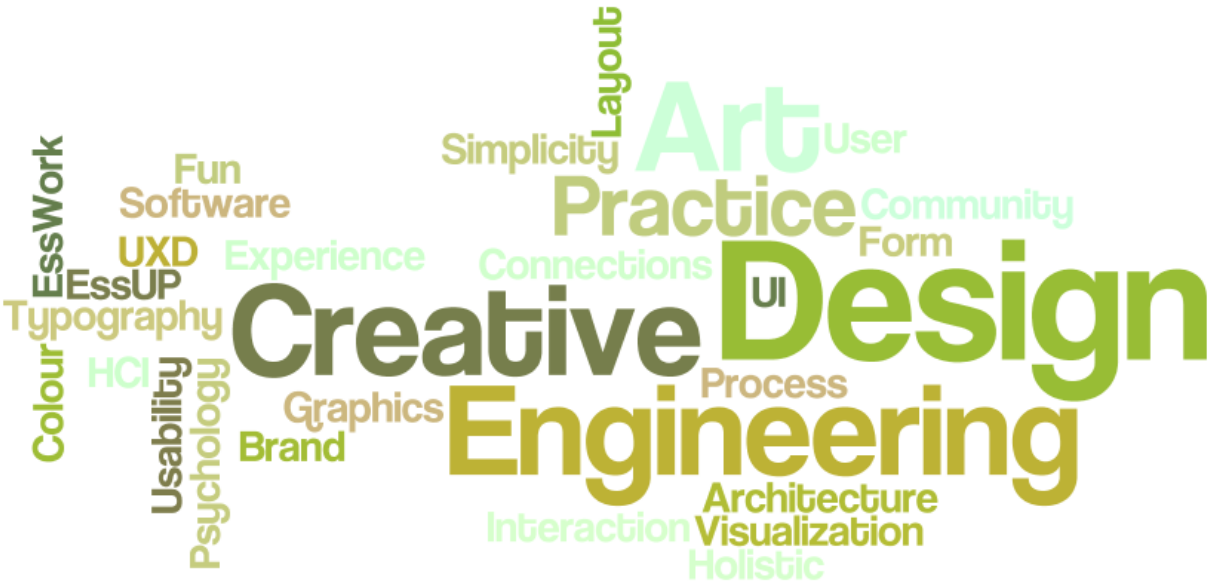
ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Τίτλος Πτυχιακής Εργασίας:

Μελέτη καταγραφής χρήσης και επίδρασης των προτύπων σχεδίασης στις βιβλιοθήκες



Των φοιτητών

Αλεξανδρίδη Ανδρέα 04/2556

Αδαμίδου Τριάδα 04/2537

Επιβλέπων Καθηγητής

Δεληγιάννης Ιγνάτιος

Θεσσαλονίκη 2012

Πρόλογος

Το λογισμικό βιβλιοθηκών (API/Libraries) αποτελεί κλάδο κρίσιμης σημασίας για την Τεχνολογία Λογισμικού, καθώς σχεδόν όλα τα έργα λογισμικού βασίζονται στη χρήση των βιβλιοθηκών. Ταυτόχρονα, η σχεδίαση και ανάπτυξη λογισμικού βιβλιοθηκών παρουσιάζει ιδιαίτερες δυσκολίες λόγω της αναγκαιότητας για συνεχή εξέλιξή τους χωρίς να επηρεάζονται τα προγράμματα-πελάτες των βιβλιοθηκών.

Στην βελτίωση της ποιότητας του λογισμικού, αποσκοπεί η χρήση των προτύπων σχεδίασης. 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.

Στόχοι της πτυχιακής αυτής είναι:

1. Μελέτη καταγραφής χρήσης και επίδρασης των προτύπων σχεδίασης στις βιβλιοθήκες
2. Πειραματικά αποτελέσματα Χρήσης Προτύπων Σχεδίασης

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 στην ποιότητα του λογισμικού, ζήτημα επίκαιρο στην κοινωνία ανάπτυξης λογισμικού.

Περίληψη

Οι όροι «πρότυπο σχεδίασης» και «ποιότητα λογισμικού» έχουν πολύ μεγάλη σημασία για τη μηχανική λογισμικού, και εξακολουθούν να αποτελούν ένα σημαντικό πεδίο έρευνας μέχρι σήμερα. Οι δυο όροι συνδέονται άμεσα, αφού τα πρότυπα σχεδίασης, εφαρμόζονται στο λογισμικό, με σκοπό την βελτίωση της ποιότητάς του. Εργαλεία, για την μέτρηση των ποιοτικών χαρακτηριστικών του συστήματος και συνεπώς για την αξιολόγηση της επίδρασης των προτύπων στην ποιότητά, αποτελούν οι μετρικές.

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 έναν ικανοποιητικό αριθμό άρθρων, 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.

Στη συνέχεια, παρουσιάζουμε μια εμπειρική μελέτη, σχετικά με τη χρήση προτύπων σχεδίασης στις βιβλιοθήκες λογισμικού. Για τη μελέτη αυτή συλλέξαμε 168 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.

Abstract

The terms "design template" and "software quality" are very important to software engineering, and remain an important area of research today. The two terms are directly linked, since the design standards shall apply to the software in order to improve its quality. Tools for measuring the quality characteristics of the system and to evaluate the impact of the quality standards are the metrics. Initially we carried out a systematic review of the literature in order to determine and document the research activity to date in the field of object-oriented design patterns. For purposes of review, we collected and studied a sufficient number of articles that were published in top journals, conferences and workshops (workshops). Analyzing the data collected from the literature, using statistical techniques, interesting results emerged. Then, we present an empirical study on the use of design patterns in software libraries. For this study we collected 168 popular software (API and Standalone) open source, a known source repository (sourceforge), which fulfilled certain requirements. Using a detection tool design templates from object files Java mining information on the use of standards in software design them. The data collected were analyzed using statistical techniques and obtained interesting results.

Ευχαριστίες

Σε αυτό το σημείο της εργασίας μας θα θέλαμε να ευχαριστήσουμε όλους όσους συμμετείχαν στην ολοκλήρωσή της.

Ένα μεγάλο ευχαριστώ λοιπόν στον επιβλέποντα καθηγητή μας τον Κ. Δεληγιάννη Ιγνάτιο και στον εργαστηριακό συνεργάτη της σχολής μας τον Κ. Αμπατζόγλου Απόστολο που σε όλον αυτόν τον χρόνο ήταν κοντά μας ,να μας στηρίξουν με τη γνώση και την καθοδήγησή τους.

Κλείνοντας θα θέλαμε να ευχαριστήσουμε θερμά τον συμφοιτητή και φίλο Γκορτζή Αντώνη για τη συνεχή βοήθειά του σε ένα κρίσιμο στάδιο της εργασίας μας, καθώς και τον απόφοιτο της σχολής μας Παρασκευόπουλο Κώστα που συνέβαλε σημαντικά στο να ολοκληρώσουμε σε συντομο χρονικό διάστημα την εργασία μας.

Περιεχόμενα

Ευρετήριο Σχημάτων

Ευρετήριο Πινάκων

1. Εισαγωγή

1.1 Ανοιχτό Λογισμικό

Εισαγωγή

Οι όροι "**Ελεύθερο Λογισμικό**" ("**Free Software**") και "**Λογισμικό Ανοικτού Κώδικα**" ("**Open Source Software**") αναφέρονται σε προγράμματα των οποίων ο πηγαίος κώδικας είναι προσβάσιμος σε άτομα εκτός της εταιρείας παραγωγής τους και συνεργατών της. Οι όροι αυτοί δεν αναφέρονται σε λογισμικό που διατίθεται δωρεάν (freeware) καθώς το ελεύθερο λογισμικό/λογισμικό ανοικτού κώδικα μπορεί να έχει (μεγάλη) τιμή πώλησης, ενώ αντίθετα υπάρχουν πολλά πακέτα "δωρεάν" λογισμικού των οποίων ο πηγαίος κώδικας είναι μη προσβάσιμος σε άτομα εκτός της εταιρείας παραγωγής. Στον αντίποδα βρίσκεται το "κλειστό" (closed source) ή "ιδιοταγές" (proprietary) λογισμικό, του οποίο ο πηγαίος κώδικας παραμένει κρυφός σε τρίτα άτομα (συμπεριλαμβανομένων των χρηστών του λογισμικού). Το τελευταίο μοντέλο ακολουθούν οι περισσότερες μεγάλες εταιρείες λογισμικού, όπως η Microsoft, η Adobe, η EA Games, η Oracle κλπ.

Το ελεύθερο λογισμικό αντιμετωπιζόταν ως και πριν δέκα χρόνια ως μια ιδεαλιστική προσπάθεια κάποιων "ρομαντικών" που δεν απέβλεπαν σε οικονομικά ωφέλη και προτιμούσαν την αναγνώριση στην κοινότητα των hacker από μια καλοπληρωμένη θέση σε μια από τις μεγάλες εταιρείες λογισμικού. Σιγά σιγά όμως, και σε αυτό συντέλεσε κυρίως η συνειδητοποίηση του ότι το λογισμικό δεν αποτελεί προϊόν αλλά υπηρεσία, το ανοικτό λογισμικό άρχισε να θεωρείται ένα βιώσιμο μοντέλο οικονομικής ανάπτυξης, το οποίο μπορεί να αποφέρει πολύ μεγαλύτερα κέρδη, πιο αξιόπιστες υπηρεσίες και περισσότερες ευκαιρείες απ'ότι το κλειστό λογισμικό. Μετά το 2000, παρατηρείται μια έκρηξη ενδιαφέροντος από μεγάλες εταιρείες, καθώς όλο και περισσότερες "ανοίγουν" πλέον τον κώδικα των προγραμμάτων τους (IBM, Sun, Apple) αλλά και δηλώνουν ανοικτή προτίμηση σε προγράμματα ανοικτού κώδικα (Pixar, Dell). Η ανάλυση του μοντέλου ανάπτυξης βασιζομένου στο ανοικτό λογισμικό γίνεται σε άλλο κείμενο.

Τελευταία, η ιδέα του ανοικτού λογισμικού έχει επεκταθεί και σε άλλους τομείς υπηρεσιών. Έτσι, πλέον, ακούμε για "ανοικτό" hardware (που στην ουσία έγκειται σε δημοσιοποίηση του κώδικα VHDL ή Verilog που παράγει το εν λόγω κύκλωμα), "ανοικτό" φορμά αρχείων, "ελεύθερη" βιβλιογραφία/τεκμηρίωση κλπ.

Ανοικτό ή ελεύθερο;

Αν και οι περισσότεροι άνθρωποι διατυπώνουν στους όρους "Ελεύθερο" και "Ανοικτό" λογισμικό αναφερόμενοι στο ίδιο πράγμα, υπάρχει μια μικρή ιδεολογική διαφορά ανάμεσα σε αυτά τα δύο. Σύμφωνα με το Free Software Foundation, οι ελευθερίες που δίνει μια άδεια χρήσης λογισμικού είναι οι εξής:

- Η ελευθερία να τρέξεις το πρόγραμμα, για οποιονδήποτε σκοπό (freedom 0)
- Η ελευθερία να διαβάσεις ή να τροποποιήσεις τον πηγαίο κώδικα του προγράμματος (και κατά συνέπεια και το ίδιο το πρόγραμμα) για ιδιωτική χρήση (freedom 1)
- Η ελευθερία του να αντιγράψεις το αρχικό πρόγραμμα και να το δώσεις σε κάποιον τρίτο (freedom 2)
- Η ελευθερία του να μπορείς να δημοσιοποιείς τροποποιημένες και βελτιωμένες εκδόσεις του προγράμματος σε τρίτα άτομα (freedom 3)

Οι περισσότερες EULA (End-User Licence Agreement) των ιδιοταγών προγραμμάτων δίδουν μόνο την ελευθερία (0) και απαγορεύουν ρητά ως ποινικό αδίκημα κατά πνευματικής ιδιοκτησίας τις υπόλοιπες. Θεωρητικά, οποιοδήποτε πρόγραμμα δίδει και την ελευθερία (1) θεωρείται ότι εμπίπτει στην κατηγορία του ανοικτού λογισμικού (ή λογισμικού ανοικτού κώδικα, open source software), άσχετα με το εάν επιτρέπει τις ελευθερίες (2) και (3). Τα προγράμματα τα οποία δίνουν και τις τέσσερις ελευθερίες χρήσης ανήκουν στο ελεύθερο λογισμικό (free software). Στην πράξη τώρα, η συντριπτική πλειονότητα των προγραμμάτων ανοικτού κώδικα είναι και ελεύθερα, δηλαδή επιτρέπουν (υπό κάποιους όρους) στον χρήστη να τροποποιήσει τον πηγαίο κώδικα του προγράμματος και να τον δώσει σε τρίτα άτομα. Ελάχιστα είναι τα προγράμματα που παρέχουν μεν τον πηγαίο τους κώδικα, απαγορεύουν δε τη δημοσίευσή του (αυτούσιου ή

τροποποιημένου) σε τρίτους. Για τον λόγο αυτό, οι όροι "ελεύθερο" και "ανοικτό" λογισμικό έχουν γίνει πλέον σχεδόν συνώνυμοι.

Για να χαρακτηριστεί ένα λογισμικό ως ανοιχτού κώδικα θα πρέπει να πληρεί τα παρακάτω δέκα κριτήρια:

1. **Ελεύθερη Αναδιανομή**: Η άδεια δεν πρέπει να περιορίζει κάποιον από το να πουλήσει ή να δώσει το λογισμικό ως μέρος ενός άλλου λογισμικού.

2. **Πηγαίος Κώδικας**: Το πρόγραμμα πρέπει να περιέχει τον πηγαίο κώδικα.

3. **Παραγόμενο Λογισμικό**: Πρέπει να επιτρέπονται αλλαγές ή παράγωγα προϊόντα.

4. **Ακεραιότητα Πηγαίου Κώδικα του συγγραφέα**: Η άδεια μπορεί να περιορίζει τον πηγαίο κώδικα από την αναδιανομή σε άλλη τροποποιημένη έκδοση.

5. **Καμία Διάκριση Εναντίων Προσώπων ή Ομάδων**: Η άδεια χρήσης δεν πρέπει να βλάπτει κανένα άτομο ή ομάδα ατόμων.

6. **Καμία Διάκριση ως προς τα Πεδία της Χρήσης:** Πρέπει να επιτρέπονται όλες οι πιθανές χρήσεις του προγράμματος από την άδεια χρήσης του.

7. **Διανομή της Άδειας**: Τα δικαιώματα που απορρέουν από το πρόγραμμα θα πρέπει να έχουν εφαρμογή και σε όποιον χρησιμοποιεί το πρόγραμμα.

8. **Η Άδεια δεν πρέπει να Είναι Συγκεκριμένη για ένα Λογισμικό**: Τα δικαιώματα που απορρέουν από το πρόγραμμα δεν πρέπει να εξαρτώνται από το αν το πρόγραμμα είναι μέρος ενός πακέτου λογισμικού.

9. **Η Άδεια δεν Πρέπει να Περιορίζει Άλλο Λογισμικό**: Η άδεια δεν πρέπει να βάζει κανένα περιορισμό όσο αφορά άλλο λογισμικό που διανέμεται μαζί με αυτό.

10. Η **Άδεια θα Πρέπει να είναι Ουδέτερης Τεχνολογίας**: Κανένας όρος της άδειας χρήσης δεν πρέπει να εξαρτάται από μία συγκεκριμένη τεχνολογία.

Άδειες Ελεύθερου Λογισμικού

Σε ποιον ανήκει το ελεύθερο λογισμικό; Μπορώ να το αντιγράψω; Μπορώ να αλλάξω μια γραμμή στον κώδικα και το πουλήσω ως κλειστό λογισμικό; Αν και στις περισσότερες περιπτώσεις, ιδιοκτήτης του copyright παραμένει ο αρχικός συγγραφέας του λογισμικού, οι απαντήσεις στις παραπάνω ερωτήσεις εξαρτώνται από την άδεια (licence) με την οποία έρχεται το εκάστοτε πρόγραμμα, και την οποία είναι υποχρεωμένος να δεχθεί όποιος σκοπεύει να το χρησιμοποιήσει με οποιονδήποτε έμμεσο ή άμεσο τρόπο. Οι κυριότερες άδειες ανοικτού λογισμικού είναι οι εξής:

-Apache Licence

Δημιουργήθηκε από το Apache Foundation και είναι η άδεια υπό την οποία διανέμεται ο εξυπηρετητής HTTP Apache. Είναι μια πολύ αναλυτικά διατυπωμένη άδεια, που ενώ επιτρέπει την αναδιανομή και τροποποίηση του λογισμικού, απαιτεί αυτή να γίνεται υπό την ίδια άδεια, να δείχνονται αναλυτικά ποια αρχεία του πηγαίου κώδικα πείραξε ο χρήστης και απαγορεύει τη χρήση υλικού που σχετίζεται με πατέντες λογισμικού καθώς και τη χρήση ονομάτων και συμβόλων του αρχικού συγγραφέα για διαφημιστικούς σκοπούς. Τέλος, απαλάσσει το δημιουργό από κάθε ευθύνη σχετική με τη χρήση του προγράμματος. Εκτός από τον Apache, την άδεια χρησιμοποιούν πολλά προγράμματα που σχετίζονται με αυτόν, όπως ο Tomcat.

Apache Licences: http://www.apache.org/licenses/

-Artistic Licence

Την συνέταξε ο συγγραφέας της perl, Larry Wall. Η πρώτη της έκδοση ήταν τόσο περίπλοκα διατυπωμένη που πολλοί την κατηγόρησαν ότι δεν είναι άδεια ελεύθερου λογισμικού. Η γλώσσα προγραμματισμού perl αλλά και πολλά modules της ήταν συνδεδεμένα με αυτήν. Η δεύτερη έκδοση είναι πιο σαφώς διατυπωμένη, και δίδει το δικαίωμα ανάγνωσης, τροποποίσης και αναδιανομής του πηγαίου κώδικα/προγράμματος (υπό οποιαδήποτε άδεια), εφ'όσον διατηρείται η αναφορά στον αρχικό συγγραφέα του προγράμματος, σε

περίπτωση που δεν υπάρξει τροποποίηση. Επιπλέον, απαλλάσει το συγγραφέα από κάθε ευθύνη σχετική με τη χρήση του προγράμματος. Η postgreSQL διατίθεται υπό αυτήν την άδεια.

Αρχική Artistic Licence: http://www.perl.com/language/misc/Artistic.html

Artistic∞ Licence, 2nd Edition: http://dev.perl.org/rfc/346.html

-BSD Licence

Η άδεια αυτή αφορά λογισμικό που αναπτύχθηκε αρχικά στο πανεπιστήμιο Berkeley στην Καλιφόρνια των ΗΠΑ. Είναι μια από τις πιο 'ελεύθερες' άδειες, εφ'όσον επιτρέπει την ανάγνωση, την τροποποίηση και την αναδημοσίευση του προγράμματος υπό οποιαδήποτε άδεια, με ή χωρίς τον πηγαίο κώδικα, σε εμπορικά ή μη εμπορικά πακέτα. Επιπλέον, υπάρχουν τροποποιήσεις της άδειας, που αφορούν όμως μόνο το θέμα της χρήσης του ονόματος του αρχικού συγγραφέα για διαφημιστικούς σκοπούς. Παράδειγμα προγραμμάτων που τη χρησιμοποιούν είναι όλα τα είδη λειτουργικού BSD (freeBSD, netBSD, openBSD) καθώς και οι αρχικές εκδόσεις των προγραμμάτων ηλεκτρονικού σχεδιασμού Spice, Magic και IrSim. Ως αποτέλεσμα της ελευθερίας που δίδει η άδεια για χρήση κώδικα ελεύθερων προγραμμάτων σε ιδιοταγή προγράμματα, πολλά κλειστά λειτουργικά συστήματα έχουν κομμάτια βασισμένα στο BSD (π.χ. μέρος του network API στα Windows 2000∞) και πολλά ιδιοταγή πακέτα ηλεκτρονικού σχεδιασμού βασίζονται στο Spice. Αυτό εκλαμβάνεται σαν ελευθερία από τους οπαδούς της άδειας, αλλά πολλές φορές δρα ανασταλτικά προς την ανάπτυξη του ελεύθερου λογισμικού (βλ. http://www.gnu.org/philosophy/x.html).

Η BSD Licence: http://www.opensource.org/licenses/bsd-license.php

-GNU General Public Licence (GPL)

Γράφτηκε αρχικά από τον Richard Stallman για το GNU project. Επιτρέπει την ανάγνωση, τροποποίηση και αναδιανομή του λογισμικού, μαζί με τον πηγαίο κώδικα του, με τον όρο ότι τροποιημένες ή μη εκδόσεις του θα αναδιανείμονται υπό την ίδια άδεια. Εν ολίγοις, διασφαλίζει ότι οι χρήστες του τροποποιημένου λογισμικού θα απολαμβάνουν τις ίδιες ελευθερίες με το χρήστη του αρχικού λογισμικού. Συνεπώς απαγορεύει την χρήση (ολόκληρου ή τμήματος) του πηγαίου κώδικα του προγράμματος σε κλειστά πακέτα λογισμικού. Αυτό εξασφαλίζεται δίνοντας το copyright του προγράμματος στον αρχικό δημιουργό, οπότε σε περίπτωση που κάποιος δε σεβαστεί την GPL μπορεί να μυνηθεί για καταπάτηση πνευματικών δικαιωμάτων. Η GNU GPL έχει κατηγορηθεί από πολλούς ως "ιός" (επειδή θα πρέπει κάθε πρόγραμμα που έχει σχέση με το αρχικό να τη φέρει) και ότι στερεί τη δημιουργία άμεσου κέρδους στον προγραμματιστή (επειδή μπορεί οποιοσδήποτε να αναδιανείμει το λογισμικό). Παρά τις κριτικές αποτελεί τη σημαντικότερη και πιο διαδεδομένη άδεια ελεύθερου λογισμικού. Σημαντικά προγράμματα που τη χρησιμοποιούν είναι ο πυρήνας του Linux, ο μεταγλωττιστής gcc, ο επεξεργαστής κειμένου Emacs, ο διερμηνέας της Perl, ο Mozilla Firefox, η MySQL, το Cygwin, το σύστημα αρχείων ReiserFS και πλέον και η βιβλιοθήκη Qt. Μια πιο ελαστική έκδοση της GPL είναι η GNU Lesser General Public Licence (LGPL). Μια LGPL βιβλιοθήκη για παράδειγμα, μπορεί να συνδεθεί με ένα πρόγραμμα που χρησιμοποιεί άλλη άδεια, ακόμα και αν αυτό δεν είναι ελεύθερο λογισμικό. Το 2005 άρχισαν συζητήσεις για την τρίτη έκδοση της άδειας, η οποία δίνει ιδιαίτερη έμφαση σε θέματα πατεντών λογισμικού και DRM. Η GNU GPLv3 εκδόθηκε στις 27 Ιουνίου 2007.

GNU GPL, v3.0: http://www.gnu.org/licenses/gpl.html

GNU LGPL, v3.0: http://www.gnu.org/copyleft/lesser.html

-MIT Licence

Η άδεια αυτή επιτρέπει την τροποποίηση και την αναδιανομή του προγράμματος με οποιονδήποτε τρόπο, υπό οποιαδήποτε άδεια, για οποιονδήποτε σκοπό. Το πιο γνωστό πρόγραμμα που τη χρησιμοιποιεί είναι ο X Window System (X11) που χρησιμοιείται για το παραθυρικό περιβάλλον στις περισσότερες διανομές Linux, και γι'αυτό η άδεια αυτή ονομάζεται πολλές φορές και X Licence ή X11 Licence. Άλλα προγράμματα που τη χρησιμοποιούν είναι το Expat, το MetaKit, και το PuTTY.

MIT Licence Template: http://www.opensource.org/licenses/mit-license.php

-Open Software Licence

Ουσιαστικά δίνει τις ίδιες ελευθερίες και τους ίδιους περιορισμούς με την GNU GPL (δλδ. απαιτεί την αναδιανομή υπό την ίδια άδεια) με σημαντική διαφορά τον όρο που αφορά τις πατέντες λογισμικού. Ο όρος αυτός τερματίζει αυτόματα την άδεια και στερεί τον χρήστη από τις ελευθερίες της στην περίπτωση που ο χρήστης μυνήσει οποιοδήποτε λογισμικό που τη χρησιμοποιεί για καταπάτηση πατεντών λογισμικού. Αυτό γίνεται κυρίως για αντιμετωπιστεί το θέμα των πατεντών λογισμικού, που πολλοί πιστεύουν ότι έχουν γίνει επιζήμιες για το ελεύθερο λογισμικό.

Open Software Licence: http://www.opensource.org/licenses/osl.php

1.2 Πρότυπα Σχεδίασης

Τα πρότυπα σχεδίασης πρεσβεύουν τον πιο ορθό τρόπο ανάπτυξης του λειτουργικού σώματος σε ένα σύστημα, των οποίων η επίδραση πολλές φορές είναι ευεργετική. Η μορφή τους παρουσιάζεται με την χρήση διαγραμμάτων της Ενοποιημένης Γλώσσας Μοντελοποίησης, UML. Η UML αποτελεί τη κυρίαρχη γλώσσα μοντελοποίησης πληροφοριακών συστημάτων της αντικειμενοστρεφούς τεχνολογίας και τυγχάνει ευρείας αποδοχής στην βιομηχανία κατασκευής 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 κατασκευαστικά πρότυπα, δομικά πρότυπα και συμπεριφορικά πρότυπα .

•*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 κάποιααντικείμενα από τις κλάσεις που ορίζουν τα αντικείμενα αυτά και τοντρόπο 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

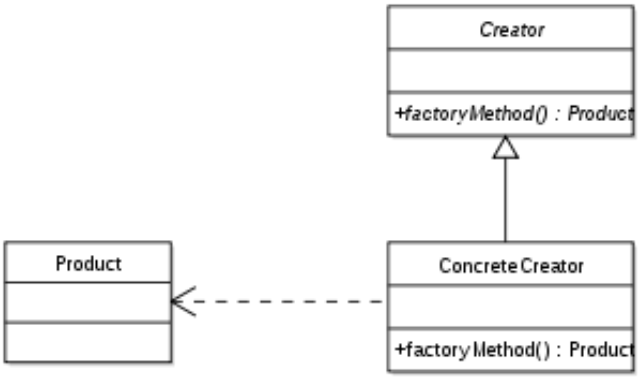
συζεύξεις χαμηλές, τις κλάσεις αμετάβλητες, και προσφέρει εναλλακτικές λύσεις στην κληρονομικότητα. Τέτοια πρότυπα είναι το Composite, το Adapter και το Decorator Pattern

•Συμπεριφορικά Πρότυπα(Behavioral 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.

1.2.1 Factory Method

Το πρότυπο αυτό συγκεντρώνει σε μία κατάλληλη κλάση, το Factory, όλη τη λειτουργικότητα κατασκευής στιγμιοτύπων μίας σειράς *παραγόμενων* κλάσεων που κληρονομούν κάποια κοινή *υπερκλάση* ή υλοποιούν την ίδια *διασύνδεση*. Οι μέθοδοι του Factory, όταν καλούνται, κατασκευάζουν ένα νέο αντικείμενο κατάλληλου τύπου και επιστρέφουν έναν αφηρημένο δείκτη προς αυτό, δηλαδή έναν δείκτη τύπος του οποίου είναι η γενική διασύνδεση, οπότε το εξωτερικό πρόγραμμα μπορεί να τις καλεί για να λαμβάνει το ζητούμενο κατά περίπτωση αντικείμενο χωρίς το ίδιο να χρειάζεται να γνωρίζει κάθε παραγόμενο τύπο δεδομένων που υλοποιεί τη διασύνδεση. Με αυτόν τον τρόπο το πρόγραμμα είναι κλειστό ως προς πιθανές επεκτάσεις και μόνο το Factory είναι ανοιχτό ως προς αυτές, καθώς μόνον ο δικός του κώδικας χρειάζεται να τροποποιηθεί σε περίπτωση π.χ. προσθήκης μίας νέας παραγόμενης κλάσης. Το Factory συνήθως παρέχει μία μέθοδο για κάθε δυνατό παραγόμενο τύπο, αλλά μία παραλλαγή ονόματι παραμετροποιημένο Factory περιέχει μία μοναδική μέθοδο η οποία επιλέγει το αντικείμενο που θα δημιουργήσει και θα επιστρέψει, αναλόγως με την τιμή ενός ορίσματος που δέχεται. Το όρισμα αυτό μπορεί π.χ. να διαβάζεται

από κάποιο αρχείο ρυθμίσεων ή να μεταβιβάζεται ως όρισμα γραμμής εντολών στο εξωτερικό πρόγραμμα (στον *πελάτη*), έτσι ώστε το τελευταίο να είναι κλειστό ως προς το σύνολο των παραγόμενων κλάσεων και να μη χρειάζεται επαναμεταγλώττιση σε περίπτωση που τροποποιηθεί το σύνολο αυτό.

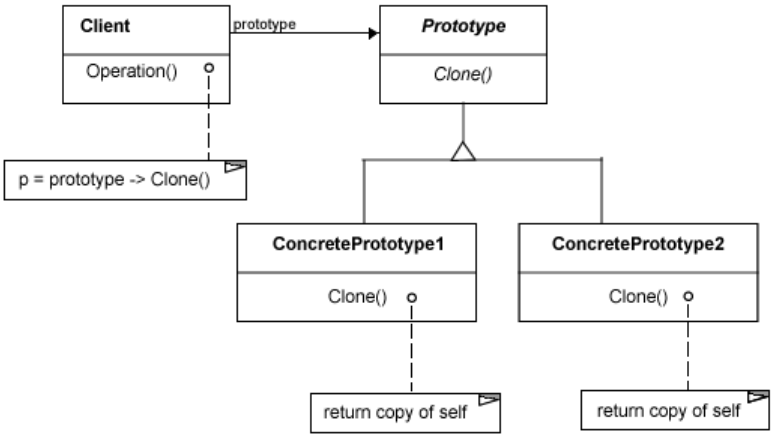


Σχήμα 1. Διάγραμμα κλάσεων προτύπου Factory

1.2.2 Prototype

Το Prototype είναι ένα σχεδιαστικό πρότυπο με το οποίο ένα νέο αντικείμενο κατασκευάζεται με "κλωνοποίηση" κάποιου υπάρχοντος. Η κλωνοποίηση αυτή γίνεται μέσω μίας μεθόδου clone η οποία παρέχεται από μία αφηρημένη κλάση ή διασύνδεση A και υλοποιείται σε κάθε παραγόμενη κλάση B η οποία κληρονομεί την A. Έτσι η κλήση της clone σε ένα στιγμιότυπο της B επιστρέφει ένα αντίγραφο του εν λόγω στιγμιότυπου, το οποίο αναλόγως με την υλοποίηση μπορεί να είναι είτε *ρηχό*, δηλαδή να περιέχει δείκτες προς τις εσωτερικές δομές δεδομένων του αρχικού στιγμιότυπου, είτε *βαθύ*, δηλαδή να περιέχει πλήρη, νεοδημιουργηθέντα αντίγραφα αυτών των δομών δεδομένων. Το Prototype χρειάζεται σε περιπτώσεις όπου πρέπει να κατασκευαστεί μία ρέπλικα ενός αντικειμένου αλλά με κάποιον άλλον τρόπο, π.χ. στην Java με χρήση του τελεστή

new και ενός κατασκευαστή αντιγράφου (copy constructor), προσβάλλεται η αρχή ανοιχτότητας-κλειστότητας. Η μέθοδος clone μπορεί να επικαλύπτει την κλήση του εκάστοτε κατασκευαστή αντιγράφου με ένα κοινό επίπεδο αφαίρεσης έτσι ώστε να μη χρειάζεται το εξωτερικό πρόγραμμα να γνωρίζει όλους τους παραγόμενους τύπους δεδομένων που υλοποιούν τη διασύνδεση Α, καθώς η clone επιστρέφει αναφορά του αφηρημένου τύπου Α.

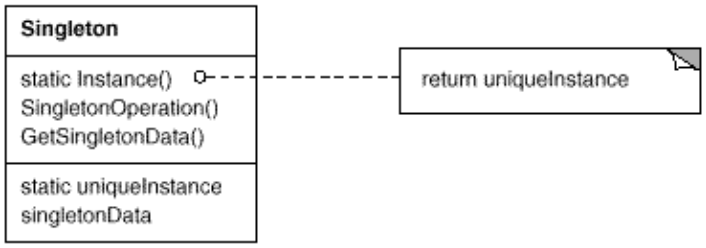


Σχήμα 2. Διάγραμμα κλάσεων προτύπου Prototype

1.2.3 Singleton

Το Singleton είναι ένα σχεδιαστικό πρότυπο που επιλύει το ζήτημα της εξασφάλισης της ύπαρξης το πολύ ενός στιγμιότυπου κάποιας κλάσης Α κατά το χρόνο εκτέλεσης, ένας περιορισμός που μπορεί να ανακύψει για διάφορους λόγους. Με το πρότυπο Singleton η αρμοδιότητα για την ικανοποίηση αυτού του περιορισμού ανατίθεται στην ίδια την κλάση Α και δε μεταβιβάζεται στο εξωτερικό πρόγραμμα που τη χρησιμοποιεί. Αυτό γίνεται μέσω μίας στατικής δημόσιας μεθόδου της Α η οποία δημιουργεί το πρώτο στιγμιότυπο της κλάσης και ακολούθως, σε κάθε επόμενη κλήση της, επιστρέφει έναν δείκτη ή μία αναφορά προς αυτό. Το Singleton διαφέρει από μία απλή καθολική μεταβλητή (global variable), η οποία επίσης δημιουργείται αναγκαστικά μόνο μία φορά καθ'

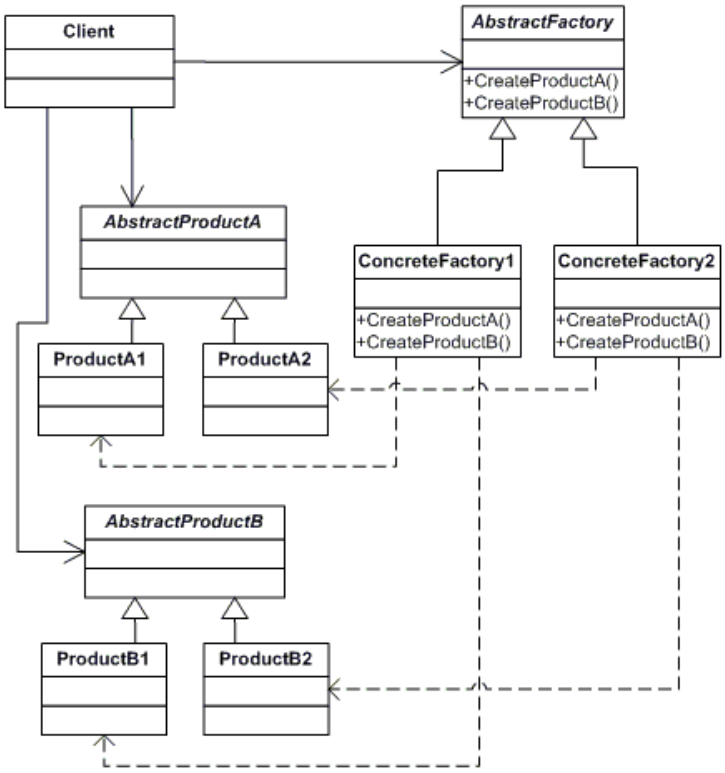
όλο το χρόνο εκτέλεσης του προγράμματος, καθώς το αντικείμενο Singleton δε δεσμεύει μνήμη μέχρι τη στιγμή της πρώτης κλήσης της μεθόδου κατασκευής. Από εκεί κι έπειτα όμως το αντικείμενο παραμένει στη μνήμη ως τον τερματισμό του προγράμματος γιατί είναι υλοποιημένο ως στατική μεταβλητή.



Σχήμα 5. Διάγραμμα κλάσεων προτύπου Singleton

1.2.4 Abstract Factory

Το Abstract Factory είναι ένα πρότυπο που αποτελεί παραλλαγή του Factory και χρησιμοποιείται όταν έχουμε παράλληλες κληρονομικές ιεραρχίες κλάσεων οι οποίες μοιράζονται κάποια κοινή ιδιότητα. Έστω π.χ. ότι γράφουμε μία βιβλιοθήκη για κατασκευή γραφικών διασυνδέσεων χρήστη (GUI) η οποία περιέχει διάφορα οπτικά στοιχεία (widgets) τα οποία κληρονομούν από μία κοινή διασύνδεση ονόματι *Widget* (π.χ. κουμπιά, μπάρες κύλισης κλπ). Υποθέτοντας ότι η βιβλιοθήκη υλοποιείται σε δύο εκδοχές, μία για λειτουργικό σύστημα Windows και μία για Unix, μπορούμε να δηλώσουμε ένα αφηρημένο Factory το οποίο παρέχει μεθόδους για την κατασκευή κάθε πιθανού οπτικού στοιχείου ώστε αυτές οι μέθοδοι να υποσκελίζονται διαφορετικά στις δύο παραγόμενες κλάσεις που υλοποιούν το αφηρημένο Factory: μία για Windows και μία για Unix.

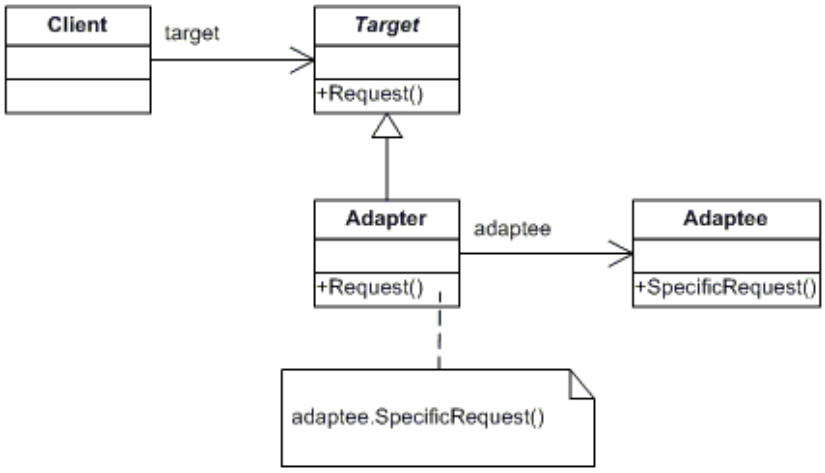


Σχήμα 4. Διάγραμμα κλάσεων προτύπου Abstract Factory

1.2.5 Adapter

Το πρότυπο αυτό επιτρέπει την προσαρμογή της διασύνδεσης που εξάγει μία κλάση Α σε μία πρότυπη διασύνδεση, έστω Interface, που αναμένεται από ένα εξωτερικό πρόγραμμα ώστε να μην παραβιάζεται η αρχή ανοιχτότητας-κλειστότητας. Αυτό γίνεται μέσω μίας ενδιάμεσης κλάσης, του Adapter, η οποία υλοποιεί το Interface και ταυτοχρόνως είτε κληρονομεί την Α (υλοποίηση με κληρονομικότητα), είτε περιέχει μία ιδιωτική αναφορά σε αντικείμενο τύπου Α (υλοποίηση με συνάθροιση). Σε κάθε περίπτωση ο Adapter έχει πρόσβαση στις μεθόδους της Α και οι δικές του μέθοδοι, οι υπογραφές των οποίων συμφωνούν με αυτές που αναμένονται από κάποιον πελάτη καθώς προδιαγράφονται από τη διασύνδεση / αφηρημένη κλάση Interface, τις καλούν εσωτερικά και επιστρέφουν τα αποτελέσματα, αποκρύπτοντας παράλληλα τη διαδικασία αυτή

από το εκάστοτε πρόγραμμα πελάτη. Το πρόβλημα που επιλύει το εν λόγω σχεδιαστικό πρότυπο εν γένει προκύπτει όταν μία υπάρχουσα ιεραρχία κλάσεων, οι οποίες υλοποιούν ένα συγκεκριμένο στοιχείο αφαίρεσης, πρέπει να επεκταθεί με την προσθήκη μίας νέας κλάσης η διασύνδεση της οποίας δεν είναι συμβατή με το υπάρχον στοιχείο αφαίρεσης. Η υλοποίηση του προτύπου Adapter μέσω κληρονομικότητας επιτρέπει την προσαρμογή στη ζητούμενη διασύνδεση και των προστατευμένων μεθόδων της κλάσης Α, αντί μόνο των δημοσίων της, ενώ είναι και ελαφρώς πιο αποδοτική από την υλοποίηση με συνάθροιση, αφού κάθε κλήση στον Adapter «μεταφράζεται» σε κλήση σε μια άλλη μέθοδο του ιδίου αντί για κλήση σε μέθοδο άλλου αντικειμένου. Από την άλλη η υλοποίηση με συνάθροιση δεν αντιμετωπίζει προβλήματα ακόμα και αν η Α δεν μπορεί να κληρονομηθεί, ενώ επιτρέπει την προσαρμογή όχι μόνο της Α αλλά και κάθε υποκλάσης της λόγω του πολυμορφισμού.

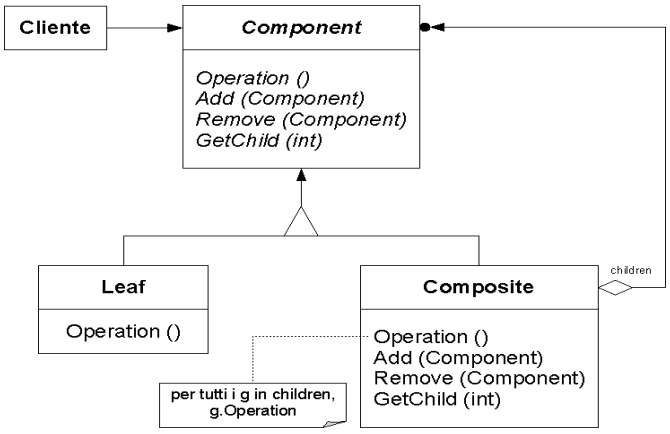


Σχήμα 5. Διάγραμμα κλάσεων προτύπου Adapter

1.2.6 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).

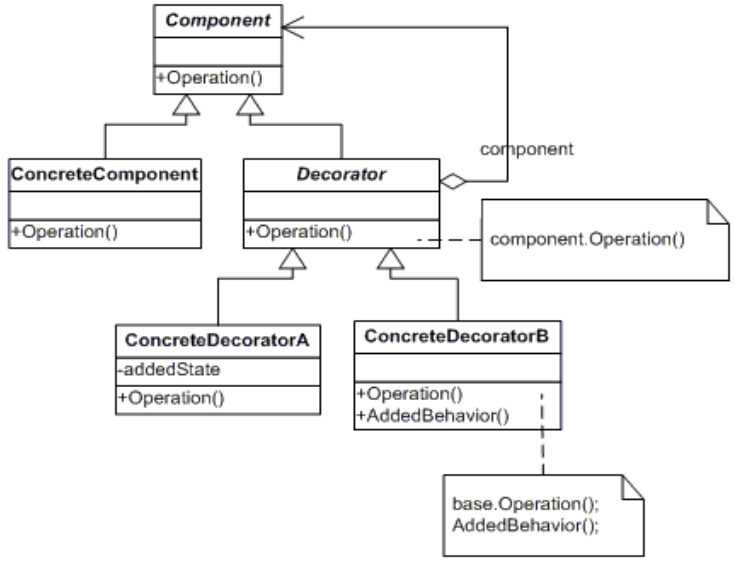


Σχήμα 6. Διάγραμμα κλάσεων προτύπου Composite

1.2.7 Decorator

Το πρότυπο αυτό επιτρέπει την εύκολη και δυναμική επέκταση της λειτουργικότητας κάποιων υπαρχόντων κλάσεων A, B κλπ, οι οποίες υλοποιούν την ίδια διασύνδεση ή κληρονομούν την ίδια αφηρημένη κλάση (έστω Interface), σε χρόνο εκτέλεσης. Αυτό γίνεται μέσω του Decorator, μίας νέας κλάσης η οποία επίσης υλοποιεί την Interface αλλά περιέχει ως ιδιωτικό πεδίο και μία αναφορά σε ένα στιγμιότυπο του γενικού τύπου Interface (έστω το instance), η οποία τυπικά μεταβιβάζεται ως όρισμα στον κατασκευαστή της Decorator. Έτσι οι μέθοδοι της τελευταίας υλοποιούν εσωτερικά την καινούργια λειτουργικότητα αλλά για τις κοινές εργασίες καλούν τις αντίστοιχες μεθόδους του instance. Κατά τον χρόνο εκτέλεσης θα μπορούσε το αντικείμενο Decorator να κατασκευάζεται με όρισμα οποιοδήποτε στιγμιότυπο τύπου Interface (ακόμα και του ίδιου του Decorator, αν και αυτό δε θα είχε ιδιαίτερο νόημα) ώστε κατά περίπτωση το

αντικείμενο να παρέχει τη λειτουργικότητα οποιασδήποτε κλάσης τύπου Interface, είτε της A είτε κάποιας άλλης, επεκτεταμένης με ένα συγκεκριμένο σύνολο δυνατοτήτων. Με αυτόν τον τρόπο γίνεται εφικτός ένας δυναμικός συνδυασμός λειτουργιών από στοιχειώδεις δομικούς λίθους κατά τον χρόνο εκτέλεσης.

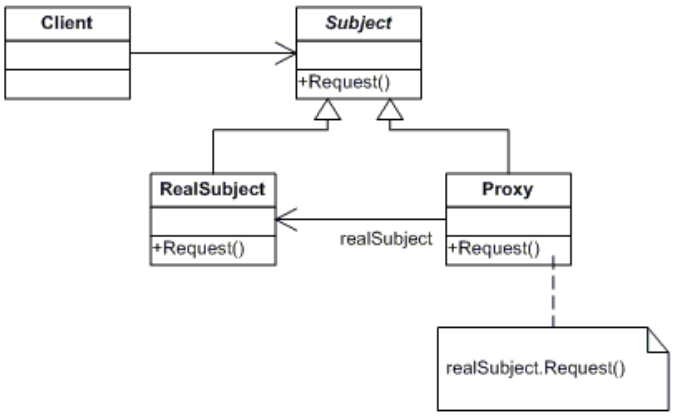


Σχήμα 7. Διάγραμμα κλάσεων προτύπου Decorator

1.2.8 Proxy

Όταν η πρόσβαση ενός εξωτερικού προγράμματος σε αντικείμενα κάποιας συγκεκριμένης κλάσης (έστω της *A*) πρέπει να είναι ελεγχόμενη και να πληροί ορισμένες προϋποθέσεις (π.χ., για λόγους εξοικονόμησης μνήμης, να μη φορτώνεται πραγματικά η *A* μέχρι να κληθεί μία μέθοδος της), το πρότυπο Proxy παρέχει έναν τυποποιημένο τρόπο ώστε αυτό να γίνεται διατηρώντας την κλειστότητα του εξωτερικού προγράμματος ως προς την εν λόγω κλάση και τον τρόπο λειτουργίας της· ακόμα και αν η υλοποίηση της αλλάξει ο πελάτης είναι αδιάφορος απέναντι σε αυτές τις αλλαγές (δε χρειάζεται τροποποίηση) χάρη σε ένα επίπεδο αφαίρεσης που του παρέχεται από μία ενδιάμεση κλάση, τον Proxy,

η οποία παίζει το ρόλο του μεσολαβητή πρόσβασης. Ο Proxy είναι που εκτελεί όλους τους απαραίτητους ελέγχους και προσπελαύνει πραγματικά τα αντικείμενα, ενώ ταυτόχρονα υλοποιεί την ίδια διασύνδεση με την *A* κι έτσι ο πελάτης, ο οποίος κατέχει μία αναφορά προς στιγμιότυπο του Proxy αντί της *A*, δεν αντιλαμβάνεται καν τη μεσολάβησή του· ο Proxy ουσιαστικώς εικονικοποιεί την πρόσβαση στη ζητούμενη κλάση. Συνήθως κατασκευάζεται, αντί για την *A*, από ένα Factory και εντελώς διαφανώς για το εξωτερικό πρόγραμμα, ενώ δεν είναι σπάνιο να περιέχει μία αναφορά στο πραγματικό στιγμιότυπο της *A* ως ιδιωτικό πεδίο. Το σχεδιαστικό αυτό πρότυπο αποτελεί ειδική εφαρμογή της γενικότερης έννοιας του proxy στην πληροφορική. Με αυτήν τη γενικότερη έννοια ο proxy μπορεί να μεσολαβεί μεταξύ ενός πελάτη και ενός οποιουδήποτε ακριβού, σπάνιου ή περίπλοκου πόρου. Μία ιδιαίτερη περίπτωση εφαρμογής της έννοιας του proxy αποτελεί ο απομακρυσμένος proxy, όπως τα στελέχη (stubs) πελάτη και διακομιστή στο υπόδειγμα απομακρυσμένης κλήσης διαδικασιών στα συστήματα ενδιάμεσου λογισμικού.

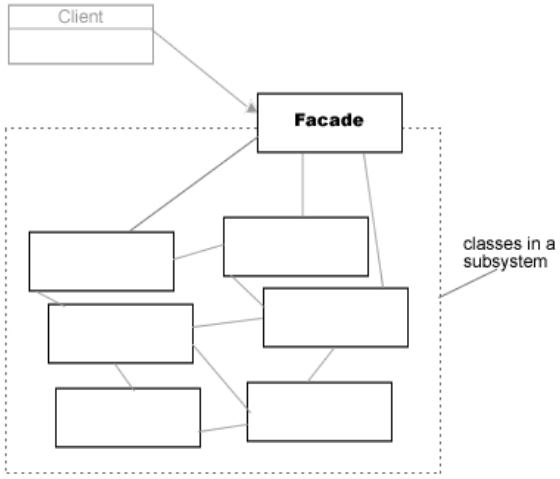


Σχήμα 8.Διάγραμμα κλάσεων προτύπου Proxy

1.2.9 Facade

Σε περίπτωση που, για την εκτέλεση μίας εργασίας, διαθέσιμο είναι ένα περίπλοκο σύνολο αλληλεπιδρώντων κλάσεων (έστω σύνολο A) οι οποίες συνδυάζονται με διάφορους τρόπους, τότε ο προγραμματιστής μπορεί να

αποκρύψει αυτόν τον ιστό αλληλεπιδράσεων "καλύπτοντας" τις επίμαχες μονάδες λογισμικού με μία κλάση η οποία εξάγει μία απλή και εύχρηστη διασύνδεση, ένα σύνολο κατάλληλων δημοσίων μεθόδων το οποίο χρησιμοποιεί εσωτερικά το σύνολο Α. Η διασύνδεση αυτή μπορεί ακολούθως να αξιοποιηθεί για την παραγωγή νέου κώδικα χωρίς ανάγκη άμεσης καταφυγής στις περίπλοκες κλάσεις του Α, αποτελεί δηλαδή μία "βιτρίνα" υψηλού επιπέδου. Ένα μη αντικειμενοστρεφές παράδειγμα εφαρμογής του προτύπου Facade είναι ορισμένες συναρτήσεις της πρότυπης βιβλιοθήκης της C οι οποίες αποτελούν απλουστευμένες εκδοχές αντίστοιχων κλήσεων συστήματος του Unix.

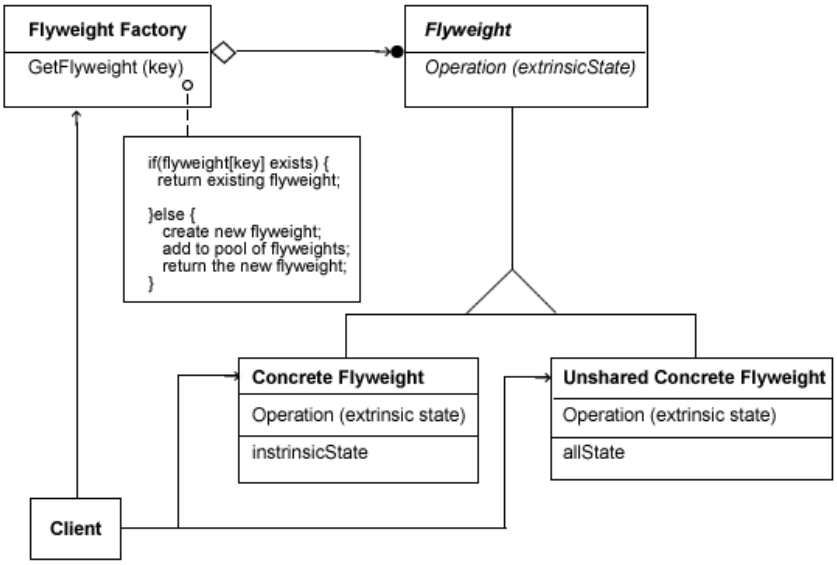


Σχήμα 9. Διάγραμμα κλάσεων προτύπου Πρόσοψη

1.2.10 Flyweight

Το πρότυπο σχεδίασης "Flyweight" προτείνει την κοινή χρήση κάποιων αντικειμένων ώστε να είναι αποδοτικές και διαχειρίσιμες καταστάσεις στις

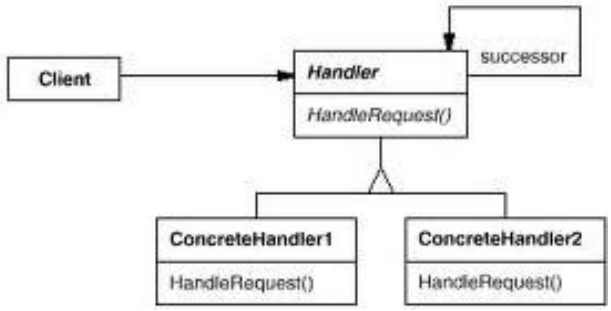
οποίες υπάρχει πολύ μεγάλος αριθμός αντικειμένων. Ένα flyweight δηλαδή, είναι ένα αντικείμενο το οποίο μπορεί να χρησιμοποιηθεί ταυτόχρονα από πολλές συλλογές αντικειμένων.



Σχήμα 10. Διάγραμμα κλάσεων προτύπου Flyweight

1.2.11 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.

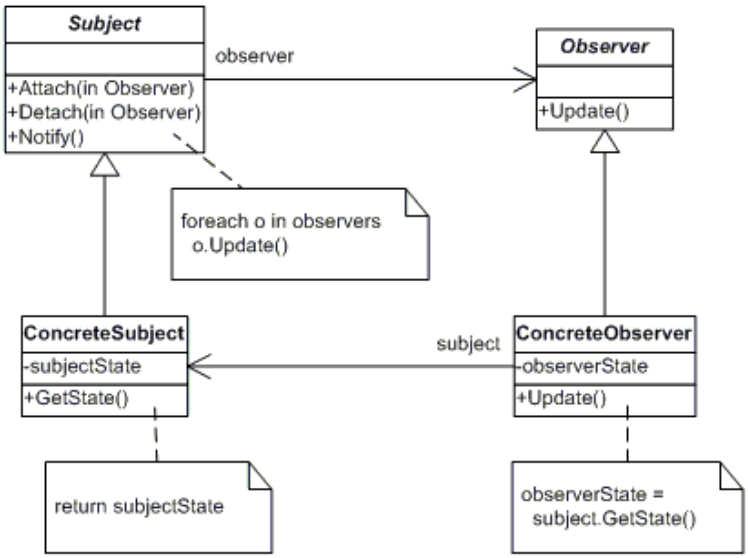


Σχήμα 11. Διάγραμμα κλάσεων προτύπου Chain of Responsibility

1.2.12 Observer

Σε περιπτώσεις που κάποια αντικείμενα (*παρατηρητές*) ενδιαφέρονται να λαμβάνουν ειδοποιήσεις για τυχόν αλλαγές στην κατάσταση κάποιου άλλου αντικειμένου A (π.χ. τροποποιήσεις τιμών κάποιων πεδίων του), υπάρχουν δύο προσεγγίσεις για την υλοποίηση αυτών των ενημερώσεων: είτε τις κατάλληλες στιγμές το A να καλεί προκαθορισμένες μεθόδους των παρατηρητών και να τους μεταβιβάζει έτσι δεδομένα, είτε οι παρατηρητές να καλούν μία μέθοδο του A για να λαμβάνουν κατά βούληση πληροφορίες. Η ενδιαφέρουσα περίπτωση είναι η πρώτη καθώς μόνον το A γνωρίζει πότε υπάρχουν αλλαγές στην κατάσταση του και τέτοιες αλλαγές είναι που πρέπει να πυροδοτούν τις ενημερώσεις. Ο απλούστερος τρόπος είναι να διατηρεί μία συνδεδεμένη λίστα με όλους τους παρατηρητές που κατά καιρούς έχουν εκδηλώσει ενδιαφέρον και να καλεί τις κατάλληλες μεθόδους τους τις κατάλληλες στιγμές. Όμως αυτή η λύση προκαλεί προβλήματα κλειστότητας καθώς ένας άλλος τύπος παρατηρητή μπορεί να έχει διαφορετικές μεθόδους, ενώ το A παρουσιάζει υψηλή σύζευξη με άλλες κλάσεις. Το πρότυπο Observer δίνει μία λύση σε αυτό το πρόβλημα: ορίζει μία διασύνδεση Observable, η οποία περιέχει μία μέθοδο register (για την εκούσια δήλωση παρατηρητών) και μία μέθοδο unregister (για την εκούσια αποχώρηση παρατηρητών), και μία διασύνδεση Observer, με μία μέθοδο synchronize για τη μεταβίβαση ενημερωμένων δεδομένων. Η κλάση κάθε αντικειμένου A πρέπει να υλοποιεί τη διασύνδεση Observable και η κλάση κάθε παρατηρητή τη

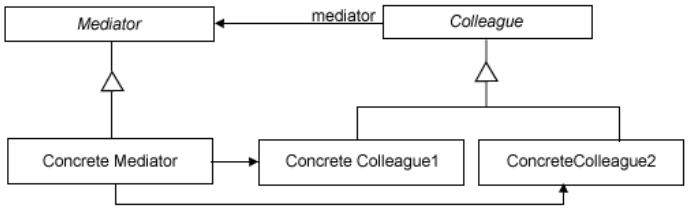
διασύνδεση Observer. Ένα αντικείμενο Observable διατηρεί μία λίστα με αντικείμενα του αφηρημένου τύπου Observer τα οποία έχουν καλέσει τη μέθοδο register του εν λόγω αντικειμένου. Οι εγγραφές αυτές χρησιμοποιούνται τις κατάλληλες χρονικές στιγμές για την ειδοποίηση των παρατηρητών.



Σχήμα 12. Διάγραμμα κλάσεων προτύπου Observer

1.2.13 Mediator

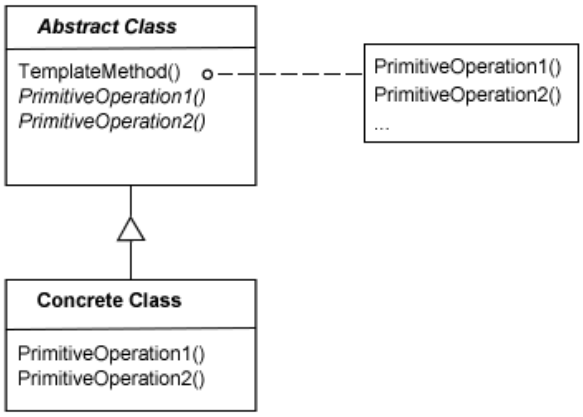
Το πρότυπο σχεδίασης "Μεσολαβητής" προσδιορίζει ένα αντικείμενο που ενσωματώνει ένα σύνολο από άλλα αντικείμενα και ορίζει το πως αυτά θα επικοινωνούν μεταξύ τους. Το πρότυπο αυτό μειώνει την σύζευξη με το να μην επιτρέπει την άμεση επικοινωνία μεταξύ δύο αντικειμένων και επιτρέποντας στον χρήστη να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



Σχήμα 13. Διάγραμμα κλάσεων προτύπου Mediator

1.2.14 Template Method

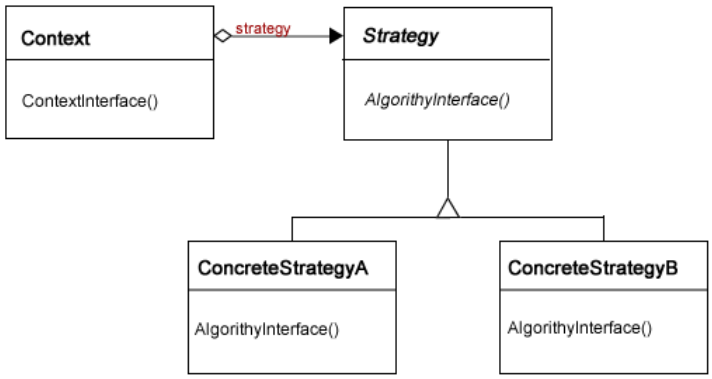
Το πρότυπο αυτό είναι εκλέπτυνση του Strategy για την περίπτωση που κάθε αλγόριθμος έχει πολλαπλές παραλλαγές οι οποίες διαφέρουν σε ορισμένα βήματα αλλά κάποια άλλα σημεία τους είναι κοινά. Τότε για κάθε αλγόριθμο μπορούμε να ορίσουμε ένα στοιχείο αφαίρεσης Β το οποίο υλοποιεί τη διασύνδεση Α και με τη σειρά του κληρονομείται από πολλές παραλλαγές του αλγορίθμου. Το κάθε Β περιέχει την υλοποίηση των αμετάβλητων βημάτων και, στα σημεία που οι παραλλαγές διαφέρουν, καλεί αφηρημένες προστατευμένες μεθόδους. Οι μέθοδοι αυτές υλοποιούνται διαφορετικά σε κάθε παραλλαγή που κληρονομεί το Β.



Σχήμα 64. Διάγραμμα κλάσεων προτύπου Template Method

1.2.15 Strategy

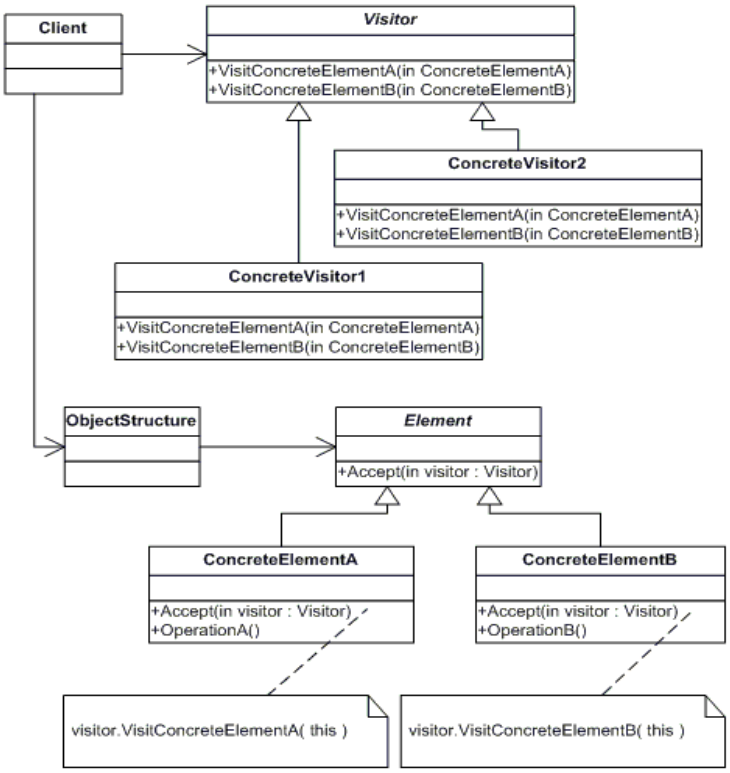
Όταν είναι διαθέσιμοι πολλαπλοί τρόποι επίλυσης ενός προβλήματος (π.χ. διαφορετικοί αλγόριθμοι), είναι προτιμότερο ο καθένας από αυτούς να μην υλοποιείται μέσα στις κλάσεις-πελάτες που τον χρησιμοποιούν (π.χ. ως ιδιωτική μέθοδος), έτσι ώστε οι πελάτες να έχουν μικρότερη περιπλοκότητα και οι διάφοροι αλγόριθμοι να είναι επαναχρησιμοποιήσιμοι και προσπελάσιμοι από πολλαπλά εξωτερικά προγράμματα. Με το πρότυπο Strategy όλοι οι διαφορετικοί αλγόριθμοι ορίζονται ως ξεχωριστές κλάσεις που υλοποιούν μία κοινή διασύνδεση A και οι πελάτες διατηρούν ως πεδίο μία αναφορά προς τον αφηρημένο τύπο A. Στο πεδίο αυτό δίνεται τιμή μέσω του ορίσματος κάποιας μεθόδου του πελάτη (πιθανώς του κατασκευαστή του), έτσι ώστε η ταυτότητα του εκάστοτε χρησιμοποιούμενου αλγορίθμου από όλους τους διαθέσιμους για την εκτέλεση της ζητούμενης εργασίας να είναι εύκολα παραμετροποιήσιμη, με μία απλή αλλαγή αυτού του ορίσματος κατά την κλήση της προαναφερθείσας μεθόδου.



Σχήμα 75. Διάγραμμα κλάσεων προτύπου Strategy

1.2.16 Visitor

Το πρότυπο Visitor είναι ένας τρόπος διαχωρισμού κάποιου αλγορίθμου, ο οποίος πιθανώς εκφράζει μία επέκταση στη λειτουργικότητα μίας ιεραρχίας κλάσεων, από τις εν λόγω κλάσεις. Στόχος είναι η σταδιακή προσθήκη νέων δυνατοτήτων σε όλες τις κλάσεις που υλοποιούν μία διασύνδεση Α χωρίς να χρειάζεται να τροποποιηθούν οι κλάσεις αυτές και χωρίς το εξωτερικό πρόγραμμα να γνωρίζει τους ακριβείς τύπους των αντικειμένων (μπορεί να χειρίζεται τα τελευταία με αφηρημένες αναφορές τύπου Α). Το μέσον για την επίτευξη του σκοπού αυτού είναι η ομαδοποίηση όλων των μεθόδων που περιγράφουν τη νέα λειτουργικότητα για κάθε κλάση της ιεραρχίας σε μία νέα, ξεχωριστή κλάση Visitor η οποία δεν κληρονομεί από την Α. Η Visitor περιέχει πολυμορφικές μεθόδους visit η καθεμία από τις οποίες δέχεται διαφορετικού τύπου όρισμα· οι πιθανοί τύποι είναι οι παραγόμενες κλάσεις που υλοποιούν την Α. Κάθε μέθοδος visit περιγράφει τη νέα λειτουργικότητα που αντιστοιχεί στην κλάση του ορίσματος της (π.χ. μία επεξεργασία των δεδομένων της εν λόγω κλάσης) αλλά οι μέθοδοι αυτές δεν καλούνται άμεσα από τον προγραμματιστή. Αντιθέτως το εξωτερικό πρόγραμμα, για κάθε αντικείμενο Β της ιεραρχίας στο οποίο θέλει να προσθέσει τη νέα λειτουργικότητα, καλεί μία μέθοδο accept του Β με όρισμα το αντικείμενο Visitor. Η accept πρέπει να προδιαγράφεται από την Α και να υλοποιείται σε κάθε παραγόμενη κλάση, ενώ το μόνον που κάνει είναι να καλεί με τη σειρά της τη μέθοδο visit του ορίσματος της με όρισμα το αντικείμενο όπου περιέχεται. Έτσι τελικώς γίνεται η κατάλληλη επεξεργασία, με κλήση της κατάλληλης μεθόδου, χωρίς το εξωτερικό πρόγραμμα να γνωρίζει καν τον ακριβή παραγόμενο τύπο της εκάστοτε Β. Αν πολλές διαφορετικές επεκτάσεις είναι επιθυμητές, τότε μπορεί η κλάση Visitor να γίνει διασύνδεση και να υλοποιείται με διαφορετικές κλάσεις για κάθε ζητούμενη επέκταση.



Σχήμα 86. Διάγραμμα κλάσεων προτύπου Visitor

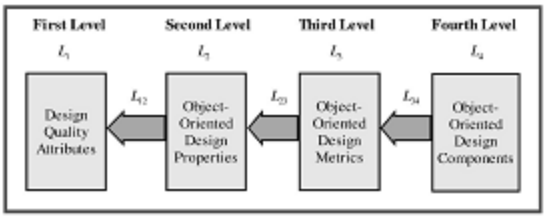
1.3 Μετρικές Ποιότητας

Η ζήτηση για την ποιότητα του λογισμικού συνεχίζει να εντείνει λόγω της αυξανόμενης εξάρτησης της κοινωνίας μας σχετικά με το λογισμικό και οι συχνά καταστρεπτικές συνέπειες που έχουν ένα σφάλμα του λογισμικού μπορεί να έχουν από την άποψη της ζωής, οικονομική ζημία, ή το χρόνο καθυστέρησης.Σήμερα για τα συστήματα λογισμικού πρέπει να εξασφαλιστεί η συνεπής και χωρίς λάθη λειτουργία κάθε φορά που χρησιμοποιούνται. Αυτή η ζήτηση για τη βελτίωση της ποιότητας του λογισμικού έχει ως αποτέλεσμα η ποιότητα να είναι περισσότερο διαφοροποιημένη μεταξύ των προϊόντων από ό, τι ήταν πριν. Σε μια αγορά άκρως ανταγωνιστική, η σημασία της παροχής ποιότητας δεν είναι πλέον ένα πλεονέκτημα, αλλά ένα απαραίτητο στοιχείο για την επιτυχία των εταιρειών. Ενώ υπάρχει ενιαία συμφωνία που χρειαζόμαστε ποιότητας λογισμικού, το ζήτημα του πώς, πότε, και όπου μπορείτε μέτρηση και διασφάλιση της ποιότητας είναι πολύ από πάγια ζητήματα. Η μετάβαση στο αντικειμενοστρεφές παράδειγμα έχει αλλάξειτα στοιχεία που χρησιμοποιούμε για την αξιολόγηση της ποιότητας λογισμικού. Παραδοσιακές μετρικές λογισμικού που αξιολογούν χαρακτηριστικά όπως το μέγεθος, την πολυπλοκότητα ,τις επιδόσεις,και ποιότητας πρέπει να αλλάξουν και να στηρίζονται σε κάποιες διαφορετικές έννοιες ,όπως η ενθυλάκωση, κληρονομικότητα, και πολυμορφισμός που είναι συνυφασμένες με αντικείμενο τον προσανατολισμό.Αυτό έχει οδηγήσει στον ορισμό πολλών νέων μετρικών για τη μέτρηση των προϊόντων της αντικειμενοστρεφούς προσέγγισης. Ωστόσο, οι νέες αντικειμενοστρεφείς μετρικές διαφοροποιούνται στο πώς χρησιμοποιούνται για τη μέτρηση όταν αυτές εφαρμόζονται. Πολλές από τις νεότερες μετρήσεις έχουν επικυρωθεί μόνο με μικρά, και μερικές φορές μη ρεαλιστικά συνόλα δεδομένων και, κατά συνέπεια, η πρακτική εφαρμοσιμότητα και αποτελεσματικότητα των μετρικών σε μεγάλα σύνθετα έργα, όπως όπως αυτές που συναντώνται σε ένα βιομηχανικό περιβάλλον δεν είναι

γνωστά. Τέλος, αν ο στόχος είναι η εκτίμηση της ποιότητας των εξωτερικών χαρακτηριστικών του προϊόντος και όχι απλώς η συλλογή μεμονωμένων μετρήσεων, τότε πρέπει να υπάρχει ένας καλά καθορισμένος τρόπος συνδεσής τους.

Πρόσφατα, ένα απαραίτητο πλαίσιο για τη δημιουργία προϊόντων με βάση τα μοντέλα της ποιότητας έχει αναπτυχθεί από Dromey. Απευθύνεται σε μερικά από τα προβλήματα των προηγούμενων μοντέλων όπως το McCall και ISO 9126.Το πλαίσιο είναι μια μεθοδολογία για την ανάπτυξη των μοντέλων ποιότητας,παρέχοντας μια προσέγγιση που θα διασφαλίζει ότι οι χαμηλότερου επιπέδου λεπτομέρειες είναι υπολογίσιμες. Υπάρχουν τρία κύρια στοιχεία στο μοντέλο ποιότητας Dromey: ιδιότητες του προϊόντος που επηρεάζουν την ποιότητα,ένα σύνολο ποιοτικών χαρακτηριστικών ψηλού επιπέδου,και ένα μέσο σύνδεσης τους .

Πίνακας 1. Επίπεδα και σύνδεσμοι σε QMOOD



Η μεθοδολογία που χρησιμοποιήθηκε για την ανάπτυξη του ιεραρχικού μοντέλου ποιότητας για Αντικειμενοστρεφή Σχεδιασμό (QMOOD) εκτείνεται της μεθοδολογίας του Dromey μοντέλου και περιλαμβάνει τα βήματα που φαίνονται στο σχήμα. Το σχήμα δείχνει το τέσσερα επίπεδα (L1 μέσω L4) που χρησιμοποιούνται για να συνδέσετε τα τέσσερα επίπεδα σε QMOOD . Ενώ για τον καθορισμό των επιπέδων περιλαμβάνονται ο προσδιορισμός των ποιοτικών χαρακτηριστικών του σχεδιασμού, τις ιδιότητες ποιότητας που μεταφέρουν ,object oriented μετρήσεις,και συστατικά αντικειμενοστραφούς σχεδιασμού,

ορίζοντας τη χαρτογράφηση που αφορά τη σύνδεση παρακείμενων επιπέδων με σύνδεση ένα χαμηλότερο επίπεδο στο επόμενου ψηλότερο επίπεδο.

Το ISO-9126 αποδίδει «λειτουργικότητα», «αξιοπιστία», "απόδοση", "χρηστικότητα," "συντηρησιμότητα," και "portability" επιλέχθηκαν ως το αρχικό σύνολο χαρακτηριστικών ποιότητας στο μοντέλο QMOOD. Αυτό το σύνολο των χαρακτηριστικών ήταν μεμονωμένα για να δούμε αν συνέβαλε στην αξιολόγηση τον καθορισμό της ποιότητας του σχεδιασμού και αν το σύνολο ήταν αρκετά ευρύ ώστε να περιλαμβάνει όλες τις πτυχές της ποιότητας του σχεδιασμού. Τα χαρακτηριστικά "Αξιοπιστία" και "χρηστικότητα" αποκλείστηκαν από το σύνολο λόγω της προφανής κλίση τους προς την εφαρμογή και όχι τον σχεδιασμό. Ο όρος «δυνατότητα μεταφοράς» είναι πιο κατάλληλος για το πλαίσιο της εφαρμογής της ποιότητας λογισμικού και αντικαθίστανται με "επεκτασιμότητα", το οποίο αντανακλά καλύτερα αυτό το χαρακτηριστικό σε σχέδια.Παρομοίως, ο όρος "αποδοτικότητα" αντικαταστάθηκε με «αποτελεσματικότητα», που περιγράφει καλύτερα την ποιότητα των σχεδίων και υποδειγμάτων. Ο όρος "συντηρησιμότητα" επίσης υπονοεί την ύπαρξη ενός προϊόντος λογισμικού και αντικαθίσταται από "κατανόηση", το οποίο επικεντρώνεται περισσότερο από τα χαρακτηριστικά του σχεδιασμού.Ένας σημαντικός στόχος του object-oriented σχεδιασμού για την υλοποίηση έχει ανάπτυξη, αξιόπιστο, προσαρμόσιμο και ευέλικτο λογισμικό συστημάτων. Ένας τρόπος για να επιτευχθεί αυτό με την ενθάρρυνση της επαναχρησιμοποίησης σε όλα τα επίπεδα της ανάπτυξης. Ο στόχος αυτός δικαιολογεί την "επαναχρησιμοποίηση" ως ένα σημαντικό χαρακτηριστικό του object-oriented σχεδιασμού.Ευελιξία των συστημάτων λογισμικού είναι επίσης ένα σημαντικό χαρακτηριστικό ανάπτυξης και τελικού χρήστη.Η αρχική δέσμη ποιότητας σχεδιασμού που αποδίδει η QMOOD είναι:«λειτουργικότητα», "αποτελεσματικότητα», «κατανόηση" ,"ικανότητα», «επεκτασιμότητα», «επαναχρησιμοποίηση» και «ευελιξία».Αυτό το σύνολο των γνωρισμάτων είναι αρκετά ευρύ ώστε να επιτρέπει επιθυμητά χαρακτηριστικά των object-oriented

συστήματων να προσδιοριστούν.Η επικύρωση του μοντέλου το επιβεβαιώνει. Ωστόσο, αυτό το σύνολο των χαρακτηριστικών ποιότητας δεν είναι αποκλειστική, και μπορεί εύκολα να τροποποιηθεί ώστε αντιπροσωπεύουν διαφορετικούς στόχους και σκοπούς.Αυτά τα ποιοτικά χαρακτηριστικά, όπως η συνολική ποιότητα ,είναι οι ίδιες αφηρημένες έννοιες και ως εκ τούτου δεν είναι άμεσα παρατηρήσιμα. Επίσης, όπως η ποιότητα, δεν υπάρχουν καθολικά συμφωνημένων ορισμών για κάθε μία από την υψηλού επιπέδου ποιότητας ιδιότητες της QMOOD. Ο Πίνακας συνοψίζει τους ορισμούς από τις ιδιότητες ποιότητας που χρησιμοποιούνται στην ανάπτυξη QMOOD .

Πίνακας 2. Ορισμοί Ιδιοτήτων Ποιότητας

Quality Attribute	Definition
Reusability	Reflects the presence of object-oriented design characteristics that allow a design to be reapplied to a new problem without significant effort.
Flexibility	Characteristics that allow the incorporation of changes in a design. The ability of a design to be adapted to provide functionally related capabilities.
Understandability	The properties of the design that enable it to be easily learned and comprehended. This directly relates to the complexity of the design structure.
Functionality	The responsibilities assigned to the classes of a design, which are made available by the classes through their public interfaces.
Extendibility	Refers to the presence and usage of properties in an existing design that allow for the incorporation of new requirements in the design.
Effectiveness	This refers to a design's ability to achieve the desired functionality and behavior using object-oriented design concepts and techniques.

Σχεδιασμός ιδιότητες είναι απτές έννοιες που μπορεί να είναι άμεσα αξιολογείται με εξέταση την εσωτερική και εξωτερική δομή ,σχέση, και η λειτουργικότητα των στοιχείων σχεδιασμού, ιδιότητες, τις μεθόδους, και τις τάξεις. Μια αξιολόγηση μιας τάξης ορισμό για τις εξωτερικές σχέσεις της (τύπος κληρονομικότητας)με άλλα μαθήματα και την εξέταση του εσωτερικού του στοιχεία, ιδιότητες, μεθόδους και αποκαλύπτει σημαντικές πληροφορίες που αποτυπώνει αντικειμενικά την δομική και λειτουργικά χαρακτηριστικά μιας

κατηγορίας και τα αντικείμενά της.Οι ιδιότητες του σχεδιασμού της αφαίρεσης, ενθυλάκωση, σύζευξη, τη συνοχή, την πολυπλοκότητα και το μέγεθος του σχεδιασμού που χρησιμοποιείτε συχνά ως αντιπροσωπευτικά της ποιότητας του σχεδιασμού χαρακτηριστικά τόσο δομικά όσο και αντικειμενοστρεφή ανάπτυξης. Μηνύματα, σύνθεση, κληρονομικότητα, πολυμορφισμό, και ιεραρχίες κατηγορίας αντιπροσωπεύουν το νέο σχέδιο έννοιες που έχουν εισαχθεί από το object-oriented σχεδιασμό και ως εκ τούτου είναι ζωτικής σημασίας για την ποιότητα του αντικειμενοστραφή σχεδιασμού. Η αρχική έκδοση του QMOOD περιλαμβάνειτα δύο σύνολα των ιδιοτήτων που ορίζονται στον Πίνακα.

Πίνακας 3. Ορισμοί Σχεδιασμού

Design Property	Definition
Design Size	A measure of the number of classes used in a design.
Hierarchies	Hierarchies are used to represent different generalization-specialization concepts in a design. It is a count of the number of non-inherited classes that have children in a design.
Abstraction	A measure of the generalization-specialization aspect of the design. Classes in a design which have one or more descendants exhibit this property of abstraction.
Encapsulation	Defined as the enclosing of data and behavior within a single construct. In object-oriented designs the property specifically refers to designing classes that prevent access to attribute declarations by defining them to be private, thus protecting the internal representation of the objects.
Coupling	Defines the interdependency of an object on other objects in a design. It is a measure of the number of other objects that would have to be accessed by an object in order for that object to function correctly.
Cohesion	Assesses the relatedness of methods and attributes in a class. Strong overlap in the method parameters and attribute types is an indication of strong cohesion.
Composition	Measures the "part-of," "has," "consists-of," or "part-whole" relationships, which are aggregation relationships in an object-oriented design.
Inheritance	A measure of the "is-a" relationship between classes. This relationship is related to the level of nesting of classes in an inheritance hierarchy.
Polymorphism	The ability to substitute objects whose interfaces match for one another at run-time. It is a measure of services that are dynamically determined at run-time in an object.
Messaging	A count of the number of public methods that are available as services to other classes. This is a measure of the services that a class provides.
Complexity	A measure of the degree of difficulty in understanding and comprehending the internal and external structure of classes and their relationships.

Κάθε μια από τις ιδιότητες του σχεδιασμού που προσδιορίζονται στο μοντέλο QMOOD αντιπροσωπεύουν ένα χαρακτηριστικό ή χαρακτηριστικά ενός σχεδίου .

Για αντικειμενοστρεφή σχέδια, αυτή η πληροφορία θα πρέπει να περιλαμβάνει τον ορισμό των τάξεων, ιεραρχίες τάξεων, και δηλώσεις λειτουργίας, μαζί με όλες τις παραμέτρους των τύπων και δηλώσεις.

Μια έρευνα των υφιστάμενων μετρικών σχεδιασμού αποκάλυψε ότι υπάρχουν διάφορες μετρήσεις που μπορούν να τροποποιηθούν και να χρησιμοποιηθούν στην εκτίμηση ορισμένων ιδιοτήτων σχεδιασμού, όπως αφαίρεση, μηνυμάτων, και την κληρονομιά.

Ωστόσο, υπάρχουν αρκετές άλλες ιδιότητες σχεδίασης, όπως η ενθυλάκωση και σύνθεση, για τις οποίες δεν υπάρχουν μετρήσεις. Επίσης, ενώ οι μετρήσεις για την εκτίμηση της πολυπλοκότητας, συνοχή, και σύζευξη έχουν ήδη προσδιοριστεί αυτές οι μετρήσεις απαιτούν μία σχεδόν πλήρη εφαρμογή των τάξεων πριν να μπορούν να υπολογίζονται. Αυτό οδήγησε στον ορισμό των πέντε νέων μετρήσεων, τα δεδομένα έχουν πρόσβαση μετρικό (DAM), η άμεση σύζευξη μετρική κατηγορίας (DCC), η συνοχή μεταξύ των μεθόδων μέτρησης της κατηγορίας (CAM), η μέτρο της συγκέντρωσης μετρική (MOA), και το μέτρο της λειτουργικής μετρικό αφαίρεσης (MFA) που θα μπορούσε να υπολογίζονται από τις πληροφορίες σχεδιασμού μόνον. Η πλήρης σουίτα των μετρήσεων που χρησιμοποιούνται σε QMOOD απαριθμούνται και περιγράφονται στον πίνακα.

Πίνακας 4. Περιγραφή Μετρικών

METRIC	NAME	DESCRIPTION
DSC	Design Size in Classes	This metric is a count of the total number of classes in the design.
NOH	Number of Hierarchies	This metric is a count of the number of class hierarchies in the design.
ANA	Average Number of Ancestors	This metric value signifies the average number of classes from which a class inherits information. It is computed by determining the number of classes along all paths from the "root" class(es) to all classes in an inheritance structure.
DAM	Data Access Metric	This metric is the ratio of the number of private (protected) attributes to the total number of attributes declared in the class. A high value for DAM is desired. (Range 0 to 1)
DCC	Direct Class Coupling	This metric is a count of the different number of classes that a class is directly related to. The metric includes classes that are directly related by attribute declarations and message passing (parameters) in methods.
CAM	Cohesion Among Methods of Class	This metric computes the relatedness among methods of a class based upon the parameter list of the methods [3]. The metric is computed using the summation of the intersection of parameters of a method with the maximum independent set of all parameter types in the class. A metric value close to 1.0 is preferred. (Range 0 to 1)
MOA	Measure of Aggregation	This metric measures the extent of the part-whole relationship, realized by using attributes. The metric is a count of the number of data declarations whose types are user defined classes.
MFA	Measure of Functional Abstraction	This metric is the ratio of the number of methods inherited by a class to the total number of methods accessible by member methods of the class. (Range 0 to 1)
NOP	Number of Polymorphic Methods	This metric is a count of the methods that can exhibit polymorphic behavior. Such methods in C++ are marked as virtual.
CIS	Class Interface Size	This metric is a count of the number of public methods in a class
NOM	Number of Methods	This metric is a count of all the methods defined in a class.

Τα στοιχεία του σχεδιασμού που είναι αναγνωρίσιμα να καθορίσουν την αρχιτεκτονική ενός αντικειμενοστραφούς σχεδιασμού είναι αντικείμενα, τάξεις, και οι σχέσεις μεταξύ τους. Αντικείμενα ενσωματώσουν δομές δεδομένων που αντιπροσωπεύουν τα χαρακτηριστικά αντικείμενα μιας κατηγορίας.

Μια σειρά από λειτουργίες (μέθοδοι) που ορίζονται στις τάξεις και μπορούν να λειτουργούν σε δεδομένα ενθυλακώνονται από το αντικείμενο. Η ποιότητα του αντικειμένου καθορίζεται από τα συστατικά του, περιλαμβάνει ιδιότητες, οι μέθοδοι, ή άλλα αντικείμενα (σύνθεση).

Άλλο συστατικό που μπορεί να εντοπιστεί σε object-oriented σχεδιασμό είναι γενίκευση-ειδίκευση δομές, ή κατηγορία hier-Archies που οργανώνουν τις οικογένειες των σχετικών κατηγοριών. Ετσι, ένα σύνολο των στοιχείων που μπορούν να συμβάλουν στην ανάλυση, αντιπροσωπεύουν, και εφαρμόζουν ένα αντικειμενοστραφή σχεδιασμό θα πρέπει να περιλαμβάνει μεθόδους, αντικείμενα (μαθήματα), σχέσεις, και την κατηγορία ιεραρχίας.

Γενικά, όλες οι αντικειμενοστρεφείς γλώσσες προγραμματισμού παρέχουν συντακτικές κατασκευές που αντιπροσωπεύουν θεμελιώδη στοιχεία σχεδιασμού. Από μια object-oriented γλώσσα προγραμματισμό (C + +) που χρησιμοποιήθηκε για την εκπροσώπηση του σχεδιασμού, η ποιότητα ενός σχεδιασμού μπορεί εύκολα να αξιολογείται από την αξιολόγηση της ποιότητας των εν λόγω συστατικών.

Φτάνοντας σε ένα σύνολο προτύπων ποιότητας που φέρει τις ιδιότητες για κάθε συνιστώσα του σχεδιασμού είναι μια εμπειρική διαδικασία. Η διαδικασία καθοδηγήθηκε από μια σειρά από ερωτήματα, όπως: "Ποιος είναι ο ρόλος που παίζει το στοιχείο στο σχεδιασμό (σκοπός); "" Τι είναι η σημασία του στοιχείου στο σχεδιασμό; "" Πώς είναι το στοιχείο που εκπροσωπείται; "" Ποιές είναι οι διαφορετικές μορφές του συστατικού; "

Σημαντικές εκτιμήσεις συμπεριλαμβανομένης της ποιότητας μεταφοράς ιδιοκτησίας ήταν η απαίτηση ότι οι πληροφορίες που αξιολογεί θα πρέπει να είναι διαθέσιμες κατά τη διάρκεια της φάσης του σχεδιασμού, και ότι υπάρχουν αρκετές πληροφορίες στην αναπαράσταση του συστατικού προς προσδιορισμό και αξιολόγηση. Οι ιδιότητες ποιότητας μεταφέρουν από τα χαρακτηριστικά, μεθόδους, και εξαρτήματα τάξη που χρησιμοποιήθηκαν για την εκτίμησης της ποιότητας του σχεδιασμού.

Χαρακτηριστικά είναι τα πιο θεμελιώδη συστατικά σε ένα αντικείμενο και την εκπροσώπησή τους είναι άμεσα υποστηριζόμενο σε αντικειμενοστρεφείς γλώσσες προγραμματισμού. Ένα σύνολο (δεδομένα) "χαρακτηριστικά " ιδιότητες που μπορεί άμεσα να επηρεάζουν την ποιότητα ενός αντικειμένου και, κατά

συνέπεια, την συνολική ποιότητα του σχεδιασμού, θα πρέπει να περιλαμβάνουν: Όνομα(αυτο-περιγραφικό), ενθυλάκωση (δημόσιο, ιδιωτικό, προστατευμένο), προκριματικά (sta-tic/constant), προετοιμασία (για τη δημιουργία; ναι / όχι), το μέγεθος (σε bytes), τον τύπο (πρωτόγονες ή οριστεί από το χρήστη), η σχέση (ναι /όχι), και καταμέτρηση (αξία ή δείκτη / αναφοράς).

Οι δηλώσεις μεθόδων επίσης άμεσα υποστηρίζονται από γλώσσες προγραμματισμού. Ένα σύνολο των αναγνωρίσιμων και ιδιοτήτων μιας μεθόδου που, είτε άμεσα είτε έμμεσα, επηρεάζει την ποιότητα της τάξης εντός της οποίας έχει δηλωθεί θα πρέπει να περιλαμβάνουν: όνομα(αυτο-περιγραφικό), ενθυλάκωση (ιδιωτικό, προστατευμένο, δημόσιο), τους τύπους των παραμέτρων, τον αριθμό των παραμέτρων, η παράμετρος μηχανισμού πέρασμα (από τιμή, αναφορά ή διεύθυνση), ανάλυση(στατική/δυναμική ή nonvirtual / εικονικά), και ο τύπος (constructor, destructor, con-σταντ, inline, στατική).

Μαθήματα σε περισσότερες αντικειμενοστρεφείς γλώσσες προγραμματισμού περιγράφονται με τη χρήση συντακτικών κατασκευασμάτων που είναι εύκολα αναγνωρίσιμα. Η "Goodness" μιας κατηγορίας μπορούν να αναλυθούν με βάση τα συστατικά του (ιδιότητες και μέθοδοι), και την δομική και επικοινωνιακή σχέση (αλληλεπίδραση) με άλλες κατηγορίες. Μια σειρά από ιδιότητες τάξης που μπορεί να επηρεάζουν τη συνολική ποιότητα του σχεδιασμού περιλαμβάνει: το όνομα (Αυτο-περιγραφικό), τον τύπο κληρονομικότητας και ενθυλάκωση (δημόσιων, ιδιωτικών, ή προστατευόμενη κληρονομιά), κληρονομικότητα (βάση, παράγωγα, μόνο, πολλαπλές, αφηρημένη, ή σκυρόδεμα), ο αριθμός των γονείς, τον αριθμό των παιδιών, το βάθος της κληρονομιάς, το πρότυπο τάξη, τάξη (αντικείμενο) το μέγεθος, τον αριθμό των μεθόδων (δημόσιων, ιδιωτικών, προστατεύονται), τον αριθμό των χαρακτηριστικών, τον αριθμό των εικονικών μεθό-δων, ο αριθμός των κατασκευαστών, καταστροφείς, φορείς, con-σταντ και στατικές μεθόδους, ο αριθμός των ενσωματωμένων λειτουργιών, ζεύξης, και τη συνοχή.

Η ποιότητα που φέρει τις ιδιότητες των συστατικών του σχεδιασμού ταξινομούνται με βάση τις ιδιότητες του σχεδιασμού που επηρεάζουν. Ενώ η σειρά της ποιότητας που φέρει τις ιδιότητες του θεμελιώδεις (ιδιότητες, μέθοδοι, τάξεις και) στοιχεία είναι μεγάλο, είναι λίαν επικαλυπτόμενες. Για παράδειγμα, τα χαρακτηριστικά γνωρίσματα, μεθόδους, και τα συστατικά έχουν ένα όνομα κατηγορίας. Αυτό βοηθά στην καλύτερη κατανόηση και, κατά συνέπεια, επηρεάζει (μείωση) την πολυπλοκότητα του σχεδιασμού.Οι ιδιότητες που προσδιορίζονται για τα χαρακτηριστικά, μεθόδους, και μαθήματα μπορούν όλες να ομαδοποιηθούν σε έντεκα μικρότερα σύνολα.

Πίνακας 5. Μετρικές Σχεδιασμού και Ιδιότητες

Design Property	Derived Design Metric
Design Size	Design Size in Classes (DSC)
Hierarchies	Number of Hierarchies (NOH)
Abstraction	Average Number of Ancestors (ANA)
Encapsulation	Data Access Metric (DAM)
Coupling	Direct Class Coupling (DCC)
Cohesion	Cohesion Among Methods in Class (CAM)
Composition	Measure of Aggregation (MOA)
Inheritance	Measure of Functional Abstraction (MFA)
Polymorphism	Number of Polymorphic Methods (NOP)
Messaging	Class Interface Size (CIS)
Complexity	Number of Methods (NOM)

Οι μετρήσεις Μέγεθος Σχεδιασμός των κατηγοριών (DSC) και Αριθμός Ιεραρχίες (NOH) χρησιμοποιούνται για την εκτίμηση των δύο properties Design σχεδιασμού Size and Hierarchiesin QMOOD. Abstraction αναφέρεται στην γενίκευση-ειδίκευση δομή του σχεδιασμού και αξιολογείται με βάση τον αριθμό των προγόνων theAverage (ANA) μετρική. Ο ορισμός του ακινήτου σχεδιασμού ενθυλάκωση αναφέρεται στην πρόσβαση ελέγχου των δηλώσεων χαρακτηριστικό σε μια τάξη η οποία αντικατοπτρίζεται στην περιγραφή της DAM μετρική. Class Coupling (DCC) μετρική ,οι τάξεις των σχετιζόμενων άμεσα με μια τάξη, χρησιμοποιείται για αξιολογήσει ιδιοκτησία theCouplingdesign. Η

metricsCohesion Μέθοδος της κατηγορίας (CAM) Μέτρο του Aggregation (MOA), και ανά κατηγορία μεγέθους Interface που χρησιμοποιείται για τη μέτρηση Συνοχής, Συγκέντρωση, και ιδιότητες Messagingdesign. Κληρονομικότητα αναφέρεται στην έκταση της επαναχρησιμοποίησης της λειτουργικότητας (Μετράται με τη μετρική MFA), η οποία μπορεί να επιτευχθεί δημιουργώντας υποκατηγορίες των υφισταμένων τάξεων και, επομένως, η Μετρική MFA χρησιμοποιήθηκε ως ένα μέτρο για την theInheritance ιδιοκτησίας στο σχεδιασμό QMOOD. Η ιδιοκτησία του σχεδιασμού του πολυμορφισμού μέτρο των Virtualmethods σε μια τάξη αξιολογείται από την μετρική Αριθμό Πολυμόρφων Μέθοδοι (NOP). Ο αριθμός των Μεθόδων (NOM) μετρικό που έχει χρησιμοποιηθεί ως ένα μέτρο της πολυπλοκότητας τάξης και από Chidamber Kemerer στις Σταθμισμένες μεθόδους ανά κατηγορία (WMC) μετρική του σχεδιασμού. Όταν όλες οι μέθοδοι σταθμίζονται εξίσου, η μετρική WMC έχει το ίδιο μέτρο ο αριθμός των Μεθόδων (NOM) σε μια τάξη.

Η αξιολόγηση πληροφοριών δείχνει ότι η αφαίρεση στο design διαθέτει σημαντική επιρροή στην ευελιξία, αποτελεσματικότητα, τη λειτουργικότητα, την ποιότητα και την επεκτασιμότητα.

Ενθυλάκωση θεωρείται η προώθηση της ευελιξία, επαναχρησιμοποίησης, και κατανόησης. Ενώ η χαμηλή ζεύξης θεωρείται καλό για την κατανόηση και επαναχρησιμοποίηση είναι υψηλότερα από τα μέτρα ζεύξης εμφανισμένα να επηρεάσουν αρνητικά τα ποιοτικά χαρακτηριστικά. Συνοχή θεωρείται ότι έχει σημαντική επίπτωση στη σχεδίαση είναι κατανοητό και επαναχρησιμοποίησης. Αντικείμενα επικοινωνούν με ανταλλαγή μηνυμάτων και, ως εκ τούτου, επηρεάζει άμεσα τη λειτουργικότητα και την αποτελεσματικότητα και βοηθά προωθεί την επαναχρησιμοποίηση. Ενώ η χρήση της κληρονομικότητας προωθεί εσωτερική επαναχρησιμοποίησης, τη λειτουργικότητα,επεκτασιμότητα, και την αποτελεσματικότητα, δεν έχει τη δυνατότητα να επηρεάζουν αρνητικά την ευελιξία και την κατανόηση. Ομοίως, ενώ η προσεκτική χρήση της σύνθεσης μπορεί να αυξήσει σημαντικά την εσωτερική επαναχρησιμοποίηση, τη λειτουργικότητα, και την ευελιξία, η υπερβολική και λανθασμένη χρήση του

μπορεί να κάνει ένα σχέδιο πιο δύσκολο να κατανοηθεί. Η χρήση της σύνθεση μπορεί επίσης να επηρεάσει την αποτελεσματικότητα και επεκτασιμότητα. Ενώ η χρήση πολυμορφισμού θεωρείται ότι αυξάνει την ευελιξία, επεκτασιμότητα, την αποτελεσματικότητα, τη λειτουργικότητα . Πολυπλοκότητα θεωρείται ως ένδειξη της κατανόησης ενός σχεδίου ή υποδείγματος. Σε γενικές γραμμές, όσο πολύπλοκο είναι ένα σχέδιο, τόσο πιο δύσκολο είναι να γίνει κατανοητό, το οποίο μπορεί επίσης να επηρεάζει την ευελιξία και την επαναχρησιμοποίηση.

Ένα σύμβολο επάνω βέλος δείχνει ότι η ιδιότητα του σχεδιασμού επηρεάζει θετικά την ποιότητα χαρακτηριστικό. Με βάση την ιδιότητα του σχεδιασμού για ποιοτικό χαρακτηριστικό συσχετισμού η σχετική σημασία των ατομικών ιδιοτήτων που επηρεάζουν το σχεδιασμό ενός ποιοτικού χαρακτηριστικού σταθμίζεται αναλογικά, έτσι ώστε η υπολογιζόμενη τιμή όλων των χαρακτηριστικών ποιότητας έχουν την ίδια σειρά.

Πίνακας 6. Χαρακτηριστικά ποιότητας - Σχέσεις ιδιοτήτων σχεδιασμού

	Reusability	Flexibility	Understandability	Functionality	Extendibility	Effectiveness
Design Size	↑			↑		
Hierarchies				↑		
Abstraction					↑	↑
Encapsulation		↑	↑			↑
Coupling						
Cohesion	↑		↑	↑		
Composition		↑				↑
Inheritance					↑	↑
Polymorphism		↑		↑	↑	↑
Messaging	↑			↑		
Complexity						

Το μοντέλο ποιότητας QMOOD επιτρέπει τις αλλαγές να είναι εύκολες για την αντιμετώπιση διαφορετικού συντελεστή στάθμισης, άλλα και για νέες προοπτικές και νέους στόχους. Στο χαμηλότερο επίπεδο, οι μετρήσεις χρησιμοποιούνται για την εκτίμηση αλλαγών των ιδιοτήτων στο σχεδιασμό, ή ένα διαφορετικό σύνολο ιδιοτήτων σχεδιασμού μπορεί να χρησιμοποιηθεί για να εκτιμήσει ποιοτικά χαρακτηριστικά, ή ακόμα και το σύνολο της ποιότητας που πρέπει να αξιολογηθεί μπορεί να αλλάξει.

Πίνακας 7. Τύποι Υπολογισμού για ποιοτικά χαρακτηριστικά

Quality Attribute	Index Computation Equation
Reusability	-0.25 * Coupling + 0.25 * Cohesion + 0.5 * Messaging + 0.5 * Design Size
Flexibility	0.25 * Encapsulation - 0.25 * Coupling + 0.5 * Composition + 0.5 * Polymorphism
Understandability	-0.33 * Abstraction + 0.33 * Encapsulation - 0.33 * Coupling + 0.33 * Cohesion - 0.33 * Polymorphism - 0.33 * Complexity - 0.33 * Design Size
Functionality	0.12 * Cohesion + 0.22 * Polymorphism + 0.22 Messaging + 0.22 * Design Size + 0.22 * Hierarchies
Extendibility	0.5 * Abstraction - 0.5 * Coupling + 0.5 * Inheritance + 0.5 * Polymorphism
Effectiveness	0.2 * Abstraction + 0.2 * Encapsulation + 0.2 * Composition + 0.2 * Inheritance + 0.2 * Polymorphism

2. Βιβλιογραφική Αναφορά

Στο προηγούμενο κεφάλαιο είδαμε κάποια εισαγωγικά στοιχεία για τις βιβλιοθήκες και τα πρότυπα σχεδίασης, σε αυτό το κεφάλαιο θα δούμε όλη την ερευνητική δραστηριότητα γύρω απο την επίδραση των προτύπων σχεδίασης στις βιβλιοθήκες.

2.1 ΜΕΘΟΔΟΛΟΓΙΑ ΣΥΣΤΗΜΑΤΙΚΗΣ ΑΝΑΣΚΟΠΗΣΗΣ ΤΗΣ ΒΙΒΛΙΟΓΡΑΦΙΑΣ

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.

2.1.1 Τα Ερωτήματα της έρευνας

Στη μελέτη αυτή θ ασχοληθούμε με διάφορα ζητήματα που αφορούν την μεχρι τώρα ερευνητική δραστηριότητα πάνω στα αντικειμενοσταφεί πρότυπα σχδεδίασης. Το βασικό ερώτημα που προκύπτει είναι το εξής :

- Υπάρχουν ερευνητικές μελέτες που αναφέρονται στην επίδραση των προτύπων σχεδίασης στις βιβλιοθήκες, και ποια συμπεράσματα προκύπτουν ;

2.1.2 Η Διαδικασία Αναζήτησης

Η αναζήτηση έγινε στις παρακάτω βιβλιοθήκες :

- http://www.computer.org/
- http://www.springerlink.com/
- http://www.sciencedirect.com/
- http://dl.acm.org/
- http://scholar.google.gr/

Σαν λέξεις κλειδιά χρησιμοποιήθηκαν οι λέξεις «pattern» και «framework». Απο τα αποτελέσματα της αναζήτησης αναγκαστήκαμε να εξαιρέσουμε κάποια γιατί δεν είχαν άμεση σχέση με το θέμα μας. Ο αποκλεισμός των μη σχετικών άρθρων διεξήχθη με το χέρι, σύμφωνα με τα κριτήρια συμπερίληψης ή αποκλεισμού που ορίζονται στο επόμενο υποκεφάλαιο.

2.1.3 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.
- Αποκλεισμός μελετών βάση του τίτλο τους.
- Αποκλεισμός μελετών βάση της περίληψή τους.
- Προσεκτική μελέτη των άρθρων και επιλογή των πιο σχετικών με τα πρότυπα σχεδίασης, βάση του πλήρους κειμένου.

2.1.4 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.

2.1.5 Συλλογή Δεδομένων

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

- Τύπο δημοσίευσης (περιοδικό, συνέδριο, workshop)
- Τόπο δημοσίευσης (όνομα συνεδρίου ή περιοδικού)
- Έτος δημοσίευσης
- Τόπος δημοσίευσης (όνομα συνεδρίου ή περιοδικού)

2.2 Συζήτηση των αποτελεσμλάτων

Σε αυτήν την φάση θα συζητήσουμε τα αποτελέσματα που πήραμε απο τη συλλογή των επιστημονικων άρθρων. Θα εξετάσουμε και θα αναλύσουμε το καθε άρθο ξεχωριστά.

2.2.1 Reconciling usability and interactive system architecture using patterns

- The Journal of Systems and Software 81 (2008) 1845–1852

Παραδοσιακά διαδραστικά συστήματα αρχιτεκτονικής ,όπως το MVC, αποσυνθέτουν το σύστημα σε υποσυστήματα που είναι σχετικά ανεξάρτητα, επιτρέποντας έτσι στη σχεδιαστική δουλεία να κατανέμεται ανάμεσα στις διεπαφές χρηστών και τις βασικές λειτουργίες. Τέτοιες αρχιτεκτονικές επεκτείνουν την υπόθεση ανεξαρτησίας σε χρηστικότητα, προσεγγίζοντας το σχεδιασμό της διεπαφής χρήστη ως ένα υποσύστημα που μπορεί να σχεδιαστεί και δοκιμαστεί ανεξάρτητα από την υποκείμενη λειτουργικότητα. Αυτή η καρτεσιανή διχοτομία μπορεί να είναι παραπλανητική, όπως λειτουργίες

θαμμένες στη λογική της εφαρμογής μπορεί να επηρεάσουν μερικές φορές τη χρηστικότητα του συστήματος.

Οι έρευνές μας μοντελοποιούν τοις σχέσεις μεταξύ των εσωτερικών χαρακτηριστικών του λογισμικού και των εξωτερικά ορατών παραγόντων χρηστικότητας. Εμείς προτείνουμε ένα προσεγγιστικό μοτίβο για την αντιμετώπιση αυτών των σχέσεων.

Καταλήγουμε συζητώντας πως αυτά τα πρότυπα οδηγούν σε ένα μεθοδολογικό πλαίσιο (framework) για τη βελτίωση διαδραστικών αρχιτεκτονικών συστημάτων, και πως αυτά τα πρότυπα μπορούν να υποστηρίξουν την ενσωμάτωση της ευχρηστίας στη διαδικασία σχεδίασης του λογισμικού.

Σε αυτήν την εργασία ,αρχικά ταυτοποιήσαμε συγκεκριμένα σενάρια για το πως αόρατα συστατικά λογισμικού μπορούν να έχουν επίδραση στη χρηστικότητα των διαδραστικών συστημάτων. Στη συνέχεια παραδώσαμε μια λίστα με πρότυπα που λύνουν το πρόβλημα που περιγράφεται στα σενάρια . Αυτή η ερευνητική προσπάθεια μπορεί να επωφελήσει σχεδιαστές αρχιτεκτονικού λογισμικού και προγραμματιστές που μπορούν να χρησιμοποιήσουν την προσέγγισή μας με δυο διαφορετικούς τρόπους.

- Πρώτον, τα σενάρια μπορούν να χρησιμοποιηθούν σαν μια λίστα ελέγχου για να διαπιστωθεί αν χαρακτηριστικά χρηστικότητας (εξωτερικά χαρακτηριστικά) έχουν ληφθεί υπόψη στον σχεδιασμό των χαρακτηριστικών και των συστατικών UI.
- Δεύτερον , τα πρότυπα μπορούν να βοηθήσουν τον σχεδιαστή να ενσωματώσει ορισμένες απο τις ανησυχίες χρηστικότητας στο σχεδιασμό.

Περισσότερο απο τον καθορισμό ενός καταλόγου σεναρίων που περιγράφουν τα αποτελέσματα των αόρατων ιδιοτήτων του λογισμικού πάνω στην ευχρηστία του λογισμικού , ο μακροπρόθεσμος στόχος μας είναι να δημιουργήσουμε και να επικυρώσουμε ένα ολοκληρωμένο πλαίσιο (framework) για τη ταυτοποίηση των

σεναρίων. Ο στόχος αυτού του πλαισίου είναι να καθορίσει αυτα τα πρότυπα σαν μια σχέση μεταξύ παραγόντων ποιότητας λογισμικού και παραγόντων ευχρηστίας. Σε αυτήν την εργασία έχουμε προτείνει διαφορετικά HCI και πρότυπα σχεδίασης λογισμικού σαν λύση στα προβλήματα που περιγράφονται σε αυτά τα σενάρια αλλα και σε παρόμοια. Κάθε πρότυπο έχει μια σειρά απο προβλήματα που πρέπει να επιλυθούν και ένα σύνολο στόχων που πρέπει να επιτευχθούν.

Αναμένουμε οτι καθώς αποκτούμε καλύτερη κατανόηση της σχέσης μεταξύ αλληλεπιδραστικών προτύπων σχεδίασης και προτύπων αρχιτεκτονικού λογισμικού , αυτή η γνώση θα επηρεάσει την εξέλιξη των βάσεων στην αρχιτεκτονική σχεδίαση και στη βιβλιοθήκη σχεδιασμού GUI. Για την ακρίβεια , αυτό έχει ήδη ξεκινήσει.

Όλο και περισσότερο, οι προγραμματιστές κάνουν σωστή χρήση των βιβλιοθηκών GUI και σέβονται τις κατευθυντήριες γραμμες για τον σχεδιασμό διεπαφής με τρόπο που αυξάνει σημαντικά τη χρηστικότητα των διαδραστικών εφαρμογών. Ωστόσο, περισσότερα μπορούν να γίνουν σ αυτή την κατεύθυνση , και η προσέγγιση που έχουμε περιγράψει σε αυτήν την εργασία είναι μια προσπάθεια για να φτιάξουμε μια καλύτερη και πιο συστηματική κατανόηση του τρόπου με τον οποίο η χρηστικότητα και η αρχιτεκτονική λογισμικου μπορούν να ενωθούν.

2.2.2 Developing adaptable software architectures using design patterns: an NFR approach

- Computer Standards & Interfaces 25 (2003) 253–260

Ακριβώς όπως και όλα αλλάζουν, έτσι θα πρέπει και ένα σύστημα λογισμικού για να μπορέσει να επιβιώσει και να επιτύχει. Αλλά πώς μπορούμε να φτιάξουμε ένα τέτοιο σύστημα λογισμικού ; Πρόσφατα, ένας αυξανόμενος αριθμός έχει

δείξει τεράστιο ενδιαφέρον στη χρήση προτύπων σχεδίασης προς την ανάπτυξη ενός προσαρμόσιμου συστήματος, αφού τα πρότυπα σχεδίασης αντιπροσωπεύουν υψηλού επιπέδου αφαιρέσεις (abstraction) που αντανακλούν στην εμπειρία ειδικευμένων επαγγελματιών. Τα σχεδιαστικά πρότυπα περιγράφουν προβλήματα , λύσεις και συνέπειες του να παίρνεις συγκεκριμένες λύσεις σχεδιασμού. Η εργασία αυτή παρουσιάζει , τον Proteus, ένα πλαίσιο (framework) το οποίο προορίζεται για την υποστήριξη της ανάπτυξης προσαρμόσιμων αρχιτεκτονικών λογισμικού χρησιμοποιώντας σχεδιαστικά πρότυπα.

Οι κύριες ιδέες του Proteus απεικονίζονται με τη μορφή μιας συσκευής ελέγχου του συστήματος στο σπίτι.

Η βελτίωση της προσαρμοστικότητας του λογισμικού είναι ένα τόσο τεράστιο έργο όσο και η προσαρμογή του λογισμικού είναι αναπόφευκτη στην πραγματικότητα. Πρόσφατα έχει κερδίσει το ενδιαφέρον των ακαδημιών αλλα και των βιομηχανιών , ότι τα πρότυπα σχεδίασης προωθούν την προσαρμοστικότητα. Στην εργασία αυτήν παρουσιάσαμε τον Proteus ο οποίος έχει ως στόχο να βοηθήσει τους αρχιτέκτονες λογισμικού να αναπτύξουν ένα προσαρμόσιμο αρχιτεκτονικό λογισμικό χρησιμοποιώντας τα πρότυπα σχεδίασης. Ειδικότερα , αυτή η εργασία έχει παρουσιάσει πως να αναλύεις και να χρησιμοποιείς τα πρότυπα σχεδίασης ως εν δυνάμει ενισχυτές προσαρμοστικότητας στην ανάπτυξη συστημάτων λογισμικού.

2.2.3 Quality-driven software re-engineering

- The Journal of Systems and Software 66 (2003) 225–239

Η αναδιοργάνωση λογισμικού αποτελείται απο ένα σύνολο δραστηριοτήτων που προορίζονται για την αναδιάρθρωση ενός κληρονομικού συστήματος σε ένα νέο σύστημα συμμορφώνεται σε μαλακούς και σκληρούς περιορισμούς

ποιότητας . Η εργασία αυτή παρουσιάζει ένα πλαίσιο (framework) που επιτρέπει σε συγκεκριμένα NFR όπως η απόδοση και η συντηρισημότητα να καθοδηγήσει την αναδιοργάνωση της διαδικασίας . Τέτοιες απαιτήσεις για το νέο σύστημα έχουν μοντελοποιηθεί χρησιμοποιώντας ανεξάρτητα γραφήματα και συνδέονται με συγκεκριμένες μετατροπές λογιγσμικού. Τέλος , μια διαδικασία αξιολόγησης σε κάθε βήμα μετασχηματισμού καθορίζει εαν συγκεκριμένες ποιότητες για το νέο σύστημα μπορούν να επιτευχθούν.

Σε αυτήν την εργασία , παρουσιάσαμε ένα πλαίσιο (framework) με βάση την ποιότητα για ανασχεδιασμό λογισμικού σε αρχιτεκτονικό επίπεδο. Το πλαίσιο αυτό χρησιμοποιεί επιθυμητές ιδιότητες για τον ανασχεδιασμό κώδικα για να καθορίσει και να καθοδηγήσει την διαδικασία ανασχεδιασμού. Το πλαίσιο προσφέρει έναν παγκο εργασίας όπυ οι δραστηριότητες ανασχεδιασμού δεν συμβαίνουν στο κενό, αλλά μπορούν να αξιολογηθούν κα να τελειοποιηθούν πορκειμένου να φτάσουν συγκεκριμένες απαιτήσεις ποιότητας για το σύστημα-στόχο.

Συγκεκριμένα , το προτεινόμενο πλαίσιο αντιμετωπίζει ζητήματα σχετικά με :

1. Ο σχεδιασμός και η ανάπτυξη μιας συλλογής γραφημάτων αλληλεξάρτησης που να αφορούν χαρακτηριστικά λογισμικού, όπως η απόδοση και η συντήρηση των μεγάλων κληρονομικών συστημάτων , σε αρχιτεκτονικό επίπεδο.

2. Η ανάπτυξη ενός ολοκληρωμένου καταλόγου μετασχηματισμών όπως αυτά έχουν μοντελοποιηθεί στα γραφήματα αλληλεξάρτησης και μπορούν να εφαρμόζονται στο αρχιτεκτονικό επίπεδο του κληρονομικού συστήματος για την επίτευξη συγκεκριμένων στόχων ανασχεδιασμού.

3. Ο σχεδιασμός και η ανάπτυξη, μιας αναλυτικής μεθοδολογίας που επιτρέπει την ταυτοποίηση λαθών αρχιτεκτονικού σχεδίου και προγραμματιστικά πρότυπα κώδικα που μπορούν να ¨επισκευαστούν ¨ σύμφωνα με τα βήματα μετασχηματισμού που ταυτοποιούνται απο τους γράφους.

4. Ο σχεδιασμός και η ανάπτυξη ενός προτότυπου συστήματος που βοηθά την διαδικασία ανασχεδιασμού και αναφέρεται σε βελτιώσεις της απόδοσης και της συντηρησιμότητας του συστήματος.

Επι του παρόντος, εργαζόμαστε σε μια αναβάθμιση του πλαισίου που θα μας επιτρέπει την εκτίμηση των επιπτώσεων ενός μετασχηματισμού στη συντηρισιμότητα και την απόδοση ενός λογισμικού συστήματος. Στόχος μας είναι επίσης η δημιουργία μιας αλγοριθμικής διαδικασίας που μπορεί να χρησιμοποιηθεί για την αυτοματοποίηση της επιλογής και εφαρμογής μετασχηματισμών που δίνονται απο ένα συγκεκριμένο σενάριο ανασχεδιασμού.

2.2.4 Measuring software evolution at a nuclear fusion experiment site: a test case for the applicability of OO and reuse metrics in software characterization

- Information and Software Technology 44 (2002) 593–600

Ένα σύνολο μετρικών λογισμικού έχει χρησιμοποιηθεί για να παρέχει εμπειρικά στοιχεία σχετικά με το πως η οργάνωση του κώδικα αλλάζει όταν ένα προιόν λογισμικού εξελίσσεται. Μία java εφαρμογή για την εμφάνιση γραφικών στοιχείων που χρησιμοποιήθηκε στην πειραματική φυσική , έχει χρησιμοποιηθεί σαν περίπτωση δοκιμής. Αξιοποιώντας κοινά πρότυπα είναι επιθυμητή η εξέλιξη εφαρμογών λογισμικού , δεδομένου οτι θα δώσει στους σχεδιαστές και διαχειριστές μια καλύτερη κατανόηση της διαδικασίας λογισμικού. Η ανάλυση στην οποία η επαναχρησιμοποίηση πλαισίου έχει επίσης λυφθεί υπόψιν, τόνισε μια περιορισμένη χρήση μηχανισμού κληρονομιάς και, παρά την αύξηση στην συνολική πολυπλοκότητα , μια σημαντική αμεταβλητότητα της εσωτερικής οργάνωσης της εφαρμογής. Αυτό το γεγονός δικαιολογείται απο την αυξανόμενη χρήση πλαισίου κατα τη διάρκεια ζωής του προιόντος.

Αυτή η εργασία παρουσίασε μια έρευνα πάνω στην αποτελεσματικότητα ορισμένων μερτικών ΟΟ για την κατανόηση της διαδικασίας της εξέλιξης

λογισμικού. Για τον σκοπό αυτόν , έχει παρουσιαστεί ένα σύνολο απο μετρικές λογισμικού που έχουν παρθεί απο τρία διαφορετικά στάδια της ζωής μιας java εφαρμογής με σκοπό να δώσει εμεπιρικές αποδείξεις στο πως η οργάνωση του κώδικα αλλάζει όταν ένα προιόν λογισμικού εξελίσεται. Αξίζει να σημειωθεί οτι τα αποτελέσματα που προέρχονται απο συλλογή μετρικών λογισμικού δεν παρέχουν απο μόνες τους μία ταξινόμηση της οργάνωσης λογισμικού. Με άλλα λόγια , προγράμματα με μια εντελώς διαφορετική οργάνωση θα μπορούσαν να έχουν την ίδια τιμή σε κάποια συγκεκριμένη μετρική. Τα αποτελέσματα των μετρήσεων λογισμικού σε κάθε περίπτωση παρέχουν μια πληροφορία η οποία , η οποία όταν συνδυάζεται με άλλες μετρήσεις , μπορεί να προτείνει στοιχεία για διάφορες πτυχές της οργάνωσης λογισμικού.

Τόσο η επαναχρησιμοποίηση οσο και C & K μετρικές δίνουν αποδείξεις για το γεγονός οτι μια μικρή αναδιάρθρωση τάξης έγινε κατα τη διάρκεια ζωής του προιόντος. Με άλλα λόγια , η αρχική οργάνωση της τάξης δείχνει να έχει διατηρηθεί και η ένταξη των νέων χαρακτηριστικών έχει επιτευχθεί με την ένταξη νέων κλάσεων παρά με την αλλαγή της τρέχουσας οργάνωσης . Ενω αυτή η περίπτωση μελέτης έδωσε αποδείξεις για μικρή αναδιάρθρωση τάξης , οι C & K και οι μετρικές επαναχρησιμοποίησης έδειξαν μια εξέλιξη προς μια βαθύτερη ενοποίηση των τάξεων μλεσα στην εφαρμογή.

Αξίζει να αναφέρουμε πως η εργασία ξεκίνησε το 1996 , και οι προγραμματιστές απόκτησαν περαιτέρω εμπειρία μετά την πρώτη έκδοση του εργαλείου. Αυτό το γεγονός είναι μάλλον κοινό σε πολλές εφαρμογές java που αναπτύχθηκαν την ίδια περίοδο, αλλά μπορεί να αλλάξει για μερικά έργα λογισμικού που οι προγραμματιστές τους έχουν ήδη μια πιο ώριμη εμπειρία.

Σε κάθε περίπτωση , η πλήρης κατανόηση όλων των πτυχών ενός java πλαισίου απαιτεί χρόνια εμπειρίας και μπορούμε επομένως να περιμένουμε οτι ακόμα και μελλοντικές εργασίες σε java θα παρουσιάσουν μια αυξανόμενη ολοκλήρωση του πλαισίου κατα τη διάρκεια ζωής του.

2.2.5 Frameworks: Putting Design Patterns into Perspective

- Henrik Bærbak Christensen Department of Computer Science University of Aarhus

Τα πρότυπα σχεδίασης έκαναν μια ισχυρή επίδραση στον τρόπο με τον οποίο τα αντικειμενοστραφεί λογισμικά σχεδιάζονται, υλοποιούνται και κοινοποιούνται σε βιομηχανικά έργα. Τα πρότυπα έχουν ένα φυσικό κατάλογο που μπορεί εύκολα να μας παραπλανήσει στο να στο να διαλέξεις ένα τη φορά. Αυτό αφήνει την εντύπωση των προτύπων ως μεμονωμένες λύσεις σε ανεξάρτητα προβλήματα. Σας παρουσιάζουμε την εμπειρία μας με την προσπάθεια για την αντιμετώπιση αυτού προβλήματος με τη χρήση ενός καλού μηχανικά πλαισίου (framework) , JhotDraw, σαν περίπτωση μελέτης για το πως τα πρότυπα δουλεύουν μαζί για να καθορίσουν ένα ευέλικτο και σύνθετο λογισμικό σύστημα υψηλής ποιότητας.

Σε αυτήν την εργασία έχουμε υποστηρίξει οτι το να διδάξεις πρότυπα σχεδίασης παρουσιάζει ορισμένες παγίδες που μπορεί να οδηγήσουν μαθητές απο το να μην δουν τις πλήρεις δυνατότητές τους. Άποψη μας είναι ότι τα πρότυπα είναι περιγραφές των ρόλων των συνεργαζομένων αντικειμένων, αυτό μπορεί να μην γίνει ξεκάθαρο στους μαθητές αν δεν δείξουμε αυτό το επιχείρημα στην πράξη.

Για να τη δείξουμε πρακτικά αυτή την άποψη αποφασίσαμε να χρησιμοποιήσουμε σαν μία καλή περίπτωση μελέτης , το πλαίσιο JhotDraw. Το JhotDraw στηρίζεται σε πρότυπα τα οποία είναι προφανείς επιλογές στη διδασκαλία :

- observer,
- composite
- adapter

- model-view-controller.

Τα πλαίσια είναι παραδείγματα της επαναχρησιμοποίησης του σχεδίου αλλα και του κώδικα , ενώ τα πρότυπα σχεδίασης είναι μόνο του σχεδίου. Έτσι ενισχύεται η άποψη του προγραμματισμού σαν μια διαδικασία επαναχρησιμοποίησης κώδικοποίησης , και όχι μόνο σαν κωδικοποίηση. Τα πλαίσια και τα οφέλη της χρήσης τους έχουν ήδη φανεί στο βιομηχανικό λογισμικό απο άποψη κόστους, αποτελεσματικότητας και αξιοπιστίας. Τώρα περιμένουμε να δούμε και κάτι ανάλογο και στη διδασκαλία.

2.3 Κίνδυνοι Εγκυρότητας

Στο κεφάλαιο αυτό συζητάμε τους πιθανούς κινδύνους εγκυρότητας της έρευνάς μας. Όσον αφορά τη διαδικασία αναζήτησης, οποιαδήποτε μελέτη δεν ανέφερε τη λέξη «πρότυπο» στον τίτλο του άρθρου της, 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 αν 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.

3. Εμπειρική Μεθοδολογία

3.1 Μεθοδολογία

Σ αυτό το κεφάλαιο θα παρουσιάσουμε ολη τη μεθοδολογία με την οποία δουλέψαμε, δηλαδή τα ερωτήματα της έρευνας, τη διαδικασία που ακολουθήθηκε και τις μεθόδους ανάλυσης των δεδομένων.

3.2 Τα Ερωτήματα

Σε αυτήν την ενότητα θέτουμε τα ερωτήματα της έρευνας

Ερώτημα 1) Ποιά πρότυπα σχεδίασης χρησιμοποιούνται συχνότερα στις βιβλιοθήκες ;

Ερώτημα 2) Υπάρχει μεγάλη διαφορά απο στατιστικής άποψης μεταξύ των κατηγοριών λογισμικού API και Standalone ;

3.3 Πλάνο Της Μελέτης

Για να δημιουργήσουμε μια σωστή μεθοδολογία για μια εμπειρική μέθοδο επιβεβαίωσης , φτιάξαμε το παρακάτω πλάνο μελέτης που αποτελείται απο πέντε βήματα :

1. Επιλογή των κατηγοριών του λογισμικού ανοικτού κώδικα που θα μελετήσουμε
2. Εύρεση των κατάλληλων εφαρμογών απο κάθε κατηγορία
3. Ανίχνευση των προτύπων σχεδίασης σε κάθε εφαρμογή

4. Σύνοψη των δεδομένων
5. Ανάλυση των δεδομένων όσον αφορα τα ερωτήματα της έρευνας

Στη συνέχεια ανατρέξαμε στη σελίδα http://sourceforge.net/ όπου και συλλέξαμε τις εφαρμογές μας. Οι εφαρμογές είναι επιλεγμένες απο τις παρακάτω κατηγορίες :

1. Communications
2. Graphics
3. Audio & Video
4. Business & Enterprise

Επιλέξαμε τις εφαρμογές μας με βάση τα παρακάτω κριτήρια :

1. Είναι γραμμένα σε java, σύμφωνα με τους περιορισμούς του εργαλείου που χρησιμοποιούμε για την ανίχνευση των προτύπων.
2. Διαθέτουν δυαδικό κώδικα, σύμφωνα με τους περιορισμούς του εργαλείου που χρησιμοποιούμε για την ανίχνευση των προτύπων.

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. 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.

3.4 Ανάλυση Των 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».

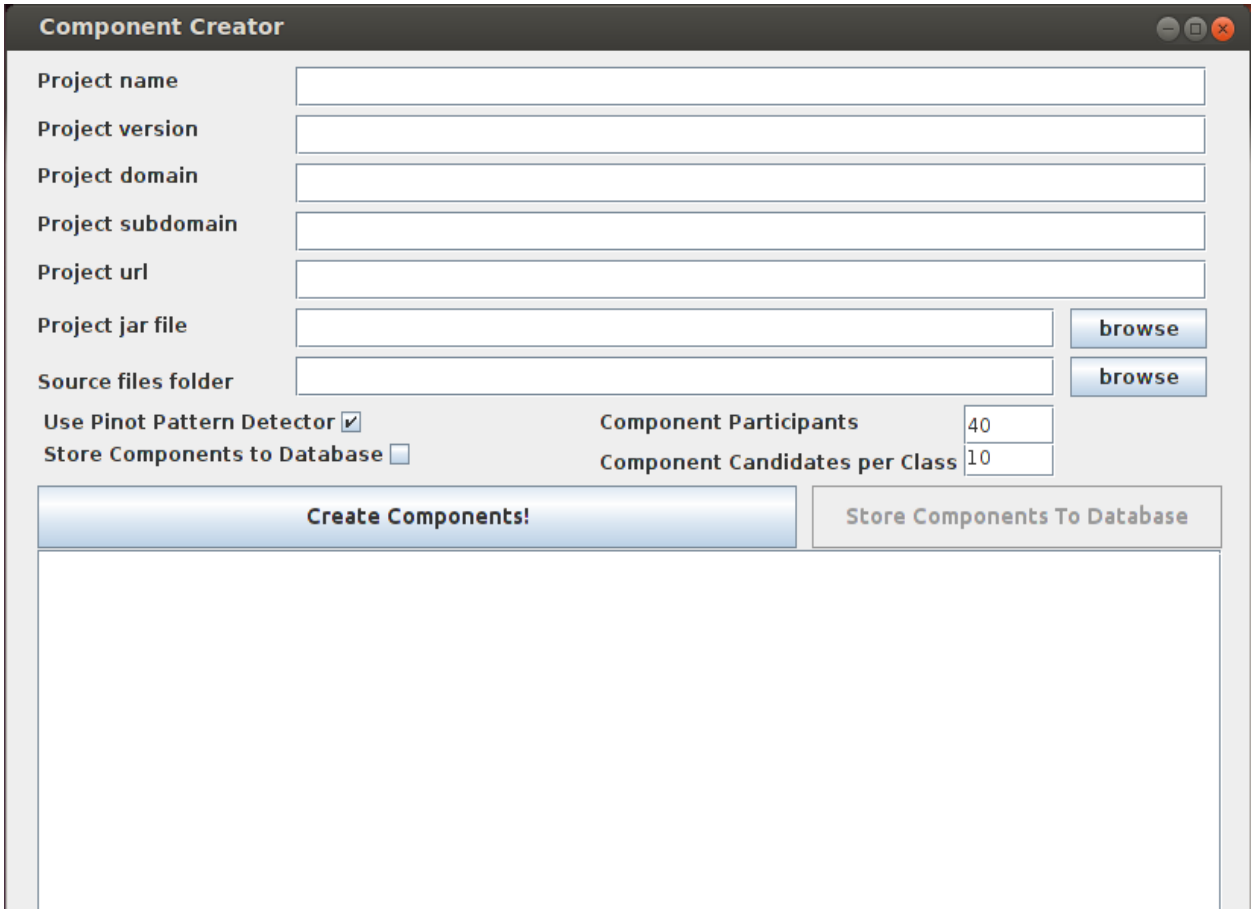
Προκειμένου τα δεδομένα του προγράμματος να συμβαδίζουν με την μελέτη που αναλύουμε, έγιναν από τον προγραμματιστή, οι απαραίτητες τροποποιήσεις σε αυτό. Οι αλλαγές αφορούσαν κυρίως τις θεματικές ενότητες που χρησιμοποιήθηκαν αλλά και στις προδιαγραφές του προγράμματος (κυρίως την μνήμη) ώστε να είναι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

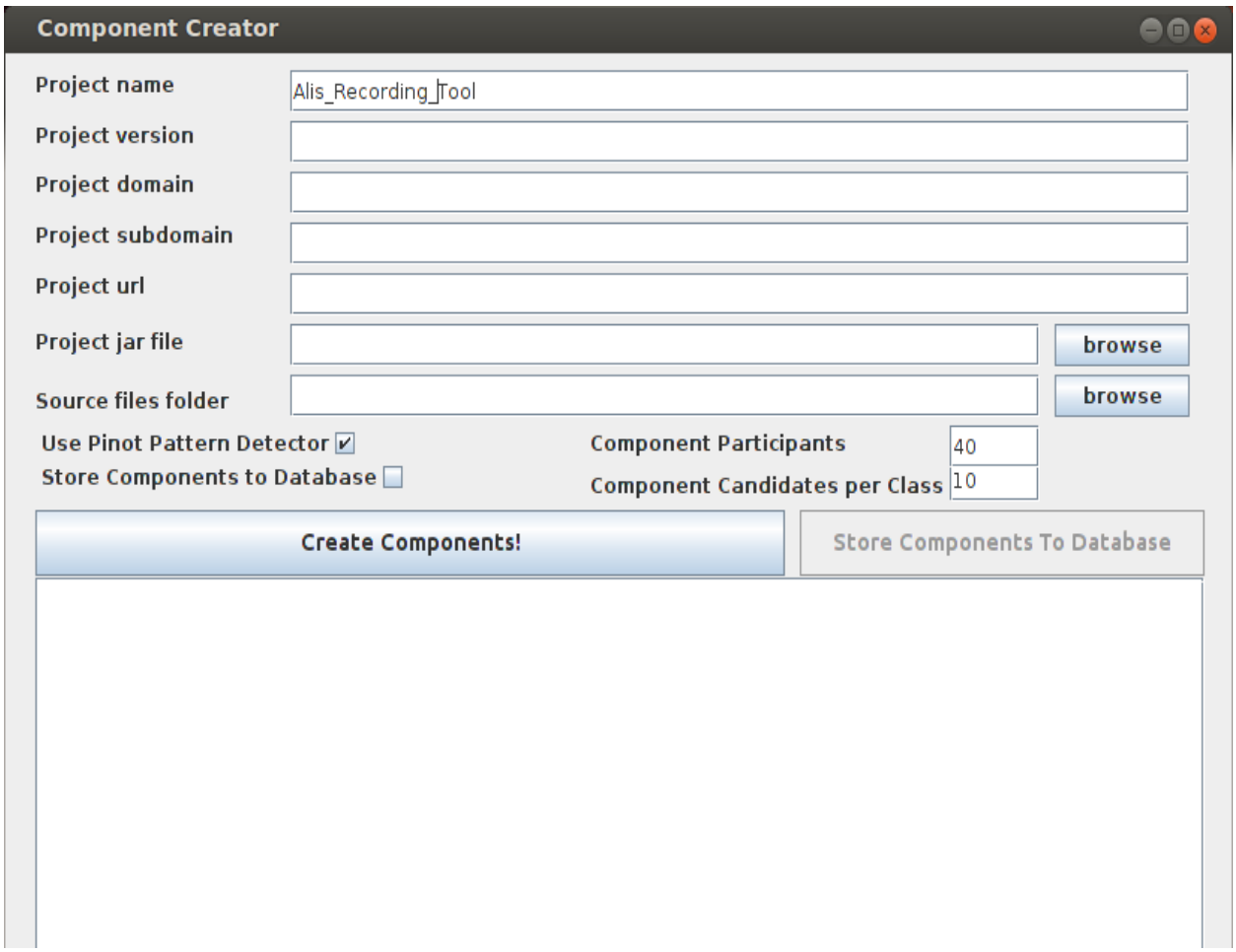
αξιολόγηση σε τρία προγράμματα ανοιχτού λογισμικού κάνει εμφανή την ακρίβεια και την αποδοτικότητα της προτεινόμενης μεθόδου.

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.

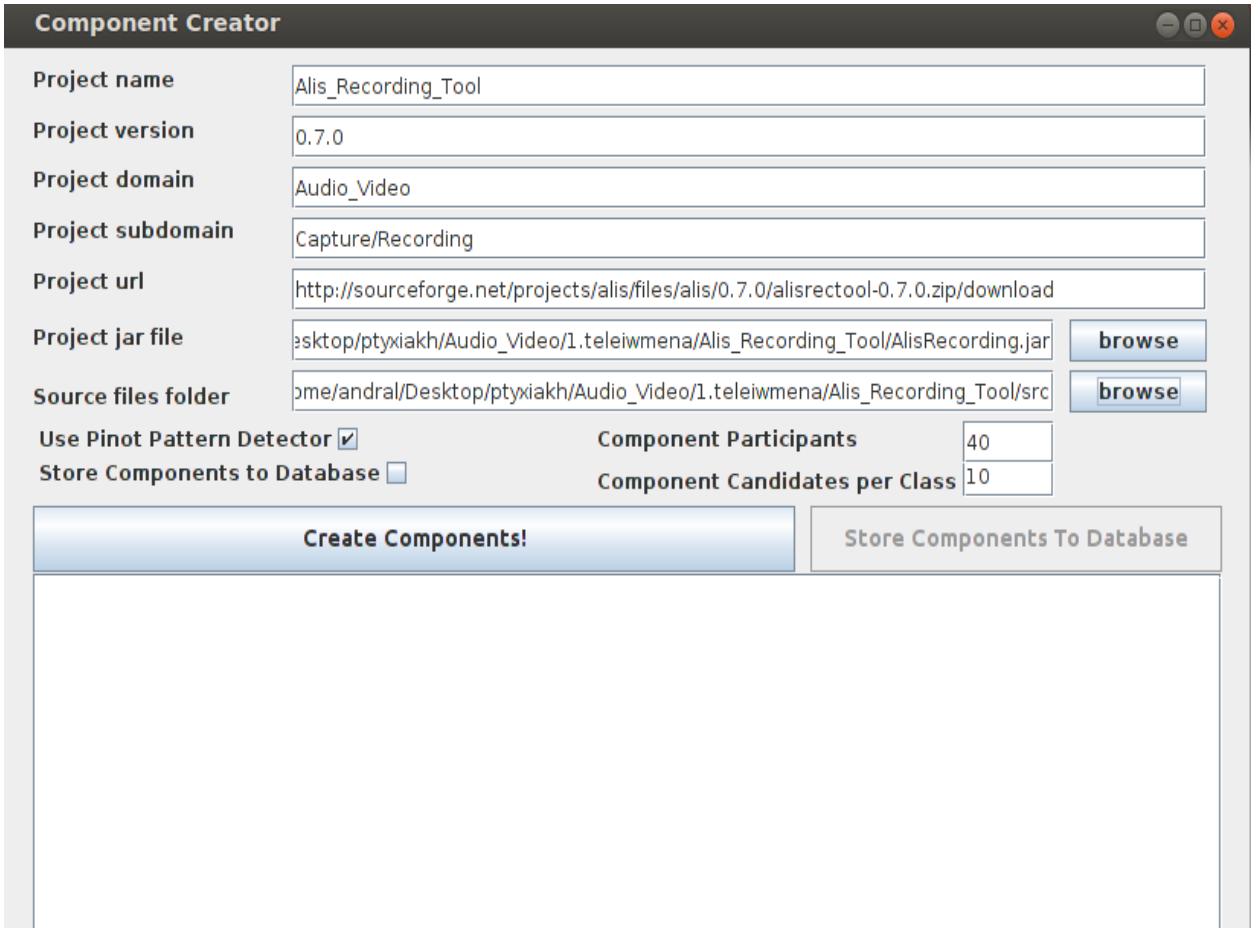
Σε αυτή τη φάση θα παρουσιάσουμε μερικά στιγμιότυπα της εκτέλεσης μίας εφαρμογής απο το εργαλείο. Ενδεικτικά επιλέξαμε την εφαρμογή «Alis Recording Tool» .



Στιγμιότυπο κατα την εκκίνηση του εργαλείου



Στιγμιότυπο κατα την επιλογή του project



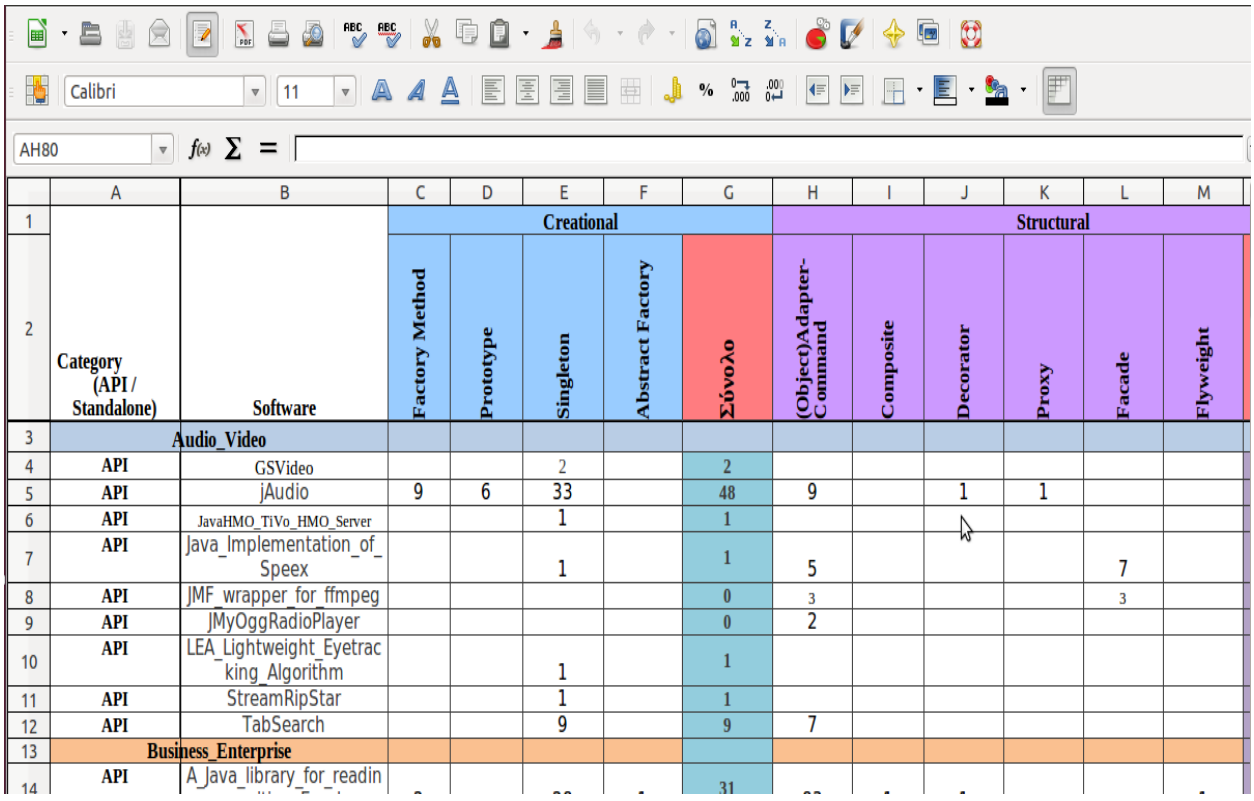
Στιγμιότυπο κατα την τελική φάση εκτέλεσης του project

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.

Στη συνέχεια συγκεντρώσαμε σε ένα excel για κάθε μία εφαρμογή όλα τα πρότυπα και τον αριθμό εμφάνισής τους, ακριβώς όπως φαίνεται και παρακάτω :

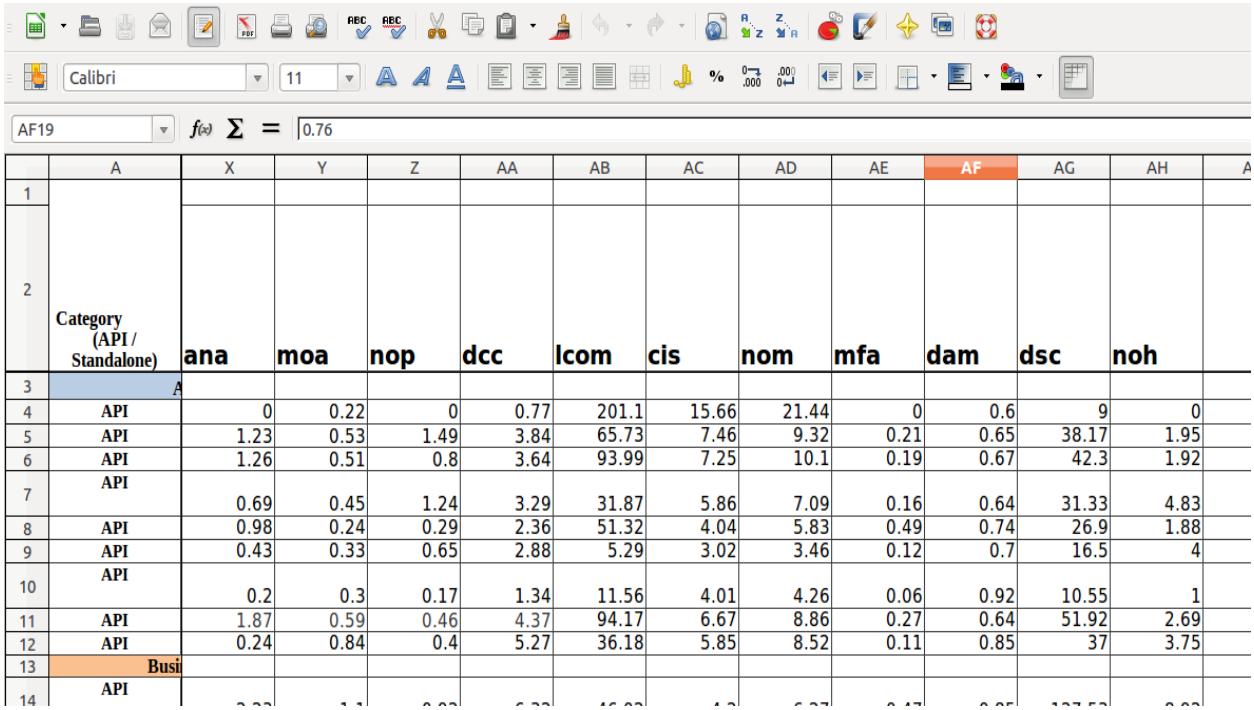


	A	B	C	D	E	F	G	H	I	J	K	L	M
1			Creational					Structural					
2	Category (API / Standalone)	Software	Factory Method	Prototype	Singleton	Abstract Factory	Σύνολο	(Object)Adapter-Command	Composite	Decorator	Proxy	Facade	Flyweight
3	Audio_Video												
4	API	GSVideo			2		2						
5	API	jAudio	9	6	33		48	9		1	1		
6	API	JavaHMO_TiVo_HMO_Server			1		1						
7	API	Java_Implementation_of_Speex			1		1	5				7	
8	API	JMF_wrapper_for_ffmpeg					0	3				3	
9	API	JMyOggRadioPlayer					0	2					
10	API	LEA_Lightweight_Eyetracking_Algorithm			1		1						
11	API	StreamRipStar			1		1						
12	API	TabSearch			9		9	7					
13	Business_Enterprise												
14	API	A_Java_library_for_readin					31						

Αμέσως μετά υπολογίσαμε τις εξής μετρικές για κάθε μία απο τις εφαρμογές :

1. Ana
2. Moa
3. Nop
4. Dcc
5. Lcom
6. Cis
7. Nom
8. Mfa
9. Dam
10. Dsc
11. Noh

Και ενημερώσαμε το excel με όλες τις νέες πληροφορίες. Το επόμενο στιγμιότυπο οθόνης αποτελεί συνέχεια του προηγούμενου για τις ίδιες γραμμές μετα νέα δεδομένα μετά την ενημέρωση.



	A	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH
1												
2	Category (API / Standalone)	ana	moa	nop	dcc	lcom	cis	nom	mfa	dam	dsc	noh
3	A											
4	API	0	0.22	0	0.77	201.1	15.66	21.44	0	0.6	9	0
5	API	1.23	0.53	1.49	3.84	65.73	7.46	9.32	0.21	0.65	38.17	1.95
6	API	1.26	0.51	0.8	3.64	93.99	7.25	10.1	0.19	0.67	42.3	1.92
7	API	0.69	0.45	1.24	3.29	31.87	5.86	7.09	0.16	0.64	31.33	4.83
8	API	0.98	0.24	0.29	2.36	51.32	4.04	5.83	0.49	0.74	26.9	1.88
9	API	0.43	0.33	0.65	2.88	5.29	3.02	3.46	0.12	0.7	16.5	4
10	API	0.2	0.3	0.17	1.34	11.56	4.01	4.26	0.06	0.92	10.55	1
11	API	1.87	0.59	0.46	4.37	94.17	6.67	8.86	0.27	0.64	51.92	2.69
12	API	0.24	0.84	0.4	5.27	36.18	5.85	8.52	0.11	0.85	37	3.75
13	Busi											
14	API											

Κατά τη τελευταία φάσης της ανάλυσης στην έρευνά μας, χρησιμοποιήσαμε το πρόγραμμα SPSS 20 προκειμένου να βγάλουμε τα στατιστικά αποτελέσματα που μας ενδιέφεραν. Οι τεχνικές που χρησιμοποιήθηκαν είναι:

- Descriptive statistics
- Pearson correlation

Τα αποτελέσματα που πήραμε απο την προιγούμενη διαδικασία θα αναλυθούν λεπτομερώς στο επόμενο κεφάλαιο της εργασίας μας.

4. Αποτελέσματα

Πίνακας 8 .API - Creational

		Factory Method	Prototype	Singleton	Abstract Factory
ANA	Pearson Sig N	-0.184 0.566 12	-0.491 0.673 3	-0.009 0.956 37	0.003 0.995 7
MOA	Pearson Sig N	0.229 0.474 12	-0.975 0.142 3	-0.169 0.318 37	0.583 0.169 7
NOP	Pearson Sig N	0.008 0.979 12	0.962 0.176 3	0.256 0.144 34	-0.184 0.728 6
DCC	Pearson Sig N	0.204 0.526 12	-,978 ,133 3	0.050 0.777 34	0.594 0.214 6
LCOM	Pearson Sig N	-0.211 0.510 12	0.184 0.882 3	0.550 0.001 34	-0.382 0.455 6
CIS	Pearson	-0.240	0.547	0.251	-0.263

	Sig	0.453	0.631	0.153	0.615
	N	12	3	34	6
NOM	Pearson	-0.310	0.065	0.200	-0.390
	Sig	0.327	0.959	0.257	0.445
	N	12	3	34	6
MFA	Pearson	-0.260	-0.259	-0.110	-0.312
	Sig	0.415	0.833	0.535	0.547
	N	12	3	34	6
DAM	Pearson	-0.220	-0.975	0.194	0.363
	Sig	0.492	143	0.270	0.480
	N	12	3	34	6
DSC	Pearson	-0.316	-0.992	0.174	-0.206
	Sig	0.318	0.083	0.325	0.696
	N	12	3	34	6
NOH	Pearson	-0.035	-0.454	0.256	0.007
	Sig	0.915	0.700	0.144	0.990
	N	12	3	34	6
Reusability	Pearson	-0.225	-0.650	0.538	-0.326
	Sig	0.483	0.549	0.001	0.528
	N	12	3	34	6

Flexibility	Pearson Sig N	-0.040 0.901 12	0.929 0.242 3	-0.003 0.986 34	-0.372 0.468 6
Understandability	Pearson Sig N	-0.203 526 12	-0.699 0.508 3	0.554 0.001 34	-0.436 0.388 6
Functionality	Pearson Sig N	-0.223 0.485 12	-0.647 0.552 3	0.540 0.001 34	-0.322 0.533 6
Extendibility	Pearson Sig N	-0.166 0.606 12	1.000 0.002 3	-0.017 0.925 34	-0.652 0.161 6
Effectiveness	Pearson Sig N	-0.504 0.868 12	-0.193 0.877 3	-0.006 0.975 34	-0.139 0.793 6

Πίνακας 9. API - Structural

		(Object)Adapter-Command	Composite	Decorator	Proxy	Facade	Flyweight
ANA	Pearson	0.122	0.316	-0.469	-0.708	0.092	-0.936
	Sig	0.477	0.795	0.689	0.115	0.734	0.064
	N	36	3	3	6	16	4
MOA	Pearson	-0.125	0.727	0.975	-0.295	-0.142	-0.472
	Sig	0.468	0.482	0.142	0.571	0.600	0.528
	N	36	3	3	6	16	4
NOP	Pearson	0.045	-1.000	-0.333	0.743	-0.026	0.951
	Sig	0.798	-	0.784	0.091	0.928	0.049
	N	35	2	3	6	14	4
DCC	Pearson	0.199	1.000	0.958	-0.628	0.403	-0.723
	Sig	0.252	-	0.186	0.182	0.153	277
	N	35	2	3	6	14	4
LCOM	Pearson	0.003	1.000	-0.866	-0.260	0.384	0.967
	Sig	0.986	-	0.333	0.619	0.175	0.033
	N	35	2	3	6	14	4
CIS	Pearson	-0.018	1.000	0.267	-0.344	0.324	0.279

	Sig N	0.919 35	- 2	0.828 3	0.505 6	0.259 14	0.721 4
NOM	Pearson Sig N	-0.051 0.771 35	1.000 - 2	0.251 0.839 3	-0.404 0.427 6	0.377 0.184 14	0.205 0.795 4
MFA	Pearson Sig N	0.081 0.642 35	-1.000 - 2	-0.339 0.780 3	-0.625 0.184 6	0.353 0.216 14	-0.949 0.051 4
DAM	Pearson Sig N	-0.036 0.837 35	1.000 - 2	-0.984 0.113 3	0.418 0.409 6	0.249 0.390 14	-0.030 0.970 4
DSC	Pearson Sig N	-0.012 0.944 35	-1.000 - 2	0.998 0.037 3	-0.472 0.345 6	0.708 0.005 14	-0.641 0.359 4
NOH	Pearson Sig N	0.056 0.749 35	-1.000 - 2	0.767 0.443 3	-0.239 0649 6	0.721 0.004 14	0.648 0.352 4
Reusability	Pearson Sig N	0.001 0.996 35	-1.000 - 2	0.969 0.160 3	-0.360 0.483 6	0.555 0.039 14	0.966 0.034 4

Flexibility	Pearson Sig N	-0.203 0.243 35	-1.000 - 2	-0.420 0.724 3	0.768 0.074 6	-0.328 0.252 14	0.958 0.042 4
Understandability	Pearson Sig N	0.004 0.983 35	1.000 - 2	-0.998 0.037 3	-0.068 0.898 6	0.170 0.561 14	0.911 0.089 4
Functionality	Pearson Sig N	0.002 0.991 35	-1.000 - 2	0.952 0.199 3	-0.356 0.489 6	0.558 0.038 14	0.925 0.075 4
Extendibility	Pearson Sig N	-0.113 0.519 35	-1.000 - 2	-0.984 0.113 3	0.651 0.162 6	-0.329 0.251 14	0.804 0.196 4
Effectiveness	Pearson Sig N	0.042 0.812 35	-1.000 - 2	-1.000 0.014 3	0.553 0.255 6	0.080 0.784 14	0.554 0.446 4

Πίνακας 10. API - Behavioral

		Chain of Responsibility	**Observer**	**Mediator**	**Template Metho**	**Strategy**	**Visitor**

					d		
ANA	Pearson	-	-	0.040	0.056	0.273	0.979
	Sig	-	-	0.908	0.793	0.307	0.021
	N	2	1	11	24	16	4
MOA	Pearson	-	-	0.013	-0.001	0.007	0.794
	Sig	-	-	0.971	0.995	0.979	0.206
	N	2	1	11	24	16	4
NOP	Pearson	-	-	-0.197	0.154	-0.198	0.726
	Sig	-	-	0.612	0.482	0.478	0.274
	N	2	0	9	23	15	4
DCC	Pearson	-	-	0.682	0.032	0.298	-0.467
	Sig	-	-	0.043	0.885	0.280	0.533
	N	2	0	9	23	15	4
LCOM	Pearson	-	-	0.011	0.063	-0.268	-0.391
	Sig	-	-	0.978	0.776	0.334	0.609
	N	2	0	9	23	15	4
CIS	Pearson	-	-	-0.022	-0.025	-0.294	-0.598
	Sig	-	-	0.955	0.911	0.287	0.402
	N	2	0	9	23	15	4
NOM	Pearson	-	-	0.036	-0.080	-0.276	-0.448

	Sig N	- 2	- 0	0.927 9	0.717 23	0.319 15	0.552 4
MFA	Pearson Sig N	- - 2	- - 0	0.047 0.905 9	-0.068 0.758 23	0.218 0.436 15	0.635 0.365 4
DAM	Pearson Sig N	- - 2	- - 0	0.398 0.289 9	-0.197 0.367 23	0.078 0.783 15	-0.962 0.038 4
DSC	Pearson Sig N	- - 2	- - 0	0.411 0.272 9	-0.064 0.771 23	-0.041 0.885 15	-0.186 0.814 4
NOH	Pearson Sig N	- - 2	- - 0	0.565 0.113 9	0.157 0.474 23	0.001 0.996 15	-0.035 0.965 4
Reusability	Pearson Sig N	- - 2	- - 0	0.174 0.655 9	0.052 0.815 23	-0.200 0.475 15	-0.310 0.690 4
Flexibility	Pearson Sig N	- - 2	- - 0	-0.569 0.110 9	0.090 0.682 23	-0.346 0.206 15	0.959 0.041 4

Understandability	Pearson Sig N	- - 2	- - 0	-0.167 0.668 9	0.069 0.756 23	-0.322 0.241 15	-0.514 0.486 4
Functionality	Pearson Sig N	- - 2	- - 0	0.178 0.646 9	0.054 0.807 23	-0.201 0.474 15	-0.308 0.692 4
Extendibility	Pearson Sig N	- - 2	- - 0	-0.730 0.026 9	0.068 0.757 23	-0.291 0.292 15	0.913 0.087 4
Effectiveness	Pearson Sig N	- - 2	- - 0	-0.094 0.806 9	0.083 0.706 23	-0.031 0.912 15	0.955 0.045 4

Πίνακας 11. Standalone - Creational

		Factory Method	Prototype	Singleton	Abstract Factory
ANA	Pearson Sig N	-0.256 0.202 26	0.095 0.757 13	-0.019 0.863 81	-0.481 0.412 5
MOA	Pearson Sig N	-0.076 0.711 26	-0.333 0.266 13	0.140 0.211 81	0.456 0.440 5
NOP	Pearson Sig N	-0.270 0.182 26	0.513 0.088 12	0.360 0.001 78	0.020 0.974 5
DCC	Pearson Sig N	-0.172 0.402 26	-0.254 0.425 12	-0.121 0.291 78	0.593 0.291 5
LCOM	Pearson Sig N	-0.175 0.392 26	0.000 1.000 12	0.241 0.034 78	-0.115 0.854 5
CIS	Pearson Sig	-0.221 0.277	0.170 0.597	0.325 0.004	0.234 0.705

	N	26	12	78	5
NOM	Pearson	-0.273	0.190	0.268	0.369
	Sig	0.178	0.555	0.018	0.541
	N	26	12	78	5
MFA	Pearson	-0.258	-0.028	-0.053	0.305
	Sig	0.204	0.930	0.648	0.618
	N	26	12	78	5
DAM	Pearson	-0.051	0.378	-0.211	-0.541
	Sig	0.804	0.225	0.064	0.347
	N	26	12	78	5
DSC	Pearson	0.502	-0.181	-0.039	0.362
	Sig	0.009	0.574	0.735	0.549
	N	26	12	78	5
NOH	Pearson	-0.241	-0.029	-0.139	-0.055
	Sig	0.236	0.928	0.225	0.930
	N	26	12	78	5
Reusability	Pearson	-0.133	-0.102	0.255	-0.091
	Sig	0.518	0.751	0.047	0.884
	N	26	12	78	5
Flexibility	Pearson	-0.128	0.370	0.374	-0.507

	Sig N	0.532 26	0.236 12	0.001 78	0.383 5
Understandability	Pearson Sig N	-0.194 0.342 26	0.190 0.554 12	0.246 0.030 78	-0.132 0.832 5
Functionality	Pearson Sig N	-0.139 0.497 26	-0.097 0.765 12	0.225 0.048 78	-0.091 0.884 5
Extendibility	Pearson Sig N	-0.099 0.630 26	0.553 0.062 12	0.290 0.010 78	-0.797 0.107 5
Effectiveness	Pearson Sig N	-0.388 0.50 26	0.358 0.254 12	0.201 0.078 78	0.004 0.995 5

Πίνακας 12.Standalone - Structural

		(Object)Adapter-Command	Composite	Decorator	Proxy	Facade	Flyweight
ANA	Pearson	-0.108	-	-0.841	-0.354	0.087	0.503
	Sig	0.335	-	0.018	0.164	0.619	0.665
	N	82	3	7	17	35	3
MOA	Pearson	0.067	-	-0.107	-0.248	0.075	-0.092
	Sig	0.547	-	0.819	0.337	0.668	0.941
	N	82	3	7	17	35	3
NOP	Pearson	0.102	-	0.002	0.047	0.409	0.543
	Sig	0373	-	0.996	0.859	0.016	0.634
	N	79	3	7	17	34	3
DCC	Pearson	-0.046	-	-0.389	-0.371	0.240	0.980
	Sig	0.690	-	0.388	0.143	0.172	0.126
	N	79	3	7	17	34	3
LCOM	Pearson	0.102	-	-0.389	-0.134	0.589	-0.662
	Sig	0.370	-	0.388	0.608	0.000	0.540
	N	79	3	7	17	34	3
CIS	Pearson	0.136	-	-0.238	0.011	0.541	0.935
	Sig	0.233	-	0.608	0.967	0.001	0.230
	N	79	3	7	17	34	3

NOM	Pearson Sig N	0.152 0.182 79	- - 3	-0.331 0.468 7	-0.012 0.964 17	0.516 0.002 34	0.248 0.840 3
MFA	Pearson Sig N	-0.094 0.409 79	- - 3	-0.572 0.180 7	-0.424 0.090 17	0.056 0.752 34	0.000 1.000 3
DAM	Pearson Sig N	-0.029 0.799 79	- - 3	-0.175 0.707 7	0.271 0.293 17	-0.050 0.777 34	0.694 0.512 3
DSC	Pearson Sig N	0.072 0.531 79	- - 3	0.690 0.086 7	-0.371 0.143 17	0.095 0.593 34	0.920 0.257 3
NOH	Pearson Sig N	-0.015 0.894 79	- - 3	-0.491 0.263 7	-0.364 0.151 17	0.030 0.866 34	0.687 0.518 3
Reusability	Pearson Sig N	0.113 0.320 79	- - 3	0.030 0.950 7	-0.161 0.537 17	0.583 0.000 34	0.408 0.733 3
Flexibility	Pearson Sig	0.144 0.205	- -	0.270 0.558	0.336 0.187	0.184 0.297	-1.000 0.019

	N	79	3	7	17	34	3
Understandability	Pearson Sig N	0.095 0.407 79	- - 3	-0.549 0.202 7	-0.117 0.655 17	0.591 0.000 34	-0.956 0.191 3
Functionality	Pearson Sig N	0.112 0.325 79	- - 3	-0.031 0.948 7	-0.162 0.533 17	0.583 0.000 34	0.470 0.688 3
Extendibility	Pearson Sig N	0.059 0.605 79	- - 3	0.035 0.941 7	0.308 0.228 17	0.040 0.821 34	-0.887 0.306 3
Effectiveness	Pearson Sig N	0.042 0.711 79	- - 3	-0.689 0.087 7	-0.209 0.420 17	0.272 0.119 34	0.641 0.557 3

Πίνακας 13. Standalone - Behavioral

		Chain of Responsibility	**Observer**	**Mediator**	**Template Method**	**Strategy**	**Visitor**
ANA	Pearson Sig N	- - 2	- - 0	0.157 0.473 23	-0.096 0.473 58	-0.150 0.446 28	- - 1

MOA	Pearson	-	-	-0.048	-0.022	0.195	-
	Sig	-	-	0.827	0.869	0.320	-
	N	2	0	23	58	28	1
NOP	Pearson	-	-	0.416	0.117	0.047	-
	Sig	-	-	0.054	0.392	0.819	-
	N	2	0	22	56	26	1
DCC	Pearson	-	-	0.166	-0.120	0.207	-
	Sig	-	-	0.461	0.377	0.311	-
	N	2	0	22	56	26	1
LCOM	Pearson	-	-	0.358	-0.065	-0.112	-
	Sig	-	-	0.102	0.636	0.586	-
	N	2	0	22	56	26	1
CIS	Pearson	-	-	0.392	0.067	0.044	-
	Sig	-	-	0.071	0.625	0.832	-
	N	2	0	22	56	26	1
NOM	Pearson	-	-	0.356	0.002	0.048	-
	Sig	-	-	0.104	0.988	0.817	-
	N	2	0	22	56	26	1
MFA	Pearson	-	-	0.373	-0.102	-0.103	-
	Sig	-	-	0.088	0.455	0.615	-

	N	2	0	22	56	26	1
DAM	Pearson	-	-	-0.160	0.063	0.117	-
	Sig	-	-	0.477	0.643	0.569	-
	N	2	0	22	56	26	1
DSC	Pearson	-	-	0.359	0.055	-0.109	-
	Sig	-	-	0.101	0.688	0.597	-
	N	2	0	22	56	26	1
NOH	Pearson	-	-	0.443	-0.024	0.103	-
	Sig	-	-	0.039	0.861	0.618	-
	N	2	0	22	56	26	1
Reusability	Pearson	-	-	0.367	-0.048	-0.128	-
	Sig	-	-	0.093	0.725	0.533	-
	N	2	0	22	56	26	1
Flexibility	Pearson	-	-	0.208	0.170	-0.067	-
	Sig	-	-	0.354	0.209	0.744	-
	N	2	0	22	56	26	1
Understandability	Pearson	-	-	0.353	-0.072	-0.103	-
	Sig	-	-	0.107	0.600	0.618	-
	N	2	0	22	56	26	1
Functionality	Pearson	-	-	0.369	-0.049	-0.125	-

	Sig N	- 2	- 0	0.091 22	0.718 56	0.543 26	- 1
Extendibility	Pearson Sig N	- - 2	- - 0	0.244 0.275 22	0.136 0.316 56	-0.365 0.067 26	- - 1
Effectiveness	Pearson Sig N	- - 2	- - 0	0.226 0.312 22	0.019 0.889 56	-0.080 0.697 26	- - 1

Πίνακας 14.Συσχετισμένα προτύπα-μετρικές

API		
LCOM- Singleton	Pearson Sig N	0.550 0.001 34
Reusability- Singleton	Pearson Sig N	0.538 0.001 34
Understandability-Singleton	Pearson Sig	0.554 0.001

	N	34
Functionality-Singleton	Pearson Sig N	0.540 0.001 34
Extendibility-Prototype	Pearson Sig N	1.000 0.002 3
DSC- Decorator	Pearson Sig N	0.998 0.037 3
Understandability-Decorator	Pearson Sig N	-0.998 0.037 3
Effectiveness-Decorator	Pearson Sig N	-1.000 0.014 3
DSC- Facade	Pearson Sig N	0.708 0.005 14
NOH- Facade	Pearson	0.721

	Sig	0.004
	N	14
Reusability- Facade	Pearson	0.555
	Sig	0.039
	N	14
NOP- Flyweight	Pearson	0.951
	Sig	0.049
	N	4
LCOM- Flyweight	Pearson	0.967
	Sig	0.033
	N	4
Reusability- Flyweight	Pearson	0.966
	Sig	0.034
	N	4
Flexibility- Flyweight	Pearson	0.958
	Sig	0.042
	N	4
DCC- Mediator	Pearson	0.682
	Sig	0.043
	N	9

Extendibility-Mediator	Pearson Sig N	-0.730 0.026 9
ANA- Visitor	Pearson Sig N	0.979 0.021 4
DAM- Visitor	Pearson Sig N	-0.962 0.038 4
Flexibility- Visitor	Pearson Sig N	0.959 0.041 4
Effectiveness- Visitor	Pearson Sig N	0.955 0.045 4
Standalone		
DSC- Factory Method	Pearson Sig N	0.502 0.009 26
NOP- Singleton	Pearson	0.360

	Sig	0.001
	N	78
LCOM- Singleton	Pearson	0.241
	Sig	0.034
	N	78
CIS- Singleton	Pearson	0.325
	Sig	0.004
	N	78
NOM- Singleton	Pearson	0.268
	Sig	0.018
	N	78
Reusability- Singleton	Pearson	0.255
	Sig	0.047
	N	78
Flexibility- Singleton	Pearson	0.374
	Sig	0.001
	N	78
Functionality- Singleton	Pearson	0.225
	Sig	0.048
	N	78

Extendibility-Singleton	Pearson Sig N	0.290 0.010 78
ANA- Decorator	Pearson Sig N	-0.841 0.018 7
LCOM- Facade	Pearson Sig N	0.589 0.000 34
CIS- Facade	Pearson Sig N	0.541 0.001 34
NOM- Facade	Pearson Sig N	0.516 0.002 34
Reusability- Facade	Pearson Sig N	0.583 0.000 34
Functionality- Facade	Pearson Sig	0.583 0.000

	N	34
NOH- Mediator	Pearson Sig N	0.443 0.039 22
Flexibility- Flyweight	Pearson Sig N	-1.000 0.019 3
Reusability- Template Method	Pearson Sig N	-0.048 0.725 5
Understandability- Singleton	Pearson Sig N	0.246 0.030 78

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Singleton** είναι συσχετισμένη σε ποσοστό 55% με την μετρική **LCOM.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Singleton** είναι συσχετισμένη σε ποσοστό 53,8 % με την μετρική **Reusability.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Singleton** είναι συσχετισμένη σε ποσοστό 54,5% με την μετρική **Understandability.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Singleton** είναι συσχετισμένη σε ποσοστό 54,5% με την μετρική **Functionality.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Prototype** είναι συσχετισμένη σε ποσοστό 100% με την μετρική **Extendibility.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Decorator** είναι συσχετισμένη σε ποσοστό 99,8% με την μετρική **DSC.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Decorator** είναι συσχετισμένη σε ποσοστό 99,8% με την μετρική **Understandability.**

Η συσχέτιση αυτή είναι αντίστροφη δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μικραίνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Decorator** είναι συσχετισμένη σε ποσοστό 99,8% με την μετρική **Effectiveness.**

Η συσχέτιση αυτή είναι αντίστροφη δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μικραίνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Facade** είναι συσχετισμένη σε ποσοστό 70,8% με την μετρική **DSC.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Facade** είναι συσχετισμένη σε ποσοστό 72,1% με την μετρική **NOH.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Facade** είναι συσχετισμένη σε ποσοστό 55,5% με την μετρική **Reusability.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Flyweight** είναι συσχετισμένη σε ποσοστό 95,1% με την μετρική **NOP.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Flyweight** είναι συσχετισμένη σε ποσοστό 96,7% με την μετρική **LCOM.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Flyweight** είναι συσχετισμένη σε ποσοστό 96,6% με την μετρική **Reusability.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Flyweight** είναι συσχετισμένη σε ποσοστό 95,8% με την μετρική **Flexibility.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Mediator** είναι συσχετισμένη σε ποσοστό 68,2% με την μετρική **DCC.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Mediator** είναι συσχετισμένη σε ποσοστό 73% με την μετρική **Extendibility.**

Η συσχέτιση αυτή είναι αντίστροφη δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μικραίνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Visitor** είναι συσχετισμένη σε ποσοστό 97,9% με την μετρική **ANA.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Visitor** είναι συσχετισμένη σε ποσοστό 96,2% με την μετρική **DAM.**

Η συσχέτιση αυτή είναι αντίστροφη δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μικραίνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Visitor** είναι συσχετισμένη σε ποσοστό 95,9% με την μετρική **Flexibility.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Visitor** είναι συσχετισμένη σε ποσοστό 95,5% με την μετρική **Effectiveness.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Factory Method** είναι συσχετισμένη σε ποσοστό 50,2% με την μετρική **DSC.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Singleton** είναι συσχετισμένη σε ποσοστό 36% με την μετρική **NOP.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Singleton** είναι συσχετισμένη σε ποσοστό 24,1% με την μετρική **LCOM.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Singleton** είναι συσχετισμένη σε ποσοστό 32,5% με την μετρική **CIS.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Singleton** είναι συσχετισμένη σε ποσοστό 26,8% με την μετρική **NOM.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Singleton** είναι συσχετισμένη σε ποσοστό 25,5% με την μετρική **Reusability.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Singleton** είναι συσχετισμένη σε ποσοστό 37,4% με την μετρική **Flexibility.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Singleton** είναι συσχετισμένη σε ποσοστό 22,5% με την μετρική **Functionality.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Singleton** είναι συσχετισμένη σε ποσοστό 29% με την μετρική **Extendibility.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Decorator** είναι συσχετισμένη σε ποσοστό 84,1% με την μετρική **ANA.**

Η συσχέτιση αυτή είναι αντίστροφη δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μικραίνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Facade** είναι συσχετισμένη σε ποσοστό 58,9% με την μετρική **LCOM.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Facade** είναι συσχετισμένη σε ποσοστό 54,1% με την μετρική **CIS.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Facade** είναι συσχετισμένη σε ποσοστό 51,6% με την μετρική **NOM.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Facade** είναι συσχετισμένη σε ποσοστό 58,3% με την μετρική **Reusability.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Facade** είναι συσχετισμένη σε ποσοστό 58,3% με την μετρική **Functionality.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Mediator** είναι συσχετισμένη σε ποσοστό 44,3% με την μετρική **NOH.**

Η συσχέτιση αυτή είναι ευθεία δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μεγαλώνει η τιμή της μετρίκης

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Flyweight** είναι συσχετισμένη σε ποσοστό 100% με την μετρική **Flexibility.**

Η συσχέτιση αυτή είναι αντίστροφη δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μικραίνει η τιμή της μετρίκης.

Από τα αποτελέσματα του πίνακα παρατηρούμε ότι η χρήση του προτύπου σχεδίασης **Template Method** είναι συσχετισμένη σε ποσοστό 4,8% με την μετρική **Reusability.**

Η συσχέτιση αυτή είναι αντίστροφη δηλαδή όσο πιο πολλά στιγμιότυπα του προτύπου σχεδίασης χρησιμοποιούνται τόσο μικραίνει η τιμή της μετρίκης.

ΑΝΑΦΟΡΕΣ

- The Journal of Systems and Software 81 (2008) 1845–1852, Reconciling usability and interactive system architecture using patterns, Ahmed Seffah *, Taleb Mohamed, Halima Habieb-Mammar, Alain Abran
- Computer Standards & Interfaces 25 (2003) 253–260, Developing adaptable software architectures using design patterns:
- an NFR approach, Lawrence Chung*, Kendra Cooper, Anna Yi
- The Journal of Systems and Software 66 (2003) 225–239, Quality-driven software re-engineering, Ladan Tahvildari a,*, Kostas Kontogiannis a,*, John Mylopoulos b
- Information and Software Technology 44 (2002) 593–600, Measuring software evolution at a nuclear fusion experiment site: a test
- case for the applicability of OO and reuse metrics in software
- characterization, G. Manduchi*, C. Taliercio
- Information and Software Technology 46 (2004) 301–307, On the composition of Java frameworks control-flows, Ana C.V. de Melo*, Bruno M. Moutinho
- Electronic Notes in Theoretical Computer Science 72 No. 4 (2003), High-level Transformations to Support
- Framework-Based Software Development, Tom Tourw´, Tom Mens
- Frameworks: Putting Design Patterns into Perspective, Henrik Bærbak Christensen, Department of Computer Science, University of Aarhus
- Patterns in Complex Systems Modeling, Janet Wiles and James Watson, ARC Centre for Complex Systems, School of Information Technology and Electrical Engineering
- The University of Queensland, Brisbane, 4072, Australia
- Towards a Semantic-Rich Collaborative Environment
- for Learning Software Patterns, Zoran Jeremić1, Jelena Jovanović1, and Dragan Gašević2, FON-School of Business Administration, University of

Belgrade, Serbia, School of Computing and Information Systems, Athabasca University, Canada

Χρήσιμα link

- http://www.computer.org/
- http://www.springerlink.com/
- http://www.sciencedirect.com/
- http://dl.acm.org/
- http://scholar.google.gr/
- http://www.sourceforge.net/
- http://en.wikipedia.org/wiki/Framework_%28computer_science%29
- http://en.wikipedia.org/wiki/Design_pattern_%28computer_science%29

ΒΙΒΛΙΟΓΡΑΦΙΑ

- Γκορτζής Αντώνιος (2012),Πτυχιακή εργασία με θέμα : «Στρατηγική επιλογής κλάσεων με στόχο την βελτιστοποίηση της ποιότητας του επιλεγμένου κώδικα».
- Χαραλαμπίδου Σοφία (2010), Πτυχιακή εργασία με θέμα: «Εμπειρική μελέτη για τη χρήση προτύπων σχεδίασης σε παιχνίδια ανοιχτού λογισμικού»,
- Chatzigeorgiou A. (2005), “Object-Oriented Design: UML, Principles, Patterns and Heuristics”, Kleidarithmos, Athens, 1st edition.