Αλεξάνδρειο Τεχνολογικό Εκπαιδευτικό Ίδρυμα
Θεσσαλονίκης
Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Πληροφορικής

Θέμα : «Αξιολόγηση της επίδρασης της χρήσης προτύπων σχεδίασης στις τιμές των μετρικών αντικειμενοστρεφούς σχεδίασης. Εφαρμογή σε εργαλείο σχεδίασης λογισμικού.»

Βούλγαρης Μιχάλης

Επίβλεψη :

Ιγνάτιος Δεληγιάννης
Αμπατζόγλου Απόστολος

Θεσσαλονίκη,
Μάρτιος 2009

ΠΡΟΛΟΓΟΣ

Η Μηχανική Λογισμικού με τη γέννηση της *αντικειμενοστρεφούς φιλοσοφίας* (object-oriented philosophy) άλλαξε ριζικά τον τρόπο σκέψης κατά την ανάλυση και σχεδίαση των συστημάτων λογισμικού, δίνοντας ένα ελπιδοφόρο μήνυμα για την αντιμετώπιση της κρίσης λογισμικού. Η συνολική αρχιτεκτονική ενός συστήματος είναι σημαντική, όχι μόνο για τη διευκόλυνση της υλοποίησης και της δοκιμής του, αλλά και για την ταχύτητα και την αποτελεσματικότητα με την οποία θα γίνει η συντήρηση και η τροποποίηση του.

Η εισαγωγή μετρήσεων στην ποιότητα λογισμικού βοήθησε στην ορθή ανάπτυξή του. Οι μετρήσεις έχουν εισαχθεί στη διαδικασία ανάπτυξης λογισμικού προκειμένου να ικανοποιηθεί η ανάγκη ελέγχου του κύκλου ζωής και η παραγωγή υψηλότερων ποιοτικά αποτελεσμάτων.

Τα Ελεύθερα Λογισμικά παρέχουν ένα ευρύ φάσμα εφαρμογών λογισμικού, ανοικτά για μελέτη και εξαγωγή συμπερασμάτων. Η ελεύθερη πρόσβαση στο πηγαίο κώδικα των προγραμμάτων είναι μια ευκαιρία μελέτης του τρόπου ανάπτυξής του, από αναλυτές, σχεδιαστές, προγραμματιστές και γενικά από όλα τα μέλη της ομάδας ανάπτυξης λογισμικού προϊόντος.

Η σχεδίαση και η διεξαγωγή της παρούσας μελέτης αποδείχτηκε ιδιαίτερα χρήσιμη για την εξοικείωση του συγγραφέα με προηγμένα θέματα τεχνολογίας λογισμικού, καθώς και με τους τρόπους εμπειρικής αξιολόγησης εργαλείων και μεθόδων τεχνολογίας λογισμικού. Επιπλέον, τα αποτελέσματα της εργασίας θεωρούνται σημαντικά και ευρύτερου επιστημονικού ενδιαφέροντος από την στιγμή που αξιολογούν την επίδραση ενός επίκαιρου αντικειμένου, των προτύπων σχεδίασης, στην ποιότητα του λογισμικού. Τέλος, τα ευρήματα της εργασίας χρησιμοποιήθηκαν ως μέρος ενός επιστημονικού άρθρου, προς κρίση, σε γνωστό συνέδριο του χώρου.

ΠΕΡΙΛΗΨΗ

Η διασφάλιση της ποιότητας του λογισμικού συνδέεσαι άμεσα με την έννοια της μετρικής, που είναι μια διαδικασία απαραίτητη για την εκτίμηση της κατάστασης των προϊόντων, των διαδικασιών και των πόρων παραγωγής λογισμικού. Με την εφαρμογή των μετρικών σε ένα λογισμικό, μετρούνται εκείνα τα χαρακτηριστικά που συμβάλουν σημαντικά στην ποιότητά του. Έτσι, είναι δυνατό να εξαχθούν συμπεράσματα για το κατά πόσο το λογισμικό πληρεί τα κριτήρια της ποιότητας.

Στην αντικειμενοστρέφεια ένα *πρότυπο σχεδίασης* ορίζεται ως μία αποδεδειγμένα καλή λύση που έχει εφαρμοστεί με επιτυχία στην επίλυση ενός επαναλαμβανόμενου προβλήματος σχεδίασης συστημάτων λογισμικού. Τα πρότυπα σχεδίασης πρεσβεύουν τον πιο ορθό τρόπο ανάπτυξης του λειτουργικού σώματος ενός συστήματος, των οποίων οι ωφέλειες είναι ευεργετικές.

Η εμπειρική μελέτη αυτή αποσκοπεί να ελέγξει την επίδραση των προτύπων σχεδίασης (Design Pattern) στη ποιότητα του λογισμικού. Για τη μελέτη περίπτωσης χρησιμοποιήθηκε ένα λογισμικό ανοιχτού κώδικα της κατηγορίας «Εργαλεία Σχεδίασης Λογισμικού» (Software Design Tools) σε πέντε (5) διαδοχικές εκδόσεις με επιβεβαιωμένη προσθήκη σχεδιαστικών προτύπων. Σε αυτές τις εκδόσεις μελετηθήκαν οι τιμές των αντικειμενοστρεφών μετρικών με σκοπό να παραχθούν συμπεράσματα για την ωφέλιμη ή επιζήμια χρήση των σχεδιαστικών προτύπων στην ανάπτυξη λογισμικού.

ABSTRACT

Software quality is related with the concept of metrics. Metrics are considered essential to estimate the state of product, the procedures and the resources for the software production. Through the application of metrics to a software product, the characteristics that contribute to its quality can be measured. Thus, conclusions concerning the software quality standards can be drawn.

In object-oriented programming a design pattern is defined as a proven solution, successfully applied, for solving commonly occurring problems in software design. Design patterns constitute a beneficial way to develop the operational body of a system.

This empirical study aims at exploring the effect of design patterns employment in software quality. For the case study, an open-source software design tool was investigated in five (5) successive versions that employed a different number of design patterns. In these versions, the scores of object-oriented metrics were calculated, in order to draw conclusions concernicng the impact, beneficial or harmful, of design pattern employment in software development.

ΕΥΧΑΡΙΣΤΙΕΣ

Η εκπόνηση της παρούσας πτυχιακής εργασίας γίνεται μετά από πολύ προσπάθεια. Βοηθοί και συμπαραστάτες σε όλη αυτήν την προσπάθεια ήταν ο αναπληρωτής καθηγητής κ.Δεληγιάννης Ιγνάτιος και ο υποψήφιος διδάκτορας κ.Αμπατζόγλου Απόστολος. Ιδιαίτερες ευχαριστίες στον Απόστολο που τόσο καιρό με την υποστήριξή του, την καθοδήγησή του, και την υπομονή του, στάθηκε αρωγός στη ολοκλήρωση της πτυχιακής εργασίας. Η γνωριμία μου με τον Απόστολο μου έδωσε εμπειρία και γνώση.

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΕΙΣΑΓΩΓΗ

Το ελεύθερο λογισμικό αντιπροσωπεύει ένα αποδοτικό μοντέλο ανάπτυξης λογισμικού, αντηχώντας την γενικότερη φιλοσοφία του λεγόμενου δικτυακού τρόπου παραγωγής. Μια κατηγορία λογισμικών που περιλαμβάνεται στα ελευθέρα λογισμικά είναι τα εργαλεία σχεδίασης λογισμικού.

1.1 Εργαλεία Σχεδίασης Λογισμικού

Το ελεύθερο λογισμικό όπως ορίζεται από το Ίδρυμα Ελευθέρου Λογισμικού (Free Software Foundation), είναι λογισμικό που μπορεί να χρησιμοποιηθεί, αντιγραφεί, μελετηθεί, τροποποιηθεί και αναδιανεμηθεί χωρίς περιορισμό. Το ελεύθερο λογισμικό ορισμένες φορές αναφέρεται και σαν ανοιχτό λογισμικό ή λογισμικό ανοιχτού κώδικα αλλά οι δύο αυτές έννοιες δεν είναι ταυτόσημες [Stamelos]
.

Με τον όρο Λογισμικό Ανοικτού Κώδικα εννοείται το λογισμικό του οποίου ο πηγαίος κώδικας διατίθεται ελεύθερα σε αυτούς που θέλουν να τον εξετάσουν, και/ή τροποποιήσουν ή να το χρησιμοποιήσουν σε άλλες εφαρμογές *[Sowe]* . Σε γενικές γραμμές το λογισμικό ανοικτού κώδικα δεν σημαίνει απαραίτητα δωρεάν λογισμικό, αλλά αναφέρεται κυρίως στην ελευθερία του κάθε χρήστη να εξετάσει και να χρησιμοποιήσει την γνώση και τις δυνατότητες που του προσφέρει ο κώδικας προγραμματισμού.

Οι εκδόσεις Ελεύθερου Λογισμικού κάνουν χρήση ειδικής άδειας (free software licence) σύμφωνα με την οποία, παραχωρείται το δικαίωμα αντιγραφής, τροποποίησης και αναδιανομής του λογισμικού, στους χρήστες. Η άδεια χρήσης ελεύθερου λογισμικού περιλαμβάνει την ελευθερία χρήσης του προγράμματος για οποιονδήποτε σκοπό και την ελευθερία βελτίωσης του προγράμματος και επανέκδοσης του προς το συμφέρον της κοινότητας των χρηστών με ελεύθερη πρόσβαση στον πηγαίο κώδικα του λογισμικού.

Στον αντίποδα βρίσκεται το "κλειστό" (closed source), του οποίου ο πηγαίος κώδικας παραμένει κρυφός στους χρηστές του λογισμικού. Το μοντέλο του κλειστού λογισμικού ακολουθούν οι περισσότερες μεγάλες εμπορικές εταιρείες λογισμικού.

Ο στόχος της αρχιτεκτονικής λογισμικού είναι να επιτρέψει στη σχεδίαση του συστήματος να πραγματοποιηθεί σε υψηλό επίπεδο αφαίρεσης. Το επίπεδο αυτό αφορά τα συστατικά , τις συνδέσεις, και τους περιορισμούς. Τα εργαλεία σχεδίασης λογισμικου αποσκοπούν σε τέτοιου είδους αρχιτεκτονική, σχεδιάζοντας λογισμικές λύσεις, συγκεκριμενοποιώντας τις απαιτήσεις και τη μορφή του συστήματος. Συμπεριφορές συστήματος όπως ευρωστία, συντηρισιμότητα, ευελιξία, μεταφερσιμότητα, αξιοπιστία, επαναχρησιμότητα, και ανοχή σε λάθη, είναι απόρροια της χρήσης και εφαρμογής των εργαλείων σχεδίασης λογισμικου. Τα εργαλεία αυτά χρησιμοποιούν μεντελικές γλώσσες για τη σχεδίαση όπως Behavior Trees, U.M.L. , γράφους του Petri, LePUS3, IDEF4, Java Modeling Language (JML), Framework-Specific Modeling Language(FSML), Domain-specific modeling (DSM).

1.2 Στόχος

Στόχο της πτυχιακής αποτελεί η μελέτη της επίδρασης των προτύπων σχεδίασης (Design Pattern) στη ποιότητα του λογισμικού. Ως ποιότητα του λογισμικού θεωρούνται οι τιμές των γνωστότερων μετρικών αντικειμενοστρεφούς σχεδίασης (Object-Oriented Metrics).

Για την επίτευξη του προαναφερθέντα στόχου εκτελέστηκε μια εμπειρική μελέτη και πιο συγκεκριμένα μια μελέτη περίπτωσης. Ως σύνολο δεδομένων για την μελέτη χρησιμοποιήθηκε ένα λογισμικό ανοιχτού κώδικα της κατηγορίας «Εργαλεία Σχεδίασης Λογισμικού» (Software Design Tools). Πιο συγκεκριμένα, το εν λόγω λογισμικό μελετήθηκε σε πέντε (5) εκδόσεις του. Η μελέτη περιλάμβανε ανίχνευση των προτύπων σχεδίασης, οπτική επιβεβαίωση της ύπαρξης του προτύπου και την εξαγωγή των μετρικών. Τέλος, στα πλαίσια της μελέτης χρησιμοποιήθηκαν περιγραφικές στατιστικές μέθοδοι και τεστ υποθέσεων για την εξερεύνηση των στόχων.

Η σχεδίαση και η διεξαγωγή της παρούσας μελέτης αποδείχτηκε ιδιαίτερα χρήσιμη για την εξοικείωση του συγγραφέα με προηγμένα θέματα τεχνολογίας λογισμικού, καθώς και με τους τρόπους εμπειρικής αξιολόγησης εργαλείων και μεθόδων τεχνολογίας λογισμικού. Επιπλέον, τα αποτελέσματα της εργασίας θεωρούνται σημαντικά και ευρύτερου επιστημονικού ενδιαφέροντος από την στιγμή που αξιολογούν την επίδραση ενός επίκαιρου αντικειμένου, των προτύπων σχεδίασης, στην ποιότητα του λογισμικού. Τέλος, τα ευρήματα της εργασίας χρησιμοποιήθηκαν ως μέρος ενός επιστημονικού άρθρου, προς κρίση, σε γνωστό συνέδριο του χώρου.

Η πτυχιακή εργασία αποτελείται από πέντε κεφαλαία. Το πρώτο κεφαλαίο περιλαμβάνει τα εισαγωγικά μέρη της πτυχιακής εργασίας όπως μια εισαγωγή στην κατηγορία του λογισμικού που πρόκειται να μελετηθεί και τον σκοπό της πτυχιακής εργασίας. Στο δεύτερο κεφάλαιο παρουσιάζεται η Ενοποιημένη Γλώσσα Μοντελοποίησης UML, οι Αρχές Αντικειμενοστρεφούς σχεδίασης, η αναλυτική περιγραφή των σχεδιαστικών προτύπων και τα ελεγχόμενα πειράματα που έχουν διεξαχθεί για τη μελέτη τους. Στο τρίτο κεφάλαιο αναλύεται η έννοια της μέτρησης στην ποιότητα λογισμικού, περιγράφονται αναλυτικά οι αντικειμενοστρεφείς μετρησεις και οι εμπειρικές μελέτες που έχουν εξετάσει την επίδραση των σχεδιαστικών προτύπων στις τιμές των μετρικών. Στο τέταρτο κεφαλαίο που ακολουθεί γίνεται αναφορά στους τρόπους εκπόνησης μιας μελέτης, και παρουσιάζεται η μεθοδολογία της μελέτης πεδίου της παρούσας πτυχιακής εργασίας. Τα συμπεράσματα της μελέτης πεδίου παρουσιάζονται στο πέμπτο κεφαλαίο. Ακολουθεί η βιβλιογραφία, το ευρετήριο όρων και η λίστα πινάκων και σχημάτων.

2. ΠΡΟΤΥΠΑ ΣΧΕΔΙΑΣΗΣ

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 Ακολουθεί παρουσίαση της UML και εκτενής ανάλυση των σχεδιαστικών προτύπων.

2.1 Εισαγωγή στη U.M.L.

Η Ενοποιημένη Γλώσσα Μοντελοποίησης (Unified Modeling Language) είναι μια γραφική γλωσσά γενικού σκοπού, η οποία χρησιμοποιείται για τον προσδιορισμό, την οπτικοποίηση, ανάπτυξη και τεκμηρίωση των κατασκευμάτων ενός συστήματος λογισμικού, παρέχοντας μια σημειογραφική προσέγγιση για την περιγραφή αντικειμενοστρεφών λύσεων. Μπορεί να προσαρμοστεί ώστε να ταιριάζει σε διαφορές καταστάσεις ανάπτυξης και κύκλους ζώης λογισμικού. Μάλιστα, οοργανισμοί όπως η Ομάδα Διαχείρισης Αντικειμένων (Object Management Group) έχουν υιοθετήσει τη UML ως πρότυπη γλωσσά στην αντικειμενοστραφή σημειογραφία.

Στα σύνθετα έργα λογισμικού όπου η μοντελοποίηση του συστήματος είναι αναγκαία η UML χρησιμοποιείται για να απεικονίσει (visualize), προσδιορίσει (specify), κατασκευάσει (construct) και να τεκμηριώσει (document) τα επιμέρους συστατικά (artifacts) των πληροφοριακών συστημάτων. Η απεικόνιση αφορά τη γραφική αναπαράσταση του συστήματος, που βοηθάει στην κατανόηση των επί μέρους προβλημάτων του συστήματος αλλά και στην επικοινωνία με τους υπόλοιπους συμμετέχοντες στην ανάπτυξη. Η UML διαθέτει μια συλλογή από κατάλληλες μεθόδους αλλά και μηχανισμούς επεκτασιμότητας (extensibility) και εξειδίκευσης (specialization) ώστε να μπορεί να απεικονίσει με ακριβή και πλήρη τρόπο μεγάλα και πολύπλοκα συστήματα. Είναι ανεξάρτητη από συγκεκριμένες

γλώσσες προγραμματισμού (programming languages) και διαδικασίες ανάπτυξης λογισμικού (development processes).

Βασισμένη στις αρχές της αντικειμενοστρέφειας, η UML μπορεί να χρησιμοποιηθεί σε όλες τις φάσεις της ανάπτυξης λογισμικού, από την προδιαγραφή των απαιτήσεων μέχρι τη σχεδίαση και τη συγγραφή του κώδικα. Χρησιμοποιώντας το μηχανισμό της αφαίρεσης εστιάζεται στις απαραίτητες πληροφορίες που πρέπει να κατανοηθούν, απεικονισθούν και τεκμηριωθούν πριν αλλά και κατά την διάρκεια της ανάπτυξης του συστήματος. Είναι εξαιρετικά χρήσιμη για την περιγραφή διαφορετικών εναλλακτικών σχεδίων και προσεγγίσεων του ιδίου προβλήματος, παρέχοντας μια κοινή γλώσσα επικοινωνίας σε όλες τις ομάδες ανάπτυξης.

Η UML (Unified Modeling Language) είναι μία γενικού σκοπού τυποποιημένη γλώσσα απεικόνισης – μοντελοποίησης συστημάτων που χρησιμοποιείται κατά κόρον στην Τεχνολογία Λογισμικού. Περιέχει συμβολισμούς ικανούς να περιγράψουν αφαιρετικά ένα σύστημα επικεντρώνοντας στα σημαντικά στοιχεία της αρχιτεκτονικής του. Μπορεί να οπτικοποιήσει, να προσδιορίσει, να τεκμηριώσει εα πρόβλημα, όπως επίσης μπορεί να χρησιμοποιηθεί για την ανάλυση, την σχεδίαση και την παρακολούθηση της προόδου αντικειμενοστρεφών πληροφοριακών συστημάτων. Είναι εξαιρετικά χρήσιμη για την περιγραφή διαφορετικών εναλλακτικών σεναρίων όπως επίσης μπορεί να αναπαραστήσει έννοιες όπως επιχειρηματικές διαδικασίες και λειτουργίες συστημάτων αλλά και πιο υπαρκτές έννοιες όπως δηλώσεις γλωσσών προγραμματισμού, σχήματα βάσεων δεδομένων αλλά και επαναχρησιμοποιούμενων συστατικών λογισμικού.

Παρόλο που η UML δεν είναι μία μεθοδολογία ανάπτυξης λογισμικού αποτελεί απαραίτητο εργαλείο για την επιτυχή ανάπτυξη λογισμικού. Διατηρεί μία ουδέτερη στάση ως προς τις διάφορες μεθοδολογίες και είναι συμβατή με τις περισσότερες αν όχι όλες.

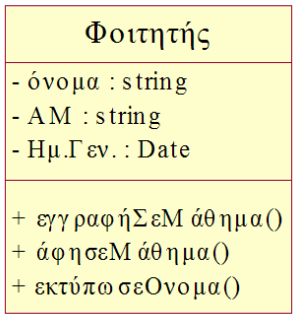
Η δύναμη της UML κρύβεται στους διάφορους τύπους διαγραμμάτων που παρέχει. Τα διαγράμματα UML περιλαμβάνουν τη δυναμική όψη του συστήματος, τη στατική όψη, τους περιορισμούς, και την τυποποίηση τους.

Η δυναμική όψη απεικονίζεται με το διάγραμμα περιπτώσεων χρηστής (use case diagram), το οποίο χρησιμοποιείται για τη μοντελοποίηση της συμπεριφοράς

ενός συστήματος και υποσυστημάτων του από περιπτώσεις και εναλλακτικά σενάρια, όπως αυτή γίνεται αντιληπτή από το τελικό χρηστή. τις λίστες δραστηριοτήτων, τα διαγράμματα αλληλεπίδρασης που επιδεικνύουν τις ακολουθίες και τη συνεργασία μεταξύ των κλάσεων, και τις μηχανές καταστάσεων που περιγράφουν τις καταστάσεις και τις αλλαγές τους. Στη διεργασία των απαιτήσεων, τα διαγράμματα ροής εργασίας (workflow diagrams) καθορίζουν τη συνολική επιχειρηματική διεργασία που θα υποστηρίξει το σύστημα. Τα διαγράμματα δραστηριοτήτων (activity diagrams) εμφανίζουν όλες τις δραστηριότητες που μπορούν να προκύψουν στο σύστημα καθώς μαλάζουν οι τιμές ενός αντικειμένου. Τα διαγράμματα καταστάσεων (state diagrams) υποδεικνύουν τις πιθανές καταστάσεις στις οποίες ένα αντικείμενο μπορεί να βρεθεί μέσω ανταλλαγής μηνυμάτων τα οποία αναπαριστούν ένα γεγονός .

Η στατική όψη απεικονίζεται με διαγράμματα τω κλάσεων (class diagrams), τις διαφορές σχέσεις (συσχέτιση, γενίκευση, εξάρτηση) και την επεκτασιμότητα (περιορισμοί, τιμές ετικετών, και στερεότυπα). Τα διαγράμματα ακολουθίας (sequence diagrams) δείχνουν τον τρόπο με τον οποίο τα μηνύματα ρέουν από ένα αντικείμενο σε ένα άλλο, τυποποιώντας τις άτυπες περιγραφές των γεγονότων στις απαιτήσεις. Τα διαγράμματα συνεργασίας (collaboration diagrams) χρησιμοποιούν πληροφορίες για τα αντικείμενα και τις ακολουθίες για να δείξουν τη ροή των γεγονότων μεταξύ των αντικειμένων. Τα διαγράμματα πακέτων (package diagrams)δείχνουν το τρόπο που οι κλάσης διαιρούνται λογικά σε ομάδες ενώ τα διαγράμματα συστατικών (component diagrams) αντανακλούν τις τελικές μονάδες του συστήματος. Οι κλάσεις αποτελούν την βάση της κατασκευής οποιουδήποτε αντικειμενοστρεφούς συστήματος. Μετά την ανάλυση των προδιαγραφών οι κατασκευαστές έχουν αποκτήσει γνώση για το πεδίο προβλήματος του συστήματος και το απεικονίζουν με ένα διάγραμμα κλάσεων, που ονομάζεται μοντέλο του πεδίου προβλήματος (domain model). Στο μοντέλο αυτό καταγράφονται κλάσεις που αποτελούν έννοιες του πεδίου προβλήματος, χωρίς αυτό να σημαίνει ότι θα χρησιμοποιηθούν στο λογισμικό που θα κατασκευαστεί. Επίσης, για κάθε κλάση του μοντέλου μπορούν να αναγραφούν κάποιες υψηλού επιπέδου υποχρεώσεις (λειτουργίες) αλλά και να απεικονισθούν οι συσχετίσεις μεταξύ των κλάσεων. Στο στάδιο της επεξεργασίας το μοντέλο του πεδίου προβλήματος θα αποτελέσει τη βάση για την επιλογή των κλάσεων αλλά και των συσχετίσεων, ίσως με

περισσότερες αλλαγές και βελτιώσεις. Οι λειτουργίες των κλάσεων αποτελούν τις *μεθόδους* (methods) των αντικειμένων αυτών των κλάσεων, που καλούνται από άλλα αντικείμενα άλλων συνεργαζόμενων κλάσεων. Στα διαγράμματα κλάσεων υπάρχει η δυνατότητα απεικόνισης υψηλού επιπέδου λεπτομερειών, όπως οι ιδιότητες και λειτουργίες μιας κλάσης σχέσεις κληρονομικότητας μεταξύ κλάσεων, εξαρτήσεις μεταξύ πακέτων κ.ο.κ..



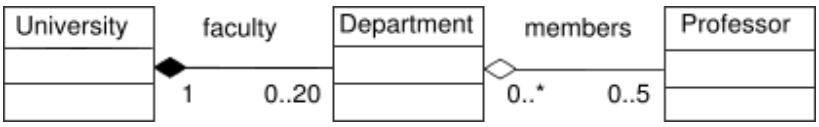
□□□□□α 1. □□□□□□□□σ□□□ □□□σ□□ σε □□□□□α□□α □□□σε□□

Ο συμβολισμός μιας κλάσης σε ένα τέτοιο διάγραμμα είναι ένα ορθογώνιο διαμελισμένο σε τρία τμήματα. Στο άνω τμήμα αναγράφεται το όνομα της κλάσης, στο μεσαίο τμήμα οι ιδιότητες δηλαδή καταστάσεις αντικειμένων και στο κάτω τμήμα οι μέθοδοι της δηλαδή οι λειτουργίες της κλάσης.

Οι συσχετίσεις (associations) παρέχουν στατικές διασυνδέσεις βάσει των οποίων τα αντικείμενα διαφορών κλάσεων μπορούν να αλληλεπιδρών και να ανταλλάσουν μηνύματα. Χωρίς συσχετίσεις υπάρχουν μονό ανεξάρτητα αντικείμενα που δε συνεργάζονται. Μια συσχέτιση συμβολίζεται σε ένα διάγραμμα κλάσεων ως μια συνεχής γραμμή που συνδέει τις κλάσης που συμμετέχουν. Το όνομα της συσχέτισης τοποθετείται πλησίον της γραμμής, ενώ το όνομα ρολών των ακρών και η πολλαπλότητα στις άκρες τις γραμμής. Πολυπλοκότητα ή αλλιώς πληθικότητα είναι ο αριθμός των στιγμιότυπων μιας κλάσης που μπορούν να συσχετισθούν με ένα στιγμιότυπο μιας άλλης κλάσης.

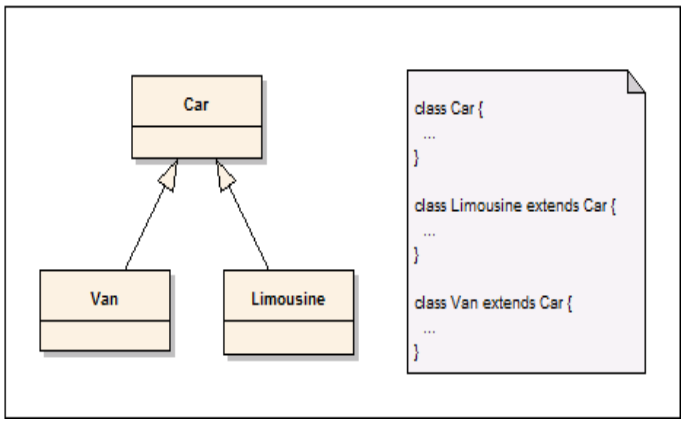
Η συσσωμάτωση (aggregation) είναι μια συσχέτιση ειδικής μορφής που αναπαριστά μια σχέση συνόλου-τμήματος ή όλου-τμήματος, δηλαδή ότι μια κλάση είναι τμήμα μιας άλλης κλάσης. Η δήλωση αυτής της σχέσης είναι μια γραμμή που

καταλήγει σε ένα λευκό-κενό ρόμβο, ο οποίος υποδεικνύει την ευρύτερη οντότητα της οποίας η συνδεδεμένη κλάση αποτελεί τμήμα. Η σύνθεση (composition) είναι μια ησυχότερης μορφής συσχέτιση, στην οποία το σύνολο έχει την αποκλειστική διαχείρισης των τμημάτων όπως η δημιουργία και η διαγραφή τους. Συμβολίζεται με ένα μαύρο ρόμβο στην άκρη του συνόλου.



□□□□□α 2. □α□□δε□□□α ε□δ□□□□ σ□σ□ετ□σε□□

Η κληρομικότητα (inheritance) είναι μια συσχέτιση μεταξύ μιας γενικής και μιας ειδικότερης περιγραφής που την επεκτείνει. Η ειδικότερη περιγραφή είναι απολύτως συνεπής με την γενική δηλαδή έχει τις ίδιες ιδιότητες, λειτουργίες και σχέσεις αλλά μπορεί να περιλαμβάνει και επιπρόσθετη πληροφορία. Η κληρονομικότητα συμβολίζεται με ένα μικρό τρίγωνο βέλος στο άκρο που αντιστοιχεί στην γονική περιγραφή δηλαδή τη βασική κλάση γονέας. Χρησιμοποιείται κάθετη διάταξη, Έτσι ώστε η βασική κλάση να
βρίσκεται υψηλοτέρα από τις παραγωγές κλάσης. Η κληρονομικότητα επιτρέπει πολυμορφική συμπεριφορά και λειτουργίες στο σύστημα



□□□□□α 3. □α□□δε□□□α □□□□□□□□□□□□□τ□τα□

Στην παραπάνω εικόνα η οντότητα Van και Limousine ανήκουν στην γενικευμένη οντότητα Car, κληρονομώντας όλες τις ιδιότητες και τα χαρακτηριστικά της κλάσης Car

2.2. Αρχές σχεδίασης

Το κόστος σχεδίασης στον κύκλο ζωής προϊόντος λογισμικού είναι σημαντικό και εξαρτάται συνήθως από το μέγεθος του συστήματος και κατά επέκταση από το πλήθος και το είδος των απαιτήσεων. Ωστόσο αυτό που καθιστά τη σχεδίαση ιδιαιτέρως σημαντική από πλευράς κόστους, είναι ότι καθορίζει σε εξαιρετικά μεγάλο βαθμό, την ευκολία με την οποία μπορούν να πραγματοποιηθούν αλλαγές στο σύστημα. Όσο πιο μεγάλη πρόνοια ληφθεί κατά την διάρκεια της σχεδίασης του συστήματος ώστε να μπορεί να τροποποιηθεί και να επεκταθεί με μικρή προσπάθεια, τόσο οικονομικότερη γίνεται η εξέλιξη του λογισμικου.

Τα συμπτώματα μιας κακής ποιότητας σχεδίασης μπορεί να παρουσιαστούν σε οποιαδήποτε φάση της διαρκείας ζωής με αρνητικές πάντα συνέπειες. Φαινόμενα όπως, δυσκαμψία σε αλλαγές και τροποποιήσεις(Rigidity), ευθραυστότητα κατά την εμφάνιση σφαλμάτων(Fragility), ακινησία σε περιπτώσεις μη επαναχρησιμοποίησης(Immobility), έλλειψη ρευστότητας σε αλλαγές (Viscosity), περιττή πολυπλοκότητα(Needless Complexity) και περιττή επανάληψη(Needless Repetition), μπορούν να παρουσιαστούν στην ανάπτυξη και εξέλιξη του λογισμικου.

Η αντικειμενοστραφής σχεδίαση πρέπει να διέπεται από αρχές. Η παραβίαση τους επιφέρει συμπτώματα όπως αυτά που καταγραφήκαν παραπάνω. Οι αρχές σχεδίασης είναι προϊόν εμπειρίας δεκαετιών στην τεχνολογία λογισμικου. Αξίζει να σημειωθεί ότι δεν πρέπει να πραγματοποιούνται διαρκώς και χωρίς λόγο, αλλά μονό όταν παρατηρηθούν κάποια συμπτώματα ανορθόδοξης σχεδίασης.

Συμφωνά με την Αρχή της Ανοικτής – Κλειστής Σχεδίασης οι οντότητες του λογισμικου θα πρέπει να είναι ανοικτές για επέκταση και πρόσθεση νέων λειτουργιών, αλλά κλειστές σε τροποποιήσεις. Αυτό επιτυγχάνεται με αναδόμηση (refactoring) δηλαδή με προσθήκη νέου κώδικα και οχι με την τροποποίηση του

υπάρχοντος. Η εφαρμογή αυτής της αρχής σε ένα σύστημα εξαλείφει συμπτώματα ευθραυστότητας και ακινησίας.

Η Αρχή της Ενσωμάτωσης επιβάλει την εσωτερική κατάσταση ενός αντικειμένου να είναι τροποποιήσιμη μονό μέσω της δημοσιάς διασύνδεσης του. Σύμφωνα με την ανωτέρω αρχή οι ιδιότητες πρέπει να μην είναι προσπελάσιμες εκτός κλάσης, δηλαδή η τροποποίηση των τιμών των ιδιοτήτων θα πρέπει να επιτρέπεται μόνο μέσω της πρόσβασης που παρέχουν οι δημόσια διαθέσιμες μέθοδοι. Η παραβίαση της αρχής της ενσωμάτωσης προκαλεί σημαντικά προβλήματα συντήρησης, δυσκαμψίας, ευθραυστότητας και ακινησίας.

Η Αρχή της Χαμηλής Σύζευξης υποστηρίζει ότι σε ένα σχέδιο λογισμικου πρέπει να επιδιώκεται η επίτευξη της μικρότερης δυνατής Σύζευξης μεταξύ των συστατικών του. Με αυτόν τον τρόπο εξασφαλίζεται η ανεξαρτησία μεταξύ των συστατικών του σχεδίου και γίνεται ευκολότερη η υλοποίηση, ο έλεγχος και η συντηρηση.

Η Αρχή της Μοναδικής Αρμοδιότητας επιβάλει ότι μια κλάση πρέπει να έχει μονό ένα λόγο να αλλάξει και οχι παραπάνω από μια αρμοδιότητες. Ουσιαστικά εξασφαλίζει υψηλή συνεκτικότητα στα συστατικά ενός συστήματος λογισμικου.

Συμφωνά με την Αρχή της Υποκατάστασης της Liskov, οι παράγωγοι τύποι πρέπει να μπορούν να υποκαθιστούν τους βασικούς τύπους. Αυτό ουσιαστικά σημαίνει ότι η Αρχή της Υποκατάστασης της Liskov επιβάλει σε μια σχέση κληρονομικότητας η συμπεριφορά των παραγόμενων κλάσεων να μπορεί να υποκαταστήσει τη συμπεριφορά των βασικών κλάσεων. Κάθε παραβίαση της ανωτέρω αρχής είναι μια εν δυνάμει παραβίαση της αρχής Ανοικτής – Κλειστής σχεδίασης.

Η Αρχή της Αντιστροφής των Εξαρτήσεων δηλώνει ότι οι μονάδες υψηλού επιπέδου δεν θα πρέπει να εξαρτώνται από μονάδες χαμηλού επιπέδου, καθώς επίσης οι αφαιρέσεις δεν θα πρέπει να εξαρτώνται από λεπτομέρειες αλλά το αντίστροφο. Στόχος της Αρχή της Αντιστροφής των Εξαρτήσεων είναι να αντιστρέφει πλήρως τη δομή ενός συστήματος λογισμικου, έτσι ώστε οι μονάδες υψηλού επιπέδου να μην εξαρτώνται, αλλά να καθορίζουν τους κανόνες για τις μονάδες χαμηλού επιπέδου.

Η Αρχή του Διαχωρισμού των Διασυνδέσεων διαπραγματεύεται τα μειονεκτήματα των ογκωδών διασυνδέσεων και διατυπώνει ότι πολλές διασυνδέσεις είναι προτιμότερες από μια γενική. Οι κλάσεις που έχουν μεγάλο αριθμο δημόσιων μεθόδων είναι κλάσεις των οποίων η διασύνδεση δεν είναι συνεκτική και οι μέθοδοι μπορούν να χωριστούν σε ένα σύνολο διασυνδέσεων. Επίσης η Αρχή του Διαχωρισμού των Διασυνδέσεων διατυπώνει ότι οι πελάτες δε θα πρέπει να εξαναγκάζονται σε εξάρτηση από μεθόδους που δεν χρησιμοποιούν γιατί η εξάρτηση αυτή έχει ως αποτέλεσμα ανεπιθύμητη σύζευξη. Ο διαχωρισμός των Διασυνδέσεων μπορεί να προκληθεί είτε μέσω πολλαπλής κληρονομικότητας, είτε μέσω διαβίβασης μηνυμάτων.

2.3. Ανάλυση Προτύπων Σχεδίασης

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

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

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

Το όνομα κάθε προτύπου διευκολύνει την επικοινωνία μεταξύ προγραμματιστών καθώς και την εύκολη αναφορά σε κοινά είδη προβλημάτων. Το πρόβλημα σε κάθε πρότυπο προσδιορίζει ένα γενικότερο πλαίσιο όπου υπό κανονική αντιμετώπιση(χωρίς τη χρήση προτύπων) θα προέκυπταν ανεπιθύμητες συνέπειες αναφορικά με την λειτουργία, τον έλεγχο και τη συντήρηση του λογισμικού.

Κάθε πρότυπο επιτρέπει να αλλάξει μια όψη από την αρχιτεκτονική δομή χωρίς να επηρεάζει τις υπόλοιπες. Έχουν οριστεί διάφορες κατηγορίες προτύπων, για διαφορετικά προβλήματα και κάθε κατηγορία περιλαμβάνει πολλαπλά στοιχεία. Έτσι υπάρχουν κατασκευαστικά πρότυπα, δομικά πρότυπα και συμπεριφορά 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 κάποια αντικείμενα από τις κλάσεις που ορίζουν τα αντικείμενα αυτά και τον τρόπο που κατασκευάζονται στη μνήμη, σύμφωνα με την αρχή ανοιχτότητας-κλειστότητας για ορθή αντικειμενοστρεφή σχεδίαση. Τέτοια πρότυπα είναι το *Factory Method,* το *Singleton και το Prototype.*
- Δομικά Πρότυπα (Structural Patterns) : Τα δομικά πρότυπα αφορούν τυποποιημένους τρόπους δυναμικής κατασκευής σύνθετων αντικειμένων τα οποία χρησιμοποιούν υπάρχουσες ιεραρχίες κλάσεων. Κρατάνε τις συζεύξεις χαμηλές, τις κλάσεις αμετάβλητες, και προσφέρει εναλλακτικές

λύσεις στην κληρονομικότητα. Τέτοια πρότυπα είναι το *Composite, το Adapter και το Decorator Pattern*

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

2.3.1 Πρότυπο ΄΄Προσαρμογέας΄΄ (Adapter Pattern)

Το πρότυπο σχεδίασης ΄΄Προσαρμογέας΄΄ (Adapter) έχει ως στόχο τη μετατροπή της διασύνδεσης μιας κλάσης σε μια άλλη που αναμένει το πρόγραμμα πελάτης. Ο ΄΄Προσαρμογέας΄΄ επιτρέπει τη συνεργασία κλάσεων, η οποία σε διαφορετική περίπτωση θα ήταν αδύνατη λόγω ασύμβατων διασυνδέσεων. Χρησιμοποιούμε τους ΄΄Προσαρμογείς΄΄ όποτε θέλουμε ανεξάρτητες κλάσεις να λειτουργούν μαζί σε ένα ενιαίο πρόγραμμα. Γράφουμε μια κλάση που έχει την επιθυμητή διεπαφή και την κάνουμε έπειτα να επικοινωνήσει με την κλάση που έχει μια διαφορετική διεπαφή.

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.

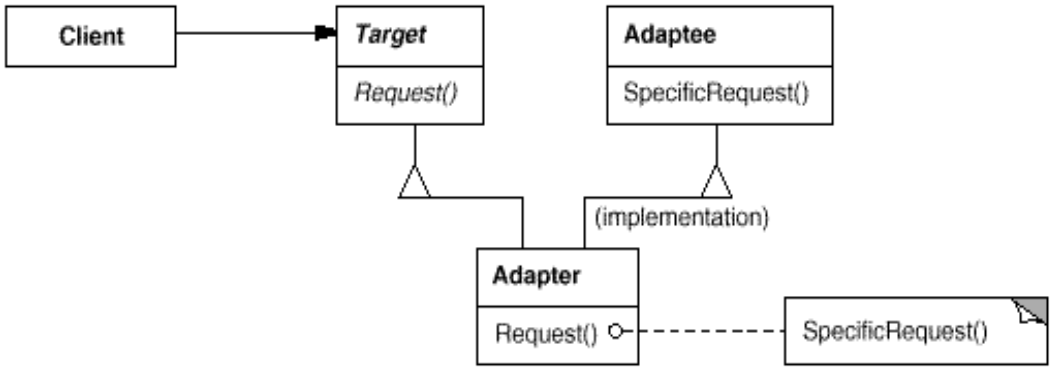
Το πρότυπο αυτό επιτρέπει την προσαρμογή της διασύνδεσης που εξάγει μία κλάση Α σε μία πρότυπη διασύνδεση, έστω Interface, που αναμένεται από ένα εξωτερικό πρόγραμμα ώστε να μην παραβιάζεται η Αρχή Ανοιχτότητας-Κλειστότητας. Αυτό γίνεται μέσω μίας ενδιάμεσης κλάσης, του Adapter, η οποία υλοποιεί το Interface 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.

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

Ένας ΄΄Προσαρμογέας΄΄ χρησιμοποιείται όταν θέλουμε να χρησιμοποιήσουμε μια υπάρχουσα κλάση, αλλά η διασύνδεσή της δεν συμβαδίζει με τις ανάγκες μας. Ένας προσαρμογέας αντικειμένου (object adapter)βασίζεται στη σύνθεση αντικειμένων και στη διαβίβαση μηνυμάτων (delegation). Ένας προσαρμογέας κλάσης (class adapter) χρησιμοποιεί πολλαπλή κληρονομικότητα για να προσαρμόσει μια διασύνδεση σε μια άλλη.

Γενική δομή



□□□□□α□□α □□□σε□□ π□□τ□π□□ □□□□□σα□□□□□α□□□

2.3.2 Πρότυπο ΄΄Μοναδιαίο΄΄ (Singleton Pattern)

Το πρότυπο σχεδίασης ΄΄Μοναδιαίο΄΄ (Singleton) εξασφαλίζει ότι μια κλάση θα έχει μόνο ένα στιγμιότυπο και παρέχει ένα καθολικό σημείο πρόσβασης σε αυτό. Χρησιμοποιείται όταν σε κάποιο σύστημα λογισμικού υπάρχει η απαίτηση από μια κλάση να δημιουργείται ένα και μόνο ένα αντικείμενο. Επίσης παρέχεται ένας συγκεκριμένος τρόπος για να μπορούμε να το ανακτούμε από οπουδήποτε. Το αντικείμενο αυτό συνήθως δημιουργείται κατά την έναρξη της εφαρμογής και διαγράφεται μετά το πέρας της.

Ο ρόλος αυτού του μοναδικού αντικειμένου είναι η διαχείριση των υπόλοιπων αντικειμένων της εφαρμογής και για το λόγο αυτό, αποτελεί λογικό σφάλμα να δημιουργηθούν περισσότερα του ενός τέτοια αντικείμενα-διαχειριστές. Σε μια τέτοια περίπτωση, η εφαρμογή θα έχει περισσότερα του ενός σημεία εκκίνησης, και ο υποθετικός χρήστης, ανάλογα με το σημείο εκκίνησης, μπορεί να καταλήξει να χρησιμοποιεί ένα υποσύνολο των αντικειμένων του συστήματος.

Επιπλέον αν υπάρχουν περισσότεροι του ενός διαχειριστές, ενώ ο επιθυμητός στόχος είναι η ακολουθιακή εκτέλεση δραστηριοτήτων, πολλές δραστηριότητες θα εκτελούνται παράλληλα.

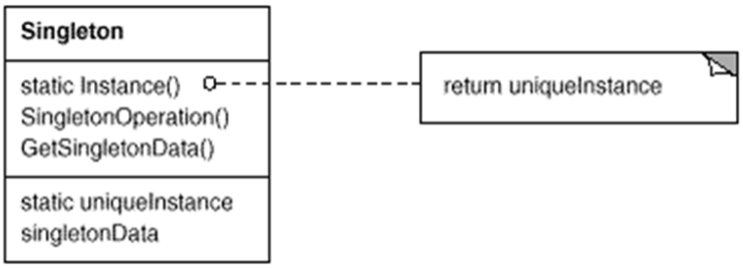
Το πρότυπο σχεδίασης ΄΄Μοναδιαίο ΄΄, εξασφαλίζει τη δημιουργία ενός και μόνο αντικειμένου, περιλαμβάνοντας μια ειδική μέθοδο κατασκευής στιγμιοτύπων:

- Όταν καλείται αυτή η μέθοδος, ελέγχει αν κάποιο αντικείμενο έχει δημιουργηθεί. Αν ναι, η μέθοδος επιστρέφει απλώς ένα δείκτη προς το υπάρχον αντικείμενο. Αν όχι, η μέθοδος δημιουργεί ένα νέο αντικείμενο και επιστρέφει δείκτη προς αυτό.
- Για να εξασφαλισθεί ότι αυτός είναι και ο μοναδιαίος τρόπος δημιουργίας αντικειμένων από αυτή την κλάση, ο κατασκευαστής της κλάσης δηλώνεται ως προστατευμένος (PROTECTED) ή ιδιωτικός (PRIVATE). Με αυτόν τον τρόπο δεν είναι δυνατόν να δημιουργηθεί ένα αντικείμενο παρακάμπτοντας την παραπάνω ειδική μέθοδο.

Η απλότητα του προτύπου επιτρέπει οποιαδήποτε κλάση να μετατραπεί σε μοναδιαία, κάνοντας τον κατασκευαστή ιδιωτικό ή προστατευμένο και προσθέτοντας

μια στατιστική μέθοδο και μια στατιστική ιδιότητα. Ένα πλεονέκτημα του προτύπου είναι ότι σε περίπτωση δημιουργίας παραγώγων κλάσεων από μια μοναδιαία κλάση, κάθε απόγονος μπορεί να είναι επίσης μοναδιαία κλάση αν προστεθούν και σε αυτές η στατιστική ιδιότητα και μέθοδος.

Γενική Δομή



□□□□□**α**□□**α** □□□**σε**□□ **π**□□**τ**□**π**□□ **“**□□□**αδ**□**α**□□□**”**

2.3.3 Πρότυπο Παρατηρητής (Observer Pattern)

Το πρότυπο σχεδίασης ΄΄Παρατηρητής΄΄ ορίζει μια σχέση εξάρτησης ένα-προς-πολλά (1:N) μεταξύ των αντικειμένων έτσι ώστε όταν μεταβάλλεται η κατάσταση ενός αντικειμένου, όλα τα εξαρτώμενα αντικείμενα να ενημερώνονται και να τροποποιούνται αυτόματα.

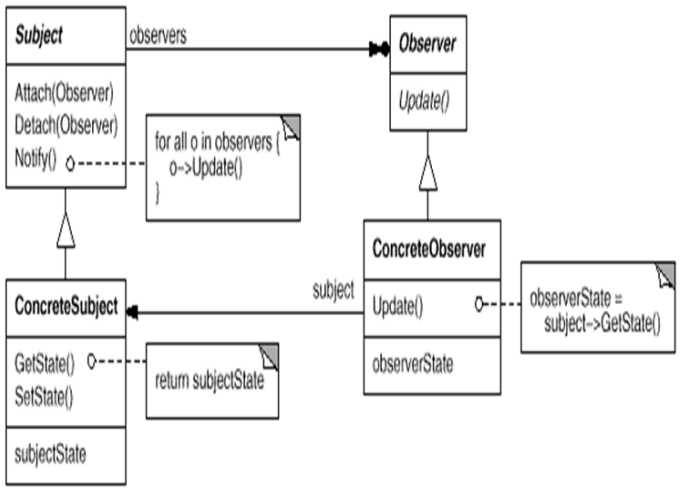
Το πρότυπο σχεδίασης ΄΄Παρατηρητής΄΄ επιδιώκει να μειώσει τη σύζευξη μεταξύ των αντικειμένων, παρέχοντας αυξημένη δυνατότητα επαναχρησιμοποίησης και τροποποίησης του συστήματος. Το συγκεκριμένο πρότυπο, επιτρέπει την αυτόματη ειδοποίηση και ενημέρωση ενός συνόλου αντικειμένων τα οποία ΄΄αναμένουν΄΄ ένα γεγονός, που εκδηλώνεται ως αλλαγή στην κατάσταση ενός αντικειμένου.

Ο στόχος είναι η απόζευξη των παρατηρητών από το παρακολουθούμενο αντικείμενο (υποκείμενο), έτσι ώστε κάθε φόρα που προστίθεται ένας νέος παρατηρητής (με διαφορετική διασύνδεση ενδεχομένως), να μην απαιτούνται αλλαγές στο παρακολουθούμενο αντικείμενο.

Η χρήση του προτύπου είναι ωφέλιμη όταν η λίστα των παρατηρητών μπορεί να αλλάζει κατά τη διάρκεια εκτέλεσης του προγράμματος, ή όταν η αλλαγή της κατάστασης ενός αντικειμένου(υποκειμένου) απαιτεί την ειδοποίηση άλλων αντικειμένων(παρατηρητών), αλλά το υποκείμενο δεν θα πρέπει να κάνει καμιά υπόθεση για το ποια είναι αυτά τα αντικείμενα. Η χρήση αυτού του προτύπου θα πρέπει να γίνεται μονό σε κατάλληλες περιπτώσεις , ειδάλλως προσθέτει περιττή πολυπλοκότητα στο σύστημα.

Η εφαρμογή αυτού του προτύπου προϋποθέτει τον εντοπισμό των εξής δύο τμημάτων: ενός υποκειμένου και του παρατηρητή. Μεταξύ των δύο υφίσταται μια σχέση ένα-προς-πολλά. Το υποκείμενο θεωρείται ότι διατηρεί το μοντέλο των δεδομένων και η λειτουργικότητα που αφορά στην παρατήρηση των δεδομένων κατανέμεται σε διακριτά αντικείμενα-παρατηρητές. Οι παρατηρητές καταχωρούνται στο υποκείμενο κατά τη δημιουργία τους. Οποτεδήποτε το υποκείμενο αλλάζει, ΄΄ ανακοινώνει ΄΄ προς όλους τους καταχωρημένους παρατηρητές το γεγονός της αλλαγής, και κάθε παρατηρητής ρωτά το υποκείμενο για το υποσύνολο της κατάστασης του υποκειμένου που το ενδιαφέρει. Το πρότυπο Παρατηρητής εφαρμόζεται για χρόνια στην ευρέως γνωστή αρχιτεκτονική Μοντέλου-Όψης-Ελεγκτή (Model-View-Controller).

Γενική Δομή



□□□□□α□□α □□□σε□□ π□□τ□π□□ □□□α□ατ□□□τ□□□□

Το πρότυπο παρατηρητής είναι επίσης γνωστό ως πρότυπο δημοσίευση-Εγγραφή(Publish-Subscribe).

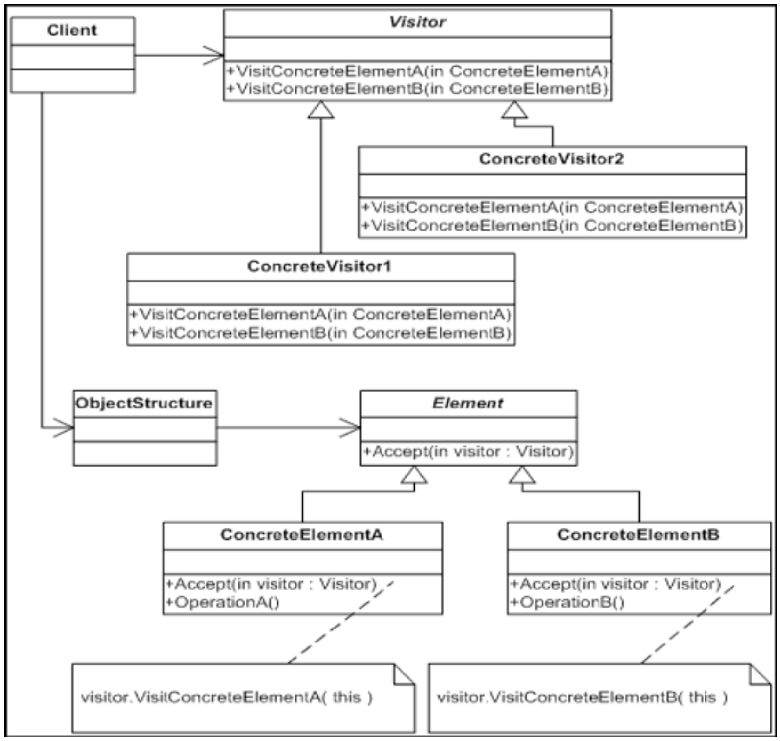
2.3.4 Πρότυπο Επισκέπτης (Visitor Pattern)

Το πρότυπο σχεδίασης ΄΄Επισκέπτης΄΄ έχει στόχο την αναπαράσταση μιας λειτουργίας που πρόκειται να πραγματοποιηθεί στα στοιχεία μιας δομής αντικειμένων. Το πρότυπο επιτρέπει την προσθήκη μιας νέας λειτουργίας στις υπάρχουσες κλάσεις του συστήματος χωρίς όμως την τροποποίηση του κώδικα στις κλάσεις στις οποίες επιδρά. Έτσι νέες λειτουργικότητες μπορούν να προστεθούν εύκολα στην αρχική ιεραρχία, χωρίς την τροποποίηση των ιδίων των κλάσεων, μονό με τη δημιουργία μιας νέας υποκλάσης στο πρότυπο ΄΄Επισκέπτης΄΄.

Το πρότυπο ΄΄Επισκέπτης΄΄ έχει αρκετά μεγάλη πολυπλοκότητα στη λειτουργία του καθώς υλοποιεί τη λεγόμενη ΄΄ διπλή αποστολή ΄΄ (double ή dual dispatch). Τα μηνύματα στον αντικειμενοστραφή προγραμματισμό κατά κανόνα επιδεικνύουν συμπεριφορά που αντιστοιχεί στην ‘απλή αποστολή’ δηλαδή ότι η λειτουργία που εκτελείται εξαρτάται από το όνομα της αίτησης και τον τύπο του αποδεκτή. Στη διπλή αποστολή που αποβλέπει το πρότυπο ΄΄Επισκέπτης΄΄ η λειτουργία που εκτελείται εξαρτάται από το όνομα της αίτησης και τον τύπο δύο αποδεκτών (από τον τύπο του ΄΄Επισκέπτη΄΄ και από τον τύπο του στοιχείου που επισκέπτεται).

Η χρήση του προτύπου σχεδίασης ΄΄Επισκέπτης΄΄ είναι ωφέλιμη κατά την προσθήκη λειτουργιών στα αντικείμενα μιας ιεραρχίας χωρίς να προκαλέσει ‘’μόλυνση’’ στις κλάσεις τους. Το πρότυπο σχεδίασης ΄΄Επισκέπτης΄΄ επιτρέπει να διατηρούνται οι σχετιζόμενες λειτουργίες σε ένα μέρος ορίζοντας αυτές σε μια νέα κλάση.

Γενική Δομή



□□□□□α□□α □□□σε□□ π□□τ□π□□ "□π□σ□□πτ□□"

2.3.5 Πρότυπο Μέθοδος Υπόδειγμα (Template Method)

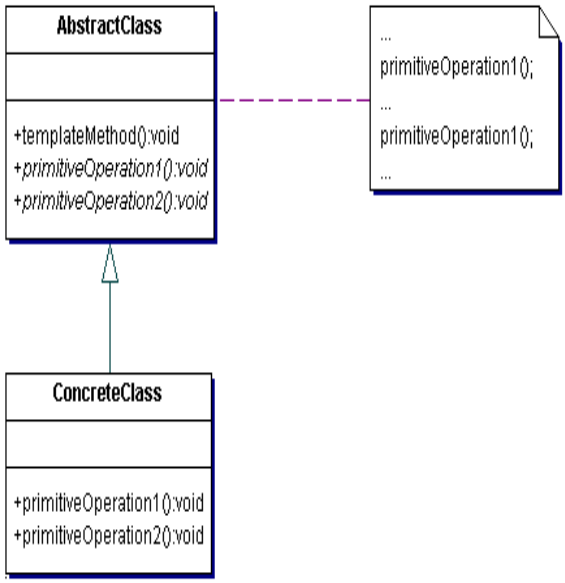
Το πρότυπο σχεδίασης ΄΄Μέθοδος Υπόδειγμα΄΄ ορίζει το περίγραμμα ενός αλγορίθμου σε μια λειτουργία, αφήνοντας ορισμένα βήματα στις παραγωγές κλάσεις. Το πρότυπο επιτρέπει στις παραγωγές κλάσεις να επαναπροσδιορίσουν ορισμένα βήματα του αλγορίθμου χωρίς να αλλάξουν τη δομή του.

Είναι ένα πολυχρησιμοποιημένο πρότυπο. Εφαρμόζεται συχνά από τους σχεδιαστές αντικειμεστρεφους λογισμικού, ακόμα και αν δεν γίνεται αντιληπτό ως ξεχωριστή τεχνική. Κλασικότερο παράδειγμα εφαρμογής του προτύπου είναι στους αλγορίθμους ταξινόμησης. Στόχος του προτύπου είναι ο διαχωρισμός ενός γενικού αλγορίθμου από συγκεκριμένες υλοποιήσεις, καθώς και η πλήρης εκμετάλλευση του μηχανισμού της κληρονομικότητας.

Το πρότυπο αυτό είναι εκλέπτυνση του σχεδιαστικού προτύπου ΄΄Στρατηγική΄΄, 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 αφηγημένες προστατευμένες μεθόδους. Οι μέθοδοι αυτές υλοποιούνται διαφορετικά σε κάθε παραλλαγή που κληρονομεί το Β.

Το σχεδιαστικό πρότυπο ΄΄Μέθοδος Υπόδειγμα ΄΄ χρησιμοποιείται για τον ορισμό των αμεταβλήτων τμημάτων και τη μετάθεση της υλοποίησης των μεταβλητών τμημάτων του αλγορίθμου σε παραγωγές κλάσης.

Γενική Δομή



□□□□□α□□α □□□σ□□ τ□□ π□□τ□π□□ □□□□δ□□ □π□δε□□□α

2.3.6 Πρότυπο΄΄Στρατηγική΄΄ (Strategy Pattern)

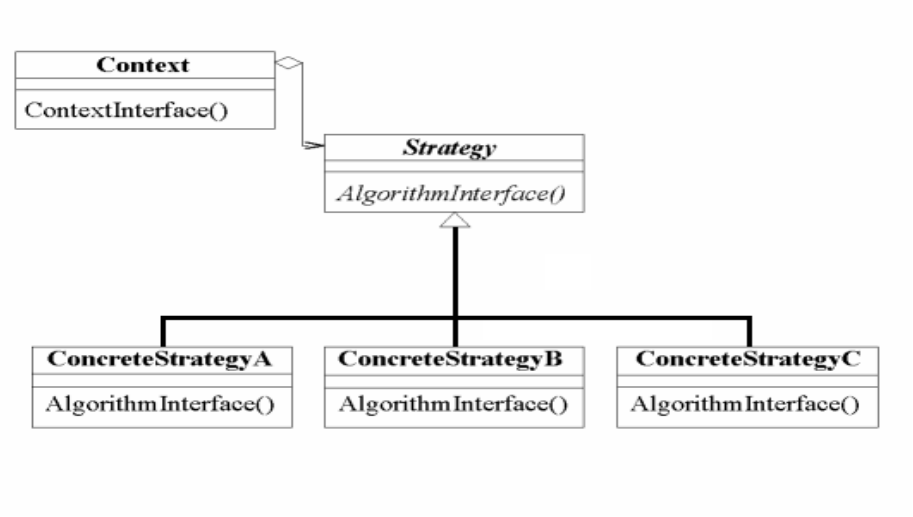
Το πρότυπο σχεδίασης ΄΄Στρατηγική΄΄ ορίζει μια οικογένεια αλγορίθμων, τους ενσωματώνει και επιτρέπει την εναλλαγή μεταξύ αυτών. Το πρότυπο δίνει τη

δυνατότητα μεταβολής των αλγορίθμων, ανεξάρτητα από τους πελάτες που τους χρησιμοποιούν.

Το πρότυπο σχεδίασης ΄΄Στρατηγική΄΄ σχετίζεται άμεσα με την αναγκαιότητα συχνής τροποποίησης του λογισμικού και είναι ίσως το πλέων ευρέως χρησιμοποιημένο, ανεξαρτήτως του αν η εφαρμογή του γίνεται αντιληπτή από τον σχεδιαστή. Το πρότυπο έχει την ικανότητα να αναγνωρίζει ότι υπάρχει στην εφαρμογή μια γενική φιλοσοφία-στρατηγική που είναι κοινή σε πολλές περιπτώσεις, η συγκεκριμένη υλοποίηση κάθε περίπτωσης είναι διαφορετική. Ουσιαστικά υπάρχει ένας κοινός γενικός αλγόριθμος με διαφορετική λεπτομερής υλοποίηση σε κάθε περίπτωση. Αυτό επιτυγχάνεται με τη χρήση της κληρονομικότητας αλλά και με τη σύνθεση αντικειμένων.

Το πρότυπο σχεδίασης ΄΄Στρατηγική΄΄, εφαρμόζει την αρχή της Αντιστροφής των Εξαρτήσεων αναγκάζοντας και τον γενικό αλγόριθμο αλλά και τις λεπτομερείς υλοποιήσεις να εξαρτώνται από τις αφαιρέσεις. Για αυτό το λόγο το πρότυπο ΄΄Στρατηγική΄΄ αυξάνει τον αριθμό των αντικειμένων στο σύστημα.

Γενική Δομή



□□□□□α□□α □□□σε□□ π□□τ□π□□ □□□τ□ατ□□□□□□□□□

2.3.7 Πρότυπο ΄΄Σύνθετο΄΄ (Composite Pattern)

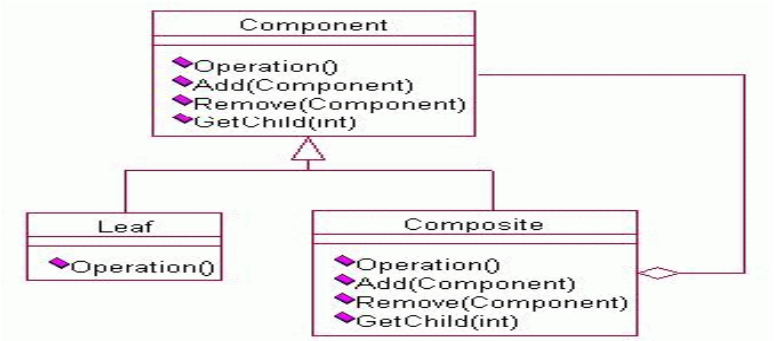
Το πρότυπο σχεδίασης ΄΄Σύνθετο΄΄ επιτρέπει τη σύνθεση αντικειμένων σε δενδροειδείς δομές για την αναπαράσταση ιεραρχιών τμήματος-όλου. Επιτρέπει στα προγράμματα πελάτες να διαχειρίζονται με ενιαίο τρόπο τόσο τα ανεξάρτητα αντικείμενα όσο και συνθέσεις αντικειμένων.

Το πρότυπο σχεδίασης ΄΄Σύνθετο΄΄ δίνει λύσεις με κομψό τρόπο σε προβλήματα χρήσης σχέσεων περιεκτικότητας μεταξύ κλάσης που αντιπροσωπεύει το όλον (περικλείουσα κλάση) και των κλάσεων που αντιπροσωπεύουν τα τμήματα.

Το σημείο κλειδί στο πρότυπο σχεδίασης ΄΄Σύνθετο΄΄ είναι η ύπαρξη μιας αφηρημένης κλάσης που αναπαριστά τόσο τις πρωταρχικές κλάσεις όσο και τις περικλείουσες κλάσεις. Έτσι, είναι πλέον δυνατή η δημιουργία οποιουδήποτε πρωταρχικού ή συνθέτου αντικειμένου επιτρέποντας ομοιόμορφο χειρισμό των αντικειμένων από ένα πρόγραμμα πελάτης.

Ο χρηστής είναι σε θέση να δημιουργήσει οποιαδήποτε συνθέτη οντότητα και να την προσθέσει στην εφαρμογή. Η σχεδίαση ενός συνθέτου αντικειμένου ουσιαστικά συνιστάται στη σχεδίαση των επιμέρους τμημάτων του. Για το λόγο αυτό, είναι επιθυμητή η ενιαία αντιμετώπιση όλων των αντικειμένων. Το πρότυπο σχεδίασης ΄΄Σύνθετο΄΄, επιτρέπει τον αναδρομικό ορισμό περιεκτικότητας, ώστε οι πελάτες να μην αντιλαμβάνονται τη διαφορά μεταξύ πρωταρχικών και συνθετών αντικειμένων.

Γενική Δομή



□□□□□α□□α □□□σε□□ π□□τ□π□□ σ□εδ□ασ□□ □□□□□□ετ□□□

2.3.8 Πρότυπο μέθοδος Εργοστάσιο (Factory Method)

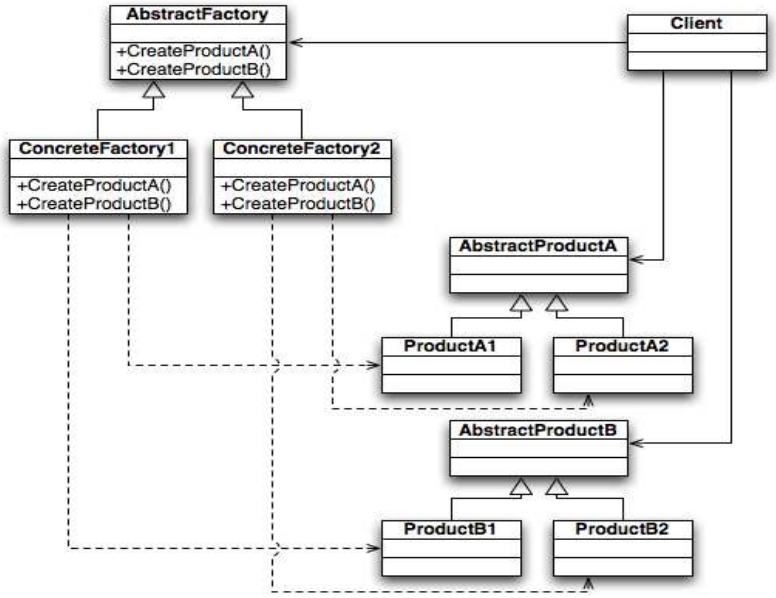
Το πρότυπο αυτό συγκεντρώνει σε μία κατάλληλη κλάση, τη Factory, όλη τη λειτουργικότητα κατασκευής στιγμιοτύπων, μίας σειράς παραγόμενων κλάσεων που κληρονομούν κάποια κοινή υπερκλάση ή υλοποιούν την ίδια διασύνδεση.

Οι μέθοδοι της κλάσης 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 η κλάση Factory είναι ανοιχτή ως προς αυτές, καθώς μόνον ο δικός της κώδικας χρειάζεται να τροποποιηθεί σε περίπτωση π.χ. προσθήκης μίας νέας παραγόμενης κλάσης.

Η κλάση Factory συνήθως παρέχει μία μέθοδο για κάθε δυνατό παραγόμενο τύπο, αλλά και μία παραλλαγή με ονόματα για παραμετροποιημένη κλάση Factory, η οποία περιέχει μία μοναδική μέθοδο που 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 δηλαδή 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.

Ένας εναλλακτικός τύπος Factory είναι όταν ορίζεται ως αφηρημένη κλάση και η παρεχόμενη μέθοδος κατασκευής αντικειμένων υποσκελίζεται από υποκλάσεις του, έτσι ώστε η καθεμία από τις τελευταίες να κατασκευάζει αντικείμενο διαφορετικού τύπου. Σε κάθε περίπτωση στόχος είναι να μπορεί το πρόγραμμα να δημιουργεί στιγμιότυπα κλάσεων χωρίς να προσδιορίζει ρητά τον ακριβή τύπο τους και αφήνοντας το Factory να τον αποφασίσει εσωτερικά· το πρόγραμμα αρκεί να έχει γνώση του γενικού αφηρημένου τύπου.

Γενική Δομή



□□□□□α□□α □□□σε□□ □□εδ□αστ□□□□ □□□τ□π□□ □□□□□□στ□σ□□□□

Υπάρχει ωστόσο και μία εναλλακτική, αρκετά διαφορετική περίπτωση όπου, αντί το Factory να δηλώνεται ως κάποια/ες ξεχωριστή/ες κλάση/εις, αποτελείται απλώς από μία ή περισσότερες στατικές μεθόδους της κλάσης της οποίας πρέπει να κατασκευάζει αντικείμενα (έστω της κλάσης A). Τότε ο κατασκευαστής (constructor) της κλάσης δηλώνεται ως ιδιωτική μέθοδος ώστε κάθε απόπειρα του εξωτερικού προγράμματος να δημιουργήσει στιγμιότυπα της A να γίνεται αναγκαστικά μέσω των μεθόδων Factory οι οποίες επιστρέφουν ένα νέο αντικείμενο τύπου A. Σε αυτήν την περίπτωση όμως η A δεν μπορεί να κληρονομηθεί, αφού για τον σκοπό αυτό πρέπει να υπάρχει τουλάχιστον ένας δημόσιος κατασκευαστής της.

2.3.9 Πρότυπο ΄΄Διακοσμιτής΄΄(Decorator Pattern)

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

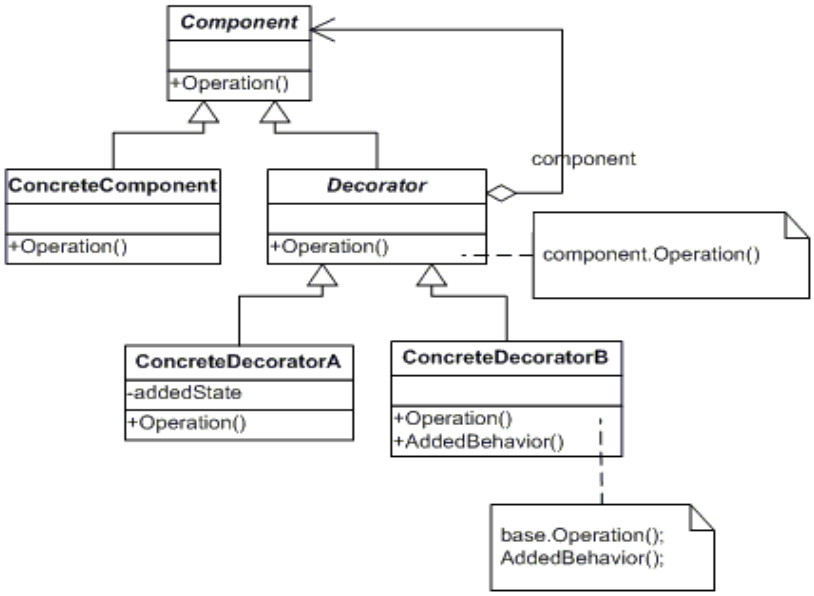
εκτέλεσης. Επισυνάπτει επιπρόσθετες ευθύνες σε αντικείμενα δυναμικά και παρέχει εναλλακτική ευελιξία σε υποκλάσεις για περαιτέρω λειτουργικότητες, χωρίς να επηρεάζονται τα αλλά αντικείμενα. Αυτό γίνεται μέσω του Decorator, μίας νέας κλάσης η οποία επίσης υλοποιεί την διεπαφή (Interface) αλλά περιέχει 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 Α είτε κάποιας άλλης, εκτεταμένης με ένα συγκεκριμένο σύνολο δυνατοτήτων. Με αυτόν τον τρόπο γίνεται εφικτός ένας δυναμικός συνδυασμός λειτουργιών από στοιχειώδεις δομικούς λίθους κατά τον χρόνο εκτέλεσης εναλλακτική λύση, χωρίς χρήση κάποιου σχεδιαστικού προτύπου, θα ήταν η απλή κληρονομικότητα, με τον ορισμό κλάσεων οι οποίες επεκτείνουν τις Α, Β κλπ. και προσθέτουν τη νέα λειτουργικότητα.

Ωστόσο η λύση αυτή δεν είναι εφικτή σε περίπτωση που οι Α, Β κλπ. δεν μπορούν να επεκταθούν με κληρονομικότητα (π.χ. αν δηλωθούν ως τελικές κλάσεις στην Java), ενώ σε άλλες περιπτώσεις δεν είναι καθόλου πρακτική, π.χ. αν έχουμε πολλαπλά διαφορετικά σύνολα νέων δυνατοτήτων τα οποία πρέπει να συνδυαστούν με τις Α, Β κλπ. Το πρόβλημα έγκειται στο ότι με την κληρονομικότητα όλοι οι πιθανοί συνδυασμοί δυνατοτήτων πρέπει να προβλεφθούν και να ληφθούν υπ' όψων κατά τη συγγραφή του προγράμματος. Αντιθέτως με την κλάση Decorator, η οποία δρα ως περίβλημα (wrapper) άλλων αντικειμένων τύπου Interface προς τα οποία περιέχει αναφορές / δείκτες, η σύνθεση νέων αντικειμένων ουσιαστικά γίνεται δυναμικά ενώ το πρόγραμμα εκτελείται.

Το πρότυπο Decorator είναι ωφέλιμο σε περιπτώσεις προέκτασης υποκλάσεων που είναι μη πρακτικές. Έτσι αποφεύγεται πιθανός πολλαπλασιασμός υποκλάσεων

Γενική Δομή



□□□□□α□□α □□□σε□□ π□□τ□π□□ □□□□α□□σ□□τ□□□□

2.3.10 Πρότυπο ΄΄Προτότυπο΄΄ (Prototype Pattern)

Το σχεδιαστικό πρότυπο ΄΄Προτότυπο΄΄ προσδιορίζει τους τύπους των αντικειμένων, δημιουργεί στιγμιότυπα χρησιμοποιώντας πρωτότυπα με σκοπό να δημιουργεί νέα αντικείμενα αντιγράφοντας αυτά τα πρότυπα.

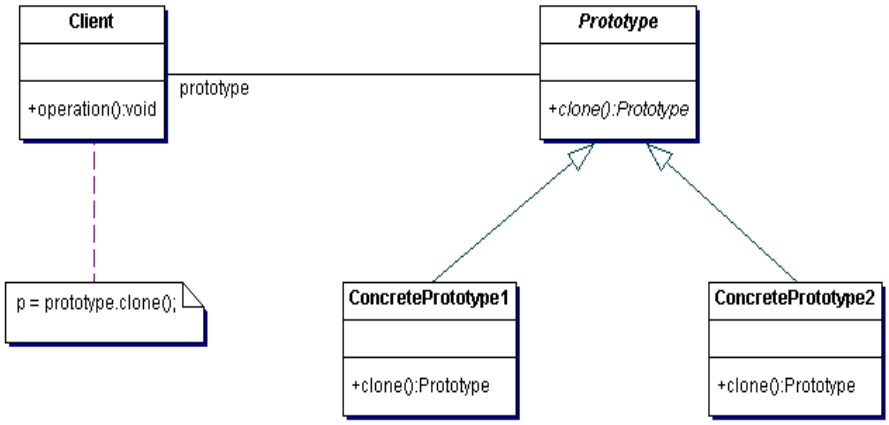
Το πρότυπο ΄΄Προτότυπο΄΄ είναι ένα σχεδιαστικό πρότυπο με το οποίο ένα νέο αντικείμενο κατασκευάζεται με "κλωνοποίηση" κάποιου υπάρχοντος. Η κλωνοποίηση αυτή γίνεται μέσω μίας μεθόδου clone η οποία παρέχεται από μία αφηρημένη κλάση ή διασύνδεση A και υλοποιείται σε κάθε παραγόμενη κλάση B η οποία κληρονομεί 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 σχεδιαστικό πρότυπο ΄΄Προτότυπο΄΄ 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.

Με τη χρήση αυτού του προτύπου πετυχαίνουμε μείωση του αριθμού των κλάσεων. Είναι πιο ωφέλιμο να εγκατασταθεί ένας αριθμός προτύπων και να δημιουργηθούν κλώνοι από το να δημιουργούνται κλάσεις χειροκίνητα κάθε φορά για συγκεκριμένη κατάσταση.

Γενική Δομή



□□□□□α□□α □□□σε□□ π□□τ□π□□ □□□□□τ□τ□π□□□

Συνέπειες χρήσης του ΄΄Προτότυπου΄΄ προτύπου είναι πολλές και αρκετά ωφέλιμες για τη λειτουργία του συστήματος όπως :

- Προσθαφαίρεση προϊόντων κατά τη διάρκεια εκτέλεσης (run-time)
- Προσδιορισμός νέων αντικειμένων με ποικιλομορφία τιμών

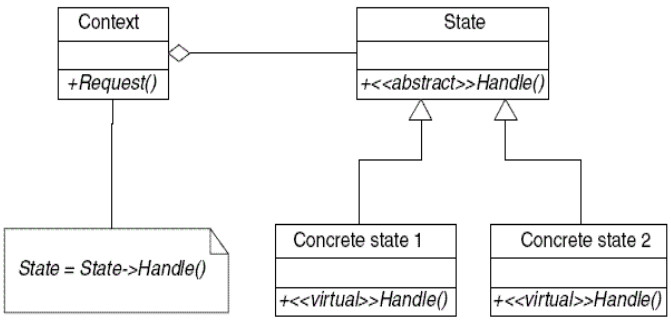
- Προσδιορισμός νέων αντικειμένων με ποικιλομορφία δομών
- Μείωση υποκλάσεων
- Φόρτωμα κλάσεων στην εφαρμογή δυναμικά

2.3.11 Πρότυπο ΄΄Κατάσταση΄΄ (State Pattern)

Ο σκοπός του προτύπου σχεδίασης ΄΄Κατάσταση΄΄ είναι να επιτρέπει σε ένα αντικείμενο να τροποποιεί τη συμπεριφορά του όταν αλλάζει η εσωτερική κατάσταση. Το αντικείμενο μοιάζει να αλλάζει κλάση.

Το πρότυπο Κατάσταση δίνει τη δυνατότητα σε ένα αντικείμενο να συμπεριφέρεται σαν να αλλάζει η κλάση του κάτι που στις περισσότερες αντικειμενοστραφείς γλώσσες προγραμματισμού είναι αδύνατο. Στο πρότυπο ΄΄Κατάσταση΄΄, η κλάση-πελάτης, περιέχει μια αφηρημένη κλάση, η οποία όμως δεν αντιπροσωπεύει μια στρατηγική αλλά μια κατάσταση. Οι παραγωγές κλάσεις υλοποιούν τις διαφορές συγκεκριμένες καταστάσεις και κατά συνεπεία η κλάση πελάτης μπορεί να εναλλάσσει την κατάστασή της αλλάζοντας την τιμή του δείκτη αναφοράς προς την περιεχόμενη κατάσταση. Σε μια παραλλαγή του προτύπου, η κάθε κατάσταση μπορεί να διατηρεί αναφορά προς την κλάση πελάτη, έτσι ώστε μετά την εκτέλεση κάποιας ενέργειας, να ενημερώνει την κατάσταση του πελάτη και να την προωθεί ενδεχομένως σε κάποια άλλη.

Γενική Δομή



□□□□□α□□α □□□σε□□ π□□τ□π□□ σ□εδ□ασ□□ □□□ατ□στασ□□□□

Το πρότυπο σχεδίασης ΄΄Κατάσταση΄΄ σχετίζεται άμεσα με την αναγκαιότητα συχνής τροποποίησης του λογισμικού και είναι ίσως το πλέων ευρέως χρησιμοποιημένο, ανεξαρτήτως του αν η εφαρμογή του γίνεται αντιληπτή από τον σχεδιαστή. Το πρότυπο έχει την ικανότητα να αναγνωρίζει ότι υπάρχει στην εφαρμογή μια γενική φιλοσοφία-κατάσταση που είναι κοινή σε πολλές περιπτώσεις, η συγκεκριμένη υλοποίηση κάθε περίπτωσης είναι διαφορετική. Ουσιαστικά υπάρχει ένας κοινός γενικός αλγόριθμος με διαφορετική λεπτομερής υλοποίηση σε κάθε περίπτωση. Αυτό επιτυγχάνεται με τη χρήση της κληρονομικότητας αλλά και με τη σύνθεση αντικειμένων.Ορισμένοι σχεδιαστές θεωρούν ότι η μοναδική διαφορά του σχεδιαστικού πρότυπο ΄΄Στρατηγική΄΄ με αυτό της ΄΄Κατάστασης΄΄ είναι η ονομασία των κλάσεων.

2.4 Πειράματα με Σχεδιαστικά Πρότυπα

Τα αντικειμενοστρεφή σχεδιαστικά πρότυπα όπως παρουσιαστήκαν από το βιβλίο του Gamma et.al. έχουν εξελιχθεί στο πιο δημοφιλές εργαλείο στην απόκτηση γνώσης για σχεδίαση ακόμα και από λιγότερο εμπείρους σχεδιαστές. Ο στόχος τους είναι να μπορούν να επαναχρησιμοποιούνται εύκολα, μειώνοντας τη σύζευξη και αυξάνοντας την ευελιξία στο σύστημα, ενώ ταυτόχρονα όρισε μια κοινή ορολογία στους σχεδιαστές και αυτούς που συντηρούν το σύστημα.

Η χρήση των σχεδιαστικών προτύπων μπορεί να επιφέρει τα επιθυμητά αποτελέσματα που προαναφερθήκαν. Από την άλλη μεριά όμως, δίνοντας παραπάνω ευελιξία στο σύστημα από όση χρειάζεται, μετατρέπεται σε πολύπλοκο και δυσνόητο. Συνεπεία αυτού είναι να μην είναι πάντα ωφέλιμη χρήση των σχεδιαστικών προτύπων αλλά μερικές φορές αρκετά επιζήμια.

Για αυτό το λόγο, έχουν γίνει ελεγχόμενα πειράματα από ερευνητικές ομάδες όπως του Vokac και του Prechelt με σκοπό να αποδειξουν ποτε τα σχεδιαστκα πρότυπα είναι ωφέλιμα για να χρησιμοποιηθούν και πότε όχι. Οι μετρήσεις που έγιναν αφορούσαν τον παρερχόμενο χρόνο (elapsed time), την ορθότητα (correctness) και την συντήρηση (maintenance). Η έρευνα του Vokac θεωρείται πιο

ολοκληρωμένη καθώς χρσιμοποιεισε παραπάνω θεματα προς εξέταση και με μεγαλύτερο αριθμό επαγγελματιών μηχανικοί λογισμικού (44 πληρωτέοι του Vokac έναντι 29 εθελοντές του Prechelt). Τα σχεδιαστικά πρότυπα που εξεταστήκαν ήταν τα: ΄΄Επισκέπτης΄΄(Visitor), ΄΄Διακοσμιτής΄΄ (Decorator), ΄΄Αφηρημένο Εργοστάσιο΄΄ (Abstract Factory), ΄΄Παρατηρητής΄΄(Observer) και ΄΄Σύνθετο΄΄(Composite) σε τέσσερα (4) διαφορετικά προγράμματα-εφαρμογές : το Stock Ticker για την εξεταση του ΄΄Παρατηρητής΄΄(Observer), το Graphics Library για την εξεταση των ΄Σύνθετο΄΄(Composite) και ΄΄Αφηρημένο Εργοστάσιο΄΄ (Abstract Factory), το Boolean Formulas για την εξεταση των ΄Σύνθετο΄΄(Composite) και ΄΄Επισκέπτης΄΄(Visitor) και το Communication Library για την εξεταση του ΄΄Διακοσμιτής΄΄ (Decorator).

Η χρήση των σχεδιαστικών προτύπων από άτομα μετά από εκπαίδευση, επέφερε υψηλότερη ορθότητα και 50% λιγότερο χρόνο στη συντήρηση. Η ποιοτική ανάλυση των λύσεων απέδειξε ότι τα πρότυπα που χρησιμοποιούν αναδρομικές δομές δεδομένων δημιούργησαν πρόβλημα σε ένα αριθμό περιπτώσεων. Χαρακτηριστικό παράδειγμα αυτού του φαινόμενου είναι το σχεδιαστικό πρότυπο ΄΄Σύνθετο΄΄. Τα πρότυπα ΄΄Παρατηρητής΄΄ και ΄΄Διακοσμιτής΄΄ αποδειχθήκαν κατανοητά ακόμα και σε άτομα με περιορισμένη γνώση προτύπων. Το πρότυπο ΄΄Επισκεπτης΄΄ το οποίο έχει αρκετά πολύπλοκη δομή παρήγαγε μεγάλο κόστος στην ανάπτυξη και φτωχά αποτελέσματα στην ορθοτητα. Το πρότυπο ΄΄Αφηρημένο Εργοστάσιο΄΄ είχε μικρή επιρροή στον απαιτούμενο χρονο, αλλά συνεισφέρει θετικά στην ποιότητα. Το πρότυπο ΄΄Διακοσμιτής΄΄ είχε θετική επίδραση παρέχοντας αξιόλογα γρηγορότερη ανάπτυξη στο σύστημα.

Περίληψη Ποιοτικών Αποτελεσμάτων		
Σχεδιαστικά Πρότυπα	**Προσδοκίες**	**Αποτελέσματα**
Observer	*Πολύπλοκες και επιβλαβείς λύσεις αν και προσφέρει ευελιξία.*	*Δεν παρουσιαστήκαν επιζήμιες επιδράσεις ακόμα και χωρίς ιδιαίτερη γνώση του προτύπου.*
Composite & Visitor	*Το πρότυπο Visitor είναι αρκετά δύσκολο για την*	*Αποφυγή χρήσης του προτύπου. Ιδιαιτέρα*

	κατανόηση του και συνεπώς επιζήμιο.	*χαμηλές τιμές στην ορθοτητα των λύσεων.*
Decorator	*Η εκτόπιση της λειτουργικότητας αναμένεται να κάνει ευκολότερα τις αλλαγές. Δυσκολία στο να τις αναλύσει και να τις καλέσει.*	*Το πρώτο μέρος της προσδοκίας επιτευχθεί. Η ορθοτητα και ο χρόνος βελτιωθήκαν.*
Composite & Abstract Factory	*Μικρές αλλαγές.*	*Δεν παρατηρηθήκαν διαφορές που να υπερίσχυσαν.*

Τα συμπεράσματα που παρήχθησαν από την ερευνα του Prechelt έχει μερικές διαφορές από την ερευνα του Vokac. Ειδικότερα στις περιπτώσεις των προτύπων ΄΄Επισκέπτης΄΄ (Visitor Pattern) και ΄΄Παρατηρητής΄΄(Observer Pattern). Η ερευνα του Prechelt απέδειξε ότι το πρότυπο ΄΄Επισκέπτης΄΄ δεν επέφερε αρκετά επιζήμιες επιδράσεις και ότι για πολλά θεματα πέτυχε ικανοποιητικές λύσεις. Όσον αφορά το πρότυπο ΄΄Παρατηρητής΄΄ η έρευνα του Vokac απέδειξε ότι δεν προκάλεσε αρκετή ζημιά κάτι που αντικρούεται από την ερευνα του Prechelt.

Το συμπέρασμα είναι ότι κάθε σχεδιαστικό πρότυπο έχει την δική του φύση και τη δική του πρέπουσα στιγμή για τη χρήση του, και δεν είναι ορθό να χαρακτηρίζεται ωφέλιμη ή επιζήμια η χρήση τους κατά κανόνα.

3. Μετρικές

Η μέτρηση λογισμικού αποτελεί πλέον μια βασική πτυχή των καλών πρακτικών σχεδιασμού και ανάπτυξης λογισμικού. Σε αυτό το κεφάλαιο αναλύονται οι βασικές μετρικές Αντικειμενοστρεφούς σχεδίασης ενώ στην ενότητα 3.2 περιγράφονται εμπειρικές μελέτες που έχουν λάβει χωρά με θέμα την επίδραση σχεδιαστικών προτύπων σε μετρικές Αντικειμενοστρεφούς σχεδίασης.

3.1 Εισαγωγή στις μετρικές Λογισμικού

Η βελτίωση αποτελεί την κινητήρια δύναμη στον τομέα της ερευνάς της τεχνολογίας λογισμικού και απευθύνεται στη βελτίωση των διεργασιών, των πόρων και των μεθόδων ώστε να παράγουμε και να συντηρούμε καλυτέρα τα λογισμικά προϊόντα. Μερικές φορές όμως εκφράζουμε τους στόχους της βελτίωσης με πολύ γενικό τρόπο, χωρίς να περιγράφουμε ποσοτικά το που βρισκόμαστε. Για αυτόν τον λόγο, η μέτρηση του λογισμικου αποτελεί πλέον μια βασική πτυχή των καλών πρακτικών σχεδιασμού και ανάπτυξης λογισμικου. Επιπλέον η ποσοτική προσέγγιση ενός συστήματος μας δίνει τη δυνατότητα της σύγκρισής του με άλλο έργο λογισμικου ώστε να παραχθούν τα αντίστοιχα συμπεράσματα.

Η μέτρηση είναι ένα αναπόσπαστο τμήμα πολλών δραστηριοτήτων της ανάπτυξης λογισμικού, παρέχοντας ποσοτική και αντικειμενική θεώρηση των δυνατοτήτων και αδυναμιών των διαδικασιών, προϊόντων και πόρων. Η μέτρηση επιτρέπει στους ερευνητές και τους επαγγελματίες να κατανοήσουν συμπεριφορές και αποτελέσματα και τους βοηθά να επιλέξουν τις καλύτερες τεχνικές και εργαλεία. Οι κατασκευαστές μετρούν τα χαρακτηριστικά του λογισμικού, για να κατανοήσουν την πρόοδο τους ως προς την ολοκλήρωση του έργου, για να διαπιστώσουν εάν ικανοποιούνται πλήρως οι απαιτήσεις των πελατών και για να εξακριβώσουν εάν ο σχεδιασμός είναι σωστός. Οι διευθυντές των έργων συμβουλεύονται τις μετρήσεις, για να διαπιστώσουν την πορεία του έργου σε σχέση με τα χρονοδιαγράμματα και τους καθορισμένους προϋπολογισμούς. Οι πελάτες κοιτούν τις μετρήσεις, για να

προσδιορίσουν την ποιότητα και λειτουργικότητα των προϊόντων ενώ οι συντηρητές, για να αποφασίσουν ποιο κομμάτι του κώδικα θα αντικαταστήσουν, ποιο θα αναδομήσουν και ποιο θα επαναχρησιμοποιήσουν.

Η μέτρηση είναι η διαδικασία από την οποία αριθμοί ή σύμβολα ορίζονται για τις ιδιότητες οντοτήτων στον πραγματικό κόσμο ώστε να περιγράφουν τέτοιες οντότητες σύμφωνα με τους σαφώς καθορισμένους κανόνες (Fenton και Pfleeger, 2004). Οι μετρήσεις είναι συνυφασμένες με την καθημερινή μας ζωή και πλέον είναι αποδεκτές από όλους. Στην ανάπτυξη λογισμικού, οι μετρήσεις διεξάγονται με τη χρησιμοποίηση των μετρικών. Μία μετρική είναι μια εμπειρική ανάθεση μιας αξίας σε μια οντότητα στοχεύοντας να περιγράψει το συγκεκριμένο χαρακτηριστικό αυτής της οντότητας.

Στην Τεχνολογία Λογισμικού οι όροι ‘μέτρηση’ (measure) και ‘μετρική’ (metric) χρησιμοποιούνται πολλές φορές ως συνώνυμες. Μέτρηση (measure) είναι η αντιστοίχηση ενός μεγέθους με μία τιμή, ενώ μετρική (metric) είναι ποσοτική μέτρηση του βαθμού στον οποίο ένα σύστημα ή τμήμα αυτού έχει ένα χαρακτηριστικό. Ο Fenton ορίζει τη μετρική ως μία εμπειρική και αντικειμενική αντιστοίχηση ενός αριθμού ή συμβόλου σε μία οντότητα με σκοπό να χαρακτηρίσει ένα συγκεκριμένο χαρακτηριστικό της. Ο Edward Bernard ορίζει τη μετρική ως μία μονάδα μέτρησης και αναφέρει ότι ο όρος χρησιμοποιείται για να δηλώσει ένα σύνολο μετρήσεων που πραγματοποιούνται πάνω σε συγκεκριμένο αντικείμενο ή διαδικασία. Προκύπτει, λοιπόν, το συμπέρασμα ότι μετρική είναι μία ‘μεθοδολογία’ μέτρησης που αντιστοιχεί μία τιμή σε κάποια προϋπάρχου ιδιότητα του αντικειμένου.

Μετρική είναι μια εμπειρική αντικειμενική αντιστοίχηση ενός αριθμού (ή συμβόλου) σε μια οντότητα με στόχο να χαρακτηρίσει ένα συγκεκριμένο χαρακτηριστικό της οντότητας αυτής. Οι μετρήσεις έχουν εισαχθεί στη διαδικασία ανάπτυξης λογισμικού προκειμένου να ικανοποιηθεί η ανάγκη ελέγχου ανάπτυξης λογισμικού και η παραγωγή ποιοτικά υψηλότερων αποτελεσμάτων. Από τα μέσα δεκαετίες του 70 όταν προτάθηκαν οι πρώτες μετρικές λογισμικού, ένας μεγάλος αριθμός μετρικών έχει προταθεί στα επόμενα έτη. Ο πολλαπλασιασμός των μετρικών ακολουθήθηκε από πρακτικότερες προτάσεις σχετικά με το πώς να ερμηνευτούν τα

αποτελέσματα από τις μετρικές, και από μεθόδους συνδυαστικών μετρικών στις μεθοδολογίες μέτρησης.

Η ποιότητα του λογισμικού είναι μία πολυσυζητημένη έννοια στις μέρες μας. Παρόλο που δεν υπάρχει ένας και μόνο ορισμός που να την περιγράφει, όλοι αντιλαμβάνονται την έννοια της ποιότητας λογισμικού, ιδιαίτερα μέσω της απουσίας της. Η διασφάλιση της ποιότητας του λογισμικού συνδέεται άμεσα με την έννοια της μετρικής, που είναι μία διαδικασία απαραίτητη για τη εκτίμηση της κατάστασης των προϊόντων, των διαδικασιών και των πόρων παραγωγής λογισμικού. Με την εφαρμογή των μετρικών σε ένα λογισμικό, μετρώντας εκείνα τα χαρακτηριστικά του που συμβάλλουν σημαντικά στην ποιότητά του, είναι δυνατό να εξαχθούν συμπεράσματα για το κατά πόσο το λογισμικό πληρεί τα κριτήρια ποιότητας. Η μέτρηση κάποιων χαρακτηριστικών του λογισμικού είναι μια διαδικασία απαραίτητη για την εκτίμηση της κατάστασης των έργων, των προϊόντων, των διαδικασιών και των πόρων παραγωγής λογισμικού. Κάθε μέτρηση, όμως, πρέπει να εξυπηρετεί μια συγκεκριμένη ανάγκη, η οποία πρέπει σαφώς κατανοηθεί και καθοριστεί. Ακόμα, ερευνητικό αντικείμενο αποτελεί η μελέτη, η χρήση και -κυρίως- η ανάπτυξη εργαλείων για τη διεξαγωγή των μετρήσεων ποιότητας λογισμικού, καθώς και η ανάλυση των αποτελεσμάτων.

Αντικείμενο των μετρήσεων λογισμικου είναι η διεξαγωγή μετρήσεων ποιότητας λογισμικού με τη χρήση μετρικών λογισμικού και η μελέτη και ανάλυση των αποτελεσμάτων. Η έρευνα επικεντρώνεται στην ανάπτυξη και εφαρμογή μετρικών λογισμικού και στη μελέτη της συσχέτισης εσωτερικών και εξωτερικών (user perceived) μετρήσεων. Η έρευνα επικεντρώνεται επίσης στη μελέτη των εσωτερικών μετρικών ποιότητας λογισμικού (κυρίως μετρικών προϊόντος) με τη χρήση μεταμετρικών ποιότητας λογισμικού.

Η μέτρηση αποσκοπεί στην κατανόηση, στον έλεγχο και στη βελτίωση του λογισμικού. Κάποιες από τις μετρήσεις αποσκοπούν στην κατανόηση των δραστηριοτήτων της ανάπτυξης και της συντήρησης του λογισμικού. Με τις μετρήσεις αυτές καθίστανται ορατές οι σχέσεις μεταξύ των δραστηριοτήτων και των οντοτήτων, βοηθώντας στην αξιολόγηση της τρέχουσας κατάστασης και στην θέσπιση στόχων για τη μελλοντική εξέλιξη.

Οι μετρήσεις επιτρέπουν τον έλεγχο των έργων και των επί μέρους προγραμμάτων. Βασικός έλεγχος στα προγράμματα είναι ο έλεγχος της πολυπλοκότητας του κώδικα, βοηθώντας στην ανακατασκευή εκείνων των τμημάτων, που υπερβαίνουν τα αποδεκτά όρια. Βασιζόμενοι στις μετρήσεις μπορούμε να βελτιώσουμε τόσο τη διαδικασία ανάπτυξης όσο και το τελικό προϊόν.

Λαμβάνοντας υπόψη το μεγάλο αριθμό μετρικών (κάνοντας μετρήσεις σχεδόν όλα), οποιαδήποτε προσπάθεια να επιλεχτεί μία μετρική χωρίς εκτενής έρευνα στις αναπτυξιακές ανάγκες του λογισμικού, τότε θα επιφέρει δευτερεύοντα οφέλη ή και καθόλου. Για να ωφεληθεί από τη χρήση των μετρικών, εκτός από την πλήρως κατανόηση των διάφορων υπαρκτών μετρικών, πρέπει επίσης να καθοριστούν καλά το γιατί θέλουμε να μετρήσουμε, τι είναι αυτό, και πότε είναι ο σωστός χρόνος να μετρηθεί. Τα αποτελέσματα μετρικών για τις μεθόδους και για την κλάση γενικά είναι πολύτιμα για την ανάλυση.

Με τις μετρήσεις ανακαλύπτονται λάθη και προβλήματα που δεν εντοπίζονται εύκολα. Επιπλέον, τεκμηριώνονται και οι αλλαγές που προκύπτουν, οι τάσεις και το μέγεθος της διορθωτικής δράσης. Είναι αναγκαίο να ελέγχεται το λογισμικό μέσω των μετρήσεων και όχι απλά να εκτελείται. Οι μετρήσεις πρέπει να εκτελούνται με συνέπεια και να είναι όσο το δυνατόν πιο αντικειμενικές. Οι δύο αυτές προϋποθέσεις είναι πολύ σημαντικές, αν και καμία μέτρηση δεν μπορεί να θεωρηθεί πραγματικά αντικειμενική, γιατί πάντα θα υπάρχει κάποιος βαθμός υποκειμενικότητας σχετικά με τις οντότητες και τα χαρακτηριστικά. Άλλωστε, πολλά ενδιαφέροντα χαρακτηριστικά στην μηχανική λογισμικού δεν είναι απευθείας μετρήσιμα, οπότε χρησιμοποιείται η έμμεση μέτρηση που εμπεριέχει στοιχεία υποκειμενικότητας.

Οι μετρικές χωρίζονται σε:

- μετρικές προϊόντος (product metrics)
- μετρικές έργου (project metrics)
- μετρικές διαδικασίας (process metrics)

Οι *μετρικές προϊόντος* σχετίζονται με το προϊόν, για παράδειγμα τον πηγαίο κώδικα ή τις δηλώσεις ελέγχου. Χωρίζονται σε δύο επιπλέον κατηγορίες:

- τις εσωτερικές – αριθμός γραμμών (LOC), χρόνος εκτέλεσης, λάθη του κώδικα
- τις εξωτερικές – λειτουργικότητα (functionality), ποιότητα (quality), πολυπλοκότητα (complexity), αποτελεσματικότητα (efficiency), αξιοπιστία (reliability), συντηρησιμότητα (maintainability).

Ενδιαφέρον παρουσιάζει η σχέση μεταξύ των εσωτερικών και των εξωτερικών μετρικών και πιο συγκεκριμένα, πώς οι εσωτερικές μετρικές ερμηνεύονται σε εξωτερικές, ώστε να είναι άμεσα μετρήσιμες. Επίσης, οι εσωτερικές μετρικές είναι πιο εύκολα αυτοματοποιήσιμες από τις εξωτερικές μετρικές – και επομένως πιο οικονομικές – μιας και το να μετρώντας οι γραμμές του κώδικα είναι πολύ εύκολο σε σχέση με το να μετράται το πόσο ευχαριστημένος είναι ο χρήστης από τις υπηρεσίες που προσφέρει το λογισμικό. Ένα ακόμη πλεονέκτημα των εσωτερικών μετρικών είναι ότι έχουν μικρότερη συχνότητα λαθών, γιατί μετρούν πολύ καθορισμένα χαρακτηριστικά του λογισμικού, για παράδειγμα τον αριθμό των γραμμών. Αντίθετα, οι εξωτερικές μετρικές είναι υψηλού επιπέδου, και έτσι βρίσκονται πιο κοντά στην έννοια της ποιότητας. Επιπλέον, τα αποτελέσματά τους είναι άμεσα ερμηνεύσιμα.

Οι *μετρικές έργου* χρησιμοποιούνται για τον καθορισμό στρατηγικής, σχετίζονται με την τακτική εκτέλεσης του έργου και καθορίζουν τη ροή του και τις τεχνικές που θα ακολουθηθούν.

Οι *μετρικές διαδικασίας* μετρούν τη διαδικασία κατασκευής ενός προϊόντος, για παράδειγμα το σχεδιασμό, τη συγγραφή κώδικα, τους απαιτούμενους πόρους.

Λόγω του ότι οι αρχές του αντικειμενοστρεφούς προγραμματισμού είναι διαφορετικές σε σχέση με αυτές του συναρτησιακού, οι μετρικές που ασχολούνται με το αντικειμενοστρεφές λογισμικό θα πρέπει να ακολουθούν τις αρχές αυτές, αν και ορισμένες από τις ‘αντικειμενοστρεφείς’ μετρικές αποτελούν μία παραλλαγή των ‘μη-αντικειμενοστρεφών’. Οι ‘αντικειμενοστρεφείς’ μετρικές εστιάζουν στα σημεία εκείνα του πηγαίου κώδικα που ακολουθούν τις αρχές του αντικειμενοστρεφούς προγραμματισμού. Αφ' ετέρου, άλλες αντικειμενοστρεφείς μετρικές έχουν αναπτυχθεί συγκεκριμένα για τον αντικειμενοστρεφή προγραμματισμό και θα ήταν άσκοπο να τους εφαρμόσει στο δομημένο προγραμματισμό. Οι μετρικές που παρουσιάζονται και που αξιολογούνται είναι και για αντικειμενοστρεφείς μετρικές και μετρικές που

προτείνονται για το δομημένο προγραμματισμό που θα μπορούσε επίσης να εφαρμοστεί αντικειμενοστραφής προγραμματισμός.

Συμπερασματικά, οι μετρήσεις βοηθούν στην παραγωγή ποσοτικών περιγραφών, μέσω των οποίων καθίστανται κατανοητές οι συμπεριφορές και τα αποτελέσματα. Η χρήση των βοηθά στην επιλογή καλύτερων τεχνικών και εργαλείων, για τον έλεγχο και τη βελτίωση των προϊόντων, των διαδικασιών και των πόρων. Για αυτούς τους λόγους, οι μετρήσεις έγιναν αναπόσπαστο κομμάτι στην ποιοτική διαδικασία ανάπτυξης του λογισμικού

3.1.1 Coupling between Object Classes (CBO)

Η μετρική '*Σύζευξη μεταξύ Αντικειμένων Κλάσεων*' είναι μια αρίθμηση των κλάσεων με τις οποίες μια κλάση συνδέεται ή συσχετίζεται εκτός κληρονομικότητας. Είναι μέτρηση του βαθμού διασύνδεσης, γνώσης ή εξάρτησης με άλλα αντικείμενα. Μετριέται με τον υπολογισμό του αριθμού διακριτών μη κληρονομούμενων σχετικών κλάσεων από τις οποίες μια κλάση εξαρτάται. Δύο κλάσεις είναι συζευγμένες όταν μέθοδοι που οριστήκαν σε μια κλάση χρησιμοποιούν μεθόδους ή στιγμιότυπα μεταβλητών που έχουν οριστεί σε άλλη κλάση. Η μέτρηση της σύζευξης είναι χρήσιμη για να καθορίσει πόσο σύνθετη είναι η δομή του συστήματος μας.

Υπάρχουν πολλοί τρόποι εξάρτησης των συστατικών μεταξύ τους, επομένως η σύζευξη εξαρτάται από διαφορά πράγματα:

- Τις αναφορές που γίνονται από ένα συστατικό στα υπόλοιπα. Για παράδειγμα το συστατικό Α μπορεί να καλεί το συστατικό Β, επομένως το συστατικό Α εξαρτάται από το συστατικό Β όσον αφορά την ολοκλήρωση της λειτουργίας της διεργασίας.
- Την ποσότητα των δεδομένων που περνούν από ένα συστατικό στα υπόλοιπα. Για παράδειγμα, το συστατικό Α μπορεί να περνά μια παράμετρο, τα περιεχόμενα ενός πίνακα, ή ένα μπλοκ δεδομένων, στο συστατικό Β.
- Την ποσότητα ελέγχου που ένα συστατικό εξασκεί επί των άλλων. Για παράδειγμα , το συστατικό Α μπορεί να περνά μια σημαία ελέγχου στο στοιχειό Β. Η τιμή αυτής της σημαίας προσδιορίζει στο συστατικό Β την κατάσταση

κάποιου πόρου ή υποσυστήματος, την διεργασία που πρέπει να καλέσει ή το αν πρέπει να καλέσει τελικά κάποια διεργασία.

- Το βαθμό πολυπλοκότητας στη διασύνδεση ανάμεσα στα συστατικά. Για παράδειγμα, το συστατικό Α περνά μια παράμετρο στο συστατικό Β για να το καθοδηγήσει στην εκτέλεσή του.

Στην πραγματικότητα, είναι απίθανο ένα σύστημα να κατασκευαστεί με βάση αποκλειστικά μη συζευγμένα συστατικά. Στόχος δεν είναι η πλήρης ανεξαρτησία, αλλά είναι η διατήρηση ενός βαθμού σύζευξης όσο το δυνατόν σε χαμηλότερα επίπεδα. Αν η σύζευξη είναι χαλαρή τότε μονό ελάχιστα συστατικά θα επηρεαστούν από την αλλαγή, αν όμως συμβαίνει το αντίθετο τότε μεγάλα τμήματα του συστήματος θα διαταραχθούν από την αλλαγή.

Μερικά από τα προβλήματα αυξημένου βαθμού σύζευξης είναι:

- Οι τοπικές αλλαγές επιφέρουν επιπτώσεις (αλλαγές) σε σχετιζόμενες κλάσεις
- Δυσκολία κατανόησης μεμονωμένα
- Δυσκολότερη επαναχρησιμοποίηση

Η υπερβολική σύζευξη μεταξύ των αντικειμένων είναι καταστρεπτική και αποτρέπει την επαναχρησιμοποίηση. Όσο πιο ανεξάρτητη κλάση είναι, τόσο ευκολότερο είναι επαναχρησιμοποιηθεί σε μια άλλη εφαρμογή. Προκειμένου να βελτιωθεί η συναρμολογισημότητα και να προωθηθεί η ενθυλάκωση, τα εσωτερικά αντικείμενα ζευγών κλάσης πρέπει να περιοριστούν στο ελάχιστο.Όσο μεγαλύτερος ο αριθμός ζευγών, τόσο ψηλότερη η ευαισθησία στις αλλαγές σε άλλα μέρη του σχεδίου, και επομένως συντήρηση είναι δυσκολότερη. Όσο υψηλότερη η σύζευξη κλάσης διά-αντικειμένου, τόσο αυστηρότερες είναι οι εξεταστικές ανάγκες. Υψηλότερο CBO υποδηλώνει αδυναμία κατανόησης κλάσεων, καθώς επίσης μειώνεται η επαναχρησιμοποίηση και αυξάνεται η συχνότητα συντήρησης. Η Ελάχιστη Σύζευξη είναι αρχή αξιολόγησης καλής σχεδίασης λογισμικού.

Η τιμή της μετρικής '*Σύζευξη μεταξύ Αντικειμένων Κλάσεων'* για τις διαδικασίες δεν πρέπει να υπερβεί 10. Για να μειωθεί η τιμή της μετρικής σε μια μέθοδο, σπάει η μέθοδος σε διάφορες χωριστές μεθόδους. Η τιμή της μετρικής για τις κλάσεις δεν πρέπει να υπερβαίνει το 30. Για να μειωθεί η μετρική αξία για μια κλάση, προτείνεται το 'σπάσιμο' της κλάσης σε διάφορες υποκλάσεις.

3.1.2 Cyclomatic Complexity (CC)

Η 'Κυκλωματική Πολυπλοκότητα' προτάθηκε από τον McCabe το 1976. Αναπαριστά των αριθμό των πιθανών μονοπατιών μέσα στον αλγόριθμο, μετρώντας τον αριθμό των διακριτών περιοχών στον γραφώ ροής, δηλαδή τον αριθμό των if, for και while βροχών στο λειτουργικό σώμα αριθμός των εκτελέσιμων μονοπατιών σε μια διεργασία (πχ. σε τμήμα κώδικα, μέσω περιπτώσεων χρήσης) δίνεται από τον τύπο:

Cyclomatic Complexity : $\mathrm{M} = \mathrm{E} - \mathrm{N} + 2\mathrm{P}$

όπου :

- *E, είναι ο αριθμός των ακμών γράφου,*
- *N, είναι ο αριθμός των κόμβων γράφου*
- *P, είναι ο αριθμός των συνδεδεμένων τμημάτων.*

Η θεωρία αυτή αναπτύχθηκε από τον Thomas J. McCabe το 1976. Η κυκλωματική πολυπλοκότητα υπολογίζεται κανονικά με τη δημιουργία μιας γραφικής παράστασης του πηγαίου κώδικα με κάθε γραμμή πηγαίου κώδικα που είναι ένας κόμβος στη γραφική παράσταση και βέλη μεταξύ των κόμβων που παρουσιάζουν διαβάσεις εκτέλεσης.

Γενικά, προκειμένου να εξεταστεί πλήρως μια ενότητα, όλες οι πορείες εκτέλεσης μέσω της ενότητας πρέπει να επεξεργασθούν. Αυτό υπονοεί ότι μια ενότητα με έναν υψηλό αριθμό πολυπλοκότητας απαιτεί περισσότερη εξεταστική προσπάθεια από μια ενότητα με μια χαμηλότερο αριθμο δεδομένου ότι ο υψηλότερος αριθμός πολυπλοκότητας δείχνει περισσότερες διαβάσεις μέσω του κώδικα. Αυτό επίσης υπονοεί ότι μια ενότητα με την υψηλότερη πολυπλοκότητα είναι δυσκολότερο για έναν προγραμματιστή να καταλάβει δεδομένου ότι ο προγραμματιστής πρέπει να καταλάβει τις διαφορετικές διαβάσεις και τα αποτελέσματα των διαβάσεων.

Επίσης, κάτι που είναι αναμενόμενο είναι ότι μια ενότητα με υψηλή πολυπλοκότητα τείνει να έχει και χαμηλότερη συνοχή (λιγότερο από τη λειτουργική

συνοχή) από μια ενότητα με τη χαμηλότερη πολυπλοκότητα. Ο πιθανός συσχετισμός μεταξύ του υψηλότερου μέτρου πολυπλοκότητας με ένα χαμηλότερο επίπεδο συνοχής βεβαιώνεται σε μια ενότητα με περισσότερα σημεία απόφασης. Εντούτοις υπάρχουν ορισμένοι τύποι ενοτήτων που κάποιος θα ανέμενε να έχει έναν υψηλό αριθμό πολυπλοκότητας, όπως οι ενότητες επαφών-χρήστη (User Interface) καθώς περιέχουν τον πηγαίο κώδικα για την επικύρωση στοιχείων και την αποκατάσταση λάθους.

Μόνο η κυκλωματική πολυπλοκότητα των μεθόδων κλάσης πρέπει να εξεταστεί για την ανάλυση. Η τιμή της πολυπλοκότητας δεν πρέπει να υπερβεί το 10. Για τις μεθόδους με μια υψηλότερη μετρική αξία, είναι σωστό να χωρίζονται σε διάφορες χωριστές μεθόδους. Η συνολική κυκλωματική πολυπλοκότητα μπορεί να χρησιμοποιηθεί για να αναλύσει τις αλλαγές σε μια κλάση.

3.1.3 CF - Coupling Factor (CF)

Η μετρική παράγοντας Σύζευξης είναι από το σύνολο των μετρικών για την αντικειμενοστρεφή ανάπτυξη MOOD (Metrics for Object-Oriented Development). Υπολογίζεται ως κλάσμα. Ο αριθμητής αντιπροσωπεύει τον αριθμό μη-κληρονομικών συζεύξεων . Ο παρονομαστής είναι ο μέγιστος πιθανός αριθμός συζεύξεων σε ένα σύστημα.

Συνέπειες αυτής της μετρικής:

- Η ικανότητα επαχρησιμοποίησης επηρεάζεται αρνητικά από τις συζεύξεις
- Σύστημα με υψηλή σύζευξη δυσκολεύει την κατανοησιμότητα, οπότε η κατανοησιμότητα μειώνεται με την αύξηση του Coupling Factor.
- Η συντηρισημότητα μειώνεται όσο αυξάνεται το Coupling Factor.
- Η δυνατότητα αλλαγών μειώνεται όσο αυξάνεται το Coupling Factor.
- Η ασφάλεια η διαλειτουργικότητα η προσαρμοστικότητα και η αξιοπιστία μειώνεται όσο αυξάνεται το Coupling Factor.

Η μετρική υπολογίζεται για το πρόγραμμα συνολικά, και είναι ενδιαφέρον για την ανάλυση των αλλαγών προγράμματος και τη σύγκριση των προγραμμάτων. Στόχος των σχεδιαστών είναι χαμηλότερη πιθανή τιμή για αυτή τη μετρική (όπως κοντά σε 0% όπως πιθανό).

3.1.4 Comments Ratio(CR)

Η Αναλογία Σχολίων είναι μια μετρική που υπολογίζει το ποσοστό των Σχολίων στο αρχείο. Φανερώνει το κατά ποσό ο συντάκτης του κώδικα θέλει να σχολιάσει τη μεθοδολογία που ακολουθεί στην συγγραφή του .

Ο κατάλληλος αριθμός σχολίων εξαρτάται από την πολιτική του προγραμματιστή που συντάσσει τον κώδικα. Τα σχόλια βελτιώνουν την αναγνωσιμότητα του προγράμματος και απλοποιούν τις αλλαγές. Συστήνεται να παρέχονται τουλάχιστον 5% σχόλια σχετικά με το συνολικό ποσό των γραμμών κώδικα και σχολίου. Η ανώτερη τιμή είναι απεριόριστη.

3.1.5 Lines of Code (LOC)

Η μετρική 'Γραμμές Κώδικα' υπολογίζει το μέγεθος ενός προγράμματος λογισμικού μετρώντας τις γραμμές του κώδικα χρησιμοποιούνται στον πηγαίο κώδικα του προγράμματος (SLOC-source lines of code).

Χρησιμοποιείται για να προβλέψει το ποσό προσπάθειας που θα απαιτηθεί για να αναπτυχθεί ένα πρόγραμμα, καθώς επίσης και για να υπολογίσει την παραγωγικότητα προγραμματισμού ή της προσπάθειας αφού παραχθεί το λογισμικό. Ένα αποδοτικό σχέδιο παρέχει λειτουργικότητα με χαμηλότερη προσπάθεια εφαρμογής και λιγότερα LOCs. Αποκλίσεις ενός μεγέθους μπορεί να είναι σαφείς δείκτες της πολυπλοκότητας λογισμικού ή των ανθρωποωρών.

Είναι ορθό να διαχωρίζουμε τον τύπο και την λειτουργικότητα των γραμμών σε κενές γραμμές, γραμμές σχολίων, δηλώσεις στοιχείων ή άλλες εντολές, γραμμές που περιέχουν διάφορες χωριστές οδηγίες, γραμμές κώδικα δημιουργημένες από ένα εργαλείο.

Υπάρχουν δύο σημαντικοί τύποι μετρικών LOC: φυσικό LOC και λογικό LOC. Οι συγκεκριμένοι ορισμοί αυτών των δύο μετρικών ποικίλλουν, αλλά ο πιο κοινός καθορισμός φυσικού LOC είναι η αρίθμηση των γραμμών στο κείμενο του πηγαίου κώδικα του προγράμματος συμπεριλαμβανομένων των γραμμών σχολίου. Οι κενές γραμμές συμπεριλαμβάνονται επίσης. Οι μετρικές του λογικού LOC προσπαθούν να μετρήσουν τον αριθμό "δηλώσεων :" , αλλά σε συγκεκριμένες γλώσσες υπολογιστών όπως η C γλωσσά προγραμματισμού. Είναι πολύ ευκολότερο να δημιουργηθούν τα εργαλεία που μετρούν φυσικό LOC, όπως Επίσης και οι φυσικοί ορισμοί LOC είναι ευκολότεροι να εξηγήσουν. Εντούτοις, τα φυσικά μέτρα LOC είναι ευαίσθητα στις λογικά άσχετες συμβάσεις μορφοποίησης και ύφους.

Οι κατηγορίες αυτής της μετρικής είναι οι εξής:

- *LOC* (Lines of Code): Συνολικός αριθμός γραμμών στο αρχείο πηγαίου κώδικα
- *NCLOC* (Non-Commented Lines of Code): Συνολικός αριθμός γραμμών στο αρχείο του πηγαίου κώδικα που δεν είναι τεκμηρίωση (δηλ. είναι κείμενο του προγράμματος – εφαρμογής)
- *CLOC* (Commented Lines of Code): Συνολικός αριθμός γραμμών στο αρχείο του πηγαίου κώδικα που έχουν να κάνουν με σχόλια - τεκμηρίωση
- *CD* (Comment Density): Ο λόγος που ποσοτικοποιεί τον όγκο των σχολίων σε σχέση με το μέγεθος του αρχείου του πηγαίου κώδικα
- *ES* (Executable Statements): Αριθμός εκτελέσιμων εντολών στο αρχείο του πηγαίου κώδικα – εξαιρούνται οι headers, και τα οι ορισμοί δομών δεδομένων
- *DSI* (Delivered Source Instructions): Ο αριθμός των εντολών που τελικά θα παραδοθούν / παραδόθηκαν στον πελάτη / χρήστη (π.χ. Δεν περιλαμβάνει κώδικα που γράφτηκε για έλεγχο, πρωτότυπα κλπ.)

Οπότε οι μετρικές είναι:

LOC = NCLOC + CLOC
CD = CLOC/LOC

Εάν το μήκος μιας κλάσης υπερβαίνει 1000 γραμμές (μην λαμβάνοντας υπόψη τα σχόλια και τις κενές γραμμές), είναι καλό να εξετάζεται η δυνατότητα διάσπασης της κλάσης σε υποκλάσεις.

Για την ανάλυση του μήκους κλάσης ανεξάρτητα των μεμονωμένων μορφών σχεδίασης λογισμικού των υπεύθυνων για την ανάπτυξή του, προτείνεται η χρήση της μετρικής HPLen. Εντούτοις, αυτό θα επιβραδύνει σημαντικά την ταχύτητα του υπολογισμού μετρικών.

Η μετρική αξία για το πρόγραμμα συνολικά μπορεί να χρησιμοποιηθεί και για την ανάλυση των αλλαγών στο πρόγραμμα, και για τη σύγκριση των διαφορετικών προγραμμάτων. Δεν είναι ενδεδειγμένο να προσπαθήσει αμέσως να ξανασχεδιάσει τις μεγάλες κλάσεις. Σε μερικές περιπτώσεις το μεγάλο μέγεθος της κλάσης δικαιολογείται.

3.1.6 LOCOM - Lack Of Cohesion Of Methods

Μια από τις πιο κλασικές μετρικές είναι η «έλειψη συνεκτικότητας μεταξύ των Μεθόδων» (Lack Of Cohesion Of Methods) που προτάθηκε από τους Chidamber και Kemmerer (Chidamber 1994). Σύμφωνα με αυτή, κάθε μέθοδος σε μια κλάση θεωρείται ότι προσπελαύνει ένα ή περισσότερα κοινά μέλη δεδομένων. Δύο μέθοδοι είναι συνεκτικές εάν τα σύνολα των μελών δεδομένων που χρησιμοποιούν έχουν κοινά στοιχεία.

Η συνεκτικότητα (cohesion) αναφέρεται στο βαθμό ¨εσωτερικής¨ λειτουργικής συνάφειας μεταξύ των τμημάτων ενός συστατικού. Όσο πιο συνεκτικό είναι ένα συστατικό, τόσο πιο πολύ τα επιμέρους τμήματά του σχετίζονται μεταξύ και συνεργάζονται για την επίτευξη ενός κοινού σκοπού. Ένα συστατικό είναι συνεκτικό αν όλα τα τμήματά του προορίζονται και είναι απαραίτητα για την εκτέλεση της ιδίας διεργασίας. Μια συνεκτική κλάση θα τείνει να παρέχει έναν υψηλό βαθμό ενθυλάκωσης, ενώ μια έλλειψη συνοχής μειώνει την ενθυλάκωση και αυξάνει την πολυπλοκότητα. Για μια κλάση η μετρική ορίζεται ως το πλήθος των μη επικαλυπτόμενων τοπικών μεθόδων μιας κλάσης.

Αν σε μια μονάδα λογισμικού τοποθετηθούν άσχετες λειτουργίες, τότε η συνεκτικότητα είναι χαμηλή. Επιπροσθέτως χαμηλή συνεκτικότητα αυξάνει την πολυπλοκότητα, καθώς επίσης υπονοεί ότι οι κλάσεις θα μπορούσαν να είχαν διαιρεθεί σε δύο ή περισσότερες κλάσεις. Η ιδανική περίπτωση είναι όταν μια μονάδα περιλαμβάνει πλήρως συσχετισμένα τμήματα που όλα είναι απαραίτητα για την εκτέλεση της ίδιας μοναδικής λειτουργίας.

Συμβολίζουμε με {I,j} το σύνολο των μελών των δεδομένων που χρησιμοποιείται από τη μέθοδο M_I της κλάσης. Καθώς κάθε κλάση C_1 περιλαμβάνει *n* μεθόδους (M_1,M_2,…,M_n), σχηματίζονται δύο σύνολα από ζεύγη μεθόδων σε κάθε κλάση:

$P=\{(M_i,M_j)\,|\,M_i\ I\ M_j=0\}$, *το σύνολο των μη συνεκτικών ζευγών μεθόδων*

$Q=\{(M_i,M_j)\,|\,M_i\ I\ M_j\neq 0\}$, *το σύνολο των συνεκτικών ζευγών μεθόδων*

Η μετρική LCOM ορίζεται ως εξής:

$$LCOM = \begin{cases} |P|-|Q|, & \text{αν } |P|>|Q| \\ 0, & \text{αν } |P|\leq|Q| \end{cases}$$

Η μετρική LCOM λειτουργεί ως εξής. Παίρνει κάθε ζευγάρι μεθόδων στην κλάση και καθορίζει το σύνολο πεδίων που έχουν πρόσβαση. Αναλύονται μόνο εκείνες οι κλάσεις για τις οποίες υφίσταται αυτή η μετρική. Αυτή η μετρική δίνει τις απόλυτες τιμές της συνοχής για μια κλάση, έτσι είναι αδύνατο να παρασχεθούν τα καθορισμένα όρια για τις αποδεκτές τιμές. Μια χαμηλή τιμή προτείνει μια φτωχή σχεδίαση κλάσης. Σε αυτήν την περίπτωση είναι πιθανώς απαραίτητο να ξανασχεδιαστεί η κλάση.

Εάν πρέπει να χωρίσουν τα σύνολα πεδίων πρόσβασης, η αρίθμηση P αυξάνεται από ένα. Εάν μοιράζονται τουλάχιστον μια πρόσβαση τομέων, το Q αυξάνεται από ένα. Μια χαμηλή τιμή δείχνει την υψηλή σύζευξη μεταξύ των μεθόδων, Αυτό δείχνει επίσης την ενδεχομένως χαμηλή ικανότητα επαναχρησιμοποίησης.

Τα μειονεκτήματα του LCOM1 είναι τα εξής :

- Δεν είναι ο πιο σωστός τρόπος να μετράμε την συνοχή καθώς μετρά την αλληλεπίδραση μεθόδων - attributes που ίσως να μην είναι και ο ποιο ενδεδειγμένος τρόπος σε έναν αντικειμενοστρεφή κόσμο.
- Πολύ διαφορετικές κλάσεις μπορούν να έχουν το ίδιο LCOM1.
- Οι κλάσεις που έχουν getters/setters για τα attributes τους παρουσιάζουν υψηλό LCOM1. Αυτό όμως δεν συνεπάγεται κακή σχεδίαση.

Υπάρχουν και δύο παραπλήσιες μετρικές της LCOM:

- **LCOM2 - Lack Of Cohesion Of Methods 2**

Μετρά το ποσοστό των μεθόδων που δεν έχουν πρόσβαση σε ένα συγκεκριμένο χαρακτηριστικό εξάγοντας το μέσο όρο όλων των χαρακτηριστικών στην κλάση. Η υψηλή αξία της συνοχής (μια χαμηλή έλλειψη συνοχής) υπονοεί ότι η κλάση σχεδιάστηκε καλά. Μια συνεκτική κλάση θα τείνει να παρέχει έναν υψηλό βαθμό ενθυλάκωσης, ενώ μια έλλειψη συνοχής μειώνει την ενθυλάκωση και αυξάνει την πολυπλοκότητα.

Αναλύονται μόνο εκείνες οι κλάσεις για τις οποίες υφίσταται αυτή η μετρική. Αποφεύγονται οι κλάσεις που έχουν μια τιμή λιγότερο από 30%. Μια χαμηλή τιμή προτείνει μια φτωχή σχεδίαση κλάσης, το οποίο απαιτεί την αυξανόμενη προσπάθεια για τη δοκιμή ή την τροποποίηση της κλάσης. Για να αυξήσει την τιμή, συστήνεται να ξανασχεδιαστεί η κλάση. Αυτή μετρική, μαζί με LOCOM1, έχει μερικά μειονεκτήματα (αντικειμενοστρεφείς μετρικές του Brian Henderson-Sellers "*Μέτρο της πολυπλοκότητας*"). Είναι δυνατό να χρησιμοποιηθεί LOCOM3 ως εναλλακτική λύση.

- **LCOM3 - Lack Of Cohesion Of Methods 3**

Ο καθορισμός αυτού της μετρικής προτάθηκε από Henderson-Sellers το 1995. Η χαμηλή αξία δείχνει την καλή υποδιαίρεση κλάσης που υπονοεί την απλότητα και

την υψηλή ικανότητα επαναχρησιμοποίησης. Η υψηλή έλλειψη συνοχής αυξάνει την πολυπλοκότητα, αυξάνοντας την πιθανότητα των λαθών κατά τη διάρκεια της διαδικασίας ανάπτυξης.

Ακολουθεί ανάλυση του τύπου LCOM3. Έστω ότι εξετάζεται ένα σύνολο m μεθόδων M_1, M_2, ... , M_m. Οι μέθοδοι έχουν πρόσβαση σε a χαρακτηριστικά, A_1, A_2, ..., A_a. Έστω a(M_k) = αριθμός χαρακτηριστικών που προσεγγίζονται με τη μέθοδο M_K και m(A_k) = αριθμός μεθόδων που έχουν πρόσβαση στα στοιχεία Ak.
Τότε:

$$LOCOM3 = \frac{1/a\sum_{i=1}^{a} m(A_i) - m}{1-m} 100$$

Αναλύονται μόνο εκείνες οι κλάσεις για τις οποίες υφίσταται αυτή η μετρική. Αποφεύγονται οι κλάσεις που έχουν μια τιμή λιγότερο από 30%. Μια υψηλή τιμή προτείνει μια φτωχή σχεδίαση κλάσης, το οποίο απαιτεί την αυξανόμενη προσπάθεια για τη δοκιμή ή την τροποποίηση της κλάσης. Για να μειωθεί η τιμή, συστήνεται να ξανασχεδιαστεί η κλάση.

Για τα LCOM2 και LCOM3 ισχύουν τα εξής:

- Οι χαμηλές τιμές δείχνουν υψηλή συνοχή και καλή σχεδίαση της κλάσης.
- Οι υψηλές τιμές υποδεικνύουν μειωμένη ενθυλάκωση και αυξημένη πολυπλοκότητα.
- Είναι παρόμοιες μετρικές με διαφορετικούς τύπους. Το σύνολο τιμών της LCOM2 είναι το [0,2] ενώ της LCOM3 το [0,1].
- Το LCOM2 είναι παρόμοιο με το LCOM3 αλλά έχει ιδανική τιμή 1.

3.1.7 Weighted Methods per Class (WMC)

Η μετρική 'Ζυγισμένος αριθμός Μεθόδων ανά κλάση' υπολογίζει το άθροισμα της πολυπλοκότητας των μεθόδων μιας κλάσης. Είναι μια αρίθμηση των μεθόδων

που εφαρμόζονται μέσα σε μια κλάση ή το ποσό των περιπλοκών των μεθόδων (cyclomatic complexity).

Θεωρούμε μία κλάση C1 με μεθόδους M1, M2, . . ., Man. Av c1, c2, . . ., can είναι η στατική πολυπλοκότητα κάθε μεθόδου και n ο αριθμός των μεθόδων στην κλάση, τότε:

$$WMC = \sum_{i=1}^{n} c_i$$

Υπάρχουν δυο υποκατηγορίες σε αυτήν τη μετρική:

- **WMPC1 - Weighted Methods Per Class 1**

Αυτή η μετρική έχει να κάνει με το ποσό της πολυπλοκότητας όλων των μεθόδων σε μια κλάση, όπου κάθε μέθοδος σταθμίζεται από την κυκλωματική πολυπλοκότητά της. Ο αριθμός μεθόδων και της πολυπλοκότητας των μεθόδων που περιλαμβάνονται είναι πρόβλεψη του πόσου χρόνου και προσπάθειας απαιτείται για να αναπτύξει και να διατηρήσει την κλάση. Μόνο οι μέθοδοι που ορίζονται σε μια κλάση συμπεριλαμβάνονται, δηλαδή όσες μέθοδοι κληρονομούνται από έναν γονέα αποκλείονται.

Τα αποτελέσματα των μετρικών και για τις μεθόδους και για την κλάση γενικά είναι πολύτιμα για την ανάλυση. Η τιμή της μετρικής για τις διαδικασίες δεν πρέπει να υπερβεί 10. Για να μειωθεί η τιμή, μπορεί να σπάσει η μέθοδος σε διάφορες μεθόδους. Η τιμή της μετρικής για τις κλάσης δεν πρέπει να υπερβεί 30, Για να μειωθεί η τιμή, προτείνεται η διάσπαση της κλάσης σε διάφορες υποκλάσεις.

- **WMPC2 - Weighted Methods Per Class 2**

Αυτή η μετρική προορίζεται για να μετρήσει την πολυπλοκότητα μιας κλάσης, υποθέτοντας ότι μια κλάση με περισσότερες μεθόδους είναι πιο σύνθετη από μια άλλη, και ότι μια μέθοδος με περισσότερες παραμέτρους από άλλη είναι επίσης πιθανό να είναι πιο σύνθετη.

Η μετρική απαριθμεί μεθόδους και παραμέτρους για μια κλάση. Μόνο οι μέθοδοι που ορίζονται σε μια κλάση συμπεριλαμβάνονται, δηλαδή οποιεσδήποτε μέθοδοι κληρονομούνται από έναν γονέα αποκλείονται. Είναι θεμιτό να αποφεύγονται οι κλάσεις με τιμή μετρικής άνω των 100. Για να μειωθεί η τιμή , προτείνεται η διαίρεση της κλάσης σε υποκλάσεις, ή ανασχεδιασμός της κλάσης.

3.1.8 Number of Classes (NOC)

Αυτή η μετρική υπολογίζει τον αριθμό των κλάσεων στο πρόγραμμα. Αποφεύγονται περισσότερες από 5 εσωτερικές κλάσεις. Για να μειωθεί η τιμή για αυτή τη μετρική, προτείνεται η μετακίνηση των εσωτερικών κλάσεων σε χωριστές κλάσεις. Τα αποτελέσματα της μετρικής μπορούν να χρησιμοποιηθούν για την εκτίμηση αλλαγών στο πρόγραμμα και για τη σύγκριση με άλλα προγράμματα.

3.1.9 Fan-out (FO)

Η μετρική Fan-out εκτιμάει την πολυπλοκότητα συντήρησης του λογισμικού. Υποδηλώνει τον αριθμό των συναρτήσεων που καλεί μια συνάρτηση. Η τροποποίηση μιας συνάρτησης μπορεί να έχει αποτελέσματα στις συναρτήσεις που καλούνται από την τροποποιημένη συνάρτηση. Ο συντηρητής της μονάδας πρέπει να καταλάβει όλες τις συναρτήσεις που καθιστούν τη συντήρηση σκληρότερη και χρονοβόρα. Επομένως, λειτουργίες με μεγάλο fan-out απαιτούν μεγάλο κόστος για συντήρηση. Επίσης, μετρά τον αριθμό των τύπων αναφορών που χρησιμοποιούνται στις δηλώσεις ιδιοτήτων, επίσημες παράμετροι, τύποι επιστροφής, ρίχνει τις δηλώσεις και τις τοπικές μεταβλητές. Οι απλοί τύποι και οι super types δεν μετρούνται. Ορθό είναι να αποφεύγονται τιμές αυτής της μετρικής άνω των 15.

3.2 Μετρικές και Πρότυπα Σχεδίασης

Οι αντικειμενοστραφείς μετρικές λογισμικού χωρίζονται σε τέσσερις βασικές κατηγορίες: μετρικές μεγέθους, πολυπλοκότητας, σύζευξης και συνοχής. Έχουν

γραφεί αρκετά άρθρα ύστερα από έρευνες που υποδεικνύουν την επίδραση των σχεδιαστικών προτύπων σε αυτές τις μετρικές. Σύμφωνα με τον [Gamma] η παρουσία των σχεδιαστικών προτύπων ενισχύει τη συντηρισιμότητα του λογισμικού, την ευελιξία και την ευκολία σε μελλοντικές προσαρμογές. Αυτή την επίδραση στην λειτουργικότητα του συστήματος προσπάθησαν να την ερμηνεύσουν με τις τιμές των μετρικών.

Ο σκοπός της μελέτης των σχεδιαστικών προτύπων σε συνύπαρξη με τις σχεδιαστικές μετρικές είναι για να αποφασιστεί η συμβατότητα τους στην βελτίωση της σχεδιαστικής ποιότητας. Αντιστρόφως, θα υπάρξουν και περιπτώσεις που η εφαρμογή του δοσμένου σχεδιαστικού προτύπου δε θα είναι η κατάλληλη και θα υπάρξει αθέτηση των μετρικών.

Ερευνώντας τις τιμές των μετρικών μπορούν να ανιχνευτούν στιγμιότυπα σχεδιαστικών προτύπων. Η πιο κατάλληλη μέθοδος για να αποφασιστεί η αρνητική ή θετική επιρροή της χρήσης των σχεδιαστικών προτύπων στα λογισμικά έργα είναι η σύγκριση της ποιότητας λογισμικών έργων υπό το πρίσμα των αντικειμενοστεφών μετρικών.

Στην ποιοτική και ποσοτική μελέτη πεδίου των [Ampatzoglou & Chatzigeorgiou] χρησιμοποιήθηκαν δυο παιχνίδια ανοιχτού κώδικα (open-source games) με σκοπό την αξιολόγηση της εξέλιξης τους σύμφωνα με τα προστιθέμενα σχεδιαστικά πρότυπα και των αντικειμενοτρεφών μετρικών.

Ο έλεγχος των μετρήσεων σύζευξης (CBO) είναι η πιο προφανής επιλογή στην μεθοδολογία της έρευνάς τους. Η χρήση των αφαιρέσεων στα σχεδιάστηκα πρότυπα πρωτίστως αποσκοπεί στην εξάλειψη των εξαρτήσεων μεταξύ των κλάσεων. Έτσι παρατηρήθηκε μείωση των τιμών της μετρικής Coupling Between Objects στις κλάσεις που εμπεριείχαν επιπρόσθετα σχεδιαστικά πρότυπα στα δυο αυτά παιχνίδια.

Επίσης, εξετασθήκαν οι μετρικές που αφορούσαν τη συνοχή του συστήματος στις νεότερες εκδόσεις. Παρατηρήθηκε μείωση των τιμών σε μετρικές Lack of Cohesion Of Methods(LCOM) και Coupling Factory (CF) και αυτό οφείλεται στην υποχρέωση των προτύπων να συμμορφώνονται στην Αρχή Μοναδικής Υπευθυνότητας (Single Responsibility Principle). Αποτέλεσμα αυτού του γεγονότος είναι η αύξηση της συνοχής των κλάσεων όταν τα πρότυπα χρησιμοποιούνται σωστά.

Επιπλέον, η εφαρμογή σχεδιαστικών προτύπων σίγουρα εμπεριέχει την εισαγωγή νέων κλάσεων και διασυνδέσεων, τη μεταφορά μεθόδων και σε πολλές περιπτώσεις απαιτείται επιπρόσθετος κώδικας. Αυτό το συμπέρασμα επιβεβαιώθηκε και με την μελέτη των [Ampatzoglou & Chatzigeorgiou] καθώς παρατηρήθηκε αύξηση στις τιμές των μετρικών Lines of Code (LOC) και Number of Classes (NOC) στις νέες εκδόσεις των παιχνιδιών.

Πολλά σχεδιαστικά πρότυπα εμπεριέχεται ο πολυμορφισμός, τα οφέλη του οποίου γίνονται φανερά στα πολύπλοκα κομμάτια κώδικα (καταστάσεις if και switch τύπου καταρράκτη) καθώς προσπαθεί να περιορίσει τέτοια κομμάτια κώδικα. Σαν αποτέλεσμα αυτού, η χρήση των σχεδιαστικών προτύπων αναμένεται να μειώσει την πολυπλοκότητα των κλάσεων που απαρτίζουν το λειτουργικό σώμα του συστήματος. Σε αυτό το συμπέρασμα κατέληξαν και οι [Ampatzoglou & Chatzigeorgiou] αφότου εξέτασαν τις μετρικές μεγέθους πολυπλοκότητας Weighted Methods Per Class (WMPC) και Cyclomatic Complexity (CC) στις δύο αυτές εφαρμογές. Παρουσίασαν μείωση των τιμών στις μετρικές των νεότερων εκδόσεων των παιχνιδιών φανερώνοντας την θετική επίδραση και επιρροή των σχεδιαστικών προτύπων.

Στο ποσοτικό πείραμα του [Huston] εξετάστηκε η επίδραση των σχεδιαστικών προτύπων στις τιμές των μετρικών σε τρεις κατηγορίες : σύζευξη, κληρονομικότητα, μέγεθος. Εφαρμοσθήκαν τα Σχεδιαστικά πρότυπα Mediator, Bridge, Visitor σε λειτουργικό σώμα χωρίς πρότυπα. Όσο αφορά την επίδραση των προτύπων στις μετρησεις της σύζευξης CBO του Chidamber και του παράγοντα σύζευξης του COF Abreu et al χρησιμοποιήθηκε το πρότυπο Mediator. Το σχεδιαστικό πρότυπο Mediator προτείνεται για να συμβάλει θετικά σε καταστάσεις οπού αλληλεπιδράσεις μεταξύ μιας ομάδας κλάσεων είναι αρκετά πολύπλοκες. Χαλαρώνει τις συζεύξεις συγκεντρώνοντας την ευθύνη της επικοινωνίας σε μια καινούρια κλάση , αποφεύγοντας τις συνεχόμενες ανάγκες των αντικειμένων για ρητή αναφορά μεταξύ τους. Έτσι καθιστά ικανή κάθε κλάση να επαναχρησιμοποιηθεί μεμονωμένα που σίγουρα αποτελεί πιο ευέλικτη κατάσταση. Παρατηρήθηκε μείωση των τιμών της CF η οποία παρόλα αυτά δεν αποδίδεται τόσο στην εφαρμογή του προτύπου Mediator . Για την κληρομικότητα χρησιμοποιήθηκε το πρότυπο Bridge. Η λειτουργία του σχεδιαστικού προτύπου Bridge συμφωνά με τον Gamma έχει ως στόχο την

αποσύνδεση μιας αφαίρεσης από την υλοποίησή της, ώστε να μπορούν να μεταβάλλονται αμετάβλητα. Αρχικά εξετάστηκε η μετρική DIT(Depth in Inheritance) που έχει προταθεί από τον Chidambaram . Η φιλοσοφία αυτής της μετρικής είναι όσο πιο βαθειά είναι η κλάση στην ιεραρχία τόσο πιο πολύ συμπεριφορά είναι πιθανό να κληρονομήσει, άρα θα είναι και πιο πολύπλοκη. Αντιστρόφως οι πολύ ρηχές δενδρικές ιεραρχίες υπονοούν ανεπαρκή επίπεδο αφαίρεσης. Οι μειωμένες τιμές της μετρικής DIT υστέρα από την παρουσία του σχεδιαστικού προτύπου Bridge Φανερώνει το γεγονός ότι η εφαρμογή του θα μειώσει τη μέση τιμή του DIT. Αυτό ήταν και το επιθυμητό Αποτέλεσμα, να μειωθεί η πολυπλοκότητα στους κόμβους-φύλλα.

Επίσης με το ίδιο σχεδιαστικό πρότυπο εξετάσθηκε η επίδραση του στη μετρική NOC (Number of children) η οποία Επίσης έχει προταθεί από τον Chidamber. Η φιλοσοφία αυτής της μετρικής είναι μεγάλη τιμή στο NOC υπονοεί μεγάλο επίπεδο επαναχρησιμοποίησης από τη στιγμή που η κληρονομικότητα είναι στοιχειό και τακτική της επαναχρησιμοποίησης. Βεβαία κλάση με μεγάλο NOC έχει μεγάλη επιρροή στο σχεδιασμό του συστήματος αυτό και είναι πιθανό να απαιτείται πιο πολύ έλεγχος. Το πρότυπο Bridge παρατηρήθηκε να προκαλεί ευκρινή μείωση της επίδρασης στο μέσο ορό της NOC και δείχνει υψηλή συσχέτιση όταν συμπεριλαμβάνονται οι κλάσεις-φύλλα στον υπολογισμό της μετρικής. Αντίθετα όταν δεν συμπεριλαμβάνονται οι κομβόι-φύλλα παρατηρήθηκε μικρή συσχέτιση και μη αποφασιστική επίδραση στο μέσο ορό του NOC. Πάντως η φιλοσοφία του προτύπου Bridge υπονοεί ότι τα φύλλα-κομβόι θα πρέπει να συμπεριλαμβάνονται.

Ο [Huston] εξέτασε την επιρροή του σχεδιαστικού προτύπου Visitor πάνω σε μια μετρική που μετρά τις μεθόδους μιας κλάσης NOM(Number of methods). Η μετρική αυτή προτάθηκε από τους [Lorenz and Kidd] είναι ισοδύναμη με την WMPC(weighted methods per class) του Chidamber. Το Σχεδιαστικό πρότυπο Visitor φαίνεται να έχει μειώνει σημαντικά τις τιμές του NOM . Έχει λιγότερη επίδραση εκεί που ο αριθμός των λειτουργιών του προτύπου Visitor είναι περιορισμένη, παρόλα αυτά όμως σε τέτοια περίπτωση μπορεί να αυξήσει το μέσο ορό του NOM.

Το συμπέρασμα της μελέτης του [Huston] είναι ότι όντως κάποιες φορές τα σχεδιαστικά πρότυπα μπορούν να ρίξουν τις υψηλές τιμές των μετρικών αλλά άλλες

φορές προκαλούν σύγχυση. Η μελέτη του ήταν συμπαγής αλλά το εύρος των σχεδιαστικών προτύπων και σχεδιαστικών μετρικών που εξετασθήκαν ήταν ανεπαρκές για να παραχθούν ολοκληρωμένα και πειστικά συμπεράσματα. Προτείνεται επέκταση σε μετρικές που παρουσιάζουν αποτελέσματα για άλλες πτυχές του πολυμορφισμού όπως η συνοχή.

4. Αξιολόγηση της επίδρασης χρήσης προτύπων σχεδίασης

Στο κεφάλαιο αυτό της πτυχιακής εργασίας γίνεται αναφορά στους τρεις τρόπους εκπόνησης μιας μελέτης, της μελέτης πεδίου, της ερευνάς πεδίου, και των τυπικών πειραμάτων συμφωνά με τη φιλοσοφία της Kitchenham. Πάνω σε αυτήν την μεθοδολογία βασίστηκε και η παρούσα μελέτη πεδίου της πτυχιακής εργασίας. Επιπροσθέτως, ακολουθεί παρουσίαση των παραγόμενων αποτελεσμάτων από τη μελέτη πεδίου με τη βοήθεια περιγραφικών στατιστικών, ελέγχου υποθέσεων, και Παραμετρικά Τεστ **(**t-test, Paired t-test) .

4.1 Εισαγωγή

Η επιστημονική έρευνα στην τεχνολογία λογισμικού περιλαμβάνει διαφορετικού τύπου ερευνητικές μεθόδους, ανάλογα με το τι πρόκειται να ερευνηθεί. Οι μέθοδοι αυτοί είναι : η *επιστημονική* (scientific method), η *μηχανική* (engineering method), η *εμπειρική* (empirical method) και η *αναλυτική μέθοδος* (analytical method).

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

Η *εμπειρική τεχνολογία λογισμικού* (empirical software engineering) είναι ο κλάδος που ερευνά θέματα της τεχνολογίας λογισμικού με τη χρήση εμπειρικών μεθόδων *[Basili]* . Η διεξαγωγή εμπειρικών μελετών αποσκοπεί στην απόκτηση αντικειμενικών και στατιστικά τεκμηριωμένων αποτελεσμάτων, όσον αφορά την κατανόηση, τον

έλεγχο, την πρόβλεψη και τη βελτίωση της ανάπτυξης λογισμικού. Στις εμπειρικές μελέτες υπάρχουν δύο τύποι παραδειγμάτων ερευνών που ακολουθούν διαφορετικές προσεγγίσεις: οι *ποιοτικές έρευνες* (qualitative research) και οι *ποσοτικές έρευνες* (quantitative research). Η ποιοτική έρευνα σχετίζεται με τη μελέτη αντικειμένων στις φυσικές τους συνθήκες. Ένας ποιοτικός ερευνητής προσπαθεί να ερμηνεύσει ένα φαινόμενο βασισμένος στις ερμηνείες, που δίνουν οι άνθρωποι. Η ποσοτική έρευνα σχετίζεται με την ποσοτικοποίηση μιας σχέσης ή με τη σύγκριση δυο ή περισσότερων συνόλων. Σκοπός της είναι η δόμηση μιας σχέσης αιτίου-αποτελέσματος. Μια ποσοτική έρευνα διεξάγεται μέσα από ελεγχόμενα πειράματα ή συλλογή δεδομένων από μελέτες περιπτώσεων και είναι κατάλληλη, όταν ελέγχονται τα αποτελέσματα μιας δραστηριότητας ή ενός χειρισμού. Ένα πλεονέκτημα της είναι πως τα ποσοτικά δεδομένα επιτρέπουν τις συγκρίσεις των αποτελεσμάτων και τη στατιστική ανάλυση. Οι ερευνητικές μέθοδοι που χρησιμοποιούνται στις εμπειρικές μελέτες είναι *[Wholin]* :

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

4.1.1 Μελέτη Πεδίου -Case Study

Η μελέτη πεδίου(Case Study) βοηθάει στη βιομηχανία να αξιολογεί τις ωφέλειες των μεθόδων και των εργαλείων που χρησιμοποιηθεί, παρέχοντας ένα κοστολογημένο τρόπο στο να επιβεβαιώνει ότι οι αλλαγές στις διαδικασίες επιφέρουν τα επιθυμητά αποτελέσματα(Kitchenham). Είναι ιδιαίτερα σημαντικές, γιατί ενσωματώνουν χαρακτηριστικά, όπως η κλιμάκωση, η πολυπλοκότητα και ο δυναμισμός, που δεν μπορούν να απεικονιστούν με τις άλλες μεθόδους.

Η μελέτη περίπτωσης είναι μία παρατηρητική μελέτη ελέγχου έργων ή δραστηριοτήτων. Διεξάγεται με σκοπό να ερευνηθεί μια συγκεκριμένη οντότητα ή ένα φαινόμενο σε ένα συγκεκριμένο χρονικό διάστημα. Σε μια μελέτη περίπτωσης πρώτα εντοπίζονται οι κύριοι παράγοντες, όπως είσοδοι, περιορισμοί, πόροι και έξοδοι, που μπορούν να επηρεάσουν το αποτέλεσμα κάποιας δραστηριότητας και στη συνέχεια

τεκμηριωνέται ο καθένας από αυτούς τους παράγοντες ξεχωριστά. Ο σκοπός μια μελέτης περιπτώσης είναι η σύγκριση μιας κατάστασης με κάποια άλλη παρόμοια, όπως για παράδειγμα η σύγκριση των αποτελεσμάτων που προκύπτουν από την χρήση μεθόδων ή εργαλείων. Αποφασίζεται εκ των προτέρων τι ακριβώς θα ερευνηθεί και έπειτα εντοπίζονται οι προς έλεγχο παράγοντες και οργανώνεται ο τρόπος συλλογής των δεδομένων. Για την αποφυγή λαθών και την διασφάλιση ότι ελέγχεται η σχέση για την οποία ετέθησαν οι υποθέσεις της έρευνας, οργανώνεται η μελέτη σύμφωνα με το *αδελφικό έργο ανάπτυξης* (sister project) ή τη *γραμμή βάσης* (baseline) ή την *τυχαία επιλογή* (random selection). Αδελφικά έργα είναι αυτά που έχουν παρόμοιο πεδίο εφαρμογής, γλώσσα υλοποίησης, τεχνική ορισμού προδιαγραφών και μέθοδο σχεδίασης. Συνήθως εκτελούνται επισκοπήσεις ως προς την κατάσταση που διερευνάται στο ένα έργο με την τρέχουσα κατάσταση στο αδελφό έργο και κατόπιν συγκρίνονται τα αποτελέσματα. Τα έργα ανάπτυξης, που επιλέγονται, πρέπει να είναι όσο το δυνατόν παρόμοια και πρέπει να ελέγχονται όσο το δυνατό περισσότεροι παράμετροι. Όταν δεν μπορούν να βρεθούν δύο παρόμοια (αδελφικά) έργα, τότε η επισκόπηση μπορεί να γίνει με μια γραμμή βάσης, δηλαδή να συλλεχθούν τα δεδομένα από διάφορα έργα που υλοποιεί η εταιρία, ανεξάρτητα από τις διαφορές που παρουσιάζουν. Από τα δεδομένα των μετρήσεων, που είναι περιγραφικού τύπου, μπορεί να υπολογισθεί η κεντρική ροπή και η διασπορά των δεδομένων, παρέχοντας μια εικόνα για τη ‘μέση’ κατάσταση των έργων. Όταν δεν υπάρχει η δυνατότητα επισκόπησης σε δύο ή περισσότερα έργα, τότε μπορεί να επιλεγεί η τυχαία επιλογή, δηλαδή να διαχωριστεί ένα έργο σε μέρη, όπου σε κάποιο από αυτά εφαρμόζεται η προς διερεύνηση κατάσταση. Η τυχαία επιλογή και η επαναληπτική ανάλυση των δεδομένων βοηθούν στον καλύτερο έλεγχο και στη μείωση του πειραματικού σφάλματος. Η μελέτη πεδίου πρέπει να επιλέγεται όταν έχουμε εγκαταστήσει ένα πιλοτικό πρόγραμμα με το οποίο αποσκοπούμε να παρατηρήσουμε και εκτιμήσουμε τις εδράσεις των αλλαγών .

Παρόλα αυτά, σε αντίθεση με τα τυπικά Πειράματα(Formal Experiments) και τις έρευνες πεδίου (survey), η Μελέτη πεδίου δεν έχει ευκατανόητη θεωρητική βάση [Sfetsos]. Το μεγαλύτερο μειονέκτημα της μεθόδου είναι η έλλειψη ελέγχου της κατάστασης που διερευνάται, με συνέπεια τα αποτελέσματα να μην γίνονται

αποδεκτά με βεβαιότητα. Επίσης, οι μικρές ή απλοποιημένες μελέτες περίπτωσης σπάνια αποτελούν ένα καλό όργανο για την ανακάλυψη αρχών και τεχνικών που αφορούν την τεχνολογία λογισμικού. Η καλή ανάλυση της Μελέτης Πεδίου βοηθά στο να αποφασιστεί αν τα αποτελέσματα της μελέτης είναι εφαρμόσιμα στην εκάστοτε περίπτωση. Αν και η Μελέτη Πεδίου δεν έχει το ίδιο επιστημονικό ρίγος και αυστηρότητα όπως τα τυπικά Πειράματα , παρέχουν επαρκή πληροφορία ώστε να βοηθήσει στη σωστή κρίση. Η Μελέτη πεδίου είναι ευκολότερη στο να σχεδιαστεί αλλά δυσκολότερη στην μετάφραση-αποκωδικοποίηση της και να γενικοποιηθεί.

Η Μελέτη Πεδίου προτιμάται έναντι των τυπικών μελετών όταν οι αλλαγές των διαδικασιών είναι πολύ μεγάλου εύρους και όταν οι επιδράσεις των αλλαγών δεν μπορούν να ανιχνευτούν έμμεσα. Τα αποτελέσματα της Μελέτης Πεδίου είναι εξαρτημένα από τα συμφραζόμενα, αλλά για περισσότερη αξιοπιστία της μεθόδου και των ωφελημάτων της είναι χρήσιμο η συχνή αναφορά των αποτελεσμάτων σε πολυάριθμες πηγές υπό διαφορετικές συνθήκες.

Μια καλή μελέτη πεδίου περιέχει:

1. Συγκεκριμένη, διακριτή και ελεγμένη υπόθεση
2. Χρήση μεταβλητών κατάστασης και για το ερευνητικό έργο και για την ανάλυση των δεδομένων
3. Αναλυτικό σχεδιασμό της μελέτης
4. Χρήση της κατάλληλης παρουσίασης και ανάλυσης των τεχνικών για την ανάλυση των Αποτελεσμάτων

Γραμμές Καθοδήγησης Μελέτης Πεδίου της Kitcenham

- Ορισμός της υπόθεσης
- Επιλογή πιλοτικού ερευνητικού έργου
- Εξακρίβωση της μεθόδου σύγκρισης
- Ελαχιστοποίηση της επίδρασης των συγχεόμενων παραγόντων
- Σχεδίαση της Μελετητής Πεδίου
- Απεικόνιση της Μελέτης Πεδίου

- Αναφορά και ανάλυση των Αποτελεσμάτων.

Σύμφωνα με την παραπάνω μεθοδολογία της Kitchenham συντάχθηκε η παρούσα μελέτη πεδίου της πτυχιακής εργασίας.

4.1.2 Έρευνα Πεδίου - Survey

Η έρευνα πεδίου είναι μια μελέτη ανασκόπησης με σκοπό την τεκμηρίωση των σχέσεων και των αποτελεσμάτων μιας συγκεκριμένης κατάστασης. Μία έρευνα πεδίου μπορεί να χαρακτηριστεί ως [Sfetsos]:

- Περιγραφική (descriptive), όταν στοχεύει σε αξιολογήσεις του πληθυσμού.
- Επεξηγηματική (explanatory), όταν στοχεύει σε επεξηγήσεις σχετικά με τον πληθυσμό.
- Εξερευνητική (explorative), όταν χρησιμοποιείται σαν μια προκαταρκτική μελέτη με σκοπό τη διεξαγωγή μιας πιο λεπτομερούς έρευνας.

Στις έρευνες πεδίου καταγράφονται δεδομένα, από τα οποία θα καθοριστεί ο τρόπος με τον οποίο αντέδρασαν οι συμμετέχοντες ενός έργου ανάπτυξης σε κάποια συγκεκριμένη μέθοδο, εργαλείο, ή τεχνική, ή θα καθοριστούν οι τάσεις και οι σχέσεις. Επίσης, μπορούν να καταγραφούν πληροφορίες που αφορούν τα προϊόντα ή έργα ανάπτυξης, τον αριθμό σφαλμάτων ή της προσπάθειας που καταβλήθηκε. Τα κύρια μέσα για τη συλλογή των ποιοτικών ή ποσοτικών δεδομένων είναι οι συνεντεύξεις ή τα ερωτηματολόγια που εφαρμόζονται σε ένα αντιπροσωπευτικό δείγμα του προς διερεύνηση πληθυσμού. Οι ερωτήσεις των ερωτηματολογίων και των συνεντεύξεων πρέπει να σχεδιαστούν με ιδιαίτερη προσοχή, ώστε τα συλλεγόμενα δεδομένα να είναι τα πλέον αντιπροσωπευτικά της διερευνούμενης κατάστασης. Στη συνέχεια, τα δεδομένα αναλύονται, για να εξαχθούν περιγραφικά και επεξηγηματικά συμπεράσματα. Στις έρευνες πεδίου συνήθως δεν έχουμε τον έλεγχο των μεταβλητών της κατάστασης που εξετάζουμε. Έτσι τα συμπεράσματα αυτών των ερευνών συγκρίνονται με τα συμπεράσματα ερευνών παρομοίων καταστάσεων.

4.1.3 Τυπικό ή Ελεγχόμενο Πείραμα – Formal experiment

Το τυπικό πείραμα πραγματοποιείται, όταν απαιτείται ο πλήρης έλεγχος μιας κατάστασης καθώς και ο ακριβής και άμεσος χειρισμός της (Fenton και Pfleeger, 1997; Wohlin et al., 2000). Το τυπικό πείραμα είναι κατάλληλη μέθοδος, όταν ερευνώνται θέματα όπως :

- επιβεβαίωση θεωριών και συμβατικής γνώσης,
- έρευνα συσχετίσεων,
- αξιολόγηση της ακρίβειας των μοντέλων,
- επαλήθευση μετρήσεων.

Για τον ορισμό ενός πειράματος χρησιμοποιείται ένα πρότυπο ορισμού, που διασφαλίζει ότι όλες οι σημαντικές οπτικές γωνίες θα ληφθούν υπ' όψιν πριν τη σχεδίαση και την εκτέλεση του πειράματος.

Για την οργάνωση και εκτέλεση ενός τυπικού πειράματος απαιτείται μία ξεκάθαρα δηλωμένη *υπόθεση* (hypothesis) που διατυπώνει μία σχέση *αιτίου-αποτελέσματος* (cause-effect). Στη σχεδίαση του πειράματος ορίζεται πρώτα το αντικείμενο της μελέτης, δηλαδή η οντότητα που θα μελετηθεί στο πείραμα, για παράδειγμα ένα προϊόν, μία διαδικασία, μία πηγή ή μέσο, ένα μοντέλο, ένα σύστημα μετρικών ή τέλος μια θεωρία. Έπειτα, καθορίζονται τα διάφορα *αντικείμενα πειραματισμού* (treatments), δηλαδή οι διάφορες καταστάσεις που θα συγκριθούν ή αξιολογηθούν όπως δραστηριότητες, μέθοδοι, ή εργαλεία. Σε ένα πείραμα η ανάθεση στα *συμμετέχοντα άτομα – υποκείμενα* (subjects) ενός αντικειμένου πειραματισμού γίνεται με *τυχαία κατανομή* (random sampling) σε μία ξεχωριστή δοκιμαστική εκτέλεση που ονομάζεται *δοκιμασία* (trial). Στη συνέχεια ορίζονται οι *ανεξάρτητες* (independent variables) και οι *εξαρτημένες μεταβλητές* (dependent variables) του πειράματος. Οι ανεξάρτητες μεταβλητές που ονομάζονται *παράγοντες* (factors) είναι οι μεταβλητές εξόδου, δηλαδή οι μεταβλητές που επηρεάζονται και που ελέγχουμε κατά την διάρκεια του πειράματος. Οι εξαρτημένες μεταβλητές είναι μεταβλητές εισόδου, δηλαδή οι παράγοντες που θα πάρουν διαφορετικές τιμές, ώστε να μελετηθεί η επίδραση των αλλαγών στις ανεξάρτητες μεταβλητές.

Η φάση της εκτέλεσης ενός πειράματος ακολουθεί τρία στάδια [Sfetsos]:

- Την προετοιμασία (preparation), όπου επιλέγονται οι συμμετέχοντες και ετοιμάζεται το υλικό
- την εκτέλεση (execution), όπου οι συμμετέχοντες εκτελούν τις εργασίες σύμφωνα με τα διαφορετικά αντικείμενα πειραματισμού και συλλέγονται τα δεδομένα,
- Την επικύρωση των δεδομένων (data validation), όπου τα συλλεγόμενα δεδομένα ελέγχονται ως προς την εγκυρότητα τους.

Ο σχεδιασμός του τυπικού πειράματος ακολουθεί τα παρακάτω βήματα: την *επιλογή περιβάλλοντος* (context selection), το *σχηματισμό υποθέσεων* (hypothesis formulation), την *επιλογή μεταβλητών* (variables selection), την *επιλογή συμμετεχόντων - υποκειμένων* (selection of subjects), τη *σχεδίαση του πειράματος* (experiment design), τη *χρήση βοηθητικών εργαλείων* (instrumentation), και *την εκτίμηση εγκυρότητας* (validity evaluation).

Η διεξαγωγή ενός πειράματος απαιτεί μια διαδικασία πέντε βημάτων (Wohlin et al., 2000):

- τον *ορισμό* (definition), όπου καθορίζεται το ερευνητικό αντικείμενο.
- τον *σχεδιασμό* (planning), όπου καθορίζεται το σχέδιο του πειράματος και εντοπίζονται οι πιθανές *απειλές* (threats).
- την *εκτέλεση* (operation), όπου εκτελείται το πείραμα σύμφωνα με το σχεδιασμό.
- την *ανάλυση και ερμηνεία* (analysis and interpretation) των μετρήσεων, όπου οι μετρήσεις θα αναλυθούν και θα εκτιμηθούν.
- την *συγκέντρωση και παρουσίαση* (package and presentation) των αποτελεσμάτων, όπου τα αποτελέσματα θα συγκεντρωθούν, μορφοποιηθούν, και παρουσιαστούν.

Πολλά είναι τα πλεονεκτήματα από την διεξαγωγή ενός τυπικού πειράματος (Basili and Weiss, 1984; Basili et al., 1999). Το πιο σημαντικό είναι ότι τα αποτελέσματα ενός πειράματος μπορούν ευκολότερα να γενικευθούν. Ένα άλλο πλεονέκτημα είναι ότι διασφαλίζει τον ερευνητή με ένα μεγαλύτερο βαθμό ελέγχου. Όμως θα πρέπει να λάβουμε υπόψη ότι πραγματοποιούνται σε εργαστηριακό περιβάλλον και ότι είναι μελέτες που αφορούν έρευνες μικρής έκτασης.

4.2 Μεθοδολογία

Ο σκοπός της πτυχιακής εργασίας είναι να εξετάσει την καταλληλότητα και τις ωφέλειες χρήσης σχεδιαστικών προτύπων σε ένα πεδίο λογισμικού, τα εργαλεία σχεδίασης λογισμικού. Για την εκπλήρωση αυτού του σκοπού, η μελέτη πεδίου συντάχθηκε σύμφωνα με τις γραμμές καθοδήγησης της Kitchenham. Παρακάτω παρουσιάζεται σταδιακά η εξέλιξη της μελέτης πεδίου.

4.2.1 Ορισμός της υπόθεσης

Η διεξαγωγή της μελέτης πεδίου ερευνά τις επιπτώσεις εφαρμογής των προτύπων όπως περιγράφεται από τις παρακάτω υποθέσεις :

Έστω Vnp η έκδοση του λογισμικου η οποία δεν χρησιμοποιεί σχεδιαστικό πρότυπο για την συγκεκριμένη λειτουργικότητα, και Vp η έκδοση του λογισμικου στην οποία γίνεται χρήση σχεδιαστικού προτύπου για την παραπάνω λειτουργικότητα.

Έστω diffmetric(i) το αποτέλεσμα της αφαίρεσης των τιμών στις μετρικές του Vp από τις τιμές των μετρικών τουVnp.

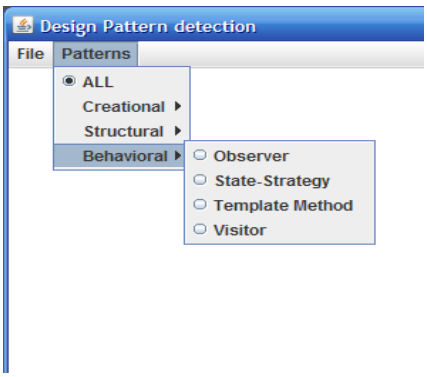
Υποθέτουμε ότι κατά τη διάρκεια τη εξέλιξης των εκδόσεων του εργαλείου σχεδίασης λογισμικου, εμφανίστηκαν Ksdt επιπροσθετα στιγμιοτυπα προτυπων.

Υπόθεση(1): Ο μέσος όρος των τιμών στις μετρικές (Mi) στις Ksdt-Vnp εκδόσεις δεν διαφέρει στατιστικά από το μέσο όρο των μετρικών (Mi) στις Ksdt-Vp εκδόσεις.

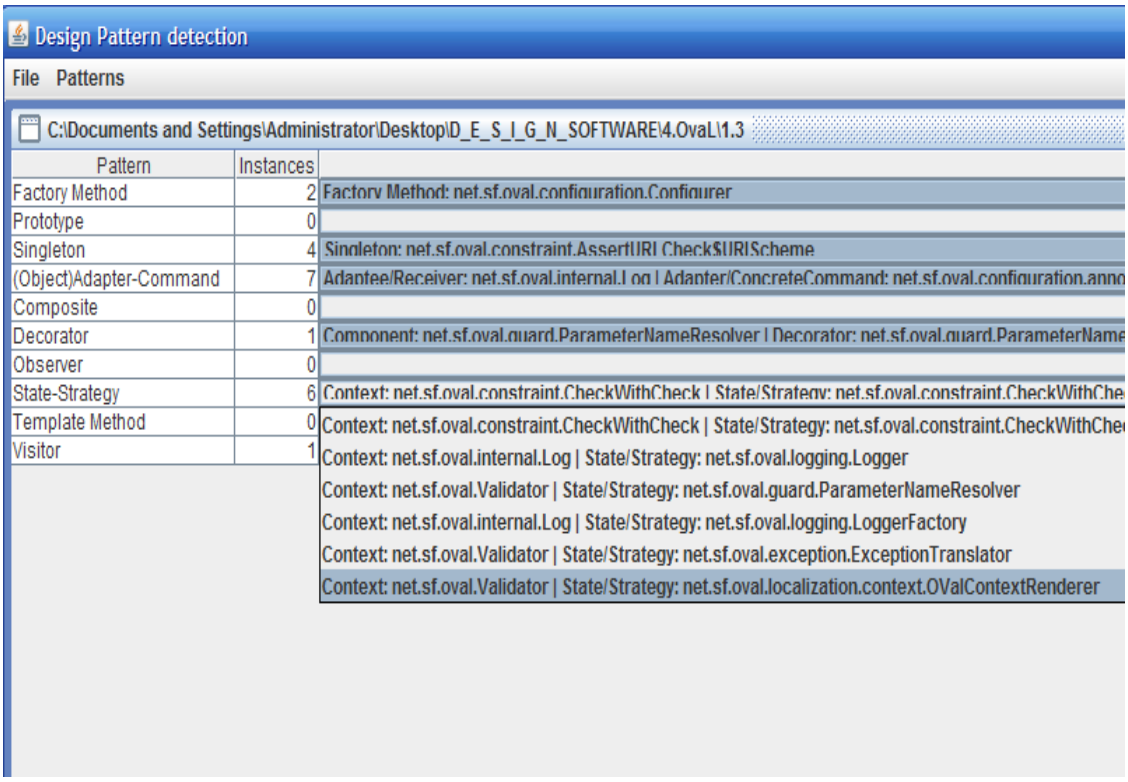
Υπόθεση(2): Ο μέσος όρος των τιμών στις μετρικές (Mi) στις Ksdt-Vnp εκδόσεις διαφέρει στατιστικά από το μέσο όρο των μετρικών (Mi) στις Ksdt-Vp εκδόσεις.

4.2.2 Επιλογή πιλοτικού ερευνητικού έργου

Η επιλογή του πιλοτικού προγράμματος έγινε με τέτοιες προϋποθέσεις ώστε να ταιριάζει στην έρευνα. Δηλαδή, να υπάρχουν διαδοχικές εκδόσεις του προγράμματος με εμφανή και επιβαιβεωμένη προσθήκη σχεδιαστικών προτύπων στο πηγαίο του κώδικα, και επιπλέον το μέγεθος αυτού να τόσο ώστε να μπορεί να μελετηθεί - να είναι δηλαδή σε πλαίσια μελέτης. Επίσης υπήρχε και ένας ακόμα περιορισμός στην ερευνα της μελέτης πεδίο. Ο πηγαίος κώδικας του προγράμματος έπρεπε να ήταν συνταγμένος σε Java, διότι το εργαλείο ανίχνευσης προτύπων Design Pattern Detection Tool (Tsantalis) ελέγχει την ύπαρξη σχεδιαστικών προτύπων σε γλώσσα προγραμματισμού Java. Το εργαλείο αυτό απαιτεί την εγκατάσταση του πακέτου Java™ 2 Runtime Environment, Standard Edition 1.5.0 και εξέταζε πακέτα Java bytecode. Τα σχεδιαστικά πρότυπα είναι χωρισμένα σε τρεις κατηγορίες Creational, Structural, Bevahioral με επιλογή μαρκαρίσματος όλων. Γενικά είναι ένα εύχρηστο και αρκετά αξιόπιστο εργαλείο.



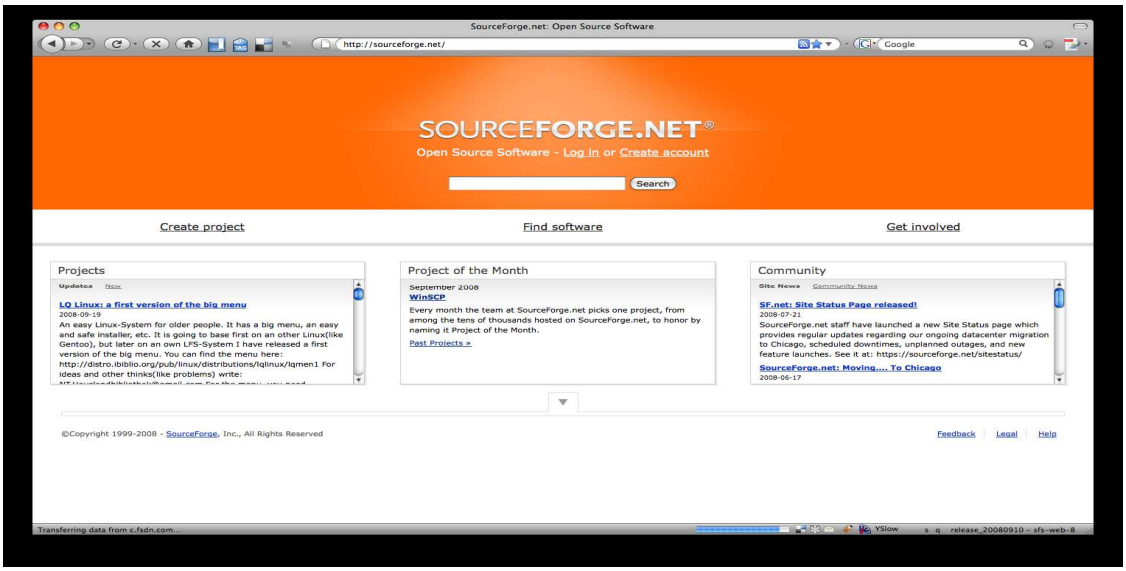
□□□□□α 4. Design Pattern Detection Tool



□□□□□**α 5. Design Pattern Detection Tool**

Για κάθε σχεδιαστικό στιγμιότυπο που εντοπίζει, εμφανίζει και το μονοπάτι της σχετικής διαδρομής (relative path) της κλάσης που συμμετέχει στο συγκεκριμένο πρότυπο. Επίσης το εργαλείο δίνει την δυνατότητα εξαγωγής των αποτελεσμάτων σε XML μορφή.

Για τους προαναφερόμενους λόγους επιλέχτηκε για πιλοτικό πρόγραμμα ένα εργαλείο σχεδίασης λογισμικου το *Oval* , το οποίο βρέθηκε σε ιστότοπο που λειτουργεί ως μια διαδυκτιακή δεξαμενή ελεύθερων λογισμικών (open source), το Sourceforge και είναι διαθέσιμο στην διεύθυνση http://sourceforge.net/projects/oval/.



□□□□□α 6. □στ□τ□π□□ SourceForge.net

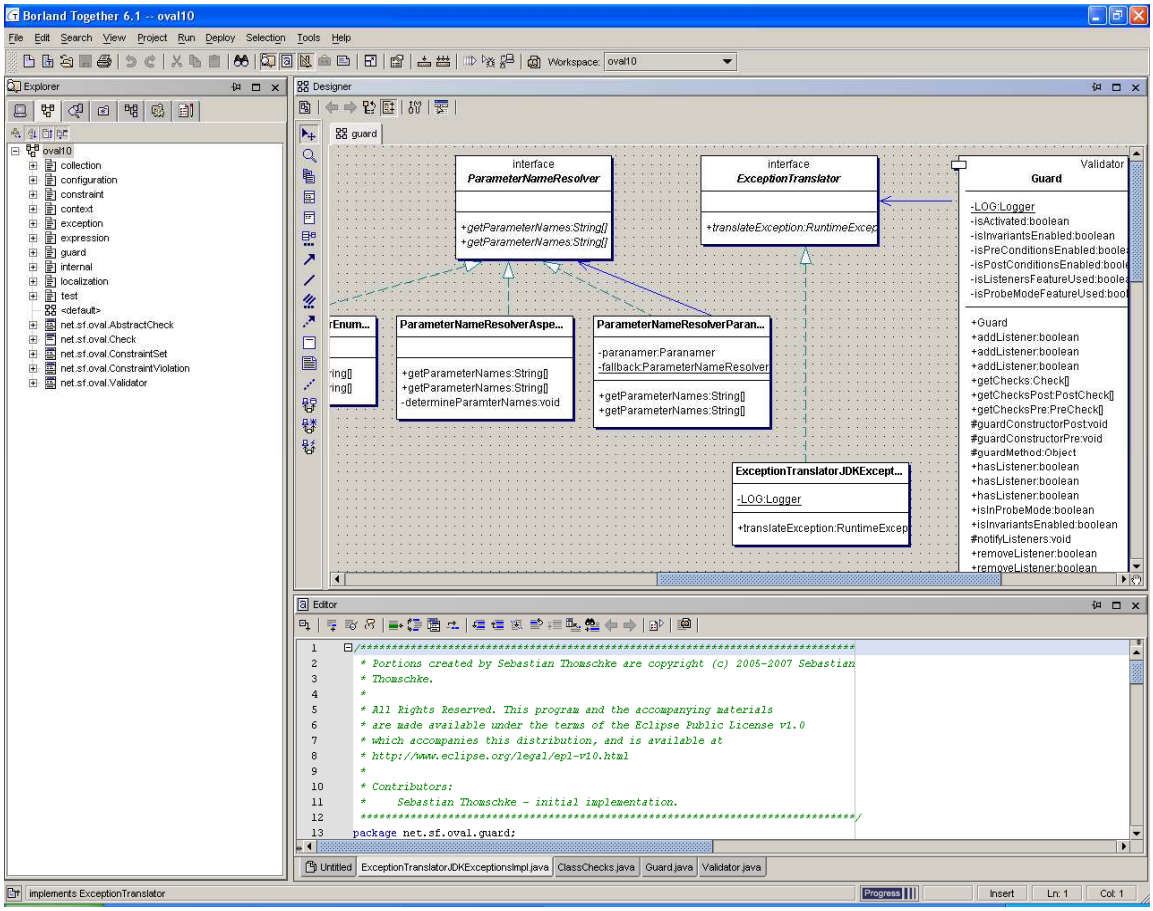
Για το Oval ερευνήθηκαν οι εξής πέντε (5) διαδοχικές εκδόσεις :

- *OVaL, Version 0.5 (SDT-1)*
- *OVaL, Version 0.7 (SDT-2)*
- *OVaL, Version 1.0 (SDT-3)*
- *OVaL, Version 1.2 (SDT-4)*
- *OVaL, Version 1.3 (SDT-5)*

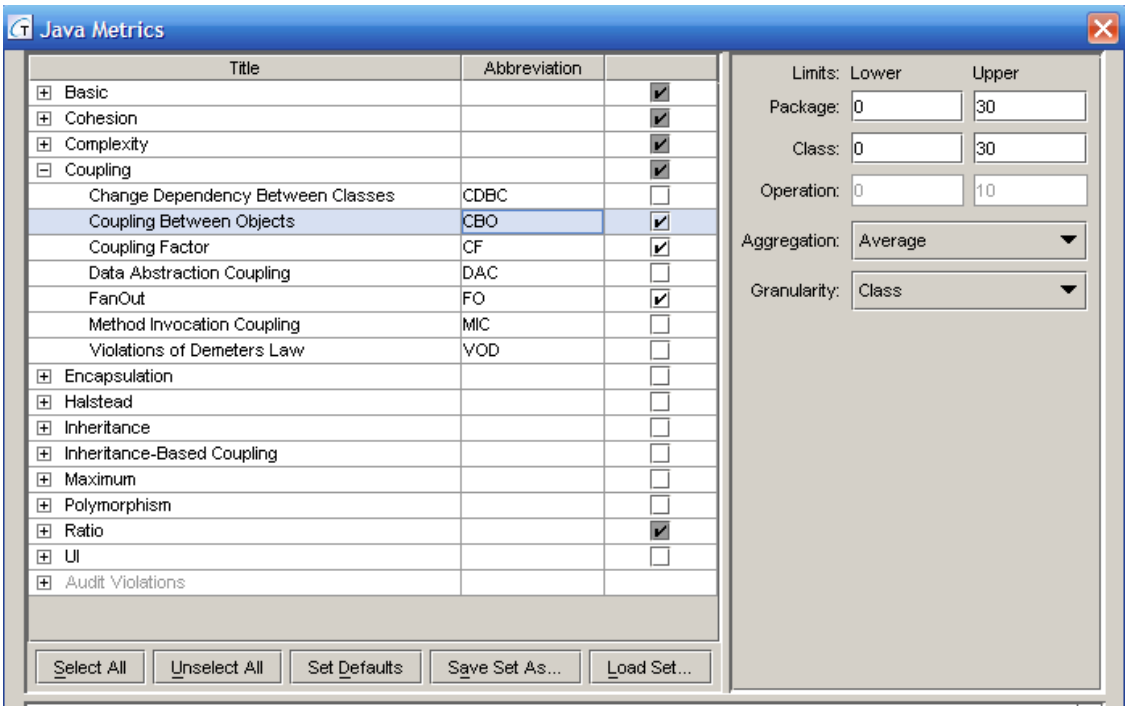
Οι μετρήσεις της ποιότητας του εργαλείου σχεδίασης λογισμικου Oval στην εξέλιξη του, έγινε με την χρήση της εφαρμογής Borland Together 6.1 Control Center. Το λογισμικό αυτό περιλαμβάνει όλα τα τυποποιημένα διαγράμματα UML, συν πρόσθετα διαγράμματα για άλλους ειδικούς τύπους διαμορφώσεων. Διαγράμματα κλάσης και ακολουθίας μπορούν να παράγουν τον πηγαίο κώδικα αυτόματα. Το Together 6.1 επιτρέπει σε μια ολόκληρη ομάδα ανάπτυξης να συνεργαστεί χρησιμοποιώντας κοινή γλώσσα, διαγράμματα, και λογισμικό σε μια ενιαία πλατφόρμα με έναν συνεπή αλλά εξατομικεύσιμη διεπαφή με τον χρήστη για όλη την εργασία τους σε όλο το κύκλο ανάπτυξης λογισμικού. Το κύριο πλεονέκτημα είναι η

δυνατότητά του να συγχρονίσει πλήρως τα διαγράμματα με τον πηγαίο κώδικα χρησιμοποιώντας LiveSource, όπως επίσης η δυνατότητα εισόδου βάσης δεδομένων και σχεδιαστικών προτύπων. Η σύνταξη κώδικα υποστηρίζεται επίσης με έτοιμα πρότυπα κώδικα και αναδόμηση (refactoring).

Το βασικό παράθυρο εργασίας διαιρείται σε πολλά βοηθητικά υποπαράθυρα, όπως αυτό που παρουσιάζει την ιεραρχική δομή του εξεταζόμενου project, αυτό που παρουσιάζει τα διαγράμματα κλάσεων με τις διασυνδέσεις, αυτό που παρουσιάζει το κώδικα του εξεταζόμενου project, και αυτό που αναλύει τις τιμές των μετρικών στα πακέτα και στις κλάσεις. Μεγάλο πλήθος από αντικειμενοστρεφείς μετρικές και ελεγχούς (audits and metrics) υποστηρίζεται από το σύστημα σε ξεχωριστό παράθυρο που αναρτάται. Ειδικά γραφήματα παρουσιάζουν την επίδραση της μετρικής, καθώς παρέχονται και συμβουλές με τρόπους για βελτίωσης των τιμών.



□□□□□α 7. Borland Together 6.1 ControlCenter



□□□□□α 8. **Borland Together 6.1 Metrics**

4.2.3 Προσδιορισμός της μεθόδου σύγκρισης

Για την καταχώρηση δεδομένων χρησιμοποιηθήκαν πίνακες αποτελούμενοι από γραμμές, για κάθε πρότυπο που προσθέτοντας μαζί τις κλάσεις που παίρνουν μέρος , και από στήλες, για κάθε μετρική ξεχωριστά και στην προυπάρχουσα έκδοση και στην διάδοχή της.

	latest version							prior version						
project module	**LOC**	**NOC**	**CF**	**CBO**	**WMPC**	**CC**	**LCOM**	**LOC**	**NOC**	**CF**	**CBO**	**WMPC**	**CC**	**LCOM**
project														
package A														
package B														
package C														
... ...														
class A														
class B														
class C														
class D														
class E														
... ...														

□□□□□α 9. **Dataset Sample**

Ο μέσος ορός των τιμών των μετρικών υπολογίστηκε σύμφωνα με τον τύπο

$$MS_jP_k = \frac{\sum_{i=1}^{i=NOC}(MS_jC_i)}{NOC}$$

NOC Ο συνολικός αριθμός κλάσεων που ορίστηκαν στο πακέτο
MS_jC_i : Η τιμή της Μετρικής j της κλάσης i
MS_jP_k : Η τιμή της Μετρικής j του πακέτου k

□□**π**□□ **1.** □□**σ**□□ □□□□

Και η μέγιστη τιμή των τιμών των πακέτων κλάσεων

$$MS_jP_k = \max_{i=1}^{i=NOC}(MS_jC_i)$$

NOC Ο συνολικός αριθμός κλάσεων που οριστήκαν στο πακέτο
MS_jC_i : Η τιμή της Μετρικής j της κλάσης i
MS_jP_k : Η τιμή της Μετρικής j του πακέτου k

□□**π**□□ **2.** □□□□**στ**□□ □□□□

Για να ερευνηθούν οι υποθέσεις που περιγράφονται στην ενότητα 4.2.1 πραγματοποιήθηκαν t-tests. Πιο συγκεκριμένα, οι Υποθέσεις 1 και 2 επιβεβαιώθηκαν με paired sample t-tests τα οποία εξετάζουν την ύπαρξη στατιστικά σημαντικών διαφορών μεταξύ των μέσων όρων δυο μεταβλητών για παράδειγμα MSjPprior, MSjPlater για προηγούμενα ή επόμενα πακέτα κλάσεων τα οποία παρέχουν συγκεκριμένη λειτουργικότητα.

4.2.4 Ελαχιστοποίηση της επίδρασης των συγχεόμενων παραγόντων

Στις μελέτες πεδίου, παράγοντες διαφορετικοί από ανεξάρτητες μεταβλητές, οι οποίες επηρεάζουν την τιμή της εξαρτημένης μεταβλητής, θεωρούνται συγχεόμενοι

παράγοντες. Στην περίπτωση της διαφοράς των τιμών στις μετρικές μεταξύ δυο εκδόσεων, διάφοροι παράγοντες θεωρούνται ικανοί να επηρεάσουν. Τέτοιοι παράγοντες είναι η προστιθεμένη λειτουργικότητα και το σχεδιαστική αναδόμηση(design refactoring). Στη μελέτη πεδίου στο συγκεκριμένο λογισμικό, το Oval, περιορίστηκε η προσθήκη λειτουργικότητας αλλά δεν εξαλείφθηκε.

4.2.5 Σχεδίαση της Μελετητής Πεδίου

Σε μια συμπαγή μελέτη πεδίου είναι ωφέλιμη η ύπαρξη ενός πλάνου αξιολόγησης. Πρώτα διατυπώνονται οι υποθέσεις, έπειτα παρουσιάζεται το πιλοτικό πρόγραμμα και εν συνεχεία καθορίζεται η μέθοδος σύγκρισης.

Τα σχεδιαστικά πρότυπα που πήραν μέρος είναι τα εξής: Method Factory, Prototype, Singleton, Adapter, Composite, Decorator, Observer, State,Strategy, Template, Visitor. Οι αντικειμενοστρεφείς μετρικές που χρησιμοποιήθηκαν είναι οι εξής : Lines of Code (LOC), Number of Classes (NOC), Weighted Method per Class (WMPC), Cyclomatic Complexity (CC), Fan Out (FO), Coupling between Objects (CBO), and Lack of Cohesion between Methods (LCOM). Στα προαναφερθέντα πρότυπα και στις προαναφερθέντες μετρικές εξετάστηκε το πιλοτικό πρόγραμμα.

4.2.6 Απεικόνιση και Ανάλυση Αποτελεσμάτων

Σε αυτό το υποκεφάλαιο παρουσιάζονται τα αποτελέσματα της μελέτης πεδίου. Ο πίνακας 1 παρουσιάζει για κάθε έκδοση, τα σχεδιαστικά πρότυπα που ανίχνευσε το εργαλείο ανίχνευσης σχεδιαστικών προτύπων, Design Pattern Detection Tool , και επιβεβαιωθήκαν με οπτική επιθεώρηση μέσω των UML διαγραμμάτων κλάσεων. Όπως είναι φανερό υπάρχει προσθήκη σχεδιαστικών προτύπων κατά την εξέλιξη του πιλοτικού προγράμματος Oval. Επίσης στον πίνακα 2 παρουσιάζεται το μέγεθος του πιλοτικού προγράμματος Oval για κάθε έκδοση με τη βοήθεια των μετρικών Γραμμές κώδικα LOC και Αριθμός Κλάσεων NOC. Όπως είναι φανερό υπάρχει σαφής αύξηση των τιμών που υποδηλώνει αύξηση της λειτουργικότητας του συστήματος.

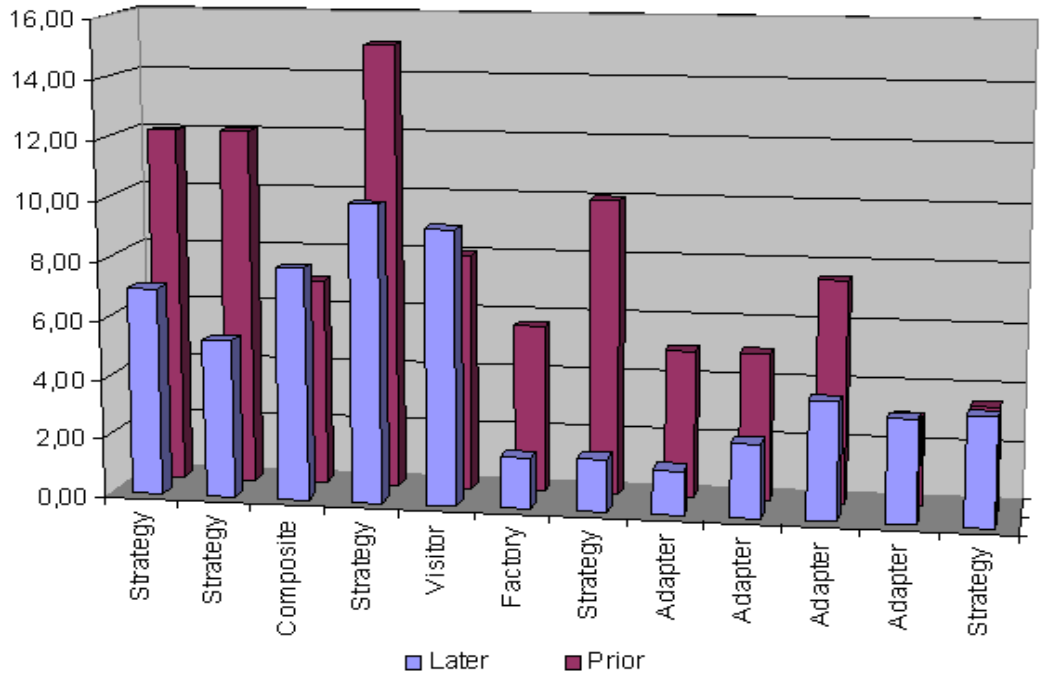
Version	Factory	Prototype	Singleton	Adapter	Composite	Decorator	Observer	State	Template	Visitor
0.5	*0*	*0*	*1*	*0*	*0*	*0*	*0*	*3*	*0*	*0*
0.7	*1*	*0*	*4*	*0*	*0*	*0*	*0*	*3*	*0*	*0*
1.0	*1*	*0*	*1*	*1*	*1*	*0*	*0*	*3*	*0*	*1*
1.2	*2*	*0*	*2*	*5*	*1*	*0*	*0*	*5*	*0*	*1*
1.3	*2*	*0*	*4*	*7*	*1*	*0*	*0*	*6*	*0*	*1*

□□□**α**□**α**□ **1.** □**τ**□□□□□**τ**□**πα** □□**εδ**□**αστ**□□□□ □□□**τ**□**π**□□

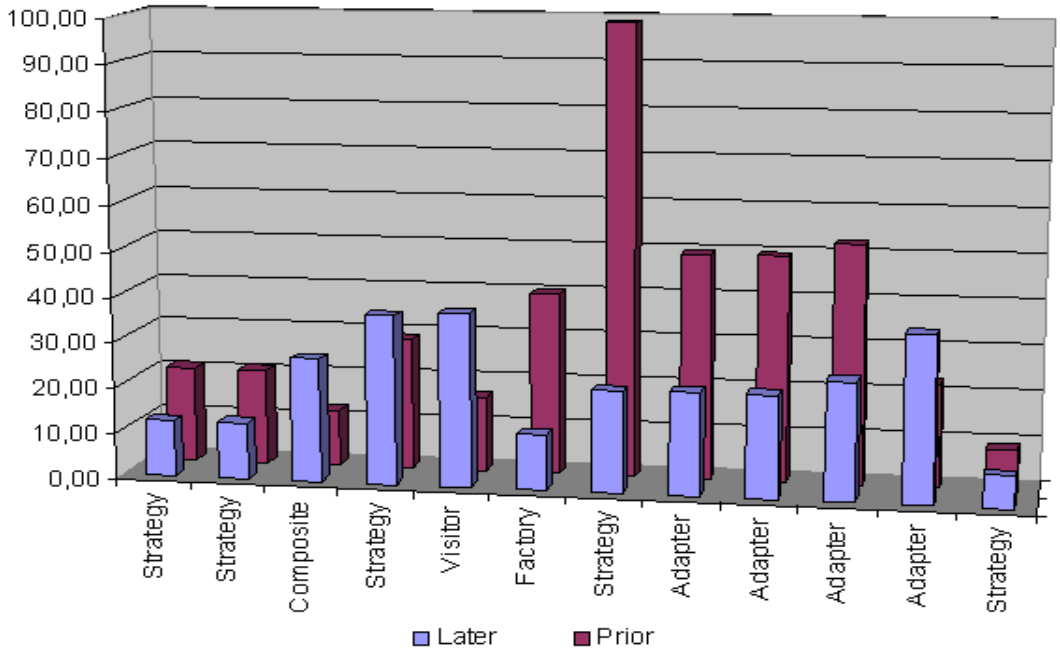
Software	Version	LOC	NOC
SDT	***0.5***	*1905*	*111*
	0.7	*4905*	*219*
	1.0	*9439*	*368*
	1.2	*13432*	*494*
	1.3	*9126*	*410*

□□□**α**□**α**□ **2.** □**έ**□**ε**□□□ □**α**□**έτ**□□

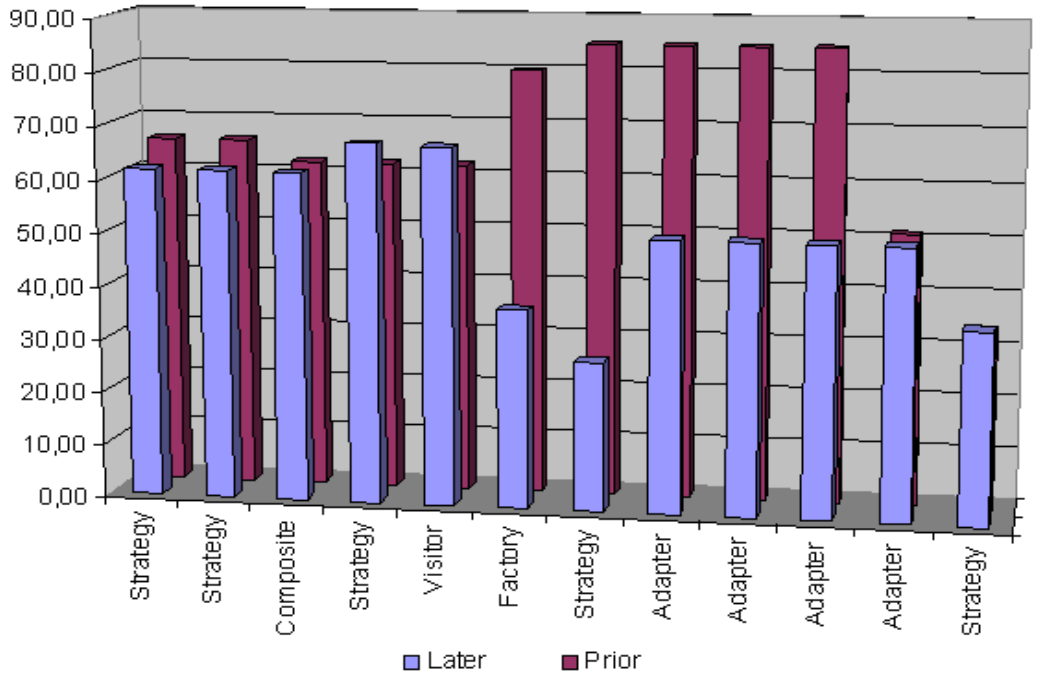
Στις εικόνες 10-12 παρουσιάζονται με τη μορφή ραβδογράμματος οι τιμές των μετρικών στην συνοχή, την πολυπλοκότητα και την σύζευξη των κλάσεων που συμμετέχουν στα πρότυπα, κατά την εξέλιξη των εκδόσεων, στο εργαλείο σχεδίασης λογισμικών Oval. Με κόκκινο συμβολίζεται η παλαιά έκδοση χωρίς την εμπλοκή προτύπων, ενώ με μπλε συμβολίζεται η νεότερη έκδοση στην οποία προστέθηκε κάποιο από τα εξεταζόμενα πρότυπα. Γίνεται φανερό πότε η επίδραση των σχεδιαστικών προτύπων είναι ωφέλιμη, όπως σχεδόν σε όλες εφαρμογές του προτύπου ΄΄Στρατηγική΄΄ (Strategy), και πότε η χρήση τους δεν έφερε τα επιθυμητά αποτελέσματα, όπως η χρήση του προτύπου ΄΄Επισκέπτης΄΄(Visitor).



□□□□□α 10. □□ □□σ□□ □□□□ τ□□ σ□□ε□□□□ στ□□ □ετ□□□□ CBO



□□□□□α 11. □□ □□σ□□ □□□□ τ□□ π□□□π□□□□τ□τα□ στ□□ □ετ□□□□ CC



□□□□□α 12. □□ □□σ□□ □□□□ τ□□ σ□□□□□□ στ□□ □ετ□□□□ **LOCOM**

Για να ερευνηθεί το κατά ποσό εκπληρώνονται οι Υποθέσεις που οριστήκαν στην ενότητα 4.2.1, εκτελεστήκαν t-tests. Με το paired-sample t-tests ερευνήθηκε αν οι διαφορές στις τιμές των μετρικών στις εκδόσεις με πρότυπα και αυτών χωρίς πρότυπα είναι στατιστικά σημαντικές ή οχι. Έτσι, για κάθε μετρική CBO, CC, FO, LOCOM3, WMPC1 και WMPC2 δυο paired-sample t-tests διεξήχθησαν –ένα για το μέσο όρο και ένα για τη μέγιστη τιμή.

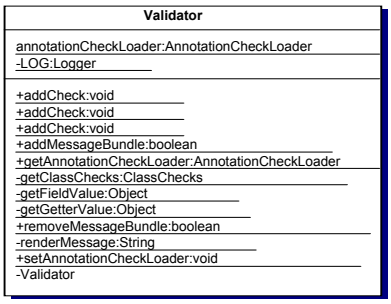
Metric	**Version**	**Average**		**Maximum**	
		Mean	**sig.**	**Mean**	**sig.**
CBO	***Prior***	*7,80*	*0,039*	*11,83*	*0,001*
	Later	*4,84*		*9,83*	
CC	***Prior***	*35,19*	*0,962*	*61,33*	*0,352*
	Later	*22,97*		*55,86*	
FO	***Prior***	*7,16*	*0,043*	*10,66*	*0,001*
	Later	*4,14*		*8,61*	

LOCOM3	***Prior***	*68,54*	*0,636*	*74,50*	*0,966*
	Later	*52,13*		*67,41*	
WMPC1	***Prior***	*35,19*	*0,962*	*61,33*	*0,352*
	Later	*22,97*		*55,86*	
WMPC2	***Prior***	*48,74*	*0,229*	*83,00*	*0,395*
	Later	*30,06*		*69,61*	

□□□**α**□**α**□ **3.** □□**σ**□□ □□□□ □**α**□ □□□□**στε**□ **τ**□□□□

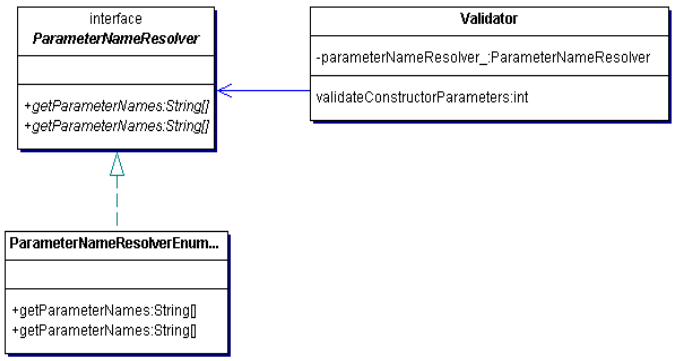
5. Συμπεράσματα

Τα αρχικά συμπεράσματα αφορούν την ποιοτική αξιολόγηση της εφαρμογής. Στις εικόνες 10-12, γίνεται φανερό από τα ραβδογράμματα ότι η εφαρμογή του ΄΄Σύνθετου΄΄ σχεδιαστικού προτύπου εισβάλει αρνητικά στην ποιότητα του συστήματος. Στην εικόνα 13 παρουσιάζεται η έκδοση χωρίς εμπλοκή πρότυπου. Υπάρχει η κλάση Validator, στην οποία έχει αποδοθεί μεγάλο μέρος λειτουργικότητας και συμπεριφοράς, δυσχεραίνοντας τη σχεδίαση του συστήματος, γεγονος που αποτυπονεται και στις τιμές των μετρικών της κλάσης.



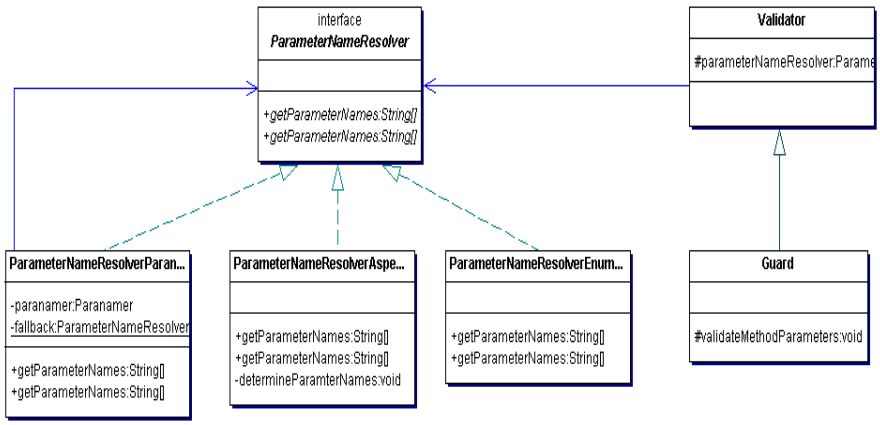
□□□□□α 13. □□□σ□ **Validator**

Στην εικονα 14, απεικονίζεται η εξέλιξη της κλάσης validator σε πρότυπο ΄΄Στρατηγικη΄΄, χωρις ομως ιδιαιτερη βελτιωση στο συστημα. Παρουσιάζονται οι εξής οντότητες: η διεπαφή ParameterNameResolver με την κλάση ParameterNameResolverEnum για την υλοποίηση της διασύνδεσης, καθώς επίσης υπάρχει και η κλάση Validator που συσχετίζεται με την ειδικής μορφής σχέση περιεκτικότητας, τη σύνθεση.



□□□□□α 14. **Strategy Pattern Version**

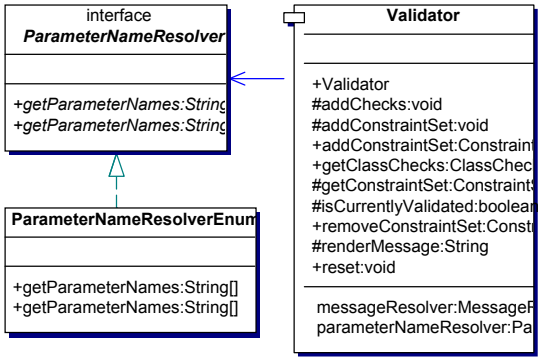
Στην εικόνα 15, απεικονίζεται η μετατροπή του προηγούμενου προτύπου ΄΄Στρατηγική΄΄ σε σχεδιαστικό πρότυπο ΄΄Σύνθετο΄΄. Η εξέλιξη αυτή φαίνεται στο διάγραμμα κλάσης που απεικονίζει την επεκτασιμότητα σε μελλοντικές προσαρμογές και προσθήκες της επαφής ParameterNameResolver ανεξαρτήτως του πρόγραμμα-πελάτη Validator και Guard. Επιπρόσθετα, σύνθετα αντικείμενα μπορούν να δημιουργηθούν από την υποκλάση ParameterNameResolverParameterImpl χωρίς να αλλαχτεί η εφαρμογή της διεπαφής ParameterNameResolver και των υπολοίπων υποκλάσεων. Παρόλα αυτά οι τιμές των μετρικών σε αυτό το πακέτο επηρεαστήκαν αρνητικά. Η πιο πιθανή εξήγηση είναι ότι η λειτουργικότητα από την μεριά του πελάτη δηλαδή του Validator και Guard έχει τριπλασιαστεί στην εξέλιξη των διαδοχικών εκδόσεων. Πιο συγκεκριμένα η τιμή της μετρικής Γραμμές Κώδικα στην αρχική έκδοση (LOCprior) ισοδυναμεί με 571 ενώ στην νεότερη έκδοση η ιδία μετρική (LOClater) ισοδυναμεί με 1607.



□□□□□α 15. Composite pattern

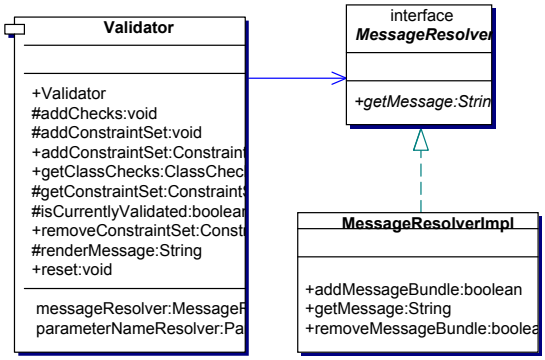
Σε αντίθεση, η χρήση του σχεδιαστικού προτύπου ΄΄Στρατηγική΄΄ (Strategy) στο επόμενο παράδειγμα επηρεάζει ωφέλιμα τη σχεδίαση. Στην έκδοση χωρίς πρότυπο υπάρχει παλι η κλάση Validator της εικόνας 13, στην οποία έχει αποδοθεί μεγάλο μέρος της λειτουργικότητας και συμπεριφοράς, για αυτό και οι τιμές των μετρικών της κλάσης φανερώνουν ότι δυσχεραίνει τη σχεδίαση του συστήματος.

Στην έκδοση με πρότυπο, η ParameterNameResolver είναι μια διασύνδεση, η οποία χρησιμοποιείται από την κλάση Validator, κατα τη διάρκεια της εκτέλεσης, τα αντικείμενα της κλάσης Validator θα χρησιμοποιούν αντικείμενα της παράγωγης κλάσης ParameterNameResolverEnum. Η χρήση του προτύπου ΄΄Στρατηγική΄΄ εφαρμόζει την Αρχή της Ανοικτής – Κλειστής Σχεδίασης. Η κλάση Validator είναι ανοικτή για επέκταση, δημιουργώντας νέες παράγωγες κλάσεις όπως η ParameterNameResolverEnum, αλλά είναι κλειστή σε τροποποιήσεις, κάτι που σημαίνει ότι η Validator παραμένει σχεδόν αμετάβλητη -με μια μικρή βελτίωση. Οι αντικειμενοστρεφείς μετρικές των δυο πακέτων φανερώνουν την ωφέλιμη εφαρμογή του προτύπου λαμβάνοντας υπόψη την πολυπλοκότητα, τη σύζευξη και τη συνοχή.



□□□□□**α 16. Strategy Pattern 1**

Το ίδιο ισχύει και στο επόμενο παράδειγμα. Το πρότυπο ΄΄Στρατηγική΄΄ εφαρμόζοντας την Αρχή της Ανοικτής – Κλειστής Σχεδίασης, αφήνει την κλάση Validator να είναι ανοικτή σε επεκτάσεις , δημιουργώντας παράγωγες κλάσεις όπως η MessageResolverImpl αλλά η ίδια η κλάση Validator είναι κλειστή σε τροποποιήσεις, παραμένοντας σχεδόν αμετάβλητη.



□□□□□α 17. Strategy Pattern 2

Και σε αυτή την περίπτωση οι τιμές των μετρικών μαρτυρούν σαφή βελτίωση στη σύζευξη, τη συνοχή και στην πολυπλοκότητα του συστήματος.

Αναλύοντας ποιοτικά τα αποτελέσματα, διεξήχθησαν paired sample t-tests για τις Υποθέσεις 1 και 2. Τα παρακάτω συμπεράσματα είναι απόρροια των αποτελεσμάτων του Πίνακα 3 με βάσει το μέσο όρο:

- 37.9% μείωση στην τιμή της μετρικής CBO παρατηρήθηκε. Η μείωση είναι στατιστικά σημαντική
- 42.1% μείωση στην τιμή της μετρικής FO παρατηρήθηκε. Η μείωση είναι στατιστικά σημαντική
- 34.7% μείωση στην τιμή της μετρικής CC παρατηρήθηκε. Η μείωση δεν είναι στατιστικά σημαντική
- 34.7% μείωση στην τιμή της μετρικής WMPC1 παρατηρήθηκε. Η μείωση δεν είναι στατιστικά σημαντική
- 38.3% μείωση στην τιμή της μετρικής WMPC2 παρατηρήθηκε. Η μείωση δεν είναι στατιστικά σημαντική
- 23.9% μείωση στην τιμή της μετρικής LOCOM3 παρατηρήθηκε. Η μείωση δεν είναι στατιστικά σημαντική

Οι μέγιστες μετρήσεις πακέτου στο εργαλείο σχεδίασης λογισμικού παρουσιάζουν παρόμοια αποτελέσματα με αυτά των μετρήσεων για τους μέσους

όρους. Επομένως, στο εργαλείο σχεδίασης λογισμικου, η μοναδική κατηγορία μετρικών που φαίνεται να επηρεάζεται σημαντικά από εφαρμογή προτύπου, είναι η σύζευξη. Αντίθετα, οι μετρικές της συνοχής και της πολυπλοκότητας δεν επηρεάζονται ιδιαιτέρα.

5.1 Περιορισμοί και Μελλοντική Μελετη

Παρόλα αυτά, η συγκεκριμένη μελέτη πεδίου δεν αποτελεί υπόδειγμα για γενικά συμπεράσματα στην κατηγορία εργαλεία σχεδίασης λογισμικού. Σημαντικός παράγοντας σε αυτόν τον περιορισμό είναι το μικρό μέγεθος του ανοικτού-κώδικα προγράμματος που χρησιμοποιήθηκε. Τα αποτελέσματα δεν μπορούν διαβαθμιστούν σε βιομηχανικού μεγέθους συστήματα. Επιπροσθέτως, εξορισμού η αξία σημαντικότητας των t-tests καθορίζει τη πιθανότητα του λάθους γενικεύοντας τα αποτελέσματα από το δείγμα στον πληθυσμό, στην πράξη μια τέτοια προσπάθεια είναι επικίνδυνη υπό την έννοια ότι κανένα πρόγραμμα ανοικτού πηγαίου κώδικα δεν μπορεί να είναι αντιπροσωπευτικό κάθε συστημάτων.

Για μελλοντική μελέτη, προτείνεται η έρευνα σε περισσότερες κατηγορίες λογισμικών. Τέτοια προσέγγιση είναι πιο πιθανή να παράγει ασφαλέστερα συμπεράσματα σχετικά με την γενίκευση των αποτελεσμάτων στον πληθυσμό. Επιπροσθέτως, πιθανή συσχέτιση μεταξύ της αναδόμησης(recapturing), της προσθήκης λειτουργικότητας, των σχεδιαστικών προτύπων και της ποιότητας, μπορούν να τεθούν ως θεματα για μελέτη, ερευνα και τυπικά πειράματα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Ampatzoglou A. and Chatzigeorgiou A, (2007), “Evaluation of object-oriented design patterns in game development”, *Information and Software Technology*, volume 49, issue 5, May 2007, pp 445-454

Basili V. R., Selby R.W. and Hutchens D.H., (1986), “Experimentation in Software Engineering”, *IEEE Transactions on Software Engineering*, volume 12, issue 7, pp. 758-773, July 1986.

Chidamber S. R., Darcy D. P., Kemmerer C. F., (1998), “Managerial Use of Metrics for Object Oriented Software: An Exploratory Analysis”, *IEEE Transactions on Software Engineering*, volume 24, pp. 629-639, 1998

Fenton N. E. and Pfleeger S.L.,(1997) *“Software Metrics: A Practical and Rigorous Approach”*, Chapman & Hall, London, 1997.

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

Halstead M., (1977)“Elements of Software Science”. New York :Elsevier North-Holland, 1977

Huston B.(2001), “The effects of design pattern application on metric scores”, *The Journal of Systems and Software*, volume 58, pp. 261-269, 2001.

Khomh F. and Gueheneuc Y.G., (2008) “Do design patterns impact software quality positively?”, *IEEE Proceedings of the 12th European Conference on Software Maintenance and Reengineering (CSMR 2008)*, pp.274-278, 1-4 April 2008, Athens, Greece

Kitchenham B., Pickard L., Pfleeger S.L.,(1995) “Case Studies for Method and Tool Evaluation”, *IEEE Software,* pp. 52-62, July 1995

McCabe T.J., (1976) “A complexity measure” , IEEE Trans.Software. Eng. Vol. SE-2,pp. 308-320, 1976

Nguyen D., Wong S.B., (2002) "Design Patterns for Games", *Proceedings of the 33rd SIGCSE technical symposium on Computer science education*, Cincinnati, Kentucky, pp. 126 – 130, 2002.

Prechelt L., Unger B., Tichy W. F., Brossler P., Votta L. G.,(2001) “A controlled experiment in maintenance comparing design patterns to simpler solutions”, *IEEE* Transactions on Software Engineering, volume 27, issue 12, pages 1134-1144, 2001

Sfetsos P. (2007) "Experimentation in Object Oriented Technology and Agile Methods", Department of Informatics, Aristotle University of Thessaloniki, Greece, 2007.

Stamelos I., Angelis L., Oikonomou A., Bleris G. L.,(2002) “Code quality analysis in open source software development” 2002 Blackwell Science Ltd, Information Systems Journal

Sowe S. K., Angelis L., Stamelos I., Manolopoulos I., 2007 “Using Repository of Repositories (RoRs) to study the growth of F/OSS projects: a meta-analysis research approach”, *Proceedings of the 3rd International Conference on Open Source Systems (OSS’ 2007)*, Limerick, Ireland, June 11–14 2007.

Tsantalis N., Chatzigeorgiou A., Stephanides G., Halkidis S. T., (2006) "Design Pattern Detection using Similarity Scoring", *IEEE Transactions on Software Engineering*, vol. 32, no. 11, November 2006, pp. 896-909

Vokác M., Tichy W., Sjøberg D.I.K., Arisholm E., Aldrin M., (2004) “A Controlled Experiment Comparing the Maintainability of Programs Designed with and without Design Patterns - A Replication in a Real Programming Environment”, *Empirical Software Engineering,* vol. 9, pp. 149-195, 2004.

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

ΕΥΡΕΤΗΡΙΟ ΟΡΩΝ

Α

Δ

Ε

Κ

Λ

Μ

Π

Σ

Τ

ΛΙΣΤΑ ΣΧΗΜΑΤΩΝ – ΠΙΝΑΚΩΝ-ΤΥΠΩΝ