ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Μελέτη Αξιολόγησης Χρήσης Προτύπων Σχεδίασης στις βιβλιοθήκες»

Της φοιτήτριας

Αλατζιά Όλγας

Αρ. Μητρώου: 06/3029

Επιβλέπων

Δεληγιάννης Ιγνάτιος, Καθηγητής

Θεσσαλονίκη 2012

ΠΡΟΛΟΓΟΣ

Η αντικειμενοστραφής σχεδίαση, είναι ένα σύγχρονο, εξελισσόμενο αντικείμενο της Μηχανικής Λογισμικού, που παρουσιάζει έντονη ερευνητική δραστηριότητα τα τελευταία χρόνια. Η νέα αυτή τάση, επηρεάζει τόσο την ανάλυση, όσο και τη σχεδίαση των συστημάτων λογισμικού, διευκολύνοντας την υλοποίησή τους και βελτιώνοντας ορισμένα χαρακτηριστικά τους, όπως η ταχύτητα και η συντηρησημότητά.

Τα πρότυπα σχεδίασης, είναι μηχανισμοί που παρέχουν λύσεις σε συνήθη σχεδιαστικά προβλήματα, που παρουσιάζονται κατά τις φάσεις της σχεδίασης, της ανάπτυξης ή της τροποποίησης του λογισμικού. Τα πρότυπα, βελτιώνουν τα ποιοτικά χαρακτηριστικά του συστήματος, παρέχοντας ευελιξία, προσαρμοστικότητα, ευκολία στη συντήρηση, δυνατότητα επαναχρησιμοποίησης συστατικών του συστήματος κ.α.

Η διεξαγωγή της παρούσας μελέτης αποδείχθηκε ιδιαίτερα χρήσιμη για την εξοικείωση του συγγραφέα με προηγμένα θέματα τεχνολογίας λογισμικού, τις μεθόδους εμπειρικής αξιολόγησης μιας μελέτης και τη συστηματική βιβλιογραφική ανασκόπηση ενός ερευνητικού πεδίου. Επιπλέον, τα αποτελέσματα της εργασίας θεωρούνται σημαντικά και ευρύτερου επιστημονικού ενδιαφέροντος, από τη στιγμή που αξιολογούν τη χρήση των προτύπων σχεδίασης στις βιβλιοθήκες, ζήτημα επίκαιρο στην κοινωνία ανάπτυξης λογισμικού.

ΠΕΡΙΛΗΨΗ

Ο όρος «πρότυπο σχεδίασης» έχει πολύ μεγάλη σημασία για τη μηχανική λογισμικού, και εξακολουθεί να αποτελεί ένα σημαντικό πεδίο έρευνας μέχρι σήμερα.

Η εργασία αυτή μελετά την επίδραση της χρήσης των προτύπων σχεδίασης στις βιβλιοθήκες λογισμικού, από δυο διαφορετικές σκοπιές, και παραθέτει δυο άμεσα συσχετιζόμενες αλλά ανεξάρτητες μελέτες.

Αρχικά διεξαγάγαμε μια συστηματική ανασκόπηση της βιβλιογραφίας, προκειμένου να διαπιστώσουμε και να καταγράψουμε την ερευνητική δραστηριότητα μέχρι σήμερα, στον τομέα των αντικειμενοστραφών προτύπων σχεδίασης. Για τις ανάγκες της ανασκόπησης, συλλέξαμε και μελετήσαμε άρθρα, που είχαν δημοσιευθεί σε κορυφαία περιοδικά, συνέδρια και συναντήσεις εργασίας (workshops). Αναλύοντας τα δεδομένα που συγκεντρώσαμε από τη βιβλιογραφία, με χρήση στατιστικών τεχνικών, προέκυψαν ενδιαφέροντα αποτελέσματα. Η μελέτη παρουσιάζεται αναλυτικά στο $2^{ο}$ κεφάλαιο.

Στη συνέχεια, παρουσιάζουμε μια εμπειρική μελέτη, σχετικά με τη χρήση προτύπων σχεδίασης στις βιβλιοθήκες λογισμικού. Για τη μελέτη αυτή συλλέξαμε 291 δημοφιλή λογισμικά (API και Standalone) ανοιχτού λογισμικού, ενός γνωστού αποθετηρίου κώδικα (sourceforge), που πληρούσαν ορισμένες προδιαγραφές. Με τη χρήση ενός εργαλείου ανίχνευσης προτύπων σχεδίασης από object αρχεία Java εξορύξαμε πληροφορίες σχετικά με τη χρήση προτύπων σχεδίασης στα λογισμικά αυτά. Τα δεδομένα που συγκεντρώθηκαν αναλύθηκαν με τη βοήθεια στατιστικών τεχνικών και προέκυψαν ενδιαφέροντα αποτελέσματα που παρουσιάζονται αναλυτικά στο κεφάλαιο 3.

ABSTRACT

The term «design pattern» is interesting field of software engineering and it present high research activity. This thesis examines the impact of the use of design patterns in software libraries from two different angles, and lists two directly associated but independent studies.

Firstly, we conducted a systematic literature review, to identify and document the state of the art on object-oriented design patterns. During the research process, we collected a satisfactory number of papers, published on top journals, conferences and workshops, on the field of software engineering. Through statistics, we analyzed our data and came up with interesting results. This study is presented in Chapter 2.

Then, we conducted an empirical study on the use of standards in the design software libraries. During this study, we collected the 291 most popular open source software (API and Standalone) found in a well-known repository (sourceforge). By using a mining tool for identifying design patterns from Java bytecode we collected information about the use of design patterns in these software's. The collected data were analyzed through statistic techniques and the results are presented in Chapter 3.

ΕΥΧΑΡΙΣΤΙΕΣ

Θεωρώ την εκπόνηση της συγκεκριμένης πτυχιακής εργασίας ως την ανώτερη δημιουργική στιγμή της φοιτητικής μου πορείας. Θα ήθελα λοιπόν να ευχαριστήσω θερμά, τον επιβλέποντα καθηγητή μου κ. Δεληγιάννη Ιγνάτιο, ο οποίος μου προσέφερε τη δυνατότητα αυτή. Ιδιαίτερα θα ήθελα να ευχαριστήσω τον εργαστηριακό συνεργάτη του τμήματος μας, κ. Αμπατζόγλου Απόστολο που από την πρώτη στιγμή της ανάθεσης, βρισκόταν δίπλα μου, σαν συνεργάτης και δάσκαλος. Επίσης θα ήθελα να τον ευχαριστήσω τον συμφοιτητή μου, κ. Γκορτζή Αντώνιο, ο οποίος παρόλο που υπηρετούσε την «Μαμά πατρίδα» βοήθησε ώστε να ολοκληρωθεί σύντομα η εργασία αυτή, καθώς και την διδα. Αρβανίτου Ελβίρα-Μαρία η οποία συνέδραμε καθ'όλη την διάρκεια της φοιτητικής μου πορείας.

Τέλος θα ήθελα να ευχαριστήσω την οικογένειά και τους φίλους μου, που έδειξαν κατανόηση στις δυσκολίες των τελευταίων εξαμήνων, δείχνοντας άπειρη κατανόηση και προθυμία να με στηρίξουν με οποιονδήποτε τρόπο.

Περιεχόμενα

Ευρετήριο σχημάτων

Ευρετήριο πινάκων

ΚΕΦΑΛΑΙΟ 1 – ΕΙΣΑΓΩΓΙΚΕΣ ΄ΕΝΝΟΙΕΣ

Στο πρώτο κεφάλαιο γίνεται ανασκόπηση και ανάλυση στους δυο από τους βασικούς όρους αυτής της εργασίας. Οι όροι αυτοί είναι το «ανοιχτό λογισμικό» καθώς η «Διεπαφή Προγραμματισμού Εφαρμογών (*APIs)*» από τους οποίους αντλούνται όλα τα δεδομένα προς ανάλυση και σύγκριση στη συγκεκριμένη πτυχιακή εργασία.

1.1 Ανοιχτό Λογισμικό

Το ανοιχτό λογισμικό αποτελεί ένα σύγχρονο μοντέλο ανάπτυξης, διάθεσης και χρήσης λογισμικού που κερδίζει συνεχώς έδαφος διεθνώς. Ανοιχτό είναι το λογισμικό που ο καθένας μπορεί ελεύθερα να χρησιμοποιεί, να αντιγράφει, να διανέμει και να τροποποιεί ανάλογα με τις ανάγκες του. Η απήχηση του ανοιχτού λογισμικού γίνεται περισσότερο αντιληπτή όταν διαπιστώνουμε πώς εταιρείες λογισμικού προσαρμόζουν το επιχειρηματικό τους μοντέλο στα νέα δεδομένα που αυτό δημιουργεί: το λογισμικό τείνει να αντιμετωπίζεται ως το όχημα παροχής υπηρεσιών και υποστήριξης λύσεων σύμφωνα με τις ανάγκες των χρηστών. Πρόκειται δηλαδή για ένα εναλλακτικό μοντέλο ανάπτυξης και χρήσης λογισμικού, το οποίο βασίζεται στην ελεύθερη διάθεση του πηγαίου κώδικα (σειρά εντολών που γράφονται σε γλώσσα προγραμματισμού υπολογιστών), το οποίο παρέχει τη δυνατότητα αλλαγών ή βελτιώσεων ώστε να καλύπτονται οι απαιτήσεις αυτού που το χρησιμοποιεί.

Σήμερα σε χώρες όπως η Γερμανία, η Γαλλία, η Ισπανία, Η Ιταλία και η Νορβηγία το Ελεύθερο λογισμικό / Λογισμικό Ανοικτού Κώδικα(ΕΛ/ΛΑΚ) αποτελεί ισότιμη επιλογή στους διαγωνισμούς του δημοσίου τομέα. Η απήχηση του ΕΛ/ΛΑΚ γίνεται περισσότερο αντιληπτή όταν εταιρείες λογισμικού όπως η IBM, η Hewlett Packard κλπ, προσαρμόζουν το επιχειρηματικό τους μοντέλο στα νέα δεδομένα που αυτό δημιουργεί, δηλαδή, το λογισμικό είναι πλέον ως μέσο για την παροχή υπηρεσιών και υποστήριξης λύσεων.

Μερικά από τα αντικειμενικά πλεονεκτήματα του ανοιχτού λογισμικού είναι:

α) χρησιμοποιεί ανοικτά πρότυπα στην κωδικοποίηση των δεδομένων διασφαλίζοντας την απρόσκοπτη πρόσβαση σε αυτά,

β) είναι ασφαλές, γιατί οποιοσδήποτε μπορεί να ελέγξει τις λειτουργίες όλων των υπο-προγραμμάτων,

γ) καλύτερη αξιοποίηση υλικού, δεν απαιτεί συνεχείς αναβαθμίσεις των υπολογιστών για κάθε νεότερη έκδοση του λογισμικού και

δ) διατίθεται ελεύθερα μέσα από το Διαδίκτυο με οδηγίες χρήσης και τεχνική τεκμηρίωση.

Το ΕΛ/ΛΑΚ έχει μια αλληλένδετη και αμφίδρομη σχέση με το Διαδίκτυο. Η ραγδαία εξάπλωση του ΕΛ/ΛΑΚ τα τελευταία δεκαπέντε χρόνια έχει στηριχθεί στην ευρεία χρήση του Διαδικτύου, αλλά και η ανάπτυξη του διαδικτύου, ειδικότερα των δικτύων νέας γενιάς, βασίζεται στο ελεύθερο λογισμικό. Όλα τα προϊόντα ΕΛ/ΛΑΚ αναπτύσσονται με διαφάνεια συνεργατικά από κοινότητες εθελοντών προγραμματιστών από όλο των κόσμο, σήμερα πάνω από 150.000 διαφορετικά προϊόντα ΕΛ/ΛΑΚ αναπτύσσονται και συντηρούνται από περίπου 2.000.000 προγραμματιστές. Τα προγράμματα διορθώνονται και εμπλουτίζονται συνεργατικά μέσω του διαδικτύου και έτσι οι βελτιωμένες εκδόσεις κυκλοφορούν ταχύτατα αντιμετωπίζοντας τα οποιαδήποτε προβλήματα παρουσιάζονται άμεσα.

Στην Ελλάδα πάνω από 4.000 προγραμματιστές και εξειδικευμένοι χρήστες συμμετέχουν σε κοινότητες[1]. Μέσα από εθελοντικές πρωτοβουλίες έχουν ελληνοποιηθεί δημοφιλή πακέτα λογισμικού, όπως Mozilla, OpenOffice.org, KDE, GNOME, κ.α. και έχουν δημιουργηθεί ελληνικοί οδηγοί χρήσης αυτών των πακέτων.

Το συνεργατικό μοντέλο ανάπτυξης διασφαλίζει την αποτελεσματική αξιοποίηση του χρόνου που αφιερώνει ο κάθε εθελοντής προγραμματιστής, ενώ καθημερινά εκατομμύρια ώρες εργασίας αφιερώνονται για τον σχεδιασμό, ανάπτυξη, βελτίωση και δοκιμή δεκάδων χιλιάδων έργων ανοιχτού λογισμικού, με αυτό τον τρόπο υλοποιείται το δικτυακό μοντέλο οργάνωσης της παραγωγής του ανοιχτού λογισμικού.

Η μεγάλη χρήση του ιντερνέτ σε όλο τον κόσμο διαμορφώνει πλέον νέες συνθήκες για το πως μοιραζόμαστε αυτά που δημιουργούμε ή αγοράζουμε. Σήμερα δεκάδες χιλιάδες αγαθά και προϊόντα όπως βιβλία, μουσική, βίντεο, άρθρα, φωτογραφίες διατίθενται και σε ψηφιακή μορφή. Τα ψηφιακά αγαθά έχουν την πολύ σημαντική ιδιότητα ότι είναι άυλα και έτσι όταν μοιραζόμαστε ένα αντίγραφο δεν στερούμαστε το πρωτότυπο.

[1] http://www.ellak.gr/,http://www.hellug.gr/
http://www.athenswireless.net/

Αυτή η νέα πραγματικότητα δημιουργεί νέες απαιτήσεις για την προστασία της πνευματικής ιδιοκτησίας, το copyright ρυθμίζει τα δικαιώματα των δημιουργών όπως υποδηλώνει και η ετυμολογία της λέξης copy–right, το δικαίωμα στην αντιγραφή. Το copy-right θεσμοθετήθηκε αρχικά για να διασφαλίσει τους συγγραφείς από την μη νόμιμη επανεκτύπωση των βιβλίων τους και στη συνέχεια επεκτάθηκε και σε άλλα έργα.

Το ιντερνέτ και οι τεχνολογίες ψηφιοποίησης οδήγησαν από πολύ νωρίς σε νέες μορφές αδειοδότησης του ψηφιακού περιεχομένου, το πρώτο παράδειγμα εναλλακτικής αδειοδότησης είναι το copyleft, το copyleft αξιοποιεί το θεσμικό πλαίσιο του copyright και το τροποποιεί έτσι ώστε ο δημιουργός να επιτρέπει την ελεύθερη αντιγραφή του έργου του. Η χρήση του διαδικτύου από εκατομμύρια χρήστες για την διακίνηση ψηφιακού περιεχόμενου συνέβαλε στο να επανεξετασθεί και ο τρόπος διάθεσης και του πολιτιστικού περιεχόμενου. Το 2001 ο Lawrence Lessig, μέσα από την εμπειρία μιας δικαστικής διαμάχης για την επέκταση της χρονικής περιόδου διαρκείας των πνευματικών δικαιωμάτων, σχεδίασε τις άδειες Creative Commons με στόχο να συμβάλει στο να αρθούν οι ελλείψεις του copyright που εξυπηρετούσε τους δημιουργούς την μη-ψηφιακή περίοδο.

Οι άδειες Creative Commons επιτρέπουν στους δημιουργούς να επιλέξουν τον τρόπο αδειοδότησης των έργων τους και τους χρήστες να χρησιμοποιούν χωρίς νομικά κωλύματα τα έργα που διατίθενται στο διαδίκτυο. Οι άδειες Creative Commons όπως και οι άδειες του ανοιχτού λογισμικού συμβάλλουν στην δημιουργία κοινών ψηφιακών αγαθών που διαθέτονται ελεύθερα μέσα από το διαδίκτυο και διαμορφώνουν τις συνθήκες για την ανάπτυξη νέων μορφών δημιουργικότητας, καινοτομίας, επιχειρηματικότητας καθώς και νέων μορφών συμμετοχής των πολιτών στα κοινά.

Ένας τρόπος ενίσχυσης του ανοιχτού χαρακτήρα του διαδικτύου είναι η δι άδοση του ανοιχτού λογισμικού και του ελεύθερου περιεχομένου. Με την αύξηση της χρήσης του ιντερνέτ από όλο και μεγαλύτερο αριθμό πολιτών – στην Ελλάδα σήμερα ένας στους τρεις πολίτες και δύο στους τρεις νέους κάτω των 35 χρησιμοποιούν το διαδίκτυο – αυξάνει η ζήτηση για ελεύθερα διαθέσιμη πληροφορία από τον δημόσιο και ιδιωτικό τομέα. Η αύξηση της ποιότητας και της ποσότητας της πληροφορίας συμβάλλει και σε αλλαγές που συνδράμουν στην ενδυνάμωση του πολίτη και οδηγούν στη διαμόρφωση νέων σχέσεων με το κράτος και την αγορά, ενώ οι συνεργατικές εφαρμογές και το γρήγορο ιντερνέτ συμβάλλουν

στο να αναπτυχθούν εναλλακτικά και οριζόντια μοντέλα οργάνωσης που καθιστούν προς το παρόν ανέφικτο τον απόλυτο έλεγχο του διαδικτύου από την αγορά η/και το κράτος.

1.2 Διεπαφή Προγραμματισμού Εφαρμογών (*APIs)*

Για να καταλάβουμε τι σημαίνει Διεπαφή Προγραμματισμού Εφαρμογών (αγγλ. API, από το Application Programming Interface) ας μείνουμε στην λέξη διεπαφή. Διεπαφή είναι το κοινό σύνορο μεταξύ δύο χωριστών συστημάτων. Είναι το μέσο μέσω του οποίου αυτά τα δύο συστήματα επικοινωνούν. Το API λοιπόν είναι ένας πηγαίος κώδικας που προορίζεται να χρησιμοποιηθεί ως διεπαφή από στοιχεία λογισμικού για την μεταξύ τους επικοινωνία. Υλοποιείται από εταιρείες για μία “εφαρμογή”, και επιτρέπει σε άλλες “εφαρμογές” να επικοινωνούν με αυτήν.

Επειδή τα API είναι γενικά αόρατα στους τελικούς χρήστες, είναι δύσκολο να τα κατανοήσουμε. Ας δούμε μερικά παραδείγματα ώστε να καταλάβουμε που, πως και πόσο συχνά τα χρησιμοποιούμε. Η αντιγραφή-επικόλληση από το “Σημειωματάριο” των Windows στο Office Word ή στο Google Chrome και αντίστροφα γίνεται δυνατή μέσω των APIs. Όλα τα προγράμματα για να εγκατασταθούν στο λειτουργικό των Windows πρέπει να είναι συμβατά με το API της έκδοσης του λειτουργικού που υπάρχει εγκατεστημένο (π.χ. Vista). Το API για μία γλώσσα προγραμματισμού, όπως η C, συνήθως αποτελείται από συναρτήσεις. Για αυτό το API περιλαμβάνει συνήθως μια περιγραφή όλων των λειτουργιών/ρουτινών που παρέχει.

Οι εταιρείες που απελευθερώνουν τα API τους συχνά το κάνουν ως μέρος ενός μεγαλύτερου πακέτου ανάπτυξης λογισμικού (π.χ. JDK, SDK) που περιλαμβάνει το API, εργαλεία προγραμματισμού και άλλα εκπαιδευτικά έγγραφα για να κάνει τη δουλειά του κατασκευαστή προγραμμάτων πιο εύκολη. Παραδείγματος χάριν, ένα πρόγραμμα σε Java το οποίο εκτυπώνει ένα έγγραφο στον εκτυπωτή χρησιμοποιεί το API του εκτυπωτή. Επίσης, μία εφαρμογή σε Android για να χειριστεί λειτουργίες συστήματος (π.χ. GPS, Bluetooth, Κάμερα, αποθήκευση εικόνων στην κάρτα κλπ) χρησιμοποιεί το API που έχει δημοσιεύσει η Android. Ουσιαστικά, η πλατφόρμα Android προσφέρει το API της έτσι ώστε οι εφαρμογές να μπορούν να αλληλεπιδρούν με το χαμηλότερο σύστημα Android.

Στο πλαίσιο της ανάπτυξης ιστοσελίδων υπάρχουν τα web APIs, τα οποία είναι ένα σύνολο οδηγιών προγραμματισμού και προτύπων για την πρόσβαση σε

μια Web-based εφαρμογή λογισμικού ή εργαλείο Web. Για παράδειγμα, η ενσωμάτωση ενός πλαισίου αναζήτησης, σε μία ιστοσελίδα, που θα επιστρέφει αποτελέσματα από τους εξυπηρετητές της Google, πρέπει να επικοινωνεί με το Google Search API.

Το πρώτο web API δημιουργήθηκε το 2000 και ήταν το eBay API. Βοηθούσε τους πωλητές να αποθηκεύσουν και να διαγράψουν από μία λίστα τα προϊόντα, και πολλά άλλα χαρακτηριστικά για τον χρήστη. Με το API, άλλες εταιρείες μπορούν να χτίζουν εργαλεία για να βοηθούν να αυτοματοποιούνται ή να βελτιώνονται τα ίδια τα χαρακτηριστικά της σελίδας.

Επίσης, στον τομέα της κοινωνικής δικτύωσης. Όλες οι μεγάλες ιστοσελίδες κοινωνικής δικτύωσης όπως το Twitter και το Facebook έχουν ελευθερώσει τις Διεπαφές Προγραμματισμού των Εφαρμογών τους. Μέσω αυτών τα μέλη τους έχουν την δυνατότητα να έχουν πρόσβαση μέσω κινητών συσκευών, και ένα μέρος της επιτυχίας τους οφείλεται στην ελεύθερη πρόσβαση μέσω διαφορετικών εφαρμογών λόγω της απελευθέρωσης των APIs.

Πως λειτουργούν τα APIs:

Ένας από τους βασικούς σκοπούς μίας διεπαφής είναι να ορίζει και να διατυπώνει το σύνολο των λειτουργιών-υπηρεσιών που μπορεί να παρέχει μια βιβλιοθήκη ή ένα λειτουργικό σύστημα σε άλλα συστήματα, χωρίς να επιτρέπει πρόσβαση στον κώδικα που υλοποιεί αυτές τις υπηρεσίες. Παραδείγματος χάριν, το λειτουργικό σύστημα Windows έχει τη δική του Διεπαφή Προγραμματισμού Εφαρμογών (κλήσεις συστήματος), η μορφή (format) της οποίας διατίθεται από την κατασκευάστρια εταιρεία Microsoft. Αυτή η διεπαφή περιγράφει τους τρόπους αξιοποίησης, από προγράμματα χρήστη, του συνόλου των υπηρεσιών που παρέχει το λειτουργικό, χωρίς όμως να χρειάζεται η γνώση του χαμηλότερου κώδικα.

Ένα API είναι μία λογισμικό-προς-λογισμικό διεπαφή και όχι μία διεπαφή χρήστη. Με τα API, οι εφαρμογές μιλούν μεταξύ τους χωρίς τη γνώση ή την παρέμβαση κάποιου χρήστη. Στη περίπτωση της αγοράς εισιτηρίων online και εισαγωγής των στοιχείων της πιστωτικής κάρτας, η ιστοσελίδα χρησιμοποιεί ένα API για να στείλει τα στοιχεία της πιστωτικής κάρτας σε μια απομακρυσμένη εφαρμογή που ελέγχει αν οι πληροφορίες είναι σωστές ή όχι. Μόλις επιβεβαιωθεί η πληρωμή, η απομακρυσμένη εφαρμογή στέλνει μια απάντηση πίσω στη ιστοσελίδα λέγοντας ότι είναι OK για την έκδοση των εισιτηρίων. Ο χρήστης μπορεί να δει μόνο

ένα interface - την ιστοσελίδα - αλλά πίσω από τις σκηνές, πολλές εφαρμογές εργάζονται από κοινού με τη χρήση APIs. Αυτό το είδος της ενσωμάτωσης ονομάζεται απρόσκοπτη (seamless), μιας και ο χρήστης δεν καταλαβαίνει πότε οι λειτουργίες του λογισμικού χειρίζονται από μία εφαρμογή ή από την άλλη.

Ένα API μπορεί να είναι:

- language-dependent (εξαρτώμενο από τη γλώσσα), που σημαίνει ότι είναι διαθέσιμο μόνο με χρήση της σύνταξης και των στοιχείων μιας συγκεκριμένης προγραμματιστικής γλώσσας, με την οποία το API έχει υλοποιηθεί.
- language-independent (ανεξάρτητο από τη γλώσσα), οι εντολές μπορεί να είναι γραμμένες σε διάφορες γλώσσες προγραμματισμού. Αυτό είναι ένα επιθυμητό χαρακτηριστικό για ένα service-oriented API το οποίο δεν είναι δεσμευμένο με μια συγκεκριμένη διαδικασία ή σύστημα και μπορεί να παρέχεται ως απομακρυσμένες κλήσεις διαδικασίας (remote procedure calls) ή ως web services.

ΚΕΦΑΛΑΙΟ 2 – ΠΡΟΤΥΠΑ ΣΧΕΔΙΑΣΗΣ

Κατά τα τέλη της δεκαετίας του '70 ένας αρχιτέκτονας ονόματα Κρίστοφερ Αλεξάντερ επιχείρησε να βρει και να καταγράψει αποδεδειγμένα ποιοτικούς σχεδιασμούς στον τομέα των κατασκευών. Έτσι μελέτησε πολλές διαφορετικές κατασκευές που εξυπηρετούσαν τον ίδιο σκοπό και προσπάθησε να ανακαλύψει κοινά στοιχεία, τα οποία κατηγοριοποίησε σε σχεδιαστικά πρότυπα (design patterns). Το 1987 η ιδέα της εύρεσης σχεδιαστικών προτύπων εφαρμόστηκε για πρώτη φορά στη μηχανική λογισμικού και μέχρι τα μέσα της δεκαετίας του '90 η εν λόγω έννοια είχε καθιερωθεί και εξαπλωθεί, στραμμένη πλέον στον κόσμο της αντικειμενοστρέφειας.

Ένα πρότυπο σχεδίασης ορίζεται ως μία αποδεδειγμένα καλή λύση που έχει εφαρμοστεί με επιτυχία στην επίλυση ενός επαναλαμβανόμενου προβλήματος σχεδίασης συστημάτων λογισμικού [Gamma] . Είναι μια περιγραφή για το πώς να λύσει ένα πρόβλημα που μπορεί να χρησιμοποιηθεί σε πολλές διαφορετικές καταστάσεις. Ένα πρότυπο σχεδίασης εξετάζει αφαιρετικά, και αναγνωρίζει τις κρίσιμες πλευρές μιας κοινής δομής σχεδιασμού που είναι χρήσιμη για την δημιουργία ενός επαναχρησιμοποιούμενου σχεδίου. Το πρότυπο σχεδίου αναγνωρίζει τις συμμετέχουσες κλάσεις και στιγμιότυπα, τους ρόλους και τις συνεργασίες , καθώς και την κατανομή ευθυνών(Gamma κ.α. 1995).

Τα πρότυπα σχεδίασης ορίζονται τόσο σε επίπεδο μακροσκοπικής σχεδίασης όσο και σε επίπεδο υλοποίησης, ενώ με τη χρήση τους ένας προγραμματιστής αντικαθιστά πρακτικώς μεγάλα τμήματα του κώδικα του με μαύρα κουτιά. Πρόκειται για αφαιρέσεις υψηλού επιπέδου που αποτελούν πλήρη υποσυστήματα, κατάλληλα ρυθμισμένα για την επίλυση συγκεκριμένων προβλημάτων και έτοιμα για χρήση. Είναι μια περιγραφή για το πώς να λυθεί ένα πρόβλημα που μπορεί να χρησιμοποιηθεί σε πολλές διαφορετικές καταστάσεις. Τα αντικειμενοστρεφή σχεδιαστικά πρότυπα παρουσιάζουν τις σχέσεις και τις αλληλεπιδράσεις μεταξύ των κλάσεων ή των αντικειμένων χωρίς διευκρίνιση των τελικών κλάσεων ή των αντικειμένων εφαρμογής που περιλαμβάνονται.

Το όνομα κάθε προτύπου διευκολύνει την επικοινωνία μεταξύ προγραμματιστών καθώς και την εύκολη αναφορά σε κοινά είδη προβλημάτων. Το πρόβλημα σε κάθε πρότυπο προσδιορίζει ένα γενικότερο πλαίσιο όπου υπό κανονική αντιμετώπιση (χωρίς τη χρήση προτύπων) θα προέκυπταν ανεπιθύμητες συνέπειες αναφορικά με την λειτουργία, τον έλεγχο και τη συντήρηση του λογισμικού. Κάθε πρότυπο επιτρέπει να αλλάξει μια όψη από την αρχιτεκτονική δομή χωρίς να επηρεάζει τις υπόλοιπες.

Έχουν οριστεί διάφορες κατηγορίες προτύπων, για διαφορετικά προβλήματα και κάθε κατηγορία περιλαμβάνει πολλαπλά στοιχεία. Έτσι υπάρχουν συμπεριφορικά πρότυπα, κατασκευαστικά πρότυπα, και δομικά πρότυπα.

- Συμπεριφορικά Πρότυπα (Behavioural Patterns) : Τα συμπεριφορικά πρότυπα αφορούν τον καταμερισμό αρμοδιοτήτων σε διάφορες κλάσεις και τον ορισμό του τρόπου επικοινωνίας μεταξύ των αντικειμένων τους κατά τον χρόνο εκτέλεσης. Σε αντίθεση με τα δομικά πρότυπα, τα συμπεριφορικά βρίσκουν εφαρμογή στον αρχικό σχεδιασμό μίας ιεραρχίας κλάσεων και όχι στην εκ των υστέρων επέκταση κάποιας υπάρχουσας ιεραρχίας. Τέτοια πρότυπα είναι το Strategy, το State, το Observer, το Template Method και το Visitor.

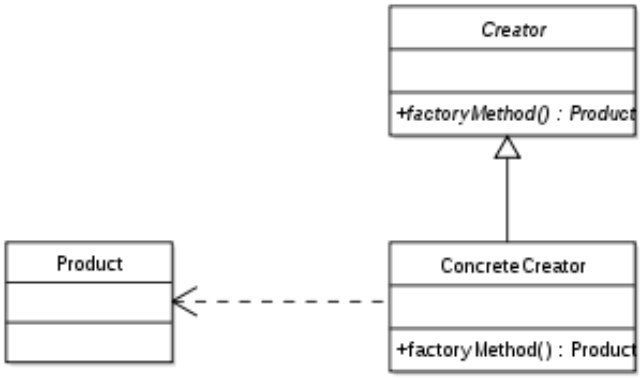
- Κατασκευαστικά Πρότυπα (Creational Patterns) : Τα κατασκευαστικά πρότυπα αφορούν τυποποιημένους τρόπους δυναμικής κατασκευής αντικειμένων κατά τον χρόνο εκτέλεσης. Απώτερος στόχος τους είναι η ανεξαρτητοποίηση του κώδικα που χρησιμοποιεί κάποια αντικείμενα από τις κλάσεις που ορίζουν τα αντικείμενα αυτά και τον τρόπο που κατασκευάζονται στη μνήμη, σύμφωνα με την αρχή ανοιχτότητας-κλειστότητας για ορθή αντικειμενοστρεφή σχεδίαση. Τέτοια πρότυπα είναι το Factory Method, το Singleton και το Prototype.

- Δομικά Πρότυπα (Structural Patterns) : Τα δομικά πρότυπα αφορούν τυποποιημένους τρόπους δυναμικής κατασκευής σύνθετων αντικειμένων τα οποία χρησιμοποιούν υπάρχουσες ιεραρχίες κλάσεων. Κρατάνε τις συζεύξεις χαμηλές, τις κλάσεις αμετάβλητες, και προσφέρει εναλλακτικές λύσεις στην κληρονομικότητα. Τέτοια πρότυπα είναι το Composite, το Adapter και το Decorator Pattern.

2.1 Μέθοδος Εργοστάσιο (Factory Method)

Το πρότυπο σχεδίασης "Μέθοδος Εργοστάσιο" συγκεντρώνει σε μια κατάλληλη κλάση όλη τη λειτουργικότητα κατασκευής στιγμιότυπων ενός συνόλου παραγόμενων υποκλάσεων που κληρονομούν κάποια κοινή υπερκλάση ή υλοποιούν την ίδια διασύνδεση. Ανήκει στην κατηγορία των κατασκευαστικών προτύπων.

Χρησιμοποιείται όταν μία κλάση δεν γνωρίζει τον ακριβή τύπο των αντικειμένων που πρέπει να δημιουργήσει όπως επίσης όταν μία κλάση θέλει οι παραγόμενές της κλάσεις να ορίζουν τον τύπο των αντικειμένων που θα δημιουργούνται. (Gamma et al., 1995).



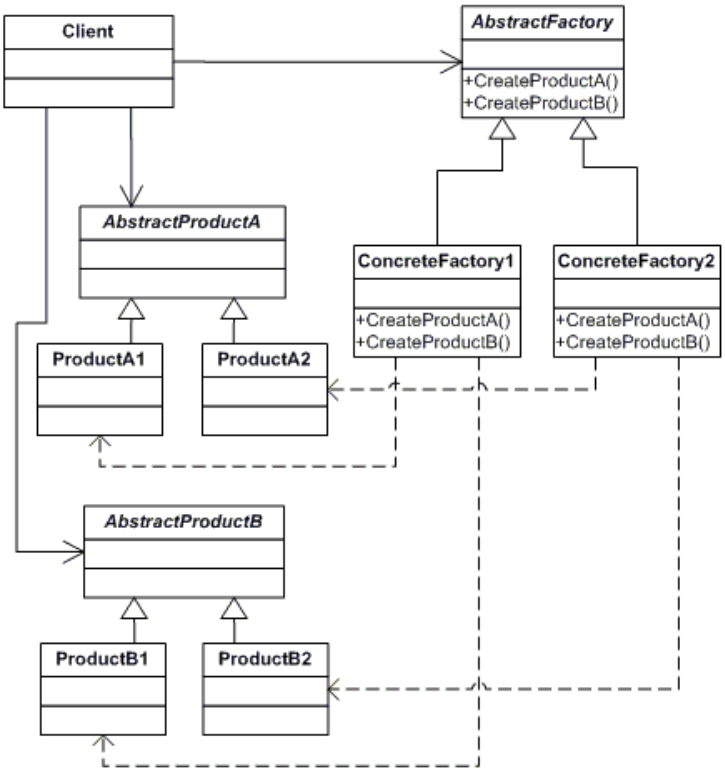
Σχήμα 1. Διάγραμμα κλάσεων προτύπου Factory

2.2 Αφηρημένο εργοστάσιο (Abstract Factory)

Σκοπός του προτύπου σχεδίασης "Αφηρημένο Εργοστάσιο" είναι η παροχή μιας διασύνδεσης για τη δημιουργία οικογενειών, συσχετιζόμενων ή εξαρτημένων αντικειμένων, χωρίς να προσδιορίζεται η υλοποίηση των συγκεκριμένων κλάσεών τους. Ανήκει στην κατηγορία των κατασκευαστικών προτύπων και ως εκ τούτου επιτρέπει τη συγγραφή μεθόδων που δημιουργούν νέα αντικείμενα, χωρίς την άμεση χρήση ιδιωμάτων (π.χ. τελεστής new), όπως συμβαίνει στις

αντικειμενοστραφείς γλώσσες προγραμματισμού. Το γεγονός αυτό επιτρέπει την ανάπτυξη μεθόδων που παράγουν ομάδες διαφορετικών αντικειμένων καθώς και την επέκτασή τους για νέα αντικείμενα χωρίς την τροποποίηση του κώδικα των μεθόδων.

Το πρότυπο Factory χρησιμοποιείται για την αποφυγή εξάρτησης από συγκεκριμένες κλάσεις όταν απαιτείται η δημιουργία αντικειμένων καθώς και για την ομαδοποίηση μεθόδων που δημιουργούν συσχετιζόμενα αντικείμενα σε μια αφηρημένη κλάση. (Chatzigeorgiou, 2005).

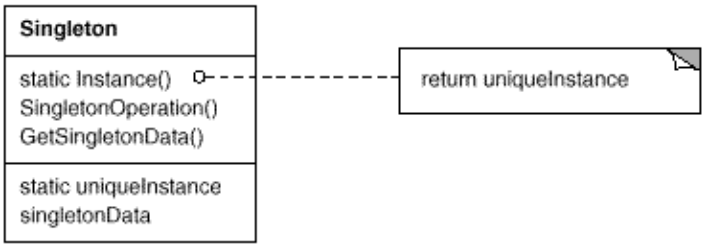


Σχήμα 2. Διάγραμμα κλάσεων προτύπου Abstract Factory

2.3 Μοναδιαίο (Singleton)

Το πρότυπο σχεδίασης "Μοναδιαίο" εξασφαλίζει ότι μια κλάση θα έχει μόνο ένα στιγμιότυπο και παρέχει ένα καθολικό σημείο πρόσβασης σε αυτό. Ανήκει στην κατηγορία των κατασκευαστικών προτύπων.

Χρησιμοποιείται όταν σε κάποιο σύστημα λογισμικού υπάρχει η απαίτηση από μια κλάση να δημιουργείται ένα και μόνο αντικείμενο. (Chatzigeorgiou, 2005).

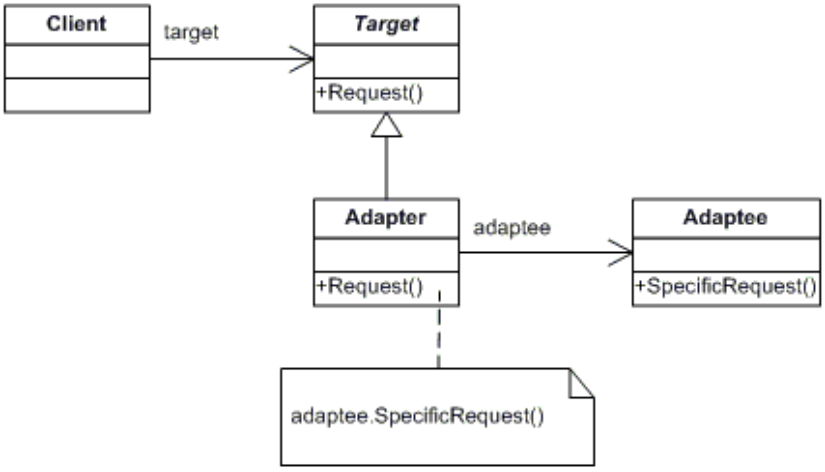


Σχήμα 3. Διάγραμμα κλάσεων προτύπου Singleton

2.4 Προσαρμογέας (Adapter)

Το πρότυπο σχεδίασης "Προσαρμογέας" έχει ως στόχο την μετατροπή της διασύνδεσης μιας κλάσης σε μια άλλη που αναμένει το πρόγραμμα πελάτης. Ο προσαρμογέας επιτρέπει τη συνεργασία κλάσεων, η οποία σε διαφορετική περίπτωση θα ήταν αδύνατη λόγω ασύμβατων διασυνδέσεων. Ανήκει στην κατηγορία των δομικών προτύπων (structural).

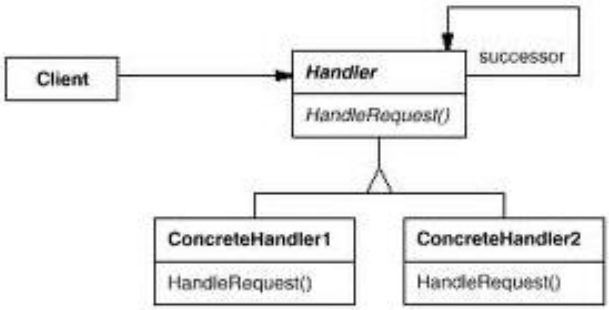
Χρησιμοποιείται σε περιπτώσεις που η διασύνδεση μιας κλάσης που μας ενδιαφέρει δεν συμβαδίζει με τις ανάγκες μας. Ένας προσαρμογέας κλάσης (class adapter) χρησιμοποιεί πολλαπλή κληρονομικότητα για να προσαρμόσει μια διασύνδεση σε μια άλλη. Ενώ ένας προσαρμογέας αντικειμένου (object adapter) βασίζεται στη σύνθεση αντικειμένων και στη διαβίβαση μηνυμάτων. (Chatzigeorgiou, 2005).



Σχήμα 4. Διάγραμμα κλάσεων προτύπου Adapter

2.5 Αλυσίδα Ευθύνης (Chain of Responsibility)

Το πρότυπο σχεδίασης "Αλυσίδα Ευθύνης" προσφέρει την δυνατότητα να αποφύγουμε την άμεση σύζευξη μεταξύ του αποστολέα μιας αίτησης και του παραλήπτη της, ορίζοντας περισσότερα του ενός αντικείμενα που μπορούν να διαχειριστούν την αίτηση. Δημιουργεί δηλαδή μια αλυσίδα μεταξύ των εμπλεκόμενων αντικειμένων και προωθεί μία αίτηση έως ότου ο εξουσιοδοτημένος παραλήπτης να την επεξεργαστεί. Ανήκει στην κατηγορία των συμπεριφορικών προτύπων. (Gamma et al., 1995).

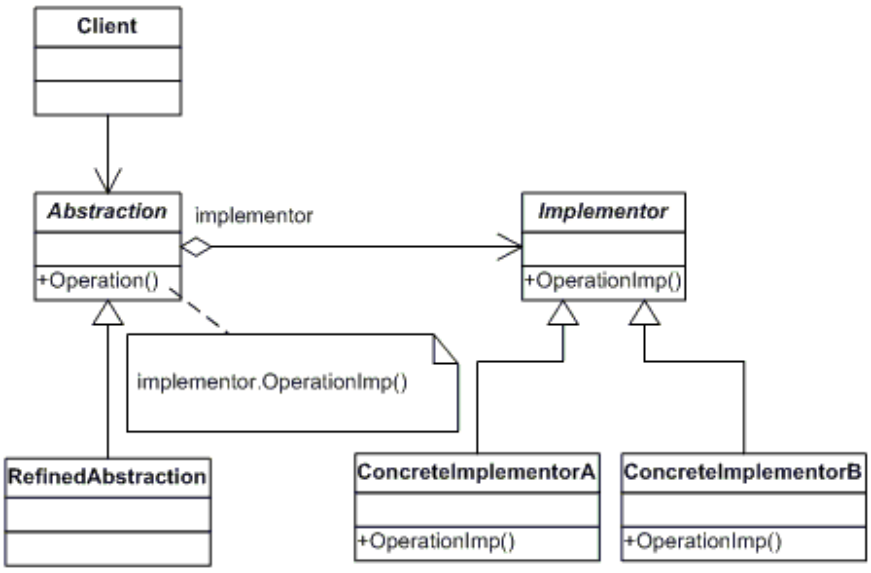


Σχήμα 5. Διάγραμμα κλάσεων προτύπου Chain of Responsibility

2.6 Γέφυρα (Bridge)

Το πρότυπο σχεδίασης "Γέφυρα" έχει ως στόχο την αποσύνδεση μιας αφαίρεσης από την υλοποίησή της, ώστε να μπορούν να μεταβάλλονται ανεξάρτητα. Όταν μια αφαίρεση μπορεί να έχει περισσότερες από μία υλοποιήσεις, ο συνήθης τρόπος οργάνωσης είναι με τη χρήση κληρονομικότητας. Με τον όρο αφαίρεση νοείται μια αφηρημένη κλάση που ορίζει μια διασύνδεση, ενώ υλοποιήσεις είναι οι συγκεκριμένες παράγωγες κλάσεις οι οποίες υλοποιούν τις μεθόδους της αφηρημένης κλάσης. Ανήκει στην κατηγορία των δομικών προτύπων (structural).

Μια "Γέφυρα" χρησιμοποιείται όταν μια έννοια (αφαίρεση) μπορεί να σχετίζεται με διάφορες υλοποιήσεις οι οποίες μπορεί να τροποποιούνται κατά τη διάρκεια της εκτέλεσης. Επίσης χρησιμοποιείται όταν οι αφαιρέσεις και οι υλοποιήσεις τους θα πρέπει να είναι επεκτάσιμες μέσω κληρονομικότητας. (Chatzigeorgiou, 2005).

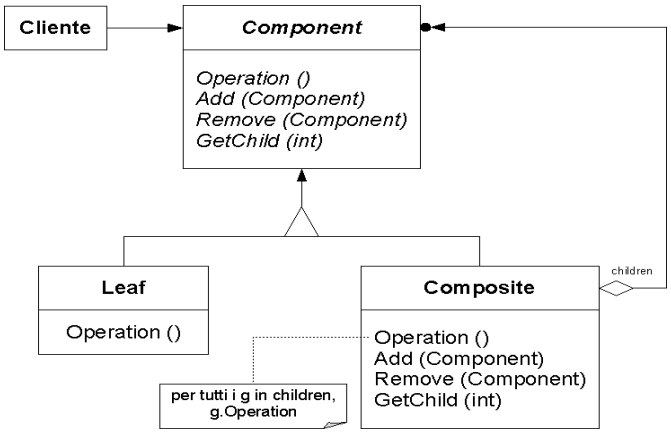


Σχήμα 6. Διάγραμμα κλάσεων προτύπου Bridge

2.7 Σύνθετο (Composite)

Το πρότυπο σχεδίασης "Σύνθετο" επιτρέπει τη σύνθεση αντικειμένων σε δενδροειδείς δομές για την αναπαράσταση ιεραρχιών τμήματος-όλου. Το πρότυπο σχεδίασης "Σύνθετο" επιτρέπει στα προγράμματα πελάτες να διαχειρίζονται με ενιαίο τρόπο τόσο τα ανεξάρτητα αντικείμενα όσο και τις συνθέσεις αντικειμένων. Ανήκει στην κατηγορία των δομικών προτύπων (structural).

Χρησιμοποιείται όταν θέλουμε τα προγράμματα πελάτες να χειρίζονται μεμονωμένα αντικείμενα και σύνθετα αντικείμενα με τον ίδιο τρόπο. (Chatzigeorgiou, 2005).



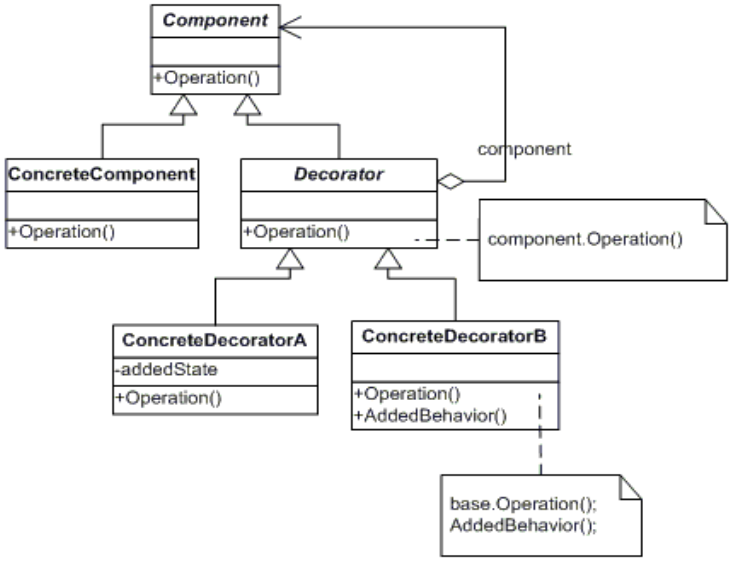
Σχήμα 7. Διάγραμμα κλάσεων προτύπου Composite

2.8 Διακοσμητής (Decorator)

Το πρότυπο "διακοσμητής" προσθέτει λειτουργίες σε κάποιο αντικείμενο δυναμικά. Οι "διακοσμητές" προσφέρουν μια ευέλικτη εναλλακτική λύση για την επέκταση των λειτουργιών μιας κλάσης αντί της μεθόδου δημιουργίας υποκλάσεων. Ανήκει στην κατηγορία των δομικών προτύπων (structural).

Εάν για παράδειγμα θέλουμε να εφαρμόσουμε μια συγκεκριμένη λειτουργία σε ένα αντικείμενό μας δεν χρειάζεται να πειράξουμε ολόκληρη την κλάση "διασύνδεσή" του αλλά αρκεί να επισυνάψουμε το αντικείμενό μας σε ένα άλλο "αντικείμενο διακοσμητής" που θα παρέχει την επιθυμητή λειτουργία. (Gamma et al., 1995).

Επίσης προσφέρει την δυνατότητα προσθήκης λειτουργιών σε κλάσης και που αποκρύπτονται οι λειτουργίες τους και δεν επιτρέπεται η επέκτασή τους. (Gamma et al., 1995).

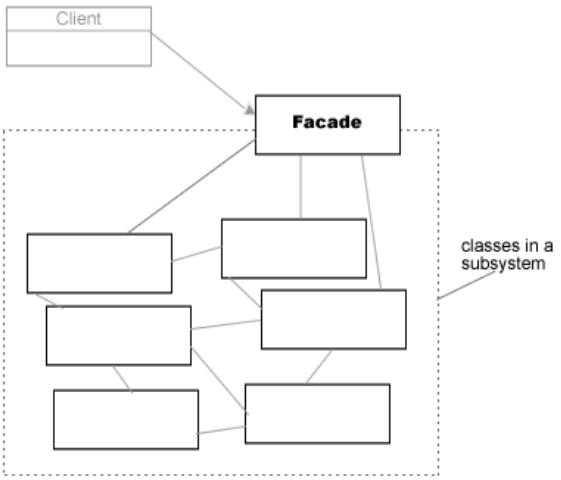


Σχήμα 8. Διάγραμμα κλάσεων προτύπου Decorator

2.9 Πρόσοψη (Facade)

Το πρότυπο σχεδίασης "Πρόσοψη" παρέχει μία ενιαία διασύνδεση για ένα σύνολο από διασυνδέσεις ενός υποσυστήματος. Προσδιορίζει δηλαδή μία υψηλού-επιπέδου διασύνδεση που διευκολύνει την χρήση ενός υποσυστήματος. Ανήκει στην κατηγορία των δομικών προτύπων.

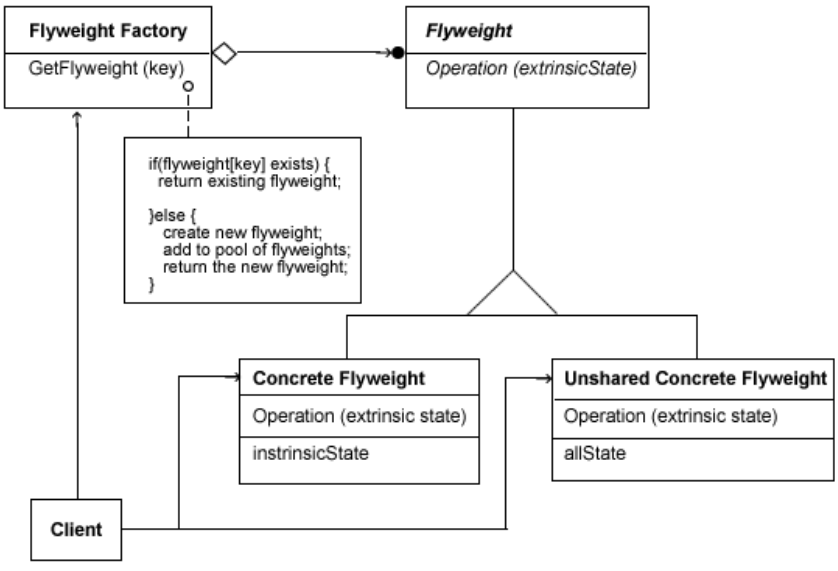
Χρησιμοποιείται όταν έχουμε μεγάλη σύζευξη μεταξύ των πελατών και των υλοποιήσεων κάποιων διασυνδέσεων με σκοπό να μειώσει την σύζευξη αυτή αλλά και να προσφέρει και μία ανεξαρτητοποίηση στο υποσύστημα. (Gamma et al., 1995).



Σχήμα 9. Διάγραμμα κλάσεων προτύπου Facade

2.10 Ελαφρού Τύπου (Flyweight)

Το πρότυπο σχεδίασης "Flyweight" προτείνει την κοινή χρήση κάποιων αντικειμένων ώστε να είναι αποδοτικές και διαχειρίσιμες καταστάσεις στις οποίες υπάρχει πολύ μεγάλος αριθμός αντικειμένων. Ένα flyweight δηλαδή, είναι ένα αντικείμενο το οποίο μπορεί να χρησιμοποιηθεί ταυτόχρονα από πολλές συλλογές αντικειμένων. Ανήκει στην κατηγορία των δομικών προτύπων. (Gamma et al., 1995).

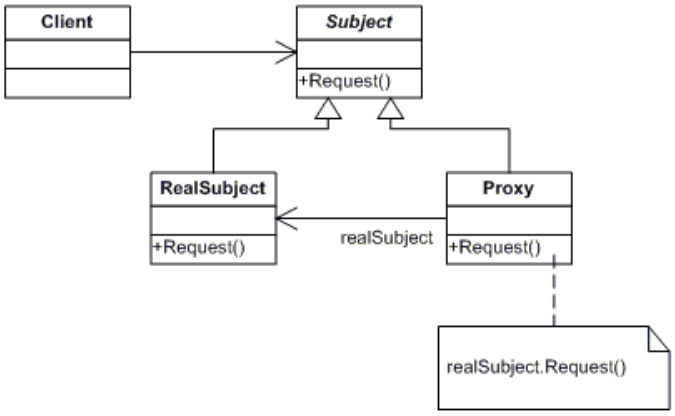


Σχήμα 10. Διάγραμμα κλάσεων προτύπου Flyweight

2.11 Πληρεξούσιο (Proxy)

Το πρότυπο σχεδίασης "Πληρεξούσιο" λειτουργεί ως υποκατάστατο ή ως ένας τρόπος κράτησης της θέσης ενός άλλου αντικειμένου. Ανήκει στην κατηγορία των δομικών προτύπων.

Μπορεί να χρησιμοποιηθεί με πολλούς τρόπους, είτε λειτουργώντας τοπικά ως αντιπρόσωπος ενός αντικειμένου, είτε αναπαριστώντας ένα μεγάλο αντικείμενο που πρέπει να φορτωθεί εφόσον ζητηθεί, είτε προστατεύοντας την πρόσβαση προς ένα ευαίσθητο αντικείμενο. Τα πρότυπα Proxy παρέχουν ένα επίπεδο ανακατεύθυνσης σε συγκεκριμένες ιδιότητες των αντικειμένων. Έτσι μπορούν να απαγορέψουν, να ενισχύσουν ή να αλλάξουν αυτές τις ιδιότητες. (Gamma et al., 1995).

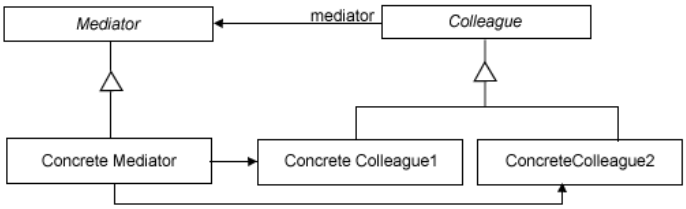


Σχήμα 11.Διάγραμμα κλάσεων προτύπου Proxy

2.12 Μεσολαβητής (Mediator)

Το πρότυπο σχεδίασης "Μεσολαβητής" προσδιορίζει ένα αντικείμενο που ενσωματώνει ένα σύνολο από άλλα αντικείμενα και ορίζει το πως αυτά θα επικοινωνούν μεταξύ τους. Το πρότυπο αυτό μειώνει την σύζευξη με το να μην επιτρέπει την άμεση επικοινωνία μεταξύ δύο αντικειμένων και επιτρέποντας στον

χρήστη να ορίσει τον τρόπο επικοινωνίας. Ανήκει στη κατηγορία των συμπεριφορικών προτύπων. (Gamma et al., 1995).

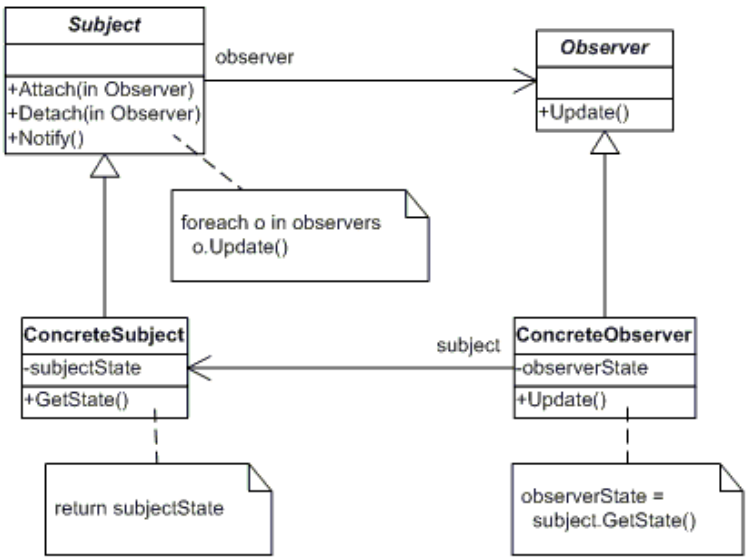


Σχήμα 12. Διάγραμμα κλάσεων προτύπου Mediator

2.13 Παρατηρητής (Observer)

Το πρότυπο σχεδίασης "Παρατηρητής" είναι επίσης γνωστό ως πρότυπο "Δημοσίευση-Εγγραφή" (Publish-Subscribe). Ορίζει μια σχέση εξάρτησης ένα-προς-πολλά μεταξύ αντικειμένων έτσι ώστε όταν μεταβάλλεται η κατάσταση ενός αντικειμένου, όλα τα εξαρτώμενα αντικείμενα να ενημερώνονται και τροποποιούνται αυτόματα. Ανήκει στην κατηγορία των συμπεριφορικών προτύπων.

Το πρότυπο Observer χρησιμοποιείται όταν η αλλαγή της κατάστασης ενός αντικειμένου (υποκείμενο) απαιτεί την ειδοποίηση άλλων αντικειμένων (παρατηρητών) , χωρίς το υποκείμενο να πρέπει να κάνει καμία υπόθεση για το ποια είναι τα αντικείμενα-παρατηρητές. Τέλος αξίζει να σημειωθεί, ότι η λίστα των παρατηρητών μπορεί να αλλάζει κατά τη διάρκεια εκτέλεσης του προγράμματος (Chatzigeorgiou, 2005).

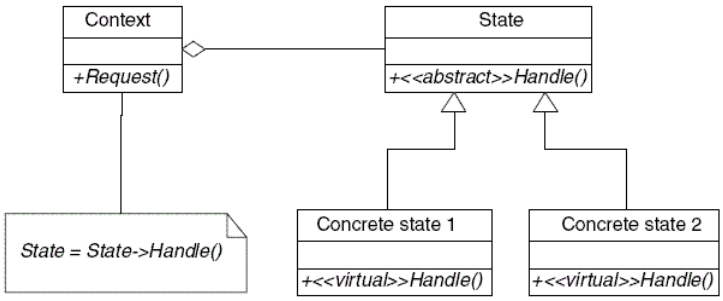


Σχήμα 13. Διάγραμμα κλάσεων προτύπου Observer

2.14 Κατάσταση (State)

Το πρότυπο σχεδίασης "Κατάσταση" ενθυλακώνει την κατάσταση ενός αντικειμένου, ώστε να μπορεί να αλλάξει τη συμπεριφορά του, όταν αλλάξει η εσωτερική κατάσταση του αντικειμένου. Ανήκει στην κατηγορία των συμπεριφορικών προτύπων.

Το πρότυπο State δίνει τη δυνατότητα σε ένα αντικείμενο να συμπεριφέρεται σαν να αλλάζει η κλάση του, κάτι που στις περισσότερες αντικειμενοστραφείς γλώσσες είναι αδύνατο. Στο πρότυπο, η κλάση πελάτης, περιέχει μια αφηρημένη κλάση, η οποία όμως δεν αντιπροσωπεύει μια στρατηγική αλλά μια κατάσταση. Οι παράγωγες κλάσεις υλοποιούν τις διάφορες καταστάσεις και κατά συνέπεια, η κλάση πελάτης, μπορεί να εναλλάξει την κατάστασή της αλλάζοντας την τιμή του δείκτη αναφοράς προς την επιθυμητή περιεχόμενη κατάσταση (Gamma et al., 1995).

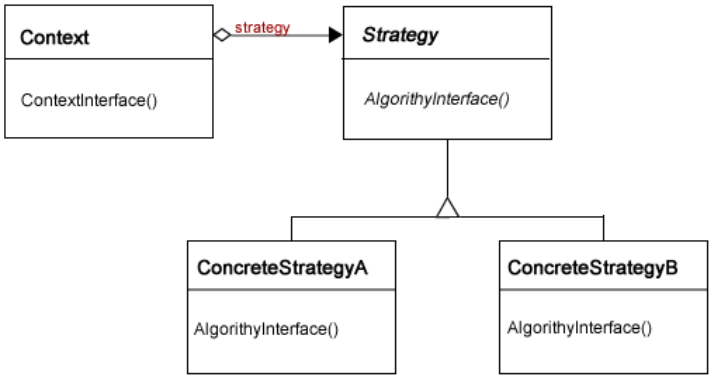


Σχήμα 14. Διάγραμμα κλάσεων προτύπου State

2.15 Στρατηγική (Strategy)

Το πρότυπο σχεδίασης "Στρατηγική" ορίζει μια οικογένεια αλγορίθμων, τους οποίους ενσωματώνει και επιτρέπει την εναλλαγή μεταξύ αυτών. Το πρότυπο δίνει την δυνατότητα μεταβολής των αλγορίθμων, ανεξάρτητα από τους πελάτες που τους χρησιμοποιούν. Ανήκει στην κατηγορία των συμπεριφορικών προτύπων.

Χρησιμοποιείται όταν ένας αλγόριθμος αναμένεται να έχει πολλές εναλλακτικές υλοποιήσεις ή ενδέχεται να προστεθούν κι άλλες στο μέλλον. (Chatzigeorgiou, 2005).

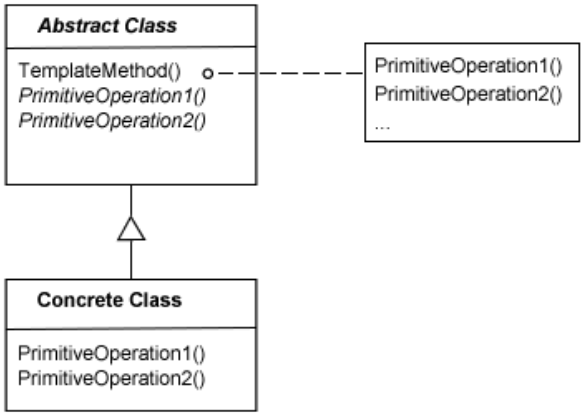


Σχήμα 15. Διάγραμμα κλάσεων προτύπου Stategy

2.16 Μέθοδος Υπόδειγμα (Template Method)

Το πρότυπο σχεδίασης "Μέθοδος Υπόδειγμα" ορίζει το περίγραμμα ενός αλγορίθμου σε μια λειτουργία, αφήνοντας ορισμένα βήματα στις παράγωγες κλάσεις. Το πρότυπο επιτρέπει στις παράγωγες κλάσεις να επαναορίσουν ορισμένα βήματα του αλγορίθμου χωρίς να αλλάξουν τη δομή του. Ανήκει στην κατηγορία των συμπεριφορικών προτύπων.

Χρησιμοποιείται για τον ορισμό των αμετάβλητων τμημάτων ενός αλγορίθμου σε μια λειτουργία μιας κλάσης και την μετάθεση της υλοποίησης των μεταβλητών τμημάτων του αλγορίθμου σε παράγωγες κλάσεις. (Chatzigeorgiou, 2005).

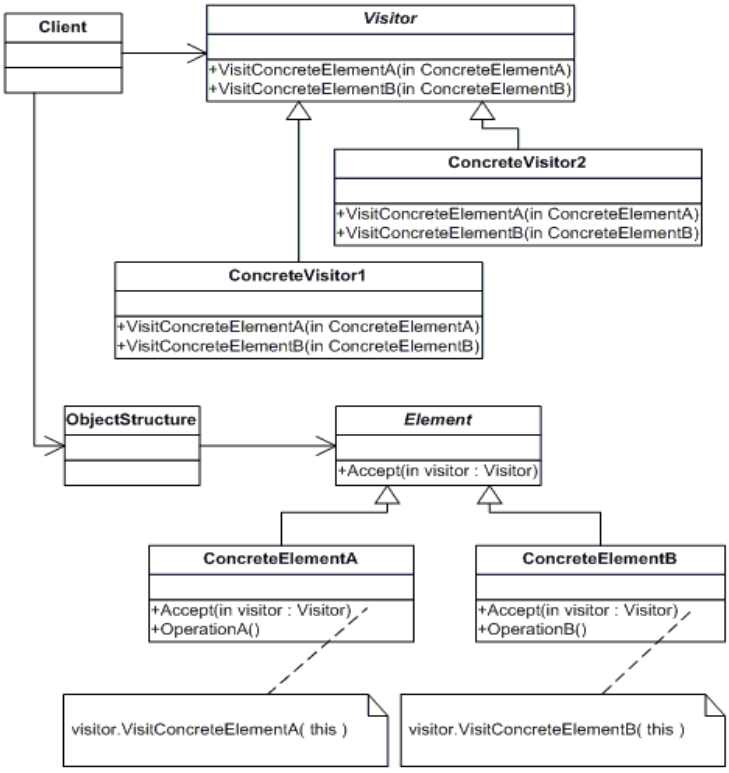


Σχήμα 16. Διάγραμμα κλάσεων προτύπου Template Method

2.17 Επισκέπτης (Visitor)

Το πρότυπο σχεδίασης "Επισκέπτης" έχει ως στόχο την αναπαράσταση μιας λειτουργίας που πρόκειται να πραγματοποιηθεί στα στοιχεία μιας δομής αντικειμένων. Το πρότυπο επιτρέπει τον ορισμό μιας νέας λειτουργίας χωρίς την τροποποίηση των κλάσεων των στοιχείων στα οποία επιδρά. Ανήκει στην κατηγορία των συμπεριφορικών προτύπων.

Χρησιμοποιούμε το πρότυπο "Επισκέπτης" όταν θέλουμε να προσθέσουμε λειτουργίες στα αντικείμενα μιας ιεραρχίας χωρίς να "μολύνουμε" τις κλάσεις τους με αυτές τις λειτουργίες. Η χρήση του προτύπου μας επιτρέπει να διατηρήσουμε σχετιζόμενες λειτουργίες σε ένα μέρος ορίζοντας αυτές σε μια ξεχωριστή κλάση. (Chatzigeorgiou, 2005).

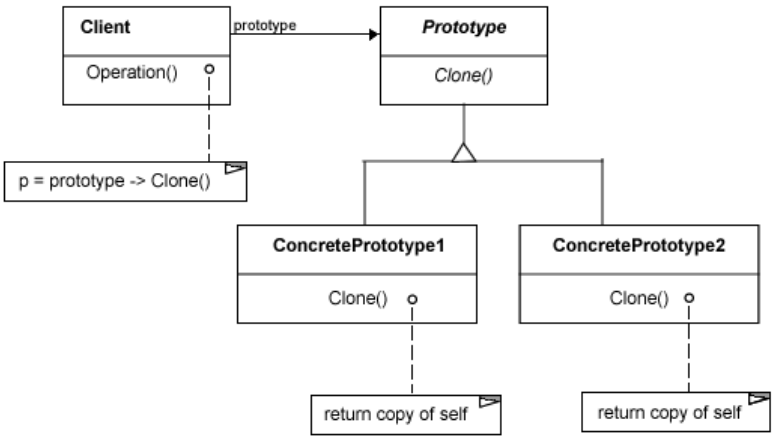


Σχήμα 17. Διάγραμμα κλάσεων προτύπου Visitor

2.18 Πρωτότυπο (Prototype)

Το πρότυπο σχεδίασης "Πρωτότυπο" προσδιορίζει τους τύπους των αντικειμένων που θα δημιουργηθούν βάση μιας πρωτότυπης διασύνδεσης, και δημιουργεί νέα αντικείμενα αντιγράφοντας το πρωτότυπο. Ανήκει στην κατηγορία των κατασκευαστικών προτύπων.

Το πρότυπο αυτό, χρησιμοποιείται για την αποφυγή δημιουργίας υποκλάσεων ενός αντικειμένου δημιουργού στην εφαρμογή και την αποφυγή του κόστους κληρονομικότητας για την δημιουργία ενός νέου αντικειμένου με το συμβατικό τρόπο (Gamma et al., 1995).



Σχήμα 18. Διάγραμμα κλάσεων προτύπου Prototype

ΚΕΦΑΛΑΙΟ 3 - ΜΕΘΟΔΟΛΟΓΙΑ

3.1 Μεθοδολογία Συστηματικής Ανασκόπησης Της Βιβλιογραφίας

Στο (Brereton et al., 2007) προτείνεται ότι μια συστηματική ανασκόπηση της βιβλιογραφίας πρέπει να αποτελείται από τρία βήματα: το σχεδιασμό, τη διενέργεια και την τεκμηρίωση της ανασκόπησης.

Κατά τη φάση του σχεδιασμού πρέπει να καθοριστούν τα ερευνητικά ερωτήματα, να ανεπτύχθη ένα πρωτόκολλο ανασκόπησης και στο τέλος να επικυρωθεί. Ο καθορισμός των ερευνητικών ερωτημάτων, αποτελεί το ποιο κρίσιμο στοιχείο της συστηματικής ανασκόπησης. Τα ερευνητικά ερωτήματα χρησιμοποιούνται για να εντοπίσουν τις λέξεις κλειδιά που πρέπει να χρησιμοποιηθούν για την αυτόματη αναζήτηση των σχετικών ερευνών. Οι λέξεις αυτές, ουσιαστικά καθορίζουν τα δεδομένα που πρέπει να εξαχθούν από κάθε πρωταρχική μελέτη και περιορίζουν τη συνολική διαδικασία. Τα ερωτήματα της έρευνας είναι το κομμάτι του πρωτοκόλλου που δεν μπορεί να τροποποιηθεί μετά τη αποδοχή του πρωτοκόλλου. Το πρωτόκολλο ανασκόπησης που αναπτύσσεται,

δίνει πληροφορίες για τον σχεδιασμό της ανασκόπησης, συμπεριλαμβάνοντας την προδιαγραφή της διαδικασίας που θα ακολουθηθεί, τις συνθήκες και τις μετρικές ποιότητας που θα εφαρμοστούν όταν επιλεγεί μια πρωταρχική μελέτη, και την κατανομή ορισμένων ενεργειών. Η επικύρωση του πρωτοκόλλου θεωρείται απαραίτητη, αφού το πρωτόκολλο αποτελεί κρίσιμο στοιχείο της ανασκόπησης.

Στη φάση της διενέργειας της ανασκόπησης, και εφόσον το πρωτόκολλο έχει τελειοποιηθεί, πρέπει να καταστρωθεί μια στρατηγική έρευνας για να εντοπιστούν οι σχετικές έρευνες που έχουν διεξαχθεί. Αμέσως μετά πρέπει να επιλεχθούν όσες κρίνονται κατάλληλες. Η διαδικασία επιλογής αποτελείται συνήθως από δυο στάδια. Αρχικά εξετάζεται ο τίτλος και στη συνέχεια η περίληψη των άρθρων που προκύπτουν από την αρχική αναζήτηση. Άρθρα που προκύπτει ότι δεν είναι σχετικά απορρίπτονται. Στην επόμενη φάση της διαδικασίας, εξετάζεται ολόκληρο το κείμενο των άρθρων που δεν έχουν απορριφτεί. Η ανασκόπηση γίνεται με βάση τα κριτήρια συμπερίληψης ή αποκλεισμού των άρθρων. Έπειτα, πρέπει να αξιολογηθεί η ποιότητα των ερευνών προκειμένου να ελαχιστοποιηθούν οι πιθανές προκαταλήψεις και να μεγιστοποιηθεί η εσωτερική και εξωτερική εγκυρότητα. Τέλος πρέπει να εξαχθούν τα απαιτούμενα δεδομένα ώστε να καταγραφούν με ακρίβεια οι πληροφορίες που θέλουν να αποσπάσουν οι ερευνητές από τις πρωταρχικές μελέτες και να ανασυντεθούν με τρόπο τέτοιο, ώστε να απαντούν στα ερωτήματα της έρευνας.

Η τελευταία φάση μιας συστηματικής ανασκόπησης είναι η διαδικασία της τεκμηρίωσης, κατά την οποία μετά την ολοκλήρωση της συστηματικής ανασκόπησης της βιβλιογραφίας, γράφεται μια λεπτομερής αναφορά για την ανασκόπηση και έπειτα αξιολογείται.

Σύμφωνα με τα (Kitchenham, Joint Technical Report και Kitchenham et al., 2009) το σχεδιάγραμμα της ανασκόπησης αποτελείται από έξι μέρη:

(α) ορισμό των ερευνητικών ερωτημάτων,

(β) ορισμό της διαδικασίας αναζήτησης,

(γ) ορισμό των κριτηρίων συμπερίληψης και αποκλεισμού,

(δ) ορισμό της ποιοτικής αξιολόγησης,

(ε) ορισμό της διαδικασίας συλλογής δεδομένων, και (στ) ορισμό της ανάλυσης δεδομένων.

3.2 Εμπειρικές μελέτες

Σύμφωνα με το (Wohlin et al., 2000), υπάρχουν τρεις βασικές προσεγγίσεις εμπειρικής έρευνας, οι μελέτες πεδίου, οι μελέτες περίπτωσης και τα πειράματα. Η επιλογή της προσέγγισης που θα ακολουθηθεί γίνεται συνήθως λαμβάνοντας υπόψη τη φύση και το αντικείμενο της εκάστοτε έρευνάς.

Η τεχνική των μελετών περίπτωσης χρησιμοποιείται για την παρακολούθηση έργων, ενεργειών ή εργασιών. Καθ' όλη τη διάρκεια της μελέτης συλλέγονται δεδομένα για έναν συγκεκριμένο σκοπό τα οποία συνήθως περνούν από στατιστική ανάλυση για την εξαγωγή αποτελεσμάτων. Τα πειράματα διεξάγονται συνήθως σε εργαστηριακό περιβάλλον, που παρέχει υψηλά επίπεδα ελέγχου. Τα υποκείμενα του πειράματος που έχουν επιλεγεί με βάση ορισμένα κριτήρια, εξετάζονται τυχαία σε κάποια καθήκοντα. Έπειτα ακολουθεί στατιστική ανάλυση των αποτελεσμάτων του πειράματος, ενώ υπολογίζεται και ο βαθμός χειραγώγησης ορισμένων μεταβλητών. Η διαφορά μεταξύ μιας μελέτης περίπτωσης και ενός πειράματος είναι ότι το δείγμα των μεταβλητών που χρησιμοποιούνται σε ένα πείραμα μπορεί να χειραγωγηθεί, ενώ σε μια μελέτη περίπτωσης το δείγμα των μεταβλητών προέρχεται από την αναπαράσταση μιας πραγματικής κατάστασης. Τέλος, μια μελέτη πεδίου, χρησιμοποιείται συνήθως για την διεξαγωγή μιας αναδρομικής εξέτασης, όταν για παράδειγμα ένα εργαλείο ή μια τεχνική χρησιμοποιείται για ένα χρονικό διάστημα. Η μελέτη γίνεται με χρήση ερωτηματολογίων που διανέμονται σε ένα αντιπροσωπευτικό δείγμα πληθυσμού που θέλουμε να μελετήσουμε. Τα αποτελέσματα της έρευνας αναλύονται και στην τελική φάση γενικεύονται για τον πληθυσμό απ' όπου προέρχονταν το δείγμα.

Στο κεφάλαιο 3 παρουσιάζεται μια μελέτη περίπτωσης. Η πληθώρα των εφαρμογών ανοιχτού κώδικα αναγάγει την μελέτη περίπτωσης στη βέλτιστη προσέγγιση έρευνας. Αντίθετα, μία μελέτη πεδίου δεν θα ταίριαζε στη δική μας έρευνα διότι θα οδηγούσε στο να μην συνυπολογίσουμε τα πρότυπα εκείνα που χρησιμοποιήθηκαν από τους προγραμματιστές χωρίς πρόθεση, τις περισσότερες

φορές κατά τύχη. Τέλος ένα πείραμα που θα βασίζονταν σε προγραμματιστές ανοιχτού λογισμικού θα μείωνε τον αριθμό των παρατηρήσεων στην έρευνα μας.

3.2.1 Μελέτες Περίπτωσης

Μια μελέτη περίπτωσης συντάσσεται για να μελετήσει μια οντότητα ή ένα φαινόμενο μέσα σε συγκεκριμένα χρονικά πλαίσια (Wohlin et al., 2000). Ο ερευνητής συγκεντρώνει λεπτομερείς πληροφορίες συχνά εφαρμόζοντας διάφορες διαδικασίες συλλογής πληροφοριών.

Οι μελέτες περίπτωσης είναι ιδανικές για την βιομηχανική αξιολόγηση μεθόδων και εργαλείων της μηχανικής λογισμικού γιατί μπορούν να αποφύγουν κλιμακωτά αυξανόμενα προβλήματα. Ένα πλεονέκτημα των μελετών περίπτωσης είναι ότι προσφέρουν ευκολία στη φάση του σχεδιασμού και προγραμματισμού των εργασιών, ενώ μειονεκτήματα είναι ότι τα αποτελέσματα που προκύπτουν είναι δύσκολο να γενικευθούν και ακόμα δυσκολότερο να αποκωδικοποιηθούν. Στην περίπτωση που η επίδραση μιας διαδικασίας αλλαγής είναι πολύ διαδεδομένη, τότε μια μελέτη περίπτωσης είναι πιο κατάλληλη. Αυτό συμβαίνει επειδή η επίδραση μιας αλλαγής μπορεί να αξιολογηθεί μόνο σε υψηλό επίπεδο αφαίρεσης, γιατί μια διαδικασία τροποποίησης, περιλαμβάνει μικρότερες και πιο λεπτομερείς αλλαγές πέραν της διαδικασίας ανάπτυξης και επειδή τα αποτελέσματα μιας αλλαγής δεν μπορούν πάντα να είναι άμεσα ορατά.

Η έρευνα μέσω μελετών περίπτωσης είναι μια καθιερωμένη μέθοδος που χρησιμοποιείται για εμπειρικές μελέτες σε διάφορες επιστήμες. Στα πλαίσια της μηχανικής λογισμικού, οι μελέτες περίπτωσης χρησιμοποιούνται όχι μόνο για να αξιολογήσουν πώς και γιατί συμβαίνουν συγκεκριμένα φαινόμενα, αλλά και για να αξιολογήσουν διαφορές, όπως για παράδειγμα μεταξύ δυο σχεδιαστικών μεθόδων.

3.2.2 Μεθοδολογία Σύνταξης Μελέτης Περίπτωσης

Σύμφωνα με το (Kitchenham et al., 1995) τα βήματα που απαιτούνται για να συντάξει κανείς μια μελέτη περίπτωσης περιλαμβάνουν:

(α) Ορισμό μιας υπόθεσης.

(β) Επιλογή ορισμένων εφαρμογών,

(γ) Επιλογή της μεθόδου σύγκρισης,

(δ) Ελαχιστοποίηση των παραγόντων σύγχυσης,

(ε) Σχεδιασμό πλάνου για την μελέτη περίπτωσης,

(στ) Παρακολούθηση της μελέτης περίπτωσης,

(ζ) Ανάλυση και αναφορά των αποτελεσμάτων

Για να ορίσουμε την υπόθεση, ξεκινάμε ορίζοντας την επίδραση που περιμένουμε να έχει η μέθοδος. Ο ορισμός αυτός πρέπει να είναι αρκετά λεπτομερής, και να ξεκαθαρίζει τις μετρήσεις που πρέπει να γίνουν για να προκύψει το αποτέλεσμα. Επίσης, είναι σημαντικό να οριστεί τι δεν αναμένεται να συμβεί. Αυτό είναι ιδιαίτερα σημαντικό γιατί επίσημα δεν μπορούμε να αποδείξουμε ότι μια υπόθεση αληθεύει. Μπορούμε μόνο να την καταρρίψουμε. Γι αυτό δηλώνουμε και μια μηδενική υπόθεση για να δείξουμε ότι δεν υπάρχει διαφοροποίηση στην μεταχείριση.

Κατά την επιλογή των εφαρμογών, είναι σημαντικό να επιλέξουμε εφαρμογές του ίδιου τύπου με αυτές που μας ενδιαφέρουν. Η επιλογή πρέπει να βασίζεται όχι μόνο στον τύπο της εφαρμογής, αλλά και στην συχνότητα που ο κάθε τύπος αναπτύσσεται.

Για το τρίτο βήμα πρέπει να έχουμε κατά νου, ότι η μελέτη περίπτωσης είναι από την φύση της συγκριτική μέθοδος όσον αφορά τα αποτελέσματά δύο ή περισσοτέρων μεθόδων. Για να επιβεβαιώσουμε την εσωτερική εγκυρότητα, πρέπει να βρούμε μια έγκυρη βάση για την αξιολόγηση των αποτελεσμάτων της μελέτης περίπτωσης. Για να επιτευχθεί αυτό υπάρχουν τρείς τρόποι: (1) Να επιλέξουμε ένα παρόμοιο έργο για αν συγκρίνουμε τα αποτελέσματα, (2) να συγκρίνουμε τα αποτελέσματα χρησιμοποιώντας μια νέα μέθοδο αντίθετη με τη γραμμή της μεθόδου που εμείς χρησιμοποιούμε και (3) εφόσον η μέθοδος μπορεί να εφαρμοστεί σε ανεξάρτητα συστατικά, να την εφαρμόσουμε τυχαία σε ορισμένα μόνο συστατικά προϊόντων.

Το τέταρτο βήμα αφορά την ελαχιστοποίηση των παραγόντων σύγχυσης. Τέτοιοι παράγοντες μπορεί να είναι η εκμάθηση του τρόπου χρήσης μιας μεθόδου ή ενός εργαλείου κατά τη διάρκεια της προσπάθειας αξιολόγησής του, η χρήση είτε πολύ ενθουσιώδους είτε πολύ δύσπιστου προσωπικού σχετικά με τη χρήση της μεθόδου ή του εργαλείου, η σύγκριση διαφορετικών τύπων εφαρμογών, κ.α. Ορισμένες φορές, μπορούμε να ελέγξουμε τη σύγχυση, μετρώντας τον παράγοντα σύγχυσης και ρυθμίζοντας τα αποτελέσματα ανάλογα.

Στην επόμενη φάση, το πλάνο αναγνωρίζει και καταγράφει όλες τις πτυχές που πρέπει να διευθετηθούν για την ομαλή διεξαγωγή της αξιολόγησης της μεθόδου. Σε αυτές, συμπεριλαμβάνονται τα απαραίτητα μέτρα, οι διαδικασίες συλλογής δεδομένων και το ανθρώπινο δυναμικό που είναι υπεύθυνο για τη συλλογή και ανάλυση των δεδομένων.

Η παρακολούθηση της μελέτης περίπτωσης σύμφωνα με το πλάνο του προηγούμενου βήματος, επιβεβαιώνει ότι οι μέθοδοι ή τα εργαλεία που εξετάζονται χρησιμοποιούνται σωστά, και ότι όλοι οι παράγοντες που θα μπορούσαν να προδιαθέσουν τα αποτελέσματα καταγράφονται. Αυτό, θα βοηθήσει στο τέλος της έρευνας στην συγγραφή μιας αναφοράς αξιολόγησης με σκοπό την πρόταση αλλαγών στις διαδικασίες.

Τέλος, για την ανάλυση και αναφορά των αποτελεσμάτων, η διαδικασία που ακολουθείται κάθε φορά, εξαρτάται από τον αριθμό των δεδομένων που πρέπει να αναλυθούν.

ΚΕΦΑΛΑΙΟ 4 - ΑΝΑΣΚΟΠΗΣΗ ΤΗΣ ΒΙΒΛΙΟΓΡΑΦΙΑΣ "ΠΡΟΤΥΠΑ ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΒΙΒΛΙΟΘΗΚΕΣ (API)"

Το κεφάλαιο αυτό, στοχεύει στο να συνοψίσει την υπάρχουσα μέχρι σήμερα ερευνητική δραστηριότητα στο πεδίο της χρήσης των προτύπων σχεδίασης στις βιβλιοθήκες. Πρώτο βήμα, για να επιτευχθεί αυτό, είναι η διεξαγωγή μιας συστηματικής βιβλιογραφικής ανασκόπησης.

4.1 Μεθοδολογία Ανασκόπησης

Σύμφωνα με την μεθοδολογία που περιγράφεται στο κεφάλαιο 1.2 παρουσιάζουμε τα έξι βήματα της συστηματικής ανασκόπησης της βιβλιογραφίας που διεξαγάγαμε στα πλαίσια αυτής της μελέτης.

4.1.1 Τα ερωτήματα της έρευνας

Στη μελέτη αυτή, θα διερευνήσουμε ζητήματα που αφορούν την μέχρι σήμερα ερευνητική δραστηριότητα πάνω στα αντικειμενοστραφή πρότυπα σχεδίασης. Το βασικό ερευνητικό ερώτημα που αποτυπώνεται στην μελέτη είναι:

RQ1: Υπάρχουν μελέτες που αναφέρονται στον βαθμό χρήσης των προτύπων σχεδίασης στις βιβλιοθήκες (API);

4.1.2 Η διαδικασία αναζήτησης

Η διαδικασία αναζήτησης διεξήχθη μέσω των ιστοχώρων των εξής πέντε ψηφιακών βιβλιοθηκών:

1. http://www.computer.org/ ,
2. http://www.springerlink.com/,
3. http://www.sciencedirect.com/,
4. http://dl.acm.org/ ,
5. http://scholar.google.gr/.

Σαν λέξεις κλειδιά για την αναζήτηση, χρησιμοποιήθηκαν τρεις διαφορετικοί συνδυασμοί. Και στις τρεις περιπτώσεις διεξήχθησαν σύνθετες αναζητήσεις, όπου η δεύτερη λέξη κλειδί, δηλαδή η λέξη «pattern» που είχε τον περιορισμό να βρίσκετε σε όλο το κείμενο ή την περίληψη, ήταν σε όλες η ίδια, αλλάζοντας κάθε φορά την πρώτη λέξη κλειδί που είχε σαν περιορισμό να βρίσκετε στον τίτλο. Στην πρώτη περίπτωση χρησιμοποιήθηκε η λέξη «API» στην δεύτερη, η λέξη «Framework», ενώ στην Τρίτη περίπτωση η λέξη «Library». Το σύνολο των άρθρων που επιστράφηκαν από τις παραπάνω αναζητήσεις αντιστοιχούσε σε 1158 άρθρα. Ωστόσο, ο μεγαλύτερος αριθμός από τα άρθρα αυτά θεωρήθηκε ότι δεν σχετίζονταν ικανοποιητικά με τα πρότυπα σχεδίασης των API και αποκλείστηκαν. Ο αποκλεισμός των μη σχετικών άρθρων διεξήχθη με το χέρι, σύμφωνα με τα κριτήρια συμπερίληψης ή αποκλεισμού που ορίζονται στο επόμενο υποκεφάλαιο.

4.1.3 Κριτήρια Συμπερίληψης και Αποκλεισμού.

Σύμφωνα με το (Dyba and Dingsoyr, 2008), υπάρχουν τέσσερα στάδια φιλτραρίσματος του συνόλου των άρθρων, προκειμένου να προκύψει το σύνολο των πρωταρχικών μελετών. Τα βήματα αυτά είναι τα εξής:

- Εύρεση των σχετικών μελετών – αναζήτηση σε ψηφιακές βιβλιοθήκες. (με την ολοκλήρωση του βήματος αυτού το σύνολο των άρθρων ανέρχονταν στα 1158 άρθρα.)
- Αποκλεισμός μελετών βάση του τίτλο τους. (με την ολοκλήρωση του βήματος αυτού το σύνολο των άρθρων ανέρχονταν στα 565 άρθρα.)
- Αποκλεισμός μελετών βάση της περίληψή τους. (με την ολοκλήρωση του βήματος αυτού το σύνολο των άρθρων ανέρχονταν στα 128 άρθρα.) 30 άρθρα

που είχαν πολύ μικρές ή δεν είχαν περιλήψεις, θεωρήθηκε ότι πρέπει να περάσουν στην επόμενη φάση για μελέτη του πλήρους κειμένου.

- Προσεκτική μελέτη των άρθρων και επιλογή των πιο σχετικών με τα πρότυπα σχεδίασης των API, βάση του πλήρους κειμένου. (με την ολοκλήρωση του βήματος αυτού το τελικό σύνολο πρωταρχικών μελετών αποτελούνταν από 2 άρθρα.)

Ο πιο συνηθισμένος λόγος για τον αποκλεισμό ενός άρθρου από τον τίτλο του, ήταν ότι το άρθρο δεν ασχολούνταν με πρότυπα σχεδίασης λογισμικού, αλλά με άλλα είδη προτύπων. Επιπλέον, κατά την εξέταση των περιλήψεων, η πλειοψηφία των άρθρων που αποκλείστηκαν ασχολούνταν με αρχιτεκτονικά ή HCI πρότυπα. Τέλος, το βασικό κριτήριο απόρριψης άρθρων στην τελευταία φάση της μελέτης του πλήρους κειμένου, ήταν η απουσία ονομαστικών αναφορών σε τουλάχιστον ένα από τα 23 GoF πρότυπα (Gamma et al., 1995) που χρησιμοποιούνται στις βιβλιοθήκες.

4.1.4 Ποιοτική αξιολόγηση

Η ποιότητα ενός άρθρου που συντάσσει μια συστηματική ανασκόπηση, σχετίζεται άμεσα με την ποιότητα των πρωταρχικών μελετών, από την άποψη ότι τα αποτελέσματα και τα συμπεράσματα της δευτερογενούς μελέτης βασίζονται στα ευρήματα των πρωταρχικών μελετών. Έτσι, σε μια ανασκόπηση, είναι σημαντικό να συμπεριληφθούν πρωταρχικές μελέτες που βασίζονται σε σταθερές μεθόδους και παρουσιάζουν ξεκάθαρα τα αποτελέσματά τους. Για να επιτευχθεί ο στόχος αυτός, στην ανασκόπηση μας, συμπεριλάβαμε άρθρα που έχουν δημοσιευθεί στα καλύτερα περιοδικά και συνέδρια, καθώς και σε workshops που διεξήχθησαν στα πλαίσια κορυφαίων συνεδρίων στον τομέα της μηχανικής λογισμικού.

Αξίζει να σημειωθεί ότι κατά τη διάρκεια της επιλογής των άρθρων, συγκεντρώναμε ένα σύνολο μεταβλητών που περιέγραφαν την εκάστοτε πρωταρχική μελέτη. Για κάθε μελέτη, καταγράφαμε τα εξής δεδομένα:

- Τίτλος δημοσίευσης ,
- Τύπος δημοσίευσης (περιοδικό, συνέδριο, workshop),

- Τόπος δημοσίευσης (όνομα συνεδρίου ή περιοδικού) και
- Έτος δημοσίευσης.

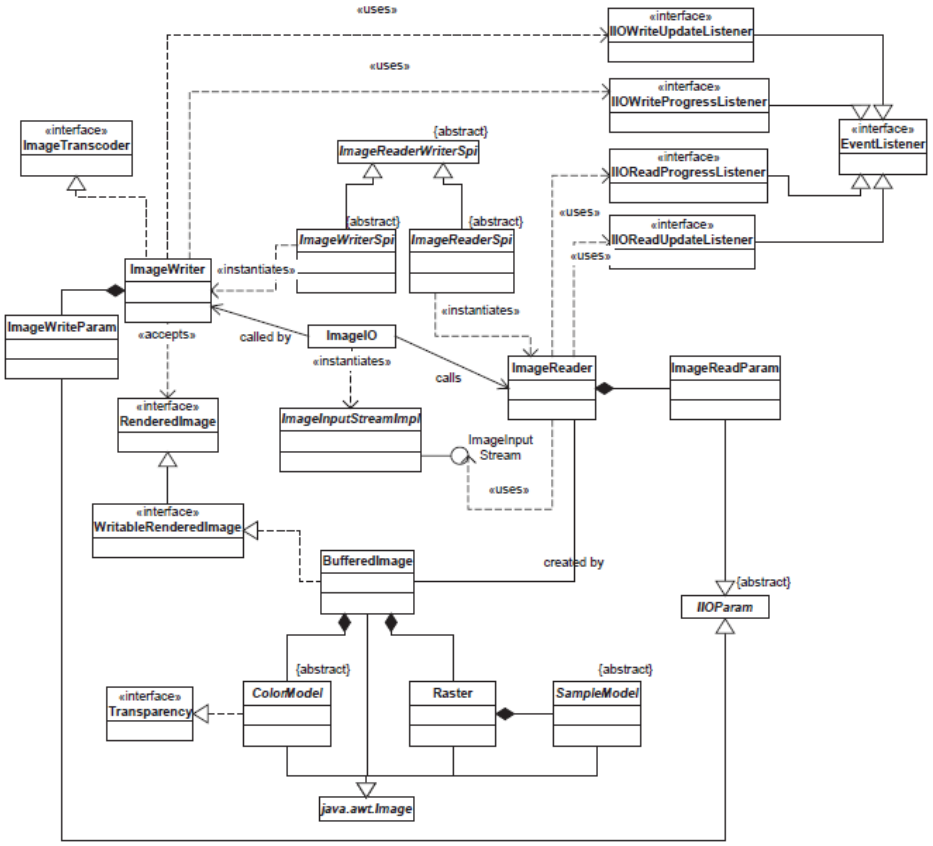
4.2 ΣΥΖΗΤΗΣΗ ΤΩΝ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΤΗΣ ΑΝΑΣΚΟΠΗΣΗΣ

Στο κεφάλαιο αυτό, συζητάμε τα αποτελέσματα της ανασκόπησής μας, σύμφωνα με τα ερωτήματα τη έρευνας που έχουν διατυπωθεί. Τα άρθρα που κρίθηκαν ότι σχετίζονται με το θέμα που μελετάμε στην εργασία αυτή, ήταν μόνο 2. Στο πρώτο άρθρο (Colin J. Neill (2003)) αναφέρονται 2 παραδείγματα των πλαισίων που έχουν αναπτυχθεί για την απεικόνιση της πλατφόρμας Java. Ενώ στο δεύτερο άρθρο (Brian Ellis, (2007)),το όποιο σχετίζεται 100% με την μελέτη της παρούσας εργασίας, συγκρίνονται η χρηστικότητα του προτύπου εργοστάσιο (Factory) και των κατασκευαστών (δομητών) στα σχέδια API.

Συγκεκριμένα στο άρθρο “Leveraging object-orientation for real-time imaging systems”, (Colin J. Neill (2003)) αναφέρεται ότι παραδείγματα των πλαισίων έχουν αναπτυχθεί για την απεικόνιση της πλατφόρμας Java. Δύο από αυτά, η Java Image I / O API και 2DT API, θα διερευνηθούν με αναφορά στα πρότυπα και τις αρχές του σχεδιασμού. Αυτά είναι τα συστατικά πακέτα της Java Advanced Imaging (JAI) API που δεν μπορεί να συζητηθεί λεπτομερώς εδώ λόγω μέγεθος τους και της πολυπλοκότητας τους όπου πάνω από 300 κλάσεις και διασυνδέσεις ορίζονται στο JAI API. Ο συμβολισμός που χρησιμοποιείται και στα 2 παραδείγματα είναι από τη Unified Modeling Language (UML), όπου παρέχονται οι κλάσεις ,όπως πλαίσια με τρία τμήματα (για την ταυτότητα, τις ιδιότητες και τις μεθόδους), οι διεπαφές που είναι πλαίσια με μόνο δύο χωρίσματα (μια διεπαφή δεν έχει χαρακτηριστικά), καθώς και οι σχέσεις που αναφέρονται με γραμμές.

Το πρώτο παράδειγμα (Java Image I / O API)είναι ένα παράδειγμα ενός πλαισίου που θα παρέχει μια άμεσα συνδεδεμένη αρχιτεκτονική για την εργασία με εικόνες που είναι αποθηκευμένες σε αρχεία και προσβάσιμο σε ένα δίκτυο [16]. Το API περιλαμβάνει έξι πακέτα που προσδιορίζουν 46 τάξεις και διασυνδέσεις και υποστηρίζει την επέκταση μέσω Πρόσθετων λειτουργιών τη συγκεκριμένη μορφή που δημιουργήθηκε από υποκλάσεις, βασικών αφηρημένων κατηγοριών, την εφαρμογή ορισμένων διεπαφών και παραμετροποίησης υποκατηγορίες με

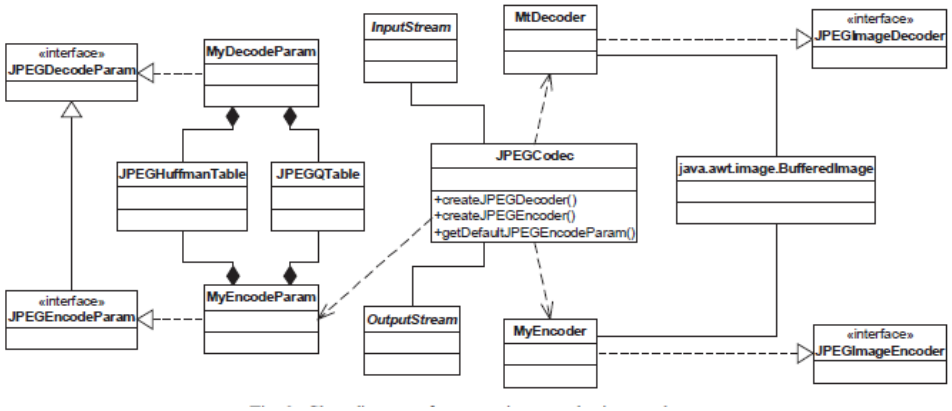
μεταδεδομένα. Ένα μέρος του διαγράμματος κλάσης αυτής της API παρουσιάζεται στο παρακάτω σχήμα.



Σχήμα 19 : Διάγραμμα κλάσεων του «Java Image I / O API»

Τα σχεδιαστικά πρότυπα που χρησιμοποιούνται σε αυτό το API είναι τα πρότυπα Observer, Façade, Singleton και Abstract Factory.Ποιο συγκεκριμένα ,αναφέρει ότι , με το πρότυπο observer υποστηρίζεται η χαμηλή σύζευξη, η οποία είναι και ένα κλειδί για την επαναχρησιμοποίηση, ενώ στο πρότυπο Façade τονίζει ότι το κυριότερο πλεονέκτημα του είναι η μείωση της εξωτερικής συζεύξεις του πακέτου, η οποία μειώνει την πολυπλοκότητα και προάγει την ανεξαρτησία του υποσυστήματος.

Το δεύτερο παράδειγμα (Java API 2DT) ,που αναλύεται στο άρθρο, χρησιμοποιείται για την ανάπτυξη των προηγμένων γραφικών των βιβλιοθηκών και των φίλτρων εικόνας. Περιλαμβάνει συνολικά, 127 κλάσεις και διεπαφές. Παρακάτω φαίνεται το Διάγραμμα κλάσεων του συγκεκριμένου παραδείγματος.



Σχήμα 20: Διάγραμμα κλάσεων του «Java Image I / O API»

Σε αυτό, τα πρότυπα που είναι εμφανές ότι χρησιμοποιούνται είναι τα πρότυπα Abstract Factory (πιο συγκεκριμένα το Factory) ,το πρότυπο Strategy και Observer.

Στο επόμενο άρθρο «The Factory Pattern in API Design: A Usability Evaluation» (Brain Ellis (2007)) περιγράφεται μια μελέτη χρήστη όπου συγκρίνοντας τη χρηστικότητα του προτύπου εργοστάσιο και των κατασκευαστών στα σχέδια API βρέθηκαν πολύ σημαντικά αποτελέσματα τα οποία δείχνουν ότι τα εργοστάσια είναι καταστρεπτικά για τη χρηστικότητα των API σε πολλές διαφορετικές καταστάσεις. Τα αποτελέσματα έδειξαν ότι οι χρήστες απαιτούν σημαντικά περισσότερο χρόνο ($p = 0,005$) για να κατασκευάσουν ένα αντικείμενο με ένα εργοστάσιο σε σχέση με ένα δομητή κατά την εκτέλεση τόσο των ευαίσθητων πλαισίων όσο και των ελεύθερων εργασιών. Τα αποτελέσματα αυτά δείχνουν ότι η χρήση των εργοστασίων μπορεί και πρέπει να αποφεύγεται σε πολλές περιπτώσεις όπου άλλες τεχνικές, όπως οι δομητές ή ομάδες κλάσεων, μπορούν να χρησιμοποιηθούν.

Στην εισαγωγή αναφέρεται ότι ένας κοινός προγραμματιστής μπορεί να χρησιμοποιήσει μόνο ένα μικρό μέρος της συνολικής λειτουργικότητας ενός API, μαθαίνοντας ότι το υποσύνολο είναι ένα δύσκολο έργο για τους νέους προγραμματιστές [1]. Οι σχεδιαστές API πρέπει να εξετάσουν πολλούς διαφορετικούς παράγοντες κατά τη δημιουργία ενός API, όπως ο βαθμός κατάτμησης της κλάσης, το επίπεδο της αφαίρεσης, τη συνέπεια με άλλα APIs, κλπ.

Η έρευνα έχει επίσης δείξει ότι σχεδιάζοντας τα API προσεκτικά για το προοριζόμενο κοινό τους βελτιώνει τη χρηστικότητα [2]. Μέχρι σήμερα, μελέτες χρηστικότητας των API έχουν ως επί το πλείστον θεωρήσει τη χρηστικότητα του API στο σύνολό της, ότι παρέχει για το μέλλον ελάχιστη καθοδήγηση στους σχεδιαστές API. Μια μικρή έρευνα εξέτασε τη χρηστικότητα των συγκεκριμένων σχεδιαστικών προτύπων και παραδειγμάτων προγραμματισμού που εφαρμόζονται στο σχεδιασμό του API.

Σε προηγούμενη εργασία, Stylos et al. [3] συζητήθηκε η δυνατότητα χρησιμοποίησης των δομητών αντικειμένου με τις απαιτούμενες παραμέτρους, σε σύγκριση με τον προεπιλεγμένο δομητή. Εδώ, εξετάζονται οι επιπτώσεις χρηστικότητας ενός από τα πιο γνωστά πρότυπα αντικειμενοστραφούς σχεδιασμού: το πρότυπο εργοστάσιο [4]. Αυτή η νέα έρευνά μας δείχνει ότι η δημιουργία αντικείμενων από τα εργοστάσια που χρησιμοποιούνται στο API είναι πολύ πιο χρονοβόρα από τους δομητές, ανεξάρτητα από το πλαίσιο ή το επίπεδο της εμπειρίας του προγραμματιστή χρησιμοποιώντας το API.

4.2.1 **Σχετικές εργασίες**

Η ανάλυση της χρηστικότητας των σχεδίων API είναι ένας σχετικά νέος τομέας. Ωστόσο, έχουν ήδη υπάρξει πολλές σχετικές εξερευνήσεις σε αυτό το θέμα. Η Microsoft έχει ασχοληθεί με το πλαίσιο «γνωστικών διαστάσεων" [10] για να συγκρίνει τη χρηστικότητα των διαφόρων σχεδίων API για τρεις "personas" που αντιπροσωπεύουν διαφορετικά αρχέτυπα των προγραμματιστών σαν να χρησιμοποιούν το API [11]. Τα αποτελέσματα αυτών των συγκρίσεων χρησιμοποιούνται για να ενημερώσουν την διαδικασία σχεδιασμού και να βελτιώσουν το API. Η έρευνα έχει διεξαχθεί, σε γενικές γραμμές για τον ρόλο των πρότυπων σχεδίασης, και ιδίως για το αφηρημένο πρότυπο εργοστάσιο [9].Η εργασία αυτή δείχνει ότι, παρόλο που οι εκπαιδευτικοί θεωρούν το πρότυπο εργοστάσιου μια ανώτερη μέθοδο, αισθάνονται ότι είναι πολύ δύσκολο να την εξηγήσουνε στους αρχάριους φοιτητές, και γι 'αυτό το αποφεύγουν υπέρ των άλλων. Η μελέτη αυτή εστιάζει στην ευχρηστία των APIs που χρησιμοποιούν το πρότυπο εργοστάσιο, ενώ η τελευταία έρευνα σχετικά με τα εργοστάσια έχει ως επί το πλείστον επικεντρωθεί στα αρχιτεκτονικά πλεονεκτήματα του προτύπου

εργοστάσιο για υλοποιήσεις του συστήματος. Μια πρόσφατη μελέτη [3] κατέδειξε ότι οι δοκιμές των χρηστών είναι ένα αποτελεσματικό μέσο για τον προσδιορισμό των ιδιοτήτων χρηστικότητας των APIs όπως η εύκολη εύρεση και τήρηση των προσδοκιών των χρηστών. Σε αυτή την έρευνα, έχουνε εφαρμόσει αυτή την προσέγγιση για τη χρήση του προτύπου του εργοστασίου όπου συγκρίνει τη χρηστικότητα των παραμετροποιημένων δομητών με εκείνους τους δομητές προεπιλογής.

4.2.2 **Μεθοδολογία της μελέτης**

Έχουν επεξεργαστεί ένα σύνολο από πέντε εργασίες προγραμματισμού γραμμένες σε γλώσσα προγραμματισμού Java για να ερευνηθεί η χρησιμοποίηση του προτύπου εργοστάσιο στο API. Κάθε εργασία κατασκευάστηκε για να διαφέρει από όλες τις άλλες με όσο το δυνατόν περισσότερές διαστάσεις. Όλες οι εργασίες εκτός από την πρώτη παρουσιάστηκαν με τη μορφή ενός έργου Eclipse [13]. Οι συμμετέχοντες είχαν επίσης την πρόσβαση στο Sun Java SE 1,5 του API [7]. Οι συμμετέχοντες επελέγησαν από ένα σύνολο των υποψηφίων που δημιουργήθηκε με χρήση μιας διαφημιστικής υπηρεσίας για πειράματα στις πανεπιστημιουπόλεις. Υπήρξε μια προ-διαλογική έρευνα για την επιλογή των υποψηφίων ώστε να δεν έχουν τουλάχιστον ένα έτος εμπειρίας στη Java.. Δώδεκα συμμετέχοντες επιλέχθηκαν, με προγραμματιστική εμπειρία που κυμαινόταν από 1-22 ετών. Οκτώ είχαν επαγγελματική εμπειρία στον προγραμματισμό, δύο ήταν μαθητές, και οι δύο ήταν φοιτητές. Έξι είχαν τουλάχιστον κάποια εμπειρία με το πρότυπο εργοστάσιο, και τέσσερις είχαν σημαντική εμπειρία με το πρότυπο εργοστάσιο. Όλοι οι συμμετέχοντες ήταν άνδρες μεταξύ 18 και 35 ετών. Οι συμμετέχοντες τυχαία χωρίστηκαν σε δύο ομάδες, στο εργοστάσιο ή τις συνθήκες κατασκευαστή . Σε κάθε συμμετέχοντα είχαν δοθεί γραπτές οδηγίες για την ολοκλήρωση κάθε εργασίας, και ζητήθηκε να εκφράσουν τους στόχους, τις παραδοχές, τις υποθέσεις, και τις στρατηγικές για την ολοκλήρωση των εργασιών χρησιμοποιώντας ένα πρωτόκολλο δυνατής-σκέψης. Οι συμμετέχοντες έπρεπε να ολοκληρώσουν κάθε εργασία με τη σειρά που τους δόθηκε χωρίς να προχωρήσουνε σε επόμενη χωρίς να έχουν ολοκληρώσει την προηγούμενη . Οι εργασίες ήταν σχεδιασμένες έτσι ώστε τα άτομα να γνώριζαν όταν είχαν ολοκληρωθεί επιτυχώς.

4.2.3 **Μέτρηση**

Ένας σημαντικός στόχος αυτής της μελέτης ήταν να προσφέρει ποσοτικές μετρήσεις των διαφορών στη χρηστικότητα μεταξύ των εργοστασίων και των κατασκευαστών. Στο πλαίσιο των API, όπου ο στόχος είναι συχνά να γράψουν σωστό κώδικα ταχύτερα και αποτελεσματικότερα, η χρηστικότητα σχετίζεται σε μεγάλο βαθμό με το χρόνο για την ολοκλήρωση εργασιών, η οποία περιλαμβάνει δραστηριότητες όπως η έρευνα για την τεκμηρίωση. Λόγω των μεγάλων μεμονωμένων διαφορών, ο χρόνος ολοκλήρωσης είναι πιο εύκολο να συγκριθεί όταν μετράται σε θέματα.

Εργασία 1: «Σημειωματάριο εργασιών Email»

Το Σημειωματάριο εργασιών e-mail ήταν πάντα η πρώτη διαχειριζόμενη εργασία.. Διαφέρει από τις άλλες τέσσερεις εργασίες στο ότι, αντί να χρησιμοποιούν Eclipse, οι συμμετέχοντες παρουσιάστηκαν με ένα κενό έγγραφο απλού κειμένου στον κειμενογράφο Notepad και ζήτησαν να γράψουν κώδικα σε Java χρησιμοποιώντας οτιδήποτε άλλο πραγματικό ή φανταστικό API αυτοί ήθελαν. Αυτή η εργασία έχει σχεδιαστεί για να αποσπάσει την προσδοκία του προγραμματιστή σχετικά με τη δημιουργία αντικειμένων. Οι συμμετέχοντες κλήθηκαν να κατασκευάσουν ένα email αντικείμενο με μια λίστα των πληροφοριών, συμπεριλαμβανομένων τη διεύθυνση του αποστολέα και του παραλήπτη, το σώμα του e-mail, και (το σημαντικότερο) αν το email ήταν απλό ή εμπλουτισμένο κείμενο. Η τελευταία παράμετρος καθιστά την εργασία ενός υποψηφίου για τη χρήση ενός προτύπου εργοστάσιο προτείνοντας δύο υποτύπους του ηλεκτρονικού ταχυδρομείου του οποίου η εφαρμογή θα μπορούσε να κρύβεται από ένα εργοστάσιο.

Εργασία 2: «Eclipse εργασιών Email»

Ένα δεύτερο μήνυμα ηλεκτρονικού ταχυδρομείου που σχετίζεται με την εργασία, διαχειριζόμενο ως έργο Eclipse, χρησιμοποίησε την ίδια περιγραφή των εργασιών, όπως το Σημειωματάριο εργασιών e-mail: γράψτε μια μέθοδο που έχει παραμέτρους για ένα μήνυμα ηλεκτρονικού ταχυδρομείου και επιστρέφει ένα Email αντικείμενο. Αυτή τη φορά, όμως, οι συμμετέχοντες κλήθηκαν να χρησιμοποιήσουν ένα απλό e-mail API εκ των προτέρων, το οποίο χτίστηκε από τους πειραματιστές.

Το API που παρουσιάστηκε δημιουργήθηκε με e-mail χρησιμοποιώντας ένα εργοστάσιο αντί για κατασκευαστές. Παρά το γεγονός ότι η έλλειψη μιας κατάστασης του κατασκευαστή για το έργο αυτό αποκλείει μια άμεση σύγκριση, η εργασία, σε συνδυασμό με τη διαδικασία δυνατής σκέψης, είχε ως στόχο να διευκρινίσει αντιδράσεις των χρηστών για να βρουν ένα εργοστάσιο, όταν αναμενόταν ένας κατασκευαστής .

Εργασία 3: «Thingies εργασία»

Το «Thingies» έργο σχεδιάστηκε για να είναι ένα εντελώς ουδέτερο πλαίσιο εργασίας. Οι συμμετέχοντες κλήθηκαν να δημιουργήσουν ", δύο υποκατηγορίες της αφηρημένης "Thingies" κλάσεις, ένα "Squark" και "Flarn και στη συνέχεια να καλέσουν μια απλή μέθοδο για εκτέλεση σε καθεμία από αυτές. Η Squark υλοποιήθηκε ως το προϊόν του (συγκεκριμένου) SquarkFactory, ενώ η Flarn υλοποιήθηκε ως μια απλή συγκεκριμένη κλάση με δημόσιο δομητή προεπιλογής.

Εργασία 4: «PIUtils εργασίας »

Μας ενδιαφέρει η ευχρηστία του προτύπου εργοστάσιο κατά τον εντοπισμό σφαλμάτων, και όχι μόνο κατά την κατασκευή νέων αντικειμένων. Η "PIUtils" εργασία αποτελείται μια προ-γραπτή μέθοδο που επρόκειτο να εμφανίσει δύο παράθυρα διαλόγου στην οθόνη, αυτό που διενεργεί ελέγχους σύμφωνα με τις κατευθυντήριες γραμμές της εμπειρίας των χρηστών των Windows, και το άλλο ανθρώπινη διεπαφή σύμφωνα με τις κατευθυντήριες γραμμές για Macintosh. Οι συμμετέχοντες κλήθηκαν να εντοπίσουν και να διορθώσουν το σφάλμα που εισήχθη στον κώδικα που προκάλεσαν τα δύο παράθυρα διαλόγου..Το σφάλμα οφειλόταν στην πραγματικότητα σε παρερμηνεία του ρόλου της μεθόδου στην κλάση PIDialogLayout, η οποία δόθηκε (με την τεκμηρίωση, αλλά χωρίς τον πηγαίο κώδικα), ως μέρος της εργασίας. Η εργασία PIUtils είχε δύο προϋποθέσεις, εκ των οποίων μόνο μία δόθηκε σε κάθε συμμετέχοντα. Στην εργοστασιακή κατάσταση, η τάξη PIDialogLayout δημιουργήθηκε με το πέρασμα παραμέτρων στη μέθοδο createLayout της διάταξης μιας κλάσης εργοστασίου. Εδώ, το σφάλμα θα μπορούσε να καθοριστεί με το πέρασμα διαφορετικών παραμέτρων στο εργοστάσιο. Στην κατάσταση κατασκευαστή, η τάξη PIDialogLayout ήταν πράγματι εφαρμοσμένη με

μια class cluster, και στιγμιότυπα δημιουργήθηκαν άμεσα χρησιμοποιώντας έναν προεπιλεγμένο δομητή. Σε αυτή την κατάσταση, το σφάλμα θα μπορούσε να καθοριστεί με την κλήση της addOperatingSystem μεθόδου με διαφορετική τιμή.

Εργασία 5: «Έργο υποδοχών»

Το έργο υποδοχών σχεδιάστηκε για να παρουσιάσει ως ρεαλιστική μια εμπειρία όσο το δυνατόν με ένα αληθινό πρότυπο εργοστάσιο από το Java API. Οι συμμετέχοντες έλαβαν εντολή να κατασκευάσουν ένα SSLSocket και ένα MulticastSocket (που ορίζεται στην Java API), τα ρύθμισαν να συνδεθούν σε ένα συγκεκριμένο διακομιστή και θύρα, και να τα περάσουν σε μια μέθοδο που θα εκτελέσει την πραγματική σύνδεση. Η κλάση SSLSocket δεν είναι άμεσα κατασκευασμένη, και θα πρέπει αντ 'αυτού να δημιουργηθεί από την πρώτη λήψη μιας αναφοράς σε μια SSLSocketFactory (η οποία είναι η ίδια μια συγκεκριμένη υποκλάση της κλάσης SocketFactory - ένα χαρακτηριστικό παράδειγμα ενός αφηρημένου πρότυπου εργοστάσιο) και στη συνέχεια καλώντας μια μέθοδο του εργοστάσιο, σε αυτό . Η MulticastSocket, από την άλλη πλευρά, είναι μια συγκεκριμένη υποκλάση των υποδοχών και έχει πολλούς δημόσιους κατασκευαστές.

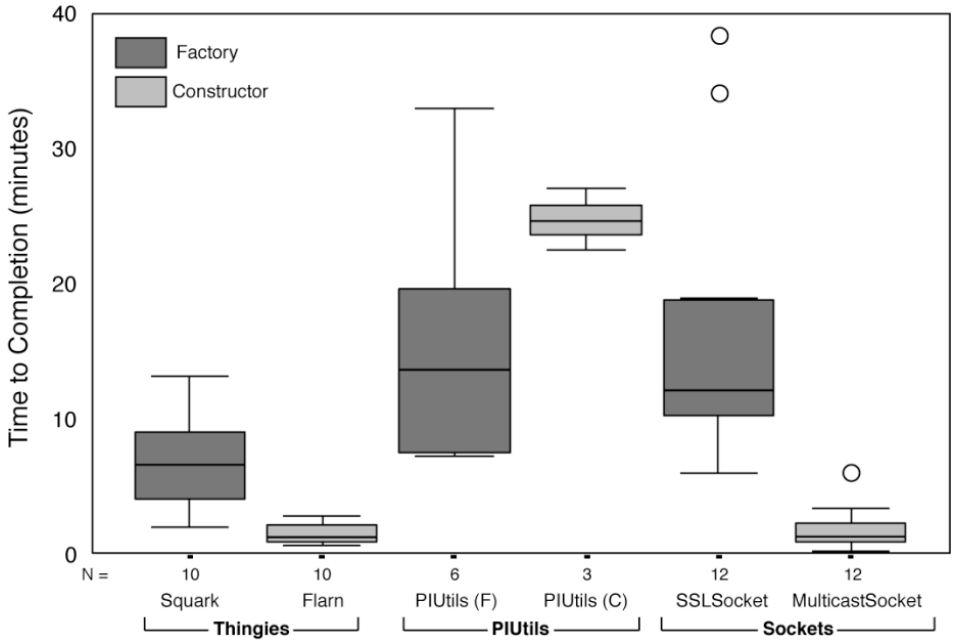
4.2.4 **Αποτελέσματα**

Στην εργασία «Σημειωματάριο αποστολής ηλεκτρονικού ταχυδρομείου» όλοι οι συμμετέχοντες χρησιμοποίησαν μια κλήση δομητή στην εφαρμογή της μεθόδου τους. Τρεις ξεχωριστές υποκλάσεις δημιουργήθηκαν για κάθε τύπο του ηλεκτρονικού ταχυδρομείου (εμπλουτισμένου κειμένου και απλού κειμένου), ενώ οι δύο πέρασαν τον τύπο του ηλεκτρονικού ταχυδρομείου ως παράμετρο, και επτά χρησιμοποίησαν μια μέθοδο ρυθμιστή σε ένα ήδη κατασκευασμένο αντικείμενο. Κανένας δεν χρησιμοποίησε, ή δήλωσε, ένα εργοστάσιο κατά τη διάρκεια της εργασίας.

Στη δεύτερη εργασία (Eclipse Αποστολή ηλεκτρονικού ταχυδρομείο) δύο από τους δώδεκα συμμετέχοντες ορίστηκαν τυχαία να κάνουν τελευταία την εργασία αυτή, και δεν είχαν αρκετό χρόνο για να το ξεκινήσουν (έτσι n = 10). Από εκείνους που έκαναν το έργο, επτά συμμετέχοντες επιχείρησαν να χρησιμοποιήσουν έναν κατασκευαστή, καταλήγοντας στο συμπέρασμα ότι δεν υπήρξε δημόσιος κατασκευαστής. Τρεις από αυτούς τους συμμετέχοντες, προσπάθησαν να δημιουργήσουν μια συγκεκριμένη υποκλάση της αφηρημένης κλάσης E-mail. Όλοι οι δέκα τελικά βρήκαν και χρησιμοποίησαν επιτυχώς το εργοστάσιο, έστω και αν όλοι τους είχαν χρησιμοποιήσει μια κλήση κατασκευαστή στην υποθετική εφαρμογή τους κατά τη διάρκεια της εργασίας.

Στην επόμενη εργασία (Thingies) δύο συμμετέχοντες δεν έχουν χρόνο για να ξεκινήσουν το έργο(n = 10). Όλοι εκείνοι που έφτασαν το έργο αυτό ολοκληρώθηκε με επιτυχία. Ο μέσος χρόνος για την κατασκευή ενός Squark (χρησιμοποιώντας ένα εργοστάσιο) ήταν 7:10 (λεπτά: δευτερόλεπτα, SD = 3:53). Ο μέσος χρόνος για την κατασκευή ενός Flarn (χρησιμοποιώντας ένα κατασκευαστή) ήταν 1:20 (SD = 0:50). Κατά μέσο όρο, οι συμμετέχοντες ξόδεψαν το 84,3% του χρόνου τους για την κατασκευή αντικειμένων για την κατασκευή Squark, σε σύγκριση με το 15,7% του χρόνου εργασίας για την κατασκευή Flarn. Τα δεδομένα του χρόνου εξετάστηκαν για την ομαλότητα, και παρόλο που οι αποκλίσεις από το φυσιολογικό δεν ήταν σημαντικές (p = 0,274 για Squark, p = 0,129 για Flarn), τα δεδομένα ήταν αρκετά στραβά που χρησιμοποιήσαμε τον έλεγχο Wilcoxon Signed Ranks. Ιδιαίτερα σημαντικές διαφορές βρέθηκαν μεταξύ του χρόνου για να ολοκληρωθεί το τμήμα της εργασίας Squarks (χρησιμοποιώντας ένα εργοστάσιο) και το τμήμα του έργου Flarns (με τη χρήση του κατασκευαστή), με ένα Z-score της -2.81 (p = 0,005).

Η παρακάτω εικόνα συνοψίζει τους χρόνους :



Σχήμα 21: Χρόνοι για την ολοκλήρωση των εργασιών

Στην προτελευταία εργασία από τους δώδεκα συμμετέχοντες, τρεις δεν είχαν αρκετό χρόνο για να ξεκινήσουν το έργο (n = 9). Όλοι οι υπόλοιποι ολοκλήρωσαν την εργασία με επιτυχία. Από αυτούς, τρεις ήταν στην κατάσταση κατασκευαστή, και οι έξι ήταν στην εργοστασιακή κατάσταση. (Αυτή η διαφορά ήταν το αποτέλεσμα μιας ατυχούς σύμπτωση. Και οι τρεις συμμετέχοντες, οι οποίοι δεν τους έμεινε χρόνος για να εκτελέσουν το έργο PIUtils είχαν επιλεγεί τυχαία για την κατάσταση του κατασκευαστή). Ο μέσος χρόνος για την ολοκλήρωση της κατάστασης του κατασκευαστεί ήταν 26:40 (SD = 2:26), και 17:00 στην κατάσταση εργοστασίου(SD = 10:26). Δεν βρέθηκαν στατιστικά σημαντικές διαφορές μεταξύ των δύο καταστάσεων (F = 2,35, p = 0,169). Ενώ τα στοιχεία για το έργο αυτό δείχνουν μια γενική τάση προς μεγαλύτερους χρόνους για την κατάσταση του κατασκευαστή, την έλλειψη στατιστικής σημαντικότητας και την πολύ υψηλή τυπική απόκλιση της κατάστασης του εργοστασίου καθιστά δύσκολο να πούμε αν αυτό είναι ένα κατασκεύασμα του δείγματος ή μια πραγματική τάση. Παρ 'όλα αυτά, το ερώτημα της ευκολίας του προτύπου εργοστασίου του εντοπισμού σφαλμάτων σε σχέση με τους κατασκευαστές μπορεί να είναι μια λεωφόρος για τις μελλοντικές εργασίες. Τέλος, στην εργασία «έργο υποδοχών» και οι δώδεκα συμμετέχοντες ξεκίνησαν το έργο. Από αυτούς, πέντε συμμετέχοντες ήταν ανίκανοι να ολοκληρώσουν το έργο πριν από το τέλος της μελέτης. Τα αποτελέσματα αυτών των συμμετεχόντων

καταγράφηκαν χρησιμοποιώντας το συνολικό χρόνο παρά τον χρόνο για την ολοκλήρωσή της. Υπήρχαν πολύ σημαντικές διαφορές μεταξύ του χρόνου για την εκτέλεση της SSLSocket υποεργασίας (χρησιμοποιώντας ένα εργοστάσιο) και της MulticastSocket υποεργασίας (χρησιμοποιώντας έναν κατασκευαστή), με ένα Z-score της -2,803 (p = 0,005).

Το πιο εντυπωσιακό αποτέλεσμα αυτής της μελέτης είναι ότι τα εργοστάσια είναι αποδεδειγμένα πιο δύσκολα από ό, τι οι κατασκευαστές για τους προγραμματιστές . Τόσο τα αποτελέσματα υποδοχών όσο και τα αποτελέσματα των Thingies δείχνουν ότι είναι μια πολύ σημαντική διαφορά στο χρόνο που απαιτείται για να κατασκευαστεί ένα αντικείμενο χρησιμοποιώντας ένα εργοστάσιο αντί χρησιμοποιώντας ένα δομητή. Η διαφορά αυτή είναι ιδιαίτερα σημαντική, επειδή υπονοεί ότι δεν έχει σημασία αν το εργοστάσιο παρουσιάζεται σε ένα κενό ή εφαρμόζεται για ένα συγκεκριμένο πλαίσιο. Όλα τα άτομα βρήκαν το πρότυπο κατασκευαστή πιο "φυσικό", δεδομένου ότι ανέμειναν ότι είναι ο τρόπος να δημιουργηθούν τα αντικείμενα, και αυτή ήταν η πρώτη τεχνική που δοκίμασαν. Δεδομένου ότι πολλά από τα οφέλη των εργοστασίων μπορούν να επιτευχθούν με εναλλακτικές λύσεις που δεν επιβαρύνονται με την ίδια ποινή δυνατότητας χρηστικότητας, τα αποτελέσματα της μελέτης αυτής δείχνουν ότι τέτοιες εναλλακτικές λύσεις είναι συχνά προτιμότερες στα εργοστάσια.

Συμπερασματικά για τους έμπειρους χρήστες με το πρότυπο εργοστάσιο, η υποτιθέμενη υπεροχή του εργοστασίου υποστηρίζεται από επίσημη εργασία τους και ως εκ τούτου θεωρείται πιο «ασφαλής». Για λιγότερο έμπειρους χρήστες, η ίδια η πολυπλοκότητα του εργοστασίου μπορεί να ήταν ένα ελκυστικό χαρακτηριστικό, επειδή μπορεί να έχουν ερμηνεύσει την πολυπλοκότητα του σχεδιασμού ως απόδειξη των προηγμένων βασικών ιδεών - ένας ικανός προγραμματιστής σχεδιάζοντας και κατανοώντας ώστε η πολυπλοκότητα πρέπει να είναι γνώστες και έμπειροι, η αιτιολογία θα μπορούσε να πάει, και επομένως είναι πιο πιθανό να γνωρίζουν καλύτερα τι είναι καλό και κακό. Ωστόσο τα αποτελέσματά μας δείχνουν σημαντικές αρνητικές επιπτώσεις στην πραγματική ικανότητα των προγραμματιστών να χρησιμοποιούν τα API.

4.3 ΚΙΝΔΥΝΟΙ ΕΓΚΥΡΟΤΗΤΑΣ

Στο σημείο αυτό συζητάμε τους πιθανούς κινδύνους εγκυρότητας της έρευνάς μας. Όσον αφορά τη διαδικασία αναζήτησης, οποιαδήποτε μελέτη δεν ανέφερε τις λέξεις «API»,«Framework»,«Library» στον τίτλο του άρθρου της, καθώς και την λέξη «πρότυπο» στην περίληψη ή σε όλο το κείμενο, δε συμπεριλήφθηκε στο σύνολο των πρωταρχικών ερευνών. Συνεπώς, μπορεί να έχει παραλειφθεί, ένας μικρός αριθμός σχετικών άρθρων, αν και πιστεύουμε ότι τα άρθρα που ασχολούνται με τα πρότυπα σχεδίασης το πιθανότερο θα το αναφέρουν και στον τίτλο τους. Επιπλέον, το γεγονός ότι δεν κάναμε μια γενική αναζήτηση σε ένα σύστημα δεικτοδότησης όπως τα SCOPUS, EI COMPENDIX ή το Web of Science, συνεπάγεται ότι άρθρα σε λιγότερο γνωστά περιοδικά μπορεί να έχουν παραλειφθεί από την μελέτη. Παρόλα αυτά, εμείς θεωρούμε ότι συμπεριλαμβάνοντας στην ανασκόπηση μόνο κορυφαία περιοδικά, συνέδρια και workshops, αυξάνονται τα ποιοτικά κριτήρια των πρωταρχικών μελετών και έτσι εξασφαλίζεται η ποιότητα της συστηματικής μας ανασκόπησης.

4.4 ΣΥΜΠΕΡΑΣΜΑΤΑ

Τα αποτελέσματα από την βιβλιογραφική ανασκόπηση που κάναμε, δεν δίνουν απάντηση στο ερώτημα της έρευνας μας (RQ1). Θεωρώντας λοιπόν ότι Δεν υπάρχουν μελέτες που αναφέρονται στο βαθμό χρήσης των προτύπων σχεδίασης στις βιβλιοθήκες, προχωράμε στο επόμενο βήμα που είναι η εμπειρική μελέτη πάνω στο θέμα που μας απασχολεί.

ΚΕΦΑΛΑΙΟ 5 - ΕΜΠΕΙΡΙΚΗ ΜΕΛΕΤΗ ΤΗΣ ΧΡΗΣΗΣ ΠΡΟΤΥΠΩΝ ΣΧΕΔΙΑΣΗΣ ΣΤΙΣ ΒΙΒΛΙΟΘΗΚΕΣ

Προκειμένου να εκτελέσουμε την επιστημονική έρευνα στην τεχνολογία λογισμικού, πρέπει να κατανοήσουμε τις μεθόδους που έχουμε στη διάθεσή μας, τα όριά τους και πότε μπορούν να εφαρμοστούν. Σύμφωνα με το [Glass94], συνοψίζονται τέσσερις μέθοδοι έρευνας στον τομέα της μηχανικής λογισμικού.

Οι μέθοδοι είναι οι εξής:

- 1) Επιστημονική μέθοδος (scientific method)
- 2) Μηχανική μέθοδος (engineering method)
- 3) Εμπειρική μέθοδος (empirical method)
- 4) Αναλυτική μέθοδος (analytical method)

Στην επιστημονική μέθοδο παρατηρείται το περιβάλλον και δημιουργείται ένα μοντέλο με βάση την παρατήρηση, για παράδειγμα, ένα μοντέλο προσομοίωσης. Στην μηχανική μέθοδο οι υπάρχουσες λύσεις μελετώνται και προτείνονται οι αλλαγές, οι οποίες στη συνέχεια αξιολογούνται. Στην εμπειρική μέθοδο προτείνεται ένα μοντέλο και στη συνέχεια αξιολογείται με εμπειρικές μελέτες, όπως για παράδειγμα, μελέτες περιπτώσεων ή πειράματα. Τέλος, στην αναλυτική μέθοδο προτείνεται μια επίσημη θεωρία, συλλέγονται εμπειρικές παρατηρήσεις και στη συνέχεια με βάση τις παρατηρήσεις συγκρίνεται η προς διερεύνηση θεωρία. Η μηχανική μέθοδος και η εμπειρική μέθοδος μπορούν να θεωρηθούν ως παραλλαγές της επιστημονικής μεθόδου [Basili 1993]. Κάθε μία από αυτές τις μεθόδους κρίνεται ως καταλληλότερη και εφαρμόζεται σε κάποιο συγκεκριμένο τομέα.

Σύμφωνα με τους συντάκτες του [16], υπάρχουν τρεις μεγάλες ερευνητικές μέθοδοι που χρησιμοποιούνται στις εμπειρικές μελέτες οι οποίες είναι οι εξής:

- μελέτη περίπτωσης (case study).
- έρευνα πεδίου(survey),
- τυπικό ή ελεγχόμενο πείραμα (formal experiment).

Η ενότητα αυτή, στοχεύει στην διερεύνηση της χρήσης αντικειμενοστρεφών προτύπων σχεδίασης στις βιβλιοθήκες. Ειδικότερα χρησιμοποιήσαμε μια εμπειρική μέθοδο, δηλ. μια μελέτη περίπτωσης ώστε να αξιολογήσουμε ποια πρότυπα χρησιμοποιούνται πιο συχνά στις βιβλιοθήκες.

5.1 Μεθοδολογία

Λαμβάνοντας υπόψη τη φύση, το αντικείμενο της έρευνάς και την πληθώρα των διαθέσιμων έργων ανοικτού λογισμικού, πιστεύουμε ότι μια μελέτη περίπτωσης είναι η πιο κατάλληλη για τις ανάγκες της έρευνας μας. Η μελέτη περίπτωσης της έρευνάς ήταν σύμφωνα με τις κατευθυντήριες γραμμές που περιγράφονται στο [B. Kitchenham, 1995]. Σύμφωνα με το [B. Kitchenham, 1995], τα βήματα για τη διεξαγωγή μιας μελέτης περίπτωσης περιλαμβάνουν :

(α) Καθορισμός υποθέσεων
(β) Επιλογή υποκειμένων έρευνας
(γ) Μέθοδος επιλογής-σύγκρισης
(δ) Ελαχιστοποίηση παραγόντων εκτός των ανεξαρτήτων μεταβλητών
(ε) Σχεδιασμός της μελέτης περίπτωσης
(στ) Παρακολούθηση της μελέτης περίπτωσης και
(ζ) Ανάλυση και αναφορά των αποτελεσμάτων

5.2 Τα ερωτήματα της έρευνας

Στην ενότητα αυτή θέτουμε τα ερωτήματα που ερευνούμε στη μελέτη μας:

RQ1: Ποια πρότυπα σχεδίασης χρησιμοποιούνται συχνότερα στις βιβλιοθήκες (API);

RQ2: Υπάρχει στατιστικά σημαντική διαφορά μεταξύ των κατηγοριών λογισμικού API και Standalone;

5.3 Πλάνο της μελέτης περίπτωσης

Σύμφωνα με το (Basili et al, 1986), προκειμένου να δημιουργήσουμε μια σωστή μεθοδολογία για μια εμπειρική μέθοδο επιβεβαίωσης, πρέπει να φτιάξουμε προσεκτικά ένα πλάνο μελέτης. Στη συγκεκριμένη μελέτη περίπτωσης το πλάνο αυτό αποτελείται από μια διαδικασία πέντε βημάτων:

1. Επιλογή των κατηγοριών του λογισμικού ανοιχτού κώδικα που θα μελετήσουμε
2. Εύρεση ενός αριθμού εφαρμογών, που πληρούν τα κριτήρια επιλογής, για κάθε κατηγορία
3. Ανίχνευση των προτύπων σχεδίασης που χρησιμοποιούνται σε κάθε λογισμικό που έχει επιλεγεί
4. Σύνοψη των δεδομένων
5. Ανάλυση των δεδομένων όσον αφορά τα ερωτήματα της έρευνας.

Στη μελέτη αυτή, οι εφαρμογές που επιλέξαμε είναι έργα ανοιχτού λογισμικού που χωρίζονται σε 2 επιμέρους κατηγορίες: 1) Api 2) Standalone Software. Για κάθε κατηγορία τα λογισμικά που χρησιμοποιήθηκαν ήταν των παρακάτω θεματικών ενοτήτων: 1)Development 2) Audio Video 3) Business Enterprise 4) Communications 5)Graphics. 6)Home/Education Οι κατηγορίες αυτές επιλέχθηκαν ως οι δημοφιλέστερες.

Από τις κατηγορίες αυτές επιλέξαμε λογισμικά που πληρούν τα εξής κριτήρια:

- Είναι γραμμένα σε java, σύμφωνα με τους περιορισμούς του εργαλείου που χρησιμοποιούμε για την ανίχνευση των προτύπων (Tsantalis et al., 2006).

- Διαθέτουν δυαδικό κώδικα, σύμφωνα με τους περιορισμούς του εργαλείου που χρησιμοποιούμε για την ανίχνευση των προτύπων
- Πρόκειται για λογισμικά που έχουν τουλάχιστον 100 λήψεις, έτσι ώστε να μπορούμε να τα θεωρήσουμε ενεργά.

Στα Παραρτήματα στο τέλος του άρθρου υπάρχει μια πλήρης λίστα των λογισμικών που χρησιμοποιήσαμε σε αυτή την μελέτη περίπτωσης.

Στις μελέτες περίπτωσης, οι παράγοντες πέραν των ανεξάρτητων μεταβλητών, που επηρεάζουν την τιμή της εξαρτημένης μεταβλητής, θεωρούνται συγκεχυμένοι παράγοντες. Μερικοί τέτοιοι παράγοντες που περιμένουμε να επηρεάζουν τα πρότυπα σχεδίασης είναι η προγραμματιστική εμπειρία του προγραμματιστή και το μορφωτικό επίπεδο του προγραμματιστή όσον αφορά το αντικείμενο της μηχανικής λογισμικού. Τέτοιες πληροφορίες ωστόσο δεν είναι δυνατό να μελετηθούν σε μια μελέτη περίπτωσης, όπου τα δεδομένα που αφορούν την έρευνα συλλέγονται μέσω παρατήρησης. Κάτι τέτοιο θα ήταν εφικτό σε ένα ελεγχόμενο πείραμα (Wohlin et al., 2000). Από την άλλη πλευρά, αναμένεται ότι σε ένα τυχαίο δείγμα προγραμματιστών μιας μεγάλης προγραμματιστικής κοινότητας, η κατανομή εκείνων που έχουν προγραμματιστική ικανότητα και εμπειρία πλησιάζει κατά πολύ την κατανομή του πληθυσμού.

5.4 Μέθοδοι Ανάλυσης Δεδομένων

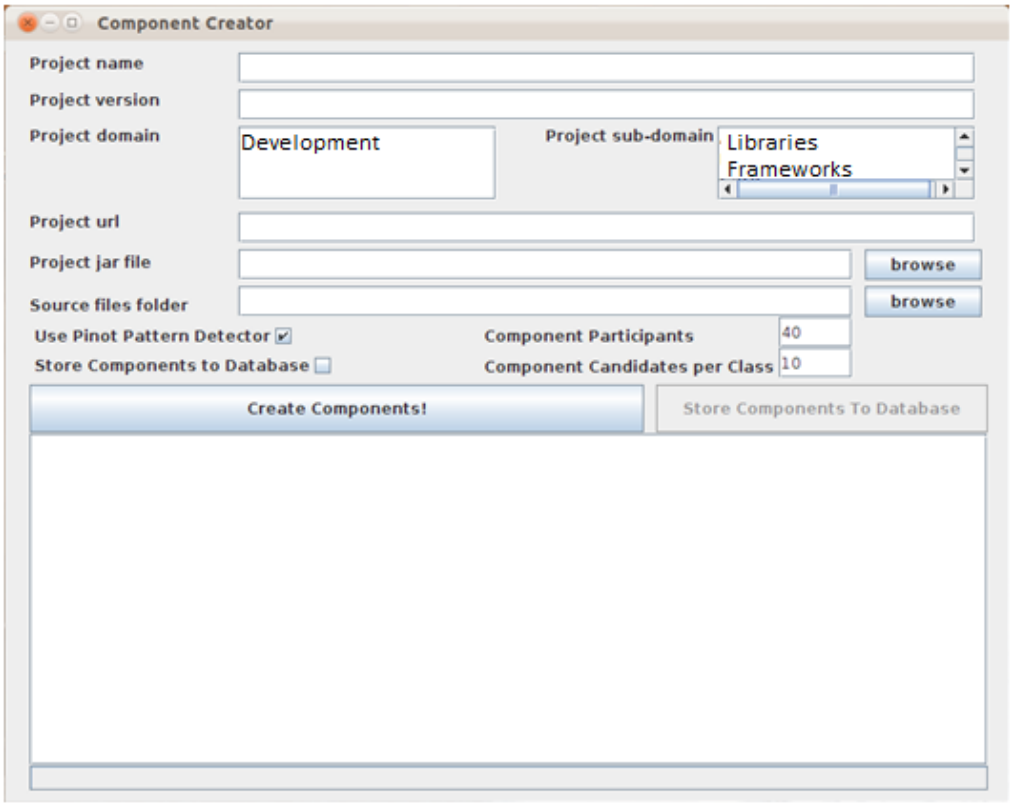
Το επόμενο βήμα μετά την συγκέντρωση των λογισμικών είναι η ανάλυση των δεδομένων. Προκειμένου να γίνει αυτό, χρησιμοποιήθηκε η πτυχιακή εργασία του απόφοιτου φοιτητή του τμήματός μας, Γκορτζή Αντώνιου, με θέμα «Στρατηγική Επιλογής κλάσεων με στόχο την βελτιστοποίηση της ποιότητας του επιλεγμένου κώδικα».

Προκειμένου τα δεδομένα του προγράμματος να συμβαδίζουν με την μελέτη που αναλύουμε, έγιναν από τον προγραμματιστή, οι απαραίτητες τροποποιήσεις σε αυτό. Οι αλλαγές αφορούσαν κυρίως τις θεματικές ενότητες που χρησιμοποιήθηκαν αλλά και στις προδιαγραφές του προγράμματος (κυρίως την μνήμη) ώστε να είναι συμβατό με τον Ηλεκτρονικό Υπολογιστή που έτρεξε αυτό το λογισμικό.

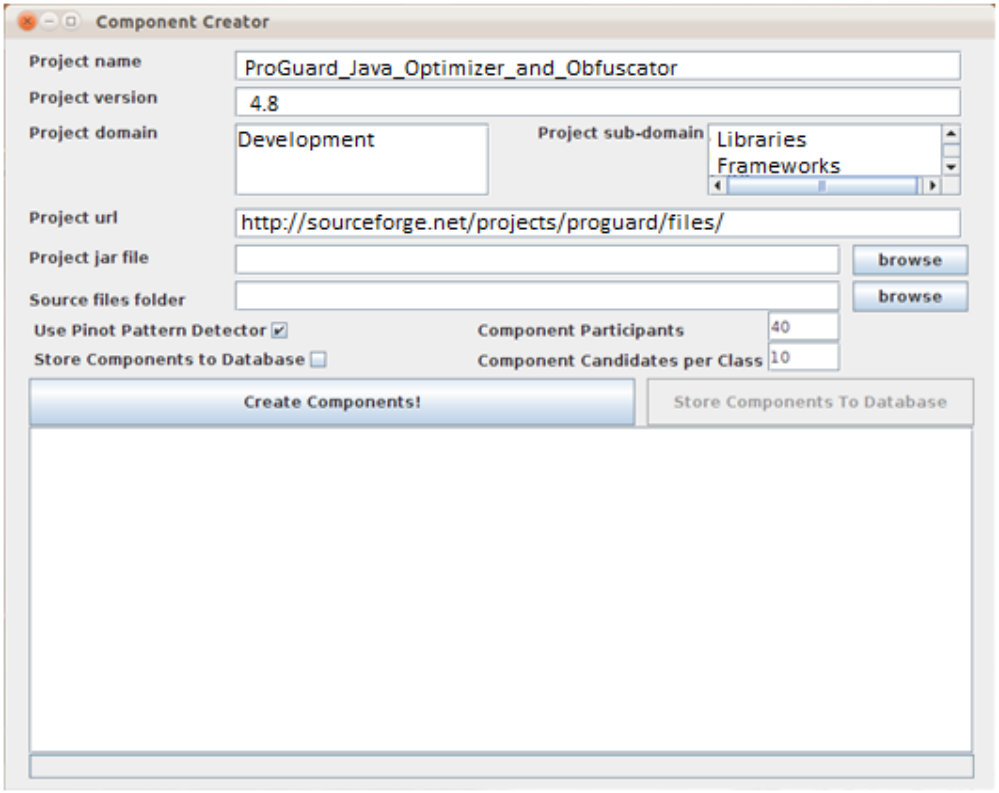
Στην συγκεκριμένη πτυχιακή για την ανίχνευση και την εξαγωγή προτύπων από τα συστήματα ανοιχτού λογισμικού, με σκοπό την παραγωγή «υποψήφιων συστατικών (components candidates)», χρησιμοποιήθηκαν δύο εργαλεία υλοποιούν δύο διαφορετικές προσεγγίσεις. Το εργαλείο «Design Pattern Detection Using Similarity Scoring» είναι γραμμένο στην γλώσσα προγραμματισμού Java και έχει την ικανότητα να αναγνωρίζει τα πρότυπα Προσαρμογέας, Σύνθετο, Διακοσμητής, Μέθοδος Εργοστάσιο, Παρατηρητής, Πρωτότυπο, Μοναδιαίο, Πληρεξούσιο, Κατάσταση/Στρατηγική, Μέθοδος Υπόδειγμα και επισκέπτης μελετώντας τον μεταγλωττισμένο κώδικα (bytecode) εφαρμογών γραμμένων στην Java. Οι συγγραφείς που το ανέπτυξαν προτείνουν μια μεθοδολογία, που βασίζεται στην καταγραφή των ομοιοτήτων μεταξύ των κορυφών ορισμένων γραφικών παραστάσεων (Tsantalis et al., 2006). Η αξιολόγηση σε τρία προγράμματα ανοιχτού λογισμικού κάνει εμφανή την ακρίβεια και την αποδοτικότητα της προτεινόμενης μεθόδου.

Το δεύτερο εργαλείο «Pattern Inference and Recovery Tool (PINOT)», ένα πρωτότυπο εργαλείο που χρησιμοποιείται για την εφαρμογή μιας νέας, πλήρως αυτοματοποιημένης προσέγγιση ανίχνευσης προτύπων (Shi και Olsson, 2006), που βασίζεται σε μια νέα επαναταξινόμηση των GoF προτύπων σύμφωνα με τις προθέσεις του καθενός, η οποία λέγεται ότι ταιριάζει καλύτερα στην αντίστροφη μηχανική.Το PINOT ανιχνεύει όλα τα πρότυπα GoF που έχουν σαφείς ορισμούς καθοδηγούμενους από τη δομή του κώδικα ή τη συμπεριφορά του συστήματος και είναι ένα πλήρως αυτοματοποιημένο εργαλείο ανίχνευσης προτύπων που είναι γρηγορότερο, ακριβέστερο, και περιεκτικότερο από τα υπάρχοντα εργαλεία. Το μεγαλύτερο μέρος της ανάπτυξής του έγινε σε γλώσσα C++ ενώ κάποιες λειτουργίες υλοποιούνται στις γλώσσες Java και Perl. Τέλος, το εργαλείο επεξεργάζεται τον πηγαίο κώδικα (source code) εφαρμογών γραμμένων σε Java.

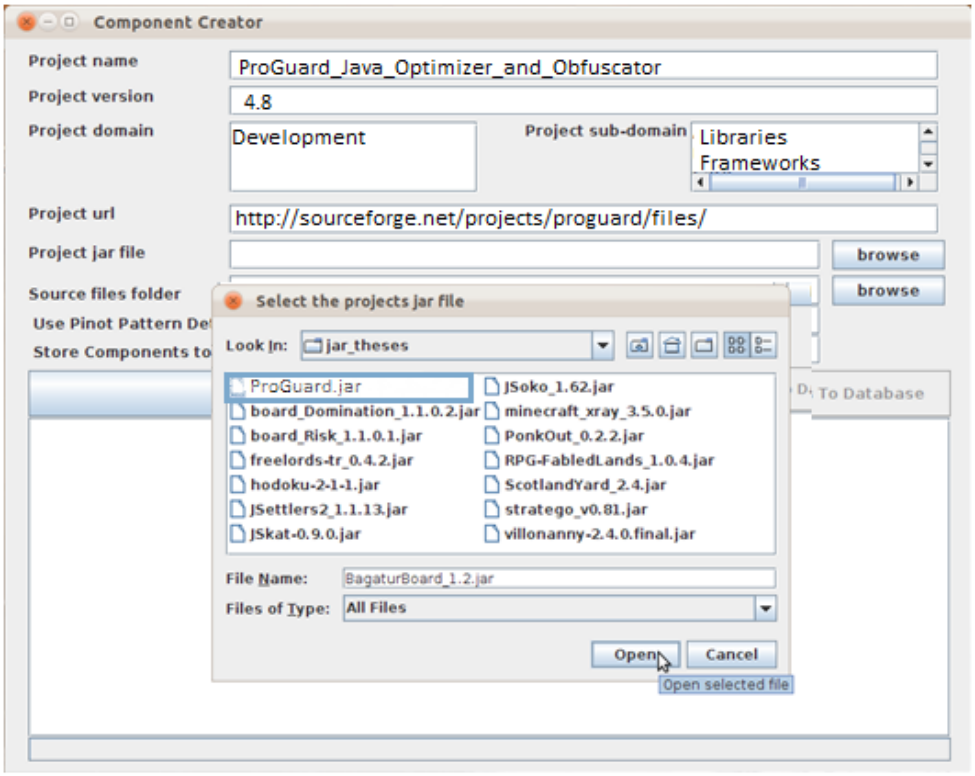
Στην συνέχεια παρουσιάζονται κάποια στιγμιότυπα οθόνης (screen shots) από την εκτέλεση του εργαλείου. Τα στιγμιότυπα αυτά αποθηκεύτηκαν κατά τη διάρκεια ανάλυσης του Project " ProGuard Java Optimizer and Obfuscator 4.8 " με αριθμό κλάσεων 142 από το οποίο παράχθηκαν 24.312 "υποψήφια συστατικά" εκ των οποίων 55 βασίζονται σε πρότυπα σχεδίασης.



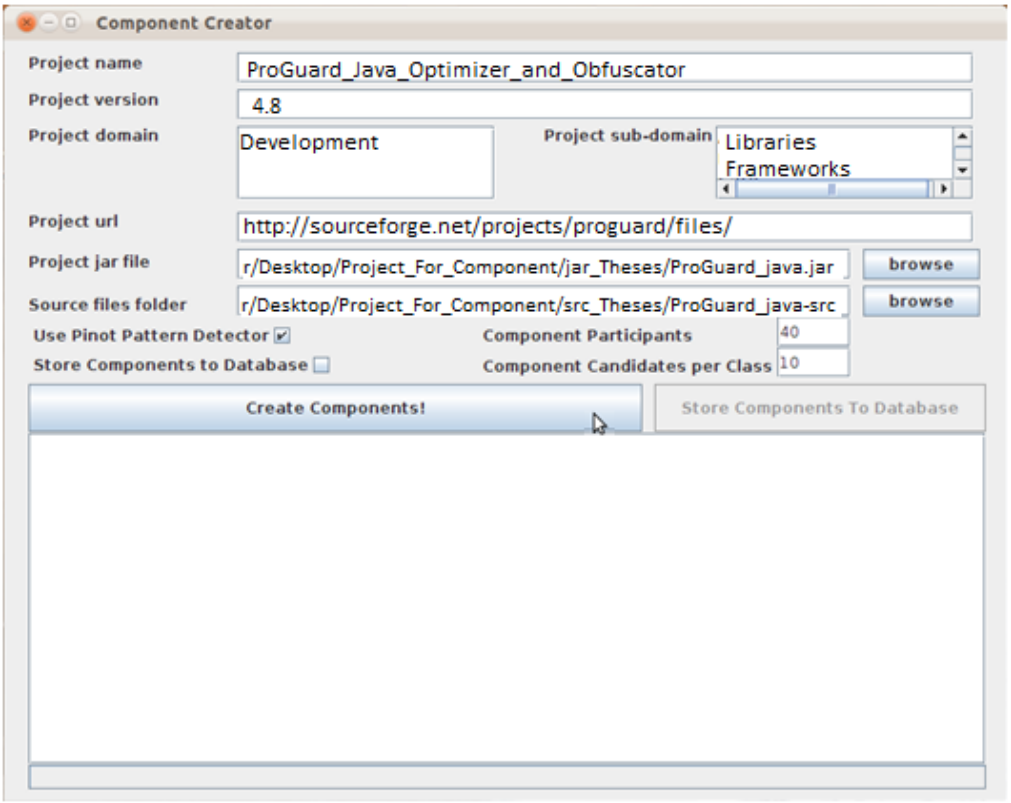
Σχήμα 22: Στιγμιότυπο από την εκτέλεση του εργαλείου



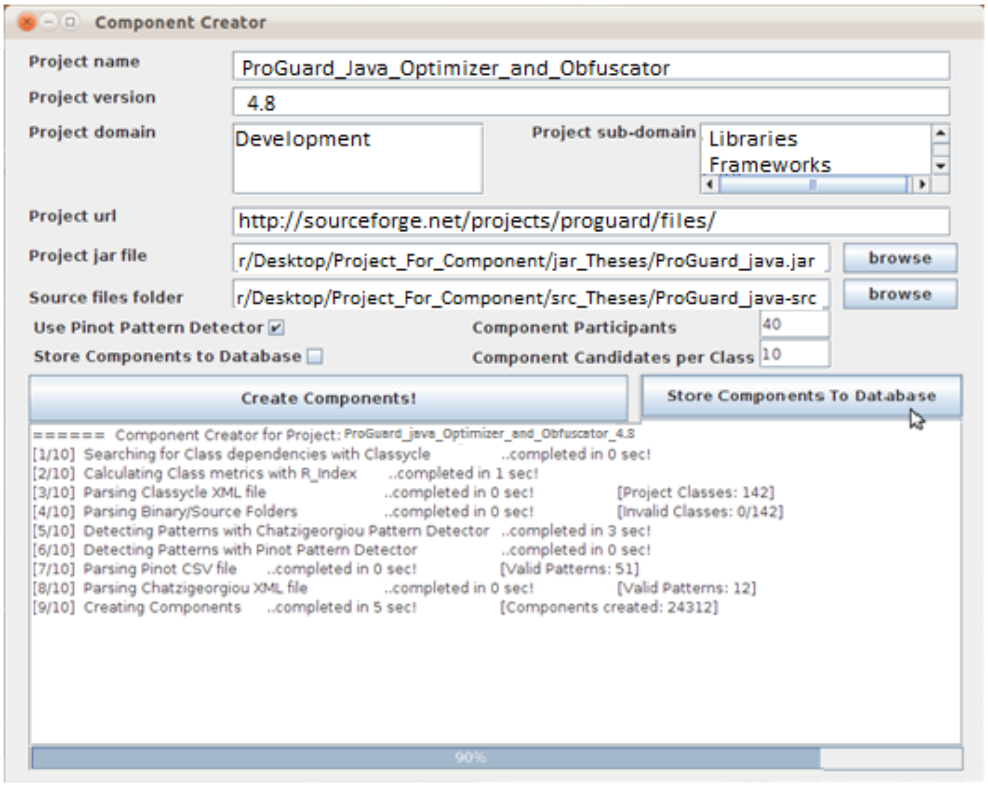
Σχήμα 23: Στιγμιότυπο κατά την εκτέλεση της ανάλυσης του project "ProGuard Java Optimizer and Obfuscator"



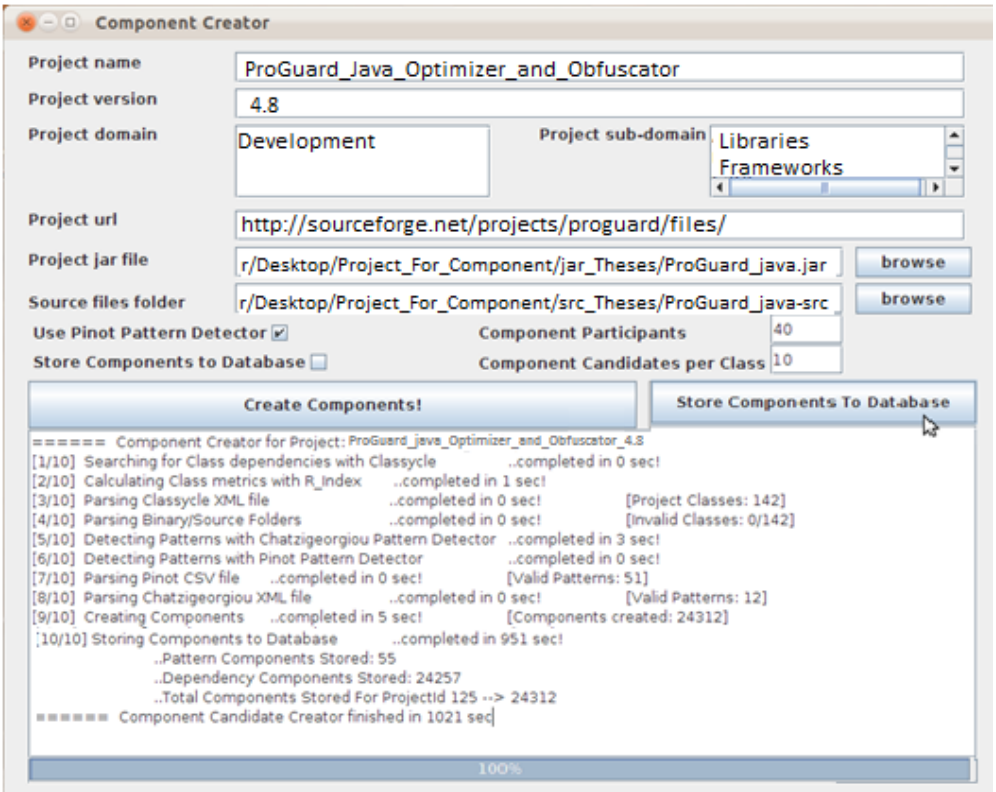
Σχήμα 24: Στιγμιότυπο κατά την εκτέλεση της ανάλυσης του project "ProGuard Java Optimizer and Obfuscator"



Σχήμα 25: Στιγμιότυπο κατά την εκτέλεση της ανάλυσης του project "ProGuard Java Optimizer and Obfuscator"



Σχήμα 26: Στιγμιότυπο κατά την εκτέλεση της ανάλυσης του project "ProGuard Java Optimizer and Obfuscator"

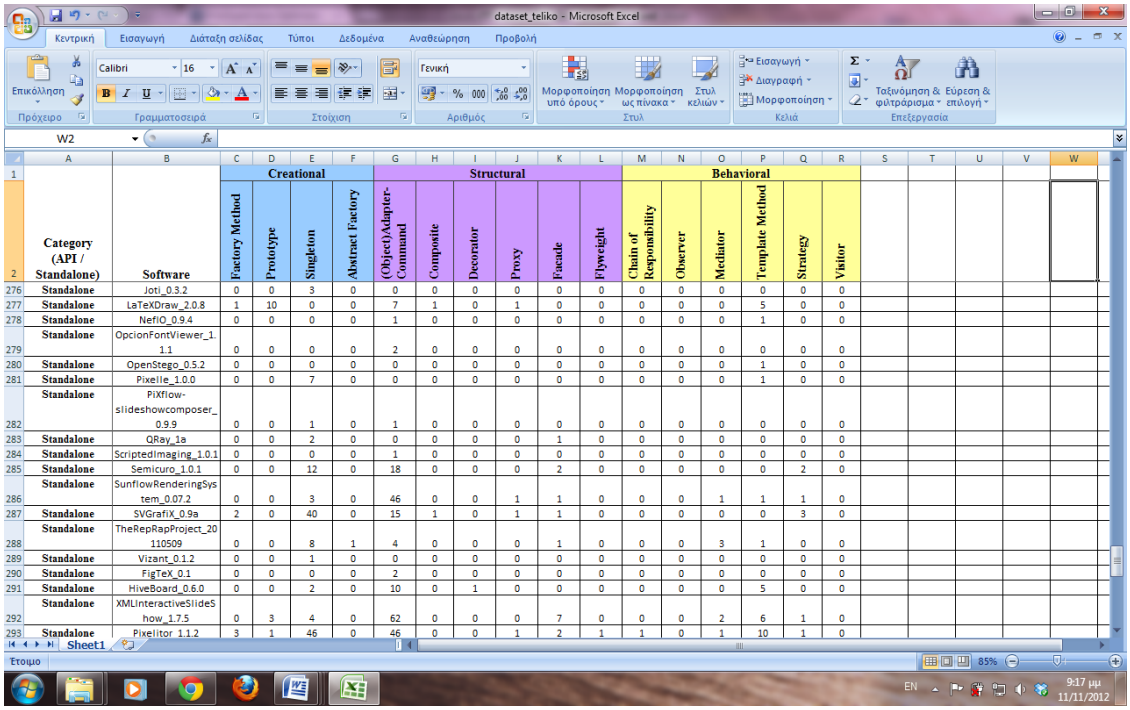


Σχήμα 27: Στιγμιότυπο κατά την εκτέλεση της ανάλυσης του project "ProGuard Java Optimizer and Obfuscator"

Η παραπάνω διαδικασία έτρεξε και για τα 291 λογισμικά που χρησιμοποιήθηκαν στην μελέτης μας. Τα αποτελέσματα από αυτό το εργαλείο αποθηκευόταν σε ένα αρχείο τύπου .sql (πχ Development_components.sql). Στην συνέχεια μέσω του προγράμματος mysql και τρέχοντας, για κάθε λογισμικό ξεχωριστά, την εντολή:

SELECT patternType, count(id) from Development_Components where projectId=xxx and patternType is not null group by patternType,

συγκεντρώσαμε τα patterns που υπήρχαν σε κάθε Software. Όπως φαίνεται και στο παρακάτω στιγμιότυπο εικόνας, σε ένα αρχείο excel σημειώθηκε ο ακριβής αριθμός των πρότυπων που χρησιμοποιούνται σε κάθε λογισμικό, ενώ στα πρότυπα που δεν χρησιμοποιούνται ο αριθμός αυτός ήταν το μηδέν (0).

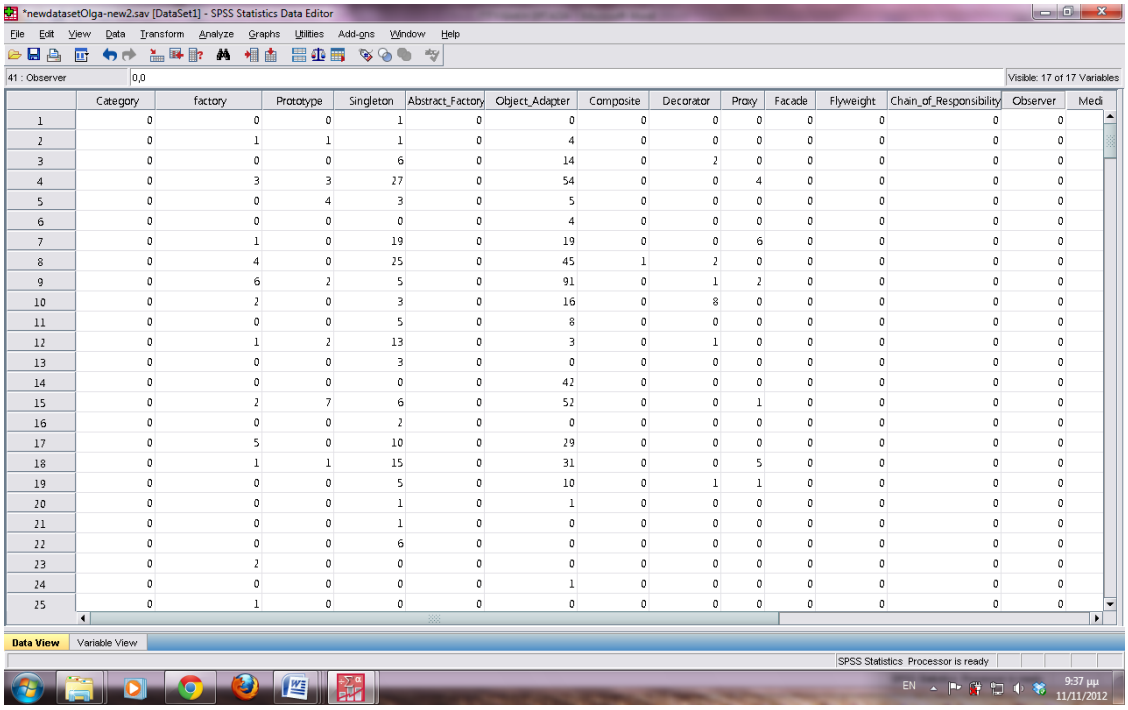


	Category (API / Standalone)	Software	Creational				Structural						Behavioral					
			Factory Method	Prototype	Singleton	Abstract Factory	(Object)Adapter-Command	Composite	Decorator	Proxy	Facade	Flyweight	Chain of Responsibility	Observer	Mediator	Template Method	Strategy	Visitor
276	Standalone	Joti_0.3.2	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0
277	Standalone	LaTeXDraw_2.0.8	1	10	0	0	7	1	0	1	0	0	0	0	0	5	0	0
278	Standalone	NetIO_0.9.4	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
279	Standalone	OpcionFontViewer_1.1.1	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
280	Standalone	OpenStego_0.5.2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
281	Standalone	Pixelle_1.0.0	0	0	7	0	0	0	0	0	0	0	0	0	0	1	0	0
282	Standalone	PiXflow-slideshowcomposer_0.9.9	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
283	Standalone	QRay_1a	0	0	2	0	0	0	0	0	1	0	0	0	0	0	0	0
284	Standalone	ScriptedImaging_1.0.1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
285	Standalone	Semicuro_1.0.1	0	0	12	0	18	0	0	0	2	0	0	0	0	0	2	0
286	Standalone	SunflowRenderingSystem_0.07.2	0	0	3	0	46	0	0	1	1	0	0	0	1	1	1	0
287	Standalone	SVGrafiX_0.9a	2	0	40	0	15	1	0	1	1	0	0	0	0	0	3	0
288	Standalone	TheRepRapProject_20110509	0	0	8	1	4	0	0	0	1	0	0	0	3	1	0	0
289	Standalone	Vizant_0.1.2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
290	Standalone	FigTeX_0.1	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
291	Standalone	HiveBoard_0.6.0	0	0	2	0	10	0	1	0	0	0	0	0	0	5	0	0
292	Standalone	XMLInteractiveSlideShow_1.7.5	0	3	4	0	62	0	0	0	7	0	0	0	2	6	1	0
293	Standalone	Pixelitor_1.1.2	3	1	46	0	46	0	0	1	2	1	1	0	1	10	1	0

Σχήμα 28: Στιγμιότυπο του excel dataset αρχείου"

Κατά τη τελευταία φάσης της ανάλυσης στην έρευνά μας, χρησιμοποιήσαμε το πρόγραμμα SPSS 17 προκειμένου να βγάλουμε τα στατιστικά αποτελέσματα που μας ενδιέφεραν. Οι τεχνικές που χρησιμοποιήθηκαν είναι:

- Descriptive statistics
- Independent sample t-test



<u>Σχήμα 29: Στιγμιότυπο του spss dataset αρχείου.</u>

(Στην στήλη "Category" όπου «0» θεωρείται API και όπου «1» Standalone)

5.5 <u>ΑΠΟΤΕΛΕΣΜΑΤΑ</u>

Στην ενότητα αυτή, παρουσιάζουμε τα ευρήματα της εμπειρικής μας μελέτης. Τα αποτελέσματα αφορούν ολόκληρο το σύνολο δεδομένων χωρίς διάκριση μεταξύ των κατηγοριών. Για κάθε πρότυπο που συναντήσαμε στο πείραμα μας, εμφανίσαμε 2 πίνακες. Ο πρώτος πίνακας (Group Statistics) μας δείχνει τη μέση τιμή για κάθε πρότυπο σχεδίασης, τον μέγιστο αριθμό προτύπων που εντοπίσθηκαν καθώς και τη τυπική απόκλιση για την κάθε μεταβλητή .

Προκειμένου να απαντηθεί το ερώτημα που αφορά το πρότυπο που χρησιμοποιείται συχνότερα στις βιβλιοθήκες (API), δημιουργήσαμε τους παρακάτω πίνακες, όπου εκεί φαίνεται η μέση τιμή για όλα τα πρότυπα σχεδίασης, καθώς και η τυπική απόκλιση για την κάθε μεταβλητή ανά κατηγορία λογισμικού.

Πίνακας 1 : Συγκεντρωτικός πίνακας Μέσων όρων χρήσης των προτύπων σε λογισμικό «API»

	Mean	Std. Deviation
Factory	0,73	1,609
Prototype	0,40	1,281
Singleton	5,90	9,431
Adapter	**9,04**	**15,514**
Composite	0,05	0,261
Decorator	0,32	1,117
Proxy	0,41	1,400
Observer	0,01	0,094
Abstract Factory	0,18	0,989
Template	2,24	3,622
Visitor	0,05	0,261
Facade	0,96	3,138
Flyweight	0,25	1,927
Chain of Responsibility	0,02	0,132
Mediator	*1,54*	*7,280*
Strategy	0,86	4,087

Πίνακας 2: Συγκεντρωτικός πίνακας Μέσων όρων των προτύπων σε λογισμικό «Standalone»

	Mean	Std. Deviation
Factory	0,45	1,033
Prototype	0,46	1,777
Singleton	6,99	13,379
Adapter	**10,75**	**16,418**
Composite	0,03	0,210
Decorator	0,16	1,197
Proxy	0,48	2,065
Observer	0,01	0,150
Abstract Factory	0,06	0,323
Template	2,47	4,777
Visitor	0,1	0,075
Facade	1,31	4,091
Flyweight	0,02	0,183
Chain of Responsibility	0,02	0,149
Mediator	*1,86*	*7,832*
Strategy	0,53	1,638

Σύμφωνα με αυτούς του πίνακες λοιπόν, βλέπουμε ότι τόσο στα API, όσο και στα Standalone Software τα πρότυπα που χρησιμοποιούνται περισσότερο είναι το πρότυπο του «Προσαρμογέα (Adapter)» με μέσο όρο 9,04 και ακολουθεί το «Μοναδιαίο (Singleton)» πρότυπο με μέσο όρο 5,90. Την τελευταία θέση, σε αυτό το ερώτημα, για την κατηγορία API καταλαμβάνει το πρότυπο «Παρατηρητής (Observer)» με μέσο όρο 0,01, ενώ για την Standalone κατηγορία, το πρότυπο «Επισκέπτης (Visitor)» , με μέσο όρο 0,01.

Όσον αφορά το δεύτερο ερώτημα της ερευνάς μας , δηλαδή για το αν υπάρχει στατιστικά σημαντική διαφορά ανάμεσα στις 2 κατηγορίες λογισμικού (API και Standalone), Χρησιμοποιήσαμε το πρόγραμμα «SPSS 17», (Analyze→ Compare means→ Independent Samples T-Test) και εμφανίσαμε τους πίνακες που φαίνονται παρακάτω. Ο πίνακας που μας ενδιαφέρει για το συγκεκριμένο ερώτημα είναι ο πίνακας «Independent Samples Test». Η μεθοδολογία που ακολουθήσαμε προκειμένου να βγάλουμε τα συμπεράσματα μας ήταν η εξής:

Το t τεστ έχει δύο "κατευθύνσεις". Η μία κατεύθυνση είναι αυτή που δεν μπορούμε να υποθέσουμε ότι οι διακυμάνσεις των δύο δειγμάτων είναι περίπου ίσες και αυτή που μπορούμε να υποθέσουμε ότι είναι ίσες. Ο πίνακας «Independent Samples Test» έχει δύο γραμμές αποτελεσμάτων, η πρώτη αναφέρεται στην περίπτωση που μπορούμε να υποθέσουμε ισότητα των δύο διακυμάνσεων και η δεύτερη στην περίπτωση που δεν μπορούμε να υποθέσουμε ισότητα των δύο διακυμάνσεων. Ο πίνακας είναι χωρισμένος σε δύο κατηγορίες αποτελεσμάτων, η μία αφορά το Levene για την ισότητα των διακυμάνσεων και η άλλη περιέχει τα αποτελέσματα του t test που επιλέξαμε να κάνουμε. Όπως αναφέραμε, ο πίνακας έχει δύο γραμμές αποτελεσμάτων, το αν θα κοιτάξουμε την πρώτη ή τη δεύτερη γραμμή αποτελεσμάτων του t τεστ θα μας το "πει" το τεστ του Levene. Το τεστ του Levene ελέγχει την υπόθεση της ισότητας των δύο διακυμάνσεων και υπολογίζει μία p-value. Αν η p-value είναι μικρότερη του 0.05, απορρίπτεται η υπόθεση της ισότητας των διακυμάνσεων και θα κοιτάξω τη δεύτερη γραμμή αποτελεσμάτων του πίνακα. Αν η p-value για τον έλεγχο της ισότητας των δύο μέσων είναι ίση με μηδέν (Sig. (2-tailed)) ,η μηδενική υπόθεση απορρίπτεται, δηλαδή οι μέσοι των δύο πληθυσμών από τα οποία προήλθαν τα δύο δείγματα λέμε ότι διαφέρουν στατιστικά σημαντικά σε επίπεδο στατιστικής σημαντικότητας α=5% πάντα. Σε αντίθετη περίπτωση , δηλαδή p-value μεγαλύτερη του 0.05, δεν απορρίπτεται. Επομένως, ανάλογα με την p-value (Sig.) του τεστ του Levene, κοιτάζουμε την πρώτη ή τη δεύτερη γραμμή αποτελεσμάτων.

1. Πρότυπο Factory:

Πίνακας 3 : Πίνακες «Group Statistics» και «Independent Samples Test» για το πρότυπο Factory.

Group Statistics

	ApiStandAlone	N	Mean	Std. Deviation	Std. Error Mean
Factory	0	114	0,73	1,609	,151
	1	177	0,45	1,033	,078

Independent Samples Test

factory		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
	Equal variances assumed	7,452	**,007**	1,783	289	,076	,276	,155	-,029	,581
	Equal variances not assumed			1,629	173,132	**,105**	,276	,170	**-,059**	**,611**

Για το πρότυπο Factory:

Sig=0,007<0.05 οι διασπορές δεν έχουν διαφορά, θεωρούνται άνισες.

Θεωρούμε το t -test για τον έλεγχο των μέσων τιμών υποθέτοντας ότι οι διασπορές είναι άνισες.

Sig (2-tailed) = 0.105>0.05 οι μέσες τιμές δεν έχουν διαφορά.

Το Διάστημα Εμπιστοσύνης 95% της διαφοράς των μέσων τιμών θεωρώντας άνισες τις διασπορές είναι: (-0,059, 0,611)

Επομένως, εφόσον οι διασπορές είναι μικρότερες από το 0.05 αλλάι το Διάστημα Εμπιστοσύνης 95% περιέχει το 0 μπορούμε να πούμε ότι δεν υπάρχει στατιστικά σημαντική διαφορά μεταξύ των κατηγοριών API και StandAlone Software .

---

2. Πρότυπο Prototype:

Πίνακας 4 : Πίνακες «Group Statistics» και «Independent Samples Test» για το πρότυπο Prototype.

Group Statistics

	ApiStandAlone	N	Mean	Std. Deviation	Std. Error Mean
Prototype	0	114	,40	1,281	,120
	1	177	,46	1,777	,134

Independent Samples Test

	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
Prototype	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	,519	**,472**	-,311	289	**,756**	-,060	,192	**-,438**	**,319**
Equal variances not assumed			-,333	285,316	,739	-,060	,180	-,413	,294

Για το πρότυπο Prototype:

Sig=0,472>0.05 οι διασπορές δεν έχουν διαφορά, θεωρούνται ίσες.

Θεωρούμε το t -test για τον έλεγχο των μέσων τιμών υποθέτοντας ότι οι διασπορές είναι ίσες.

Sig (2-tailed) =0.756>0.05 οι μέσες τιμές δεν έχουν διαφορά.

Το Διάστημα Εμπιστοσύνης 95% της διαφοράς των μέσων τιμών θεωρώντας ίσες τις διασπορές είναι: (-0.438, 0,319).

Επομένως, εφόσον οι διασπορές είναι μεγαλύτερες από το 0.05 καθώς και το Διάστημα Εμπιστοσύνης 95% περιέχει το 0 μπορούμε να πούμε ότι δεν υπάρχει στατιστικά σημαντική διαφορά μεταξύ των κατηγοριών API και StandAlone Software .

---

3. Πρότυπο Singleton:

Πίνακας 5 : Πίνακες «Group Statistics» και «Independent Samples Test» για το πρότυπο Singleton.

Group Statistics

	ApiStandAlone	N	Mean	Std. Deviation	Std. Error Mean
Singleton	0	114	5,90	9,431	,883
	1	177	6,99	13,379	1,006

Independent Samples Test

Singleton		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
	Equal variances assumed	,748	**,388**	-,758	289	**,449**	-1,091	1,440	**-3,925**	**1,743**
	Equal variances not assumed			-,815	286,617	,416	-1,091	1,338	-3,725	1,544

Για το πρότυπο Singleton:

Sig=0,388>0.05 οι διασπορές δεν έχουν διαφορά, θεωρούνται ίσες.

Θεωρούμε το t -test για τον έλεγχο των μέσων τιμών υποθέτοντας ότι οι διασπορές είναι ίσες.

Sig (2-tailed) =0.449>0.05 οι μέσες τιμές δεν έχουν διαφορά.

Το Διάστημα Εμπιστοσύνης 95% της διαφοράς των μέσων τιμών θεωρώντας ίσες τις διασπορές είναι: (-3,925, 1,743)

Επομένως, εφόσον οι διασπορές είναι μεγαλύτερες από το 0.05 καθώς και το Διάστημα Εμπιστοσύνης 95% περιέχει το 0 μπορούμε να πούμε ότι δεν υπάρχει στατιστικά σημαντική διαφορά μεταξύ των κατηγοριών API και StandAlone Software .

4. Πρότυπο AbstractFactory:

Πίνακας 6 : Πίνακες «Group Statistics» και «Independent Samples Test» για το πρότυπο AbstractFactory

Group Statistics

	ApiStandAlone	N	Mean	Std. Deviation	Std. Error Mean
Abstract_Factory	0	114	,18	,989	,093
	1	177	,06	,323	,024

Independent Samples Test

	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
Abstract_ Factory	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	7,325	**,007**	1,413	289	,159	,113	,080	-,045	,271
Equal variances not assumed			1,183	128,635	**,239**	,113	,096	**-,076**	**,303**

Για το πρότυπο AbstractFactory:

Sig=0,007<0.05 οι διασπορές έχουν διαφορά, θεωρούνται άνισες.

Θεωρούμε το t -test για τον έλεγχο των μέσων τιμών υποθέτοντας ότι οι διασπορές είναι άνισες.

Sig (2-tailed) =0.239>0.05 οι μέσες τιμές δεν έχουν διαφορά.

Το Διάστημα Εμπιστοσύνης 95% της διαφοράς των μέσων τιμών θεωρώντας άνισες τις διασπορές είναι: (-0,076, 0,303)

Επομένως, εφόσον οι διασπορές είναι μικρότερες από το 0.05 αλλά το Διάστημα Εμπιστοσύνης 95% περιέχει το 0 μπορούμε να πούμε ότι δεν υπάρχει στατιστικά σημαντική διαφορά μεταξύ των κατηγοριών API και StandAlone Software .

---

5. Πρότυπο AdapterCommand:

Πίνακας 7: Πίνακες «Group Statistics» και «Independent Samples Test» για το πρότυπο AdapterCommand

Group Statistics

	ApiStandAlone	N	Mean	Std. Deviation	Std. Error Mean
Object_Adapter	0	114	9,04	15,514	1,453
	1	177	10,75	16,418	1,234

Independent Samples Test

	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
Object_ Adapter	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	,205	**,651**	-,886	289	**,376**	-1,711	1,930	**-5,509**	**2,088**
Equal variances not assumed			-,897	250,968	,370	-1,711	1,906	-5,465	2,044

Για το πρότυπο AdapterCommand:

Sig=0,651>0.05 οι διασπορές δεν έχουν διαφορά, και θεωρούνται ίσες.

Θεωρούμε το t -test για τον έλεγχο των μέσων τιμών υποθέτοντας ότι οι διασπορές είναι ίσες.

Sig (2-tailed) =0.376>0.05 οι μέσες τιμές δεν έχουν διαφορά.

Το Διάστημα Εμπιστοσύνης 95% της διαφοράς των μέσων τιμών θεωρώντας ίσες τις διασπορές είναι: (-5.509,2,088)

Επομένως, εφόσον οι διασπορές είναι μεγαλύτερες από το 0.05 και το Διάστημα Εμπιστοσύνης 95% περιέχει το 0 μπορούμε να πούμε ότι δεν υπάρχει στατιστικά σημαντική διαφορά μεταξύ των κατηγοριών API και StandAlone Software .

---

6. Πρότυπο Composite:

Πίνακας 8 : Πίνακες «Group Statistics» και «Independent Samples Test» για το πρότυπο Composite

Group Statistics

	ApiStandAlone	N	Mean	Std. Deviation	Std. Error Mean
Composite	0	114	,05	,261	,024
	1	177	,03	,210	,016

Independent Samples Test

Composite	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	1,786	**,182**	,674	289	**,501**	,019	,028	**-,036**	**,073**
Equal variances not assumed			,644	204,591	,520	,019	,029	-,039	,076

Για το πρότυπο Composite:

Sig=0,182>0.05 οι διασπορές δεν έχουν διαφορά, θεωρούνται ίσες.

Θεωρούμε το t -test για τον έλεγχο των μέσων τιμών υποθέτοντας ότι οι διασπορές είναι ίσες.

Sig (2-tailed) =0.501>0.05 οι μέσες τιμές δεν έχουν διαφορά.

Το Διάστημα Εμπιστοσύνης 95% της διαφοράς των μέσων τιμών θεωρώντας ίσες τις διασπορές είναι: (-0,036, 0,073)

Επομένως, εφόσον οι διασπορές είναι μεγαλύτερες από το 0.05 καθώς και το Διάστημα Εμπιστοσύνης 95% περιέχει το 0 μπορούμε να πούμε ότι δεν υπάρχει στατιστικά σημαντική διαφορά μεταξύ των κατηγοριών API και StandAlone Software .

7. Πρότυπο Decorator:

Πίνακας 9 : Πίνακες «Group Statistics» και «Independent Samples Test» για το πρότυπο Decorator

Group Statistics

	ApiStandAlone	N	Mean	Std. Deviation	Std. Error Mean
Decorator	0	114	,32	1,117	,105
	1	177	,16	1,197	,090

Independent Samples Test

Decorator		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
	Equal variances assumed	3,538	**,061**	1,147	289	**,252**	,161	,140	**-,115**	**,436**
	Equal variances not assumed			1,165	253,106	,245	,161	,138	-,111	,432

Για το πρότυπο Composite:

Sig=0,061>0.05 οι διασπορές δεν έχουν διαφορά, θεωρούνται ίσες.

Θεωρούμε το t -test για τον έλεγχο των μέσων τιμών υποθέτοντας ότι οι διασπορές είναι ίσες.

Sig (2-tailed) =0.252>0.05 οι μέσες τιμές δεν έχουν διαφορά.

Το Διάστημα Εμπιστοσύνης 95% της διαφοράς των μέσων τιμών θεωρώντας ίσες τις διασπορές είναι: (-0,115,0,436)

Επομένως, εφόσον οι διασπορές είναι μεγαλύτερες από το 0.05 καθώς και το Διάστημα Εμπιστοσύνης 95% περιέχει το 0 μπορούμε να πούμε ότι δεν υπάρχει στατιστικά σημαντική διαφορά μεταξύ των κατηγοριών API και StandAlone Software .

---

8. Πρότυπο Proxy:

Πίνακας 10 : Πίνακες «Group Statistics» και «Independent Samples Test» για το πρότυπο Proxy

Group Statistics

	ApiStandAlone	N	Mean	Std. Deviation	Std. Error Mean
Proxy	0	114	,41	1,400	,131
	1	177	,48	2,065	,155

Independent Samples Test

Proxy	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	,349	**,555**	-,309	289	**,758**	-,068	,220	**-,501**	**,365**
Equal variances not assumed			-,334	288,200	,738	-,068	,203	-,468	,332

Για το πρότυπο Proxy:

Sig=0.555>0.05 οι διασπορές δεν έχουν διαφορά, θεωρούνται ίσες.

Θεωρούμε το t -test για τον έλεγχο των μέσων τιμών υποθέτοντας ότι οι διασπορές είναι ίσες.

Sig (2-tailed) =0.758>0.05 οι μέσες τιμές δεν έχουν διαφορά.

Το Διάστημα Εμπιστοσύνης 95% της διαφοράς των μέσων τιμών θεωρώντας ίσες τις διασπορές είναι: (-0.501,0.365)

Επομένως, εφόσον οι διασπορές είναι μεγαλύτερες από το 0.05 καθώς και το Διάστημα Εμπιστοσύνης 95% περιέχει το 0 μπορούμε να πούμε ότι δεν υπάρχει στατιστικά σημαντική διαφορά μεταξύ των κατηγοριών API και StandAlone Software .

---

9. Πρότυπο Facade:

Πίνακας 11 : Πίνακες «Group Statistics» και «Independent Samples Test» για το πρότυπο Facade

Group Statistics

	ApiStandAlone	N	Mean	Std. Deviation	Std. Error Mean
Facade	0	114	,96	3,138	,294
	1	177	1,31	4,091	,308

Independent Samples Test

Facade	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	1,626	**,203**	-,788	289	,431	**-,355**	,450	**-1,240**	**,531**
Equal variances not assumed			-,834	280,218	,405	-,355	,425	-1,192	,483

Για το πρότυπο Facade:

Sig=0.203>0.05 οι διασπορές δεν έχουν διαφορά, θεωρούνται ίσες.

Θεωρούμε το t -test για τον έλεγχο των μέσων τιμών υποθέτοντας ότι οι διασπορές είναι ίσες.

Sig (2-tailed) =0.431>0.05 οι μέσες τιμές δεν έχουν διαφορά.

Το Διάστημα Εμπιστοσύνης 95% της διαφοράς των μέσων τιμών θεωρώντας ίσες τις διασπορές είναι: (-1.240,0.531)

Επομένως, εφόσον οι διασπορές είναι μεγαλύτερες από το 0.05 καθώς και το Διάστημα Εμπιστοσύνης 95% περιέχει το 0 μπορούμε να πούμε ότι δεν υπάρχει στατιστικά σημαντική διαφορά μεταξύ των κατηγοριών API και StandAlone Software .

10.Πρότυπο Flyweight:

Πίνακας 12 : Πίνακες «Group Statistics» και «Independent Samples Test» για το πρότυπο Flyweight

Group Statistics

	ApiStandAlone	N	Mean	Std. Deviation	Std. Error Mean
Flyweight	0	114	,25	1,927	,180
	1	177	,02	,183	,014

Independent Samples Test

	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
Flyweight	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	9,833	**,002**	1,591	289	,113	,232	,146	-,055	,519
Equal variances not assumed			1,281	114,317	**,203**	,232	,181	**-,127**	**,590**

Για το πρότυπο Flyweight:

Sig=0.002<0.05 οι διασπορές έχουν διαφορά, θεωρούνται άνισες.

Θεωρούμε το t -test για τον έλεγχο των μέσων τιμών υποθέτοντας ότι οι διασπορές είναι άνισες.

Sig (2-tailed) =0.203>0.05 οι μέσες τιμές δεν έχουν διαφορά.

Το Διάστημα Εμπιστοσύνης 95% της διαφοράς των μέσων τιμών θεωρώντας άνισες τις διασπορές είναι: (-0.127,0.590)

Επομένως, εφόσον οι διασπορές είναι μικρότερες από το 0.05 καθώς και το Διάστημα Εμπιστοσύνης 95% περιέχει το 0 μπορούμε να πούμε ότι δεν υπάρχει στατιστικά σημαντική διαφορά μεταξύ των κατηγοριών API και StandAlone Software .

---

11. Πρότυπο Chain of Responsibility:

Πίνακας 13 : Πίνακες «Group Statistics» και «Independent Samples Test» για το πρότυπο Chain of Responsibility

Group Statistics

	ApiStandAlone	N	Mean	Std. Deviation	Std. Error Mean
Chain_of_Responsibility	0	114	,02	,132	,012
	1	177	,02	,149	,011

Independent Samples Test

	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
Chain_of_ Responsibility	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	,350	**,555**	-,295	289	**,768**	-,005	,017	**-,039**	**,029**
Equal variances not assumed			-,303	261,700	,762	-,005	,017	-,038	,028

Για το πρότυπο Chain of Responsibility:

Sig=0.555>0.05 οι διασπορές δεν έχουν διαφορά, θεωρούνται ίσες.

Θεωρούμε το t -test για τον έλεγχο των μέσων τιμών υποθέτοντας ότι οι διασπορές είναι ίσες.

Sig (2-tailed) =0.768>0.05 οι μέσες τιμές δεν έχουν διαφορά.

Το Διάστημα Εμπιστοσύνης 95% της διαφοράς των μέσων τιμών θεωρώντας ίσες τις διασπορές είναι: (-0.039,0.029)

Επομένως, εφόσον οι διασπορές είναι μεγαλύτερες από το 0.05 καθώς και το Διάστημα Εμπιστοσύνης 95% περιέχει το 0 μπορούμε να πούμε ότι δεν υπάρχει στατιστικά σημαντική διαφορά μεταξύ των κατηγοριών API και StandAlone Software .

12.Πρότυπο Observer:

Πίνακας 14 : Πίνακες «Group Statistics» και «Independent Samples Test» για το πρότυπο Observer

Group Statistics

	ApiStandAlone	N	Mean	Std. Deviation	Std. Error Mean
Observer	0	114	,01	,094	,009
	1	177	,01	,150	,011

Independent Samples Test

Observer		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
	Equal variances assumed	,107	**,744**	-,161	289	**,873**	-,003	,016	**-,034**	**,028**
	Equal variances not assumed			-,177	288,728	,860	-,003	,014	-,031	,026

<u>Για το πρότυπο Observer:</u>

Sig=0.744>0.05 οι διασπορές δεν έχουν διαφορά, θεωρούνται ίσες.

Θεωρούμε το t -test για τον έλεγχο των μέσων τιμών υποθέτοντας ότι οι διασπορές είναι ίσες.

Sig (2-tailed) =0.873>0.05 οι μέσες τιμές δεν έχουν διαφορά.

Το Διάστημα Εμπιστοσύνης 95% της διαφοράς των μέσων τιμών θεωρώντας ίσες τις διασπορές είναι: (-0.034,0.028)

Επομένως, εφόσον οι διασπορές είναι μεγαλύτερες από το 0.05 καθώς και το Διάστημα Εμπιστοσύνης 95% περιέχει το 0 μπορούμε να πούμε ότι δεν υπάρχει στατιστικά σημαντική διαφορά μεταξύ των κατηγοριών API και StandAlone Software .

13. Mediator

Πίνακας 15 : Πίνακες «Group Statistics» και «Independent Samples Test» για το πρότυπο Mediator

Group Statistics

	ApiStandAlone	N	Mean	Std. Deviation	Std. Error Mean
Mediator	0	114	1,54	7,280	,682
	1	177	1,86	7,832	,589

Independent Samples Test

	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
Mediator	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	,252	**,616**	-,350	289	**,726**	-,321	,915	**-2,122**	**1,481**
Equal variances not assumed			-,356	253,735	,722	-,321	,901	-2,094	1,453

Για το πρότυπο Mediator:

Sig=0.616>0.05 οι διασπορές δεν έχουν διαφορά, θεωρούνται ίσες.

Θεωρούμε το t -test για τον έλεγχο των μέσων τιμών υποθέτοντας ότι οι διασπορές είναι ίσες.

Sig (2-tailed) =0.726>0.05 οι μέσες τιμές δεν έχουν διαφορά.

Το Διάστημα Εμπιστοσύνης 95% της διαφοράς των μέσων τιμών θεωρώντας ίσες τις διασπορές είναι: (-2.122,1.481)

Επομένως, εφόσον οι διασπορές είναι μεγαλύτερες από το 0.05 καθώς και το Διάστημα Εμπιστοσύνης 95% περιέχει το 0 μπορούμε να πούμε ότι δεν υπάρχει στατιστικά σημαντική διαφορά μεταξύ των κατηγοριών API και StandAlone Software .

14.Πρότυπο TemplateMethod:

Πίνακας 16 : Πίνακες «Group Statistics» και «Independent Samples Test» για το πρότυπο TemplateMethod

Group Statistics

	ApiStandAlone	N	Mean	Std. Deviation	Std. Error Mean
Template_Method	0	114	2,24	3,622	,339
	1	177	2,47	4,777	,359

Independent Samples Test

	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
Template Method	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	,214	**,644**	-,454	289	**,650**	-,238	,524	**-1,269**	**,793**
Equal variances not assumed			-,481	281,314	,631	-,238	,494	-1,210	,735

Για το πρότυπο TemplateMethod:

Sig=0.644>0.05 οι διασπορές δεν έχουν διαφορά, θεωρούνται ίσες.

Θεωρούμε το t -test για τον έλεγχο των μέσων τιμών υποθέτοντας ότι οι διασπορές είναι ίσες.

Sig (2-tailed) =0.650>0.05 οι μέσες τιμές δεν έχουν διαφορά.

Το Διάστημα Εμπιστοσύνης 95% της διαφοράς των μέσων τιμών θεωρώντας ίσες τις διασπορές είναι: (-1.269,0.793)

Επομένως, εφόσον οι διασπορές είναι μεγαλύτερες από το 0.05 καθώς και το Διάστημα Εμπιστοσύνης 95% περιέχει το 0 μπορούμε να πούμε ότι δεν υπάρχει στατιστικά σημαντική διαφορά μεταξύ των κατηγοριών API και StandAlone Software .

---

15. Πρότυπο Strategy:

Πίνακας 17 : Πίνακες «Group Statistics» και «Independent Samples Test» για το πρότυπο Strategy

Group Statistics

	ApiStandAlone	N	Mean	Std. Deviation	Std. Error Mean
Strategy	0	114	,86	4,087	,383
	1	177	,53	1,638	,123

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Strategy	Equal variances assumed	3,107	**,079**	,974	289	**,331**	,334	,343	**-,341**	**1,010**
	Equal variances not assumed			,831	136,644	,407	,334	,402	-,461	1,129

Για το πρότυπο Strategy:

Sig=0.079>0.05 οι διασπορές δεν έχουν διαφορά, και θεωρούνται ίσες.

Θεωρούμε το t -test για τον έλεγχο των μέσων τιμών υποθέτοντας ότι οι διασπορές είναι ίσες.

Sig (2-tailed) =0.331>0.05 οι μέσες τιμές δεν έχουν διαφορά.

Το Διάστημα Εμπιστοσύνης 95% της διαφοράς των μέσων τιμών θεωρώντας ίσες τις διασπορές είναι: (-0.341,1.010)

Επομένως, εφόσον οι διασπορές είναι μεγαλύτερες από το 0.05 αλλά το Διάστημα Εμπιστοσύνης 95% περιέχει το 0 μπορούμε να πούμε ότι δεν υπάρχει στατιστικά σημαντική διαφορά μεταξύ των κατηγοριών API και StandAlone Software .

16.Πρότυπο Visitor:

Πίνακας 18 : Πίνακας «Group Statistics» και «Independent Samples Test» για το πρότυπο Visitor

Group Statistics

	ApiStandAlone	N	Mean	Std. Deviation	Std. Error Mean
Visitor	0	114	,05	,261	,024
	1	177	,01	,075	,006

Independent Samples Test

	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
Visitor	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	21,356	**,000**	2,258	289	,025	,047	,021	,006	,088
Equal variances not assumed			1,874	125,187	**,063**	,047	,025	**-,003**	**,097**

Για το πρότυπο Visitor:

Sig=0.000<0.05 οι διασπορές έχουν διαφορά, θεωρούνται άνισες.

Θεωρούμε το t -test για τον έλεγχο των μέσων τιμών υποθέτοντας ότι οι διασπορές είναι άνισες.

Sig (2-tailed) =0.063>0.05 οι μέσες τιμές δεν έχουν διαφορά.

Το Διάστημα Εμπιστοσύνης 95% της διαφοράς των μέσων τιμών θεωρώντας άνισες τις διασπορές είναι: (-0.003,0.097)

Επομένως, εφόσον οι διασπορές είναι μικρότερες από το 0.05 αλλά το Διάστημα Εμπιστοσύνης 95% περιέχει το 0 μπορούμε να πούμε ότι δεν υπάρχει στατιστικά σημαντική διαφορά μεταξύ των κατηγοριών API και StandAlone Software.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Σύμφωνα με τα παραπάνω αποτελέσματα, οι απαντήσεις που μπορούν να δοθούν στα ερωτήματα της ερευνάς μας είναι:

1. Στο ερώτημα «*RQ1: π*οια πρότυπα σχεδίασης χρησιμοποιούνται συχνότερα στις βιβλιοθήκες (API)» η απάντηση βρίσκεται στον συγκεντρωτικό πίνακα των μέσων όρων των προτύπων. Τόσο λοιπόν στα API, όσο και στα Standalone Software τα πρότυπα που χρησιμοποιούνται περισσότερο είναι το πρότυπο του «Προσαρμογέα (Adapter)» με μέσο όρο 9,04 και ακολουθεί το «Μοναδιαίο (Singleton)» πρότυπο με μέσο όρο 5,90. Την τελευταία θέση, σε αυτό το ερώτημα για την κατηγορία API καταλαμβάνει το πρότυπο «Παρατηρητής (Observer)» με μέσο όρο 0,01, ενώ για την Standalone κατηγορία, το πρότυπο «Επισκέπτης (Visitor)» , με μέσο όρο 0,01.

2. Στο ερώτημα «*RQ2:* Υπάρχει στατιστικά σημαντική διαφορά μεταξύ των κατηγοριών λογισμικού API και Standalone;, την απάντηση βρήκαμε στους πίνακες «Independent Samples Test» που παρουσιάσαμε λίγο ποιο πάνω. Σύμφωνα με αυτούς τους πίνακες λοιπόν, βγάλαμε το συμπέρασμα ότι και στα 16 διαφορετικά πρότυπα που περιέχονται στα λογισμικά που χρησιμοποιήσαμε στην έρευνα μας, δεν υπάρχει στατιστικά σημαντική διαφορά μεταξύ των κατηγοριών API και StandAlone Software.

ΚΕΦΑΛΑΙΟ 6 - *ΣΥΜΠΕΡΑΣΜΑΤΑ*

Η εργασία αυτή στοχεύει στη μελέτη της χρήσης των προτύπων σχεδίασης στις βιβλιοθήκες . Τα πρότυπα σχεδίασης προσεγγίστηκαν από δυο πλευρές. Αρχικά μέσω μιας συστηματικής ανασκόπησης της βιβλιογραφίας, προσπαθήσαμε να προσεγγίσουμε την ύπαρξη μελετών που αναφέρονται στην χρήση των προτύπων στα API. Στη συνέχεια και εφόσον δεν βρήκαμε στην βιβλιογραφική ανασκόπηση κάποιο άρθρο που να μας λύνει τα ερωτήματα μας, μέσω μιας εμπειρικής μελέτης αξιολογήσαμε το βαθμό χρήσης προτύπων σχεδίασης στις ανοιχτές Βιβλιοθήκες.

Σχετικά με τα αποτελέσματα της μελέτης μας για τη χρήση αντικειμενοστραφών προτύπων σχεδίασης σε 291 έργα ανοιχτού λογισμικού, βγαίνει το συμπέρασμα ότι το συχνότερο σε χρήση πρότυπο σχεδίασης, τόσο σε λογισμικά API, όσο και σε λογισμικά Standalone είναι το πρότυπο Adapter. Σε αντίθετη περίπτωση, το πρότυπο που χρησιμοποιείτε λιγότερο στην κατηγορία API είναι το πρότυπο Observer , ενώ στην κατηγορία Standalone είναι το πρότυπο Visitor. Επίσης, σύμφωνα με τα αποτελέσματα μας, συμπεράναμε ότι στα 16 πρότυπα που χρησιμοποιήθηκαν στην έρευνα μας, δεν υπάρχει στατιστικά σημαντική διαφορά μεταξύ των κατηγοριών API και StandAlone Software.

ΑΝΑΦΟΡΕΣ

[1] A. J. Ko, B. A. Myers, and H. Aung, “Six Learning Barriers in End-User Programming Systems”, IEEE Symposium on Visual Languages and Human-Centric Computing, Rome, Italy, Sep 26-29, 2004, 199-206.

[2] Basili V.R., Selby R.W., Hutchens D.H. (1986), “Experimentation in Software Engineering”, *In IEEE Transactions on Software Engineering,* IEEE Computer Society, Vol.13, No 07, pp. 733-743.

[3] Brereton P., Kitchenham B., Budgen D., Turner M., Khalil M. (2007), “Lessons from applying the systematic literature review process within the software engineering domain”, *Journal of Systems and Software*, Elsevier, Vol. 80, No. 4, pp 571-583

[4] Brian Ellis, Jeffrey Stylos, and Brad Myers(2007),” The Factory Pattern in API Design: A Usability Evaluation”, 29th International Conference on Software Engineering (ICSE'07), Carnegie Mellon University.

[5] Chatzigeorgiou A. (2005), “Object-Oriented Design: UML, Principles, Patterns and Heuristics”, *Kleidarithmos*, Athens, 1st edition.

[6]“Cocoa Fundamentals Guide: Class Clusters”, *http://developer.apple.com/documentation/Cocoa/Conceptual/CocoaFundamentals/CocoaObjects/chapter_3_section_9.html*

[7] Colin J. Neill (2003), “Leveraging object-orientation for real-time imaging systems”, Real-Time Imaging 9 (423–432) ,The Pennsylvania State University, Malvern, USA.

[8] Dyba T., Dingsoyr T. (2008), “Empirical studies of agile software development: A systematic review”, *Information and Software Technology*, Elsevier, Vol 50, No 9-10, pp. 833-859.

[9] Gamma E, Helms R, Johnson R, Vlissides J. (1995), “Design Patterns: Elements of Reusable Object-Oriented Software”, *Addison-Wesley Professional*, Reading, MA, 1st edition.

[10] G. Florijn, M. Meijers, and P. van Winsen, “Tool Support for Object-Oriented Patterns”, Proceedings, ECOOP ‘97, Springer-Verlag, 1997, pp. 472-495.

[11] “Java 2 Platform Standard Edition 5.0 API Specification”, *http://java.sun.com/j2se/1.5.0/docs/api/index.html*

[12] J. Stylos, S. Clarke, and B. A. Myers, “Comparing API Design Choices with Usability Studies: A Case Study and Future Directions”, PPIG 2006.

[13] J. Stylos, S. Clarke, “Usability Implications of Requiring Parameters in Objects’ Constructors”, to appear in ICSE‘07.

[14] Kitchenham B. (2007), “Procedures for undertaking systematic literature reviews”, *Joint Technical Report*, Computer Science Department, Keele University.

[15] Kitchenham B., Brereton O. P., Budgen D., Turner M., Bailey J., Linkman S. (2009), “Systematic literature reviews in software engineering – A systematic literature review”, *Information and Software Technology*, Elsevier, Vol 51, No 1, pp 7-15.

[16] O. Astrachan, G. Mitchener, G. Berry, and L. Cox, “Design patterns: an essential component of CS curricula, SIGCSE Bull. 30, 1, Mar. 1998, pp. 153-160.

[17] Race P, Rice D, Vera R. Java Image I/O API Guide, April 2001, available at http://java.sun.com/j2se/1.4/docs/guide/imageio/.

[18] S. Clarke, “Measuring API usability”, Dr. Dobb’s Journal Windows/.NET Supplement, May 2004, pp. S6-S9.

[19] S. Clarke, “API Usability and the Cognitive Dimensions Framework”, 2003,*http://blogs.msdn.com/stevencl/archive/2003/10/08/57040.aspx*,

[20] T.R.G. Green and M. Petre, “Usability Analysis of Visual Programming Environments: A ‘Cognitive Dimensions’ Framework.” *Journal of Visual Languages and Computing*, 1996. 7(2): pp. 131-174.

[21] “The Eclipse Project”, *http://www.eclipse.org*

[22] Wohlin C., Runeson P., Host M., Ohlsson M.C., Regnell B., Wesslen A. (2000), “Experimentation in Software Engineering”, *Kluwer Academic Publishers*, Boston/ Dordrecht/ London, 1st edition.

[23] Tsantalis N., Chatzigeorgiou V, Stephanides G., Halkidis S. T. (2006). "Design Pattern Detection using Similarity Scoring", *In IEEE Transaction on Software Engineering,* IEEE Computer Society, Vol. 32, No 11, pp. 896-909.

[24] “.NET Framework Class Library, *http://msdn2.microsoft.com/en-us/library/ms229335.aspx*

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Αρβανίτου Ελβίρα-Μαρία (2011), Πτυχιακή εργασία με θέμα: «Εμπειρική μελέτη της επίδρασης των προτύπων σχεδίασης στα σφάλματα λογισμικού παιχνιδιών»

2. Γκορτζής Αντώνιος (2012),Πτυχιακή εργασία με θέμα : «Στρατηγική επιλογής κλάσεων με στόχο την βελτιστοποίηση της ποιότητας του επιλεγμένου κώδικα».

3. Καρούνος θ. (2010), “Ανοιχτό Λογισμικό: ευκαιρία για την ανάπτυξη της τοπικής αγοράς υπηρεσιών;”, Άρθρο στο ΒΗΜΑ ΙΔΕΩΝ.

4. Κοντοπρία Φωτεινή (2012), Πτυχιακή εργασία με θέμα: «Ανάπτυξη εφαρμογών κοινωνικής δικτύωσης σε κινητές συσκευές» .

5. Χαραλαμπίδου Σοφία (2010), Πτυχιακή εργασία με θέμα: «Εμπειρική μελέτη για τη χρήση προτύπων σχεδίασης σε παιχνίδια ανοιχτού λογισμικού»,

6. Gamma E, Helms R, Johnson R, Vlissides J. (1995), “Design Patterns: Elements of Reusable Object-Oriented Software”, Addison-Wesley Professional, Reading, MA, 1st edition.

7. Chatzigeorgiou A. (2005), “Object-Oriented Design: UML, Principles, Patterns and Heuristics”, Kleidarithmos, Athens, 1st edition.

ΠΑΡΑΡΤΗΜΑΤΑ Α

Λογισμικά που χρησιμοποιήθηκαν κατά την μελέτη:

	Category (API/Standalone)	Software
	API	Mosquito_1.0.72
	API	Beautyj_1.1
	API	DependencyFinder_1.2.1_beta4
	API	Odal_1.1.1
	API	jaffa_2.1.0
	API	Jiapi_0.3.1
	API	Kieker_1.5
	API	Hsqldb_2.2.8
	API	Mxquery_0.60
	API	Smallsql_0.21
	API	Teiid_8.1
	API	oracle_jutils_20110411
	API	Stendhal-starter_1.0.0
	API	toolbox_b.31
	API	Jvx_1.1_beta1
	API	Entityfs_1.2
	API	Makumba_0.9.5.1
	API	Ifw2_1.33
	API	Mosquito_1.0.73
	API	Beautyj_1.2
	API	DependencyFinder_1.2.1_beta5
	API	Odal_1.1.2
	API	jaffa_2.1.1
	API	Jiapi_0.3.2
	API	Kieker_1.6
	API	Hsqldb_2.2.9
	API	Mxquery_0.61
	API	Smallsql_0.22
	API	Teiid_8.2
	API	oracle_jutils_20110412

	Category (API/Standalone)	Software
	API	toolbox_b.32
	API	Jvx_1.1_beta2
	API	Entityfs_1.3
	API	Makumba_0.9.5.2
	API	Ifw2_1.34
	API	Mogwai-erdesignerng-SHAPSHOT_3.0.0_M6
	API	Orbada-1.0.9.205
	API	Rete-db_1.2
	API	Jsqsh_1.4
	API	Jackhare-core_0.1.11
	API	Dbfree-1.0M7
	API	Exzellenz_1.5
	API	Rdbform-09.73.00
	API	Jdbc-bench_1.0
	API	Jasperreportjsf_1.0_beta4
	API	Zeus-jscl_1.70
	API	Itextpdf_5.3.1
	API	Junit_4.10
	API	Jwords_0.03
	API	Simbio_1.0
	API	domainMath_IDE_0.06
	API	InteractiveCalculator_v1
	API	Jbookshelf_20120523
	API	orthoInspector_v.140
	API	jasperreportjs_jsflogin_1.0_beta4
	API	Spooler_1.16
	API	Jpdfunit_1.2
	API	extended_java_worknet_Library_1.6.4
	API	Jcalculator_1.0.0
	API	GSVideo
	API	jAudio
	API	JavaHMO_TiVo_HMO_Server
	API	Java_Implementation_of_Speex
	API	JMF_wrapper_for_ffmpeg
	API	JMyOggRadioPlayer
	API	LEA_Lightweight_Eyetracking_Algorithm

	Category (API/Standalone)	Software
	API	StreamRipStar
	API	TabSearch
	API	A_Java_library_for_readingwriting_Excel
	API	Barbecue_Java_bar_code_generator
	API	Barcode4J
	API	Card_Me_VCard_Java_Library
	API	iCal4j
	API	MPXJ_Microsoft_Project_Exchange
	API	olap4j
	API	Java_Matrix_Library
	API	Jlue
	Standalone	CheckStyle_5.5
	Standalone	Launchj4-3_3.0.2
	Standalone	Proguard_4.8
	Standalone	Dedexer_1.22
	Standalone	Fmpp_0.9.1.4
	Standalone	jibx_1.2.4
	Standalone	retrotranslator_1.2.9
	Standalone	sablecc-4-_beta_4
	Standalone	css2xslfo_1.62
	Standalone	dbmaintain_2.4
	Standalone	olap4j_1.0.1
	Standalone	xbasej_20120119
	Standalone	Bristlecone_0.6
	Standalone	Adamstore_1.1
	Standalone	Persistence_1.05
	Standalone	ConnFarm_0.93
	Standalone	AppServer4RPG_2012_05_24
	Standalone	Hyperic-Sigar_1.6.4
	Standalone	encache_2.5.2
	Standalone	Jvx_1.1
	Standalone	heuroph_2.6
	Standalone	Simple-xml_2.6
	Standalone	ohla_0.5
	Standalone	vdrassistant_0.2.318
	Standalone	Jenes_2.0.0
	Standalone	Ztemplates_2.40

	Category (API/Standalone)	Software
	Standalone	Aranea-mvc_2.0
	Standalone	nextFramework_3.5.3
	Standalone	extcos_0.3b
	Standalone	Cartago_2.0.1
	Standalone	Whiteness_1.4.0
	Standalone	ujoframework_1.22
	Standalone	esigate_4.0_beta1
	Standalone	J4fry_1.2
	Standalone	dans_dbflib-beta-09
	Standalone	expressionj_0.9.2
	Standalone	Sormula_2.1.1
	Standalone	ServiceCloud_0.1
	Standalone	Ebus_2.1.0
	Standalone	QwicGUI_1.1_beta
	Standalone	Jave_1.0.2
	Standalone	Atan_0.4.3
	Standalone	Jailer_4.0.11
	Standalone	Jsqsh_1.4
	Standalone	easysql_1.1
	Standalone	gestdb_2.4.001
	Standalone	Csvtosqljdk5_3.10
	Standalone	qform_1.4
	Standalone	toscanaj_1.6
	Standalone	dbconsole_2.10
	Standalone	jsqltool_1.1
	Standalone	FreeQueryBuilder_1.0
	Standalone	Doolin_1.0
	Standalone	vsyntaxtextarea_2.0.3
	Standalone	Tico_bin_other_e10
	Standalone	Zekr:Multimedia_Quran_Study_Software_1.1.0
	Standalone	Dynamic_Reports_3.0.0
	Standalone	Jpod_intarsys_PDF_Renderer_5.5.1.20120711
	Standalone	openpcl_v0.0.9
	Standalone	The_reprap_project_20110509
	Standalone	aisgedcom_0.81
	Standalone	Histoire_Mondiale_1.0
	Standalone	Schoodule_1.21
	Standalone	Simured_v1

	Category (API/Standalone)	Software
	Standalone	Easypastry_1.0
	Standalone	S-Math_20110422
	Standalone	Perseus-java-Hopper_20110527
	Standalone	Heritrix-Internet-Archive-web-Crawler-engine_3.10
	Standalone	java-Genetic-Algorith-Library_1.0.0
	Standalone	Jcart2d_3.2.2
	Standalone	ProjectX-DVB-Demux-tool_0.91.0
	Standalone	Chord_4.0
	Standalone	Dguitar_a_Guitar_Pro_viewer_player
	Standalone	Eisenkraut
	Standalone	FireflyClient_a_Java_FireflyClient
	Standalone	FScape
	Standalone	Impro_Visor
	Standalone	IpodCopy
	Standalone	Jajuk
	Standalone	JavaMod_The_Java_Mod_Player
	Standalone	Java_MPEG-1_Video_Decoder_and_Player
	Standalone	Java_Multiple_Audio_Format_Converter
	Standalone	Java_VoiceXML_Editor
	Standalone	MeDs_Movie_Manager
	Standalone	MIDI_Rules
	Standalone	MIDPlayer
	Standalone	MJPEG_Lossless_Rotate
	Standalone	myPod
	Standalone	Napsack
	Standalone	orDrumbox_Java_Software_Drum_Machine
	Standalone	Project_X_DVB_demux_Tool
	Standalone	RTSP_Proxy
	Standalone	StLab
	Standalone	Tuning_Fork
	Standalone	VdrAssistant
	Standalone	Visage
	Standalone	XleTView
	Standalone	Xtreme_Media_Player
	Standalone	YaMeG_Yet_another_Mencoder_Gui
	Standalone	Amnesia

	Category (API/Standalone)	Software
	Standalone	DavMail_POPIMAPSMTPCaldav_to_Exchange
	Standalone	EDIReader
	Standalone	EXcompCEL
	Standalone	FreeMercator_Java_POS_Point_of_Sale
	Standalone	IBController
	Standalone	Java_Library_Management_System
	Standalone	JMoney
	Standalone	JSignPdf
	Standalone	JStock_Free_Stock_Market_Software
	Standalone	Memoranda
	Standalone	More_Wiki_in_a_jar
	Standalone	Open_Java_Trading_System
	Standalone	OpenReports
	Standalone	OpenSearchServer
	Standalone	OYSTER_Entity_Resolution
	Standalone	PDF_Forms_Designer
	Standalone	FIX_Pusher
	Standalone	PlayBilling
	Standalone	Rapla_resource_scheduling
	Standalone	Rivulet_Enterprise_Search
	Standalone	StatSVN
	Standalone	synPOS
	Standalone	Teamcenter_Engineering_Admin_View
	Standalone	Universal_Password_Manager
	Standalone	VCS_to_ICS_Calendar_Converter
	API	Cairo_0.2
	API	Coffeehouse_1.0
	API	DSHub_RC1
	API	EteriaIRCClient_1.0.3
	API	Fax4j_0.32
	API	GridChat_20.11.2006
	API	HotSAX_0.1.2c
	API	Jes-gui_1.0.0
	API	Jeti_0.7.7
	API	JRAS32_2.2
	API	ModbusforJava_1.1
	API	OpenAS2_20100816
	API	Open-dis_4.02

	Category (API/Standalone)	Software
	API	OpenFAST_1.1.1
	API	WebTranslatorJavaAPI_0.2a
	API	WirelessInternetrelaychat(IRC)client_2.0.1
	API	Lizzy_1.1.1
	API	BiRMI_0.0.1
	API	CelerFTP_0.1alpha
	API	JEmval_0.2
	API	Arbaro_1.9.8
	API	Barbecue-Javabarcodegenerator_1.5-beta1
	API	Ditaa_0.9
	API	Djatoka_1.1
	API	Form4j_0.5
	API	GeOxygene_1.4
	API	Hexagraph_0.1
	API	Im4java_1.2.3
	API	Mysfstats_1.0.4
	API	PolyTreeTable_0.1
	API	SASlib_first
	API	ProteinShader_0.9.4
	API	ZoomableVisualTransformationMachine_0.11.0
	API	VietOCR_3.4
	Standalone	ConneX_1.0.0
	Standalone	DKIMforJavaMail_1.3
	Standalone	Galena_2.0.0
	Standalone	JASEN_0.9
	Standalone	JavaHyperTerminal_1.1
	Standalone	Jaxen_1.0
	Standalone	JID-JavaImageDownloader_1.8
	Standalone	Mantaray_2.0.1
	Standalone	netsim_pre1
	Standalone	NetworkService_1.0.0.2
	Standalone	SimpleForumAutoPoster_SDK
	Standalone	SIPProxy-VoIPSecurityTestTool_2.1
	Standalone	Smile3D_0.1.1
	Standalone	Sonar_0.59
	Standalone	V.R.C.C_1.2
	Standalone	ZanzibarOpenIVR_0.1

	Category (API/Standalone)	Software
	Standalone	SafePeer_2.5.1
	Standalone	Softray_0.1.90
	Standalone	SASPresensi
	Standalone	Stringer_1.0b1
	Standalone	FOray_0.3
	Standalone	Genalyze_0.3.2
	Standalone	JEngine_2.01
	Standalone	Barcode4J_2.1.0
	Standalone	Delineate_0.5
	Standalone	DoubleType_0.2.3.61
	Standalone	FreeMind_0.9.0
	Standalone	GameExtractor_2.01
	Standalone	Geotag_0.082
	Standalone	jAMOS_jamos
	Standalone	JARP_1.1.13
	Standalone	Java-Comix_4.2
	Standalone	JavaTreeview_4.2
	Standalone	JKiwi_0.9.5
	Standalone	Joti_0.3.2
	Standalone	LaTeXDraw_2.0.8
	Standalone	NefIO_0.9.4
	Standalone	OpcionFontViewer_1.1.1
	Standalone	OpenStego_0.5.2
	Standalone	Pixelle_1.0.0
	Standalone	PiXflow-slideshowcomposer_0.9.9
	Standalone	QRay_1a
	Standalone	ScriptedImaging_1.0.1
	Standalone	Semicuro_1.0.1
	Standalone	SunflowRenderingSystem_0.07.2
	Standalone	SVGrafiX_0.9a
	Standalone	TheRepRapProject_20110509
	Standalone	Vizant_0.1.2
	Standalone	FigTeX_0.1
	Standalone	HiveBoard_0.6.0
	Standalone	XMLInteractiveSlideShow_1.7.5
	Standalone	Pixelitor_1.1.2