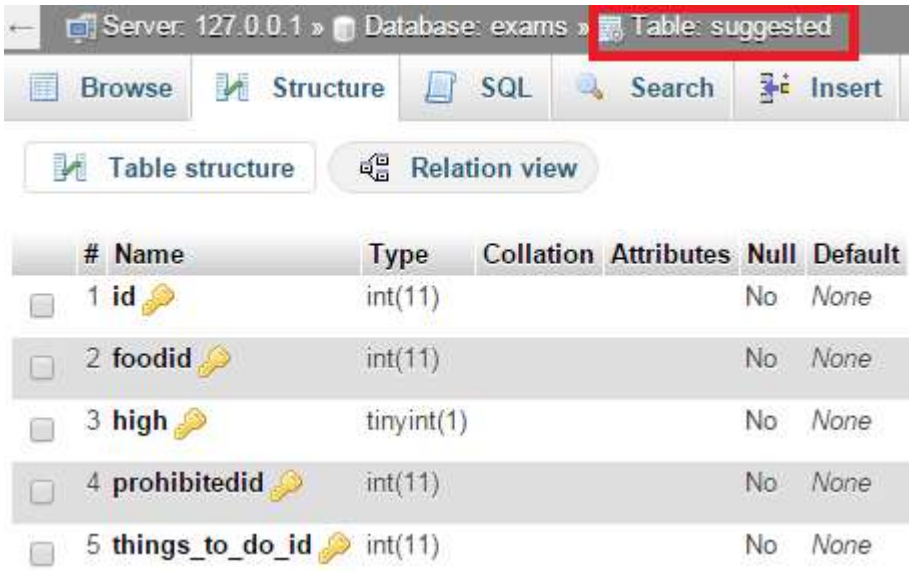


Ο πίνακας **prohibited** περιέχει τις εξής στήλες:

1. Κωδικός (prohibitedid)
2. Την ονομασία (prohibitedname)

Το **primary key** του πίνακα είναι η στήλη (prohibitedid).

3.1.5 Πίνακας suggested



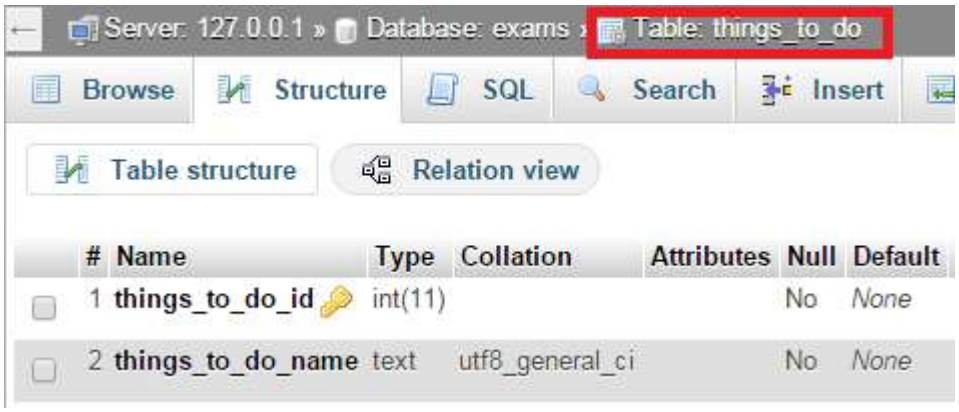
#	Name	Type	Collation	Attributes	Null	Default
1	id	int(11)			No	None
2	foodid	int(11)			No	None
3	high	tinyint(1)			No	None
4	prohibitedid	int(11)			No	None
5	things_to_do_id	int(11)			No	None

Ο πίνακας **suggested** περιέχει τις εξής στήλες:

1. Κωδικός εξετάσεων (id)
2. Κωδικός τροφίμων (foodid)
3. Ένδειξη εάν οι τιμές είναι χαμηλότερες ή υψηλότερες του φυσιολογικού (high)
4. Κωδικός απαγορευμένων (prohibitedid)
5. Κωδικός τρόπων αντιμετώπισης (things_to_do_id)

Το **primary key** του πίνακα είναι ο συνδυασμός όλων των στηλών του.

3.1.6 Πίνακας things_to_do



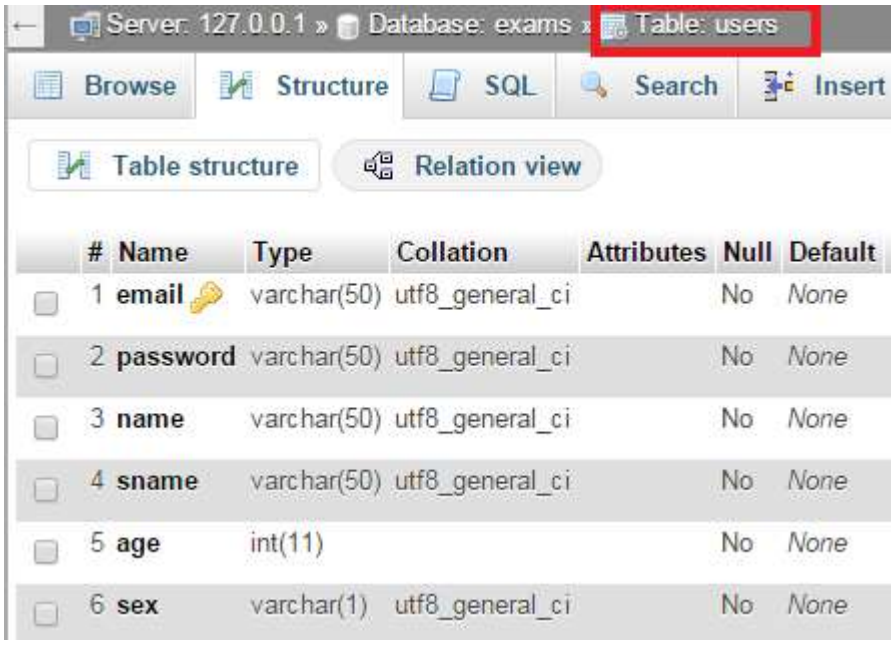
#	Name	Type	Collation	Attributes	Null	Default
1	things_to_do_id	int(11)			No	None
2	things_to_do_name	text	utf8_general_ci		No	None

Ο πίνακας **things_to_do** περιέχει τις εξής στήλες:

1. Κωδικός (things_to_do _id)
2. Την ονομασία (things_to_do_name)

Το **primary key** του πίνακα είναι η στήλη (things_to_do_name).

3.1.7 Πίνακας users



#	Name	Type	Collation	Attributes	Null	Default
1	email	varchar(50)	utf8_general_ci		No	None
2	password	varchar(50)	utf8_general_ci		No	None
3	name	varchar(50)	utf8_general_ci		No	None
4	sname	varchar(50)	utf8_general_ci		No	None
5	age	int(11)			No	None
6	sex	varchar(1)	utf8_general_ci		No	None

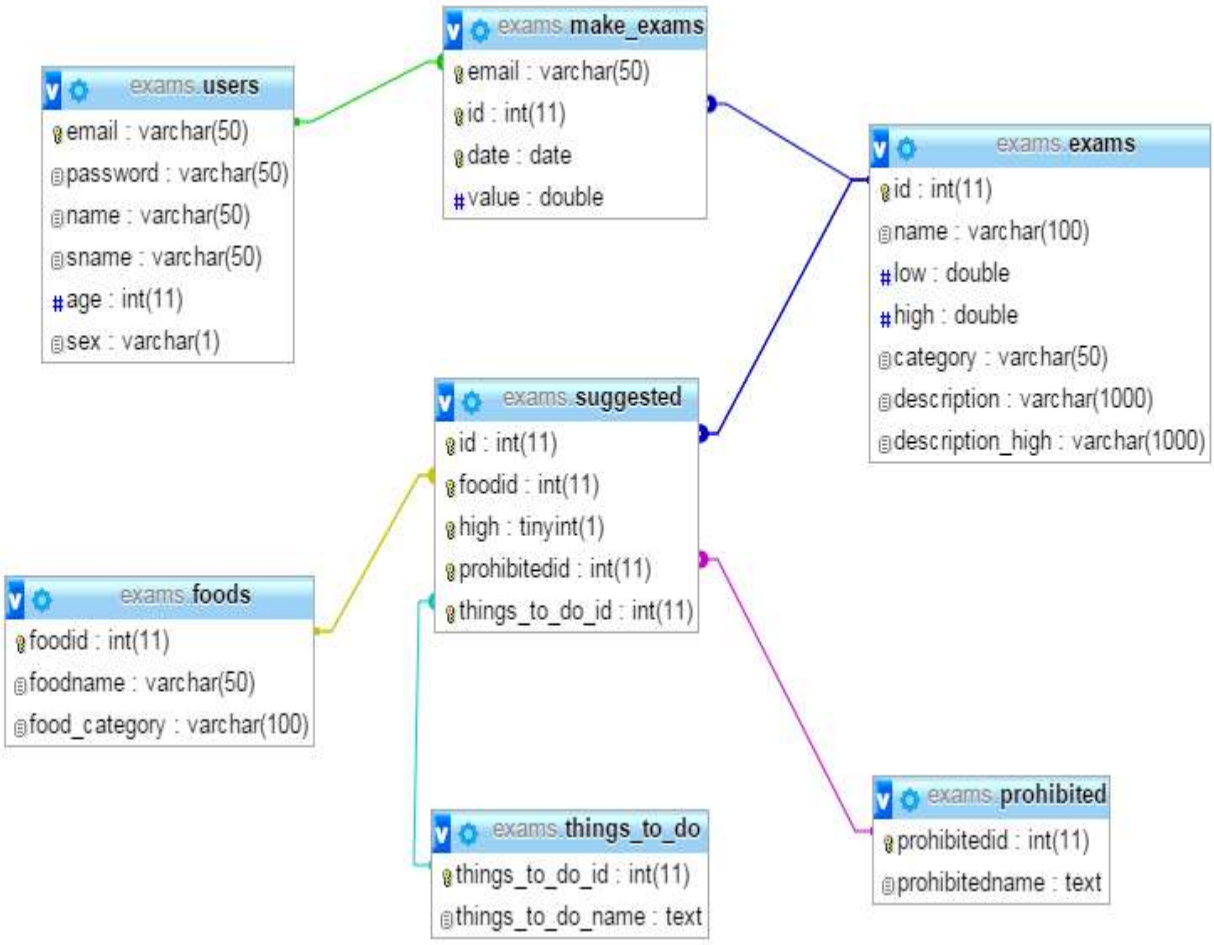
Ο πίνακας **users** περιέχει τις εξής στήλες:

1. E-mail χρήστη (email)
2. Κωδικός σύνδεσης (password)
3. Όνομα χρήστη (name)
4. Επώνυμο χρήστη (smane)
5. Ηλικία χρήστη (age)
6. Φύλο χρήστη (sex)

Το **primary key** του πίνακα είναι η στήλη (email).

3.1.8 Σύνδεση πινάκων

Στην παρακάτω εικόνα βλέπουμε την σύνδεση των πινάκων της βάσης δεδομένων.



Παρατηρούμε ότι :

1. Ο πίνακας **users** συνδέεται με τον πίνακα **make_exams**.
2. Ο πίνακας **make_exams** συνδέεται με τον πίνακα **exams**.
3. Ο πίνακας **exams** συνδέεται με τον πίνακα **suggested** και **make_exams.**
4. Ο πίνακας **suggested** συνδέεται με τους πίνακες **foods**, **things_to_do** και **prohibited.**

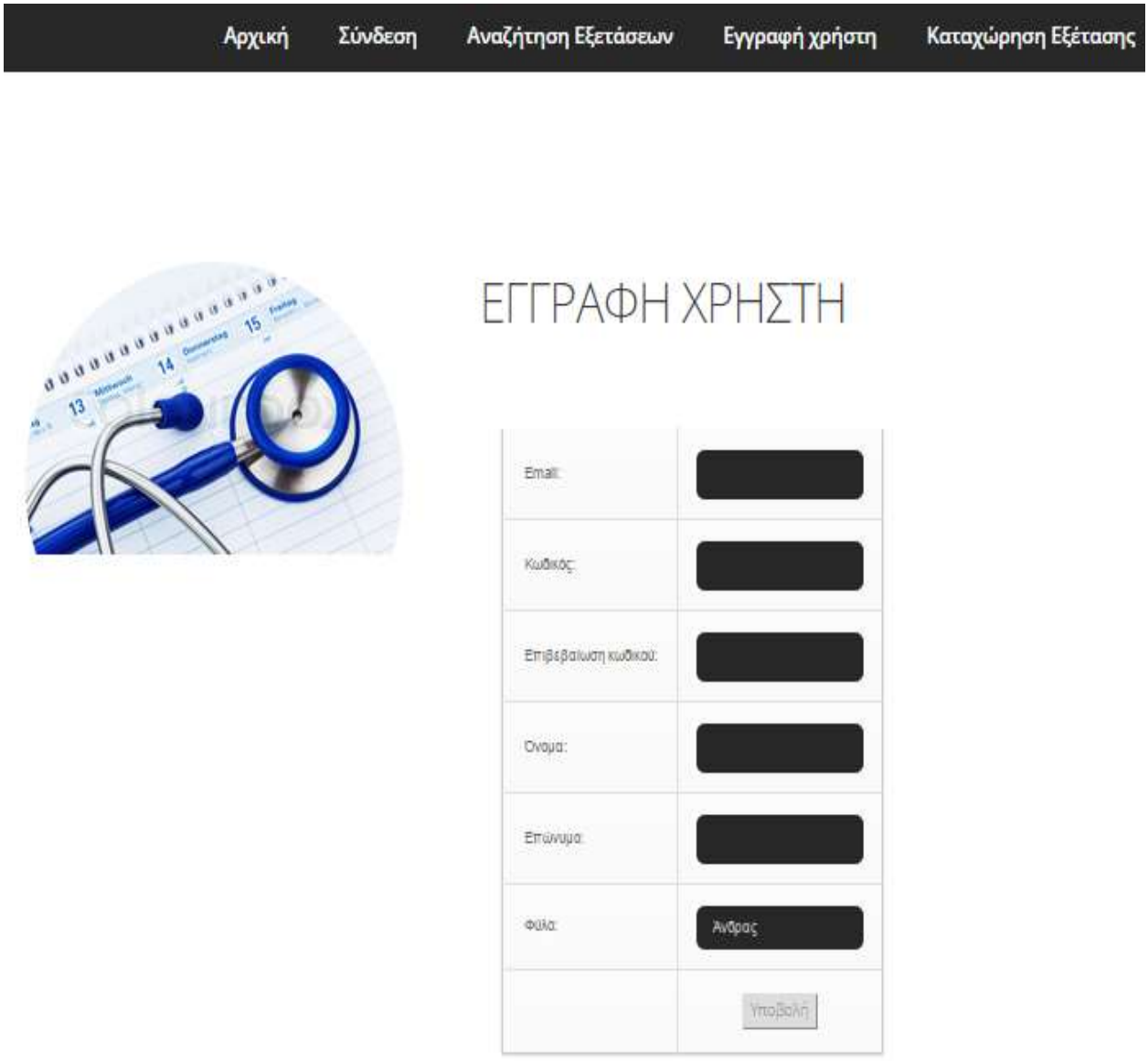
3.2 Εγχειρίδιο Χρήστη

3.2.1 Αρχική σελίδα

Στην αρχική σελίδα ο χρήστης έχει την δυνατότητα να διαβάσει τους όρους χρήσης της ιστοσελίδας και να αποφασίσει εάν θέλει να προχωρήσει στην εγγραφή του.

3.2.2 Εγγραφή Χρήστη

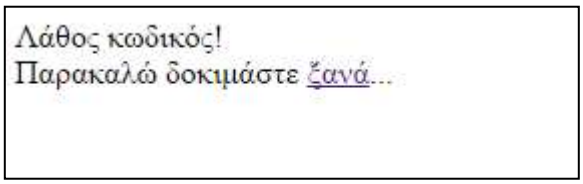
Σε αυτή τη σελίδα ο χρήστης έχει την δυνατότητα να δημιουργήσει λογαριασμό καταχωρώντας το e-mail του, τον κωδικό (δύο φορές), το όνομα του, το επώνυμο του, την ηλικία του και το φύλο του.



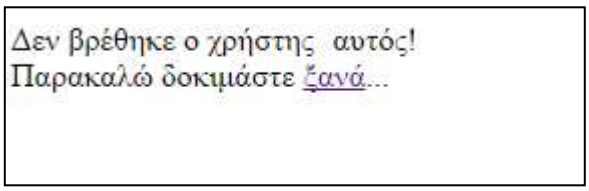
3.2.3 Σύνδεση Χρήστη

Σε αυτή τη σελίδα ο χρήστης έχει την δυνατότητα να συνδεθεί στο σύστημα καταχωρώντας το e-mail του, τον κωδικό πρόσβασης.

Εάν ο χρήστης καταχωρήσει λάθος κωδικό, το σύστημα θα του εμφανίσει ένα σχετικό μήνυμα λάθους:



Εάν ο χρήστης καταχωρήσει λάθος e-mail, το σύστημα θα του εμφανίσει ένα σχετικό μήνυμα λάθους:



3.2.4 Καταχώρηση Εξετάσεων

Σε αυτή την σελίδα ο χρήστης έχει την δυνατότητα να καταχωρήσει τις εξετάσεις του. Όπως παρατηρούμε υπάρχουν τρείς κατηγορίες "Αιματολογικές", "Βιοχημικές" και Ορμονολογικές".

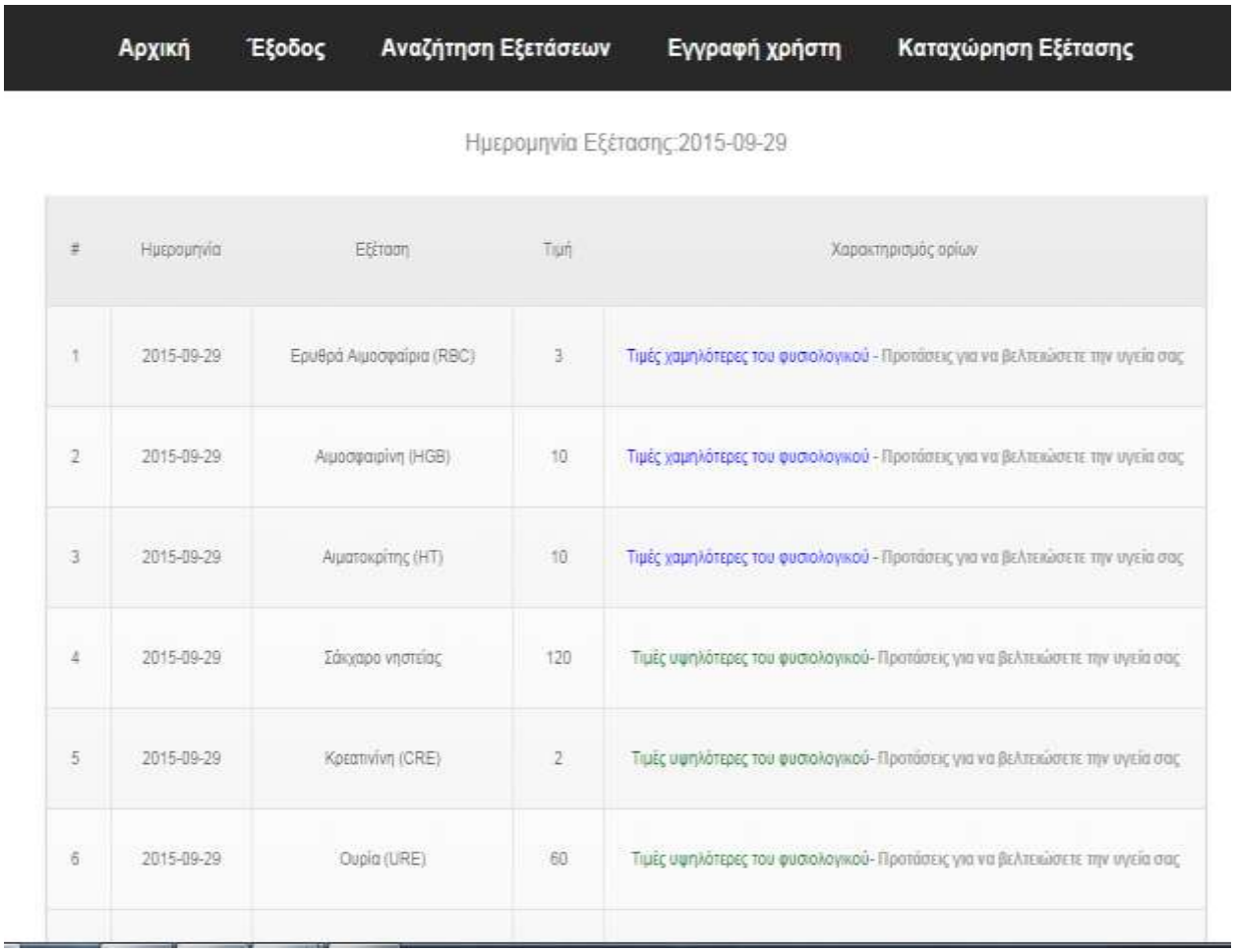
Στις τρείς κατηγορίες υπάρχουν διάφορα είδη εξετάσεων τις οποίες μπορεί να καταχωρήσει ο χρήστης, πληκτρολογώντας στο τέλος το πλήκτρο "Υποβολή".

3.2.5 Αναζήτηση Εξετάσεων

Σε αυτή την σελίδα ο χρήστης έχει την δυνατότητα να κάνει αναζήτηση εξετάσεων. Οι δύο τρόποι για α κάνει την αναζήτηση είναι βάσει της ημερομηνίας καταχώρησης, ή βάσει της κατηγορίας της εξέτασης.

3.2.6 Αποτελέσματα Εξετάσεων

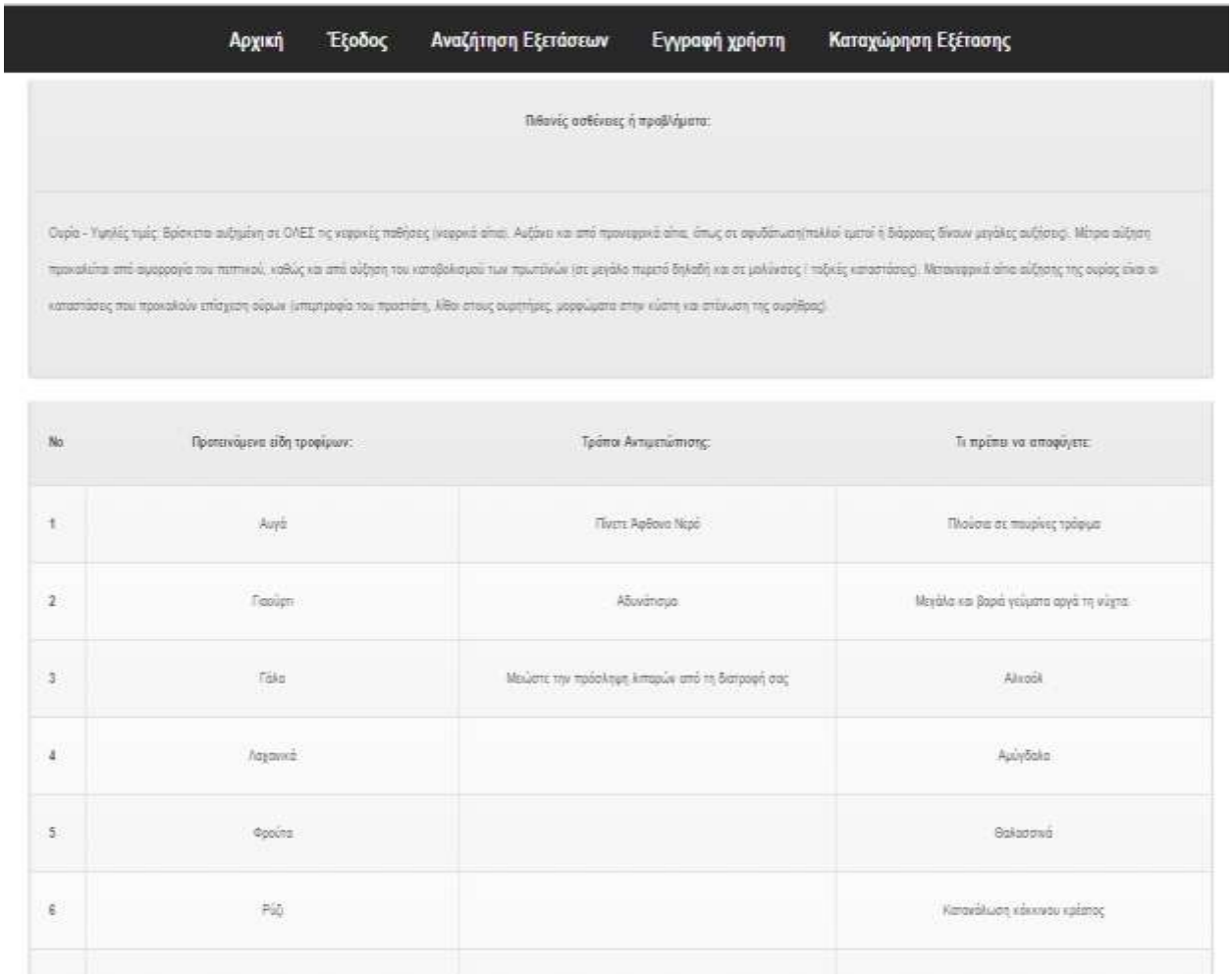
Στην προηγούμενη σελίδα θα επιλέξουμε να αναζητήσουμε εξετάσεις βάση της ημερομηνίας καταχώρησης των εξετάσεων. Θα επιλέξουμε μια ημερομηνία, για παράδειγμα " 2015-09-29" και θα πατήσουμε "Υποβολή".



Παρατηρούμε ότι τα αποτελέσματα δίνουν την εξής πληροφορία: "Ημερομηνία καταχώρησης", την ονομασία της εξέτασης, την τιμή της εξέτασης και τον χαρακτηρισμό των ορίων.

Παρατηρούμε επίσης ότι εάν οι τιμές είναι εκτός ορίων, υψηλότερες δηλαδή ή χαμηλότερες του φυσιολογικού, τότε στον χρήστη παρουσιάζεται ένας σύνδεσμος που θα τον παραπέμψει σε μια σελίδα με συμβουλές για το πως ο χρήστης μπορεί να βελτιώσει την υγεία του.

3.2.7 Συμβουλές προς χρήστη



Διαλέξαμε τυχαία έναν σύνδεσμο και παρατηρούμε τα εξής: την περιγραφή της ασθένειας, όπου περιγράφονται πιθανές ασθένειες ή προβλήματα που μπορούν να προκύψουν από τις τιμές αυτές.

Παρακάτω υπάρχουν τρείς στήλες, μία στήλη με τα τρόφιμα που ο χρήστης θα πρέπει να καταναλώσει για να βελτιώσει την υγεία του. Η δεύτερη στήλη περιέχει τους τρόπους με τους οποίους ο χρήστης μπορεί να αντιμετωπίσει το συγκεκριμένο πρόβλημα. Και τέλος η τρίτη στήλη περιέχει τα πράγματα τα οποία ο χρήστης πρέπει να αποφύγει να κάνει για να βελτιώσει την υγεία του.

Κεφάλαιο 4° - Έλεγχος

4.1 Εισαγωγή

4.1.1 Γενικοί ορισμοί

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

Ορίζουμε έτσι ως *Έλεγχο Λογισμικού (Software Testing)* τη διαδικασία εκτέλεσης του λογισμικού χρησιμοποιώντας ένα επιλεγμένο σύνολο από δεδομένα με σκοπό την ανίχνευση των ελαττωμάτων, την ανάλυση του κώδικα για την πρόληψη ελαττωμάτων και την αξιολόγηση της *ποιότητά του λογισμικού (software quality)*.

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

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

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

Εκτέλεση ελέγχου (test run) ονομάζεται η εκτέλεση ενός ή περισσότερων περιπτώσεων ελέγχου, ενώ *πακέτο σεναρίων ελέγχου* ή απλά *σενάρια ελέγχου* (*test procedure/test suite*) είναι ένα σύνολο περιπτώσεων ελέγχου, όπου το *εξαγόμενο αποτέλεσμα ελέγχου* (*output*) αποτελεί το *δεδομένο εισόδου* (*input*) της επόμενης περίπτωσης ελέγχου.

Η διαδικασία εκτέλεσης ενός συγκεκριμένου συνόλου περιπτώσεων ελέγχου πάνω στο αντικείμενο υπό έλεγχο με σκοπό την ανίχνευση σφαλμάτων, ονομάζεται *έλεγχος (test)*. Η οργάνωση, ο σχεδιασμός, η υλοποίηση και η ανάλυση ενός ελέγχου είναι βασικό κομμάτι του ελέγχου λογισμού (test ≤ testing).

Ο εντοπισμός κα η διόρθωση ελαττωμάτων *(debugging)* συνιστά δουλειά του προγραμματιστή και είναι κάτι εντελώς διαφορετικό από τον έλεγχο λογισμικού, γι'αυτό οι δύο αυτές έννοιες δεν πρέπει να συγχέονται.

4.1.2 Βασικές Αρχές

Οι επτά βασικές αρχές του Ελέγχου Λογισμικού είναι οι εξής:

1. Ο έλεγχος δείχνει την παρουσία ελαττωμάτων λογισμικού και όχι την απουσία τους. Μπορεί να καταδείξει ελαττώματα, αλλά δε μπορεί να αποδείξει ότι δεν υπάρχει κανένα άλλο ελάττωμα. Δηλαδή, ο έλεγχος μειώνει την πιθανότητα ύπαρξης κρυφών ελαττωμάτων στο λογισμικό, αλλά ακόμα και αν δεν ευρεθούν ελαττώματα, αυτό δε σημαίνει ότι δεν υπάρχουν.

2. Ο διεξοδικός έλεγχος είναι αδύνατος.

3. Οι διαδικασίες ελέγχου πρέπει να ξεκινάνε το νωρίτερο δυνατό. Οι δραστηριότητες του ελέγχου πρέπει να ενσωματώνονται στον κύκλο ζωής του λογισμικού (software lifecycle).

4. Τα ελαττώματα λογισμικού συγκεντρώνονται μαζεμένα (cluster). Η πιθανότητα ύπαρξης πρόσθετων σφαλμάτων σε ένα κομμάτι λογισμικού είναι συνήθως ανάλογη του συνόλου των σφαλμάτων που έχουν ανιχνευθεί εκεί.

5. Αν οι ίδιες περιπτώσεις ελέγχου τρέξουν επανειλημμένα στο αντικείμενο υπό έλεγχο, κάποια στιγμή δε βρίσκονται νέα ελαττώματα (*παράδοξο του παρασιτοκτόνου - pesticide paradox).* Για το λόγο αυτό, οι περιπτώσεις έλεγχου πρέπει να επανεξετάζονται και να διορθώνονται συστηματικά, και νέες περιπτώσεις ελέγχου πρέπει να γράφονται για τον έλεγχο διαφορετικών κομματιών του κώδικα ή του συστήματος.

6. Ο έλεγχος λογισμικού εξαρτάται από το περιεχόμενο και το περιβάλλον. Για κάθε προϊόν λογισμικού διαφορετικού περιεχομένου ακολουθείται διαφορετικός έλεγχος. Για παράδειγμα, το λογισμικό πρόγραμμα για τα τελωνεία μιας χώρας ελέγχεται διαφορετικά από μια σελίδα ηλεκτρονικού εμπορίου.

7. Το λάθος του «δεν υπάρχουν άλλα σφάλματα». Η ανίχνευση και η διόρθωση των ελαττωμάτων λογισμικού δε σημαίνει ότι το σύστημα είναι έτοιμο και ικανοποιεί τις προδιαγραφές και τις ανάγκες του πελάτη ή χρήστη.

4.1.3 Ποιότητα λογισμικού

Βασικός στόχος του ελέγχου λογισμικού είναι επίσης η βελτίωση της ποιότητας του λογισμικού, όχι μόνο ανιχνεύοντας τα σφάλματα και ελαχιστοποιώντας τα ελαττώματα του, αλλά αποκτώντας γνώση πάνω στα ποιοτικά χαρακτηριστικά του.

Σύμφωνα με το Πρότυπο *IEEE Std 729-1983* ως ποιότητα λογισμικού ορίζεται:

a. Το σύνολο γνωρισμάτων και χαρακτηριστικών ενός προϊόντος που εξυπηρετούν δεδομένες ανάγκες και προδιαγραφές.

b. Ο βαθμός που το λογισμικό κατέχει ένα επιθυμητό συνδυασμό από ιδιότητες.

c. Ο βαθμός στον οποίο ο πελάτης ή ο χρήστης αντιλαμβάνεται ότι το προϊόν ικανοποιεί τις προσδοκίες του.

d. Τα σύνθετα χαρακτηριστικά του λογισμικού τα οποία καθορίζουν το βαθμό που το λογισμικό υπό χρήση ικανοποιεί τις προσδοκίες του πελάτη.

Δεδομένου της αφηρημένης έννοιας του λογισμικού, ερευνητές αλλά και επαγγελματίες του χώρου αναζητούν τρόπους *μέτρησης της ποιότητας* (*metrics*) από τις αρχές της δεκαετίας του 70. Διάφορα μοντέλα ποιότητας λογισμικού έχουν αναπτυχτεί, από τα οποία έχουν επικρατήσει μέχρι σήμερα τα παρακάτω μοντέλα:

- McCall (1977)
- Boehm (1978)
- FURPS, FURPS+
- Dromey (1995)
- ISO 9000, ISO 9126 (1993)

4.1.3.1 Μοντέλο McCall

Το πιο παλιό αλλά και πιο διαδεδομένο μοντέλο, το μοντέλο McCall, δημιουργήθηκε για τις US Air force Electronic System Division (ESD), Rome Air Development Centre (RADC) και General Electric. Στο μοντέλο αυτό αναγνωρίστηκαν αρχικά 55 παράγοντες μεγάλης επιρροής στην ποιότητα λογισμικού, και στη συνέχεια 10 κατέληξαν, για λόγους οικονομίας, σε 11 *παράγοντες ποιότητας (quality factors)* , οι οποίοι είναι οι εξής:

1. *Συντηρησιμότητα (Maintainability)*
2. *Προσαρμοστικότητα (Flexibility)*
3. *Ελεγξιμότητα (Testability)*
4. *Μεταβιβασιμότητα (Portability)*
5. *Χρησικανότητα (Reusability)*
6. *Διασυνδεσιμότητα (Interoperability)*
7. *Σωστή λειτουργία (Correctness)*
8. *Αξιοπιστία (Reliability)*
9. *Αποτελεσματικότητα (Efficiency)*
10. *Ακεραιότητα (Integrity)*
11. *Χρησικανότητα (Usability).*

Το μοντέλο McCall αναπτύχθηκε για να προσδιορίσει τις σχέσεις μεταξύ των εξωτερικών παραγόντων και των κριτηρίων ποιότητας του προϊόντος. Για το λόγο αυτό, οι ποιοτικοί παράγοντες διαχωρίζονται σε τρεις βασικές κατηγορίες:

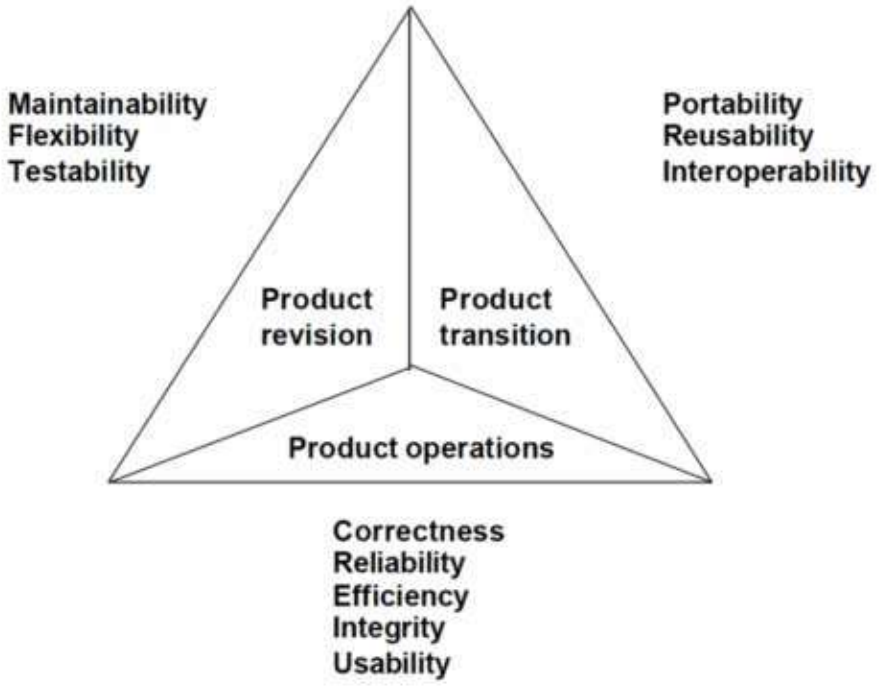


Figure 2. McCall's Quality Factors

Σχήμα 1.1: Απεικόνιση των ποιοτικών παραγόντων του μοντέλου McCall.

Σύμφωνα με την ικανότητα του λογισμικού να αλλάζει (*product revision*), διακρίνονται οι παρακάτω παράγοντες:

1. *Συντηρησιμότητα:* Η ικανότητα να ανιχνεύει και να διορθώνει τα ελαττώματα του.
2. *Προσαρμοστικότητα:* Η ικανότητα να αλλάζει σύμφωνα με της απαιτήσεις των προδιαγραφών.
3. *Ελεγξιμότητα:*Η ικανότητα να επικυρώνει τις προδιαγραφές του.

Επιπλέον, η προσαρμοστικότητά του προϊόντος σε νέα περιβάλλοντα (*product transition*) αναγνωρίζει τους εξής παράγοντες:

4. *Μεταβιβασιμότητα*: Η ικανότητα να μεταφέρεις το λογισμικό από ένα περιβάλλον σε ένα άλλο.
5. *Χρησικανότητα*: Η δυνατότητα να επαναχρησιμοποιηθούν υπάρχοντα κομμάτια λογισμικού σε διαφορετικά σημεία.
6. *Διασυνδεσιμότητα*: Ο βαθμός στον οποίο μπορούν να λειτουργήσουν μαζί διαφορετικά κομμάτια λογισμικού.

Τέλος τα βασικά χαρακτηριστικά διαχείρισης του προϊόντος (*product operations*). αφορούν τους παράγοντες:

7. *Σωστή λειτουργία:* Όλα τα χαρακτηριστικά ικανοποιούν τις προδιαγραφές του προιόντος.
8. *Αξιοπιστία:* ο βαθμός στον οποίο το σύστημα αποτυγχάνει.
9. Αποτελεσματικότητα: Σωστή χρήση των πόρων (CPU, μνήμη, δίκτυο) του συστήματος.
10. *Ακεραιότητα:* Προστασία από μη εγκεκριμένη πρόσβαση και χρήση.
11. *Χρησικανότητα:* Το μέτρο ικανότητας ενός προϊόντος να ανταπεξέλθει στις ανάγκες των χρηστών και η ευκολία χρήσης του.

Η μεγαλύτερη συνεισφορά του μοντέλου McCall είναι η σύνδεση μεταξύ ποιοτικών παραγόντων και μετρικών. Παρόλα αυτά, το μοντέλο δεν λαμβάνει άμεσα υπόψη του τη *λειτουργικότητα (functionality)* του προϊόντος λογισμικού.

4.1.3.2 Μοντέλο Boehm

Το μοντέλο Boehm αναπτύχθηκε το 1978 με βάση το μοντέλο McCall, προσθέτοντας ένα δεύτερο σύνολο ποιοτικών παραγόντων με έμφαση στη συντηρησιμότητα. Η επιπλέον ποιοτικοί παράγοντες του συγκεκριμένου μοντέλου είναι:

- *Σαφήνεια (Clarity)*
- *Επεξεργασιμότητα (Modifiability)*
- *Επίσημα έγγραφα (Documentation)*
- *Ανθεκτικότητα στην αποτυχία (Resilience)*
- *Κατανόηση (Understandability)*
- *Εγκυρότητα (Validity)*

- *Λειτουργικότητα (Functionality)*
- *Γενικότητα (Generality)*
- *Εξοικονόμηση (Economy)*

Ο στόχος του μοντέλου αυτού ήταν να προσδιορίσει τα ελαττώματα των μοντέλων που αυτόματα και ποσοτικά αξιολογούν την ποιότητα του λογισμικού. Το μοντέλο αναπαριστά τα χαρακτηριστικά του λογισμικού ιεραρχικά και με αναφορά στη χρησικανότητα του προγράμματος ή της εφαρμογής. Ωστόσο, το μοντέλο περιέχει μόνο διαγράμματα χωρίς καμία αναφορά σε μέτρηση των ποιοτικών παραγόντων.

4.1.3.3 Μοντέλο FURPS και FURPS+

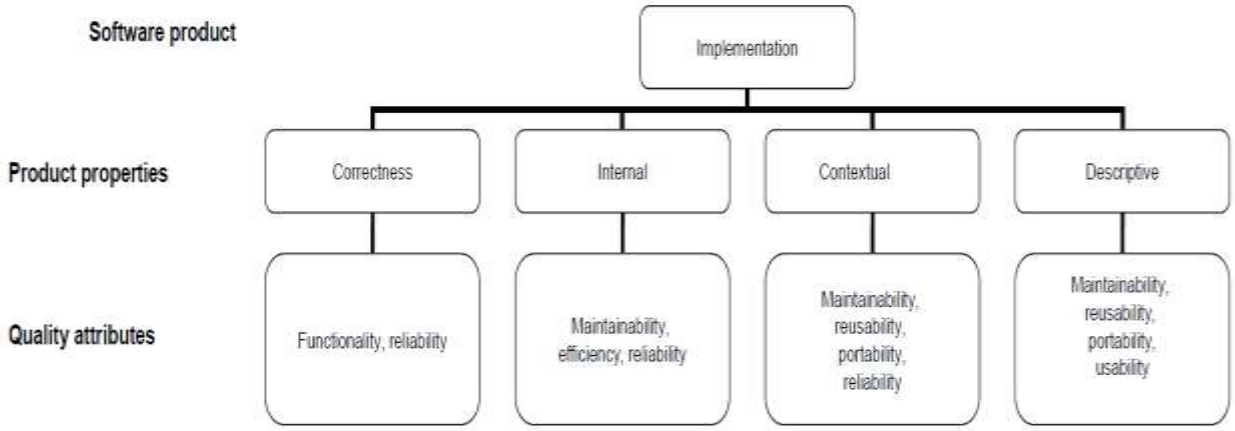
Το μοντέλο FURPS προτάθηκε από τον Robert Grady και την εταιρία Hewlett-Packard Co. Τα χαρακτηριστικά του ταξινομούνται σε δύο κατηγορίες, σύμφωνα με τις λειτουργικές και μη λειτουργικές προϋποθέσεις.

Λειτουργικές προϋποθέσεις (functional requirements) ορίζονται τα εισερχόμενα και τα αναμενόμενα εξερχόμενα, ενώ μη λειτουργικές προϋποθέσεις (non-functional requirements) θεωρούνται η χρησικανότητα, η αξιοπιστία, η επίδοση (performance) και η υποστήριξη (supportability).

Στη συνέχεια το μοντέλο επεκτάθηκε από την IBM Rational Software σε FURPS+. Στην τελική του μορφή το μοντέλο λαμβάνει υπόψη του τις προϋποθέσεις του χρήστη αλλά αψηφά τις απαιτήσεις του προγραμματιστή. Επιπλέον, αποτυγχάνει να συμπεριλάβει σημαντικά χαρακτηριστικά του προϊόντος όπως τη μεταβιβασιμότητα και τη συντηρησιμότητα.

4.1.3.4 Μοντέλο Dromey

Το μοντέλο Dromey παρουσιάστηκε πρώτη φορά το 1995 από τον R. Geoff Dromey και αναφέρει ότι η εξέλιξη κάθε προϊόντος λογισμικού είναι διαφορετική, με την δυναμική προσέγγιση του μοντέλου να είναι απαραίτητη. Για το λόγο αυτό, η βασική ιδέα του μοντέλου είναι να είναι αρκετά ευρύ ώστε να εφαρμόζεται σε διαφορετικά συστήματα. Το μοντέλο ορίζει 2 επίπεδα χαρακτηριστικών ποιότητας, τα ανώτερα χαρακτηριστικά (high-level attributes) και τα κατώτερα χαρακτηριστικά (subordinate attributes), επιδιώκοντας να αυξήσει την κατανόηση μεταξύ των δύο επιπέδων χαρακτηριστικών. Επίσης, προσπαθεί να συνδέσει τα γνωρίσματα του προϊόντος λογισμικού και τα γνωρίσματα ποιότητας λογισμικού.

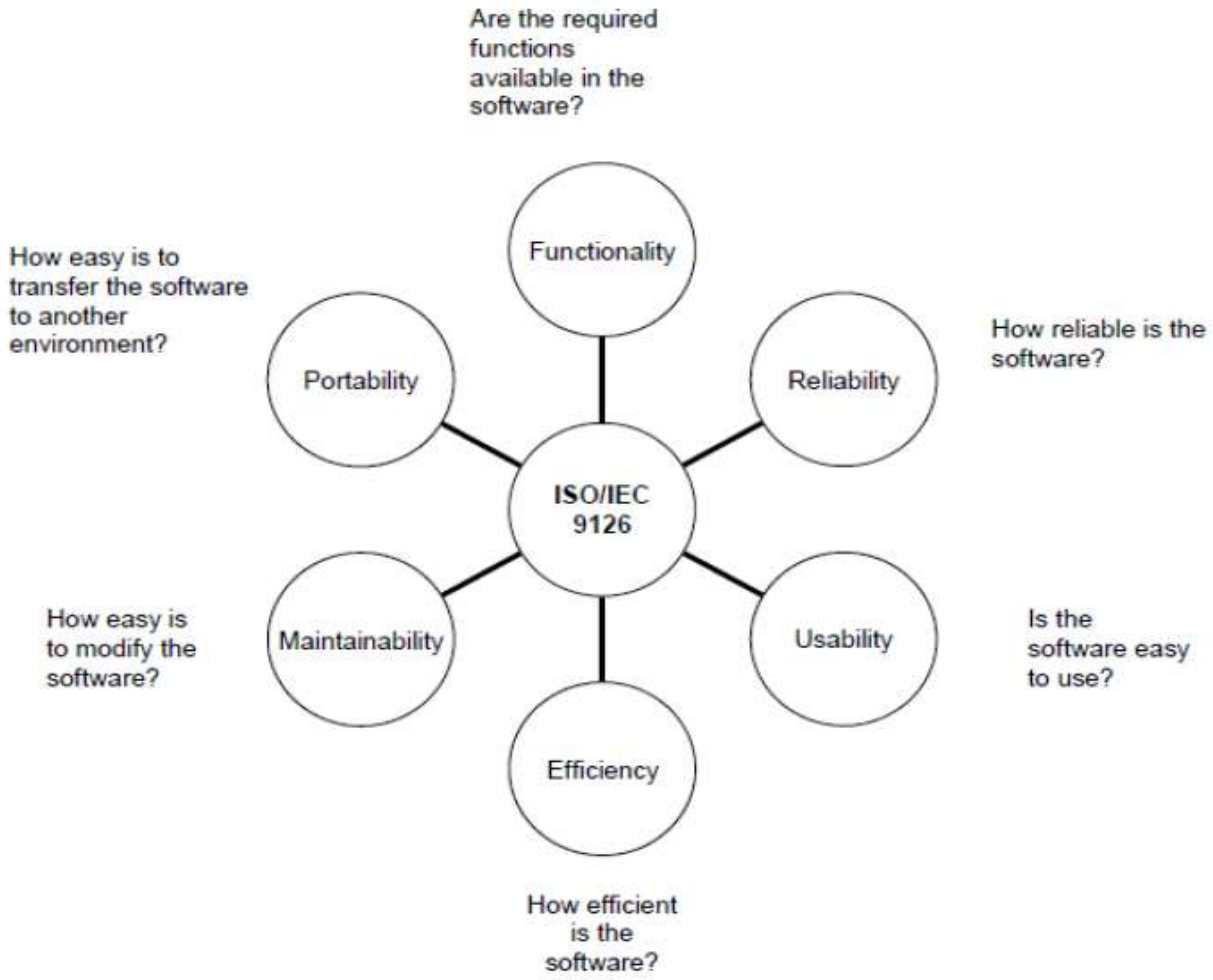


Σχήμα 1.2: Σχεδιαγραμματική απεικόνιση των γνωρισμάτων προϊόντος και γνωρισμάτων ποιότητας λογισμικού κατά Dromey.

4.1.3.5 Μοντέλο ISO

Με την ύπαρξη τόσο πολλών και διαφορετικών μοντέλων δεν άργησε να υπάρξει μια σύγχυση ως προς το πώς ορίζεται και μετράται η ποιότητα λογισμικού. Έτσι, η ISO/IEC JTC1 1 δημιούργησε σειρές προτύπων, όπως το ISO 9000 , το οποίο είναι το πιο παλιό και σημαντικό στο τομέα του *Ελέγχου Ποιότητας* (*Quality Assurance*). Επίσης το 1993 αναπτύχθηκε το πρότυπο ISO 9126: *Αξιολόγηση Προϊόντος Λογισμικού: Πρότυπο για τα Ποιοτικά Χαρακτηριστικά και Οδηγοί για τη Χρήση τους (Software Product Evaluation - Quality Characteristics and Guidelines for their Use – standard)*.

Το πρότυπο αυτό βασίστηκε στο μοντέλο McCall και Boehm προσθέτοντας ένα νέο παράγοντα, τη λειτουργικότητα (functionality), και αναγνωρίζοντας τα εσωτερικά και εξωτερικά χαρακτηριστικά ποιότητας του προϊόντος λογισμικού. Επίσης, διαχωρίζει τους ποιοτικούς παράγοντες σε έξι κατηγορίες: Λειτουργικότητα, Αξιοπιστία, Αποτελεσματικότητα, Συντηρησιμότητα, Μεταβιβασιμότητα και Χρησικανότητα.

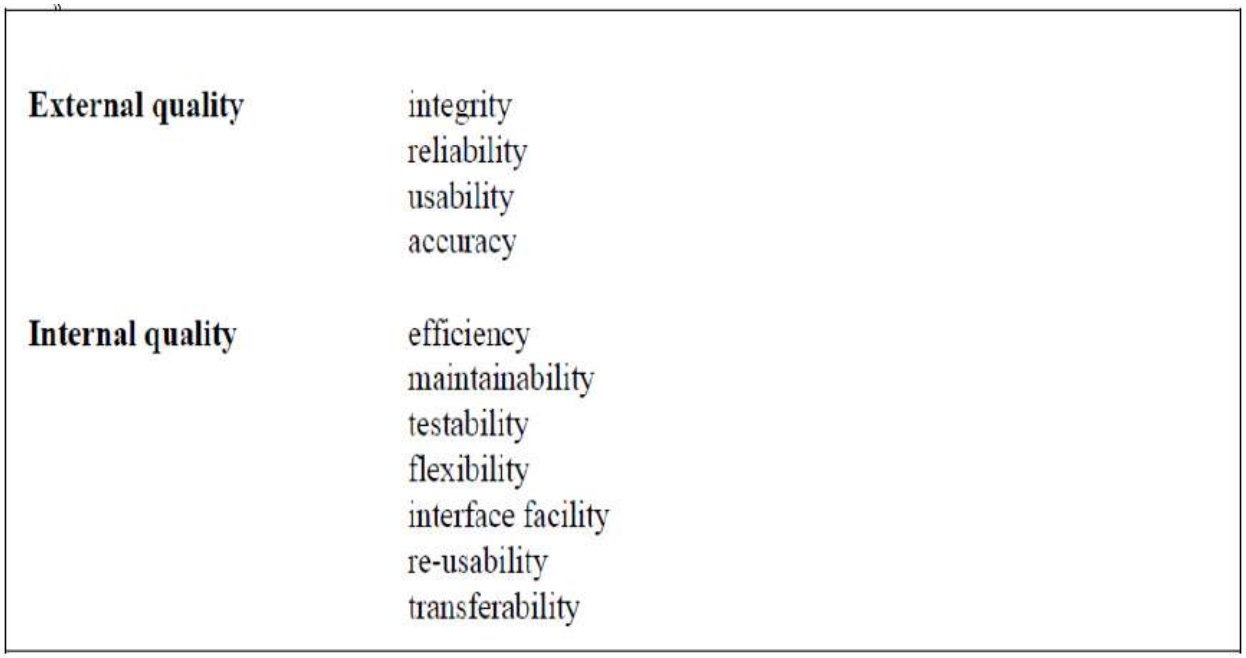


Σχήμα 1.3: Σχεδιαγραμματική απεικόνιση των ποιοτικών παραγόντων λογισμικού κατά ISO 9126.

Criteria/goals	*McCall, 1977*	*Boehm, 1978*	*ISO 9126, 1993*
Correctness	*	*	maintainability
Reliability	*	*	*
Integrity	*	*	
Usability	*	*	*
Effiency	*	*	*
Maintainability	*	*	*
Testability	*		maintainability
Interoperability	*		
Flexibility	*	*	
Reusability	*	*	
Portability	*	*	*
Clarity		*	
Modifiability		*	maintainability
Documentation		*	
Resilience		*	
Understandability		*	
Validity		*	maintainability
Functionality			*
Generality		*	
Economy		*	

Πίνακας 1: Συγκριτική παρουσίαση παραγόντων ποιότητας λογισμικού.

Ολοκληρώνοντας τις κατηγοριοποιήσεις των παραγόντων ποιότητας με βάση τα διάφορα κριτήρια που θέτουν τα μοντέλα, υπάρχει ένας ακόμα γενικός διαχωρισμός των παραγόντων, σε αυτά που συνδέονται με την *εξωτερική ποιότητα (external quality)* και την *εσωτερική ποιότητα (internal quality)*. Ως εξωτερική ποιότητα ορίζεται η ποιότητα του τελικού προϊόντος, δηλαδή η ποιότητα όπως εμφανίζεται στον κόσμο όταν τελειώνει από τη γραμμή παραγωγής. Από την άλλη πλευρά εσωτερική ονομάζεται η ποιότητα του προϊόντος όταν αυτό παράγεται, δηλαδή κατά τη διάρκεια της γραμμής παραγωγής. Γενικά, θα μπορούσαμε να πούμε ότι η εξωτερική ποιότητα συνίσταται στα χαρακτηριστικά που είναι σαφώς εμφανή στον τελικό χρήστη, ενώ εσωτερική ποιότητα θεωρείται το σύνολο των τεχνικών χαρακτηριστικών του λογισμικού. Σύμφωνα με τον Ghezzi *et al.*(1991), «σε γενικές γραμμές, οι χρήστες ενδιαφέρονται μόνο για την εξωτερική ποιότητα, αλλά η εσωτερική ποιότητα είναι αυτή που βοηθάει τους προγραμματιστές να πετύχουν την εξωτερική.»



Σχήμα 1.4: Κατηγοριοποίηση των ποιοτικών παραγόντων λογισμικού ως προς την εσωτερική και την εξωτερική ποιότητα του προϊόντος

4.2 Έλεγχος λογισμικού

4.2.1 Το V- Μοντέλο

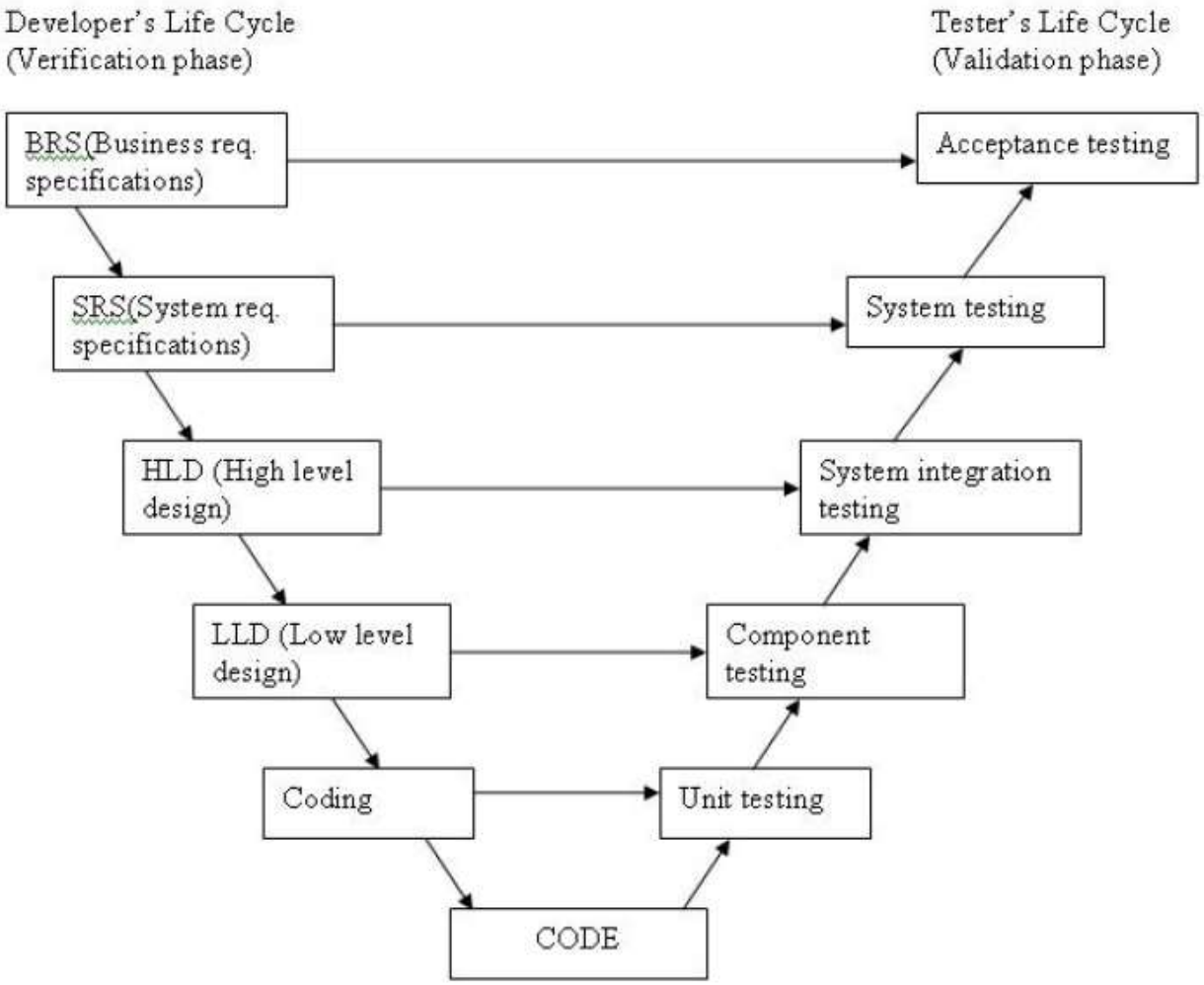
Όλες οι δραστηριότητες ελέγχου λογισμικού δεν υπάρχουν αποκλειστικά μόνες τους αλλά σε σχέση με τις δραστηριότητες ανάπτυξης λογισμικού. Εάν η μεθοδολογία της ανάπτυξης λογισμικού αλλάξει, τότε άμεσα αλλάζει και η προσέγγιση του ελέγχου λογισμικού. Ο κύκλος ζωής του λογισμικού αποτελεί μια διαδικασία που περιλαμβάνει το *σχεδιασμό*, τη *δημιουργία*, τον *έλεγχο*, την *εφαρμογή* και τη *συντήρηση* του προϊόντος *(planning, creating, testing, deploying and maintaining).* Στο κεφάλαιο αυτό περιγράφεται ο ρόλος του ελέγχου λογισμικού μέσα στο κύκλο ζωής του λογισμικού.

Μία από τις βασικές αλλά και πιο παλιές μεθοδολογίες ανάπτυξης λογισμικού είναι το μοντέλο Waterfall. Το μοντέλο αυτό περιγράφει τις δραστηριότητες του κύκλου ζωής του λογισμικού γραμμικά, από τις προδιαγραφές στην εφαρμογή και στη διατήρηση. Ο έλεγχος λογισμικού σε αυτό το μοντέλο είναι το τελικό βήμα. Μια βελτιωμένη έκδοση του Waterfall είναι το γενικό V-μοντέλο, το οποίο υποδεικνύει πως ο έλεγχος λογισμικού μπορεί να ενσωματωθεί σε κάθε φάση της διαδικασίας του κύκλου ζωής.

Το V-μοντέλο έχει τη μορφή του λατινικού γράμματος V. Τα δύο κλαδιά του συμβολίζουν την ισοδύναμη σημασία της ανάπτυξης και του ελέγχου λογισμικού. Το αριστερό τμήμα ορίζει τα επίπεδα ανάπτυξης από τη πλευρά του προγραμματιστή ενώ το δεξί τμήμα ορίζει τα επίπεδα ελέγχου για κάθε επίπεδο ανάπτυξης από τη πλευρά του *ειδικού ελέγχου (tester)*.

Τα επίπεδα ελέγχου σύμφωνα με αυτό το μοντέλο, είναι οι *Δοκιμασίες Μονάδας* (*Unit testing και Component testing*), οι *Δοκιμασίες Συνένωσης* (*Integration testing*), οι *Δοκιμασίες Συστήματος* (*System testing*) και οι *Δοκιμασίες Αποδοχής* (*Acceptance testing*).

Το unit testing είναι το ίδιο με το component testing με τη μόνη διαφορά ότι το δεύτερο μπορεί να καλεί και άλλες μονάδες με τη βοήθεια drivers. Συνήθως αποτελεί ένα επίπεδο ελέγχου.



Σχήμα 2.1: Σχηματική αναπαράσταση του γενικού V-μοντέλου.

Επίσης, το V-μοντέλο αντιπροσωπεύει τα verification και validation, δηλαδή την *επαλήθευση* και την *επικύρωση*, δυο σημαντικές διαδικασίες ελέγχου. Η *επαλήθευση (verification)* είναι η διαδικασία αξιολόγησης ενός συστήματος για το κατά πόσο αυτό ικανοποιεί τις συνθήκες-απαιτήσεις που τέθηκαν αρχικά. Η αξιολόγηση γίνεται με *φορμαλιστικές μεθόδους* (*formal methods*) και η διαδικασία απαντά στο ερώτημα «φτιάχνουμε το προϊόν σωστά;» ("are we building the product right?"). Από την άλλη πλευρά, η *επικύρωση (validation)* είναι η διαδικασία αξιολόγησης ενός συστήματος στο τέλος της ανάπτυξής του για να βεβαιώσουμε ότι είναι ελεύθερο σφαλμάτων και ικανοποιεί τις προδιαγραφές. Η αξιολόγηση αυτή γίνεται με δεδομένα και η διαδικασία απάντα στο ερώτημα «φτιάχνουμε το σωστό προϊόν;» ("are we building the right product?").

4.2.2 Επίπεδα ελέγχου

Όπως αναφέραμε και προηγουμένως, τα επίπεδα ελέγχου ενός λογισμικού προϊόντος είναι τέσσερα:

a) Οι *Δοκιμασίες Μονάδας* (*Unit testing και Component testing*) ελέγχουν εάν η μονάδα λογισμικού λειτουργεί σωστά και σύμφωνα με τις απαιτήσεις.

b) Οι *Δοκιμασίες Συνένωσης* (*Integration testing*) ελέγχουν εάν οι ομάδες μονάδων λογισμικού ανταποκρίνονται μεταξύ τους σωστά και σύμφωνα με το σχέδιο συστήματος (*technical system design*).

c) Οι *Δοκιμασίες Συστήματος* (*System testing*) επιβεβαιώνουν ότι το σύστημα ως ολότητα ανταποκρίνεται στις προδιαγραφές του.

d) Οι *Δοκιμασίες Αποδοχής* (*Acceptance testing*) ελέγχουν αν το σύστημα ικανοποιεί τις προδιαγραφές που τέθηκαν στο συμβόλαιο από τη πλευρά του τελικού χρήστη.

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

4.2.3 Δοκιμασίες Μονάδας

Σε αυτό το επίπεδο ελέγχονται οι μονάδες λογισμικού ξεχωριστά και σε απομόνωση από το υπόλοιπο σύστημα. Μονάδα θεωρείται το μικρότερο στοιχείο που μπορεί να μεταγλωτιστεί σε γλώσσα μηχανής (*compile*). Ανάλογα με τη γλώσσα προγραμματισμού, μονάδες μπορεί να είναι κλάσεις, συναρτήσεις, μέθοδοι ή ένα *σύνολο ανεξάρτητων κομματιών κώδικα (modules)*. Ο tester στην συγκεκριμένη περίπτωση ελέγχει τα εσωτερικά στοιχεία και τις συμπεριφορές των μονάδων. Οι δοκιμασίες μονάδας είναι το χαμηλότερο επίπεδο ελέγχου και ονομάζεται επίσης και δοκιμασίες προγραμματιστή καθώς συνήθως ο προγραμματιστής είναι αυτός που τις εκτελεί. Σε αντίθετη περίπτωση, ο tester έχει πρόσβαση στον κώδικα και εκτελεί τις δοκιμασίες.

Στις δοκιμασίες μονάδας, οι μονάδες αντικαθίστανται από stubs ή simulators και οι μονάδες που καλούνται μέσα σε αυτές αντικαθίστανται από drivers. *Stub* ονομάζεται μια εφαρμογή λογισμικού που έχει ως αποκλειστικό σκοπό να ελέγξει μια μονάδα. Ενώ, simulator ονομάζεται μια συσκευή, ένα πρόγραμμα υπολογιστή ή ένα σύστημα που χρησιμοποιείται κατά τη διάρκεια του testing και το οποίο συμπεριφέρεται ή λειτουργεί με τον ίδιο τρόπο με το σύστημα υπό έλεγχο, δηλαδή δέχεται τα ίδια εισερχόμενα δεδομένα και παράγει τα ίδια εξερχόμενα δεδομένα. Ως *driver* ή *test driver* ορίζεται μια μονάδα λογισμικού ή ένα *εργαλείο* (*test tool*) το οποίο αντικαθιστά μια μονάδα που αναλαμβάνει να καλεί μια άλλη μονάδα ή σύστημα.

Ο βασικός στόχος σε αυτό το επίπεδο είναι η εξασφάλιση εγγύησης ότι το αντικείμενο υπό έλεγχο εκτελεί τη λειτουργικότητα του σωστά και εξ ολοκλήρου έτσι όπως ορίζεται από τις προδιαγραφές (*functionality testing*). Δηλαδή, η μονάδα ελέγχεται με μια σειρά περιπτώσεων ελέγχου, όπου κάθε περίπτωση καλύπτει ένα συγκεκριμένο ζευγάρι εισερχόμενων/εξερχόμενων δεδομένων από όλους τους δυνατούς συνδυασμούς. Τα συνήθη ελαττώματα λογισμικού που αποκαλύπτονται μετά από τις *λειτουργικές δοκιμασίες μονάδας* (functionality component test) είναι λάθος υπολογισμοί και λάθος μονοπάτια κώδικα.

Μια άλλη πλευρά που καλύπτει αυτό το επίπεδο είναι οι δοκιμές ανθεκτικότητας (robustness testing) και οι *αρνητικές δοκιμές* (negative tests). Οι δοκιμές ανθεκτικότητας ελέγχουν το βαθμό στον οποίο μία μονάδα ή ένα σύστημα μπορεί να λειτουργήσει σωστά στην περίπτωση μη έγκυρων δεδομένων ή κακών συνθηκών του περιβάλλοντος. Οι αρνητικές δοκιμές στοχεύουν να αναδείξουν ότι μια μονάδα ή ένα σύστημα δεν δουλεύει. Αυτό γίνεται είτε με την *εισαγωγή μη έγκυρων δεδομένων ελέγχου*, τα οποία θα απορριφθούν από τη μονάδα ή το σύστημα (invalid testing), είτε με *δοκιμές εξαιρέσεων συστήματος* (system exceptions).

Η *διαχείριση λαθών* (*error/fault tolerance*) και η *διαχείριση εξαιρέσεων* (*exception handling*) συνίσταται στην ικανότητα της μονάδας να μπορεί να συνεχίσει τη λειτουργία της παρά την παρουσία λαθών και να διατηρεί ένα καλό επίπεδο επίδοσης. Είναι πολύ σημαντική διαδικασία και πρέπει να δοκιμάζεται όσο πιο νωρίς γίνεται ακόμα και στα μικρότερα κομμάτια λογισμικού.

Οι δοκιμασίες μονάδας ελέγχουν επίσης και μη λειτουργικά χαρακτηριστικά, που έχουν μεγάλη επίδραση στην ποιότητα της μονάδας και δε μπορούν να δοκιμαστούν σε ανώτερο επίπεδο ελέγχου. Τέτοια χαρακτηριστικά είναι η αποδοτικότητα και η συντηρησιμότητα. Οι δοκιμές αποδοτικότητας αποφαίνονται για το πόσο αποδοτικά η μονάδα χρησιμοποιεί τους πόρους του υπολογιστή, όπως χρήση μνήμης, *χρόνος απόκρισης* (*computing time*), χρόνος πρόσβασης σε δίσκο ή δίκτυο και πόσο χρόνο χρειάζονται οι συναρτήσεις και οι αλγόριθμοι της μονάδας να εκτελεστούν. Συνήθως, η αποδοτικότητα ελέγχεται σε υψίστης σημασίας μονάδες του συστήματος ή εάν το προσδιορίζουν οι απαιτήσεις του πελάτη.

Οι δοκιμές συντηρησιμότητας μετρούν πόσο εύκολο ή δύσκολο είναι να αλλάξεις το πρόγραμμα ή να συνεχίσεις να το αναπτύσσεις. Συνεπώς σημαντικοί παράγοντες για τη συντηρησιμότητα είναι η δομή του κώδικα, το *δομοστοιχείωση* (*modularity*), τα σχόλια στον κώδικα, η υπακοή στα πρότυπα, η κατανόηση και η επικαιροποίηση των σχετικών εγγράφων κ.ά. Αυτά τα χαρακτηριστικά δε μπορούν να ελεγχθούν με δυναμικό τρόπο, γι'αυτό χρησιμοποιούνται στατικοί μέθοδοι.

Όπως ήδη αναφέραμε είτε ο προγραμματιστής είτε ο tester έχει πρόσβαση στον κώδικα. Η μελέτη και η κατανόηση της εσωτερικής δομής της μονάδας, των συναρτήσεως και των μεταβλητών του κώδικα είναι ιδιαίτερα χρήσιμη για το σχεδιασμό περιπτώσεων ελέγχων. Συνεπώς, οι δοκιμασίες μονάδας απαιτούν από τον tester να έχει πρόσβαση στον κώδικα (ανήκουν στο white box έλεγχο λογισμικού), παρόλα αυτά μπορούν να γίνουν και με black box έλεγχο. Τέλος, σημαντική απόφαση στο επίπεδο αυτό είναι σε ποιές μονάδες θα πρέπει να γίνουν οι δοκιμασίες, σε μικρές βασικές μονάδες ή σε μεγαλύτερες μονάδες που προκύπτουν από συνενώσεις άλλων μονάδων.

4.2.4 Δοκιμασίες Συνένωσης

Συνένωση είναι η σύνθεση μίας ομάδας μονάδων για τη δημιουργία μιας μεγαλύτερης δομημένης μονάδας ή ενός υποσυστήματος. Ο στόχος των δοκιμασιών συνένωσης είναι να εκθέσουν τα ελαττώματα των διεπαφών (interfaces) και των διασυνδέσεων μεταξύ των συνενωμένων μονάδων, η εύρεση προβλημάτων συνεργασίας μεταξύ των μονάδων και η απομόνωση των αιτιών.

Για το επίπεδο αυτό χρησιμοποιούνται test drivers για τις δοκιμασίες, τα οποία ελέγχουν τα δεδομένα στο αντικείμενο υπό έλεγχο και λαμβάνουν τα αποτελέσματα. Συνήθως, αν οι δοκιμασίες μονάδας είναι καλά οργανωμένες , οι testers μπορούν να επαναχρησιμοποιήσουν τα ίδια test drivers. Σε αντίθετη περίπτωση επιβάλλεται η δημιουργία, αλλαγή ή επιδιόρθωση του περιβάλλοντος ελέγχου. Κατά τη διάρκεια των δοκιμασιών συνένωσης επιπλέον εργαλείων ελέγχου, τα επονομαζόμενα monitors, είναι απαραίτητα. *Monitor* είναι ένα εργαλείο λογισμικού ή μια *συσκευή τεχνικού εξοπλισμού (hardware device)* η οποία τρέχει ταυτόχρονα με τη μονάδα ή το σύστημα υπό έλεγχο και επιτηρεί, καταγράφει και αναλύει τη συμπεριφορά της μονάδας ή του συστήματος και την κίνηση μεταξύ δύο ή παραπάνω συστημάτων.

Σφάλματα και προβλήματα σε αυτό το επίπεδο μπορούν να βρεθούν μόνο με δυναμικό τρόπο. Επιπρόσθετα του λειτουργικού ελέγχου μπορούν να εκτελεσθούν μη λειτουργικές δοκιμές, όπως δοκιμές επίδοσης (performance testing) και η ικανότητας των διεπαφών. Μερικές φορές, οι δοκιμασίες μονάδας μπορούν να παραληφθούν, όμως το κόστος είναι λιγότερα σφάλματα και μεγαλύτερη δυσκολία στη διάγνωση των ελαττωμάτων. Έτσι, ο συνδυασμός δοκιμασιών μονάδας και συνένωσης είναι συχνά πιο αποτελεσματικός και αποδοτικός.

Επειδή όμως οι μονάδες υλοποιούνται σε διαφορετικό χρόνο, πρέπει να υπάρχει συγκεκριμένος σχεδιασμός της στρατηγικής συνένωσης έτσι ώστε να επιτευχθεί η πιο αποδοτική συνένωση. Η στρατηγική αυτή πρέπει να βελτιστοποιεί δύο παράγοντες: το χρόνο συνένωσης και το κόστος του περιβάλλοντος ελέγχου, λαμβάνοντας πάντα υπόψη την αρχιτεκτονική του συστήματος, το *σχέδιο έργου* (project plan) και το *σχέδιο ελέγχου* (test plan).

Ενδεικτικά παρουσιάζονται πέντε στρατηγικές δοκιμασιών συνένωσης. Στην πράξη επιλέγεται συνήθως ένας συνδυασμός αυτών:

a. Top-down συνένωση

Το test ξεκινάει με την *ανώτερη* (*top level*) μονάδα του συστήματος, η οποία καλεί άλλες μονάδες αλλά δεν καλεί τον εαυτό της, και στη συνέχεια προχωρά με τις *κατώτερες μονάδες* (*low level*) μέχρι την ολοκλήρωση του. Stubs αντικαθιστούν όλες τις κατώτερες μονάδες, ενώ test drivers δεν χρησιμοποιούνται. Προτέρημα της στρατηγικής αυτής είναι η ευκολότερη εύρεση

χαμένου branch link, ενώ βασικό μειονέκτημα της είναι το αυστηρά ιεραρχικό πρόγραμμα συστήματος που απαιτείται χωρίς να συναντάται εύκολα.

b. Bottom-up συνένωση

Η στρατηγική αυτή ξεκινά με τις κατώτερες βασικές μονάδες οι οποίες δεν καλούν άλλες μονάδες. Μεγαλύτερα υποσυστήματα συναθροίζονται βήμα-βήμα μετά από κάθε δοκιμασία συνένωσης. Δε χρησιμοποιούνται stubs αλλά οι ανώτερες μονάδες προσομοιώνονται από test drivers. Αυστηρά ιεραρχικό πρόγραμμα συστήματος απαιτείται και από αυτή τη στρατηγική αλλά δε συναντάται εύκολα. Βασικό της προτέρημα είναι ότι τα ελαττώματα λογισμικού βρίσκονται εύκολα.

Sandwich testing λέγεται η στρατηγική δοκιμασιών συνένωσης που συνδυάζει τις δυο στρατηγικές Top-down και Bottom-up.

c. Ad-hoc συνένωση

Οι μονάδες συνενώνονται για τις δοκιμασίες με τη σειρά με την οποία αναπτύχθηκαν. Αυτό γλιτώνει χρόνο, όμως stubs και test drivers είναι απαραίτητα.

d. Backbone συνένωση

Μια «ραχοκοκαλιά» χτίζεται, πάνω στην οποία συνενώνονται διαδοχικά οι μονάδες με τυχαία σειρά. Απαιτείται μία labor intensive skeleton.

e. Big-bang συνένωση

Με αυτή τη προσέγγιση, οι μονάδες συνενώνονται σε ένα βήμα, όταν όλα τα στοιχεία έχουν αναπτυχθεί, για να αποτελέσουν ένα ολοκληρωμένο σύστημα ή ένα μεγάλο μέρος του συστήματος. Η στρατηγική αυτή συμφέρει από κόστος χρόνου, αλλά υπάρχει μεγάλη επικινδυνότητα αποτυχίας καθώς βασίζεται στις δοκιμασίες μονάδας και περιμένει ελάχιστα προβλήματα σε αυτές. Σε περίπτωση που οι δοκιμασίες μονάδες έχουν γίνει για όλες τις μονάδες με σωστά αποτελέσματα, τότε σε αυτό το επίπεδο μπορούν να ανιχνευτούν σημαντικά προβλήματα που προκαλούνται από τη διασύνδεση των μονάδων με το περιβάλλον.

4.2.5 Δοκιμασίες Συστήματος

Οι δοκιμασίες συστήματος ελέγχουν αν το συνενωμένο σύστημα πληρεί τις απαιτήσεις και τις προδιαγραφές. Αυτό σημαίνει ότι ελέγχουν το σύστημα από τη πλευρά του πελάτη ή του μελλοντικού χρήστη, σε αντίθεση με τα κατώτερα επίπεδα ελέγχου που το εξέταζαν τεχνικά. Οι δοκιμασίες συστήματος αντιμετωπίζουν το προϊόν ή το σύστημα σαν ολότητα και έτσι πολλές λειτουργίες και χαρακτηριστικά συστήματος είναι ορατά μόνο σε αυτό το επίπεδο.

Οι δοκιμασίες που εκτελούνται σε αυτό το επίπεδο βασίζονται πάνω στις απαιτήσεις, προδιαγραφές, *διαδικασίες* (*business processes*), *περιπτώσεις χρήσης* (*use cases)* ή σε διάφορες συνθήκες με τις συμπεριφορές και τους πόρους του συστήματος, τις αλληλεπιδράσεις μεταξύ των διαφορετικών συστημάτων και τις αναλύσεις επικινδυνότητας. Γίνεται έλεγχος τόσο λειτουργικών όσο και μη λειτουργικών απαιτήσεων (functional και non-functional requirements). Τα συνήθη μη λειτουργικά χαρακτηριστικά που εξετάζονται είναι η οι επιδόσεις και η αξιοπιστία, ενώ τα λειτουργικά εξετάζονται συνήθως με τη μέθοδο black box.

Οι δοκιμασίες λαμβάνουν χώρα σε ένα ξεχωριστό ελεγχόμενο *περιβάλλον ελέγχου* (*test environment*), όμοιο με αυτό της παραγωγής (δηλαδή με ίδιο τεχνικό εξοπλισμό, λειτουργικό σύστημα, δίκτυα, εξωτερικά συστήματα κ.ά.). Είναι απολύτως απαραίτητο οι δοκιμασίες να γίνουν σε ξεχωριστό περιβάλλον από αυτό του πελάτη, γιατί αλλιώς μπορούν να προκληθούν ζημιές. Τέλος, αυτό αποτελεί το τελευταίο επίπεδο ελέγχου που μπορεί να εκτελέσει ο ίδιος ο προγραμματιστής ενώ συνήθως γίνεται από μια ανεξάρτητη *ομάδα ελέγχου (testing team).*

4.2.6 Δοκιμασίες Αποδοχής

Όταν η διαδικασία ανάπτυξης λογισμικού έχει τελειώσει και οι δοκιμασίες συστήματος έχουν αποφέρει ελαττώματα τα οποία έχουν διορθωθεί, το προϊόν ή το σύστημα είναι έτοιμο για το τελευταίο επίπεδο ελέγχου, τις δοκιμασίες αποδοχής. Πολλοί οργανισμοί ονομάζουν τον έλεγχο πριν και μετά τη μεταφορά στο περιβάλλον του πελάτη Factory *Acceptance Testing* (FAT) και *Site Acceptance Testing* (SAT).

Υπεύθυνος του επιπέδου αυτού είναι συνήθως μια ανεξάρτητη ομάδα ελέγχου, αλλά πολλές φορές, ανάλογα το προϊόν, σε αυτό το επίπεδο συμμετέχει και ο χρήστης ή ο πελάτης. Οι δοκιμασίες πρέπει να εκτελούνται σε ένα περιβάλλον ελέγχου *όσο περισσότερο κοντά στην παραγωγή* (“as-if production”). Στόχος του επιπέδου αυτού είναι να εδραιωθεί η εμπιστοσύνη προς το σύστημα, σε μέρος του συστήματος ή σε συγκεκριμένα μη λειτουργικά χαρακτηριστικά όπως η χρησικανότητα του συστήματος. Η εύρεση *ελαττωμάτων* (*defects*) δεν είναι ο βασικός στόχος του ελέγχου, αντίθετα στοχεύει στην επικύρωση και αξιολόγηση του συστήματος για το κατα πόσον είναι έτοιμο για *ανάθεση στο server (deployment)* του πελάτη και χρήση.

Γνωστές κατηγορίες δοκιμασιών αποδοχής είναι:

a. *Δοκιμασίες αποδοχής του χρήστη* (*User acceptance test*)

Αξιολογείται η ευκολία χρήσης του προϊόντος ή του συστήματος από τους τελικούς χρήστες. Προτείνεται όταν ο πελάτης και ο χρήστης είναι διαφορετικά πρόσωπα και πρέπει να λαμβάνονται υπόψη όλες οι κατηγορίες χρηστών με ανάλογες δοκιμασίες για την κάθε κατηγορία. Συνήθως εκτελείται από τους χρήστες ή από διαχειριστές εφαρμογών (application managers).

b. *Λειτουργικές δοκιμασίες αποδοχής* (*Operational acceptance test*)

Διασφαλίζουν την αποδοχή του συστήματος από το διαχειριστή συστήματος (system administrator). Εξετάζονται παράγοντες όπως backup/restore, ανάκτηση σε περίπτωση καταστροφής (disaster recovery), εργασίες διατήρησης και περιοδικοί έλεγχοι για αδυναμίες ασφάλειας.

c. *Διαδικασίες δοκιμασίας κατά το συμβόλαιο* (Contract acceptance testing)

Ελέγχεται αν οι όροι του συμβολαίου έχουν τηρηθεί. Ειδικά κριτήρια και test cases σχεδιάζονται με βάση το συμβόλαιο.

d. *Διαδικασίες αποδοχής συμβατότητας* (*Compliance acceptance testing*)

Το προϊόν ή το σύστημα δοκιμάζεται αν τηρεί τα τους κανονισμούς , τους νόμους και τους κανόνες ασφάλειας.

e. *Alpha και Beta δοκιμασίες αποδοχής* (*Alpha and Beta testing*)

Αν το προϊόν είναι *ευρέως χρήσεως* (*COTS: customers off-the-shelf software*) δεν είναι πάντα εφικτό να εκτελεστούν οι δοκιμασίες αποδοχής. Τότε επιστρατεύονται οι ίδιοι οι χρήστες για να παρέχουν γνώμες, σχόλια και εντυπώσεις. Alpha testing ονομάζονται οι δοκιμασίες που εκτελούνται στο περιβάλλον του παραγωγού/προγραμματιστή ενώ Beta testing είναι οι δοκιμασίες που γίνονται στο περιβάλλον του πελάτη.

4.2.7 Γενικοί τύποι ελέγχου

Οι έλεγχοι μπορούν να διακριθούν στους παρακάτω τύπους:

4.2.7.1 Λειτουργικός έλεγχος

Λειτουργικός αποκαλείται κάθε έλεγχος που επαληθεύει την αναμενόμενη συμπεριφορά των δεδομένων εισόδου-εξόδου. Για τον σχεδιασμό των σχετικών περιπτώσεων ελέγχου, ο tester χρησιμοποιεί κυρίως την black box μέθοδο (black box method).

Στον *έλεγχο βάσει απαιτήσεων (requirements-based testing)*, ως βάση ελέγχου χρησιμοποιούνται οι *απαιτήσεις* (requirements). Στις προδιαγραφές ελέγχου, σχεδιάζεται και τεκμηριώνεται τουλάχιστον μια περίπτωση ελέγχου, ενώ συνήθως χρειάζονται περισσότερες από μία περιπτώσεις για να ελεγχθεί μια λειτουργική απαίτηση. Αν όλες οι ορισμένες περιπτώσεις ελέγχου έχουν τρέξει χωρίς κάποια αποτυχία, τότε η ανάλογη λειτουργικότητα θεωρείται επικυρωμένη (validated functionality). Αυτός ο τύπος ελέγχου χρησιμοποιείται κυρίως για τις δοκιμασίες συστήματος και αποδοχής.

Στον *έλεγχο βάσει επιχειρησιακών διαδικασιών (business- process-based testing)*, ο σχεδιασμός ελέγχου βασίζεται στην περιγραφή και στη γνώση των διαδικασιών. Μετά από αυτό, οι περιπτώσεις και τα σενάρια ελέγχου κατασκευάζονται βάσει αυτών, ενώ η προτεραιότητα ιεραρχείται βάσει της συχνότητας και της σημαντικότητας της διαδικασίας.

Οι διαφορά των δύο τύπων ελέγχου έγκειται στο γεγονός ότι ο έλεγχος βάσει απαιτήσεων εστιάζει σε λειτουργίες ενιαίου συστήματος, ενώ ο έλεγχος βάσει επιχειρησιακών διαδικασιών επικεντρώνεται στη συνολική, πολυβηματική διαδικασία.

4.2.7.2 Μη λειτουργικός έλεγχος

Οι μη λειτουργικές απαιτήσεις περιγράφουν τα γνωρίσματα ενός συστήματος ως ενιαίου συνόλου, καθώς και το γνώρισμα της λειτουργικής συμπεριφοράς. Χαρακτηριστικά των απαιτήσεων αυτών αποτελούν η αξιοπιστία, η χρησικανότητα και η αποτελεσματικότητα. Τα παρακάτω μη λειτουργικά χαρακτηριστικά ενός συστήματος πρέπει να λαμβάνονται υπόψη στους ελέγχους, συνήθως στις δοκιμασίες του συστήματος:

- *Έλεγχος φόρτου (Load testing)*
- *Έλεγχος επίδοσης (Performance testing)*
- *Έλεγχος όγκου (Volume testing)*
- *Έλεγχος έντασης (Stress testing)*
- *Έλεγχος ασφάλειας (Security testing)*
- *Έλεγχος σταθερότητας (Stability testing)*
- *Έλεγχος ανθεκτικότητας (Robustness testing)*
- *Έλεγχος συμβατότητας και μετατροπής δεδομένων (Testing for compatibility and data conversion)*
- *Έλεγχος διαφορετικών συνθέσεων (Testing of different configurations)*
- *Έλεγχος χρησικανότητας (Usability testing)*
- *Έλεγχος για δυνατότητα τεκμηρίωσης (Checking for the documentation)*
- *Έλεγχος συντηρησιμότητας (Checking for maintainability)*

Το κύριο πρόβλημα των μη λειτουργικών χαρακτηριστικών είναι ότι συνήθως προκύπτουν από ανακριβείς απαιτήσεις και συνεπώς δεν είναι ελέγξιμα. Συνεπώς, σκοπός του tester αποτελεί η εξασφάλιση ότι οι μη λειτουργικές απαιτήσεις είναι μετρήσιμες και η δημιουργία κατάλληλων μετρικών.

4.2.7.3 Έλεγχος δομών λογισμικού ή δομικός έλεγχος

Οι δομικές τεχνικές (*structural testing*) χρησιμοποιούν πληροφορίες απ' την εσωτερική δομή κι αρχιτεκτονική του κώδικα του αντικειμένου υπό έλεγχο (white box έλεγχος). Μπορούν να χρησιμοποιηθούν δηλώσεις, ιεραρχία, δομές επιλογών μενού και αφηρημένα μοντέλα του λογισμικού, όπως το μοντέλο ροής διαδικασιών ή το μοντέλο μεταβατικής κατάστασης. Σκοπός του δομικού ελέγχου είναι η κάλυψη όλων των δομικών στοιχείων και συνήθως χρησιμοποιείται στις δοκιμασίες μονάδας και συνένωσης.

4.2.7.4 Έλεγχος αλλαγών και έλεγχος παλινδρόμησης

Αν το υπάρχον λογισμικό τροποποιηθεί, τα αλλαγμένα μέρη πρέπει να επανελεγχθούν και οι υπάρχοντες περιπτώσεις ελέγχου πρέπει να επαναληφθούν μαζί με τα καινούρια μέρη που θα ελεγχθούν πρώτη φορά. Ο *έλεγχος παλινδρόμησης (regression testing)* είναι ένας επανέλεγχος ενός ήδη ελεγμένου προγράμματος, που έχει υποστεί τροποποιήσεις, με σκοπό να εξασφαλιστεί η μη εισαγωγή νέων ή η αποκάλυψη υπαρχόντων σφαλμάτων ως αποτέλεσμα των αλλαγών. Ο έλεγχος παλινδρόμησης μπορεί να πραγματοποιηθεί σε όλα τα επίπεδα ελέγχου. Χρησιμοποιείται στον λειτουργικό, μη λειτουργικό και δομικό έλεγχο και ανάλογα με τον όγκο του ελέγχου μπορεί να διακριθεί σε:

- *Επανέλεγχος ελαττωμάτων και έλεγχος διάταξης (Defect retest and configuration testing)*
- *Έλεγχος τροποποιημένης λειτουργικότητας (Testing of altered functionality)*

- *Έλεγχος νέας λειτουργικότητας (Testing of new functionality)*
- *Έλεγχος πλήρους παλινδρόμισης (Complete regression test)*

Επειδή οι τροποποιήσεις του λογισμικού μπορούν να δημιουργήσουν παράπλευρες απώλειες αυθαίρετα σε άλλα μέρη του συστήματος, χρειάζεται ένας έλεγχος πλήρους παλινδρόμησης. Επειδή όμως αυτό είναι συνήθως χρονοβόρο και δαπανηρό, επιλέγεται ένα σύνολο περιπτώσεων παλινδρόμησης, ανάλογα με τις προτεραιότητες του προϊόντος. Παραλείπονται έτσι ορισμένες διαφοροποιήσεις και περιορίζονται οι έλεγχοι σε συγκεκριμένες διατάξεις, υποσυστήματα και επίπεδα ελέγχου.

4.2.7.5 Άλλοι τύποι ελέγχων

Εκτός από τους παραπάνω τύπους ελέγχου, υπάρχουν και άλλοι δύο σημαντικοί έλεγχοι που δεν υπόκεινται στις υπάρχουσες κατηγοριοποιήσεις.

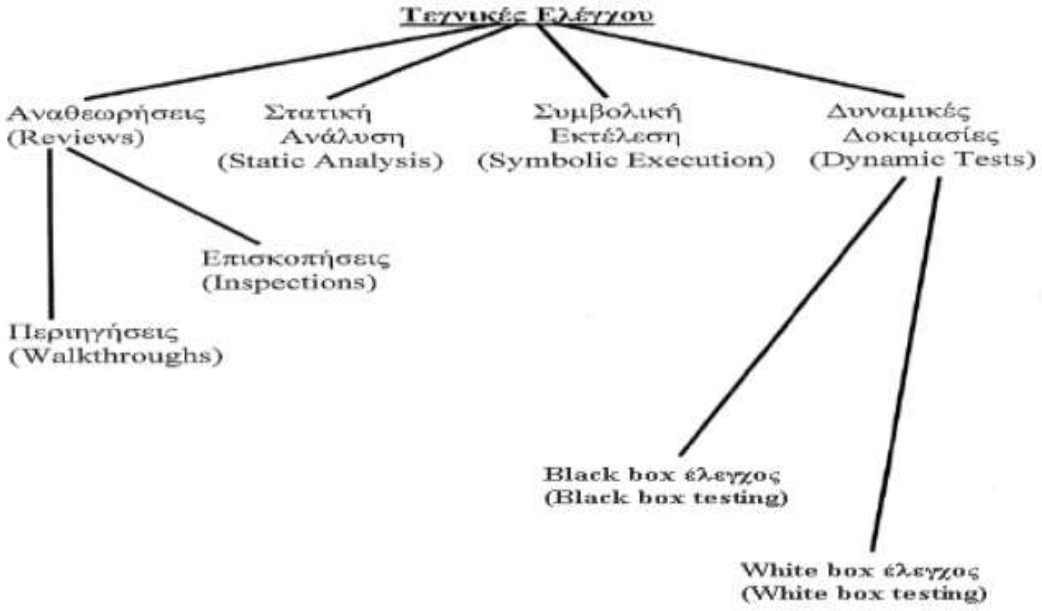
Με κριτήριο το χρονικό σημείο εκτέλεσης του ελέγχου ορίζουμε τον *έλεγχο καπνού (Smoke or Sanity testing)* ως τον προκαταρκτικό έλεγχο για την ανίχνευση σοβαρών σφαλμάτων που οδηγούν στην απόρριψη της νέας έκδοσης του λογισμικού. Αποτελείται από ένα υποσύνολο περιπτώσεων ελέγχου που καλύπτουν τις πιο σημαντικές λειτουργικότητες του προϊόντος.

Ο διερευνητικός έλεγχος (Explanatory testing) είναι ένα είδος προσέγγισης του ελέγχου λογισμικού κατά τον οποίο παίρνουμε πληροφορίες και γνώση για το σύστημα υπό έλεγχο, σχεδιάζοντας μικρούς ελέγχους και εκτελώντας τους ταυτόχρονα. Αυτός ο σύντομος κύκλος ελέγχου συνεχίζεται μέχρι ο tester να αναγνωρίσει όλες τις δυνατότητες και τους περιορισμούς του συστήματος. Τα στάδια του διερευνητικού ελέγχου είναι σχεδιασμός του ελέγχου, εκτέλεση και παρατήρηση για εξαγωγή γνώσης.

Ο σχεδιασμός ελέγχου περιέχει την αναγνώριση των περιπτώσεων ελέγχου χωρίς την καταγραφή τους σε επίσημα σενάρια ελέγχου. Ο σχεδιασμός αφορά ελέγχους για δεδομένα ή για καταστάσεις και βασίζεται σε όλες τις παραδοσιακές μεθόδους. Αφού προσδιοριστεί ένα μικρό σύνολο περιπτώσεων ελέγχου με βάση ένα συγκεκριμένο κριτήριο, στη συνέχεια αυτό εκτελείται. Εδώ έγκειται η διαφορά του διερευνητικού ελέγχου, δε χρειάζεται να σχεδιαστεί ένα μεγάλο σύνολο περιπτώσεων για να ελέγξει όλη την λειτουργικότητα σε κάποιο μελλοντικό έλεγχο. Κατά την εκτέλεση των περιπτώσεων ελέγχου παρατηρούμε την εφαρμογή για το τι κάνει και τι δεν κάνει. Ανακαλύπτουμε πως λειτουργεί, τις δυνατότητες της, τις ιδιομορφίες της και τους περιορισμούς της. Οι πληροφορίες της διερεύνησης προσφέρουν γνώση για το αντικείμενο υπό έλεγχο.

4.2.8 Τεχνικές ελέγχου

Υπάρχουν πολλές κατηγορίες *τεχνικών ελέγχου (test techniques),* η καθεμία ειδικεύεται στην εξεύρεση συγκεκριμένων ειδών ελαττωμάτων, με διαφορετικά πλεονεκτήματα και μειονεκτήματα. Γενικά, οι τεχνικές ελέγχου λογισμικού χωρίζονται σε δύο βασικές διαδικασίες, στατικές και δυναμικές.



Σχήμα 2.2: Κατηγοριοποίηση των τεχνικών ελέγχου λογισμικού.

4.2.8.1 Στατικές τεχνικές

Οι στατικές τεχνικές δεν εκτελούν το αντικείμενο υπό έλεγχο και χρησιμοποιούνται επιτυχώς για τη βελτίωση ποιότητας της διαδικασίας ελέγχου λογισμικού, είτε ελέγχοντας επίσημα έγγραφα είτε τον ίδιο τον κώδικα. Περιλαμβάνουν δύο βασικές κατηγορίες, τις αναθεωρήσεις και τη στατική ανάλυση. Η συμβολική εκτέλεση του κώδικα μπορεί να θεωρηθεί στατική μέθοδος, παρόλο που εκτελείται ο κώδικας.

Αναθεωρήσεις

Οι *αναθεωρήσεις (reviews)* περιλαμβάνουν δυο τρόπους ελέγχου, τις *περιηγήσεις (walkthroughs)* και τις *επισκοπήσεις (inspections).* Και οι δύο τρόποι ασχολούνται με την προσπάθεια ανεύρεσης των ελαττωμάτων διατρέχοντας τον κώδικα με την απλή ανάγνωση. Όσο και αν φαίνεται απλοϊκή μέθοδος, έχει παρατηρηθεί ότι πάρα πολλά σφάλματα έχουν ανιχνευτεί με απλή ανάγνωση του κώδικα και σε πολύ αρχικό στάδιο. Η διαφορά μεταξύ περιηγήσεων και επισκοπήσεων έγκειται στη διαδικασία που ακολουθείται (ποιος είναι υπεύθυνος, πως συνέρχονται οι ομάδες ελέγχου για συζήτηση των σφαλμάτων κλπ.).

Οι αναθεωρήσεις χρησιμοποιούνται επίσης συχνά στον έλεγχο επίσημων εγγράφων, όπως του σχεδίου συστήματος, *λειτουργικών προδιαγραφών* (*functional specifications*), *προδιαγραφών απαιτήσεων* (*requirement specifications*), περιπτώσεων χρήσης, σχεδίου ελέγχου, περιπτώσεων ελέγχων κ.ά.

Στατική ανάλυση

Η στατική ανάλυση (static analysis) χρησιμοποιείται για τον έλεγχο των προδιαγραφών απαιτήσεων, του σχεδίου λογισμικού και κυρίως του κώδικα. Ο έλεγχος γίνεται χωρίς την εκτέλεση του προγράμματος με δεδομένα. Η τεχνική εφαρμόζεται είτε χειρονακτικά είτε με την χρήση ειδικών αυτόματων εργαλείων, των *στατικών αναλυτών* (*static analyzers*).

Δε μπορούν όλα τα ελαττώματα λογισμικού να ανιχνευτούν με αυτή τη τεχνική, διότι μερικά χρειάζονται την εκτέλεση του κώδικα. Όμως πολλά προβλήματα μπορούν να αναγνωριστούν στα πλαίσια της στατικής ανάλυσης όπως:

a. Συντακτικά λάθη του κώδικα

Ο compiler είναι ένα εργαλείο στατικής ανάλυσης για αναγνώριση συντακτικών λαθών της γλώσσας προγραμματισμού, ο οποίος επιπλέον δημιουργεί μια λίστα με όλα τα διαφορετικά στοιχεία του κώδικα, ελέγχει αν είναι σωστοί οι τύποι των δεδομένων, αναγνωρίζει μη δηλωμένες μεταβλητές ή κώδικα που δε χρησιμοποιείται ποτέ, ελέγχει τα άνω και κάτω όρια των πεδίων κ.ά.

b. Απόκλιση από τα πρότυπα

Σε αυτή την φάση αναδιαμορφώνεται ο κώδικας για σωστή ταξινόμηση με τέτοιο τρόπο ώστε να τηρούνται οι προδιαγραφές και τα πρότυπα της γλώσσας προγραμματισμού και συγχρόνως να είναι ευκολότερη η ανάλυση του κώδικα.

c. Data flow ανωμαλίες

Με τη data flow ανάλυση ελέγχονται οι ανωμαλίες του κώδικα που ενδεχομένως να οδηγήσουν σε σφάλματα, χωρίς αυτό να είναι απαραίτητο. Τα ελαττώματα σε αυτές τις περιπτώσεις εμφανίζονται συνήθως στα δεδομένα που χρησιμοποιούνται στα διάφορα μονοπάτια του κώδικα. Έτσι, όλες οι μεταβλητές εξετάζονται λεπτομερώς κατά την ανάλυση τους.

Οι μεταβλητές χωρίζονται ανάλογα με τη χρήση ή την κατάσταση τους σε:

- *(d) Ορισμένες* (*defined*): Οι μεταβλητές έχουν λάβει τιμή.
- *(r) Αναφερόμενες (referenced*): Η τιμή της μεταβλητής διαβάζεται και/ή χρησιμοποιείται.
- *(u) Μη ορισμένες* (undefined): Οι μεταβλητές δεν έχουν λάβει ορισμένη τιμή.

Επομένως, μπορούμε να ορίσουμε τρεις τύπους data flow ανωμαλιών:

- *ur-ανωμαλία*: Μία μη ορισμένη τιμή (u) της μεταβλητής διαβάζεται στο μονοπάτι (r)
- *du-ανωμαλία*: Η μεταβλητή λαμβάνει τιμή (d) η οποία γίνεται μη έγκυρη/μη ορισμένη(u) χωρίς να έχει χρησιμοποιηθεί ακόμα.
- *dd-ανωμαλία*: Η μεταβλητή λαμβάνει τιμή για δεύτερη φορά (d) αλλά η πρώτη φορά δεν έχει χρησιμοποιηθεί ακόμα (d).

Οι data flow ανωμαλίες δεν είναι εύκολα ορατές, αλλά επίσης δεν οδηγούν πάντα σε σφάλματα.

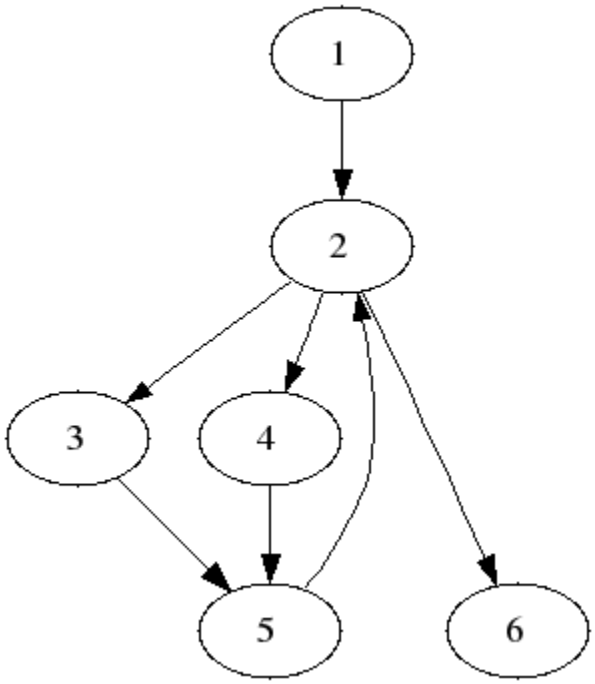
d. Control flow ανωμαλίες

Ο γράφος control flow είναι μια ακολουθία γεγονότων (μονοπάτι) κατά την εκτέλεση του κώδικα. Είναι ένας κατευθυνόμενος γράφος, όπου οι δηλώσεις (statements) του προγράμματος αναπαριστόνται με κόμβους. Χάρη στην σαφήνεια του γράφου, πιθανές ανωμαλίες είναι εύκολα αναγνωρίσιμες. Τέτοιες ανωμαλίες συνήθως είναι έξοδοι από βρόχο, κομμάτι κώδικα που έχει πολλές εξόδους κ.ά. Αν ο κώδικας είναι πολύ περίπλοκος, με τη βοήθεια του control flow γράφου επανεξετάζεται και απλοποιείται. Μερικοί στατικοί αναλυτές αυτής της μεθόδου πιθανόν να αναφέρουν και το συνολικό αριθμό των μονοπατιών που ενδεχομένως έχει το πρόγραμμα προς ανάλυση.

Τέλος ένας άλλος στόχος της στατικής ανάλυσης είναι η μέτρηση ποιοτικών χαρακτηριστικών μέσω των μετρικών. Οι μετρικές βοηθούν στο να μετρούνται ποσοτικά διάφοροι παράγοντες του λογισμικού. Για παράδειγμα, η *μετρική κυκλωματικής πολυπλοκότητας* (*cyclomatic complexity metric*) είναι μια μετρική που μετρά την πολυπλοκότητα της δομής του κώδικα μέσω του control flow γράφου. Υπολογίζει δηλαδή τον αριθμό των ανεξάρτητων μονοπατιών με την βοήθεια του τύπου:

$$v(G) = e - n + 2$$

όπου *e* ο αριθμός των ακμών και *n* ο αριθμός των κόμβων.



Σχήμα 2.3: Control flow γράφος αναπαράστασης κομματιού κώδικα

Ένα κομμάτι κώδικα αναπαριστάται από τον control flow γράφο της εικόνας, που ξεκινά από τον κόμβο 1 (αρχική κατάσταση) για να καταλήξει στον κόμβο 6 (τελική κατάσταση). Η μετρική κυκλωματικής πολυπλοκότητας υπολογίζεται ως εξής: e = 7 και n = 6 οπότε

$$v(G) = e - n + 2 = 7 - 6 + 2 = 3 \text{ ανεξάρτητα μονοπάτια.}$$

Όσο μεγαλύτερος είναι ο αριθμός κυκλωματικής περιπλοκότητας, τόσο πιο περίπλοκο και δύσκολο είναι να καταλάβεις ένα κομμάτι κώδικα. Συνεπώς, η μετρική αυτή παρέχει σημαντική πληροφορία για το μέγεθος του ελέγχου που θα ακολουθήσει και χρησιμοποιείται κυρίως για να εκτιμήσει την *ελεγξιμότητα* (*testability*) και τη συντηρησιμότητα του συστήματος.

Συμβολική εκτέλεση κώδικα

Συμβολική εκτέλεση κώδικα (*symbolic execution*) είναι η εκτέλεση ολόκληρου του κώδικα (όλων των γραμμών) με συμβολικές τιμές, διαφορετικές από τις πραγματικές τιμές εισόδου τις οποίες απαιτεί η κανονική εκτέλεση του.

4.2.8.2 Δυναμικές τεχνικές

Οι δυναμικές τεχνικές ελέγχου παρέχουν στο αντικείμενο υπό έλεγχο δεδομένα εισόδου, στη συνέχεια αυτό εκτελείται και τέλος παίρνουμε κάποια δεδομένα εξόδου. Αυτό σημαίνει ότι το αντικείμενο πρέπει να είναι εκτελέσιμο, γι'αυτό στα κατώτερα επίπεδα ελέγχου, στις δοκιμασίες μονάδας και συνένωσης, το αντικείμενο τρέχει πάνω σε ένα «κρεβάτι ελέγχου» *το οποίο* αποτελείται από stubs και test drivers για την προσομοίωση κομματιών του προγράμματος. Οι δυναμικές τεχνικές χωρίζονται σε δύο βασικές κατηγορίες, στο *black box έλεγχο* και τη *white box έλεγχο*. Υπάρχει επίσης και ο *grey box έλεγχος,* που συνδυάζει στοιχεία και των δύο μεθόδων.

Για να αποφασιστεί ποια μέθοδος ελέγχου θα ακολουθηθεί και ποιες περιπτώσεις ελέγχου θα επιλεχθούν, ο tester πρέπει να επιλέξει μια συστηματική προσέγγιση σύμφωνα με τα παρακάτω κριτήρια:

- Να οριστούν οι συνθήκες, οι προϋποθέσεις και οι στόχοι του ελέγχου.
- Να αποσαφηνιστούν οι περιπτώσεις ελέγχου και να προσδιοριστούν τα αναμενόμενα αποτελέσματα και συμπεριφορές.
- Να προσδιοριστούν με ποιο τρόπο θα εκτελεστούν οι δοκιμασίες.

Black box έλεγχος

Ο black box έλεγχος εξετάζει τη λειτουργικότητα του προϊόντος ή του συστήματος όπως αυτή απορρέει από τις προδιαγραφές απαιτήσεων, αντιμετωπίζοντας το αντικείμενο υπό έλεγχο σαν ένα μαύρο κουτί αδιαφορώντας για τις εσωτερικές του λειτουργίες και δομές. Χρησιμοποιείται συχνά στα ανώτερα επίπεδα ελέγχου, όπως στις δοκιμασίες συστήματος και αποδοχής, όμως μπορεί να εφαρμοστεί και στις δοκιμασίες μονάδας. Επίσης, η μέθοδος αυτή χρησιμοποιείται συχνά και στο σχεδιασμό σεναρίων ελέγχου όταν ο κώδικας και η υλοποίηση δεν έχει ξεκινήσει ακόμα.

Υπάρχουν πολλοί μέθοδοι σύμφωνα με τον black box έλεγχο, στην παρούσα εργασία ωστόσο θα εξεταστούν οι εξής από αυτούς:

a. ***Διαχωρισμός σε κλάσεις ισοδυναμίας (Equivalence class partitioning)***

Το πεδίο ορισμού των πιθανών εισερχόμενων δεδομένων χωρίζεται σε κλάσεις ισοδυναμίας, και μία τιμή από κάθε κλάση επιλέγεται για τον έλεγχο. Κλάση ισοδυναμίας ορίζεται μια ομάδα δεδομένων όπου ο έλεγχος τις επεξεργάζεται το ίδιο. Ας δούμε λεπτομερώς πως προκύπτουν τα δεδομένα ελέγχου. Πρώτα, πρέπει να οριστεί πλήρως το πεδίο ορισμού και να χωριστεί σε δύο κλάσεις, τις έγκυρες και μη έγκυρες τιμές. Στη συνέχεια, περισσότερος διαχωρισμός σε κλάσεις πρέπει να γίνει σύμφωνα με τις προδιαγραφές και να επιλεγεί μια αντιπροσωπευτική τιμή . Τέλος, ο tester πρέπει να ορίζει τις προϋποθέσεις και τα αναμενόμενα αποτελέσματα για κάθε περίπτωση ελέγχου. Ο διαχωρισμός σε κλάσεις μπορεί να συμβεί και στα εξερχόμενα δεδομένα.

Για να οριστεί μια περίπτωση ελέγχου, όλες οι αντιπροσωπευτικές τιμές των έγκυρων κλάσεων ισοδυναμίας πρέπει να συνδυαστούν σε μια (*πολλαπλασιαστική μέθοδος*), φτιάχνοντας έτσι μια περίπτωση σενάριο ελέγχου. Ενώ οι αντιπροσωπευτικές τιμές των μη έγκυρων κλάσεων ισοδυναμίας μπορούν να συνδυαστούν μόνο με τις τιμές των αντίστοιχων έγκυρων κλάσεων (*προσθετική μέθοδος)* δημιουργώντας έτσι μια αρνητική περίπτωση ελέγχου. Ο συνολικός αριθμός των σεναρίων ελέγχου είναι περιορισμένος, συνεπώς υπάρχουν κάποιοι κανόνες για τη μείωση του αριθμού, όπως προτεραιότητα των σεναρίων με βάση τις υψηλής σημασίας περιπτώσεις, βεβαιώνοντας έτσι ότι όλες οι αντιπροσωπευτικές τιμές των κλάσεων εμφανίζονται τουλάχιστον μία φορά.

Κριτήριο ολοκλήρωσης του ελέγχου για τη μέθοδο του διαχωρισμού σε κλάσεις ισοδυναμίας είναι το ορισμένο ποσοστό αυτοδύναμων κλάσεων:

EC-κάλυψη = (αριθμός των κλάσεων ισοδυναμίας που ελέγχθηκαν / συνολικός αριθμός των κλάσεων ισοδυναμίας) * 100 % .

b. ***Ανάλυση οριακών τιμών (Boundary value analysis)***

Η ανάλυση των οριακών τιμών στον έλεγχο λογισμικού, γίνεται συνήθως σε συνδυασμό με τη μέθοδο του διαχωρισμού σε κλάσεις ισοδυναμίας. Δίνει περισσότερες πληροφορίες για το λογισμικό και καταφέρνει να βρει πολλά ελαττώματα, συνήθως λόγω έλλειψης επακριβούς ορισμού των ορίων.

Σε κάθε όριο, πρέπει να ελέγχονται οι οριακές αλλά και οι δυο γειτονικές τιμές, μέσα και έξω από τις κλάσεις ισοδυναμίας.. Συχνά μια γειτονική τιμή αρκεί για να δώσει αποτελέσματα ελέγχου. Οι τιμές από τη μέση μιας κλάσης ισοδυναμίας δεν προσφέρουν επιπλέον πληροφορία για το έλεγχο και θεωρούνται άχρηστες. Η ανάλυση των οριακών τιμών δε μπορεί να χρησιμοποιηθεί για εισερχόμενα δεδομένα που λαμβάνουν διακριτές τιμές, μπορεί όμως να χρησιμοποιηθεί για τις εξερχόμενες κλάσεις ισοδυναμίας.

Όμοια με την προηγούμενη μέθοδο, οι έγκυρες οριακές τιμές συνδυάζονται σε μία περίπτωση ελέγχου της ανάλυσης, (πολλαπλασιαστική μέθοδος), ενώ οι μη έγκυρες οριακές τιμές πρέπει να δοκιμαστούν ξεχωριστά.

Το κριτήριο ολοκλήρωσης του ελέγχου για τη μέθοδο της ανάλυσης οριακών τιμών ορίζεται από το ποσοστό:

BV-κάλυψη = (αριθμός των οριακών τιμών που ελέγχθηκαν / συνολικός αριθμός οριακών τιμών) * 100%.

c. Έλεγχος μετάβασης καταστάσεων (State transition testing)

Σε πολλές περιπτώσεις, το ιστορικό της εκτέλεσης βημάτων είναι σημαντικό στη συμπεριφορά του εξεταζόμενου αντικειμένου. Το διάγραμμα μετάβασης καταστάσεων βασίζεται στο *διάγραμμα καταστάσεων*, το οποίο μοντελοποιεί τη δυναμική συμπεριφορά ενός αντικειμένου και δείχνει όλες τις πιθανές καταστάσεις τις οποίες μπορεί να βρεθεί αυτό. Ως *κατάσταση* (*state*), ορίζεται το σύνολο τιμών που περιγράφουν την κατάσταση ενός αντικειμένου μια δεδομένη στιγμή. Το *διάγραμμα μετάβασης*

καταστάσεων συμβαίνει ως αποτέλεσμα ενός *γεγονότος* (event) και περιγράφει την κατάσταση που θα έχει ένα σύστημα ή μια μονάδα υποδεικνύοντας τα γεγονότα ή τις συνθήκες που το προκάλεσαν.

Ο έλεγχος πρέπει να εκτελέσει όλες τις πιθανές καταστάσεις τουλάχιστον μια φορά. Για το σχεδιασμό των σεναρίων ελέγχου, δημιουργούμε ένα διάγραμμα μετάβασης καταστάσεων, από το οποίο κάθε μονοπάτι αναπαριστά μια περίπτωση, έτσι όλες οι καταστάσεις να εκτελούνται. Για να σχεδιαστεί μια περίπτωση ελέγχου με βάση αυτή τη μεθοδολογία, τα ακόλουθα δεδομένα είναι απαραίτητα:

- Η αρχική κατάσταση του αντικειμένου υπό έλεγχο.
- Τα δεδομένα εισόδου.
- Τα αναμενόμενα αποτελέσματα και δεδομένα εξόδου.
- Η αναμενόμενη τελική κατάσταση.

Για κάθε μετάβαση κατάστασης του σεναρίου ελέγχου, τα ακόλουθα πρέπει να ορίζονται:

- Η κατάσταση πριν την μετάβαση.
- Το αρχικό γεγονός που προκάλεσε τη μετάβαση.
- Η αναμενόμενη αντίδραση που θα προκαλέσει η μετάβαση.
- Η επόμενη κατάσταση που αναμένεται να πάρει το αντικείμενο.

Κριτήρια εξόδου και ολοκλήρωσης τους ελέγχου είναι:

- Το αντικείμενο έχει λάβει όλες τις καταστάσεις τουλάχιστον μία φορά.
- Κάθε μετάβαση έχει εκτελεστεί τουλάχιστον μία φορά.
- Κάθε μετάβαση που παραβιάζεις τις προδιαγραφές έχει εξεταστεί.

Στον έλεγχο μετάβασης καταστάσεων το αντικείμενο υπό έλεγχο μπορεί να είναι ένα ολόκληρο σύστημα με διαφορετικές καταστάσεις συστήματος ή μια κλάση ενός προγράμματος αντικειμενοστραφούς γλώσσας.

Η τεχνική αυτή είναι κατάλληλη για τις δοκιμασίες συστήματος και ειδικά για τον έλεγχο του *γραφικού περιβάλλοντος του χρήστη GUI* (Graphical User Interface). Το GUI αποτελείται από ένα σύνολο οθονών και ελέγχων, όπως μενού και κουτιά διαλόγου κ.ά.. Αν οι οθόνες και οι έλεγχοι θεωρηθούν σαν καταστάσεις και οι εισερχόμενες κινήσεις του χρήστη πάνω σε αυτές σαν μεταβάσεις (π.χ. “OK”, “Cancel”), τότε μπορεί να εξεταστεί με την τεχνική αυτή.

Ο έλεγχος μετάβασης καταστάσεων πρέπει να εφαρμόζεται όταν οι καταστάσεις είναι σημαντικές και όταν η λειτουργικότητα του συστήματος επηρεάζεται από αυτές. Η μέθοδος αυτή είναι ιδιαίτερα χρήσιμη για τους αντικειμενοστραφείς ελέγχους διότι λαμβάνει υπόψη της

τις διάφορες φάσεις του αντικειμένου.

Κριτήριο ολοκλήρωσης του ελέγχου για τη μέθοδο είναι:

ST-κάλυψη = (αριθμός μεταβάσεων καταστάσεων που ελέγχθηκαν / συνολικός αριθμός μεταβάσεων καταστάσεων) * 100%.

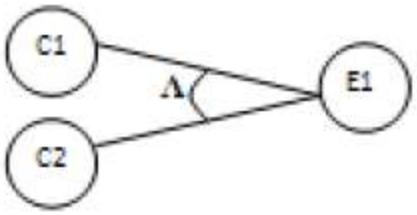
κάτι που είναι σπάνιο λόγω του πολύ μεγάλου αριθμού καταστάσεων.

d. Γράφος αίτιο-αιτιατό και πίνακας αποφάσεων (Cause-Effect graphing and Decision Table)

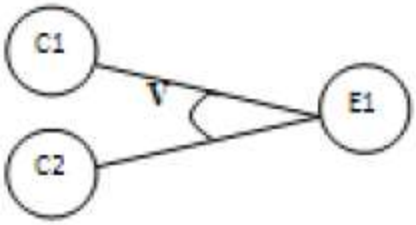
Ο *γράφος αίτιο-αιτιατό* βασίζεται στις σχέσεις ανάμεσα στα δεδομένα εισόδου και δεδομένων εξόδου σύμφωνα με τις προδιαγραφές απαιτήσεων. Τα *αίτια* (*causes*) και τα *αιτιατά* (*effects*) αναπαριστώνται στο γράφο σαν κόμβοι, ενώ οι σχέσεις μεταξύ τους σαν ακμές. Τα αίτια αποτελούνται από *εισερχόμενες προϋποθέσεις* (*conditions*) και λογικούς τελεστές AND, OR και NOT. Έτσι ένα αίτιο μπορεί να είναι *αληθές* (*true*) ή *ψευδές* (*false*).

Έστω ότι έχουμε δύο αίτια C1 και C2 και ένα αιτιατό E1. Η αναπαράσταση στο γράφο ανάλογα με τον τελεστή γίνεται ως εξής:

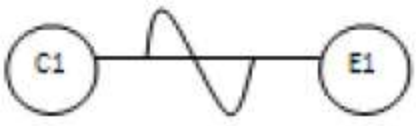
- (C1 AND C2) → E1



- (C1 OR C2) → E1



- (NOT C1) → E1



Για τη δημιουργία του γράφου φέρουμε πρώτα τους κόμβους, αριστερά τα αίτια και δεξιά τα αιτιατά, και έπειτα δημιουργούμε τις ακμές ξεκινώντας πάντα από τα αιτιατά. Εν συνεχεία, ο γράφος μετατρέπεται σε πίνακα αποφάσεων.

Ο πίνακας αποφάσεων έχει δύο μέρη, το πάνω μισό όπου καταγράφονται τα αίτια και το κάτω μισό όπου περιέχει τα αιτιατά. Όταν ικανοποιείται ένα αίτιο ή αιτιατό τότε αυτό σημειώνεται με Yes/No, True/False ή 1/0. Από τις στήλες του πίνακα αποφάσεων προκύπτουν οι

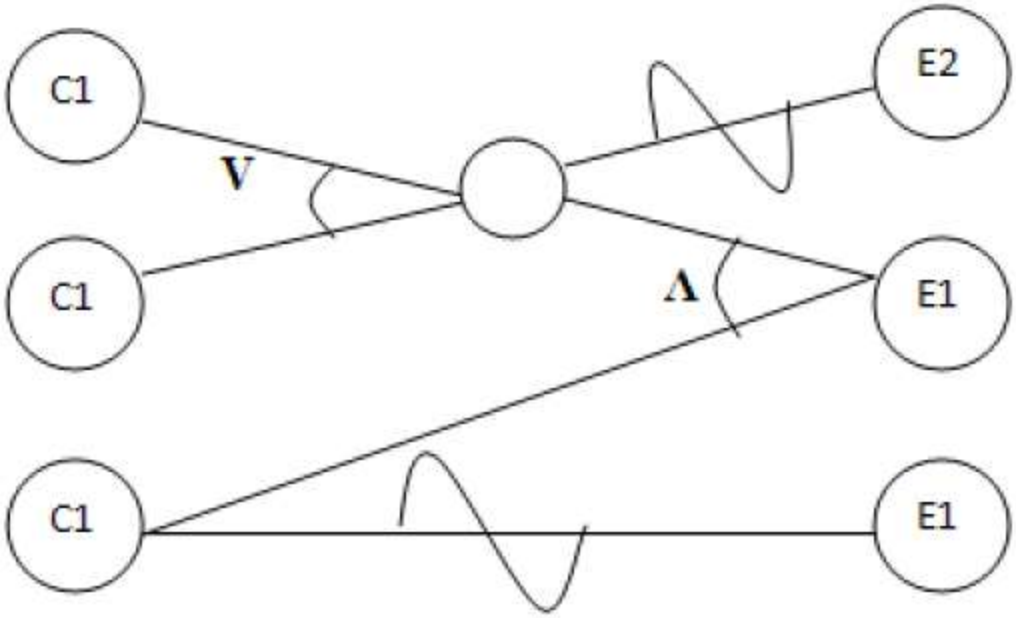
περιπτώσεις ελέγχου.

Τα βήματα για τη μετατροπή του γράφου σε πίνακα είναι: παραθέτουμε τα αίτια και τα αιτιατά στη πρώτη στήλη, επιλέγουμε το πρώτο αιτιατό, από το γράφο βρίσκουμε τους συνδυασμούς των αιτιών που έχουν αυτό το αιτιατό και τους συνδυασμούς των αιτιών που δεν έχουν αυτό το αιτιατό, προσθέτουμε μια στήλη στον πίνακα για κάθε ένα από αυτούς τους συνδυασμούς των αιτιών που προκαλούν αυτό το αιτιατό και τέλος ελέγχουμε αν ο πίνακας αποφάσεων έχει ίδιες στήλες και διαγράφοντας τις διπλοεγγραφές.

Για παράδειγμα, έχουμε το παρακάτω πρόβλημα όπου το πρόγραμμα διαβάζει τις τιμές των πρώτων δύο χαρακτήρων ενός αρχείου και ανάλογα εκτυπώνει:

- Ο πρώτος χαρακτήρας πρέπει να είναι «Α» ή «Β».
- Ο δεύτερος χαρακτήρας πρέπει να είναι ψηφίο.
- Αν ο πρώτος χαρακτήρας είναι «Α» ή «Β» και ο δεύτερος χαρακτήρας ψηφίο τότε το όνομα του αρχείου πρέπει να ανανεωθεί.
- Αν ο πρώτος χαρακτήρας είναι λάθος (ούτε «Α» ούτε «Β») τότε μήνυμα «Χ» πρέπει να εκτυπωθεί.
- Αν ο δεύτερος χαρακτήρας είναι λάθος (δεν είναι ψηφίο) τότε μήνυμα «Υ» πρέπει να εκτυπωθεί.

Τότε ο γράφος αιτίου-αιτιατού είναι:



Και ο πίνακας αποφάσεων είναι:

Actions	TC1	TC2	TC3	TC4	TC5	TC6
C1	1	0	0	0	1	0
C2	0	1	0	0	0	1
C3	1	1	0	1	0	0
E1	1	1	0	0	0	0
E2	0	0	1	1	0	0
E3	0	0	0	0	1	1

Γενικά κάθε συνδυασμός αιτιών θεωρείται μια περίπτωση ελέγχου, το οποίο σημαίνει ότι *n* συνθήκες (αίτια) έχουν *2n* συνδυασμούς. Για το λόγο αυτό συνήθως οι πίνακες βελτιστοποιούνται και πολλές στήλες φεύγουν. Οι πίνακες αποφάσεων είναι λογικοί έλεγχοι και πρέπει να έχουν ακριβή δεδομένα ελέγχου για να εκτελεστούν. Ελάχιστη προϋπόθεση για την ολοκλήρωση αυτού του τύπου ελέγχου είναι να εκτελέσουμε όλες τις στήλες από μια φορά. Τα ελαττώματα που προκύπτουν από τη μέθοδο αυτή είναι πολύ ενδιαφέρονται, γιατί προέρχονται από συνδυασμούς δεδομένων εισόδου. Ομως, πολλά λάθη προκύπτουν από την κατασκευή ή την βελτιστοποίηση των πινάκων και χωρίς κατάλληλα εργαλεία η τεχνική δεν είναι εύκολα εφαρμόσιμη.

e. Έλεγχος περιπτώσεων χρήσης (Use case testing)

Η *Ενοποιημένη Γλώσσα Μοντελοποίησης* (*Unified Modeling Language*) ορίζει τη γραφική σημειογραφία, τα UML διαγράμματα, τα οποία χρησιμοποιούνται για την παρουσίαση των απαιτήσεων σε ένα θεωρητικό επίπεδο και περιγράφουν τις αλληλεπιδράσεις του χρήστη με το σύστημα. Τα διαγράμματα των περιπτώσεων χρήσης εξυπηρετούν κυρίως για την παρουσίαση της εξωτερικής εικόνας του συστήματος. Επομένως, ένας έλεγχος βασισμένος σε μία περίπτωση χρήσης είναι χρήσιμος στις δοκιμασίες συστήματος και αποδοχής.

Ο έλεγχος δείχνει τις κανονικές ροές (normal workflow) του συστήματος και έτσι έχει μεγάλη σημασία για τον πελάτη και τον tester επίσης. Για κάθε περίπτωση χρήσης, συγκεκριμένες προϋποθέσεις (preconditions) πρέπει να τηρούνται για να εκτελεστεί ο έλεγχος, όπως και μετά την εκτέλεση ορίζονται συγκεκριμένες αναμενόμενες συνθήκες (post-conditions). Επίσης, υπάρχουν γεγονότα που οδηγούν σε εναλλακτικές ροές (alternative workflows), οι οποίες καταγράφονται στις περιπτώσεις χρήσης.

Ένα πιθανό κριτήριο εξόδου του ελέγχου είναι όταν όλες οι ροές των περιπτώσεων χρήσης , κανονικές και εναλλακτικές, εκτελεστούν τουλάχιστον μια φορά. Ωστόσο, τα δεδομένα εισόδου και τα αποτελέσματα του ελέγχου δεν απορρέουν άμεσα από τις περιπτώσεις χρήσης, Για το λόγο αυτό, ο έλεγχος περιπτώσεων χρήσης συνδυάζεται με άλλες μεθόδους, όπως η ανάλυση οριακών τιμών.

Σε γενικές γραμμές, οι διάφοροι μέθοδοι black box ελέγχου έχουν σαν βάση τις απαιτήσεις και τις προδιαγραφές του προϊόντος ή του συστήματος. Συνεπώς, δε μπορούν να βρουν ελαττώματα στις προδιαγραφές αλλά ούτε και επιπλέον λειτουργικότητες που δεν απαιτούνται. Συνεπώς έχει ως μόνο σκοπό την επιβεβαίωση των καταγεγραμμένων λειτουργιών. Επιπλέον, υπάρχουν και άλλα είδη ελέγχου αυτής της τεχνικής όπως *Syntax test*, *Random test* και *Smoke test*.

White box έλεγχος

Η βάση του white box ελέγχου είναι ο έλεγχος του κώδικα του αντικειμένου υπό εξέταση. Επομένως, ο κώδικας πρέπει να είναι διαθέσιμος ενώ σε πολλές περιπτώσεις θα πρέπει να είναι δυνατό και να τον επεξεργαστείς. Η γενική ιδέα είναι να εξεταστεί κάθε κομμάτι του κώδικα τουλάχιστον μια φορά. Υπάρχουν τέσσερις βασικές τεχνικές του white box ελέγχου:

a. Κάλυψη καταστάσεων (Statement coverage)

Η κάλυψη καταστάσεων εστιάζει σε κάθε κατάσταση του αντικειμένου υπό έλεγχο. Το πρώτο βήμα είναι η μετατροπή του κώδικα σε control flow γράφο. Όπως αναφέραμε παραπάνω, ο control flow γράφος αναπαριστά τις καταστάσεις σε κόμβους. Αν μια ακολουθία *μη υποθετικών προτάσεων* (*unconditional statements*) εμφανίζεται στον κώδικα, τότε αναπαριστώνται σαν ένας κόμβος, αφού η εκτέλεση του πρώτου εγγυάται την εκτέλεση και των υπολοίπων. Αντίθετα, οι *υποθετικές* προτάσεις (conditional statements) if και then, όπως και οι *βρόχοι* (*loops*) for, while και do…while έχουν περισσότερες ακμές να εξέρχονται για να καλύψουν όλες τις περιπτώσεις.

Μετά την εκτέλεση πρέπει να επιβεβαιωθεί ποίες από τις καταστάσεις έχουν εκτελεστεί και αν το προκαθορισμένο όριο έχει καλυφθεί τότε ο έλεγχος τερματίζεται. Η μέθοδος εστιάζει στην κάλυψη των ακμών του control flow γράφου. Το κριτήριο εξόδου της μεθόδου κάλυψη καταστάσεων, γνωστό και ως *CO-κάλυψη (C0-coverage)* είναι:

C0-κάλυψη = (αριθμών καταστάσεων που εκτελέστηκαν / συνολικός αριθμός καταστάσεων) * 100% .

Στα πλεονεκτήματα της μεθόδου είναι η επιβεβαίωση ότι ο κώδικας κάνει αυτά που πρέπει και δεν πρέπει και ο έλεγχος των μονοπατιών του προγράμματος. Επίσης, αν απαιτείται απόλυτη κάλυψη, δηλαδή 100%, αλλά ορισμένες καταστάσεις δεν μπορούν να καλυφθούν, τότε αυτό μπορεί να είναι ένδειξη *απροσπέλαστου κώδικα* (unreachable source code - dead statements). Βασικά μειονεκτήματα είναι ότι δεν μπορεί να ελέγξει τις ψευδείς συνθήκες, δεν αναφέρει αν ο βρόχος έφτασε στην τερματική του κατάσταση και δεν καταλαβαίνει τους λογικούς τελεστές.

b. Κάλυψη ακμών (Branch coverage)

Επίκεντρο της μεθόδου κάλυψης ακμών αποτελούν οι αποφάσεις (ακμές) του αντικειμένου υπό έλεγχο. Στον έλεγχο πρέπει να έχει εξασφαλιστεί ότι η εκτέλεση κάθε απόφασης μπορεί να έχει 2 αποτελέσματα: Αληθές και Ψευδές (TRUE and FALSE). Σε αντίθεση με την κάλυψη καταστάσεων, η κάλυψη ακμών λαμβάνει υπόψη της τα άδεια σημεία, επειδή εκτελούνται όλες οι υπό συνθήκη καταστάσεις. Συνεπώς είναι απαραίτητη η εισαγωγή επιπλέον σεναρίων ελέγχου, που να καλύπτουν όλες τις ακμές και να μπορούν έτσι να ανιχνεύονται οι ελλείψεις καταστάσεων σε άδειες ακμές. Αναλογικά με την κάλυψη καταστάσεων, το κριτήριο ολοκλήρωσης του ελέγχου για την μέθοδο κάλυψης των ακμών (*C1-coverage)* ορίζεται ως:

Κάλυψη ακμών = (αριθμός ακμών που εκτελέστηκαν / συνολικός αριθμός ακμών) * 100%.

c. Έλεγχος συνθηκών (Test of conditions)

Αν μια απόφαση βασίζεται σε διάφορες (μερικές) συνθήκες συνδεόμενες με λογικούς τελεστές τότε στον έλεγχο πρέπει να ληφθεί υπόψη η πολυπλοκότητα της κατάστασης. Οι παρακάτω έλεγχοι περιγράφουν διαφορετικές απαιτήσεις και διαφορετική ένταση των υπό σύνθεση προϋποθέσεων:

• Έλεγχος συνθηκών ακμών (Branch condition testing)

Το μονομερές (μερικό) σύνολο των προϋποθέσεων είναι μια λογική προϋπόθεση που δεν έχει κανέναν λογικό τελεστή (AND, OR and NOT). Σκοπός του ελέγχου είναι η αξιολόγηση κάθε μέρους του προϋποθέσεων από μία φορά, για κάθε μία απ' τις λογικές τιμές Αληθές ή Ψευδές (TRUE or FALSE). Ο έλεγχος συνθηκών ακμών αποτελεί συνεπώς ένα πιο αδύναμο κριτήριο σε σχέση με την κάλυψη καταστάσεων ή ακμών.

• Έλεχος συνδυασμού συνθηκών ακμών (Branch condition combination testing)

Αυτός ο έλεγχος, που επίσης καλείται έλεγχος κάλυψης πολλαπλών συνθηκών, απαιτεί όλοι οι συνδυασμοί Αληθές – Ψευδές να έχουν εκτελεστεί τουλάχιστον μία φορά. Συνεπώς, ο έλεγχος συνδυασμού συνθηκών ακμών ικανοποιεί τα κριτήρια τόσο της κάλυψης καταστάσεων όσο και της κάλυψης ακμών. Στα μειονεκτήματα της μεθόδου συμπεριλαμβάνονται το υψηλό της κόστος, καθώς χρειάζεται 2n περιπτώσεις ελέγχου για n μέρη προϋποθέσεων, καθώς και ότι δεν είναι δυνατόν να εκτελεστούν όλοι οι συνδυασμοί.

• Έλεγχος προσδιορισμού προϋποθέσεων (Condition determination testing)

Στην μέθοδο αυτή χρησιμοποιείται μόνο εκείνος ο συνδυασμός των λογικών τιμών, για τον οποίο η τροποποίηση τους για μια προϋπόθεση μπορεί να αλλάξει την λογική τιμή του συνόλου των προϋποθέσεων. Ο αριθμός των σεναρίων ελέγχου είναι αισθητά μικρότερος σε σχέση με αυτόν της μεθόδου ελέγχου συνδυασμού συνθηκών ακμών και συνιστά την καλύτερη τεχνική σχεδιασμού σεναρίων ελέγχου για την περίπτωση πολύπλοκών προϋποθέσεων. Επιπλέον, καταλήγει σε κάλυψη τόσο καταστάσεων όσο και ακμών. Το κριτήριο ολοκλήρωσης του ελέγχου, αναλογικά και με τις προαναφερθείσες μεθόδους, ορίζεται ως η αναλογία ανάμεσα στις εκτελεσθείσες τιμές προς όλες τις απαιτούμενες λογικές τιμές της προϋπόθεσης :

Συνθήκη τερματισμού = (αριθμός των τιμών που εκτελέστηκαν / συνολικός αριθμός των απαιτούμενων λογικών τιμών) * 100%.

d. Κάλυψη μονοπατιών (Path coverage)

Τέλος, αν το αντικείμενο περιέχει βρόχους ή επαναλήψεις, οι προηγούμενες μέθοδοι δεν επαρκούν και απαιτείται κάλυψη όλων των πιθανών μονοπατιών του αντικειμένου.

4.3 Αυτοματοποιημένος έλεγχος λογισμικού

4.3.1 Πότε και γιατί αυτοματοποίηση

Ο αυτοματοποιημένος έλεγχος λογισμικού έχει πλέον εξελιχθεί από πολυτέλεια σε αναγκαιότητα. Ο *μη αυτοματοποιημένος έλεγχος* ή *«χειρωνακτικός» (manual testing)* δεν μπορεί να συμβαδίσει με τη συνεχή εξέλιξη του μεγέθους και της πολυπλοκότητας των εφαρμογών και των συστημάτων. Τα πρώτα εργαλεία αυτοματοποιημένου ελέγχου είναι διαθέσιμα ήδη από τις αρχές τις δεκαετίας του ’90, με πρώτο το Microsoft Test4 version 1.0 που βγήκε σε beta έκδοση το 1992.

Το Microsoft Test ή MS-Test, αργότερα ονομάστηκε Visual Test, είναι ένα αυτοματοποιημένο εργαλείο λειτουργικού ελέγχου που εξετάζει εφαρμογές σε Windows και αναπτύχθηκε από τη Microsoft.

Ως *μη αυτοματοποιημένος έλεγχος* ορίζεται κάθε «χειρωνακτική» δοκιμή ελέγχου λογισμικού για την εύρεση ελαττωμάτων. Απαιτεί απο τον tester να παίξει το ρόλο του τελικού χρήστη και να χρησιμοποιήσει όλα τα χαρακτηριστικά της εφαρμογής, ώστε να διασφαλίσει την ορθή της συμπεριφορά. Απο την άλλη, *ο αυτοματοποιημένος έλεγχος* συνίσταται στην χρήση ειδικού λογισμικού, διαφορετικού από αυτό που ελέγχεται, που ρυθμίζει την εκτέλεση των έλεγχων και την σύγκριση πραγματικών με τα αναμενόμενα αποτελέσματα. Επιτρέπει το τρέξιμο μιας αλληλουχίας έλεγχων και την δημιουργία ενός αρχείου καταγραφής, που δείχνει ποιες δοκιμές πέτυχαν (*pass*) και απέτυχαν (*fail*) τα αναμενόμενα αποτελέσματα. Αυτό το αρχείο καταγραφής μπορεί είτε να αξιολογηθεί «χειρωνακτικά» από τον tester είτε να επεξεργαστεί αυτόματα για να προσδιοριστεί αν όλα τα αποτελέσματα των έλεγχων ανταποκρίνονται όπως αναμένεται στο προϊόν.

Οι δοκιμές που αυτοματοποιούνται συνήθως μπορούν να τρέξουν και «χειρωνακτικά», κάτι που όμως αποτελεί μια κοπιαστική και χρονοβόρα διαδικασία, καθώς επίσης και επιρρεπή σε ανθρώπινα λάθη. Σε γενικές γραμμές, ένας αυτοματοποιημένος έλεγχος αυτοματοποιεί κάποιες επαναλαμβανόμενες αλλά αναγκαίες διαδικασίες, π.χ. ελέγχους παλινδρόμησης ή αυτοματοποιεί διαδικασίες που θα ήταν αδύνατο να πραγματοποιηθούν «χειρωνακτικά» , π.χ. ελέγχους επίδοσης.

Ως προς το «τι» πρέπει να αυτοματοποιηθεί, «πότε», ακόμα και το «αν» χρειάζεται αυτοματοποιημένος έλεγχος συνιστούν καίριες αποφάσεις που πρέπει να λάβει η ομάδα διοίκησης (management team) ή ελέγχου (testing team). Η ορθή επιλογή κατάλληλων χαρακτηριστικών του προϊόντος που ελέγχονται αυτόματα καθορίζει την επιτυχία της αυτοματοποίησης. Τα εργαλεία αυτοματοποιημένου ελέγχου μπορεί να είναι ακριβά, καθώς επίσης και ο σχεδιασμός και η υλοποίηση των έλεγχων αυτών μπορούν επίσης να αποδειχθούν χρονοβόροι. Παρ’ όλα αυτά, η επένδυση σε μια αυτοματοποιημένη λύση ελέγχου μπορεί μακροπρόθεσμα να γίνει οικονομικά αποδοτική, ειδικά όταν οι δοκιμές τρέξουν πολλές φορές. Τα *σενάρια ελέγχου* (*test scripts*) χρειάζεται να δημιουργηθούν μόνο μία φορά και το κόστος που προστίθεται σε κάθε τρέξιμο της ελέγχου μπορεί να γίνει πολύ χαμηλό σε σύγκριση με το κόστος μιας μη αυτοματοποιημένης ελέγχου ελέγχου.

Επίσης, η επιτυχία της αυτοματοποίησης εξαρτάται και από άλλους παράγοντες. Ένας από αυτούς είναι «πόση αυτοματοποίηση» μας επιτρέπει να κάνουμε η εφαρμογή ή το σύστημα μας. Οι περιπτώσεις που μπορεί να αυτοματοποιηθεί ένα υποσύνολο έλεγχων, γενικά τουλάχιστον το 70% των έλεγχων, και ειδικά οι περιπτώσεις που είναι αναγκαίο το συχνό τρέξιμο αυτών των έλεγχων ελέγχου, συνιστούν ένα αδιάσειστο επιχείρημα υπέρ της επένδυσης σε ένα αυτοματοποιημένο σύστημα ελέγχου.

Επιπλέον, τα scripts αυτοματοποιημένου ελέγχου απαιτούν συστηματική συντήρηση για να διατηρήσουν την λειτουργικότητα και τη χρηστικότητά τους. Επομένως, ο χρόνος ζωής του αυτοματοποιημένου ελέγχου και η ικανότητά του να βρίσκει επιπλέον ελαττώματα σε σχέση με την πρώτη φορά που τρέχει συμβάλλουν καθοριστικά στην επιτυχία του. Ακόμα και μικρές αλλαγές κατά τη διάρκεια του χρόνου ζωής της εφαρμογής μπορούν να προκαλέσουν τη διακοπή των υπαρχόντων έλεγχων, οι οποίες μπορεί να δούλευαν κανονικά σε κάποια προηγούμενη φάση. Στην περίπτωση αυτή η δοκιμή πρέπει είτε να διορθωθεί ή να απορριφθεί. Για παράδειγμα, ένας αυτοματοποιημένος έλεγχος γραφικού περιβάλλοντος του χρήστη δεν έχει μεγάλο χρόνο ζωής, αφού επηρεάζεται από όλες τις αλλαγές στο επίπεδο του GUI. Η συντήρηση αυτοματοποιημένων ελέγχων συνιστά δύσκολο έργο, καθώς μπορεί να

αποδειχθεί δυσνόητη κι απαιτητική, κάτι που οδηγεί πολλές ομάδες στην απόρριψη των στρατηγικών αυτοματοποιημένου ελέγχου.

Τέλος, αξίζει να σημειωθεί ότι δεν βασίζονται όλες οι τεχνικές αυτοματοποιημένου ελέγχου στη συγγραφή κώδικα όπως για παράδειγμα οι έλεγχοι με γνώμονα *λέξεις-κλειδιά* (*keyword-driven tests*), σε περιπτώσεις όμως που τα σενάρια ελέγχου είναι γραμμένα σε μορφή πηγαίου κώδικα, τότε ο μηχανικός ελέγχου ή ο ειδικός στη διασφάλιση ποιότητας λογισμικού πρέπει να έχει προγραμματιστικές γνώσεις.

Συνοπτικά τα πλεονεκτήματα του αυτοματοποιημένου ελέγχου είναι:

- Η μείωση του χρόνου ελέγχου.
- Η αυξημένη κάλυψη ελέγχου.
- Η εφαρμογή ελέγχων που δεν μπορούν να πραγματοποιηθούν αλλιώς, π.χ. έλεγχοι επίδοσης, φόρτου και έντασης.
- Τα εργαλεία αυτοματισμού δεν κοιτάζουν απλά την οθόνη, αλλά διερευνούν και τις δομές των δεδομένων, ελέγχοντας επίσης και τις επιχειρησιακές διαδικασίες.
- Κατάλληλο για ελέγχους παλινδρόμησης (regression testing) και καπνού (smoke testing).
- Ο σχεδιασμός των σεναρίων ελέγχου γίνεται πριν την ανάληψη νέας έκδοσης του λογισμικού.
- Στην περίπτωση που δεν είναι εύκολη η αναπαραγωγή ενός σφάλματος που εντοπίστηκε, ο tester μπορεί να ανατρέξει στο αρχείο καταγραφής.

Από την άλλη πλευρά, τα μειονεκτήματα της αυτοματοποίησης του ελέγχου είναι:

- Χρονοβόρο και απαιτεί περισσότερες δεξιότητες από τους testers.
- Απλοϊκές δοκιμές μπορούν να επηρεάσουν αρνητικά την έκβαση του ελέγχου.
- Φθίνει η λειτουργικότητά του ως προς το χρόνο, λόγω συντήρησης.
- Οι tester μπορούν να παρατηρήσουν αλλόκοτα και παράξενα αποτελέσματα (τα script ελέγχου είναι προσαρμοσμένα στον σκοπό).
- Αν ο «χειρωνακτικός» έλεγχος βρει ένα ελάττωμα, ο tester θα ξανατρέξει τη δοκιμή στο ίδιο σημείο για να επιβεβαιώσει την διόρθωσή του, πιθανώς και περισσότερες από μία φορές (αξιοποιώντας πρακτικά την αρχή ότι τα σφάλματα συνήθως συγκεντρώνονται σε κοντινά σημεία).

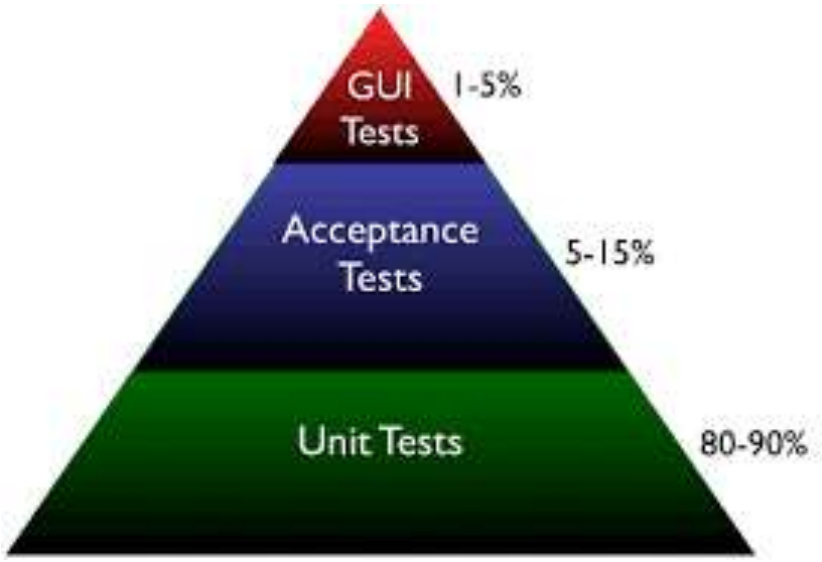
4.3.2 Γενικές κατηγορίες αυτοματοποιημένου ελέγχου λογισμικού

Ο στόχος της αυτοματοποίησης και αντίστοιχα οι τεχνικές και τα εργαλεία του αυτοματοποιημένου ελέγχου εξαρτώνται απόλυτα από το είδος και το επίπεδο ελέγχου.

Συνήθως τα παρακάτω είδη ελέγχου λογισμικού ενδείκνυται να αυτοματοποιηθούν εξολοκλήρου ή σε ένα βαθμό:

- *Λειτουργικός έλεγχος* (*Functional automated testing*)
    - ο *Έλεγχος παλινδρόμησης* (*Regression testing*)
    - ο *Αρνητικές δοκιμές* (*Negative testing*)
- *Μη λειτουργικός έλεγχος* (*Non-functional testing*)
    - ο *Έλεγχος επίδοσης* (*Performance testing*)
    - ο *Έλεγχος έντασης* (*Stress testing*)
    - ο *Έλεγχος φόρτου* (*Load testing*)

Ο λειτουργικός έλεγχος, ο οποίος ελέγχει τις σωστές λειτουργίες του λογισμικού είναι ο πιο καλός υποψήφιος για αυτοματοποίηση και μπορεί να κατηγοριοποιηθεί σε τρία επίπεδα, ανάλογα το επίπεδο της εφαρμογής που ελέγχει. Η *πυραμίδα του αυτοματοποιημένου ελέγχου* (*test automation pyramid*), σύμφωνα με την agile μεθοδολογία ανάπτυξης λογισμικού, χωρίζει τον αυτοματοποιημένο έλεγχο σε τρία επίπεδα και αναλόγως προκύπτουν οι τρεις γενικές κατηγορίες αυτοματοποιημένου ελέγχου:



Σχήμα 3.1: Πυραμίδα του αυτοματοποιημένου ελέγχου

- ***Αυτοματοποιημένες δοκιμασίες μονάδας*** (*Unit level testing or Code-driven testing*)

Οι κλάσεις, τα modules και οι *βιβλιοθήκες* (libraries) δοκιμάζονται με ένα εύρος δεδομένων εισόδου και ελέγχονται αν όλα τα αποτελέσματα είναι σωστά.

• ***Αυτοματοποιημένες δοκιμασίες γραφικού περιβάλλοντος*** (*Graphical User Interface testing or GUI testing*).

Οι δοκιμασίες παράγουν γεγονότα που αλληλεπιδρούν με το γραφικό περιβάλλον της εφαρμογής, όπως εισαγωγή κειμένου σε ένα πεδίο ή πάτημα ενός κουμπιού, και παρατηρεί τις αλλαγές που γίνονται με στόχο την επικύρωση τους.

• ***Αυτοματοποιημένες δοκιμασίες αποδοχής*** (*Acceptance level testing or Service/Middle tier based testing or API driven testing*).

Ξεπερνώντας το επίπεδο γραφικού περιβάλλοντος η εφαρμογή ελέγχεται στη μεσαία βαθμίδα, εκεί όπου επικυρώνεται η συμπεριφορά του λογισμικού υπό έλεγχο. Επίσης χρησιμοποιεί μια προγραμματιστική διεπαφή για να «οδηγήσει» το γραφικό περιβάλλον του χρήστη στις επιθυμητές ενέργειες.

4.3.2.1 Αυτοματοποιημένες δοκιμασίες μονάδας

Στη βάση της πυραμίδας βρίσκονται οι αυτοματοποιημένες δοκιμασίες μονάδας. Όπως και στο χειρωνακτικό έλεγχο λογισμικού, οι δοκιμασίες με βάση τον κώδικα ή τη μονάδα είναι μεγάλης σημασίας για την ανάπτυξη του λογισμικού και για αυτό πρέπει να είναι η βάση μιας αυτοματοποιημένης στρατηγικής ελέγχου.

Οι δοκιμασίες μονάδας γράφονται για να ορίσουν τη λειτουργικότητα πριν ο κώδικας γραφτεί. Στο μεταξύ, οι δοκιμασίες εξελίσσονται και μεγαλώνουν όσο ο κώδικας προχωράει, και όταν όλοι οι έλεγχοι για όλες τις απαιτούμενες λειτουργίες περάσουν, θεωρούμε τον κώδικα ολοκληρωμένο. Οι αυτοματοποιημένες δοκιμασίες σε αυτό το επίπεδο παράγουν λογισμικό πιο αξιόπιστο και λιγότερο δαπανηρό από το *μη αυτοματοποιημένο διερευνητικό έλεγχο* (*manual exploratory testing*).

Ο κώδικας θεωρείται πιο αξιόπιστος γιατί η κάλυψη του κώδικα είναι πολύ μεγαλύτερη ενώ η αυτοματοποίηση επιτρέπει πιο εύκολα την επανάληψη του ελέγχου πολλές φορές παράλληλα με τη διαδικασία της ανάπτυξης του λογισμικού. Επίσης, θεωρείται λιγότερο δαπανηρό γιατί ο προγραμματιστής ανακαλύπτει ελαττώματα κατευθείαν μετά από μια αλλαγή στον κώδικα, δηλαδή όσο πιο νωρίς γίνεται και όταν η διόρθωση τους κοστίζει το λιγότερο δυνατό. Τέλος, ο αυτοματοποιημένος έλεγχος μονάδας βοηθάει στην εύκολη και ασφαλή τροποποίηση του κώδικα και στην απλοποίησή αποφεύγοντας επανάληψης *μερών κώδικα* (*code duplication*) μειώνοντας έτσι τα περιθώρια για ελαττώματα στο λογισμικό.

Η αυτοματοποιημένη προσέγγιση των δοκιμασιών μονάδας είναι επίσης ιδιαίτερα δημοφιλής και πετυχημένη διότι παρέχει στους προγραμματιστές επιπλέον πληροφορίες για τα ελαττώματα που ανιχνεύουν. Συγκεκριμένα, αν ένας tester βρει ένα ελάττωμα π.χ. στο πως αποθηκεύει το σύστημα τους νέους χρήστες στη βάση δεδομένων, αυτό το ελάττωμα μπορεί να βρίσκεται σε 1.000 ή περισσότερες γραμμές κώδικα.

Από την άλλη πλευρά ένας αυτοματοποιημένος έλεγχος μονάδας μπορεί να επιστρέψει συγκεκριμένα που βρίσκεται π.χ. στη γραμμή 51 ή 62. Επίσης, ο αυτοματοποιημένος έλεγχος γράφεται συνήθως στην ίδια γλώσσα με τον κώδικα, γεγονός που καθιστά τους προγραμματιστές πιο κατάλληλους για την δημιουργία και εκτέλεση του. Τα τελευταία χρόνια αυτό το είδος αυτοματοποιημένου ελέγχου εκτελείται με τις πλατφόρμες xUnit, όπως η JUnit και NUnit, οι οποίες εξετάζουν αν μέρη του κώδικα λειτουργούν έτσι όπως αναμένεται κάτω από ειδικές συνθήκες.

4.3.2.2 Αυτοματοποιημένες δοκιμασίες γραφικού περιβάλλοντος χρήστη

Οι αυτοματοποιημένες δοκιμασίες ελέγχου γραφικού περιβάλλοντος εμφανίζονται στην κορυφή της πυραμίδας του αυτοματοποιημένου ελέγχου γιατί προτιμάται να γίνονται σε μικρό βαθμό λόγω των περιορισμών τους. Για παράδειγμα, θέλουμε να ελέγξουμε μια απλή εφαρμογή αριθμομηχανής που

επιτρέπει στο χρήστη να βάλει δύο ακεραίους και να πατήσει είτε το κουμπί του πολλαπλασιασμού είτε το κουμπί της διαίρεσης.

Για να ελεγχθεί το γραφικό περιβάλλον αυτής της εφαρμογής χρειάζεται να κατασκευαστεί ένα σενάριο με μια σειρά ελέγχων που θα «οδηγούν» το *γραφικό περιβάλλον* (*driver script*) στα κατάλληλα *αντικείμενα* (*UI objects*). Με τον τρόπο αυτό εισάγονται οι θ τιμές στα πεδία, επιλέγοντας ένα τα δύο κουμπιά και τέλος συγκρίνοντας τις αναμενόμενες με τις πραγματικές τιμές.

Αυτός ο τύπος ελέγχου αποφέρει συνήθως αποτελέσματα με εύκολο τρόπο, απαιτώντας μεγάλο αριθμό σεναρίων, γεγονός που τον καθιστά ακριβό και χρονοβόρο. Επίσης, είναι πολύ ευπαθής αφού μια μικρή αλλαγή στο γραφικό περιβάλλον απαιτεί την ανανέωση των scripts του αυτοματοποιημένου ελέγχου.

4.3.2.3 Αυτοματοποιημένες δοκιμασίες αποδοχής

Οι αυτοματοποιημένες δοκιμασίες αποδοχής εξετάζουν τις λειτουργίες και τις συμπεριφορές της εφαρμογής ξεχωριστά από το γραφικό περιβάλλον. Ο έλεγχος της μεσαίας βαθμίδας είναι κρίσιμης σημασίας, όχι μόνο γιατί ελέγχει τη σωστή λειτουργία του λογισμικού αλλά γιατί υπερβαίνει το επίπεδο του γραφικού περιβάλλοντος του χρήστη, το οποίο όπως αναφέραμε είναι δαπανηρό και χρονοβόρο.

Τα scripts των αυτοματοποιημένων δοκιμασιών αποδοχής γράφονται σε μια προγραμματιστική γλώσσα ή *γλώσσα σεναρίου* (*scripting language*), εκτελούνται από μια πλατφόρμα ελέγχου ή πάλι μέσω μιας γλώσσας σεναρίου και τέλος συγκρίνουν τα αποτελέσματα με την αναμενόμενη συμπεριφορά της εφαρμογής ή του συστήματος.

Ο έλεγχος ονομάζεται επίσης και δοκιμασίες μεσαίας βαθμίδας ή *βαθμίδας υπηρεσιών* (services). Ως υπηρεσία σε αυτό το επίπεδο θεωρείται η συμπεριφορά της εφαρμογής απέναντι σε κάποια δεδομένα εισόδου ή σύνολα δεδομένων εισόδου και ο έλεγχος εξετάζει τις υπηρεσίες της εφαρμογής ξεχωριστά από το γραφικό περιβάλλον. Στο παραπάνω παράδειγμα της αριθμομηχανής, έχουμε δυο βασικές υπηρεσίες: τον πολλαπλασιασμό και τη διαίρεση. Έτσι, αντί να ελέγξουμε πολλές περιπτώσεις ελέγχου, αντί δηλαδή να εισάγουμε πολλές διαφορετικές τιμές ακεραίων μέσω του γραφικού περιβάλλοντος, ελέγχουμε άμεσα τον πολλαπλασιασμό ή τη διαίρεση.

Το λάθος που γίνεται από τις περισσότερες *ομάδες διοίκησης* (*management teams*) ή *ομάδες ελέγχου* (*testing teams*) είναι ότι για πολλά χρόνια αγνοούσαν αυτό το επίπεδο ελέγχου κατά τον αυτοματοποιημένο έλεγχο. Επειδή οι αυτοματοποιημένες δοκιμασίες μονάδας, δεν καλύπτουν πλήρως τις ανάγκες του ελέγχου, χωρίς η μεσαία βαθμίδα ελέγχου να αναπληρώνει το κενό ανάμεσα στις δοκιμασίες μονάδος και δοκιμασίες γραφικού περιβάλλοντος, όλοι οι έλεγχοι καταλήγουν να γίνονται στο επίπεδο του γραφικού περιβάλλοντος. Αποτέλεσμα αυτού είναι οι δοκιμασίες να καθίστανται ακριβές για να γραφτούν, να τρέξουν και να εκτελεστούν.

4.3.3 Τεχνικές αυτοματοποιημένου έλεγχου λογισμικού

Στις περιπτώσεις των αυτοματοποιημένων δοκιμασιών αποδοχής και γραφικού περιβάλλοντος χρήστη, που εξετάζουν συνολικά το αντικείμενο υπό έλεγχο, τα διαθέσιμα εργαλεία ή πλαίσια ελέγχου (test frameworks) αυτοματοποίησης βασίζονται σε μια *τεχνική αυτοματοποιημένου ελέγχου (test automation technique)*.

Πλαίσιο αυτοματοποιημένου ελέγχου (*test automation framework*) είναι ένα συνενωμένο σύστημα που ορίζει τους κανόνες της αυτοματοποίησης ενός συγκεκριμένου προϊόντος ή συστήματος. Το σύστημα αποτελείται από τις *βιβλιοθήκες συναρτήσεων* (*test libraries*), τις πηγές δεδομένων ελέγχου, τις δομές για την καταγραφή των ελαττωμάτων και μία πλατφόρμα για τη δημιουργία δομών ελέγχου. Τέλος το πλαίσιο

παρέχει τη βάση του αυτοματοποιημένου ελέγχου και συμβάλει σημαντικά στην απλοποίηση της προσπάθειας αυτοματοποίησης.

Το μεγάλο πλεονέκτημα των πλαισίων αυτοματοποιημένου ελέγχου είναι το χαμηλό κόστος συντήρησης. Αν υπάρχει έστω και μία μικρή αλλαγή σε μια περίπτωση ελέγχου τότε μόνο το αρχείο με τη συγκεκριμένη περίπτωση ελέγχου χρειάζεται να ανανεωθεί, τα *σενάρια-οδηγοί* (*driver scripts*) και τα *εναρκτήρια σενάρια* (*startup scripts*) παραμένουν ως έχουν και δε χρήζουν αλλαγής. Μερικές φορές δεν χρειάζονται ούτε τα σενάρια αλλαγές, όπως για παράγειγμα στην περίπτωση που αλλάξουν τα δεδομένα ελέγχου τα οποία είναι αποθηκευμένα σε ξεχωριστά αρχεία.

Η επιλογή του σωστού πλαισίου ελέγχου, δηλαδή της σωστής τεχνικής αυτοματοποίησης που ακολουθεί το πλαίσιο, είναι η πιο σημαντική, τόσο για την επιτυχία του ελέγχου όσο και για το κόστος της αυτοματοποίησης. Το πλαίσιο αυτοματοποιημένου ελέγχου είναι υπεύθυνο για:

1. Τον ορισμό της μορφολογίας (format) στην οποία θα εκφραστούν οι περιπτώσεις

ελέγχου.

2. Τη δημιουργία ενός μηχανισμού που θα «οδηγεί» την εφαρμογή υπό έλεγχο.

3. Την εκτέλεση των δοκιμασιών.

4. Την καταγραφή των αποτελεσμάτων ελέγχου.

Όπως θα δούμε και παρακάτω, πολλά εργαλεία ή ολοκληρωμένα πλαίσια ελέγχου ακολουθούν

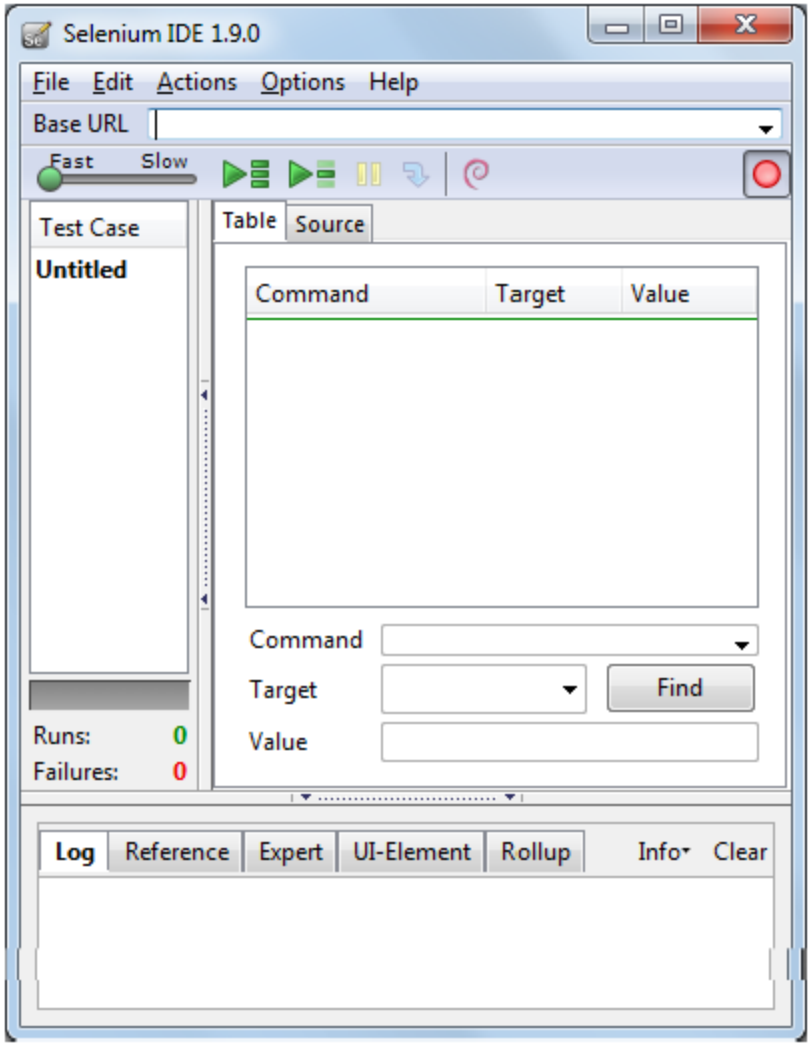
πολλές τεχνικές αυτοματοποίησης.

4.3.3.1 Τεχνική «Καταγραφή/Επανάληψη»

Πολλά εργαλεία αυτοματοποιημένου ελέγχου παρέχουν την *τεχνική της καταγραφής/επανάληψης* (*Record/Playback technique*) επιτρέποντας την καταγραφή των αλληλεπιδράσεων και ενεργειών των χρηστών πάνω στα αντικείμενα γραφικού περιβάλλοντος. Το σενάριο που παράγεται από τη διαδικασία της καταγραφής αποτελείται από μια λίστα ενεργειών στη γλώσσα του εργαλείου, οι οποίες μπορούν να επαναληφθούν αυτοματοποιημένα πάνω στο αντικείμενο υπό έλεγχο. Αφού το σενάριο καταγραφεί, ο κώδικάς μπορεί να επεξεργαστεί «χειρωνακτικά» και να προστεθούν μεταβλητές, συνθήκες, βρόχοι κτλ.

Η παραπάνω μέθοδος ανήκει στις αυτοματοποιημένες δοκιμασίες ελέγχου γραφικού περιβάλλοντος, είναι εύκολη σε χρήση και δεν απαιτεί προγραμματιστικές ικανότητες. Ωστόσο, η εξάρτηση από τα αντικείμενα γραφικού περιβάλλοντος δημιουργεί πολλά προβλήματα αξιοπιστίας και συντηρησιμότητας, ειδικά σε περίπτωση αλλαγών της διεπαφής του χρήστη. Τα σενάρια συνίστανται από μια μεγάλη λίστα ενεργειών με δεδομένα εισόδου hard-coded πάνω στα αντικείμενα του γραφικού περιβάλλοντος.

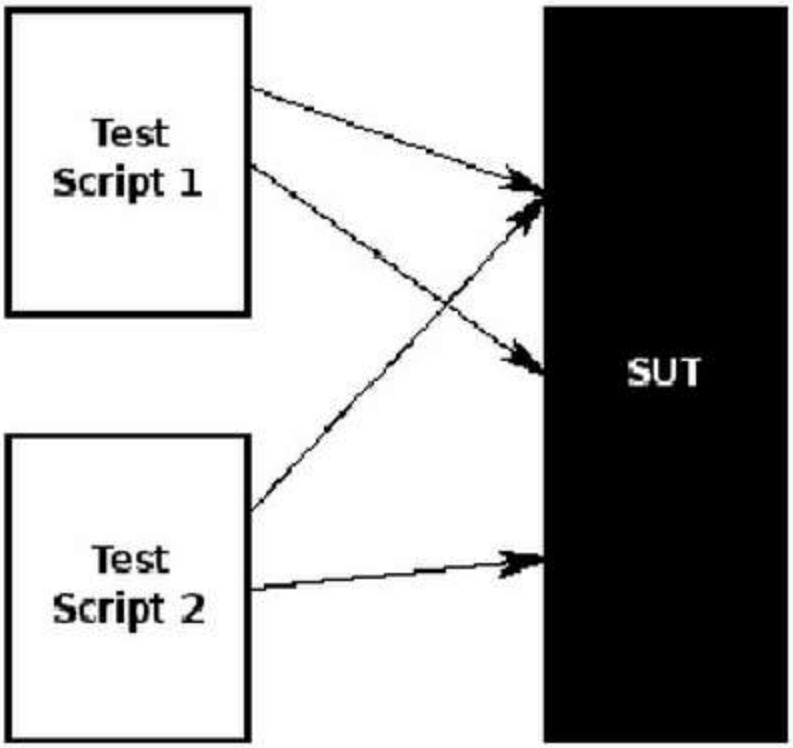
Επίσης, η τεχνική κατά το στάδιο της καταγραφής προσθέτει πολλές φορές λανθασμένα ενέργειες που δε σχετίζονται με το σενάριο με αποτέλεσμα ακόμα μεγαλύτερα σενάρια και δυσκολία στη συντήρησή τους. Τέλος, το σύστημα πρέπει να είναι έτοιμο (σεταρισμένο) έτσι ώστε να ξεκινήσει ο αυτοματοποιημένος έλεγχος, ενώ σε περίπτωση μη αναμενόμενων λαθών ή εξαιρέσεων του συστήματος δε γνωρίζει πώς να τα διαχειριστεί. Γενικά θεωρείται καλή τεχνική για αυτοματοποιημένο έλεγχο γραφικού περιβάλλοντος, αλλά δεν ενδείκνυται για μεγάλες εφαρμογές ή συστήματα.



Σχήμα 3.3: Περιβάλλον εργαλείου αυτοματοποιημένου ελέγχου Selenium.

4.3.3.2 Γραμμική τεχνική

Η *γραμμική τεχνική* (*linear scripting*) απαιτεί τη δημιουργία μικρών, ανεξάρτητων και μη δομημένων σεναρίων ελέγχου, δηλαδή σεναρίων που δεν περιέχουν συνθήκες και βρόχους, και αλληλεπιδρούν άμεσα με το υπό έλεγχο σύστημα. Τα scripts μπορούν να γραφτούν σε οποιαδήποτε γλώσσα προγραμματισμού ή μπορούν να παραχθούν από αυτοματοποιημένα εργαλεία της μεθόδου καταγραφής/επανάληψης.



Σχήμα 3.4: Σχηματική απεικόνιση της γραμμικής τεχνικής.

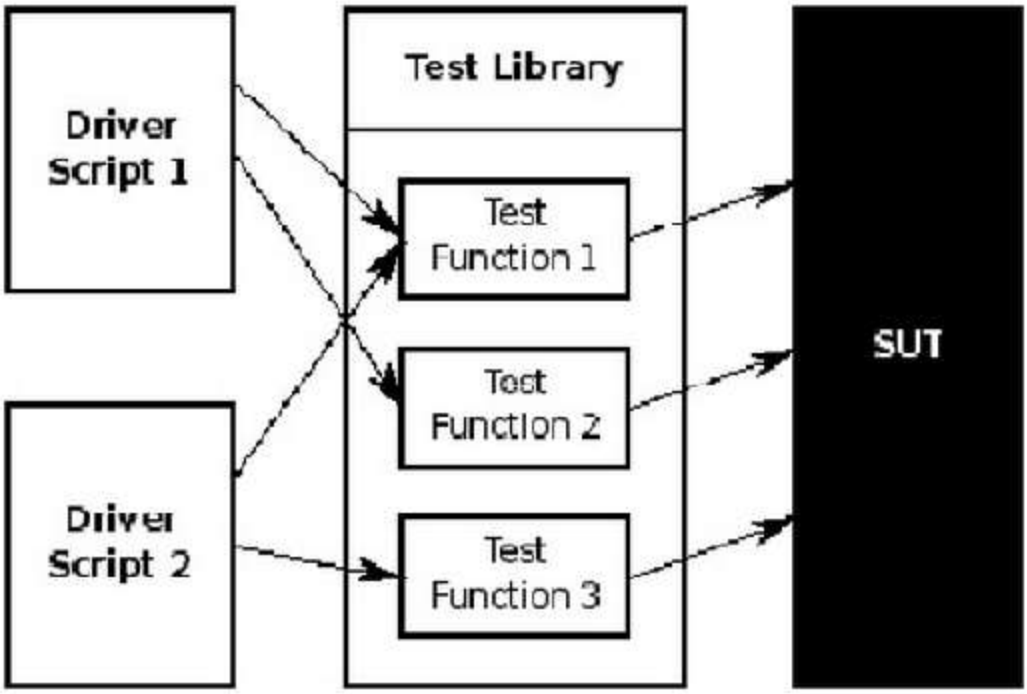
Στα πλεονεκτήματα της γραμμικής τεχνικής είναι η ευκολία δημιουργίας και προσαρμοστικότητας των σεναρίων, ενώ στα μειονεκτήματα ο tester χρειάζεται να έχει προγραμματιστικές ικανότητες και τα σενάρια ελέγχου είναι πολύ ευπαθή και δύσκολο να συντηρηθούν. Για απλούς και μικρούς ελέγχους η τεχνική θεωρείται επαρκής.

4.3.3.3 Δομημένη τεχνική

Με την βοήθεια της *δομημένης τεχνικής* (*modular scripting*) δημιουργούνται *«σενάρια-οδηγοί»* (*driver scripts*) που περιέχουν δομές, όπως *if…else, switch, for* και *while*, οι οποίες αλληλεπιδρούν με το σύστημα υπό έλεγχο μέσω συναρτήσεων που ανήκουν σε *βιβλιοθήκες*.

Στην συγκεκριμένη τεχνική η επαναχρησιμοποίηση των σεναρίων και η δημιουργία καινούργιων είναι πιο εύκολη, ενώ η συντηρησιμότητα του κώδικα σε περίπτωση αλλαγών απαιτεί λιγότερες διορθώσεις σε μικρότερες περιοχές.

Τα «σενάρια-οδηγοί» είναι γενικά απλά, ακόμα και αρχάριοι προγραμματιστές μπορούν να τα δημιουργήσουν και να επεξεργαστούν, σε αντίθεση με τις βιβλιοθήκες που απαιτούν χρόνο και καλές προγραμματιστικές ικανότητες.



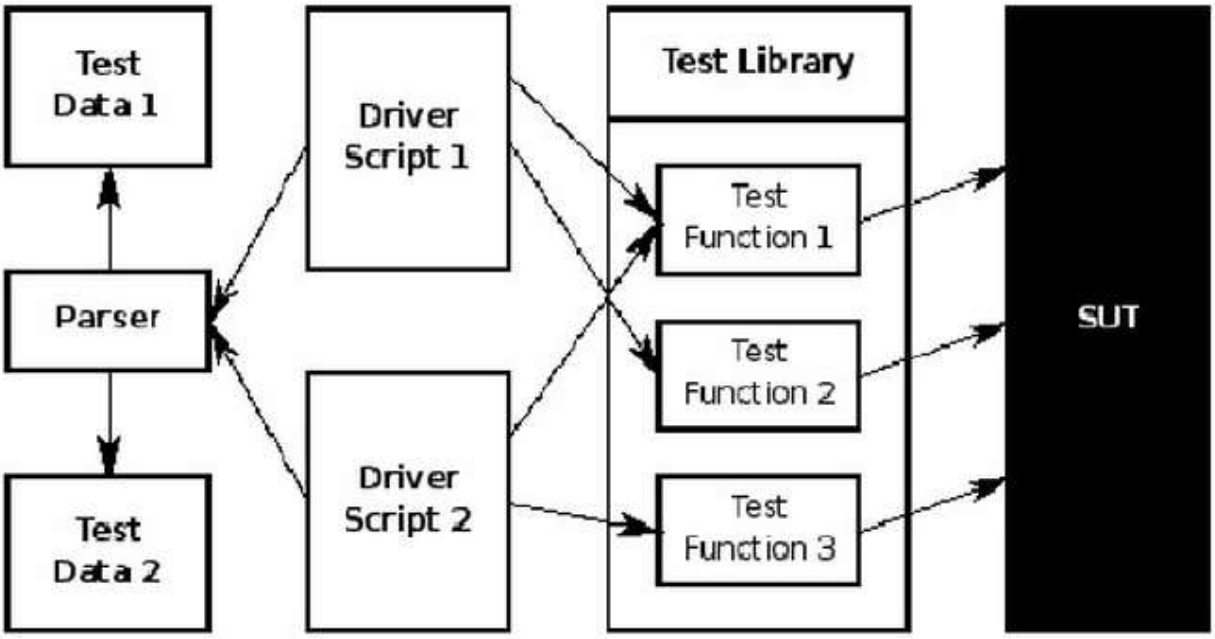
Σχήμα 3.5: Σχηματική απεικόνιση της δομημένης τεχνικής.

Επιπλέον η εισαγωγή δεδομένων ελέγχου γίνεται μόνο μέσω των σεναρίων και κάθε νέος έλεγχος απαιτεί νέα «σενάρια-οδηγούς». Γενικά, η δομημένη τεχνική είναι κατάλληλη για αυτοματοποίηση μικρών και μεσαίων εφαρμογών.

4.3.3.4 Τεχνική βάσει δεδομένων

Η *τεχνική αυτοματοποίησης βάσει δεδομένων* (*data-driven testing*) διαχωρίζει τα δεδομένα ελέγχου από το σενάριο ελέγχου. Τα δεδομένα ελέγχου βρίσκονται σε ξεχωριστά *αρχεία* (*data files*) και από εκεί φορτώνονται στις μεταβλητές του σεναρίου ελέγχου είτε ως δεδομένα εισόδου είτε ως δεδομένα εξόδου. Με την τεχνική αυτή, οι έλεγχοι εξετάζουν πολλούς συνδυασμούς δεδομένων και τιμών, επιταχύνοντας έτσι μια καλύτερη κάλυψη των περιπτώσεων ελέγχων. Για την επιλογή των δεδομένων ελέγχου γίνεται συχνά χρήση των μεθόδων διαχωρισμού σε κλάσεις ισοδυναμίας και ανάλυσης συνοριακών τιμών.

Οτιδήποτε μπορεί να αλλάξει (όπως το περιβάλλον, τα κριτήρια ελέγχου, τα δεδομένα εισόδου και εξόδου, κτλ.) διαχωρίζεται από την λογική ελέγχου, δηλαδή τα σενάρια ελέγχου και μεταφέρεται «εξωτερικά». Κάθε φορά που ο tester θέλει να προσθέσει μια περίπτωση ελέγχου, προσθέτει ένα νέο δεδομένο στο αρχείο χωρίς να χρειαστεί να γράψει επιπλέον σενάριο.Τα πλαίσια ελέγχου αυτής της τεχνικής επιτρέπουν τη δημιουργία σεναρίων ελέγχου που «οδηγούν» την εφαρμογή (driver scripts) και τρέχουν μαζί με τα σχετικά δεδομένα ελέγχου. Τα σενάρια αυτά διαχειρίζονται το περιβάλλον ελέγχου, φορτώνουν τα δεδομένα ελέγχου και καταγράφουν τα αποτελέσματα. Επίσης, κάνουν χρήση των βιβλιοθηκών του πλαισίου και προσφέρουν μια επαναχρησιμοποιήσιμη λογική που μειώνει την συντήρηση και βελτιώνει την κάλυψη του ελέγχου.



Σχήμα 3.6: Σχηματική απεικόνιση της τεχνικής βάσει δεδομένων.

Τα «σενάρια-οδηγοί» απαιτούν χρήση κώδικα, αντίθετα με τις περιπτώσεις ελέγχου που καλύπτονται από τα δεδομένα ελέγχου και είναι πολύ απλές συνήθως σε μορφή πίνακα π.χ. ‘1 + 2 = 3’ (πρόσθεση) ή ‘1 * 2 = 2’ (πολλαπλασιασμός). Κάθε νέο είδος περιπτώσεων ελέγχου π.χ. ‘1 *2 + 3 = 6’ (πρόσθεση και πολλαπλασιασμός μαζί, προτεραιότητα πράξεων) χρειάζεται νέο «σενάριο-οδηγό».

Τα δεδομένα εισόδου ή εξόδου (κριτήρια ελέγχου) μπορούν να αποθηκευτούν σε μία ή πολλές κεντρικές πηγές δεδομένων ή βάσεις δεδομένων, η μορφολογία και η οργάνωση των οποίων εξαρτάται από την εφαρμογή που ελέγχουν. Οι βάσεις δεδομένων που χρησιμοποιούνται από τη μέθοδο για την αποθήκευση των δεδομένων ελέγχου είναι:

- *«Πισίνες» δεδομένων* (*Data pools*)
- *ODBC πηγές* (*sources*)
- CSVor CVS αρχεία
- Excel αρχεία
- DAO αντικείμενα
- ADO αντικείμενα

	A	B	C	D	E
1	Test Case	Number 1	Operator	Number 2	Expected
2	Add 01	1	+	2	3
3	Add 02	1	+	-2	-1
4	Sub 01	1	-	2	-1
5	Sub 02	1	-	-2	3
6	Mul 01	1	*	2	2
7	Mul 02	1	*	-2	-2
8	Div 01	2	/	1	2
9	Div 02	2	/	-2	-1

Σχήμα 3.7: Παράδειγμα αρχείου δεδομένων για αυτοματοποιημένο έλεγχο.

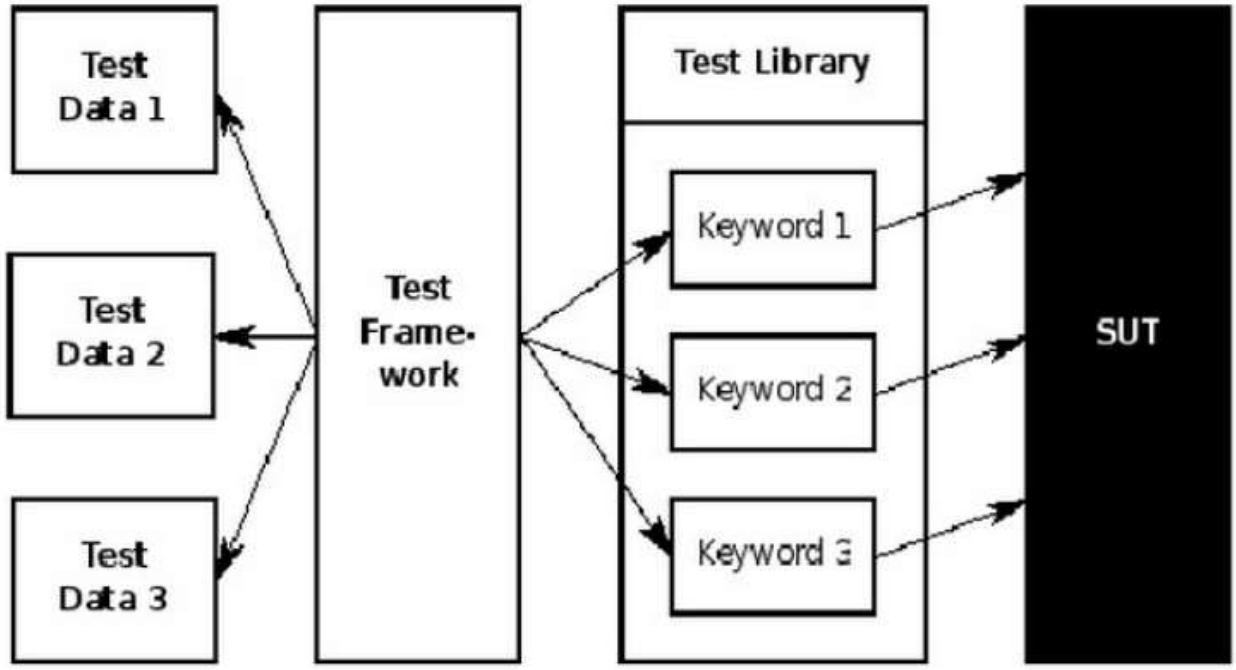
Με αυτόν τον τύπο πλαισίων αυτοματοποιημένου ελέγχου μπορεί να εκτελεστεί ένας μεγάλος αριθμός ελέγχων από ένα σενάριο, αντί να δημιουργηθεί ένας μεγάλος αριθμός σεναρίων για να ελεγχθεί κάθε μια συνθήκη ξεχωριστά. Ενδείκνυται για εφαρμογές που είναι στο στάδιο της υλοποίησης.

Μια βελτίωση της βασικής τεχνικής είναι η επιλογή της *ακολουθίας ελέγχου* (*test sequence*), δηλαδή η σειρά που εισέρχονται τα δεδομένα ελέγχου στο αντικείμενο μπορεί πλέον να καθοριστεί. Επιπλέον, στην ακολουθία ελέγχου μπορούμε να παραλείψουμε δεδομένα, μετατρέποντας έτσι τα αρχεία δεδομένων ελέγχου σε πραγματικά σενάρια ελέγχου. Αυτό είναι ένα βήμα ώστε ο έλεγχος αυτός να μην είναι απλά ένα «φόρτωμα» δεδομένων στο σύστημα.

4.3.3.5 Τεχνική βάσει λέξεων-κλειδιών

Ο *έλεγχος βάσει λέξεων-κλειδιών,* γνωστός και ως *έλεγχος βάσει πίνακα* (*keyword-driven testing or table-driven testing framework*), είναι μια μεθοδολογία ελέγχου λογισμικού τόσο για «χειρωνακτικό» όσο και για αυτοματοποιημένο έλεγχο. Η μέθοδος ξεχωρίζει την τεκμηρίωση περιπτώσεων ελέγχου, συμπεριλαμβανομένου των δεδομένων ελέγχου, από τον τρόπο που θα εκτελεστούν οι περιπτώσεις ελέγχου. Οι λέξεις-κλειδιά κάνουν χρήση μιας ειδικής *«μετα-γλώσσας» υψηλού επιπέδου* (*high-level meta language*), αρκετά περιγραφικής ως προς το αντικείμενο που ελέγχουν. Η «μετα-γλώσσα» είναι διαφορετική για κάθε πλαίσιο αυτοματοποιημένου ελέγχου.

Ο έλεγχος βάσει λέξεων-κλειδιών μοιάζει με τον έλεγχο βάσει δεδομένων, με τη διαφορά ότι οι περιπτώσεις ελέγχου του πρώτου περιλαμβάνονται στα αρχεία των δεδομένων ελέγχου και όχι στο σενάριο ελέγχου. Το σενάριο ελέγχου στην περίπτωση αυτή είναι απλώς ένας «οδηγός» για τα δεδομένα που είναι αποθηκευμένα στις πηγές δεδομένων. Αντίθετα, στη τεχνική βάσει λέξεων-κλειδιών τα δεδομένα ελέγχου γίνονται τα σενάρια ελέγχου, τα οποία ορίζουν «τι θα κάνει» ο έλεγχος και με ποία σειρά θα το κάνει, και μαζί με τα δεδομένα ελέγχου «οδηγούν» το αντικείμενο υπό έλεγχο και την εκτέλεση του ελέγχου.



Σχήμα 3.7: Σχηματική απεικόνιση της τεχνικής βάσει λέξεων-κλειδιών.

Η τεχνική χρησιμοποιεί *λέξεις-κλειδιά* (*keywords*) για να συμβολίσει τις λειτουργικότητες που θα ελεγχθούν π.χ. Insert Username, Insert Password, ορίζοντας το σύνολο των ενεργειών που πρέπει να εκτελεστούν για να κάνει ένας χρήστης login. Το συντακτικό λέξεων-κλειδιών χρησιμοποιεί τη μορφή πίνακα, όπου η πρώτη γραμμή περιέχει το τίτλο της περίπτωσης ελέγχου που ελέγχεται, Valid Login, η πρώτη στήλη περιέχει τις λέξεις-κλειδιά και η δεύτερη τα δεδομένα ελέγχου που χρειάζονται για να εκτελεστεί η κάθε λέξη κλειδί.

```
*** Test Cases ***

Valid Login
    Open Browser To Login Page
    Input Username    demo
    Input Password    mode
    Submit Credentials
    Welcome Page Should Be Open
    [Teardown]    Close Browser
```

Σχήμα 3.8: Παράδειγμα περίπτωσης ελέγχου σε πλαίσιο βάσει λέξεων-κλειδιών Robot

4.3.3.6 Υβριδική τεχνική

Το πιο συνηθισμένο πλαίσιο αυτοματοποιημένου ελέγχου είναι το *υβριδικό πλαίσιο ελέγχου* (*hybrid testing framework*), το οποίο συνδυάζει κάποιες από των παραπάνω τεχνικές, κατά κύριο λόγο τις τεχνικές βάσει δεδομένων και βάσει λέξεων-κλειδιών. Το υβριδικό πλαίσιο επιτρέπει στα σενάρια ελέγχου της τεχνικής βάσει δεδομένων να εκμεταλλευτούν τις βιβλιοθήκες και όλες τις ωφέλειες της αρχιτεκτονικής της μεθόδου βάσει λέξεων-κλειδιών, με αποτέλεσμα τα σενάρια να είναι πιο συμπαγή και λιγότερο επιρρεπή σε σφάλματα. Συνεπώς, όλοι οι έλεγχοι του υβριδικού αυτοματοποιημένου ελέγχου προσφέρουν υψηλότερο επίπεδο αυτοματοποίησης. Επίσης, τα πλαίσια αυτής της τεχνικής διαχειρίζονται λάθη, εξαιρέσεις του συστήματος ή μη αναμενόμενα παράθυρα. Συνίσταται για μεσαίες και μεγάλες εφαρμογές.

4.3.4 Εργαλεία αυτοματοποιημένου ελέγχου σήμερα

Τα εργαλεία αυτοματοποιημένου ελέγχου λογισμικού, ανάλογα με την προσβασιμότητα στο κώδικα και την διάθεσή τους, χωρίζονται στις εξής τρεις κατηγορίες:

• ***Εμπορικά εργαλεία* (*Commercial tools*).**

Καλά εργαλεία αλλά και ακριβά, ακόμα και αυτά που έχουν φθηνές άδειες (licenses) δεν επιτρέπουν τη χρήση τους από ολόκληρη την ομάδα. Επίσης, είναι δύσκολο να συνενωθούν με άλλα εργαλεία αυτοματοποιημένου ελέγχου, ειδικά με εργαλεία άλλης εταιρίας, και είναι δύσκολο έως αδύνατο να προσαρμοστούν από το χρήστη. Τέλος, υπάρχει ο κίνδυνος η εταιρία να κλείσει ή το εργαλείο να αποσυρθεί.

• ***Εργαλεία ανοιχτού λογισμικού* (Open source tools)**

Υπάρχει μεγάλη ποικιλία εργαλείων ανοιχτού λογισμικού, επιρεάζοντας όμως το επίπεδο ποιότητας. Βασικό τους χαρακτηριστικό είναι ότι διατίθονται δωρεάν και ελεύθερα να χρησιμοποιηθούν από όλους με έμφαση στη δυνατότητα προσαρμογής στις ανάγκες της εφαρμογής. Επιπλέον είναι πολύ εύκολο να συνενωθούν με άλλα εργαλεία.

• ***Ελεύθερα εργαλεία* (Freeware tools)**

Ελεύθερα εργαλεία είναι πλέον σπάνια σήμερα, αφού όλα είναι και ανοιχτού λογισμικού. Όσα υπάρχουν, είναι δωρεάν και χωρίς κόστος άδειας (license). Είναι πιο εύκολο να τα συνενώσεις με άλλα αυτοματοποιημένα εργαλεία από ότι τα εμπορικά, άλλα όχι τόσο εύκολο όσο τα ανοιχτού λογισμικού. Επίσης, ελοχεύει ο κίνδυνος απόσυρσης του εργαλείου.

Τα σημαντικότερα εργαλεία ή πλαίσια αυτοματοποιημένου λειτουργικού ελέγχου που κυκλοφορούν σήμερα, μπορούν να διαχωριστούν ανάλογα με το επίπεδο και τον έλεγχο που εκτελούν σε:

Εργαλεία για αυτοματοποιημένες δοκιμασίες μονάδας:

• **JUnit [23]**

Το JUnit είναι ένα πλαίσιο ανοιχτού λογισμικού για τις δοκιμασίες μονάδας της γλώσσας προγραμματισμού Java. Ανήκει στην κατηγορία πλαισίων αρχιτεκτονικής xUnit και συνδέεται σαν JAR κατά τη διάρκεια της μεταγλώττισης του κώδικα. Ο JUnit έλεγχος είναι ουσιαστικά ένα αντικείμενο Java, όπου οι μέθοδοι του ελέγχου συνοδεύονται από το λεκτικό “@Test”. Είναι επίσης εφικτό να οριστεί πότε η μέθοδος θα εκτελέσει τον έλεγχο πριν (@Before) ή μετά (@After), για μία ή όλες τις μεθόδους (@BeforeClass ή @AfterClass).

• **NUnit [24]**

NUnit είναι επίσης ένα πλαίσιο ανοιχτού λογισμικού για τις δοκιμασίες μονάδας που απευθύνεται σε όλες της γλώσσες .NET. Αυτό το πλαίσιο xUnit αρχιτεκτονικής ανήκει στην εταιρία Microsoft και είναι εξολοκλήρου γραμμένο σε C#.

Εργαλεία για αυτοματοποιημένες δοκιμασίες αποδοχής:

• **Robot Framework [25]**

Το Robot Framework είναι ένα γενικό πλαίσιο αυτοματοποιημένου ελέγχου για δοκιμές αποδοχής. Ακολουθεί την τεχνική βάσει λέξεων-κλειδιών και χρησιμοποιεί συντακτικό για τα δεδομένα ελέγχου σε μορφή πίνακα. Οι δυνατότητες του πλαισίου επεκτείνονται από τις διαθέσιμες βιβλιοθήκες ελέγχου που είναι υλοποιημένες σε Python ή Java. Οι χρήστες μπορούν επίσης να δημιουργήσουν λέξεις-κλειδιά από τις ήδη υπάρχουσες, χρησιμοποιώντας το ίδιο συντακτικό με αυτό των περιπτώσεων ελέγχου.

Οι πίνακες μπορούν να είναι σε μορφή απλού κειμένου (.txt), HTML (.html, .htm, xhml), tab-separated values (.tsv), ή eStructured Text rest (.robot) τύπους αρχείων και γράφονται σε ένα οποιοδήποτε επεξεργαστή κειμένου (text editor) ή στο Robot Integrated Development Environment (RIDE). Ο RIDE απλοποιεί το γράψιμο των περιπτώσεων ελέγχου παρέχοντας στο πλαίσιο *ειδική συμπλήρωση κώδικα* (*code completion*), *επισήμανση συντακτικού (syntax highlighting)* κ.ά..

Το Robot είναι ανεξάρτητο από την εφαρμογή και το λειτουργικό σύστημα στο οποίο τρέχει. Είναι επίσης ανοιχτού λογισμικού, όπως και οι περισσότερες βιβλιοθήκες και εργαλεία που υποστηρίζει. Η ανάπτυξη του πλαισίου υποστηρίζεται από την εταιρία Nokia Networks.

• **FitNesse [26]**

Το FitNesse είναι ένα πλαίσιο αυτοματοποιημένου ελέγχου λογισμικού για δοκιμασίες αποδοχής γραμμένο σε Java υποστηρίζοντας επίσης C++, Python, Ruby, Delphi, C# κ.ά..

Οι δοκιμασίες που γίνονται στο FitNesse βασίζονται στην τεχνική των λέξεων-κλειδιών και είναι οι περιπτώσεις ελέγχου σε μορφή ζευγαριού των δεδομένων εισόδου και αναμενόμενων δεδομένων εξόδου. Τα ζευγάρια εκφράζονται σε διάφορες παραλλαγές πίνακα όπως πίνακες ελέγχων που εκτελούν queries, πίνακες που εκφράζουν σενάρια ελέγχου με ακριβή αρίθμηση των βημάτων που πρέπει να ακολουθηθούν για να φτάσουν στο αποτέλεσμα, όπως επίσης και πίνακες ελεύθερης δομής.

• **Cucumber [27]**

Το Cucumber είναι ένα πλαίσιο ανοιχτού λογισμικού γραμμένο σε Ruby που αφορά πάντα δοκιμασίες αποδοχής. Δεν απευθύνεται μόνο σε εφαρμογές λογισμικού της Ruby π.χ. cuke4php και cuke4lua. Το πλαίσιο Cucumber επιτρέπει επίσης την εκτέλεση επίσημων εγγράφων με τα τεχνικά *χαρακτηριστικά λογισμικού* (*feature documentation*) σε μορφή ειδικού συντακτικού πλαισίου.

Εργαλεία για αυτοματοποιημένες δοκιμασίες γραφικού περιβάλλοντος χρήστη:

• **Selenium [28]**

Το Selenium είναι ένα εργαλείο ελέγχου λογισμικού για εφαρμογές του διαδικτύου. Αφορά μόνο δοκιμασίες γραφικού περιβάλλοντος και είναι ανοιχτού λογισμικού. Παρέχει την τεχνική «καταγραφής/επανάληψης» μέσω του Selenium IDE που υπάρχει σαν plug-in στους περισσότερους μοντέρνους *περιηγητές διαδικτύου* (web browsers), όπως οι Firefox, Chrome.

Παρέχει επίσης και ένα ολοκληρωμένο πλαίσιο για τη δημιουργία ελέγχων σε ένα μεγάλο αριθμό προγραμματιστικών γλωσσών όπως Java, C#, Groovy, Perl, PHP, Python και Ruby.

• **Watir [29]**

Το Watir (προφέρεται σαν “water”) είναι ένα εργαλείο ανοιχτού λογισμικού με όλες τις οικογένειες βιβλιοθηκών της Ruby για την αυτοματοποίηση των περιηγητών του διαδικτύου (web browsers). Υποστηρίζει όλες τις εφαρμογές διαδικτύου ανεξάρτητα από την τεχνολογία με την οποία έχουν αναπτυχθεί. Ενώ το Watir υποστηρίζει αποκλειστικά τον Internet Explorer στο λειτουργικό των Windows, το Watir-Web Driver υποστηρίζει πλέον και τους Chrome, Firefox και Opera. Το εργαλείο παρέχει τη δυνατότητα συνδέσης του έλεγχου με βάσεις δεδομένων, διαβάζει αρχεία δεδομένων και spreadsheets, εξάγει XML και δομεί τον κώδικα μέσω των διαθέσιμων βιβλιοθηκών.

• **QTP και UFT [30]**

Το Quick Test Professional (QTP) μαζί με το Service Test (ST) αποτελούν πλέον ένα ενοποιημένο πλαίσιο ελέγχου λογισμικού, το HP Unified Functional Testing (UFT), ένα εμπορικό εργαλείο που ανήκει στην εταιρία Hewlett Packard.

Παρέχει αυτοματοποιημένο λειτουργικό έλεγχο λογισμικού και έλεγχο παλινδρόμησης για εφαρμογές λογισμικού και περιβάλλοντα. Το ενοποιημένο εργαλείο επιτρέπει στους προγραμματιστές να ελέγξουν από μία κονσόλα και τις τρεις βαθμίδες του λογισμικού. Στη μεσαία βαθμίδα ακολουθεί μια υβριδική τεχνική, υποστηρίζοντας τις μεθόδους βάσει δεδομένων και βάσει λέξεων-κλειδιών. Χρησιμοποιεί τη γλώσσα σεναρίων6 Visual Basic Scripting Edition (VB Script) για να προσδιορίσει τα σενάρια ελέγχου και να διαχειριστεί τα αντικείμενα της εφαρμογής υπό έλεγχο.

• **Rational Functional Tester [31]**

Το αυτοματοποιημένο πλαίσιο ελέγχου Rational Functional Tester είναι εμπορικό εργαλείο και ανήκει στην IBM. Επιτρέπει λειτουργικό έλεγχο και έλεγχο παλινδρόμησης στο γραφικό περιβάλλον, ενώ υποστηρίζει τις τεχνικές «καταγραφής/επανάληψης» με δυνατότητα επεξεργασίας των παραγόμενων σεναρίων. Επιπλέον παρέχει τεχνική βάσει δεδομένων για την εισαγωγή δεδομένων στο υπό έλεγχο αντικείμενο. Το Rational Function Tester υποστηρίζει ένα μεγάλο φάσμα εφαρμογών, όπως τις εφαρμογές διαδικτύου, .Net, Java, Siebel, SAP, τερματικές εφαρμογές βάσει εξομοιωτή, PowerBuilder, Ajax, Adobe Flex, Dojo Toolkit, GEF, Adobe PDF αρχεία, zSeries, iSeries, και pSeries.

• **Galen Framework [32]**

Τα τελευταία χρόνια στα αυτοματοποιημένα εργαλεία ελέγχου γραφικού περιβάλλοντος του χρήστη προστίθενται και εργαλεία για τον έλεγχο του responsive web design, δηλαδή για τον έλεγχο των γραφικών περιβαλλόντων σε μικρότερες οθόνες όπως π.χ. σε smartphones και tablets.

Το πλαίσιο ελέγχου Galen χρησιμοποιεί την τεχνική έλεγχου βάσει λέξεων-κλειδιών και ελέγχει τις αποστάσεις των αντικειμένων γραφικού περιβάλλοντος σε σχέση με το μέγεθος της σελίδας ή της οθόνης. Με απλά λόγια, ανοίγει ένα περιηγητή διαδικτύου (μέσω του εργαλείου Selenium), αναπροσαρμόζει το μέγεθος του και μετά ελέγχει τα αντικείμενα σύμφωνα με τις προδιαγραφές. Για τον ορισμό των προδιαγραφών χρησιμοποιεί ένα ειδικό συντακτικό με κατανοητούς όρους και κανόνες. Τέλος, παράγει HTML αναφορές με screenshots των οθονών και υπογραμμισμένα όλα τα στοιχεία που δε συμφωνούν με τις προδιαγραφές.

Πλαίσια αυτοματοποιημένου μη λειτουργικού ελέγχου για ελέγχους επίδοσης, φόρτου και έντασης:

• **JMeter [33]**

Το Apache JMeter είναι μια εφαρμογή ανοικτού λογισμικού γραμμένη σε Java που ανάμεσα σε άλλες λειτουργίες μπορεί να φορτώνει το υπό έλεγχο αντικείμενο και να μετρά τις επιδόσεις του. Μπορεί λοιπόν να εκτελέσει ελέγχους επιδόσεων, φόρτου και έντασης.

Αρχικά, προοριζόταν μόνο για εφαρμογές διαδικτύου αλλά έχει πλέον επεκταθεί και σε άλλα αντικείμενα ελέγχου, υποστηρίζοντας πολλά διαφορετικά πρωτόκολλα: HTTP, HTTPS, SOAP, FTP, Database via JDBC, LDAP, MongoDB (NoSQL), TCP κ.ά.

• **HP LoadRunner [34]**

Το LoadRunner είναι ένα εμπορικό αυτοματοποιημένο πλαίσιο ελέγχου επιδόσεων και φόρτου που ανήκει στην Hewlett-Packard. Η λογική του πλαισίου είναι να δημιουργεί εικονικούς χρήστες που παίρνουν τη θέση των πραγματικών χρηστών του προϊόντος λογισμικού. Μπορεί να προσομοιώσει χιλιάδες χρήστες που λειτουργούν ταυτόχρονα σε μια εφαρμογή, συλλέγοντας πληροφορίες από τις μονάδες συστήματος (web servers, database servers κτλ.) και καταγράφοντας τα αποτελέσματα, τα οποία μπορούν να αναλυθούν σε λεπτομέρεια για την ανίχνευση των προβλημάτων.

Το LoadRunner αποτελείται από τέσσερις μονάδες:

• *VuGen (Virtual User Generator)* για τη δημιουργία και επεξεργασία των σεναρίων. Υποστηρίζει την τεχνική της «καταγραφής/επανάληψης» και επιτρέπει την εισαγωγή δομημένου κώδικα. Τα σενάρια γράφονται σε ANSI-C, μια ειδική γλώσσα που μοιάζει πολύ με τη C.

• *Load generator* για την παραγωγή του φόρτου στην εφαρμογή.

• *Controller* για τη δημιουργία των περιπτώσεων ελέγχου. Εκεί ορίζεται ποια σενάρια, για πόσους χρήστες και για πόση ώρα θα τρέξει ο κάθε load generator.

• Analysis για την συγκομιδή των *αρχείων* (*logs*) από τους load generators και τη δημιουργία reports για την οπτικοποίηση των αποτελεσμάτων.

4.4 Έλεγχος λογισμικού στο «πρόβλημα των τριγώνων»

4.4.1. Το «πρόβλημα των τριγώνων»

Το «Πρόβλημα των τριγώνων», γνωστό επίσης ως πρόβλημα τριγώνων των Weinberg-Myers, είναι ένα κλασικό πρόβλημα ελέγχου που τέθηκε από τον Jerry Weinberg και δημοσιεύτηκε για πρώτη φορά στο βιβλίο «Η τέχνη του Ελέγχου Λογισμικού» του Glenford Myers, το 1979. Ο ίδιος το χρησιμοποίησε ως παράδειγμα μίας απλής εφαρμογής που όμως χρειάζεται πολλές δοκιμές ελέγχου. Ο Myers υπήρξε ο πρώτος που αντιμετώπισε τον έλεγχο λογισμικού ως ένα εντελώς ανεξάρτητο κομμάτι στην ανάπτυξη λογισμικού και προκάλεσε τους αναγνώστες του να γράψουν περιπτώσεις ελέγχου για το πρόβλημα με τις παρακάτω συνθήκες:

«Το πρόγραμμα διαβάζει 3 ακέραιες τιμές από ένα διάλογο εισόδου. Αυτές αντιπροσωπεύουν το μήκος των πλευρών ενός τριγώνου. Το πρόγραμμα εμφανίζει μήνυμα που δηλώνει αν το τρίγωνο είναι σκαληνό, ισοσκελές ή ισόπλευρο.»

Στην αρχική του μορφή το πρόβλημα χρησιμοποιήθηκε για προγράμματα ανάγνωσης διάτρητων καρτών μηχανής (IBM punch cards). Η εξέταση αυτή ελέγχει την ικανότητα σκέψης και δημιουργίας περιπτώσεων ελέγχου για μια δεδομένη κατάσταση και παρουσιάζει τη διαδικασία σχεδιασμού ελέγχου (test design procedure).

Στο βιβλίο του, επίσης, ο Myers περιγράφει τις 14 δοκιμές που χρειάζονται για να ελεγχθεί επαρκώς ένα τρίγωνο. Στη συνέχεια, στο βιβλίο «Έλεγχος Λογισμικού: Μια πρόχειρη προσέγγιση», ο Paul Jorgensen καταγράφει 185 περιπτώσεις ελέγχου. Ο Bob Binder παίρνει την σκυτάλη του προβλήματος στην εισαγωγή του βιβλίου του «Έλεγχος Αντικειμενοστραφών Συστημάτων» όπου περιγράφει ένα περίτεχνο σύστημα αντικειμένων που κάνουν ουσιαστικά την ίδια δουλειά. Παρότι δεν έχει τις ίδιες προϋποθέσεις με το αρχικό πρόβλημα του Myers ισχυρίζεται ότι χρειάζεται 65 δοκιμές για να το ελέγξει. Απ' την άλλη, ο Kent Beck ισχυρίζεται ότι χρειάζεται μόνο 6 για την λύση του προβλήματος ακολουθώντας την *τεχνική ανάπτυξης λογισμικού βάσει ελέγχου* (Test-driven Development).

Το πρόβλημα των τριγώνων είναι ιδιαίτερα δημοφιλές στην κοινότητα των testers, καθώς πολλά βιβλία και ιστολόγια έχουν πολλάκις ασχοληθεί. Στην συγκεκριμένη εργασία επιλέχθηκε η web 2.0 υλοποίηση της Elizabeth Hendrickson 7 για το κλασικό πρόβλημα ελέγχου τριγώνων, καθώς ο κώδικας είναι γραμμένος σε JavaScript και συνεπώς είναι διαθέσιμος για έλεγχο. Η επιλογή αυτής της εφαρμογής διαδικτύου καθορίστηκε απ' την ανάγκη να γίνει ανεξάρτητος και αντικειμενικός έλεγχος μιας εφαρμογής που δεν έχει υλοποιηθεί στο πλαίσιο της διπλωματικής εργασίας, ικανοποιώντας έτσι το βασικό κανόνα του Ελέγχου Λογισμικού περί ανεξαρτησίας των ομάδων ανάπτυξης λογισμικού και ελέγχου.

4.4.2 Έλεγχος λογισμικού στην εφαρμογή των τριγώνων

Για τον έλεγχο λογισμικού του προβλήματος των τριγώνων, όπως και κάθε προβλήματος ελέγχου, και τη δημιουργία περιπτώσεων ελέγχου δεν υπάρχει σωστή θεωρητική απάντηση και ιδανικός τρόπος εκτέλεσης. Η απάντηση εξαρτάται από τον tester, τις υποθέσεις που θα πάρει και τις μεθόδους που θα επιλέξει.

Για αρχή ο tester οφείλει να καταλάβει το πρόβλημα και την εφαρμογή που πρέπει να ελέγξει μελετώντας τις απαιτήσεις του προβλήματος. Οι συνθήκες της εφαρμογής των τριγώνων ορίζονται από την δημιουργό της εφαρμογής παρακάτω:

«*Παίρνει ως είσοδο τρεις αριθμούς που αντιπροσωπεύουν το μήκος των τριών πλευρών του τριγώνου. Αυτόματα8 το πρόγραμμα σχεδιάζει μια εικόνα του τριγώνου με το μέγεθος των πλευρών σε αναλογία και δηλώνει τον τύπο του τριγώνου.»*

Οι συνθήκες που δόθηκαν θεωρούνται οι απαιτήσεις του προβλήματος, και όπως συμβαίνει στις περισσότερες εφαρμογές ή συστήματα του πραγματικού κόσμου, οι πληροφορίες που έχουμε για το λογισμικό είναι είτε ελλιπής είτε ασαφής. Παρατηρούμε ότι η Hendrickson, σε αντίθεση με τον Myers, δεν αναφέρει αν τα δεδομένα εισόδου περιορίζονται στους ακεραίους αριθμούς και δεν προσδιορίζει ποιοι είναι οι τύποι τριγώνων που αναγνωρίζει και εμφανίζει το σύστημα.

Συνεπώς, περαιτέρω ανάλυση των απαιτήσεων είναι αναγκαία για την κάλυψη αυτών των κενών και τη δημιουργία *υποθέσεων* (*assumptions*) πάνω στις οποίες θα βασιστεί ο έλεγχος λογισμικού.

Κατά τον έλεγχο λογισμικού που πραγματοποιείται «χειρωνακτικά» (δηλαδή χωρίς καμία αυτοματοποίηση αλλά με μία απλή αλληλεπίδραση με την εφαρμογή από τη θέση του τελικού χρήστη) θα πραγματοποιήσουμε δύο είδη διερευνητικών ελέγχων. Αρχικά θα εκτελέσουμε ένα «έλεγχο καπνού» για να την εξέταση των βασικών περιπτώσεων και στη συνέχεια ένα πιο ενδελεχή έλεγχο βάσει απαιτήσεων, με τη μέθοδο του black box.

4.4.2.1 Έλεγχος καπνού της εφαρμογής

Εν αρχή θα κάνουμε ένα «έλεγχο καπνού» για να εξοικειωθούμε με την εφαρμογή και να ελέγξουμε αν οι πολύ βασικές περιπτώσεις ελέγχου λειτουργούν σύμφωνα με τις απαιτήσεις. Στον έλεγχο αυτό θα ελέγξουμε ότι η εφαρμογή δέχεται τρεις τιμές εισόδου, που αναπαριστούν τις τρεις πλευρές του τριγώνου, (δεδομένα εισόδου) και επιστρέφει τον τύπο του τριγώνου και την γραφική αναπαράσταση του σε αναλογία (δεδομένα εξόδου).

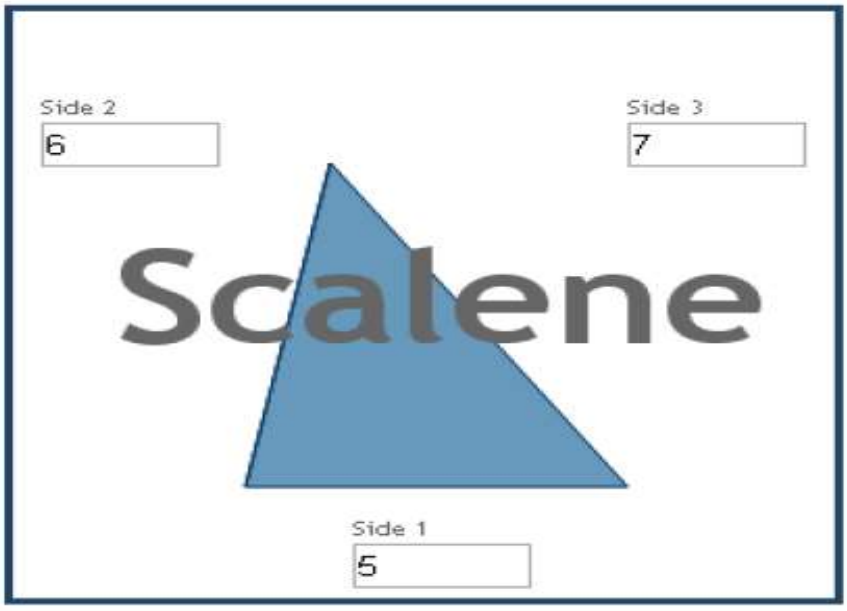
Για τον «έλεγχο καπνού» της εφαρμογής των τριγώνων αρκεί να εξεταστούν οι βασικοί τύποι τριγώνων. Επομένως, πρέπει να ελεγχθούν οι περιπτώσεις όπου τα δεδομένα εισόδου σχηματίζουν ισόπλευρο, ισοσκελές και σκαληνό τρίγωνο. Σε αυτό το σημείο αναρωτιόμαστε αν υπάρχει και άλλος τύπος τριγώνου που δεν καλύπτεται από τις προδιαγραφές του Myers.

Όντως, υπάρχει επίσης το ορθογώνιο τρίγωνο, το οποίο πρέπει να λάβουμε υπόψη μας. Σε όλες τις περιπτώσεις ελέγχουμε αν η εφαρμογή επιστρέφει το σωστό τύπο τριγώνου ("Equilateral", "Isosceles" και "Scalene"), αν σχηματίζει σωστά το τρίγωνο και αν αυτό είναι σε αναλογία με τις τιμές που δώσαμε για

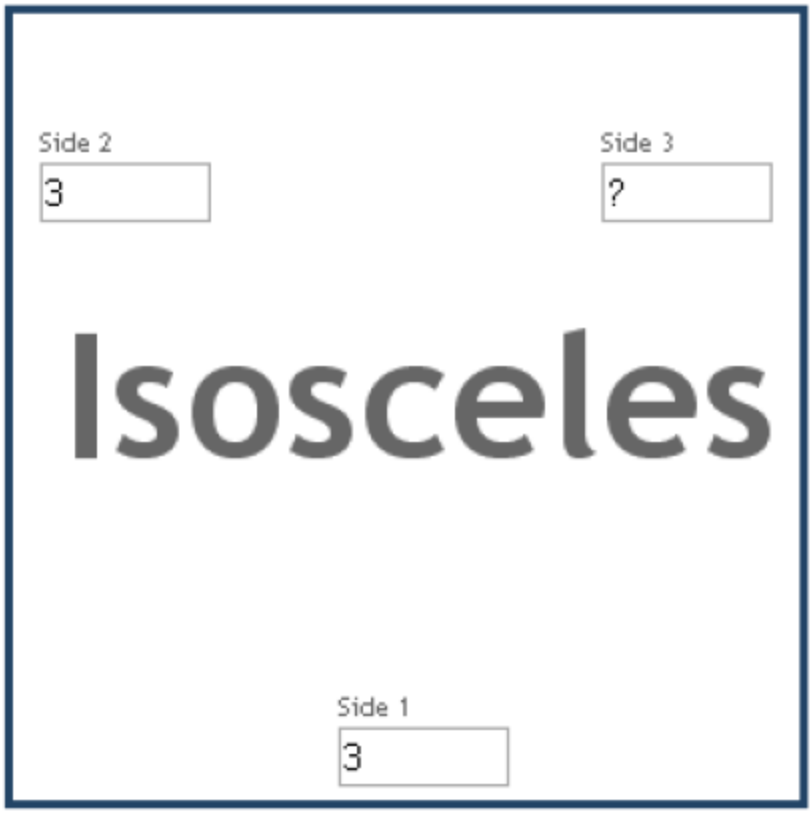
την κάθε πλευρά. Επιπλέον, θα προσθέσουμε και μια μη έγκυρη περίπτωση ελέγχου, όπου τα δεδομένα εισόδου δεν είναι σωστά, για να εξετάσουμε πως ανταποκρίνεται η εφαρμογή.

Οι περιπτώσεις του «ελέγχου καπνού» είναι:

- Για τα δεδομένα εισόδου **(1, 1, 1)** ο τύπος τριγώνου είναι: **Equilateral.**
- Για τα δεδομένα εισόδου **(2, 2, 3)** ο τύπος τριγώνου είναι: **Isosceles.**
- Για τα δεδομένα εισόδου **(5, 6, 7)** ο τύπος τριγώνου είναι: **Scalene.**
- Για τα δεδομένα εισόδου **(3, 4, 5)** ο τύπος τριγώνου είναι: **Right.**
- Για τα δεδομένα εισόδου **(3, 3, ?)** ο τύπος τριγώνου είναι: **Isosceles.**



Σχήμα 4.1: Γραφική αναπαράσταση τριγώνου για τις τιμές (5, 6, 7).



Σχήμα 4.2: Γραφική αναπαράσταση τριγώνου για τα μη έγκυρα δεδομένα (3, 3, ?)

Εκτελούμε τον «έλεγχο καπνού» και παρατηρούμε ότι οι τέσσερις έγκυρες περιπτώσεις *«πέρασαν»* τον έλεγχο (*pass*), ενώ η μη έγκυρη περίπτωση *δεν «πέρασε»* (*fail*). Κατά την μη έγκυρη περίπτωση ελέγχου το σύστημα επέστρεψε ως τύπο τριγώνου “Isosceles” και δεν σχημάτισε γραφικά καθόλου το τρίγωνο. Η εφαρμογή θα έπρεπε να μην επιτρέπει στο χρήστη την εισαγωγή ειδικού χαρακτήρα, επιστρέφοντας ίσως κάποιο *μήνυμα λάθους* (*error message*) ή επιστρέφοντας ως τύπο τριγώνου “Invalid”.

Παρόλο που ο «έλεγχος καπνού» βρήκε ένα ελάττωμα στο λογισμικό, υπάρχουν πολλά ελαττώματα που δε θα φανερωθούν με αυτές τις πέντε περιπτώσεις ελέγχου και για αυτό απαιτείται ένας πιο ενδελεχής έλεγχο. Η ερώτηση «πόσες περιπτώσεις αρκούν για να ελέγξουμε καλά» την εφαρμογή μπορεί να απαντηθεί μόνο με βάση την επικινδυνότητα της εφαρμογής και την προσπάθεια που χρειάζεται να καταβληθεί για να αποφευχθεί αποτυχία (effort). Αν η επικινδυνότητα είναι σχετικά χαμηλή τότε μπορεί αυτές οι πέντε περιπτώσεις ελέγχου να αρκούν. Στην περίπτωση μας, το επίπεδο επικινδυνότητας θεωρείται υψηλό και για το λόγο αυτό θα εκτελεστεί *εκτενής έλεγχο (thorough testing)*.

4.4.2.2 Black box έλεγχος της εφαρμογής

Αφού έχουμε μια πρώτη εικόνα για την εφαρμογή από τον «έλεγχο καπνού», το επόμενο βήμα είναι ο έλεγχος λογισμικού βάσει απαιτήσεων με τη μέθοδο του black box. Δηλαδή χωρίς να λάβουμε υπόψη μας τον κώδικα της εφαρμογής και με μόνη γνώση τις απαιτήσεις θα εκτελέσουμε τον έλεγχο λογισμικού. Απαιτείται να αναλύσουμε τις προδιαγραφές που μας έχουν δοθεί και ταυτόχρονα να διεξάγουμε διερευνητικό έλεγχο για την συμπλήρωση αυτών.

Από τις απαιτήσεις της εφαρμογής υπό έλεγχο προκύπτουν οι παρακάτω ερωτήσεις:

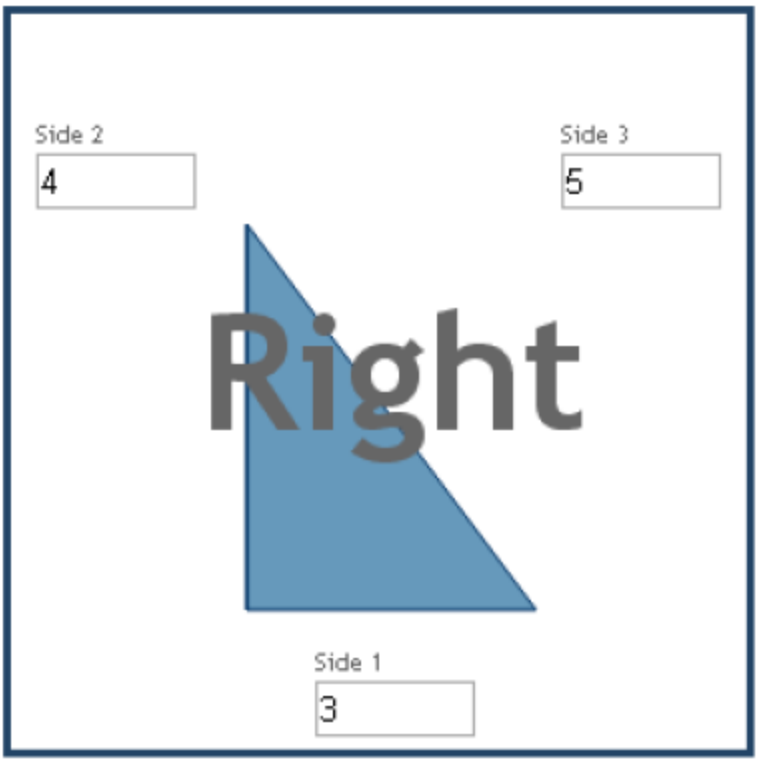
- Τι είναι τρίγωνο και πως ορίζεται;
- Πόσοι τύποι τριγώνων υπάρχουν και πώς ορίζονται αυτοί;
- Τι είδους αριθμούς δέχεται ως εισόδους η εφαρμογή;
- Οι αριθμοί αυτοί έχουν περιορισμένο πεδίο ορισμού (ακρότατες τιμές);
- Τρεις οποιοδήποτε αριθμοί φτιάχνουν ένα τρίγωνο;

Για την απάντηση αυτών των ερωτήσεων πρέπει να γίνουν κάποιες υποθέσεις, αφού οι απαιτήσεις έχουν κενά και δεν εκτελούμε white box έλεγχο για να ξέρουμε πως ο προγραμματιστής έφτιαξε τον κώδικα. Οι υποθέσεις είναι μέγιστης σημασίας για το σχεδιασμό του ελέγχου διότι πάνω σε αυτές θα βασιστούν οι περιπτώσεις ελέγχου. Επιπλέον πρέπει πάντα να καταγράφονται και να επιθεωρούνται από την υπόλοιπη ομάδα ελέγχου έτσι ώστε να υπάρχει μια κοινή αντίληψη για το αντικείμενο υπό έλεγχο.

Στην περίπτωση του προβλήματος των τριγώνων, η πιο βασική υπόθεση που πρέπει να γίνει είναι αυτή για τον ορισμό του τριγώνου. Οι υποθέσεις λαμβάνουν πάντα υπόψη της τον πελάτη ή τον τελικό χρήστη που απευθύνεται η εφαρμογή. Ως εκ τούτου, ένας διδακτορικός φοιτητής Μαθηματικών ή ένα παιδί δημοτικού έχουν μια πολύ διαφορετική αντίληψη του τριγώνου από ένα κοινό ορισμό9. Αν αναζητήσουμε τον ορισμό σε λεξικό, το τρίγωνο ορίζεται ως «το σχήμα που περιφράζεται από τρεις γραμμές»[17] ή «το πολύγωνο με τρεις πλευρές»[18], όπου «πολύγωνο είναι ένα σχήμα του δισδιάστατου χώρου περιφραγμένο με ευθείες γραμμές»[18]. Στην παρούσα ανάλυση θα βασιστούμε στο δεύτερο ορισμό που είναι πιο αυστηρός.

Επίσης, από την Ευκλείδεια Γεωμετρία ορίζουμε τους βασικούς τύπους τριγώνων ως προς τις πλευρές: Το *ισόπλευρο τρίγωνο* ως το τρίγωνο που έχει τρεις ίσες πλευρές, το *ισοσκελές* ως το τρίγωνο που έχει ακριβώς δύο πλευρές ίσες μεταξύ τους και το *σκαληνό* ως αυτό που καμία από τις πλευρές δεν είναι ίση μεταξύ τους. Υπάρχουν δύο είδη σκαληνών τριγώνων, το *οξυγώνιο* με τρεις γωνίες οξείες και το *αμβλυγώνιο* με μια γωνία αμβλεία. Επίσης, *ορθογώνιο τρίγωνο* ορίζεται το τρίγωνα με μια γωνία ορθή.

Οι πλευρές που περιέχουν την ορθή γωνία ενός ορθογωνίου λέγονται κάθετες πλευρές και η απέναντι της λέγεται υποτείνουσα. Κριτήριο για να είναι ένα τρίγωνο ορθογώνιο είναι το Πυθαγόρειο θεώρημα, δηλαδή «το τετράγωνο της υποτείνουσας ενός ορθογώνιου τριγώνου ισούται με το άθροισμα των τετραγώνων των δύο κάθετων πλευρών».



Σχήμα 4.3: Γραφική αναπαράσταση ορθογωνίου τριγώνου με τιμές (3, 4, 5).

Ένας πολύ μεγάλος αριθμός σφαλμάτων προέρχεται από τις ίδιες τις απαιτήσεις και τις προδιαγραφές του συστήματος, έτσι δεν μπορούμε να υποθέσουμε ότι οι απαιτήσεις που μας δόθηκαν για το πρόβλημα είναι ολοκληρωμένες και σωστές. Στην περίπτωση μας, οι προδιαγραφές δεν αναφέρουν τι είδους είναι τα δεδομένα εισόδου και ποιό είναι το πεδίο ορισμού τους. Οπότε είναι απαραίτητο να ελεγχθεί αν τα δεδομένα ελέγχου μπορούν να περιέχουν δεκαδικό μέρος ή είναι απλώς ακέραιοι.

• Για τα δεδομένα εισόδου **(3.2, 5.2, 6.2)** ο τύπος τριγώνου είναι: **Equilateral**

Συνεπώς οι δεκαδικοί αριθμοί είναι αποδεκτοί και πρέπει να δοκιμαστούν διεξοδικά για την εύρεση ελαττωμάτων που προκαλούνται από *εσωτερικές αριθμητικές υπερχειλίσεις (internal arithmetic overflow)* και λάθη στρογγυλοποίησης.

Επίσης, πρέπει να ελέγξουμε την εφαρμογή για το ποιο είναι το μεγαλύτερο και μικρότερο επιτρεπτό μήκος των δεδομένων εισόδου, αφού αυτό δεν προσδιορίζεται από τις απαιτήσεις. Ξέρουμε ότι οι αρνητικοί αριθμοί και το μηδέν δεν είναι επιτρεπτά δεδομένα για πλευρές τριγώνου. Δοκιμάζουμε άμεσα:

• Για τα δεδομένα εισόδου **(0.1, 0.1, 0.1)** ο τύπος τριγώνου είναι: **Degenerate.**

• Για τα δεδομένα εισόδου **(0.9, 1, 1)** ο τύπος τριγώνου είναι: **Degenerate.**

• Για τα δεδομένα εισόδου **(100, 100, 100)** ο τύπος τριγώνου είναι: **Equilateral.**

• Για τα δεδομένα εισόδου **(10.000, 10.000, 10.000)** ο τύπος τριγώνου είναι: **Equilateral.**

• Για τα δεδομένα εισόδου **(1.000.000, 1.000000, 1.000.000)** ο τύπος τριγώνου είναι: **Equilateral.**

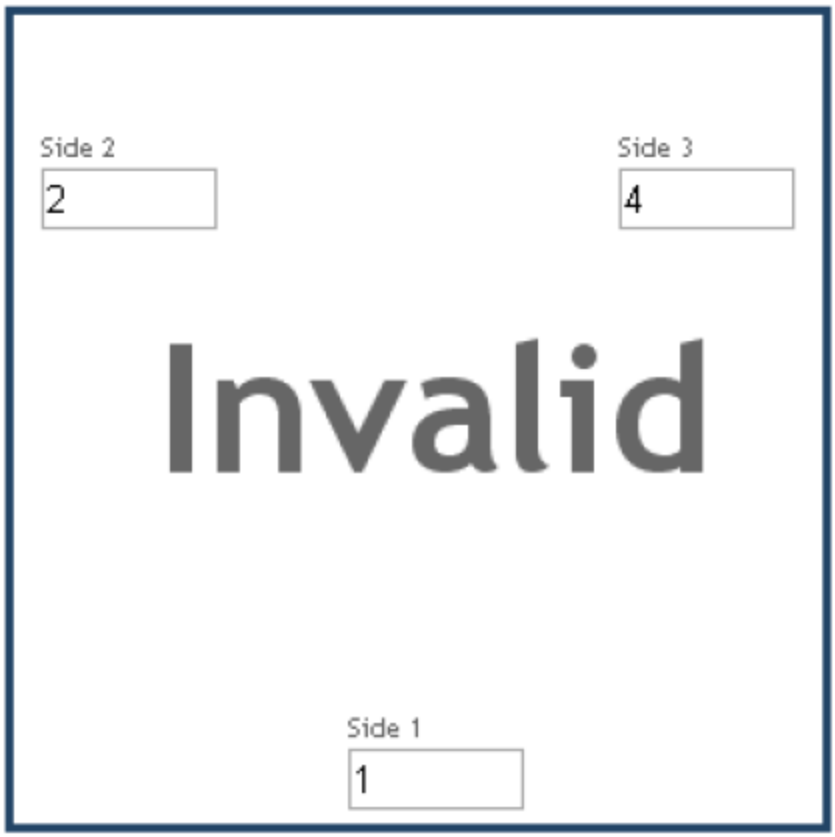
Αφού δοκιμάσουμε τις παραπάνω τριάδες αριθμών παρατηρούμε ότι το σύστημα δεν δέχεται αριθμούς μικρότερους της μονάδας, παρόλο που δέχεται δεκαδικούς αριθμούς, χωρίς όμως να παρουσιάζει κανένα περιορισμό για το άνω όριο των τιμών εισόδου. Ενδεχομένως αυτή η έλλειψη άνω ορίου να προκαλεί πολλά σφάλματα στην εφαρμογή, ειδικά για την κατά αναλογία γραφική αναπαράσταση πολύ μεγάλων τριγώνων. Υποθέτουμε λοιπόν για ακρότατες τιμές εισόδου το 1 και το 999 .

Τέλος, κατά την ανάλυση των απαιτήσεων παρατηρούμε ότι δεν επιστέφουν όλες οι τριάδες δεδομένων εισόδου τρίγωνο. Αν δοκιμάσουμε:

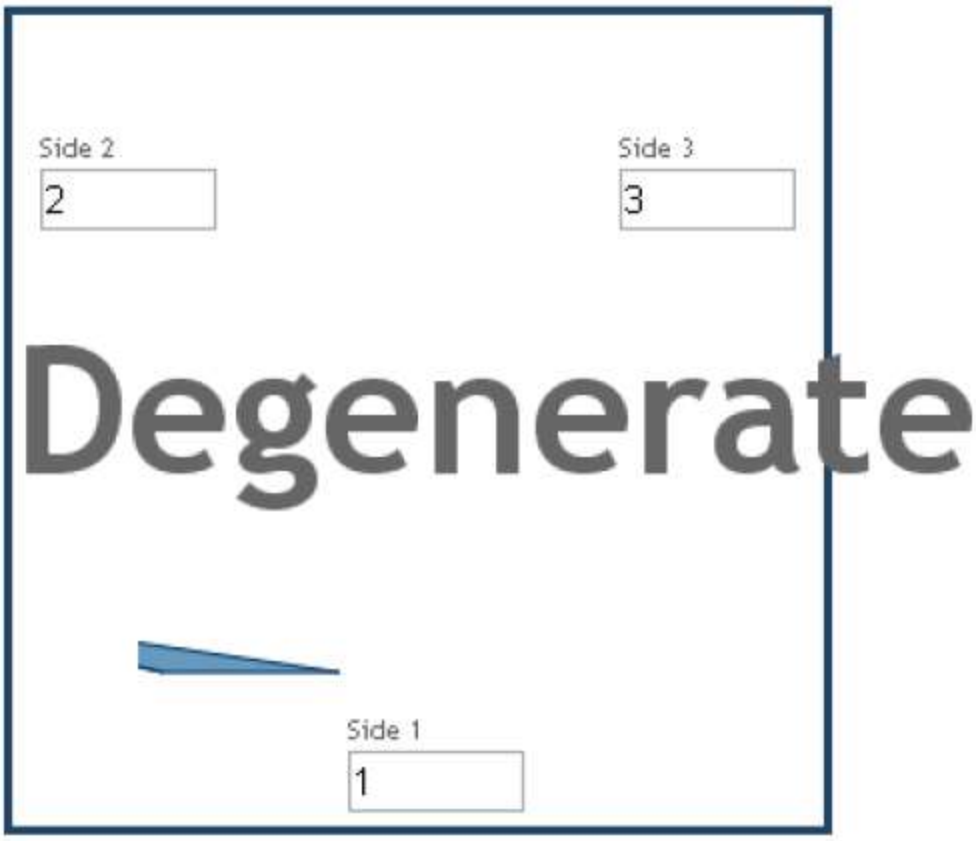
• Για τα δεδομένα εισόδου **(1, 2, 4)** ο τύπος τριγώνου είναι: **Invalid.**

• Για τα δεδομένα εισόδου **(1, 2, 3)** ο τύπος τριγώνου είναι: **Degenerate.**

Ως συνέπεια της υπόθεσης που πήραμε για τον ορισμό του τριγώνου, οι πλευρές του τρίγωνου πρέπει να περικλείουν μια περιοχή για να ορίζεται αυτό. Στην πρώτη περίπτωση που δοκιμάσαμε οι τρεις τιμές είναι έγκυρες αλλά δεν σχηματίζεται τρίγωνο γιατί δεν κλείνουν ποτέ, ενώ στη δεύτερη περίπτωση οι τιμές εισόδου είναι στην ίδια ευθεία.



Σχήμα 4.1: Γραφική αναπαράσταση τριγώνου όπου το άθροισμα των δύο μικρότερων πλευρών είναι μικρότερο από την μεγαλύτερη.



Σχήμα 4.2: Γραφική αναπαράσταση τριγώνου όπου το άθροισμα των δύο μικρότερων πλευρών είναι ίσο από την μεγαλύτερη.

Συμβουλευόμαστε πάλι την Ευκλείδεια Γεωμετρία και βλέπουμε ότι αν α, β, γ τρεις θετικοί αριθμοί, η ανίσωση **|β-γ| < α < β + γ** είναι ικανή και αναγκαία συνθήκη για να αποτελούν πλευρές τριγώνου. Επομένως, αν δεν ισχύει η τριγωνική ανισότητα τότε δεν σχηματίζεται τρίγωνο.

Από την ανάλυση των απαιτήσεων της εφαρμογής και τον διερευνητικό έλεγχο αποκτήσαμε μια πιο ολοκληρωμένη εικόνα για την εφαρμογή, τι είναι αυτό που θα ελέγξουμε και συλλέξαμε πληροφορίες που θα μας φανούν χρήσιμες για τον σχεδιασμό των περιπτώσεων ελέγχου. Συνοψίζοντας, καταλήξαμε στις παρακάτω υποθέσεις και παρατηρήσεις:

• Για να σχηματίζουν οι τρεις τιμές εισόδου τρίγωνο πρέπει το άθροισμα των τριών μικρότερων πλευρών να είναι μεγαλύτερο από την τρίτη πλευρά.

• Έχουμε τέσσερις τύπους τριγώνων: ισόπλευρο, ισοσκελές, σκαληνό και ορθογώνιο.

• Υπάρχει τουλάχιστον μια περίπτωση ελέγχου που «πέρασε» για όλους τους βασικούς τύπους τριγώνου.

• Για να είναι ένα τρίγωνο ορθογώνιο πρέπει να ικανοποιείται το Πυθαγόρειο θεώρημα.

• Οι αρνητικοί αριθμοί δεν είναι αποδεκτοί ως πλευρές του τριγώνου.

• Το σύστημα δέχεται τιμές με εύρος από το 1 μέχρι το 999.

• Το σύστημα δέχεται αριθμούς με δεκαδικό μέρος.

• Η εφαρμογή δεν επικυρώνει αν τα δεδομένα εισόδου είναι αριθμοί ή όχι.

Ο διερευνητικός έλεγχος βασίζεται περισσότερο στην εμπειρία και στις ικανότητες του tester και προσφέρει άμεση πληροφορία για το αντικείμενο υπό έλεγχο, αλλά δεν αρκεί για ένα ολοκληρωμένο έλεγχο λογισμικού. Για το λόγο αυτό θα ακολουθήσουμε μια πιο φορμαλιστική μέθοδο ελέγχου από αυτές που αναλύσαμε στο κεφάλαιο 2.

Ο έλεγχος της εφαρμογής των τριγώνων είναι κυρίως έλεγχος των δεδομένων εισόδου, οπότε κατάλληλη black box μέθοδος είναι αυτή του διαχωρισμού σε κλάσεις ισοδυναμίας σε συνδυασμό με την ανάλυση οριακών τιμών. Αρχικά θα διαχωρίσουμε τα δεδομένα εισόδου σε έγκυρα και μη έγκυρα, δηλαδή θα διαχωρίσουμε τα δεδομένα σε αυτά που παράγουν τρίγωνα και σε δεδομένα που δεν παράγουν, στη συνέχεια η κλάση των έγκυρων θα διαχωριστεί για κάθε τύπο τριγώνου και η κλάση των μη έγκυρων δεδομένων σε αριθμητικά και μη αριθμητικά δεδομένα.

Στη συνέχεια θα κάνουμε μια ανάλυση οριακών τιμών για να προσδιορίσουμε τις ακρότατες τιμές των δεδομένων εισόδου. Επομένως, σύμφωνα με τη μέθοδο διαχωρισμού σε κλάσεις ισοδυναμίας τα δεδομένα εισόδου s1, s2, s3 που αντιστοιχούν στις πλευρές Side 1, Side 2 και Side 3 της εφαρμογής υπό έλεγχο διαχωρίζονται σε:

- **΄Εγκυρα δεδομένα εισόδου**

• Τα δεδομένα εισόδου s1, s2, s3 σχηματίζουν ισόπλευρο τρίγωνο.

• Τα δεδομένα εισόδου s1, s2, s3 σχηματίζουν ισοσκελές τρίγωνο.

  ο με s1 = s2

  ο με s2 = s3

  ο με s1 = s3

• Τα δεδομένα εισόδου s1, s2, s3 σχηματίζουν σκαληνό τρίγωνο.

  ο Οξυγώνιο

  ο Αμβλυγώνιο

• Τα δεδομένα εισόδου s1, s2, s3 σχηματίζουν ορθογώνιο τρίγωνο.

  ο με s1 υποτείνουσα

  ο με s2 υποτείνουσα

  ο με s3 υποτείνουσα

- **Μη έγκυρα δεδομένα εισόδου**

• Το δεδομένο εισόδου s1 είναι **μη αριθμητικό** και είναι:

  ο Γράμμα αλφαβήτα

    Κεφαλαίο γράμμα (π.χ. Α).

    Μικρό γράμμα (π.χ. α).

  ο Αλφαριθμητικό (π.χ. 9Α9).

ο Επιστημονικός συμβολισμός (π.χ. 1.3E6).

ο Ειδικού χαρακτήρα (π.χ. ? , $).

ο Χαρακτήρας με ειδική σημασία για το περιβάλλον του συστήματος (π.χ. Enter, Space, Escape, Backspace κ.ά.).

• Το δεδομένο εισόδου είναι s1 **αριθμητικό** και είναι: ο Το μηδέν.

ο Αρνητικός αριθμός (π.χ. -1).

ο Ακρότατη τιμές (1 και 999).

Για τις ακρότατες τιμές των δεδομένων εισόδου θα εκτελέσουμε ανάλυση οριακών τιμών. Σκοπός μας είναι να βεβαιωθούμε ότι η εφαρμογή διαχειρίζεται σωστά τα πολύ μεγάλα τρίγωνα λόγω έλλειψης περιορισμού άνω ορίου στα δεδομένα εισόδου. Συνεπώς δεν μας αφορά η κάτω οριακή τιμή αλλά και ούτε και οι γειτονικές τιμές των κλάσεων, αφού το μηδέν ξέρουμε ότι δεν είναι επιτρεπτό και το άνω όριο τέθηκε από εμάς στις υποθέσεις. Συνεπώς πρέπει να ελέγξουμε τις παρακάτω οριακές τιμές:

• s1 = s2 = s3 = 999 για το μεγαλύτερο ισόπλευρο τρίγωνο.

• s1 = s2 = 999 και s3 = 998 για να εξετάσουμε πώς διαχειρίζεται το σύστημα τις μεγάλες τιμές και αν διακρίνει ότι το τρίγωνο είναι ισοσκελές.

• s1 = s2 = 999 και s3 = 99 για να εξετάσουμε πώς διαχειρίζεται το σύστημα τιμές διαφορετικής τάξης και αν διακρίνει ότι το τρίγωνο είναι ισοσκελές.

• s1 = s2 = 999 και s3 = 1 για να εξετάσουμε πώς διαχειρίζεται το σύστημα το συνδυασμό των ακραίων τιμών και αν διακρίνει ότι το τρίγωνο είναι ισοσκελές.

Στις ερωτήσεις πόσες μη έγκυρες περιπτώσεις αρκούν για έναν έλεγχο, αν υπάρχουν μη έγκυρες περιπτώσεις ελέγχου που δεν ελέγχθηκαν παραπάνω ή αν οι περιπτώσεις ελέγχου που θέσαμε είναι πάρα πολλές (*overkill*), η απάντηση εξαρτάται από δύο παράγοντες: το ρίσκο που φέρει το αντικείμενο υπό έλεγχο, αν η λειτουργία του τριγώνου επηρεάζει σημαντικά και άλλα κομμάτια του συστήματος τότε το ρίσκο είναι υψηλό, και την πιθανότητα ο τελικός χρήστης να βάλει μη έγκυρες τιμές εισόδου είναι μεγάλη. Για παράδειγμα, η εφαρμογή των τριγώνων που εξετάζουμε δεν περιορίζει τον χρήστη στο άνω όριο των τιμών εισόδου, όμως η πιθανότητα ένας χρήστης να βάλει για πλευρές πολύ μεγάλες τιμές, όπως 999.999.999, είναι ιδιαίτερα απίθανη, οπότε δε χρειάζεται να ελεγχθεί για τόσο ακραίες τιμές και για αυτό επιλέχθηκε ως άνω όριο το 999 που αρκεί για τον έλεγχο των μεγάλων τριγώνων.

Η απάντηση συνεπώς είναι ότι ο κατάλληλος συνδυασμός έγκυρων και μη έγκυρων περιπτώσεων ελέγχου εξαρτάται από της ανάγκες του αντικειμένου υπό εξέταση. Εμείς θέλουμε να πετύχουμε υψηλή αξιοπιστία και σταθερότητα στο σύστημα μας οπότε θα ελέγξουμε αρκετές μη έγκυρες περιπτώσεις.

Ένας άλλος πολύ σημαντικός παράγοντας για την επιλογή των κατάλληλων μη έγκυρων περιπτώσεων ελέγχου είναι «πόσο» έλεγχο πρέπει να κάνουμε για τις τιμές εισόδου s2 και s3. Αν είχαμε στη διάθεση μας τον κώδικα της εφαρμογής, δηλαδή αν εκτελούσαμε white box έλεγχο, τότε θα γνωρίζαμε εάν ο προγραμματιστής ακολούθησε την ίδια λογική υλοποίησης για κάθε μία από τις τιμές εισόδου. Σε αυτή την περίπτωση δε θα ήταν αναγκαίο να επαναλάβουμε όλες τις περιπτώσεις ελέγχου και για τις τρεις πλευρές του τριγώνου. Αυτό δε σημαίνει άμεσα ότι εμείς που εκτελούμε black box έλεγχο πρέπει να το κάνουμε, αρκεί μια περίπτωση για την τιμή εισόδου s2 και μια για την s3 για να ελέγξουμε αν ο προγραμματιστής έπραξε αναλόγως για όλες τις εισερχόμενες τιμές.

Μια ακόμα καλύτερη προσέγγιση είναι αντί να διεξάγουμε όλες τις μη έγκυρες περιπτώσεις ελέγχου στην s1 ακολουθούμενες από μία περίπτωση για την s2 και μια για την s3, να διαμοιράσουμε τις μη έγκυρες περιπτώσεις ισόποσα στις τρεις πλευρές. Αναλόγως, για τις έγκυρες περιπτώσεις ελέγχου, αν υποθέσουμε ισοδυναμία στον τρόπο υλοποίησης των τριών τιμών εισόδου s1, s2, s3 τότε η σειρά των

δεδομένων εισόδου δεν μας απασχολεί. Για παράδειγμα, η τριάδα (3, 3, 2) που παράγει ισοσκελές τρίγωνο, το ίδιο συμβαίνει και με τις τριάδες (3, 2, 3) και (2, 3, 3). Η μόνη διαφορά στα παραπάνω τρίγωνα είναι η γραφική απεικόνιση τους. Στην περίπτωση της εφαρμογής μας, με δοκιμές παρατηρούμε ισοδυναμία όποτε δε χρειάζεται να επαναλάβουμε τις μη έγκυρες περιπτώσεις ελέγχου για όλες τις πλευρές. Όμως τις έγκυρες περιπτώσεις ελέγχου θα τις εκτελέσουμε επί τρία, διότι βασικό στοιχείο των έγκυρων δοκιμασιών της εφαρμογής είναι ο έλεγχος των γραφικών αναπαραστάσεων των τριγώνων.

Από την ανάλυση των απαιτήσεων και με την χρήση black box μεθόδων καταλήγουμε στο παρακάτω σύνολο 28 περιπτώσεων ελέγχου που αποτελεί το σενάριο ελέγχου (test suite) της εφαρμογής των τριγώνων:

Περίπτωση ελέγχου #	**Κατάσταση υπό έλεγχο**	**Δεδομένα εισόδου**			**Αναμενόμενο αποτέλεσμα**
		s1	s2	s3	
1	Ισόπλευρο τρίγωνο	5	5	5	Equilateral
2	Ισοσκελές τρίγωνο με s1 = s2	10	10	15	Isosceles
3	Ισοσκελές τρίγωνο με s2 = s3	10	15	10	Isosceles
4	Ισοσκελές τρίγωνο με s1 = s2	15	10	10	Isosceles
5	Οξυγώνιο	4	5	6	Scalene
6	Αμβλυγώνιο	2	5	6	Scalene
7	Ορθογώνιο με s1 υποτείνουσα	5	3	4	Right
8	Ορθογώνιο με s2 υποτείνουσα	3	5	4	Right
9	Ορθογώνιο με s3 υποτείνουσα	3	4	5	Right

10	Πλευρές με ένα δεκαδικό ψηφίο	1.1	1.9	1.5	Isosceles
11	Πλευρές με έξι δεκαδικά ψηφία	3.330050	3.329951	3.330050	Equilateral
12	Όλες οι πλευρές με τη μέγιστη ακρότατη τιμή	999	999	999	Equilateral
13	Δύο πλευρές με τη μέγιστη ακρότατη τιμή και η τρίτη με μια γειτονική	999	999	998	Isosceles
14	Δύο πλευρές με τη μέγιστη ακρότατη τιμή και η τρίτη άλλης τάξης	999	99	999	Isosceles
15	Δύο πλευρές με τη μέγιστη ακρότατη τιμή και η τρίτη με την ελάχιστη	1	999	999	Isosceles
16	Μια πλευρά με κεφαλαίο γράμμα	A	2	3	Invalid
17	Μια πλευρά με μικρό γράμμα	1	b	1	Invalid
18	Μια πλευρά με αλφαριθμητικό	3	3	3c	Invalid
19	Μια πλευρά με επιστημονικό συμβολισμό	1.2E6	1	1	Invalid
20	Μια πλευρά με ειδικό χαρακτήρα	3	@	6	Invalid
21	Η πλευρά s1 με 0	0	2	3	Invalid
22	Οι πλευρές s1 και s2 με 0	0	0	5	Invalid

23	Όλες οι πλευρές με 0	0	0	0	Invalid
24	Η πλευρά s2 με αρνητικό αριθμό	2	-2	2	Invalid
25	Οι πλευρές s2 και s3 με αρνητικό αριθμό	3	-1	-1	Invalid
26	Όλες οι πλευρές με αρνητικό αριθμό	-5	-5	-5	Invalid
27	Οι κορυφές του τριγώνου είναι στην ίδια ευθεία	1	2	3	Degenerate
28	Οι πλευρές δεν σχηματίζουν ποτέ τρίγωνο	1	2	4	Invalid

4.4.3 Αυτοματοποιημένος έλεγχος της εφαρμογής των τριγώνων

4.4.3.1 Επιλογή εργαλείου αυτοματοποίησης

Το σενάριο ελέγχου που καταλήξαμε μπορεί να εκτελεστεί μη αυτοματοποιημένα, δηλαδή απλά πηγαίνοντας στην εφαρμογή και συμπληρώνοντας όλες τις τιμές εισόδου που επιλέχθηκαν για την κάθε περίπτωση. Τότε, για το κάθε τέλος περίπτωσης ελέγχου το αποτέλεσμα του ελέγχου πρέπει να σημειώνεται και να συγκρίνεται με το αναμενόμενο αποτέλεσμα. Επίσης σε περίπτωση αναβάθμισης της εφαρμογής το σενάριο πρέπει να εκτελείται συστηματικά ως έλεγχος παλινδρόμησης. Παρόλο που η εφαρμογή υπό έλεγχο είναι μικρή και ο έλεγχος είναι πιο γρήγορος αν γίνει «χειρωνακτικά», επιλέγεται η αυτοματοποίηση για την δυνατότητα επανάληψης του ελέγχου.

Για τη δημιουργία και την εκτέλεση του αυτοματοποιημένου σεναρίου ελέγχου πρέπει να επιλεχθεί ένα εργαλείο κατάλληλο για την εφαρμογή. Στόχος του ελέγχου είναι η σωστή λειτουργία της εφαρμογής διαδικτύου, συνεπώς στόχος είναι η εύρεση ενός εργαλείου μεσαίας βαθμίδας για δοκιμασίες αποδοχής. Επίσης το αντικείμενο του ελέγχου είναι η δοκιμή των δεδομένων ελέγχου, οπότε ένα εργαλείο με βάση την τεχνική βάσει δεδομένων θα ήταν ίσως το καταλληλότερο. Εν τούτοις, και ένα εργαλείο τεχνικής βάσει λέξεων-κλειδιών, η οποία επιτρέπει τη δόμηση περιπτώσεων ελέγχου σε μορφή λέξεων-κλειδιών, μπορεί να καλύψει τις ανάγκες της εφαρμογής υπό έλεγχο. Επίσης, το εργαλείο πρέπει να δημιουργεί αυτόματα αναφορές αποτελεσμάτων των περιπτώσεων ελέγχου και να είναι γενικά εύκολο στη χρήση, έτσι ώστε να υπάρξει μεγαλύτερο κέρδος από την αυτοματοποίηση. Επίσης αναζητείται ανοιχτό ή ελεύθερο λογισμικό ώστε να επιτρέπεται η χρήση του στα πλαίσια της διπλωματικής εργασίας. Από τα εργαλεία που παρουσιάσαμε στο υπόκεφάλαιο 3.3, επιλέχθηκε να χρησιμοποιηθεί το Robot Framework της τεχνικής βάσει λέξεων-κλειδιών, το οποίο παρέχει επίσης και λειτουργίες της τεχνικής βάσει δεδομένων.

4.4.3.2 Εκτέλεση αυτοματοποιημένου ελέγχου

Το Robot Framework είναι υλοποιημένο σε Python, αλλά μπορεί να τρέξει και με Jython ή IronPython, που είναι συμβεβλημένες με πλατφόρμες της Java και .NET αντίστοιχα. Για την εγκατάσταση του εργαλείου επιλέχθηκε η εγκατάσταση της Python, και στη συνέχεια εγκαταστάθηκε το Robot Framework. Επειδή το εργαλείο που εξελέγει είναι για δοκιμασίες αποδοχής, χρειάζεται ένα επιπλέον εργαλείο ή μια βιβλιοθήκη του εργαλείου για την πλοήγηση του σεναρίου ελέγχου στην διεπαφή της εφαρμογής.

Το Robot είναι πλούσιο σε βιβλιοθήκες που παρέχουν έτοιμες λέξεις-κλειδιά, κάποιες από αυτές είναι άμεσα διαθέσιμες (standard libraries), κάποιες είναι *εξωτερικές* και πρέπει να εγκατασταθούν επιπλέον (*external libraries*) και άλλες μπορούν να φτιαχτούν από τον χρήστη (custom libraries). Η βιβλιοθήκη Selenium2Library είναι εξωτερική βιβλιοθήκη και παρέχει την πλοήγηση του γραφικού περιβάλλοντος που χρειαζόμαστε.

Το εργαλείο παρέχει επιπλέον ένα ειδικό περιβάλλον για τη συγγραφή των σεναρίων ελέγχου, το RIDE (Robot integrated Development Environment), το οποίο πρέπει να εγκατασταθεί επιπλέον. Το πλαίσιο δέχεται όμως και άλλου είδους αρχεία με τις περιπτώσεις ελέγχου, όπως .txt, .html. και .tsv, τα οποία γράφονται ξεχωριστά και εκτελούνται άμεσα. Για να εκτελεστεί ένα από αυτά του είδους αρχεία στο Robot αρκεί η κατάλληλη εντολή στη γραμμή εντολών του υπολογιστή. Στην παρούσα εργασία επιλέχτηκε να γραφτούν τα σενάρια ελέγχου σε μορφή απλού κειμένου.

Τα σενάρια ελέγχου στο Robot Framework γράφονται με λέξεις-κλειδιά τα οποία ορίζονται αναλόγως μέσα στο σενάριο. Εκτός από τις λέξεις-κλειδιά του χρήστη (user keywords), μέσα στο σενάριο πρέπει να οριστούν και οι λέξεις-κλειδιά της βιβλιοθήκης. Οι πρώτες ορίζονται κάτω από τον τίτλο *** Settings ***, ενώ οι δεύτερες κάτω από το *** Keywords ***. Οι λέξεις-κλειδιά της βιβλιοθήκης στην ουσία καλούν τις βιβλιοθήκες, είτε αυτές είναι του χρήστη είτε εξωτερικές, π.χ. Library Selenium2Library. Επιπλέον εκεί προσδιορίζονται λέξεις-κλειδιά που εκτελούνται πριν ή μετά από κάθε περίπτωση ελέγχου ή στην έναρξη και το τέλος ολόκληρου του σεναρίου π.χ. με τη χρήση των εντολών Suite Setup και Suite Teardown.

Για το σενάριο ελέγχου των τριγώνων χρειάστηκε να οριστούν πέντε λέξεις-κλειδιά με τη βοήθεια μεταβλητών:

• **Enter**: καλεί εντολές της βιβλιοθήκης Selenium2Library για το άνοιγμα του πλοηγητή διαδικτύου στη διεύθυνση ιστοσελίδας που είναι η εφαρμογή υπό εξέταση.

• **Exit**: κλείνει τον πλοηγητή διαδικτύου.

• **Input values**: αναθέτει τα δεδομένα εισόδου στις πλευρές του τριγώνου μέσω των μεταβλητών ${side1}, ${side2}, ${side3}.

• **Verify triangle is identified as:** ορίζει τη μεταβλητή ${actual} ως το λεκτικό που επιστρέφει η εφαρμογή στην οθόνη για τύπο τριγώνου και τη μεταβλητή ${expected} ως το αναμενόμενο τύπο τριγώνου που θα σχηματίσουν τα δεδομένα εισόδου. Οι δύο μεταβλητές πρέπει να είναι ίσες μεταξύ τους.

• **Verify triangle is drawn inside canvas:** ελέγχει αν το σχήμα του τριγώνου είναι μέσα στο πλαίσιο. Ορίζει τις μεταβλητές ${coord_as_string}, η οποία παίρνει σαν εισόδους τις συντεταγμένες των κορυφών του τριγώνου, και ${in_range}, η οποία καλεί την μέθοδο CoordinateCheck10 και επιτστρέφει True ή False, αν το σχήμα είναι μέσα στο πλαίσιο ή όχι αντίστοιχα.

Αφού οριστούν οι λέξεις-κλειδιά, υπάρχουν όλα τα απαραίτητα εργαλεία για να γραφτούν οι περιπτώσεις ελέγχου, οι οποίες ορίζονται συνήθως στην αρχή του σεναρίου κάτω από τον τίτλο *** Test Cases ***. Οι περιπτώσεις ελέγχου χωρίζονται μεταξύ τους με μια κενή σειρά. Σε κάθε περίπτωση ελέγχου, η πρώτη σειρά της αποτελεί τον τίτλο της περίπτωσης και οι υπόλοιπες καλούν από μια λέξη-κλειδί παρέχοντας τις τιμές των μεταβλητών που έχουν οριστεί. Για παράδειγμα, η πρώτη περίπτωση ελέγχου από το σενάριο ελέγχου που επιλέχθηκε παραπάνω και ελέγχει ένα απλό ισόπλευρο τρίγωνο, έχει την μορφή:

Handles a simple Equilateral

Input values 5, 5, 5

Verify triangle is identified as "Equilateral"

Αφού γραφτούν όλες οι περιπτώσεις ελέγχου σε αποδεκτή μορφή σεναρίου του Robot, το σενάριο μπορεί να αποθηκευτεί και είναι έτοιμο για εκτέλεση. Η εκτέλεση του είναι ιδιαίτερα απλή, ανοίγοντας τη γραμμή εντολών του υπολογιστή αρκεί να εισάγουμε την εντολή:

pybot test_triangles.txt

Με την εντολή αυτή εκτελείται αυτόματα το σενάριο. Κατά την λήξη του σεναρίου, το εργαλείο παράγει άμεσα τρία αρχεία: την αναφορά των αποτελεσμάτων του ελέγχου (report.html), το αρχείο με όλες τις καταστάσεις που πέρασε το σενάριο (logs.html) και όλα τα δεδομένα εξόδου σε μορφή xml αρχείου (output.xml).

4.4.3.3 Αποτελέσματα αυτοματοποιημένου ελέγχου

Τα εξαγόμενα αποτελέσματα του αυτοματοποιημένου ελέγχου από το Robot Framework αναφέρουν ότι οι 22 από τις 28 περιπτώσεις «πέρασαν τον έλεγχο», ενώ οι υπόλοιπες απέτυχαν.

Οι 6 περιπτώσεις που απέτυχαν πρέπει να εξεταστούν περισσότερο εις βάθος για να αποσαφηνιστεί αν η αποτυχία οφείλεται σε ελάττωμα της εφαρμογής ή σε λάθος του σεναρίου ελέγχου, του περιβάλλοντος ελέγχου ή της συνδεσιμότητας μεταξύ των δύο. Αναλυτικά οι περιπτώσεις που δεν πέρασαν τον έλεγχο σύμφωνα με τον τίτλο της περίπτωσης ελέγχου στο σενάριο είναι:

• Handles an Isosceles when all sides have one decimal digit

Από το ελάττωμα αυτό παρατηρούμε ότι το σύστημα δεν επεξεργάζεται σωστά την στρογγυλοποίηση. Θεωρεί ότι οι αριθμοί 1.1 και 1.9 στρογγυλοποιούνται και οι δύο στο 1, οπότε το τρίγωνο είναι ισόπλευρο.

• Handles Invalid with an upper-case letter side

• Handles Invalid with a lower-case letter side

• Handles Invalid with a alphanumerical side

• Handles Invalid with a special character side

Και οι τέσσερις περιπτώσεις ελέγχου ανήκουν στην ίδια κλάση, αφού τα τέσσερα ελαττώματα που ανιχνεύτηκαν έχουν κοινή ρίζα. Τα αποτελέσματα του ελέγχου υποδηλώνουν ότι το σύστημα δεν κάνει επικύρωση των δεδομένων εισόδων με αποτέλεσμα να δέχεται τιμές γράμματα και ειδικούς χαρακτήρες.

• Handles Invalid with all sides zero

Η περίπτωση όλες οι πλευρές να είναι μηδέν δεν «περνάει» τον έλεγχο γιατί επιστρέφει “Equilateral” αντί για “Invalid”. Το σύστημα δεν επικυρώνει την περίπτωση του μηδενός, επομένως οι περιπτώσεις ελέγχου 21 και 22 «πέρασαν» τον έλεγχο, δηλαδή επέστρεψαν “Invalid”, λόγω της μη ικανοποίησης της τριγωνικής ανισότητας και όχι λόγω του μηδενός.

Συνολικά, παρατηρούμε ότι το αυτοματοποιημένο σενάριο ελέγχου βρήκε τρεις βασικές κλάσεις ελαττωμάτων, αυτές της στρογγυλοποίησης, υπερχείλισης και απουσίας επικύρωσης δεδομένων εισόδου. Επίσης, παρατηρούμε ότι το σύστημα δεν επιστρέφει κανένα μήνυμα λάθους στο χρήστη ώστε να τον ενημερώνει, συμπεραίνοντας ότι ο παράγοντας χρησικανότητας δεν είναι ικανοποιητικός και με ανάλογες προδιαγραφές αυτό θα μπορούσε να υποδηλώνει άλλο ένα ελάττωμα.

4.5 Selenium

4.5.1 Αυτοματοποιημένος έλεγχος για Διαδικτυακές Εφαρμογές

Πολλές, ίσως οι περισσότερες, εφαρμογές λογισμικού σήμερα είναι γραμμένες ως web-based εφαρμογές και τρέχουν σε ένα πρόγραμμα περιήγησης στο Internet. Η αποτελεσματικότητα του ελέγχου αυτών των εφαρμογών ποικίλλει ευρέως μεταξύ των εταιρειών και οργανισμών. Σε μια εποχή άκρως διαδραστική όπου πολλοί οργανισμοί χρησιμοποιούν κάποια μορφή της μεθοδολογίας Agile, ο αυτοματοποιημένος έλεγχος συχνά γίνεται απαραίτητος για έργα λογισμικού. Ο αυτοματοποιημένος έλεγχος είναι συχνά η απάντηση. Αυτοματοποιημένος έλεγχος σημαίνει χρησιμοποιώντας ένα εργαλείο λογισμικού για να τρέξει επαναλαμβανόμενους ελέγχους για την εφαρμογή που πρόκειται να ελεγχθεί.

Υπάρχουν πολλά πλεονεκτήματα στον αυτοματοποιημένο έλεγχο. Τα περισσότερα σχετίζονται με την επαναληψιμότητα των ελέγχων και τη ταχύτητα με την οποία μπορούν να εκτελεστούν οι έλεγχοι. Υπάρχουν μια σειρά από εμπορικά και ανοικτού κώδικα εργαλεία που διατίθενται για την παροχή βοήθειας με την ανάπτυξη του αυτοματοποιημένου έλεγχου. Το Selenium είναι ίσως η πιο ευρεία χρησιμοποιημένη λύση ανοικτού κώδικα.

Ο αυτοματοποιημένος έλεγχος έχει συγκεκριμένα πλεονεκτήματα για τη βελτίωση της μακροπρόθεσμης αποτελεσματικότητας των διαδικασιών μιας ομάδας ελέγχου λογισμικού.

Ο αυτοματοποιημένος έλεγχος υποστηρίζει:

- Συχνούς ελέγχους παλινδρόμησης
- Ταχεία ανατροφοδότηση για τους προγραμματιστές
- Σχεδόν απεριόριστες επαναλήψεις της εκτέλεσης των περιπτώσεων ελέγχου
- Υποστήριξη για Agile και ακραίες μεθοδολογίες ανάπτυξης
- Πειθαρχημένη τεκμηρίωση των περιπτώσεων ελέγχου
- Προσαρμοσμένη αναφορά ελαττωμάτων
- Η εύρεση ελαττωμάτων χάνει με τον χειρωνακτικό έλεγχο

4.5.2 Αυτοματοποίηση ή μη Αυτοματοποίηση;

Είναι η αυτοματοποίηση πάντα πλεονέκτημα; Πότε πρέπει κάποιος να αποφασίσει να αυτοματοποιήσει τις περιπτώσεις ελέγχου;

Δεν είναι πάντα συμφέρουσα η αυτοματοποίηση των περιπτώσεων ελέγχου. Υπάρχουν στιγμές που ο χειρωνακτικός έλεγχος μπορεί να είναι πιο κατάλληλος. Για παράδειγμα, εάν το User Interface της εφαρμογής θα αλλάξει σύντομα, τότε η αυτοματοποίηση μπορεί να χρειαστεί να ξαναγραφτεί ούτως ή άλλως. Επίσης, μερικές φορές απλά δεν υπάρχει αρκετός χρόνος για την οικοδόμηση αυτοματοποιημένου ελέγχου. Για σύντομο χρονικό διάστημα, ο χειρωνακτικός έλεγχος μπορεί να είναι πιο

αποτελεσματικός. Εάν η εφαρμογή έχει μια πολύ αυστηρή προθεσμία και δεν υπάρχει επί του παρόντος διαθέσιμος αυτοματοποιημένος έλεγχος τότε ο χειρωνακτικός έλεγχος είναι η καλύτερη λύση.

4.5.3 Παρουσιάζοντας το Selenium

Το Selenium είναι ένα σύνολο διαφορετικών εργαλείων λογισμικού καθένα με μια διαφορετική προσέγγιση για την υποστήριξη αυτοματοποιημένου ελέγχου. Οι περισσότεροι μηχανικοί QA Selenium επικεντρώνονται σε μία ή δύο εργαλεία τα οποία ικανοποιούν τις ανάγκες του έργου τους, ωστόσο, μαθαίνοντας όλα τα εργαλεία θα σας δοθούν επιλογές για την διαφορετική προσέγγιση των προβλημάτων αυτοματοποιημένου ελέγχου. Ολόκληρη η σουίτα εργαλείων οδηγεί σε ένα πλούσιο σύνολο λειτουργιών ελέγχου ειδικά για τις ανάγκες των έλεγχων των web εφαρμογών όλων των τύπων.

Οι λειτουργίες αυτές είναι ιδιαίτερα ευέλικτες, επιτρέποντας πολλές επιλογές για τον εντοπισμό στοιχείων UI και συγκρίνοντας τα αναμενόμενα αποτελέσματα των έλεγχων με βάση την πραγματική συμπεριφορά της εφαρμογής. Ένα από τα βασικά χαρακτηριστικά του Selenium είναι η υποστήριξη για τη εκτέλεση ελέγχων σε πολλαπλές πλατφόρμες προγράμματος περιήγησης.

4.5.4 Σύντομη Ιστορία του Selenium

Το Selenium ήρθε για πρώτη φορά στη ζωή το 2004, όταν ο Jason Huggins εξέταζε μια εσωτερική εφαρμογή στο ThoughtWorks. Όντας ένας έξυπνος άνθρωπος, κατάλαβε ότι υπήρχαν καλύτερες λύσεις στην εποχή του από το να περνάει από κάθε έλεγχο με κάθε αλλαγή που έκανε. Ανέπτυξε μια βιβλιοθήκη Javascript που θα μπορούσε να οδηγήσει σε αλληλεπιδράσεις με την ιστοσελίδα του, επιτρέποντάς του να ξαναεκτελέσει αυτόματα ελέγχους ενάντια σε πολλούς browsers. Η βιβλιοθήκη έγινε τελικά η Selenium Core, η οποία οφείλει το σύνολο της λειτουργικότητας της στο Selenium Remote Control (RC) και το Selenium IDE.

Ενώ το Selenium ήταν ένα τεράστιο εργαλείο, είχε και μειονεκτήματα. Ήταν αδύνατον να γίνουν διαφορετικά πράγματα, λόγω της Javascript based μηχανής αυτοματισμού και των περιορισμών ασφαλείας των browsers που εφαρμόζονται σε Javascript. Τα πράγματα έγιναν χειρότερα όταν τα webapps άρχισαν να γίνονται όλο και πιο ισχυρά με την πάροδο του χρόνου, χρησιμοποιώντας όλα τα είδη των ειδικών χαρακτηριστικών των νέων προγραμμάτων περιήγησης κάνοντας τους περιορισμούς όλο και πιο επώδυνους.

Το 2006 ένας θαρραλέος μηχανικός της Google που ονομάζεται Simon Stewart άρχισε να εργάζεται σε ένα πρόγραμμα που ονομάζεται WebDriver. Η Google ήταν από καιρό ένας φανατικός χρήστης του Selenium, αλλά οι testers έπρεπε να εργαστούν γύρω από τους περιορισμούς του προϊόντος. Ο Simon ήθελε ένα εργαλείο ελέγχου το οποίο να μιλάει απευθείας στον browser, χρησιμοποιώντας τη μέθοδο « native» για το browser και το λειτουργικό, αποφεύγοντας έτσι τους περιορισμούς ενός περιβάλλοντος sandboxed Javascript. Το έργο WebDriver ξεκίνησε με σκοπό να λύσει τα αδύνατα σημεία του Selenium.

Μετάβαση στο 2008. Η πιο σημαντική ιστορία εκείνου του έτους ήταν η συγχώνευση του Selenium με τον WebDriver. Το Selenium είχε τεράστια κοινότητα και εμπορική υποστήριξη, αλλά το WebDriver ήταν σαφώς το εργαλείο του μέλλοντος. Η ένωση των δύο εργαλείων παρέχει ένα κοινό σύνολο χαρακτηριστικών γνωρισμάτων για όλους τους χρήστες και έφερε μερικά από τα λαμπρότερα μυαλά στον τομέα του αυτοματοποιημένου ελέγχου κάτω από μία στέγη. Ίσως η καλύτερη εξήγηση για το γιατί το WebDriver και το Selenium συγχωνεύονται εκτέθηκε λεπτομερώς από τον Simon Stewart, ο δημιουργός του WebDriver, σε ένα κοινό e-mail στη κοινότητα του WebDriver και του Selenium στις 6 Αυγούστου, 2009.

"Γιατί συγχωνεύονται τα προγράμματα; Εν μέρει επειδή το WebDriver αντιμετωπίζει ορισμένες ελλείψεις στο Selenium, εν μέρει επειδή το Selenium αντιμετωπίζει ορισμένες ελλείψεις στο WebDriver (όπως για παράδειγμα, υποστηρίζει ένα ευρύτερο φάσμα των browsers) και εν μέρει επειδή οι βασικοί συντελεστές του Selenium και εγώ νιώσαμε ότι ήταν ο καλύτερος τρόπος για να προσφέρουμε στους χρήστες το καλύτερο δυνατό framework. "

4.5.5 Selenium IDE

Το Selenium-IDE (Integrated Development Environment) είναι το εργαλείο που χρησιμοποιείτε για την ανάπτυξη των περιπτώσεων ελέγχου. Είναι ένα Firefox plug-in, εύκολο στη χρήση του και γενικά είναι ο πιο αποτελεσματικός τρόπος για την ανάπτυξη περιπτώσεων ελέγχου. Περιέχει επίσης ένα μενού που σας επιτρέπει να επιλέξετε πρώτα ένα στοιχείο UI από την τρέχουσα εμφανιζόμενη σελίδα του browser και στη συνέχεια επιλέγεται από τη λίστα του Selenium εντολές με παραμέτρους προκαθορισμένες ανάλογα με το πλαίσιο του επιλεγμένου στοιχείου UI. Αυτό δεν γλιτώνει απλά πολύτιμο χρόνο, αλλά είναι και ένας εξαιρετικός τρόπος εκμάθησης σύνταξης σεναρίων Selenium.

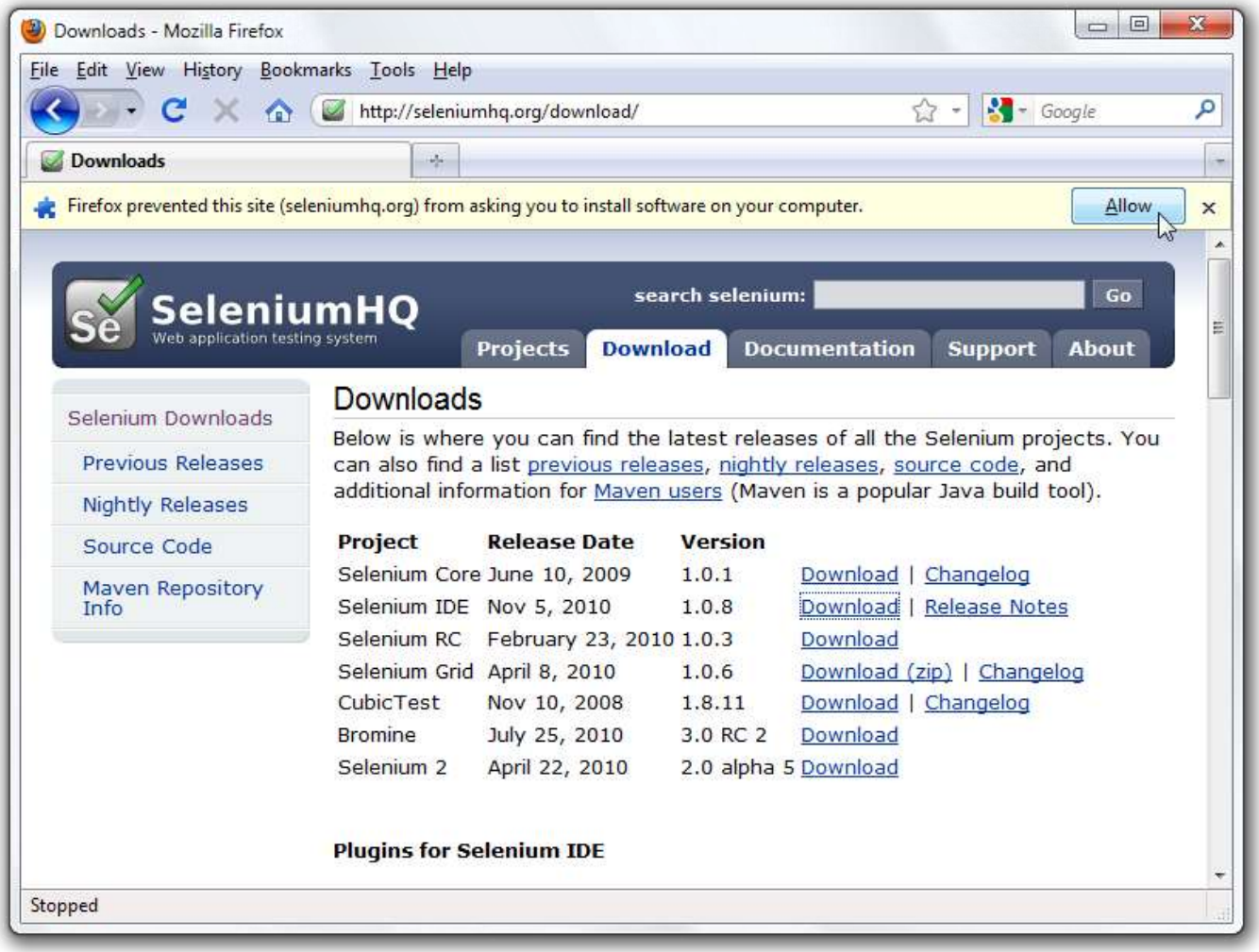
4.5.5.1 Εγκατάσταση του Selenium IDE

Για να είστε σε θέση να εργαστείτε με το Selenium και τα εργαλεία του, θα πρέπει να εγκαταστήσετε:

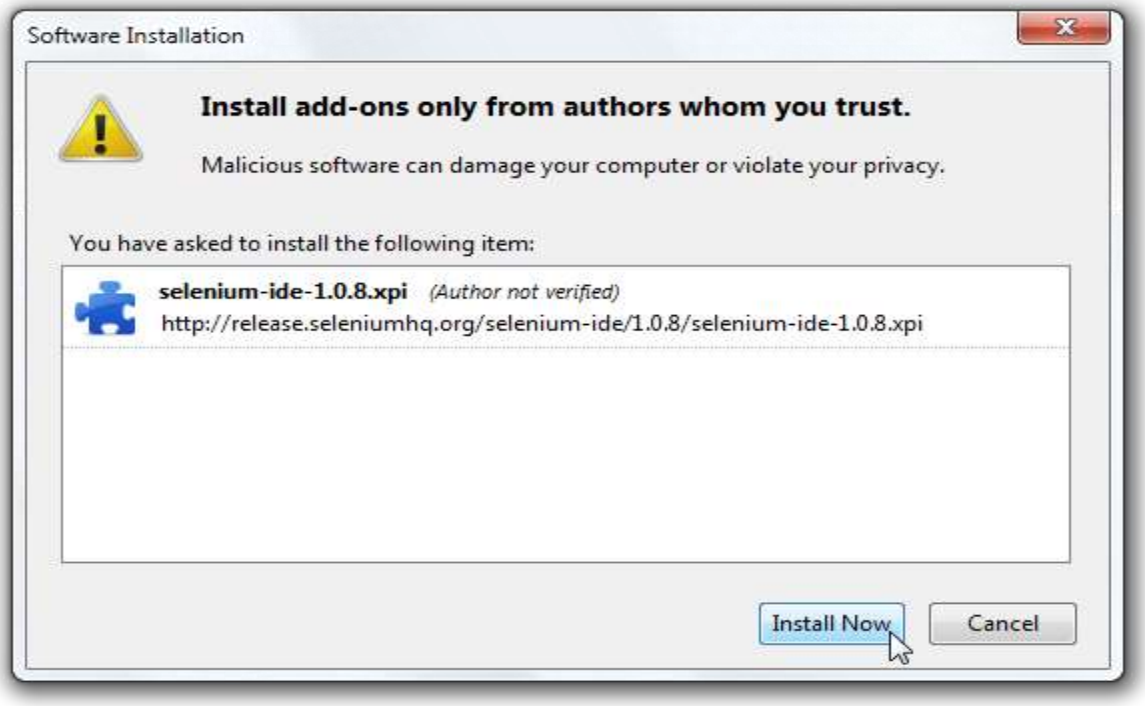
- Firefox browser : http://www.mozilla.org/
- Java JDK : http://www.oracle.com/technetwork/java/javase/downloads/index.html

Χρησιμοποιώντας Firefox, πρώτα, κατεβάστε τον IDE από τη σελίδα Selenium http://docs.seleniumhq.org/download.

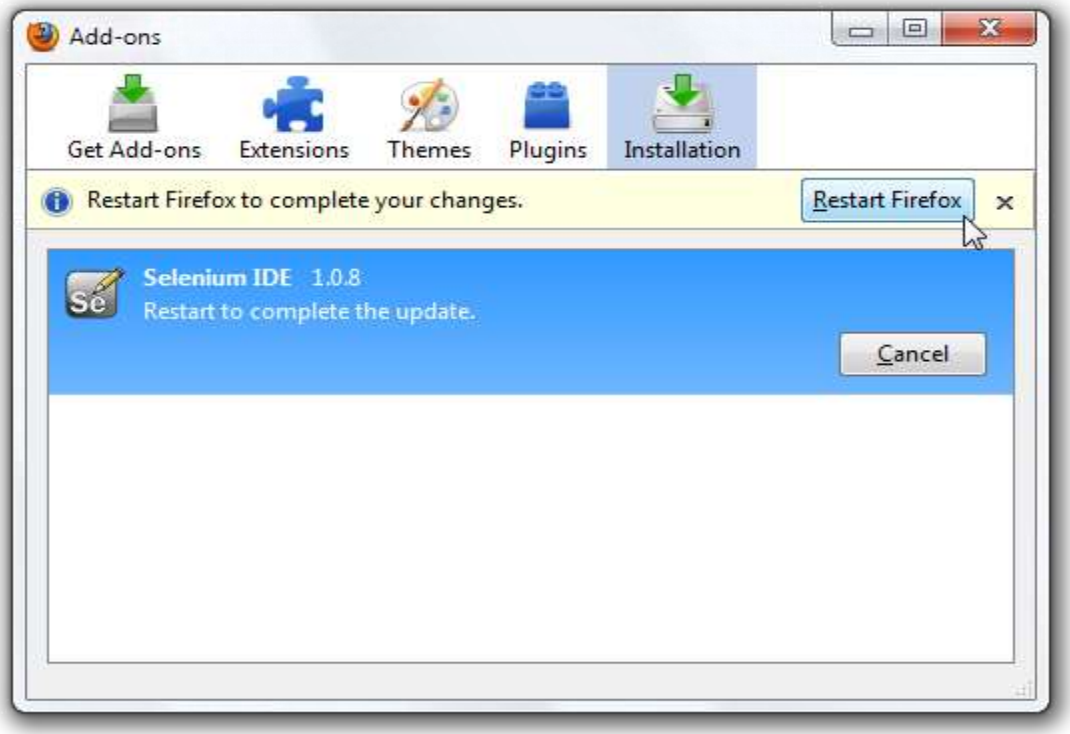
Ο Firefox θα σας προστατεύσει από την εγκατάσταση addons από άγνωστες τοποθεσίες, έτσι θα πρέπει να κάνετε κλικ στο "Να επιτρέπεται" για να συνεχίσετε με την εγκατάσταση, όπως φαίνεται στο παρακάτω screenshot.



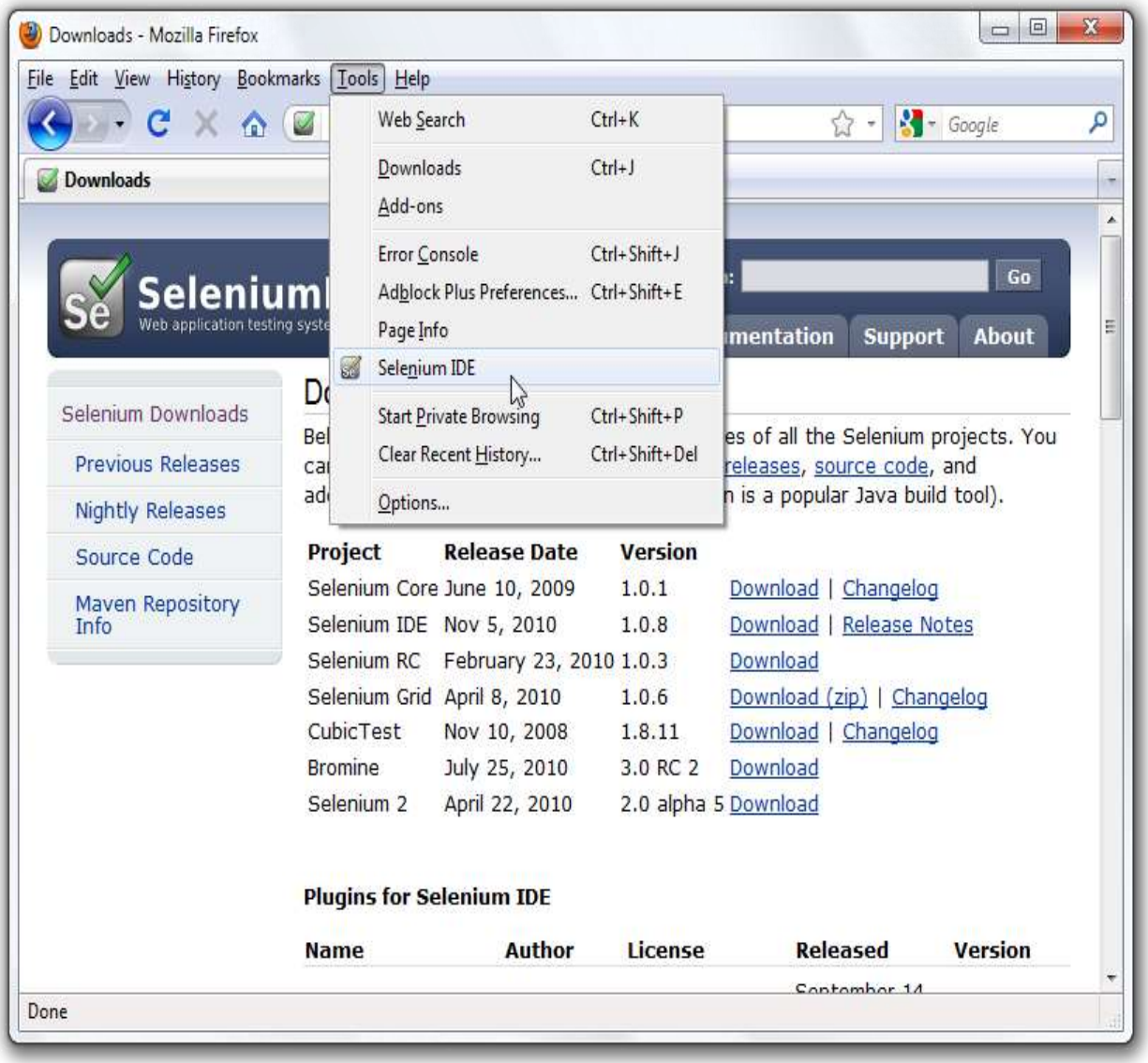
Όταν κατεβάζετε από το Firefox, θα πρέπει να σας παρουσιαστεί το παρακάτω παράθυρο.



Επιλέξτε "Install Now". Το παράθυρο του Firefox Add-ons θα εμφανιστεί, αρχικά, δείχνοντας μια γραμμή προόδου, και όταν ολοκληρωθεί η λήψη, εμφανίζεται το εξής:

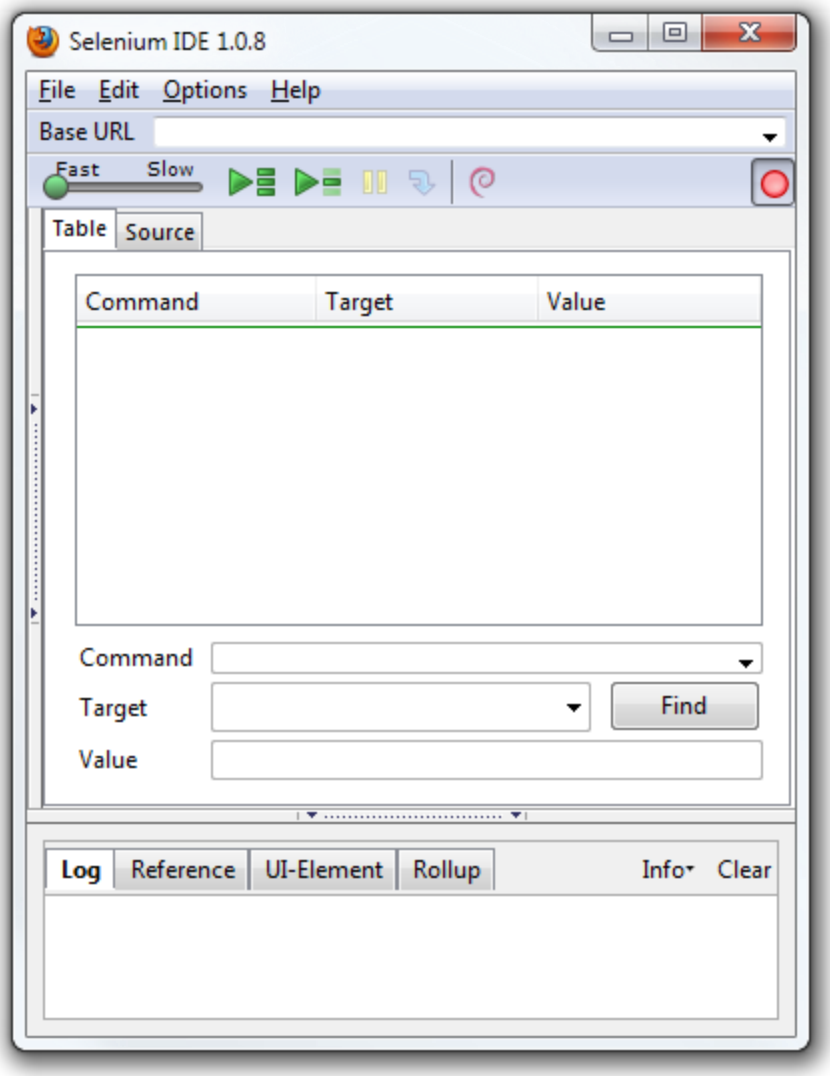


Επανεκκινήστε τον Firefox. Μετά την επανεκκίνηση του Firefox θα βρείτε το Selenium IDE στο μενού "Εργαλεία" του Firefox.



4.5.5.2 Άνοιγμα του IDE

Για να εκτελέσετε το Selenium IDE, απλά επιλέξτε το μενού "Εργαλεία" του Firefox. Ανοίγει ως εξής, με ένα άδειο παράθυρο script-editing και ένα μενού για τη φόρτωση, ή τη δημιουργία νέων περιπτώσεων ελέγχου.



4.5.5.3 Εγκατάσταση του Selenium RC

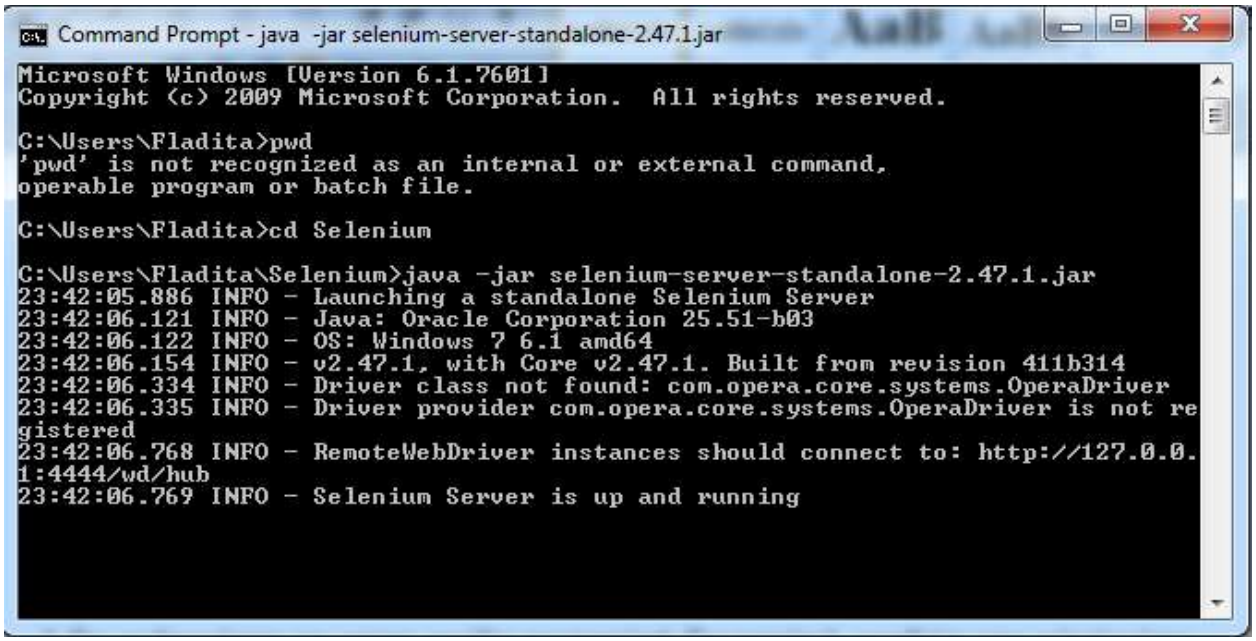
Η εγκατάσταση του Selenium Server είναι πολύ απλή:

1. Μεταβείτε στη σελίδα http://seleniumhq.org/download/ και κάντε κλικ στο σύνδεσμο για να κατεβάσετε την τελευταία έκδοση του Selenium RC.

2. Αποθηκεύστε το jar αρχείο σε ένα συγκεκριμένο φάκελο.

3. Για να ξεκινήσετε τον server, ανοίξτε τον terminal. Στο terminal, μεταβείτε στον φάκελο όπου βρίσκεται το αρχείο jar και Πληκτρολογήστε: java -jar selenium-server-standalone-2.47.1.jar

Στο τέλος θα πρέπει να δείτε ένα επιτυχές μήνυμα: "Selenium server is up and running".



4.5.6 Εγκατάσταση PHP , PEAR και PHPUnit

4.5.6.1 Εγκατάσταση PHP

Μπορείτε να κατεβάσετε την PHP σε αυτο τον σύνδεσμο http://windows.php.net/download.

Αν δεν είστε σίγουροι για το ποια έκδοση να επιλέξετε, ελέγξτε: http://stackoverflow.com/questions/5793751/what-are-the-technical-differences-between-the-thread-safe-and-non-thread-safe-p

Αφού κατεβάσετε το πρόγραμμα εγκατάστασης, τρέξτε το και εγκαταστήστε την PHP στο C: \ php \ (ή όπου αλλού θέλετε, αρκεί να θυμάστε αυτό το μονοπάτι για αργότερα). Επιλέξτε τον webserver σας (ή μην επιλέξτε τίποτα, αν δεν έχετε εγκαταστήσει webserver) και επιλέξτε τυχόν πρόσθετα στοιχεία, όπως απαιτείται, συμπεριλαμβανομένης της PEAR.

4.5.6.2 Εγκατάσταση PEAR

Αφού έχετε κατεβάσει και εγκαταστήσει την PHP, θα πρέπει να εκτελέσετε με μη αυτόματο τρόπο το αρχείο που βρίσκεται στο π.χ. c: \ php \ go-pear.bat. Εναλλακτικά, κατεβάστε το από τον σύνδεσμο http://pear.php.net/go-pear.phar και αποθηκεύσετε το σε ένα τοπικό αρχείο με το όνομα go-pear.phar. Στη συνέχεια μπορείτε να εκτελέσετε.

Ανοίξτε ένα terminal, μεταβείτε στον φάκελο που έχετε εγκαταστήσει την PHP με την εντολή

cd C:\php\

και εγκαταστήστε την PEAR με την εντολή

php go-pear.phar

Πατήστε Enter για να αποδεχτείτε την προεπιλογή όταν σας ρωτήσει “Are you installing a system-wide PEAR or a local copy?”

Πατήστε ξανά Enter για να αποδεχθεί τη διάταξη του αρχείου.

Πατήστε Enter για να ολοκληρώσετε.

4.5.6.3 Εγκατάσταση PHPUnit

Εκτελέστε τις παρακάτω εντολές (μπορεί να πάρει λίγο χρόνο για να γίνει η ενημερώση, να είστε υπομονετικοί):

pear channel-update pear.php.net

pear upgrade-all

pear update-channels

Για να εγκαταστήσετε το PHPUnit εκτελέστε:

pear install --alldeps --force phpunit

Για να ελέγξετε εάν το phpunit εγκαταστάθηκε με επιτυχία, εκτελέστε:

phpunit -v

Αν όλα είναι εντάξει, θα πρέπει να δείτε κάτι τέτοιο: PHPUnit 3.6.10 by Sebastian Bergmann ακολουθούμενο από τα περιεχόμενα.

4.5.7 Selenium IDE και PHPUnit.

Στην ενότητα αυτή, θα εξηγήσουμε πώς να χρησιμοποιήσετε το Selenium για testing. Θα ξεκινήσουμε δείχνοντάς σας πώς να γράψετε ένα τεστ χρησιμοποιώντας το Selenium IDE για τον Firefox και ύστερα πώς να εξάγετε το τεστ και να το εκτελέσετε βάσει του server Selenium (Selenium RC).

Καταγράψτε μια περίπτωση ελέγχου χρησιμοποιώντας Selenium IDE:

Μόλις εγκατασταθεί το Selenium IDE, ανοίξτε το πρόγραμμα περιήγησης Firefox, επιλέξτε Εργαλεία, στη συνέχεια, Selenium IDE ή χρησιμοποιήστε τη συντόμευση πληκτρολογίου (Ctrl + Alt + S) αν προτιμάτε.

Ο IDE θα πρέπει να εμφανιστεί, έτσι είμαστε σε καλό δρόμο μας για να δημιουργήσουμε το πρώτο αυτοματοποιημένο πρόγραμμα μας με τη χρήση του IDE. Για το πρώτο μας σενάριο, ας υποθέσουμε ότι θέλουμε να διασφαλίσουμε ότι ένας μη εγγεγραμμένος χρήστης δεν είναι σε θέση να συνδεθεί στην εφαρμογή.

Πριν φτάσουμε στη δημιουργία σεναρίου μας, ας εξοικειωθούμε με μερικές από τις βασικές λειτουργίες του Selenium, καθώς και το UI.

Τα τμήματα που χρειάζεται να γνωρίζετε αυτή τη στιγμή:

- Το πεδίο "Base URL": είναι το πεδίο στο οποίο βάζουμε τη διεύθυνση URL της ιστοσελίδας που θέλουμε να ξεκινήσει το Selenium. Στην περίπτωση μας πρέπει να καταχωρήσουμε ως διεύθυνση την : http://localhost:90/exams.

- Οι εντολές: Ακριβώς κάτω από το βασικό πεδίο URL, έχουμε τις εντολές. Από τα διάφορα κουμπιά, μπορούμε να ελέγξουμε την εκτέλεση του ελέγχου μας, καθώς και το εάν ή όχι είμαστε ενεργοί την εγγραφή.
- Ο πίνακας: Εδώ θα εμφανιστεί η λειτουργικότητα του σεναρίου μας, όπως εμείς το δημιουργούμε. Μπορούμε επίσης να επεκτείνουμε τα σενάρια μας και να προσθέσουμε επιπλέον εντολές, αν επιθυμούμε.

Για να ξεκινήσουμε την αυτοματοποίηση της περίπτωσης ελέγχου μας, πρώτα να πλοηγηθούμε στην διεύθυνση http://localhost:90/exams σε Firefox και ξεκινάμε το Selenium IDE, αν δεν είναι ήδη ανοιχτός. Εάν υπάρχουν τυχόν εντολές στον πίνακα, τα διαγράφουμε, επειδή πρόκειται να ξεκινήσουμε έναν νέο έλεγχο.

Τώρα, επιλέγουμε το κουμπί εγγραφής (το κόκκινο κύκλο στη δεξιά πλευρά των εντολών) και επιστρέφουμε στο πρόγραμμα περιήγησης Firefox με έναν πολύ φυσικό τρόπο. Από τώρα και στο εξής, όλες οι ενέργειες που κάνουμε στο πρόγραμμα περιήγησης θα πρέπει να καταγράφονται από Selenium IDE.

Θα καταγράψουμε την υπόθεση ελέγχου "Ένας μη εγγεγραμμένος χρήστης δεν είναι σε θέση να συνδεθεί στην εφαρμογή". Στη σελίδα σύνδεσης του χρήστη, στην εφαρμογή ας συμπληρώσουμε τα δύο πεδία κειμένου με εσφαλμένα στοιχεία (π.χ. «λανθασμένο» e-mail ή κωδικό πρόσβασης) και, στη συνέχεια, κάνουμε κλικ στο κουμπί "Σύνδεση".

Όπως μπορείτε να δείτε, η σελίδα μας εμφανίζει ένα μήνυμα σφάλματος. Κάντε δεξί κλικ στο μήνυμα λάθους (ή μέρος αυτής) και επιλέξτε verifyTextPresent. Η εντολή verifyTextPresent επαληθεύει εάν το συγκεκριμένο κείμενο είναι παρόν στην ιστοσελίδα ή όχι.

Τώρα ας πατήσουμε πάλι το κουμπί εγγραφής στο IDE και τελειώσαμε! Όπως μπορείτε να δείτε, όλες οι ενέργειες που κάναμε έχουν καταγραφεί στον πίνακα για να αναπαραχθούν και πάλι όποτε το επιθυμούμε.

Τώρα, αν κάνετε κλικ στο κουμπί 'Play' στο μενού εντολών, το Selenium IDE θα αναπαράγει τα ακριβή βήματα, καθώς επίσης τα βεβαιωθείτε ότι ο χρήστης δεν είναι σε θέση να συνδεθεί στην εφαρμογή και ότι το κείμενο είναι παρόν!

4.5.8 Εξαγωγή της περίπτωσης ελέγχου ως PHPUnit σενάριο:

Για να είστε σε θέση να ακολουθήσετε αυτή την ενότητα, το PHPUnit πρέπει να είναι ήδη εγκατεστημένο. Σε περίπτωση που δεν έχετε εγκατεστημένο το PHPUnit, οι παρακάτω σύνδεσμοι θα σας είναι χρήσιμοι :

- Εγκαταστήστε το PHPUnit στα Windows 7: http://www.mark-leong.com/installing-php-and-phpunit-on-windows-7/
- Εγκαταστήστε το PHPUnit στο Ubuntu Unix: http://jes.st/2011/installing-phpunit-ubuntu-linux/

Το πιό ενδιαφέρον χαρακτηριστικό του Selenium IDE είναι η εξαγωγή σεναρίων σε Selenium RC. Το Selenium IDE είναι σε θέση να εξάγει περιπτώσεις ελέγχου στις ακόλουθες μορφές:

- Java (Junit 3/4 and TestNG)
- Groovy (Junit)
- C#
- Perl
- Python
- PHP
- Ruby (RSpec and Test/Unit)

Τώρα θα εξάγουμε το τεστ που μόλις δημιουργήσαμε στο PHPUnit και θα το κτελέσουμε στο Selenium Server.

Για να εξαγάγετε το τεστ:

1. Πηγαίνετε στο File>”Export test case As…”

2. Επιλέξτε τη γλώσσα, σε αυτό το παράδειγμα: PHP

3. Εισάγετε το όνομα του αρχείου: “LoginDenied.php”

4. Confirm

Για την εκτέλεση της περίπτωσης ελέγχου που έχουμε εξαγάγει σε Selenium RC (με την προϋπόθεση ότι το Selenium Server "τρέχει"):

1. Ανοίξτε το terminal.

2.Μετακινηθείτε στον φάκελο όπου έχετε αποθηκεύσει το php αρχείο που εξάγαμε (LoginDenied.php).

3. Εκτελέστε την εντολή: phpunit LoginDenied.php

4.6 Περιπτώσεις Ελέγχου

4.6.1 Περιπτώσεις Ελέγχου για την Εγγραφή του Χρήστη.

Περιπτώσεις Ελέγχου για την Εγγραφή του Χρήστη.	
Όνομα	**Εγγραφή χρήστη**
Σκοπός	Ο σκοπός αυτής της περίπτωσης ελέγχου είναι να ελεγχθεί ότι η εγγραφή του χρήστη γίνεται σωστά.
Προϋπόθεση	Ο χρήστης έχει μεταβεί στην διεύθυνση της σελίδας και επιθυμεί να εγγραφεί.
Μετασυνθήκη	Τα δεδομένα του χρήστη έχουν αποθηκευτεί με επιτυχία και ο χρήστης λαμβάνει ένα ενημερωτικό μήνυμα.
Το προκαλεί..	Ο χρήστης επιλέγει να εγγραφεί στην ιστοσελίδα.
Δεδομένα εισόδου	Δεδομένα εγγραφής
Δεδομένα εξόδου	Μήνυμα επιτυχούς εγγραφής.
Συσχετιζόμενη "Οθόνη"	"Εγγραφή χρήστη"
Κύριο Σενάριο	
Κωδικός	ΠΕ001
Βήμα 1	Από το κύριο μενού επιλογών, επιλέξτε την επιλογή "Εγγραφή Χρήστη".
Βήμα 2	Το σύστημα θα σας εμφανίσει την οθόνη για την εγγραφή του χρήστη.
Βήμα 3	Πληκτρολογήστε τα απαραίτητα στοιχεία.
Βήμα 4	Πατήστε "Υποβολή".
Βήμα 5	Επιβεβαιώστε ότι το σύστημα σας εμφανίζει ένα μήνυμα επιτυχίας: "Η εγγραφή σας ολοκληρώθηκε με επιτυχία!".

Βήμα 6	Επιβεβαιώστε ότι το σύστημα σας εμφανίζει ένα μήνυμα και έναν σύνδεσμο για να συνδεθείτε.
Εναλλακτικό Σενάριο 1: Ο χρήστης πληκτρολογεί δύο κωδικούς οι οποίοι δεν ταιριάζουν.	
Κωδικός	ΕΣ0011
Βήμα Α1.4	Στα πεδία " Κωδικός" και " Επιβεβαίωση κωδικού " Πληκτρολογήστε δύο κωδικούς οι οποίοι δεν ταιριάζουν.
Βήμα Α1.5	Πατήστε "Υποβολή".
Βήμα Α1.6	Επιβεβαιώστε ότι το σύστημα σας εμφανίζει ένα μήνυμα λάθους: " Οι κωδικοί δεν ταιριάζουν!".
Βήμα Α1.7	Επιβεβαιώστε ότι το σύστημα σας εμφανίζει ένα μήνυμα και έναν σύνδεσμο για να προσπαθήσετε ξανά.
Εναλλακτικό Σενάριο 2: Ο χρήστης πληκτρολογεί ένα ήδη υπάρχον e-mail.	
Κωδικός	ΕΣ0012
Βήμα Α2.4	Στο πεδίο " Email" Πληκτρολογήστε ένα e-mail το οποίο υπάρχει ήδη μέσα στο σύστημα.
Βήμα Α2.5	Πατήστε "Υποβολή".
Βήμα Α2.6	Επιβεβαιώστε ότι το σύστημα σας εμφανίζει ένα μήνυμα λάθους: "Αυτό το email υπάρχει ήδη!".
Βήμα Α2.7	Επιβεβαιώστε ότι το σύστημα σας εμφανίζει ένα μήνυμα και έναν σύνδεσμο για να προσπαθήσετε ξανά.
Συμπληρωματικό Σενάριο 1: Πεδία που εμφανίζονται στη οθόνη.	
ΣΥΜ0013	Βεβαιωθείτε ότι τα πεδία που πρέπει να συμπληρώσετε στην συγκεκριμένη οθόνη είναι τα εξής: • Email • Κωδικός • Επιβεβαίωση κωδικού • Όνομα • Επώνυμο • Φύλο

4.6.2 Περιπτώσεις Ελέγχου για την Σύνδεση του Χρήστη.

Περιπτώσεις Ελέγχου για την Σύνδεση του Χρήστη.	
Όνομα	**Σύνδεση χρήστη**
Σκοπός	Ο σκοπός αυτής της περίπτωσης ελέγχου είναι να ελεγχθεί ότι η σύνδεση του χρήστη γίνεται σωστά.

Προϋπόθεση	Ο χρήστης έχει μεταβεί στην διεύθυνση της σελίδας και επιθυμεί να συνδεθεί.
Μετασυνθήκη	Τα στοιχεία του χρήστη είναι σωστά και ο χρήστης συνδέεται με επιτυχία.
Το προκαλεί..	Ο χρήστης επιλέγει να συνδεθεί στην ιστοσελίδα.
Δεδομένα εισόδου	E-mail και κωδικός πρόσβασης.
Δεδομένα εξόδου	Μήνυμα επιτυχούς σύνδεσης.
Συσχετιζόμενη "Οθόνη"	"Είσοδος"
Κύριο Σενάριο	
Κωδικός	ΠΕ002
Βήμα 1	Από το κύριο μενού επιλογών, επιλέξτε την επιλογή "Σύνδεση".
Βήμα 2	Το σύστημα θα σας εμφανίσει την οθόνη για την σύνδεση του χρήστη.
Βήμα 3	Πληκτρολογήστε το e-mail και τον κωδικό πρόσβασης.
Βήμα 4	Πατήστε "Σύνδεση ".
Βήμα 5	Επιβεβαιώστε ότι συνδεθήκατε με επιτυχία βλέποντας την επιλογή "Έξοδος" αντί για "Σύνδεση" στο κύριο μενού επιλογών.
Εναλλακτικό Σενάριο 1: Ο χρήστης πληκτρολογεί λάθος κωδικό πρόσβασης.	
Κωδικός	ΕΣ0021
Βήμα Α1.4	Στο πεδίο " Κωδικός πρόσβασης" Πληκτρολογήστε έναν κωδικό ο οποίος δεν είναι έγκυρος για το συγκεκριμένο e-mail.
Βήμα Α1.5	Πατήστε "Σύνδεση ".
Βήμα Α1.6	Επιβεβαιώστε ότι το σύστημα σας εμφανίζει ένα μήνυμα λάθους: "Λάθος κωδικός!".
Βήμα Α1.7	Επιβεβαιώστε ότι το σύστημα σας εμφανίζει ένα μήνυμα και έναν σύνδεσμο για να δοκιμάστε ξανά.
Εναλλακτικό Σενάριο 2: Ο χρήστης πληκτρολογεί ένα μη υπάρχον e-mail.	
Κωδικός	ΕΣ0022
Βήμα Α2.4	Στο πεδίο " Email" Πληκτρολογήστε ένα μη υπάρχον e-mail και κάποιο κωδικό πρόσβασης.
Βήμα Α2.5	Πατήστε "Σύνδεση ".
Βήμα Α2.6	Επιβεβαιώστε ότι το σύστημα σας εμφανίζει ένα μήνυμα λάθους: "Ο συγκεκριμένος χρήστης δεν βρέθηκε!".

Βήμα Α2.7	Επιβεβαιώστε ότι το σύστημα σας εμφανίζει ένα μήνυμα και έναν σύνδεσμο για να δοκιμάστε ξανά.
Εναλλακτικό Σενάριο 3: Ο χρήστης δεν πληκτρολογεί το e-mail ή τον κωδικό πρόσβασης.	
Κωδικός	ΕΣ0023
Βήμα Α3.3	Αφήστε κενό το πεδίο του e-mail ή του κωδικού πρόσβασης.
Βήμα Α3.4	Πατήστε "Σύνδεση ".
Βήμα Α3.5	Επιβεβαιώστε ότι το σύστημα σας εμφανίζει ένα προειδοποιητικό μήνυμα: "Please fill out this field".
Εναλλακτικό Σενάριο 4: Αποσύνδεση χρήστη.	
Κωδικός	ΕΣ0024
Βήμα Α4.1	Από το κύριο μενού επιλογών, επιλέξτε την επιλογή "Έξοδος".
Βήμα Α4.2	Επιβεβαιώστε ότι αποσυνδεθήκατε με επιτυχία βλέποντας την επιλογή "Σύνδεση" αντί για "Έξοδος" στο κύριο μενού επιλογών .
Συμπληρωματικό Σενάριο 1: Πεδία που εμφανίζονται στην οθόνη.	
ΣΥΜ0025	Βεβαιωθείτε ότι τα πεδία που πρέπει να συμπληρώσετε στην συγκεκριμένη οθόνη είναι τα εξής: • Email • Password

4.6.3 Περιπτώσεις Ελέγχου για την Εισαγωγή Εξετάσεων του Χρήστη.

Περιπτώσεις Ελέγχου για την Εισαγωγή Εξετάσεων του Χρήστη.	
Όνομα	Εισαγωγή Εξετάσεων του Χρήστη.
Σκοπός	Ο σκοπός αυτής της περίπτωσης ελέγχου είναι να ελεγχθεί ότι η εισαγωγή εξετάσεων του χρήστη γίνεται σωστά.
Προϋπόθεση	Ο χρήστης έχει συνδεθεί στην σελίδα και επιθυμεί να συμπληρώσει τον ιατρικό του φάκελο / να εισάγει τα αποτελέσματα των εξετάσεων του.
Μετασυνθήκη	Οι εξετάσεις του χρήστη εισάγονται με επιτυχία και ο χρήστης λαμβάνει ένα σχετικό μήνυμα.
Το προκαλεί..	Ο χρήστης επιλέγει να εισάγει τα αποτελέσματα των εξετάσεων του.
Δεδομένα εισόδου	Αποτελέσματα εξετάσεων.
Δεδομένα εξόδου	Μήνυμα επιτυχής καταγραφής εξετάσεων.
Συσχετιζόμενη "Οθόνη"	" ΙΑΤΡΙΚΟΣ ΦΑΚΕΛΟΣ - ΕΙΣΑΓΩΓΗ ΕΞΕΤΑΣΕΩΝ "
Κύριο Σενάριο	
Κωδικός	ΠΕ003

Βήμα 1	Από το κύριο μενού επιλογών, επιλέξτε την επιλογή "Καταχώρηση Εξέτασης".
Βήμα 2	Το σύστημα θα σας εμφανίσει την οθόνη για την Καταχώρηση Εξετάσεων του χρήστη.
Βήμα 3	Εισάγετε τα αποτελέσματα των εξετάσεων σας (από ένα - όλα τα πεδία).
Βήμα 4	Πατήστε "Υποβολή".
Βήμα 5	Επιβεβαιώστε ότι το σύστημα σας εμφανίζει ένα μήνυμα επιτυχίας: "Η καταγραφή των εξετάσεων σας ήταν επιτυχής!".
Βήμα 6	Επιβεβαιώστε ότι το σύστημα σας εμφανίζει έναν σύνδεσμο για να επιστρέψετε πίσω.
Εναλλακτικό Σενάριο 1: Ο χρήστης επιλέγει να καταχωρήσει τις εξετάσεις χωρίς να έχει συνδεθεί πρώτα.	
Κωδικός	ΕΣ0031
Βήμα Α1.2	Επιβεβαιώστε ότι το σύστημα σας εμφανίζει ένα μήνυμα: "Πρέπει να συνδεθείτε πρώτα. Παρακαλώ περιμένετε...".
Βήμα Α1.3	Επιβεβαιώστε ότι το σύστημα ανανεώνει αυτόματα την οθόνη και γυρίζει στην αρχική σελίδα.
Συμπληρωματικό Σενάριο 1: Πεδία που εμφανίζονται στις " Αιματολογικές Εξετάσεις".	
ΣΥΜ0032	Βεβαιωθείτε ότι τα πεδία που πρέπει να συμπληρώσετε στις " Αιματολογικές Εξετάσεις" είναι τα εξής: • Ερυθρά Αιμοσφαίρια (RBC) • Αιμοσφαιρίνη (HGB) • Αιματοκρίτης (HT) • Λευκά Αιμοσφαίρια (WBC)
Συμπληρωματικό Σενάριο 2: Πεδία που εμφανίζονται στις " Βιοχημικές Εξετάσεις".	
ΣΥΜ0033	Βεβαιωθείτε ότι τα πεδία που πρέπει να συμπληρώσετε στις "Αιματολογικές Εξετάσεις" είναι τα εξής: • Σάκχαρο νηστείας • Κρεατινίνη (CRE) • Ουρία (URE) • Τριγλυκερίδια • Χοληστερόλη • Υψηλής πυκνότητας λιποπρωτεΐνη (HDL) • Γλυκοζυλιωμένη αιμοσφαιρίνη (HbA1c)
Συμπληρωματικό Σενάριο 3: Πεδία που εμφανίζονται στις " Ορμονολογικές Εξετάσεις".	
ΣΥΜ0034	Βεβαιωθείτε ότι τα πεδία που πρέπει να συμπληρώσετε στις " Ορμονολογικές Εξετάσεις" είναι τα εξής: • Τριιωδοθυρονίνη (T3) • Θυροξίνη (T4) • Αλβουμίνη • Θυρεοειδοτρόπος ορμόνη (TSH)

4.6.4 Περιπτώσεις Ελέγχου για την Αναζήτηση Εξετάσεων του Χρήστη.

Περιπτώσεις Ελέγχου για την Αναζήτηση Εξετάσεων του Χρήστη.	
Όνομα	Αναζήτηση Εξετάσεων Χρήστη
Σκοπός	Ο σκοπός αυτής της περίπτωσης ελέγχου είναι να ελεγχθεί ότι η αναζήτηση των εξετάσεων του χρήστη γίνεται σωστά.
Προϋπόθεση	Ο χρήστης είναι συνδεδεμένος και έχει καταχωρημένες εξετάσεις.
Μετασυνθήκη	Στον χρήστη παρουσιάζονται τα αποτελέσματα των εξετάσεων του σύμφωνα με τα κριτήρια της αναζήτησης.
Το προκαλεί..	Ο χρήστης επιλέγει να αναζητήσει τα αποτελέσματα των εξετάσεων του.
Δεδομένα εισόδου	"Κατηγορία Εξετάσεων" ή " Ημερομηνία Εξετάσεων".
Δεδομένα εξόδου	Αποτελέσματα εξετάσεων.
Συσχετιζόμενη "Οθόνη"	" ΙΑΤΡΙΚΟΣ ΦΑΚΕΛΟΣ"
Κύριο Σενάριο	
Κωδικός	ΠΕ004
Βήμα 1	Από το κύριο μενού επιλογών, επιλέξτε την επιλογή "Αναζήτηση εξετάσεων".
Βήμα 2	Το σύστημα θα σας εμφανίσει την οθόνη "Ιατρικός Φάκελος".
Βήμα 3	Επιλέξτε να κάνετε την αναζήτηση με βάσει ένα από τα δύο κριτήρια.
Βήμα 4	Επιλέξτε να κάνετε την αναζήτηση βάσει της κατηγορίας των εξετάσεων.
Βήμα 5	Επιλέξτε μια κατηγορία εξετάσεων.
Βήμα 6	Πατήστε "Υποβολή".
Βήμα 7	Στα αποτελέσματα που θα παρουσιαστούν μπροστά σας βεβαιωθείτε ότι η τιμή στο πεδίο "Εξέταση" ανήκει όντως στην κατηγορία εξέτασης που επιλέξατε.
Εναλλακτικό Σενάριο 1: Ο χρήστης επιλέγει να αναζητήσει με βάσει την ημερομηνία εξέτασης.	
Κωδικός	ΕΣ0041
Βήμα Α1.4	Επιλέξτε να κάνετε την αναζήτηση βάσει της ημερομηνίας των εξετάσεων.
Βήμα Α1.5	Επιλέξτε μια ημερομηνία εξέτασης.
Βήμα Α1.6	Πατήστε "Υποβολή".

Βήμα Α1.7	Στα αποτελέσματα που θα παρουσιαστούν μπροστά σας βεβαιωθείτε ότι οι ημερομηνίες σε όλες τις εξετάσεις συμφωνούν με την ημερομηνία που επιλέξατε.
Συμπληρωματικό Σενάριο 1: Κριτήρια αναζήτησης.	
ΣΥΜ0042	Βεβαιωθείτε ότι τα κριτήρια αναζήτησης είναι δύο: • Αναζήτηση Βάσει Κατηγορίας Εξετάσεων • Αναζήτηση Βάσει Ημερομηνίας Εξετάσεων
Συμπληρωματικό Σενάριο 2: Κατηγορίες Εξετάσεων.	
ΣΥΜ0043	Βεβαιωθείτε ότι οι κατηγορίες εξετάσεων είναι οι εξής: • Αιματολογικές • Βιοχημικές • Ορμονολογικές
Συμπληρωματικό Σενάριο 3: Ημερομηνίες Εξετάσεων.	
ΣΥΜ0044	Βεβαιωθείτε ότι οι ημερομηνίες εξετάσεων τις οποίες μπορείτε να επιλέξετε είναι όλες οι ημερομηνίες στις οποίες έχετε καταχωρήσει εξετάσεις.

4.6.5 Περιπτώσεις Ελέγχου για την Προβολή Αποτελεσμάτων των Εξετάσεων.

Περιπτώσεις Ελέγχου για την Προβολή Αποτελεσμάτων των Εξετάσεων	
Όνομα	Προβολή Αποτελεσμάτων των Εξετάσεων
Σκοπός	Ο σκοπός αυτής της περίπτωσης ελέγχου είναι να ελεγχθεί ότι τα αποτελέσματα των εξετάσεων προβάλλονται σωστά.
Προϋπόθεση	Ο χρήστης έχει συνδεθεί στην σελίδα και επιθυμεί να ενημερωθεί για τα αποτελέσματα των εξετάσεων του.
Μετασυνθήκη	Τα αποτελέσματα των εξετάσεων εμφανίζονται στον χρήστη με συγκεκριμένη μορφή.
Το προκαλεί..	Ο χρήστης επιλέγει να αναζητήσει τα αποτελέσματα των εξετάσεων του.
Δεδομένα εισόδου	Τρόπος αναζήτησης εξετάσεων.
Δεδομένα εξόδου	Αποτελέσματα εξετάσεων.
Συσχετιζόμενη "Οθόνη"	"Αποτελέσματα εξετάσεων"
Κύριο Σενάριο	
Κωδικός	ΠΕ005
Βήμα 1	Από το αποτελέσματα που βλέπετε στην οθόνη σας επιλέξτε να έναν σύνδεσμο "Προτάσεις για να βελτιώσετε την υγεία σας".

Βήμα 2	Το σύστημα θα σας εμφανίσει την οθόνη έναν πίνακα με τις προτάσεις για να βελτιώσετε την υγεία σας.
Συμπληρωματικό Σενάριο 1: Στήλες που εμφανίζονται στην οθόνη των αποτελεσμάτων αναζήτησης.	
ΣΥΜ0051	Βεβαιωθείτε ότι η οθόνη περιλαμβάνει τέσσερεις στήλες: • Ημερομηνία • Εξέταση • Τιμή • Χαρακτηρισμός ορίων
Συμπληρωματικό Σενάριο 2: Στήλες που εμφανίζονται στην οθόνη με τα αποτελέσματα.	
ΣΥΜ0052	Βεβαιωθείτε ότι όταν οι τιμές στη στήλη "Τιμή" είναι: • Χαμηλότερες του φυσιολογικού στη στήλη "Χαρακτηρισμός ορίων εμφανίζεται η φράση "Τιμές χαμηλότερες του φυσιολογικού" ακολουθούμενη από έναν σύνδεσμο " Προτάσεις για να βελτιώσετε την υγεία σας". • Υψηλότερες του φυσιολογικού στη στήλη "Χαρακτηρισμός ορίων εμφανίζεται η φράση "Τιμές υψηλότερες του φυσιολογικού" ακολουθούμενη από έναν σύνδεσμο " Προτάσεις για να βελτιώσετε την υγεία σας". • Εντός ορίων, εμφανίζεται η φράση "Φυσιολογικές τιμές".
Συμπληρωματικό Σενάριο 3: Πεδία που εμφανίζονται στην οθόνη όταν επιλέγεται ο σύνδεσμος " Προτάσεις για να βελτιώσετε την υγεία σας".	
ΣΥΜ0053	Βεβαιωθείτε ότι όταν ο χρήστης επιλέγει τον σύνδεσμο " Προτάσεις για να βελτιώσετε την υγεία σας" εμφανίζεται ένας πίνακας με τα εξής πεδία: • Πιθανές ασθένειες ή προβλήματα • Προτεινόμενα είδη τροφίμων • Τρόποι Αντιμετώπισης • Τι πρέπει να αποφύγετε

4.7 Αυτοματοποιημένη ή χειρωνακτική εκτέλεση;

Μετά από μελέτη αποφασίστηκε ποιές από τις περιπτώσεις ελέγχου θα εκτελεστούν χειρωνακτικά και ποιές θα αυτοματοποιηθούν. Η απόφαση πάρθηκε βάσει της πολυπλοκότητας της περίπτωσης αλλά και βάσει της συχνότητας που πιθανόν θα εκτελεστεί μια περίπτωση ελέγχου καθ'ολη την διάρκεια του κύκλου ζωής του λογισμικού.

Στον παρακάτω πίνακα φαίνονται τα αποτελέσματα:

Κωδικός Περίπτωσης Ελέγχου	Αυτοματοποιημένη Εκτέλεση	Χειρωνακτική Εκτέλεση
ΠΕ001	✓	
ΕΣ0011	✓	
ΕΣ0012	✓	
ΣΥΜ0013	✓	

Κωδικός Περίπτωσης Ελέγχου	Αυτοματοποιημένη Εκτέλεση	Χειρωνακτική Εκτέλεση
ΠΕ002	✓	
ΕΣ0021	✓	
ΕΣ0022	✓	
ΕΣ0023		✓
ΕΣ0024		✓
ΣΥΜ0025		✓
ΠΕ003	✓	
ΕΣ0031		✓
ΣΥΜ0032	✓	
ΣΥΜ0033	✓	
ΣΥΜ0034	✓	
ΠΕ004	✓	
ΕΣ0041	✓	
ΣΥΜ0042	✓	
ΣΥΜ0043		✓
ΣΥΜ0044		✓
ΠΕ005		✓
ΣΥΜ0051	✓	
ΣΥΜ0052		✓
ΣΥΜ0053	✓	

4.7.1 Scripts αυτοματοποιημένων περιπτώσεων ελέγχου

Παρακάτω παραθέτουμε τα scripts για τις περιπτώσεις ελέγχου που αποφασίστηκε ότι πρέπει να αυτοματοποιηθούν.

Περίπτωση ελέγχου ΠΕ001 - Επιτυχής εγγραφή χρήστη:

```
<?php
class PE001 extends PHPUnit_Extensions_SeleniumTestCase
{
  protected function setUp()
```

```
 {
  $this->setBrowser("*chrome");
  $this->setBrowserUrl("http://localhost:90/");
 }
 public function testMyTestCase()
 {
  $this->open("/exams/index.php");
  $this->click("link=Εγγραφή χρήστη");
  $this->waitForPageToLoad("30000");
  $this->type("name=email", "flantita@gmail.com");
  $this->type("name=password_r", "123456789");
  $this->type("name=password_r2", "123456789");
  $this->type("name=fname", "Fladita");
  $this->type("name=lname", "Chazizai");
  $this->select("name=sex", "label=Γυναίκα");
  $this->click("id=saveForm");
  $this->waitForPageToLoad("30000");
  $this->verifyText("css=body", "Η εγγραφή σας ολοκληρώθηκε με επιτυχία!\nΠαρακαλώ
   συνδεθείτε...");  }}?>
```

Περίπτωση ελέγχου ΕΣ0011 - Ο χρήστης πληκτρολογεί δύο κωδικούς οι οποίοι δεν ταιριάζουν:

```
<?php
class ES0011 extends PHPUnit_Extensions_SeleniumTestCase
{
 protected function setUp()
 {
  $this->setBrowser("*chrome");
  $this->setBrowserUrl("http://localhost:90/");
 }
 public function testMyTestCase()
 {
  $this->open("/exams/index.php");
  $this->click("link=Εγγραφή χρήστη");
  $this->waitForPageToLoad("30000");
```

```
    $this->type("name=email", "flanti@gmail.com");
    $this->type("name=password_r", "123456789");
    $this->type("name=password_r2", "123456");
    $this->type("name=fname", "Fladita");
    $this->type("name=lname", "Chazizai");
    $this->select("name=sex", "label=Γυναίκα");
    $this->click("id=saveForm");
    $this->waitForPageToLoad("30000");
    $this->verifyText("css=body", "Οι κωδικοί δεν ταιριάζουν!\nΠαρακαλώ προσπαθήστε
     ξανά...");}}?>
```

Περίπτωση ελέγχου ΕΣ0012 - Ο χρήστης πληκτρολογεί ένα ήδη υπάρχον e-mail.

```
<?php
class ES0012 extends PHPUnit_Extensions_SeleniumTestCase
{
  protected function setUp()
  {
    $this->setBrowser("*chrome");
    $this->setBrowserUrl("http://localhost:90/");
  }
  public function testMyTestCase()
  {
    $this->open("/exams/index.php");
    $this->click("link=Εγγραφή χρήστη");
    $this->waitForPageToLoad("30000");
    $this->type("name=email", "flantita@gmail.com");
    $this->type("name=password_r", "123456");
    $this->type("name=password_r2", "123456");
    $this->type("name=fname", "Fladita");
    $this->type("name=lname", "Chazizai");
    $this->select("name=sex", "label=Γυναίκα");
    $this->click("id=saveForm");
    $this->waitForPageToLoad("30000");
    $this->verifyText("css=body", " Αυτό το email υπάρχει ήδη!\nΠαρακαλώ προσπαθήστε
```

```
    ξανά...");}}?>
```

Περίπτωση ελέγχου ΣΥΜ0013 - Πεδία που εμφανίζονται στη οθόνη:

```
<?php
class SYM0013 extends PHPUnit_Extensions_SeleniumTestCase
{
  protected function setUp()
  {
    $this->setBrowser("*chrome");
    $this->setBrowserUrl("http://localhost:90/");
  }
  public function testMyTestCase()
  {
    $this->open("/exams/index.php");
    $this->click("link=Εγγραφή χρήστη");
    $this->waitForPageToLoad("30000");
    $this->verifyText("css=td", "Email:");
    $this->verifyText("//article[@id='top']/div/div[2]/form/table/tbody/tr[2]/td", "Κωδικός:");
    $this->verifyText("//article[@id='top']/div/div[2]/form/table/tbody/tr[3]/td", "Επιβεβαίωση
    κωδικού:");
    $this->verifyText("//article[@id='top']/div/div[2]/form/table/tbody/tr[4]/td", "Όνομα:");
    $this->verifyText("//article[@id='top']/div/div[2]/form/table/tbody/tr[5]/td", "Επώνυμο:");
    $this->verifyText("//article[@id='top']/div/div[2]/form/table/tbody/tr[6]/td", "Φύλο:");  }}?>
```

Περίπτωση ελέγχου ΠΕ002 - Επιτυχής σύνδεση χρήστη:

```
<?php
class PE002 extends PHPUnit_Extensions_SeleniumTestCase
{
  protected function setUp()
  {
    $this->setBrowser("*chrome");
    $this->setBrowserUrl("http://localhost:90/");
  }
  public function testMyTestCase()
```

```
 {
  $this->open("/exams/index.php");
  $this->click("link=Σύνδεση");
  $this->waitForPageToLoad("30000");
  $this->type("id=email", "flantita@gmail.com");
  $this->type("id=password", "123456789");
  $this->click("css=input[type=\"submit\"]");
  $this->waitForPageToLoad("30000");
  $this->verifyText("css=center", "Σύστημα Καταγραφής Ιστορικού \nΙατρικών Εξετάσεων"); }}?>
```

Περίπτωση ελέγχου ΕΣ0021 - Ο χρήστης πληκτρολογεί λάθος κωδικό πρόσβασης.

```
<?php
class ES0021 extends PHPUnit_Extensions_SeleniumTestCase
{
 protected function setUp()
 {
  $this->setBrowser("*chrome");
  $this->setBrowserUrl("http://localhost:90/");
 }
 public function testMyTestCase()
 {
  $this->open("/exams/index.php");
  $this->click("link=Σύνδεση");
  $this->waitForPageToLoad("30000");
  $this->type("id=email", "flantita@gmail.com");
  $this->type("id=password", "8888888");
  $this->click("css=input[type=\"submit\"]");
  $this->waitForPageToLoad("30000");
  $this->verifyText("css=body", "Λάθος κωδικός! \nΠαρακαλώ δοκιμάστε ξανά...");  }}?>
```

Περίπτωση ελέγχου ΕΣ0022 - Ο χρήστης πληκτρολογεί ένα μη υπάρχον e-mail:

```
<?php
class ES0022 extends PHPUnit_Extensions_SeleniumTestCase
```

```
{
  protected function setUp()
  {
    $this->setBrowser("*chrome");
    $this->setBrowserUrl("http://localhost:90/");
  }
  public function testMyTestCase()
  {
    $this->open("/exams/login.html");
    $this->type("id=email", "flanti777@gmail.com");
    $this->type("id=password", "123456");
    $this->click("css=input[type=\"submit\"]");
    $this->waitForPageToLoad("30000");
    $this->verifyText("css=body", "Δεν βρέθηκε ο χρήστης αυτός! \nΠαρακαλώ δοκιμάστε
    ξανά...");  }}?>
```

Περίπτωση ελέγχου ΠΕ003 - Επιτυχής Εισαγωγή Εξετάσεων του Χρήστη:

```
<?php
class PE003 extends PHPUnit_Extensions_SeleniumTestCase
{
  protected function setUp()
  {
    $this->setBrowser("*chrome");
    $this->setBrowserUrl("http://localhost:90/");
  }
  public function testMyTestCase()
  {
    $this->open("/exams/index.php");
    $this->click("link=Καταχώρηση Εξέτασης");
    $this->waitForPageToLoad("30000");
    $this->type("name=1", "12");
    $this->type("name=2", "14");
    $this->type("name=3", "45");
    $this->type("name=12", "45");
```

```
    $this->type("name=4", "45");
    $this->type("name=5", "74");
    $this->type("name=6", "45");
    $this->type("name=7", "52");
    $this->type("name=8", "41");
    $this->type("name=9", "45");
    $this->type("name=11", "74");
    $this->type("name=10", "85");
    $this->click("id=saveForm");
    $this->waitForPageToLoad("30000");
    $this->verifyText("css=body", "Η καταγραφή των εξετάσεων σας ήταν επιτυχής!\nΕπιστροφή πατώντας εδώ...");    }}?>
```

Περίπτωση ελέγχου ΣΥΜ0032 - Πεδία που εμφανίζονται στις " Αιματολογικές Εξετάσεις".

```
<?php
class SYM0032 extends PHPUnit_Extensions_SeleniumTestCase
{
  protected function setUp()
  {
    $this->setBrowser("*chrome");
    $this->setBrowserUrl("http://localhost:90/");
  }
  public function testMyTestCase()
  {
    $this->open("/exams/index.php");
    $this->click("link=Καταχώρηση Εξέτασης");
    $this->waitForPageToLoad("30000");
    $this->verifyText("css=h3", "ΑΙΜΑΤΟΛΟΓΙΚΕΣ");
    $this->verifyText("css=td", "Ερυθρά Αιμοσφαίρια (RBC)");
    $this->verifyText("//li[@id='foli130']/fieldset/div/table/tbody/tr[2]/td", "Αιμοσφαιρίνη (HGB)");
    $this->verifyText("//li[@id='foli130']/fieldset/div/table/tbody/tr[3]/td", "Αιματοκρίτης (HT)");
    $this->verifyText("//li[@id='foli130']/fieldset/div/table/tbody/tr[4]/td", "Λευκά Αιμοσφαίρια (WBC)");  }}?>
```

Περίπτωση ελέγχου ΣΥΜ0033 - Πεδία που εμφανίζονται στις "Βιοχημικές Εξετάσεις".

```
<?php
```

```
class SYM0033 extends PHPUnit_Extensions_SeleniumTestCase
{
  protected function setUp()
  {
    $this->setBrowser("*chrome");
    $this->setBrowserUrl("http://localhost:90/");
  }
  public function testMyTestCase()
  {
    $this->open("/exams/index.php");
    $this->click("link=Καταχώρηση Εξέτασης");
    $this->waitForPageToLoad("30000");
    $this->verifyText("css=#header > #header > h3", "ΒΙΟΧΗΜΙΚΕΣ");
    $this->verifyText("xpath=(//li[@id='foli130']/fieldset/div/table/tbody/tr/td)[9]", "Σάκχαρο νηστείας");
    $this->verifyText("xpath=(//li[@id='foli130']/fieldset/div/table/tbody/tr[2]/td)[3]", "Κρεατινίνη (CRE)");
    $this->verifyText("xpath=(//li[@id='foli130']/fieldset/div/table/tbody/tr[3]/td)[3]", "Ουρία (URE)");
    $this->verifyText("xpath=(//li[@id='foli130']/fieldset/div/table/tbody/tr[4]/td)[3]", "Τριγλυκερίδια");
    $this->verifyText("//li[@id='foli130']/fieldset/div/table/tbody/tr[5]/td", "Χοληστερόλη");  }}?>
```

Περίπτωση ελέγχου ΣΥΜ0034 - Πεδία που εμφανίζονται στις " Ορμονολογικές Εξετάσεις".

```
<?php
class SYM0034 extends PHPUnit_Extensions_SeleniumTestCase
{
  protected function setUp()
  {
    $this->setBrowser("*chrome");
    $this->setBrowserUrl("http://localhost:90/");
  }
  public function testMyTestCase()
  {
    $this->open("/exams/index.php");
    $this->click("link=Καταχώρηση Εξέτασης");
    $this->waitForPageToLoad("30000");
    $this->verifyText("css=fieldset > #header > h3", "ΟΡΜΟΝΟΛΟΓΙΚΕΣ");
```

```
    $this->verifyText("css=fieldset > ul > #foli130 > fieldset > div > table > tbody > tr > td",
"Τριιωδοθυρονίνη (T3)");
    $this->verifyText("xpath=(//li[@id='foli130']/fieldset/div/table/tbody/tr[2]/td)[5]", "Θυροξίνη (T4)");
    $this->verifyText("xpath=(//li[@id='foli130']/fieldset/div/table/tbody/tr[3]/td)[5]", "Θυρεοειδής(TSH)");  }?>
```

Περίπτωση ελέγχου ΠΕ004 - Αναζήτηση Εξετάσεων Χρήστη βάσει κατηγορίας:

```
<?php
class PE004 extends PHPUnit_Extensions_SeleniumTestCase
{
  protected function setUp()
  {
    $this->setBrowser("*chrome");
    $this->setBrowserUrl("http://localhost:90/");
  }
  public function testMyTestCase()
  {
    $this->open("/exams/index.php");
    $this->click("link=Αναζήτηση Εξετάσεων");
    $this->waitForPageToLoad("30000");
    $this->select("name=category", "label=Βιοχημικές");
    $this->click("id=saveForm");
    $this->waitForPageToLoad("30000");
    $this->verifyText("css=center", "Βιοχημικές");  }}?>
```

Περίπτωση ελέγχου ΕΣ0041 - Ο χρήστης επιλέγει να αναζητήσει με βάσει την ημερομηνία εξέτασης.

```
<?php
class ES0041 extends PHPUnit_Extensions_SeleniumTestCase
{
  protected function setUp()
  {
    $this->setBrowser("*chrome");
    $this->setBrowserUrl("http://localhost:90/");
  }
  public function testMyTestCase()
  {
```

```
$this->open("/exams/index.php");
$this->click("link=Αναζήτηση Εξετάσεων");
$this->waitForPageToLoad("30000");
$this->select("name=date", "label=2015-10-11");
$this->click("document.form2.saveForm");
$this->waitForPageToLoad("30000");
$this->verifyText("css=center", "Ημερομηνία Εξέτασης:2015-10-11");  }}?>
```

Περίπτωση ελέγχου ΣΥΜ0042 - Κριτήρια αναζήτησης.

```
<?php
class SYM0042 extends PHPUnit_Extensions_SeleniumTestCase
{
  protected function setUp()
  {
    $this->setBrowser("*chrome");
    $this->setBrowserUrl("http://localhost:90/");
  }
  public function testMyTestCase()
  {
    $this->open("/exams/index.php");
    $this->click("link=Αναζήτηση Εξετάσεων");
    $this->waitForPageToLoad("30000");
    $this->verifyText("css=h3", "ΑΝΑΖΗΤΗΣΗ ΒΑΣΕΙ ΚΑΤΗΓΟΡΙΑΣ ΕΞΕΤΑΣΕΩΝ");
    $this->verifyText("id=title130", "Κατηγορία Εξετάσεων");
    $this->verifyText("css=#form2 > #header > h3", "ΑΝΑΖΗΤΗΣΗ ΒΑΣΕΙ ΗΜΕΡΟΜΗΝΙΑΣ
    ΕΞΕΤΑΣΕΩΝ");
    $this->verifyText("css=#form2 > ul > #foli130 > fieldset > #title130", "Ημερομηνία
    Εξετάσεων");  }}?>
```

Περίπτωση ελέγχου ΣΥΜ0051 - Στήλες που εμφανίζονται στην οθόνη των αποτελεσμάτων αναζήτησης.

```
<?php
class SYM0051 extends PHPUnit_Extensions_SeleniumTestCase
{
  protected function setUp()
```

```
 {
   $this->setBrowser("*chrome");
   $this->setBrowserUrl("http://localhost:90/");
 }
 public function testMyTestCase()
 {
   $this->open("/exams/index.php");
   $this->click("link=Αναζήτηση Εξετάσεων");
   $this->waitForPageToLoad("30000");
   $this->select("name=category", "label=Βιοχημικές");
   $this->click("id=saveForm");
   $this->waitForPageToLoad("30000");
   $this->verifyText("//th[2]", "Ημερομηνία");
   $this->verifyText("//th[3]", "Εξέταση");
   $this->verifyText("//th[4]", "Τιμή");
   $this->verifyText("//th[5]", "Χαρακτηρισμός ορίων");  }}?>
```

Περίπτωση ελέγχου ΣΥΜ0053 - Πεδία που εμφανίζονται στην οθόνη όταν επιλέγεται ο σύνδεσμος " Προτάσεις για να βελτιώσετε την υγεία σας":

```
<?php
class SYM0053 extends PHPUnit_Extensions_SeleniumTestCase
{
 protected function setUp()
 {
   $this->setBrowser("*chrome");
   $this->setBrowserUrl("http://localhost:90/");
 }
 public function testMyTestCase()
 {
   $this->open("/exams/index.php");
   $this->click("link=Αναζήτηση Εξετάσεων");
   $this->waitForPageToLoad("30000");
   $this->select("name=category", "label=Βιοχημικές");
   $this->click("id=saveForm");
```

```
$this->waitForPageToLoad("30000");
$this->click("link=Προτάσεις για να βελτειώσετε την υγεία σας");
$this->waitForPageToLoad("30000");
$this->verifyText("css=p", "Πιθανές ασθένειες ή προβλήματα:");
$this->verifyText("//th[2]/b", "Προτεινόμενα είδη τροφίμων:");
$this->verifyText("//th[3]/b", "Τρόποι Αντιμετώπισης:");
$this->verifyText("//th[4]/b", "Τι πρέπει να αποφύγετε:");  }}?>
```

4.7.2 Πρώτη Εκτέλεση

Αφού δημιουργήθηκαν και τα scripts για τις περιπτώσεις ελέγχου που θα εκτελεστούν αυτοματοποιημένα, εκτελέστηκαν όλες τις περιπτώσεις ελέγχου αυτοματοποιημένα ή χειρωνακτικά, ανάλογα με την περίπτωση και σας παραθέτουμε παρακάτω τα αποτελέσματα:

NO	Κωδικός Περίπτωσης Ελέγχου	Αποτέλεσμα Εκτέλεσης
1	ΠΕ001	**Επιτυχία**
2	ΕΣ0011	**Αποτυχία**
3	ΕΣ0012	**Αποτυχία**
4	ΣΥΜ0013	**Επιτυχία**
5	ΠΕ002	**Αποτυχία**
6	ΕΣ0021	**Επιτυχία**
7	ΕΣ0022	**Επιτυχία**
8	ΕΣ0023	**Αποτυχία**
9	ΕΣ0024	**Επιτυχία**
10	ΣΥΜ0025	**Επιτυχία**
11	ΠΕ003	**Αποτυχία**
12	ΕΣ0031	**Επιτυχία**
13	ΣΥΜ0032	**Επιτυχία**
14	ΣΥΜ0033	**Επιτυχία**
15	ΣΥΜ0034	**Αποτυχία**
16	ΠΕ004	**Αποτυχία**

NO	Κωδικός Περίπτωσης Ελέγχου	Αποτέλεσμα Εκτέλεσης
17	ΕΣ0041	Επιτυχία
18	ΣΥΜ0042	Επιτυχία
19	ΣΥΜ0043	Αποτυχία
20	ΣΥΜ0044	Αποτυχία
21	ΠΕ005	Επιτυχία
22	ΣΥΜ0051	Αποτυχία
23	ΣΥΜ0052	Επιτυχία
24	ΣΥΜ0053	Επιτυχία

Όπως παρατηρούμε παραπάνω, κατά την διάρκεια της πρώτης εκτέλεσης από τις 24 περιπτώσεις που εκτελέσαμε οι 10 αποτύχανε. Εντοπίστηκαν τα ελαττώματα των 10 αυτών περιπτώσεων ελέγχου, έγιναν οι διορθώσεις και είμαστε έτοιμοι για την δεύτερη εκτέλεση.

4.7.3 Δεύτερη Εκτέλεση

Σας παραθέτουμε παρακάτω τα αποτελέσματα της δεύτερης εκτέλεσης:

NO	Κωδικός Περίπτωσης Ελέγχου	Αποτέλεσμα Εκτέλεσης
1	ΠΕ001	Επιτυχία
2	ΕΣ0011	Επιτυχία
3	ΕΣ0012	Επιτυχία
4	ΣΥΜ0013	Επιτυχία
5	ΠΕ002	Επιτυχία
6	ΕΣ0021	Επιτυχία
7	ΕΣ0022	Επιτυχία
8	ΕΣ0023	Επιτυχία
9	ΕΣ0024	Επιτυχία
10	ΣΥΜ0025	Επιτυχία
11	ΠΕ003	Επιτυχία
12	ΕΣ0031	Επιτυχία

NO	Κωδικός Περίπτωσης Ελέγχου	Αποτέλεσμα Εκτέλεσης
13	ΣΥΜ0032	**Επιτυχία**
14	ΣΥΜ0033	**Επιτυχία**
15	ΣΥΜ0034	**Επιτυχία**
16	ΠΕ004	**Επιτυχία**
17	ΕΣ0041	**Επιτυχία**
18	ΣΥΜ0042	**Επιτυχία**
19	ΣΥΜ0043	**Επιτυχία**
20	ΣΥΜ0044	**Επιτυχία**
21	ΠΕ005	**Επιτυχία**
22	ΣΥΜ0051	**Επιτυχία**
23	ΣΥΜ0052	**Επιτυχία**
24	ΣΥΜ0053	**Επιτυχία**

Όπως παρατηρούμε και από τον παραπάνω πίνακα, οι περιπτώσεις ελέγχου για την δεύτερη εκτέλεση πραγματοποιήθηκαν με επιτυχία.

Ο έλεγχος τελειώνει εδώ!

ΣΥΜΠΕΡΑΣΜΑΤΑ

Η Ιατρική Επιστήμη εξελίσσεται και προσαρμόζεται διαρκώς στις ανάγκες της κοινωνίας. Η εξέλιξη της τεχνολογίας σε συνδυασμό με την έρευνα και τη μελέτη έχουν συμβάλει καθοριστικά στην ανάπτυξη αυτή. Κύριο στοιχείο και οδηγός αυτής της εξέλιξης στον τομέα της υγείας αποτελεί η καταγραφή των γεγονότων και η μετέπειτα ανάλυση αυτών, για να εξαχθούν συμπεράσματα που θα βοηθήσουν στην βελτίωση των συνθηκών της υγειονομικής περίθαλψης.

Η καταγραφή των περιστατικών υγείας είναι μια διαδικασία, η οποία δεν έχει νόημα να γίνεται συνολικά για ένα πληθυσμό, αλλά για κάθε άτομο χωριστά και θα πρέπει να είναι δυναμική και διαρκής. Μόνο έτσι μπορούν να εξαχθούν σωστά συμπεράσματα και να δοθούν λύσεις στα μεμονωμένα ιατρικά προβλήματα που παρουσιάζει ο καθένας μας.

Στις μέρες μας οι πολίτες έχουν γίνει πιο απαιτητικοί σε σχέση με παλιότερα σε ό,τι αφορά τις υπηρεσίες που τους προσφέρονται και ειδικά σε ένα χώρο τόσο ευαίσθητο όσο αυτός της υγείας. Έχουν την απαίτηση να έχουν την δυνατότητα ακόμα και από το σπίτι τους να είναι πλήρως ενημερωμένοι για την κατάσταση της υγείας τους, ανά πάσα στιγμή, ώστε να αισθάνονται σίγουροι και ασφαλείς.

Η ιστοσελίδα - Ηλεκτρονικός Φάκελος Υγείας, είναι χρήσιμη, όχι μόνο στον ασθενή πληθυσμό, αλλά και σε όλους όσους επιθυμούν να έχουν τα ιατρικά τους δεδομένα συγκεντρωμένα. Ένας πλήρης ιατρικός φάκελος αποτελεί από μόνος του ένα στοιχείο ασφάλειας προς το πρόσωπο το οποίο αφορά, αφού του παρέχει τη δυνατότητα να παρακολουθεί μόνος του, την πορεία της υγείας του και να ενημερώνεται κατάλληλα για τους πιθανούς τρόπους βελτίωσης της υγείας του.

Τα ιατρικά αρχεία, ωστόσο, ενός ασθενούς αποτελούν ιδιαίτερα ευαίσθητα προσωπικά δεδομένα. Αυτό σημαίνει ότι όποιος έχει πρόσβαση σε αυτά πρέπει να είναι άτομο , το οποίο δεν θα τα χρησιμοποιήσει με λάθος τρόπο και για δικό του όφελος. Η ασφάλεια των ιατρικών δεδομένων είναι ένα σημαντικότατο θέμα που πρέπει να συζητηθεί, αν και η τεχνολογία έχει δώσει αρκετές ουσιαστικές και αποτελεσματικές λύσεις.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] A. Spillner, T. Linz, H. Schaefer, Software Testing Foundations: A Certified Guide for the Certified Tester Exam, February 4, 2011

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

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

[4] A.B. AL Badareen, M. H. Selamat, M. A. Jabar, J. Din, S. Turaev, Software Quality Models: A Comparative Study, In Proc. 2nd Intl. Conference Software Engineering and Computer Systems, (ICSECS 2011), Kuantan, Pahang, Malaysia, June 27-29, 2011

[5] R. Fitzpatrick, Software Quality: Definitions and Strategic Issues, Staffordshire University, School of Computing report, April 1996

[6] G. Bath, P. Jorgensen, J. Mitchell, Certified Tester Advanced Level Syllabus Technical Test Analyst, International Software Testing Qualifications Board (ISTQB), 2010-2012

[7] J. Hinz, M. Gijsen, Fifth Generation Scriptless and Advanced Test Automation Technologies, TESTars Test Competence Centre

[8] J. Kent, Test Automation: From Record/Playback to Frameworks V1.0 , ©Simply Testing Ltd

[9] D. Hoffman, Test Automation Architectures: Planning for Test Automation, © 1999, Software Quality Methods, LLC, International Quality Week 1999

[10] G.Ukkuru, Test Automation Best Practices, November 7, 2013

[11] K.Martin, Automated testing – The Advantages and Disadvantages (QA for Software Development), June 20, 2012

[12] B. Marick, When should a test be automated?, Reliable Software Technlogies, 1998

[13] S. Hebbar, A Structured Approach to Software Test Automation, Adobe Systems India Pvt. Ltd., Bangalore, 25 December 2008

[14] L. G. Hayes, Automated testing handbook, Software Testing Inst; 2nd edition (March 1, 2004)

[15] G. J Myers, Revised and Updated by T. Badgett, T. M. Thomas, C. Sandler, The Art of Software Testing, 2nd Edition, John Wiley & Sons, Inc., Hoboken, New Jersey, 2004

[16] R. Collard, An introduction to Software Test Case Design, Workshop on Teaching Software Testing (WTST) 3, available at http://www.testingeducation.org/conference/wtst3_collard1.pdf , 2004

[17] Stevenson, Oxford Dictionary of English, Oxford University Press, USA, 3rd Revised edition edition, August 1, 2010

[18] Merriam-Webster, Collegiate Dictionary, Merriam-Webster, Inc., 11th edition, July 30, 2003

[19] R. Potts, Euclid's Elements of Geometry, HardPress Publishing, August 1, 2012

[20] ΑΕΜΥ, «Ηλεκτρονική Υγεία - Ηλεκτρονικός Ιατρικός Φάκελος», http://www.aemy.gr/web/guest/74

[21] Al Saganich, «Java and Web Services Primer», ONJava.com, Ιούλιος 2001; http://www.onjava.com/lpt/a/1025 , (24/9/2008).

[22] Amatayakul M., «Electronic Health Records: A Practical Guide for Professionals and Organizations», American Health Information Management Association, 2004, http://library.ahima.org/xpedio/groups/public/documents/ahima/bok1_015872.pd

[23] Andrew Troelsen, «Pro C# 2008 and the .NET 3.5 Platform», Fourth Edition, Apress, 2007.

[24] Arsanjani A., «Service-oriented modeling and architecture», Developer Works,Νοέμβριος 2004, http://www.ibm.com/developerworks/library/ws-soa-design1/

[25] Arsanjani A., Zhang Λ., Ellis M., Allam A., Channabasavaiah K., « Design an SOA solution using a reference architecture», Developer Works, Μάρτιος 2007, http://www.ibm.com/developerworks/library/ar-archtemp/

[26] Chappell D., «Introducing Windows Workflow Foundation», Microsoft Corporation, 2007

[27] CHIME, UCL Centre for Health Informatics & Multiprofessional Education,

http://www.ucl.ac.uk/chime/

[28] Chitnis M., Tiwari P., Ananthamurthy L., «Introducing Web Services Part 2:

Architecture», Νοέμβριος 2002, http://www.developer.com/services/article.php /1495091

[30] Dick R., Steen E., Detmer D., « The Computer-Based Patient Record», The national Academies Press, 1997, http://www.nap.edu/openbook.php?record_id =5306&page=179

[31] ebPML.org, «Web Service architecture», 2003, http://www.ebpml.org/ webservices.htm

[32] Erl T., «SOA Design paterns», http://www.soapatterns.org/

[33] Ferguson D., Stockton M., «SOA programming model for implementing Web Services», Developer Works, Ιούνιος 2005, http://www.ibm.com/developerworks /webservices /library/ws-soa-progmodel/

[34] Garets D., Davis M., «Electronic Medical Records vs. Electronic Health Records: Yes, There Is a Difference», HIMSS Analytics, Ιανουάριος 2006, http://www.himssanalytics.org/docs/WP_EMR_EHR.pdf

[35] http://www.seleniumhq.org/

[36] Johns M., «Health information management technology», American Health Information Management Association, http://library.ahima.org/xpedio/groups/public/documents/ahima/bok1_015899.pdf

[37] https://cukes.info/

[38] http://www.fitnesse.org/

[39] http://robotframework.org/

[40] Microsoft Health Vault, http://www.microsoft.com/en-us/healthvault/

[41] Microsoft HealthVault Future, XPLANE, http://etimpera.com/?page_id=21

[42] http://www.nunit.org/

[43] http://junit.org/

[44] http://www.slideshare.net/pekkaklarck/introduction-to-test-automation

[45] Psychology Wiki, «Electronic Health Record», http://psychology.wikia.com

/wiki/Electronic_Health_Record http://testobsessed.com/2007/03/testing-triangles-a-classic-exercise-updated-for-the-web/

[46] http://www.velocitypartners.net/blog/2014/01/28/agile-testing-the-agile-test-automation-pyramid/