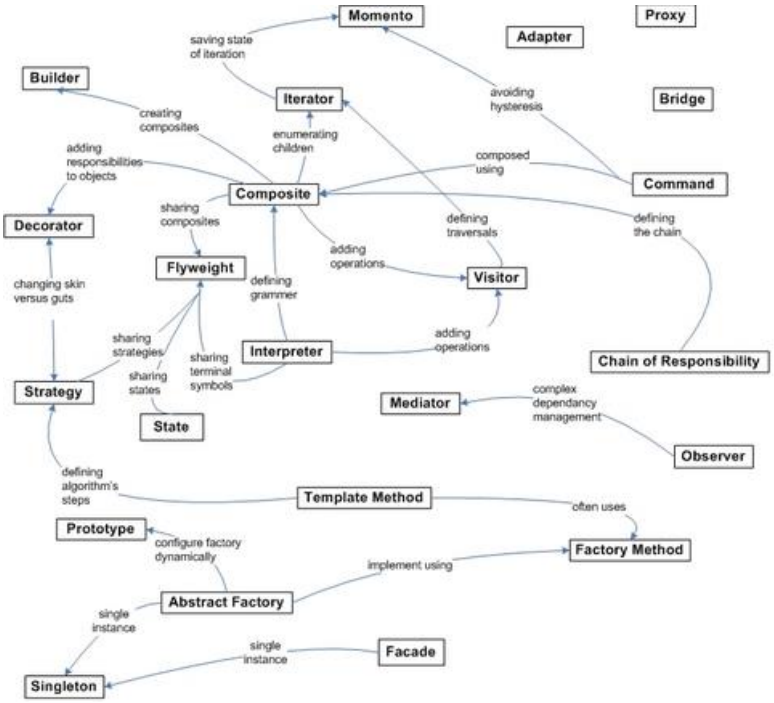
ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή εργασία

«ΕΜΠΕΙΡΙΚΗ ΜΕΛΕΤΗ ΓΙΑ ΤΗ ΧΡΗΣΗ ΠΡΟΤΥΠΩΝ ΣΧΕΔΙΑΣΗΣ ΣΕ ΠΑΙΧΝΙΔΙΑ ΑΝΟΙΧΤΟΥ ΛΟΓΙΣΜΙΚΟΥ»



Της φοιτήτριας
Χαραλαμπίδου Σοφίας
Αρ. Μητρώου: 05/2871

Επιβλέπων Καθηγητής
Αμπατζόγλου Απόστολος

Θεσσαλονίκη 2010

ΠΡΟΛΟΓΟΣ

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

Μεγάλη σημασία για την ανάπτυξη λογισμικού κατέχει η μέτρηση των ποιοτικών χαρακτηριστικών που επιδεικνύει ένα σύστημα, διαδικασία που μπορεί να συντελέσει σημαντικά στη βελτίωση της ποιότητας του λογισμικού. Το εργαλείο που χρησιμοποιείται για το σκοπό αυτό είναι οι μετρικές.

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

Η διεξαγωγή της παρούσας μελέτης αποδείχθηκε ιδιαίτερα χρήσιμη για την εξοικείωση του συγγραφέα με προηγμένα θέματα τεχνολογίας λογισμικού, τις μεθόδους εμπειρικής αξιολόγησης μιας μελέτης και τη συστηματική βιβλιογραφική ανασκόπηση ενός ερευνητικού πεδίου. Επιπλέον, τα αποτελέσματα της εργασίας θεωρούνται σημαντικά και ευρύτερου επιστημονικού ενδιαφέροντος, από τη στιγμή που αξιολογούν τη χρήση των προτύπων σχεδίασης στην ποιότητα του λογισμικού, ζήτημα επίκαιρο στην κοινωνία ανάπτυξης λογισμικού. Τέλος τα ευρήματα της εργασίας χρησιμοποιήθηκαν ως μέρος ενός δημοσιευμένου επιστημονικού άρθρου, σε γνωστό συνέδριο του χώρου και μιας υπο κρίση εργασίας σε επιστημονικό περιοδικό.

ΠΕΡΙΛΗΨΗ

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

Η εργασία αυτή μελετά την επίδραση της εφαρμογής των προτύπων σχεδίασης στην ποιότητα του λογισμικού από δυο διαφορετικές σκοπιές και παραθέτει δυο άμεσα συσχετιζόμενες αλλά ανεξάρτητες μελέτες.

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

Στη συνέχεια, παρουσιάζουμε μια εμπειρική μελέτη, σχετικά με τη χρήση προτύπων σχεδίασης σε παιχνίδια ανοιχτού λογισμικού. Για τη μελέτη αυτή συλλέξαμε τα 10 δημοφιλέστερα παιχνίδια ανοιχτού λογισμικού των 10 δημοφιλέστερων κατηγοριών ενός γνωστού αποθετηρίου κώδικα (sourceforge), που πληρούσαν ορισμένες προδιαγραφές. Με τη χρήση ενός εργαλείου ανίχνευσης προτύπων σχεδίασης από object αρχεια Java εξορύξαμε πληροφορίες σχετικά με τη χρήση προτύπων σχεδίασης στα παιχνίδια αυτά. Τα δεδομένα που συγκεντρώθηκαν αναλύθηκαν με τη βοήθεια στατιστικών τεχνικών και προέκυψαν ενδιαφέροντα αποτελέσματα που παρουσιάζονται αναλυτικά στο κεφάλαιο 3.

ABSTRACT

The terms «design pattern» and «software quality» are two interesting fields of software engineering and they present high research activity. Patterns and quality are highly related to each other, while design patterns are applied to software, aiming to improve its quality. Metrics are a kind of tool, used to measure the quality attributes, which characterize the software system. In our research we aimed at assessing the effect of patterns on software quality. This project consists of two independent, but correlated, studies investigating the effect of design patterns' application on software quality.

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

Then, we conducted an empirical study on design patterns' employment on open-source games. During this study, we collected the 10 most popular open-source games from the 10 most popular game categories, found in a well-known repository (sourceforge). By using a mining tool for identifying design patterns from Java bytecode we collected information about the use of design patterns in these game categories. The collected data were analyzed through statistic techniques and the results are presented in Chapter 3.

ΕΥΧΑΡΙΣΤΙΕΣ

Ένα μεγάλο ευχαριστώ στον επιβλέποντα καθηγητή αυτής της πτυχιακής εργασίας, κ.Αμπατζόγλου Απόστολο, που από την πρώτη στιγμή της ανάθεσης βρίσκονταν δίπλα μου, σαν συνεργάτης και δάσκαλος. Στο μεράκι και την υπομονή του οφείλω την πρώτη μου δημοσίευση σε επιστημονικό συνέδριο, και οφείλω ένα θερμό ευχαριστώ, που δε δίστασε να μου προσφέρει την ευκαιρία αυτή.

Ένα ευχαριστώ και σε όλους όσους, τους δύσκολους αυτούς μήνες εκπόνησης της πτυχιακής μου εργασίας, στάθηκαν δίπλα μου υπομονετικά δείχνοντας άπειρη κατανόηση και προθυμία να με στηρίξουν με οποιονδήποτε τρόπο.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ - ΣΧΗΜΑΤΩΝ

1. ΕΙΣΑΓΩΓΗ

Στις μέρες μας, η χρήση προτύπων σχεδίασης συνηθίζεται κατά τη φάση της ανάπτυξης λογισμικού, αφού τα πρότυπα παρέχουν, επαναχρησιμοποιήσιμες και τεκμηριωμένες λύσεις σε συνηθισμένα σχεδιαστικά προβλήματα. Παρότι μεγάλη ποικιλία μελετών έχουν εξετάσει πολλές πτυχές των προτύπων σχεδίασης, καμία έρευνα μέχρι σήμερα δεν προσπάθησε να συγκεντρώσει την τρέχουσα κατάσταση της έρευνας μέσω ανασκόπηση της βιβλιογραφίας. Ένα άλλο φαινόμενο της εποχής μας, είναι η ακμή των κοινωνιών ανοιχτού λογισμικού που έχει ως αποτέλεσμα να αυξάνεται ο αριθμός των εφαρμογών που διατίθενται μέσω πολύ γνωστών αποθετηρίων κώδικα με γρήγορους ρυθμούς. Το μέγεθος του κώδικα που διατίθεται ελεύθερα στους προγραμματιστές διευκολύνει στο να γίνεται μεγάλη επαναχρησιμοποίηση κώδικα. Μια από τις μεγαλύτερες ανησυχίες των προγραμματιστών όταν επαναχρησιμοποιούν κώδικα είναι η ποιότητα του κώδικα που πρόκειται να επαναχρησιμοποιήσουν. Τα πρότυπα σχεδίασης θεωρείται ότι παρέχουν ουσιαστικά πλεονεκτήματα όσον αφορά την ποιότητα του λογισμικού.

Το πρώτο μέρος της εργασίας, παρουσιάζει μια επισκόπηση των ερευνητικών προσπαθειών όσον αφορά τα αντικειμενοστραφή πρότυπα σχεδίασης. Τα αποτελέσματα της έρευνας δείχνουν ότι η μεγαλύτερη ανησυχία των ερευνητών είναι ο αντίκτυπος της εφαρμογής προτύπων σχεδίασης στην ποιότητα του λογισμικού. Έτσι διεξήχθη μια συστηματική βιβλιογραφική αναφορά, με σκοπό την μελέτη της εφαρμογής των προτύπων σχεδίασης στα εσωτερικά και εξωτερικά χαρακτηριστικά της ποιότητας λογισμικού. Επιπλέον, παρουσιάζονται οι απλούστερες εναλλακτικές, σχεδιαστικές λύσεις που ισοδυναμούν με τα πρότυπα σχεδίασης. Τέτοιος σχεδιασμός μπορεί να χρησιμοποιηθεί σε ερευνητικές προσπάθειες που στοχεύουν στην αξιολόγηση των προτύπων σχεδίασης.

Στο δεύτερο μέρος της εργασίας, ερευνούμε το κατά πόσο χρησιμοποιούνται τα πρότυπα σχεδίασης σε εφαρμογές ανοιχτού λογισμικού. Ποιο συγκεκριμένα, αυτή η μελέτη εκθέτει εμπειρικά αποτελέσματα βασισμένα στον αριθμό και το είδος των προτύπων σχεδίασης που προέκυψαν από εφαρμογές ανοιχτού λογισμικού. Μέχρι τώρα έχουν μελετηθεί εκατό (100) παιχνίδια ανοιχτού λογισμικού με διαφορετικά χαρακτηριστικά. Τα αποτελέσματα της έρευνας δείχνουν ότι κάποια πρότυπα σχεδίασης χρησιμοποιούνται συχνότερα από άλλα, ενώ κάποια πρότυπα εφαρμόζονται πιο συχνά σε συγκεκριμένες κατηγορίες παιχνιδιών και ότι το

μέγεθος ενός προγράμματος, ο αριθμός των λήψεων, ο αριθμός των ημερών που η εφαρμογή ήταν ενεργή και ο αριθμός των προγραμματιστών είναι σημαντικοί παράγοντες που επηρεάζουν τη χρήση των προτύπων σχεδίασης.

1.1 ΑΝΟΙΧΤΟ ΛΟΓΙΣΜΙΚΟ

Η διαδικασία ανάπτυξης κώδικα ανοιχτού λογισμικού (OSS), που παρουσιάστηκε το 1998 (Feller και Fitzgerald, 2002), είναι μια σχετικά καινούρια τάση στην παραγωγή λογισμικού. Παρά το μικρό διάστημα ζωής της, η κοινωνία ανοιχτού λογισμικού μπορεί να επιδείξει ορισμένα πολύ πετυχημένα έργα με μεγάλη αποδοχή από την κοινωνία των υπολογιστών, όπως τα Linux, Apache Server και Mozilla Firefox.

Η ανάπτυξη μιας εφαρμογής ανοιχτού λογισμικού βασίζεται στη συνεργασία. Ένας μόνος προγραμματιστής, ή μία ομάδα προγραμματιστών ξεκινάει μια εφαρμογή και ανακοινώνει, μέσω του διαδικτύου, μια έκδοση που διατίθεται ελεύθερα, τόσο για χρήση όσο και για τροποποίηση. Έπειτα, η κοινωνία ανοιχτού λογισμικού επεκτείνει και συντηρεί την εφαρμογή. Αυτός ο τρόπος ανάπτυξης έχει και πλεονεκτήματα και μειονεκτήματα. Ένα μειονέκτημα της ανάπτυξης λογισμικού ανοιχτού κώδικα είναι η έλλειψη τεκμηρίωσης και τεχνικής υποστήριξης. Από την άλλη πλευρά τα βασικά πλεονεκτήματα του ανοιχτού λογισμικού είναι το χαμηλό κόστος, η αξιοπιστία και το γεγονός ότι παρέχει τον πηγαίο κώδικά των εφαρμογών στους χρήστες ώστε να μπορούν να προσαρμόσουν το λογισμικό σύμφωνα με τις δικές τους ανάγκες (Samoladas et al., 2004).

Επιπλέον το λογισμικό ανοιχτού κώδικα παρέχει μεγάλες δυνατότητες επαναχρησιμοποίησης, από την άποψη ότι διατίθεται ελεύθερα στους προγραμματιστές παρέχοντας μια ευρεία γκάμα κώδικα Προκειμένου ένα τμήμα κώδικα να μπορεί να χρησιμοποιηθεί εύκολα και επιτυχώς από μια άλλη εφαρμογή πρέπει να είναι κατανοητό, ευκολοσυντήρητο και ευέλικτο. Σε αυτό συμβάλλουν σημαντικά τα πρότυπα σχεδίασης. Στα (Meyer και Arnout, 2006, Arnout και Meyer, 2006) περιγράφεται πώς τα αντικειμενοστραφή σχεδιαστικά πρότυπα μπορούν να μετατραπούν σε επαναχρησιμοποιήσιμα συστατικά.

1.2 ΠΟΙΟΤΗΤΑ ΛΟΓΙΣΜΙΚΟΥ

Υπάρχουν διάφοροι τρόποι να ορίσει κανείς την έννοια της ποιότητας λογισμικού. Σύμφωνα με το (Kitchenham and Pfleeger, 1996) η ποιότητα είναι μια

σύνθετη και πολύπλευρη ιδέα που μπορεί να οριστεί από πέντε διαφορετικές οπτικές. Ορίζεται ως κάτι που μπορεί να αναγνωριστεί αλλά όχι να οριστεί. Από τη σκοπιά του χρήστη, η ποιότητα αφορά την καταλληλότητα ως προς κάποιον συγκεκριμένο σκοπό. Από κατασκευαστική άποψη, ποιότητα σημαίνει συμμόρφωση με τις προδιαγραφές. Από άποψη προϊόντος, η ποιότητα αφορά τη σύνδεση των έμφυτων χαρακτηριστικών του προϊόντος, ενώ έχοντας ως βάση της αξία, η ποιότητα εξαρτάται από το ποσό που ένας πελάτης είναι διατεθειμένος να πληρώσει.

Η ποιότητα του λογισμικού διαχωρίζεται σε δυο κατηγορίες, (α) την εσωτερική ποιότητα και (β) την εξωτερική ποιότητα (Kitchenham and Pfleeger, 1996). Η εσωτερική ποιότητα ασχολείται με τα χαρακτηριστικά του λογισμικού που δεν μπορούνε να παρατηρηθούν από το χρήστη ή τον προγραμματιστή, όπως για παράδειγμα η συνοχή, η πολυπλοκότητα και η σύζευξη. Η εσωτερική ποιότητα είναι μετρήσιμη μέσω των μετρικών, που μπορούν να λαμβάνονται απευθείας είτε από τον πηγαίο κώδικα, είτε από σχεδιαστικά τεχνουργήματα, όπως διαγράμματα κλάσεων ή ακολουθίας. Η εξωτερική ποιότητα ασχολείται με τις πτυχές του λογισμικού που γίνονται αντιληπτές είτε από τον προγραμματιστή είτε από τον χρήστη, όπως η ευκολία συντήρησης, η λειτουργικότητα και η χρηστικότητα. Τα χαρακτηριστικά της εξωτερικής ποιότητας είναι μη μετρήσιμα, και προκειμένου να έχουν πρόσβαση σε αυτά, οι προγραμματιστές χρησιμοποιούν ένα μοντέλο ποιότητας ώστε να αποτυπώσουν τα χαρακτηριστικά της εσωτερικής ποιότητας σε εξωτερικά χαρακτηριστικά.

Το μοντέλο αυτό περιλαμβάνεται στο διεθνές πρότυπο, ISO 9126, που έχει αναπτυχθεί με σκοπό την εκτίμηση της ποιότητας του λογισμικού. Το ISO 9126 προσδιορίζει ορισμένα ποιοτικά χαρακτηριστικά και τις μετρικές που τα υπολογίζουν. (Jung et al., 2004) Τα έξι βασικά χαρακτηριστικά που ορίζει το πρότυπο είναι η λειτουργικότητα, η αξιοπιστία, η χρηστικότητα, η αποτελεσματικότητα, η ευκολία για συντήρηση και η φορητότητα, ενώ ορίζει και 27 υποκατηγορίες που αναλύουν αυτά τα χαρακτηριστικά. Το πρότυπο ορίζει επίσης μετρικές για την μέτρηση των 27 χαρακτηριστικών. Για παράδειγμα, όσον αφορά το ποιοτικό χαρακτηριστικό της λειτουργικότητας, μπορεί να αναπαρασταθεί από τις τιμές των 5 υποκατηγοριών από τις οποίες συντίθεται, και χρησιμοποιώντας μια κατάλληλη μέθοδο σύνθεσης των αποτελεσμάτων να προκύψει μια τιμή που θα προσδιορίζει τη λειτουργικότητα του συστήματος. Επειδή το ποιοτικό μοντέλο που

παρέχει το πρότυπο είναι πολύ γενικό, μπορεί να εφαρμοστεί σε οποιοδήποτε λογισμικό.

Αναλύοντας τώρα τις μετρικές σε επίπεδο κώδικα, βασικό πλεονέκτημά τους είναι ότι παρέχουν στους προγραμματιστές διορατικότητα στο εσωτερικό του κώδικα που αναπτύσσουν και τους βοηθούν να κατανοήσουν ποιοι κομμάτια κώδικα πρέπει να ανακατασκευαστούν ή να ελεγχθούν σχολαστικά. Βοηθούν στην αναγνώριση ενδεχόμενων κινδύνων, στην κατανόηση της εκάστοτε κατάστασης ενός έργου και στην καταγραφή της προόδου κατά την ανάπτυξη του λογισμικού. Τέτοιες μετρικές είναι (1) η WMPC (Weighted Methods Per Class), που υπολογίζει την πολυπλοκότητα μιας κλάσης, μετρώντας τον αριθμό των μεθόδων που την αποτελούν, (2) η DIT (Depth of Inheritance Tree), που συσχετίζει τον βάθος μιας κλάσης στην ιεραρχία με την πολυπλοκότητα της, θεωρώντας ότι όσο μεγαλύτερο το βάθος στην ιεραρχία, τόσο πιο πιθανή η κληρονομικότητα κλάσεων υψηλότερων στην ιεραρχία, που έχει ως αποτέλεσμα την αύξηση του αριθμού των μεθόδων στην κλάση και συνεπώς την αύξηση της πολυπλοκότητας. (3) η NOC (Number Of Children) που μετράει πόσες υποκλάσεις πρόκειται να κληρονομήσουν τις μεθόδους των κλάσεων γονέων, (4) η CBO (Coupling Between Object Classes) που υπολογίζει το κατά πόσο οι κλάσεις εξαρτώνται από άλλες κλάσεις για να λειτουργήσουν ή είναι αυτόνομες, γεγονός που επηρεάζει πολύ την δυνατότητα επαναχρησιμοποίησης, (5) η RFC (Response For a Class), που μετράει τον αριθμό των μεθόδων που ανταποκρίνονται κατά τη λήψη ενός μηνύματος από μια κλάση. Όσο μεγαλύτερος ο αριθμός αυτός, τόσο πιο σύνθετη αναμένεται να είναι η διαδικασία της αποσφαλμάτωσης και του ελέγχου, αφού για τη λειτουργία της κλάσης εμπλέκονται και μέθοδοι που καλούνται από εξωτερικές κλάσεις, αυξάνοντας έτσι την πολυπλοκότητα. (6) η LCOM (Lack of Cohesion in Methods) που μετράει τη συνοχή των μεθόδων μιας κλάσης, και είναι επιθυμητή αφού προωθεί την ενθυλάκωση και μειώνει πολυπλοκότητα και την πιθανότητα λαθών κατά τη διαδικασία της ανάπτυξης (Chidamber and Kemerer, 1994). Οι μετρικές σε επίπεδο κώδικα χαρακτηρίζονται από μεγάλη ακρίβεια, αλλά μπορούν να υπολογιστούν μόνο κατά την φάση εκτέλεσης.

Αντίθετα, οι μετρικές σε σχεδιαστικό επίπεδο, δεν είναι τόσο ακριβείς, μπορούν όμως να υπολογιστούν νωρίτερα, και να παρέχουν ενδείξεις για την τελική ποιότητα του λογισμικού. Για να μπορέσουν να χρησιμοποιηθούν πρέπει το

αντικειμενοστραφές σχέδιο να περιέχει πληροφορίες, όπως ορισμό των κλάσεων, ιεραρχία των κλάσεων, διευκρινήσεις για τη λειτουργικότητα των μελών των κλάσεων και διευκρινήσεις σχετικά με τις παραμέτρους, τους τύπους, και τις ιδιότητες που χρησιμοποιούνται (Bansiya and Davis, 2002). Σύμφωνα με μια έρευνα που έχει διεξαχθεί, υπάρχουν ορισμένες μετρικές που μπορούν να τροποποιηθούν και να χρησιμοποιηθούν για την αξιολόγηση των σχεδιαστικών χαρακτηριστικών, όπως η αφαίρεση, η ανταλλαγή μηνυμάτων και η κληρονομικότητα. Υπάρχουν όμως και κάποια σχεδιαστικά χαρακτηριστικά, όπως η ενθυλάκωση και η σύνθεση για τα οποία δεν υπάρχουν αντικειμενοστραφείς σχεδιαστικές μετρικές. Μετρικές υπάρχουν ήδη για την μέτρηση της συνοχής και της σύζευξης που απαιτούν ωστόσο μια σχεδόν ολοκληρωμένη εφαρμογή των κλάσεων για να μπορέσουν να υπολογιστούν. Ορισμένες μετρικές σε επίπεδο σχεδίου είναι οι εξής: (1) η DSC (Design Size in Classes) που μετράει το σύνολο των κλάσεων στο σχέδιο. (2) η NOH (Number Of Hierarchies) που μετράει τον αριθμό των ιεραρχιών των κλάσεων στο σχέδιο. (3) η ANA (Average Number of Ancestors) που επισημαίνει τον μέσο όρο των κλάσεων από τις οποίες μια κλάση κληρονομεί πληροφορίες. Υπολογίζεται προσδιορίζοντας όλες τις κλάσεις ξεκινώντας από την ρίζα μιας ιεραρχικής δομής. (4) η DAM (Data Access Metric), το ποσοστό δηλαδή των ιδιωτικών ιδιοτήτων προς το σύνολο των ιδιοτήτων μιας κλάσης. Επιθυμητά είναι τα μεγάλα ποσοστά της μετρικής. (5) η DCC (Direct Class Coupling) που υπολογίζει των αριθμό των διαφορετικών κλάσεων με τις οποίες σχετίζεται άμεσα μια κλάση. Η μετρική περιλαμβάνει κλάσεις που σχετίζονται άμεσα με κάποια ιδιότητα, δήλωση ή παράμετρο μιας μεθόδου. (6) η CAM (Cohesion Among Methods of Class) που υπολογίζει τη συσχέτιση μεταξύ των μεθόδων μιας κλάσης και της λίστας παραμέτρων της. (7) η MOA (Measure Of Aggregation) που μετράει τον αριθμό των δηλωτικών δεδομένων που οι τύποι τους είναι κλάσεις ορισμένες από τον χρήστη. (8) η MFA (Measure of Functional Abstraction) το ποσοστό δηλαδή των μεθόδων που κληρονομούνται από μια κλάση προς τον συνολικό αριθμό των μεθόδων που είναι προσβάσιμες από τις μεθόδους μια κλάσης. (9) η NOP (Number of Polymorphic Methods) που μετράει τις μεθόδους που εμφανίζουν πολυμορφική συμπεριφορά. (10) η CIS (Class Interface Size) που μετράει των αριθμό των δημόσιων μεθόδων σε μια κλάση. (11) η NOM (Number Of Methods) που μετράει όλες τις μεθόδους που ορίζονται σε μια κλάση.

Τέλος πρέπει να αναφερθεί η σχέση προτύπων σχεδίασης και ποιότητας λογισμικού. Στη βιβλιογραφία, υπάρχει μεγάλη ποικιλία μελετών που επιχείρησαν να αξιολογήσουν την επιρροή της εφαρμογής προτύπων σχεδίασης, στην ποιότητα λογισμικού. Η μεγαλύτερη ανησυχία ενός προγραμματιστή που χρησιμοποιεί ένα πρότυπο σχεδίασης είναι ο αντίκτυπος που θα έχει η εφαρμογή του στην ποιότητα του λογισμικού. Αυτές οι μελέτες, κυρίως οι εμπειρικές, καταλήγουν στο ότι τα αντικειμενοστραφή πρότυπα σχεδίασης δεν είναι καθολικά ή καλά ή κακά.

1.3 ΠΡΟΤΥΠΑ ΣΧΕΔΙΑΣΗΣ

Τα πρότυπα σχεδίασης παρουσιάστηκαν αρχικά στον τομέα της αρχιτεκτονικής, από τον Christopher Alexander. Ο Alexander παρατήρησε ότι υπάρχουν συγκεκριμένα αρχιτεκτονικά σχεδιαστικά προβλήματα που μπορούν να διαχειριστούν με κοινές λύσεις. Έτσι κατέγραψε αυτά τα ζευγάρια προβλημάτων και λύσεων προτείνοντας την επαναχρησιμοποίηση τους για την επίτευξη καλών ποιοτικά σχεδίων (Alexander et al., 1977). Στα μέσα του ’90 η ιδέα των προτύπων υιοθετήθηκε από τους προγραμματιστές αντικειμενοστραφούς λογισμικού. Στο (Gamma et al., 1995) καταγράφονται 23 πρότυπα που επιλύουν συνήθη προβλήματα στη σχεδίαση λογισμικού. Τα τελευταία χρόνια, τα πρότυπα σχεδίασης εξακολουθούν να ελκύουν το ενδιαφέρον των ερευνητών και πλέον θεωρούνται ως ένα αξιοσέβαστο κομμάτι έρευνας της μηχανικής λογισμικού. Ωστόσο μέχρι σήμερα δεν έχουν σημειωθεί ερευνητικές προσπάθειες που να ανακεφαλαιώνουν τη μέχρι τώρα ερευνητική δραστηριότητα σε αυτό το πεδίο.

1.3.1 Factory

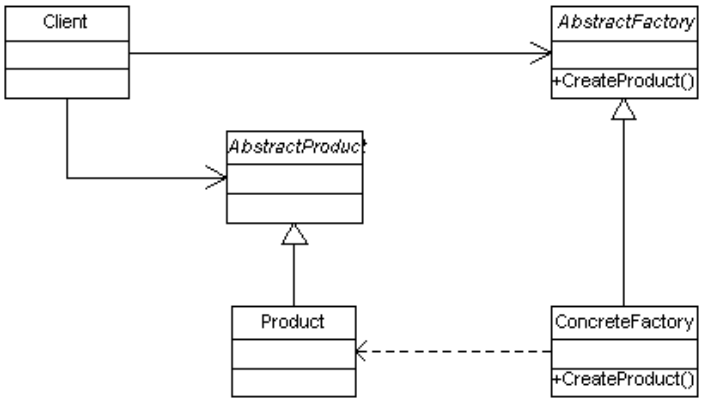
Το πρότυπο σχεδίασης Factory έχει ως σκοπό την παροχή μιας διασύνδεσης για τη δημιουργία οικογενειών, συσχετιζόμενων ή εξαρτημένων αντικειμένων, χωρίς να προσδιορίζεται η συγκεκριμένη κλάση τους (Chatzigeorgiou, 2005).

Ανήκει στην κατηγορία των κατασκευαστικών προτύπων (creational) και ως εκ τούτου επιτρέπει τη συγγραφή μεθόδων που δημιουργούν νέα αντικείμενα, χωρίς την άμεση χρήση ιδιωμάτων (π.χ. τελεστής new), όπως συμβαίνει στις αντικειμενοστραφείς γλώσσες προγραμματισμού. Το γεγονός αυτό επιτρέπει την ανάπτυξη μεθόδων που παράγουν ομάδες διαφορετικών αντικειμένων καθώς και

την επέκτασή τους για νέα αντικείμενα χωρίς την τροποποίηση του κώδικα των μεθόδων.

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

Γενική δομή



Σχήμα 1: Διάγραμμα κλάσεων προτύπου Factory

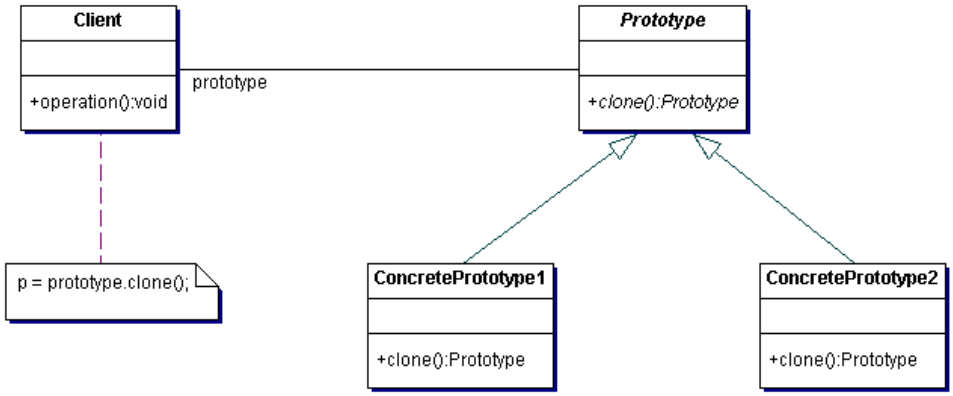
1.3.2 Prototype

Το πρότυπο Prototype προσδιορίζει τους τύπους των αντικειμένων και δημιουργεί στιγμιότυπα χρησιμοποιώντας πρωτότυπα, με σκοπό να δημιουργεί αντικείμενα αντιγράφοντας αυτά τα πρωτότυπα.

Το Prototype κατασκευάζει νέα αντικείμενα κοινοποιώντας κάποιο υπάρχον. Η κλωνοποίηση γίνεται μέσω μιας μεθόδου clone() η οποία παρέχεται από μια αφηρημένη κλάση ή διασύνδεση A και υλοποιείται σε κάθε παραγόμενη κλάση B, που κληρονομεί την A. Έτσι η κλήση της clone() σε ένα στιγμιότυπο της B επιστρέφει ένα αντίγραφο του εν λόγω στιγμιοτύπου, το οποίο αναλόγως με την υλοποίηση μπορεί να είναι είτε να περιέχει δείκτες προς τις εσωτερικές δομές δεδομένων του αρχικού στιγμιοτύπου, είτε να περιέχει πλήρως νεοδημιουργηθέντα αντίγραφα αυτών των δομών.

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

Γενική δομή



Σχήμα 2: Διάγραμμα κλάσεων προτύπου Prototype

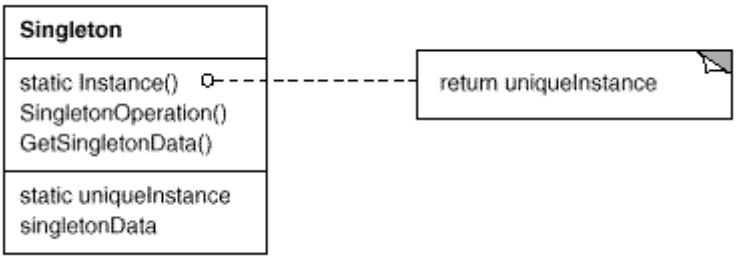
1.3.3 Singleton

Το πρότυπο Singleton εξασφαλίζει ότι μια κλάση θα έχει μόνο ένα στιγμιότυπο και παρέχει ένα καθολικό σημείο πρόσβασης σε αυτό (Chatzigeorgiou, 2005).

Συνήθως, μεταξύ των κλάσεων και των στιγμιοτύπων τους υπάρχει μια σχέση ένα προς πολλά. Κατά τη διαδικασία ανάλυσης, η ύπαρξη πολλών στιγμιοτύπων της ίδιας έννοιας στο σύστημα υποδηλώνει την αναγκαιότητα μιας κλάσης. Τα αντικείμενα δημιουργούνται δεσμεύοντας χώρο στη μνήμη, οπότε κρίνεται σκόπιμο και διαγράφονται όταν τερματιστεί η χρήση τους. Ορισμένες όμως φορές, απαιτείται η ύπαρξη κλάσεων από τις οποίες παράγεται ένα μόνο αντικείμενο. Συχνά, το αντικείμενο αυτό δημιουργείται κατά την έναρξη της εφαρμογής και διαγράφεται με το πέρας της. Ο ρόλος του μοναδικού αυτού αντικειμένου, είναι η διαχείριση των υπολοίπων αντικειμένων της εφαρμογής, και για το λόγο αυτό αποτελεί λογικό σφάλμα να δημιουργηθούν περισσότερα του ενός τέτοια αντικείμενα-διαχειριστές. Το πρότυπο περιλαμβάνει μια ειδική μέθοδο κατασκευής στιγμιοτύπων, που με την κλήση της, ελέγχει αν κάποιο αντικείμενο έχει ήδη δημιουργηθεί. Αν έχει όντως δημιουργηθεί, επιστρέφει απλώς ένα δείκτη προς το υπάρχον αντικείμενο. Διαφορετικά, δημιουργεί πρώτα το αντικείμενο και επιστρέφει και έναν δείκτη προς αυτό. Προκειμένου να εξασφαλιστεί ότι αυτός είναι ο μοναδικός τρόπος δημιουργίας αντικειμένων, ο κατασκευαστής της κλάσης δηλώνεται ως προστατευμένος (protected) ή ιδιωτικός (private).

Το πρότυπο Singleton χρησιμοποιείται όταν σε κάποιο σύστημα λογισμικού υπάρχει η απαίτηση από μια κλάση να δημιουργείται ένα και μόνο αντικείμενο.

Γενική δομή



Σχήμα 3: Διάγραμμα κλάσεων προτύπου Singleton

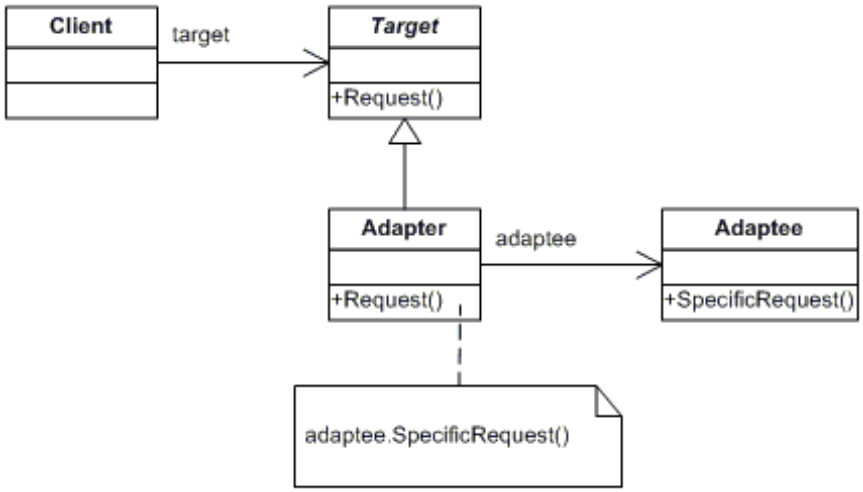
1.3.4 Adapter

Το πρότυπο σχεδίασης Adapter έχει ως στόχο την μετατροπή της διασύνδεσης μιας κλάσης σε μια άλλη που αναμένει το πρόγραμμα πελάτης. Έτσι, επιτρέπει τη συνεργασία κλάσεων, η οποία σε διαφορετική περίπτωση θα ήταν αδύνατη λόγω ασύμβατων διασυνδέσεων.

Συχνά ο κώδικας μιας κλάσης προσφέρεται για επαναχρησιμοποίηση, αλλά αυτή δεν είναι δυνατή, λόγω του ότι τα προγράμματα που επιθυμούν να χρησιμοποιήσουν τις λειτουργίες της, αναμένουν διαφορετική διασύνδεση. Στη συνήθη περίπτωση οπού τα προγράμματα πελάτες δεν είναι δυνατόν να τροποποιηθούν και η κλάση Σχεδίασης είναι επιθυμητό να χρησιμοποιηθεί χωρίς τροποποίηση, βρίσκει εφαρμογή το πρότυπο Adapter. Ένας προσαρμογέας κλάσης χρησιμοποιεί πολλαπλή κληρονομικότητα για να προσαρμόσει μια διασύνδεση σε μια άλλη, ενώ ένας προσαρμογέας αντικειμένου, βασίζεται στη σύνθεση αντικειμένων και στη διαβίβαση μηνυμάτων.

Το πρότυπο αυτό χρησιμοποιείται όταν θέλουμε να χρησιμοποιήσουμε μια υπάρχουσα κλάση, αλλά η διασύνδεσή της δεν συμβαδίζει με τις υπάρχουσες ανάγκες (Chatzigeorgiou, 2005).

Γενική δομή



Σχήμα 4: Διάγραμμα κλάσεων προτύπου Adapter

1.3.5 Composite

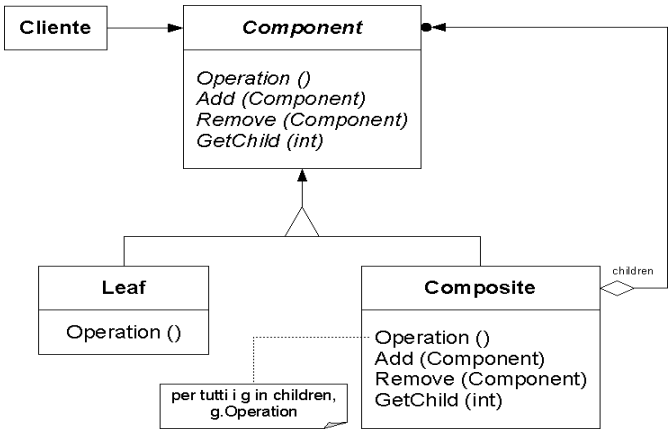
Το πρότυπο σχεδίασης Composite επιτρέπει τη σύνθεση αντικειμένων σε δενδροειδείς δομές για την αναπαράσταση ιεραρχιών τμήματος-όλου. Έτσι επιτρέπει στα προγράμματα πελάτες να διαχειρίζονται με ενιαίο τρόπο τόσο τα ανεξάρτητα αντικείμενα, όσο και σύνθετα αντικείμενα.

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

Το σημείο κλειδί στο πρότυπο είναι η ύπαρξη μιας αφηρημένης κλάσης που αναπαριστά τόσο τις πρωταρχικές όσο και τις περικλείουσες κλάσεις. Έτσι είναι δυνατή η δημιουργία οποιουδήποτε πρωταρχικού ή σύνθετου αντικειμένου επιτρέποντας ομοιόμορφο χειρισμό των αντικειμένων από ένα μόνο πρόγραμμα πελάτη.

Ο χρήστης είναι σε θέση να δημιουργήσει οποιαδήποτε σύνθετη οντότητα και να την προσθέσει στην εφαρμογή. Η σχεδίαση ενός σύνθετου αντικειμένου ουσιαστικά συνιστάται στη σχεδίαση των επιμέρους τμημάτων του. Για το λόγο αυτό είναι επιθυμητή η ενιαία αντιμετώπιση όλων των αντικειμένων. Το πρότυπο Composite , επιτρέπει τον αναδρομικό ορισμό περιεκτικότητας, ώστε οι πελάτες να μην αντιλαμβάνονται τη διαφορά μεταξύ πρωταρχικών και σύνθετων αντικειμένων (Chatzigeorgiou, 2005).

Γενική δομή



Σχήμα 5: Διάγραμμα κλάσεων προτύπου Composite

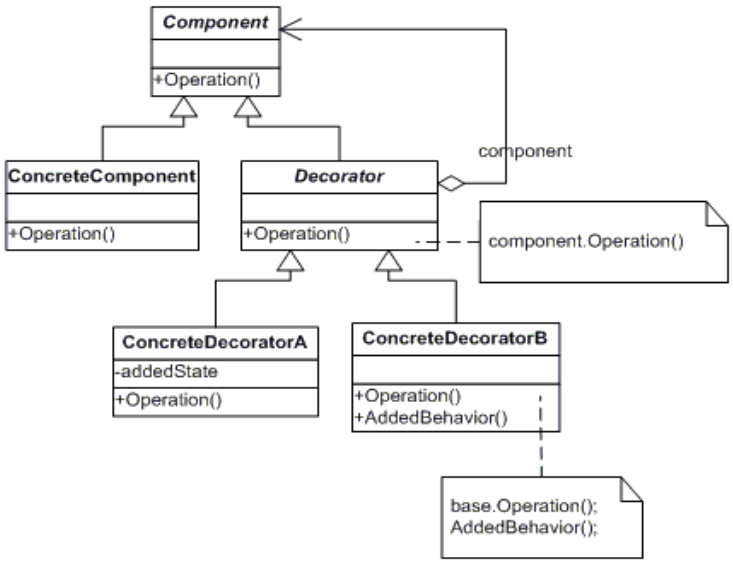
1.3.6 Decorator

Το πρότυπο Decorator περιγράφει έναν τρόπο για να προσθέσει κανείς δυναμικά ένα μη καθορισμένο αριθμό αρμοδιοτήτων σε ένα αντικείμενο. Επιτρέπει δηλαδή, την εύκολη και δυναμική επέκταση της λειτουργικότητας κάποιον υπαρχόντων κλάσεων, που υλοποιούν την ίδια διασύνδεση ή κληρονομούν την ίδια αφηρημένη κλάση, σε χρόνο εκτέλεσης

Αυτό επιτυγχάνεται μέσω μιας νέας κλάσης Decorator, που υλοποιεί την διεπαφή αλλά περιέχει ως ιδιωτικό πεδίο και μια αναφορά σε ένα στιγμιότυπο του γενικού τύπου της διεπαφής, η οποία τυπικά μεταβιβάζεται ως όρισμα στον κατασκευαστή της Decorator. Έτσι οι μέθοδοι της τελευταίας, υλοποιούν εσωτερικά την καινούρια λειτουργικότητα, αλλά για τις κοινές εργασίες καλούν τις αντίστοιχες μεθόδους της γενικής διεπαφής.

Το πρότυπο Decorator χρησιμοποιείται για την δυναμική επισύναψη αρμοδιοτήτων σε ένα αντικείμενο και παρέχει μια ευέλικτη εναλλακτική αντί της δημιουργίας υποκλάσεων για την επέκταση της λειτουργικότητας, αποφεύγοντας έτσι τον κίνδυνο πολλαπλασιασμού των υποκλάσεων (Gamma et al., 1995).

Γενική δομή



Σχήμα 6: Διάγραμμα κλάσεων προτύπου Decorator

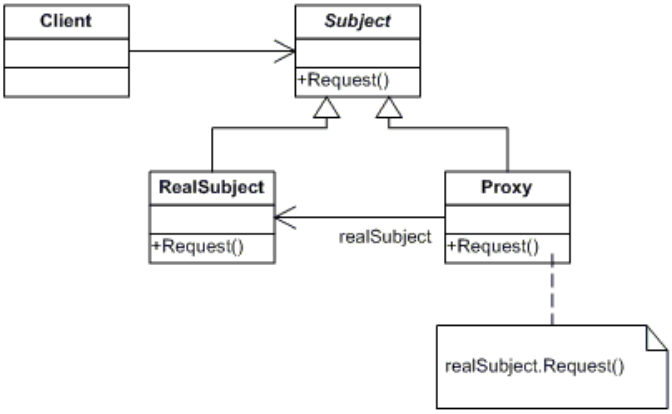
1.3.7 Proxy

Το πρότυπο Proxy λειτουργεί ως υποκατάστατο ή ως ένας τρόπος κράτησης της θέσης ενός άλλου αντικειμένου.

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

Ένας λόγος ελέγχου της πρόσβασης σε ένα αντικείμενο, είναι για να διαφοροποιηθεί το κόστος δημιουργίας του και να επιτρέπεται η αρχικοποίηση του μονάχα όταν πρόκειται να χρησιμοποιηθεί. Όταν το πρότυπο Proxy αναπαριστά ένα μεγάλο αντικείμενο, μέχρις ότου ζητηθεί από την εφαρμογή να φορτωθεί το ίδιο το αντικείμενο, επιτυγχάνεται μεγαλύτερη ταχύτητα και λιγότερη φόρτωση της μνήμης της εφαρμογής που χρησιμοποιεί το πρότυπο (Gamma et al., 1995).

Γενική δομή



Σχήμα 7: Διάγραμμα κλάσεων προτύπου Proxy

1.3.8 Observer

Το πρότυπο σχεδίασης Observer ορίζει μια σχέση εξάρτησης ένα προς πολλά, μεταξύ αντικειμένων, έτσι ώστε όταν μεταβάλλεται η κατάσταση ενός αντικειμένου, όλα τα εξαρτώμενα αντικείμενα να ενημερώνονται και να τροποποιούνται αυτόματα.

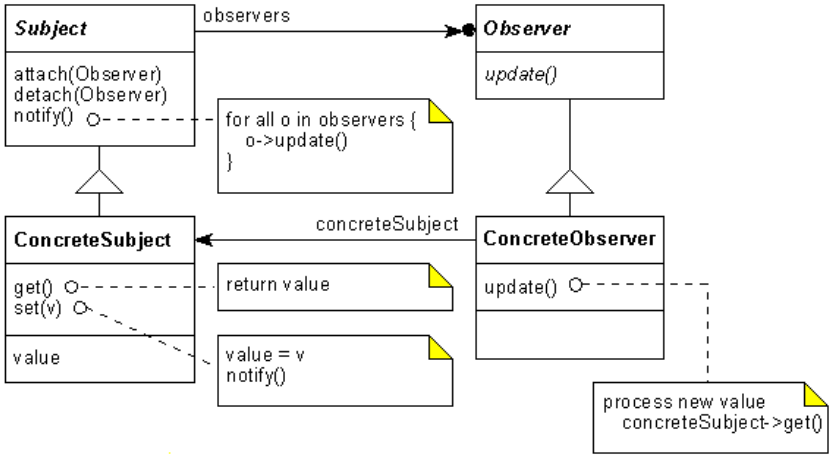
Το πρότυπο αυτό επιδιώκει την μείωση της σύζευξης μεταξύ των αντικειμένων, παρέχοντας αυξημένη δυνατότητα επαναχρησιμοποίησης και τροποποίησης του συστήματος. Επίσης, επιτρέπει την αυτόματη ειδοποίηση και ενημέρωση ενός συνόλου αντικειμένων τα οποία αναμένουν ένα γεγονός, που εκδηλώνεται ως αλλαγή στη κατάσταση ενός αντικειμένου. Στόχος, είναι η από-σύζευξη των παρατηρητών από το παρακολουθούμενο αντικείμενο , έτσι ώστε κάθε φορά που προστίθεται ένας νέος παρατηρητής (ενδεχομένως με διαφορετική διασύνδεση), να μην απαιτούνται αλλαγές στο παρακολουθούμενο αντικείμενο.

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

Γενική δομή



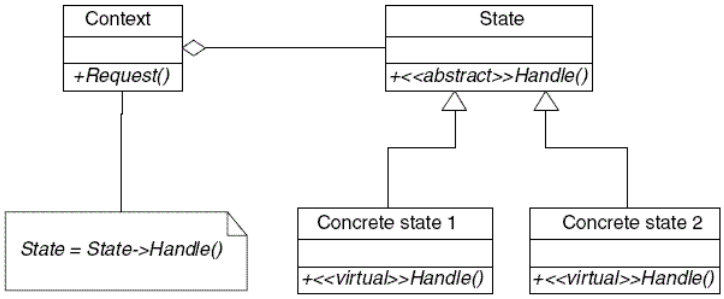
Σχήμα 8: Διάγραμμα κλάσεων προτύπου Observer

1.3.9 State

Το πρότυπο State ενθυλακώνει την κατάσταση ενός αντικειμένου, ώστε να μπορεί να αλλάξει τη συμπεριφορά του, όταν αλλάξει η εσωτερική κατάσταση του αντικειμένου. (Gamma et al., 1995).

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

Γενική δομή



Σχήμα 9: Διάγραμμα κλάσεων προτύπου State

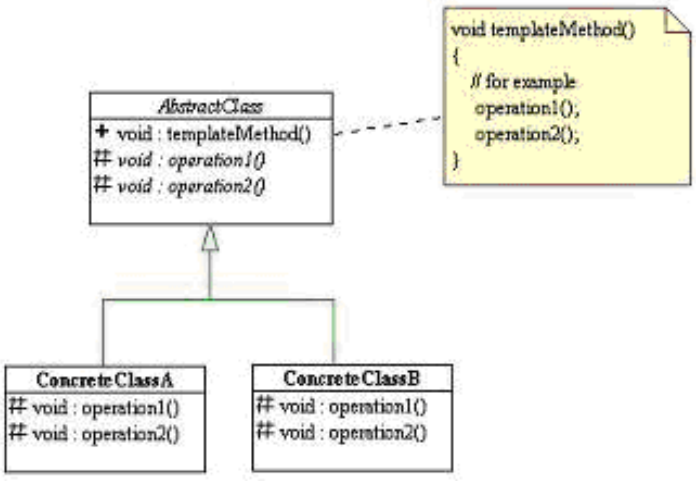
1.3.10 Template

Το πρότυπο σχεδίασης Template ορίζει το περίγραμμα ενός αλγορίθμου σε μια λειτουργία, επιτρέποντας στις παράγωγες κλάσεις να επαναορίσουν ορισμένα βήματα του αλγορίθμου, χωρίς να αλλάξουν τη δομή τους.

Στόχος είναι ο διαχωρισμός ενός γενικού αλγορίθμου σε συγκεκριμένες υλοποιήσεις, εκμεταλλευόμενος το μηχανισμό της κληρονομικότητας. Το πρότυπο Template εφαρμόζεται πολύ συχνά, ακόμα και όταν δε γίνεται αντιληπτό ως ξεχωριστή τεχνική. Το πιο κλασικό παράδειγμα εφαρμογής του προτύπου είναι στους αλγορίθμους ταξινόμησης.

Το πρότυπο Template χρησιμοποιείται για τον ορισμό των αμετάβλητων τμημάτων και τη μετάθεση της υλοποίησης των μεταβλητών τμημάτων του αλγορίθμου σε παράγωγες κλάσεις (Chatzigeorgiou, 2005).

Γενική δομή



Σχήμα 10: Διάγραμμα κλάσεων προτύπου Template

1.3.11 Visitor

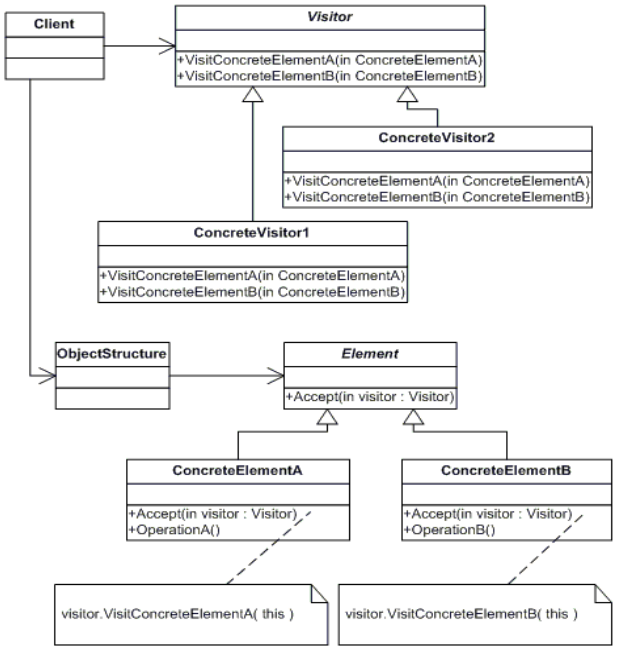
Το πρότυπο σχεδίασης Visitor έχει ως στόχο την αναπαράσταση μιας λειτουργίας που πρόκειται να πραγματοποιηθεί στα στοιχεία μιας δομής αντικειμένων. Το πρότυπο επιτρέπει τον ορισμό μιας νέας λειτουργίας χωρίς την τροποποίηση των κλάσεων των στοιχείων στα οποία επιδρά.

Συχνά απαιτείται η προσθήκη μιας νέας μεθόδου σε μια υπάρχουσα ιεραρχία κλάσεων, αλλά είναι εξαιρετικά δύσκολο να τροποποιηθούν οι ίδιες οι κλάσεις της ιεραρχίας. Η προσέγγιση αυτή ενθαρρύνει τη σχεδίαση ιεραρχιών από στοιχεία «ελαφρού τύπου» καθώς οι αντίστοιχες κλάσεις έχουν περιορισμένες αρμοδιότητες. Νέα λειτουργικότητα μπορεί εύκολα να προστεθεί στη νέα ιεραρχία μόνο με τη δημιουργία μιας νέας υποκλάσης στο πρότυπο επισκέπτης.

Το πρότυπο αυτό έχει μεγάλη πολυπλοκότητα στη λειτουργία του, καθώς υλοποιεί τη λεγόμενη «διπλή αποστολή», κατά την οποία η λειτουργία που εκτελείται εξαρτάται από το όνομα της αίτησης και τον τύπο των δυο αποδεκτών (δηλαδή του Επισκέπτη και του στοιχείου που επισκέπτεται).

Το πρότυπο Visitor χρησιμοποιείται για την προσθήκη λειτουργιών στα αντικείμενα μιας ιεραρχίας αντικειμένων χωρίς να χρειαστεί η προσθήκη των λειτουργιών μέσα στις κλάσεις (Chatzigeorgiou, 2005).

Γενική δομή



Σχήμα 11: Διάγραμμα κλάσεων προτύπου Visitor

1.4 ΜΕΘΟΔΟΛΟΓΙΑ ΣΥΣΤΗΜΑΤΙΚΗΣ ΑΝΑΣΚΟΠΗΣΗΣ ΤΗΣ ΒΙΒΛΙΟΓΡΑΦΙΑΣ

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

Κατά τη φάση του σχεδιασμού πρέπει να καθοριστούν τα ερευνητικά ερωτήματα, να ανεπτύχθη ένα πρωτόκολλο ανασκόπησης και στο τέλος να επικυρωθεί. Ο καθορισμός των ερευνητικών ερωτημάτων, αποτελεί το ποιο κρίσιμο στοιχείο της συστηματικής ανασκόπησης. Τα ερευνητικά ερωτήματα χρησιμοποιούνται για να εντοπίσουν τις λέξεις κλειδιά που πρέπει να χρησιμοποιηθούν για την αυτόματη αναζήτηση των σχετικών ερευνών. Οι λέξεις αυτές, ουσιαστικά καθορίζουν τα δεδομένα που πρέπει να εξαχθούν από κάθε πρωταρχική μελέτη και περιορίζουν τη συνολική διαδικασία. Τα ερωτήματα της έρευνας είναι το κομμάτι του πρωτοκόλλου που δεν μπορεί να τροποποιηθεί μετά τη αποδοχή του πρωτοκόλλου. Το πρωτόκολλο ανασκόπησης που αναπτύσσεται, δίνει πληροφορίες για τον σχεδιασμό της ανασκόπησης, συμπεριλαμβάνοντας την προδιαγραφή της διαδικασίας που θα ακολουθηθεί, τις συνθήκες και τις μετρικές ποιότητας που θα εφαρμοστούν όταν επιλεγεί μια πρωταρχική μελέτη, και την κατανομή ορισμένων ενεργειών. Η επικύρωση του πρωτοκόλλου θεωρείται απαραίτητη, αφού το πρωτόκολλο αποτελεί κρίσιμο στοιχείο της ανασκόπησης.

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

πρωταρχικές μελέτες και να ανασυντεθούν με τρόπο τέτοιο, ώστε να απαντούν στα ερωτήματα της έρευνας.

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

Σύμφωνα με τα (Kitchenham, Joint Technical Report και Kitchenham et al., 2009) το σχεδιάγραμμα της ανασκόπησης αποτελείται από έξι μέρη: (α) ορισμό των ερευνητικών ερωτημάτων, (β) ορισμό της διαδικασίας αναζήτησης, (γ) ορισμό των κριτηρίων συμπερίληψης και αποκλεισμού, (δ) ορισμό της ποιοτικής αξιολόγησης, (ε) ορισμό της διαδικασίας συλλογής δεδομένων, και (στ) ορισμό της ανάλυσης δεδομένων.

1.5 ΕΜΠΕΙΡΙΚΕΣ ΜΕΛΕΤΕΣ

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

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

που θέλουμε να μελετήσουμε. Τα αποτελέσματα της έρευνας αναλύονται και στην τελική φάση γενικεύονται για τον πληθυσμό απ' όπου προέρχονταν το δείγμα.

Στο κεφάλαιο 3 παρουσιάζεται μια μελέτης περίπτωσης. Η πληθώρα των εφαρμογών ανοιχτού κώδικα αναγάγει την μελέτη περίπτωσης στη βέλτιστη προσέγγιση έρευνας. Αντίθετα, μία μελέτη πεδίου δεν θα ταίριαζε στη δική μας έρευνα διότι θα οδηγούσε στο να μην συνυπολογίσουμε τα πρότυπα εκείνα που χρησιμοποιήθηκαν από τους προγραμματιστές χωρίς πρόθεση, τις περισσότερες φορές κατά τύχη. Τέλος ένα πείραμα που θα βασίζονταν σε προγραμματιστές ανοιχτού λογισμικού θα μείωνε τον αριθμό των παρατηρήσεων στην έρευνα μας.

1.5.1 Μελέτες Περίπτωσης

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

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

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

1.5.2 Μεθοδολογία Σύνταξης Μελέτης Περίπτωσης

Σύμφωνα με το (Kitchenham et al., 1995) τα βήματα που απαιτούνται για να συντάξει κανείς μια μελέτη περίπτωσης περιλαμβάνουν: (α) Ορισμό μιας υπόθεσης. (β) Επιλογή ορισμένων εφαρμογών, (γ) Επιλογή της μεθόδου σύγκρισης, (δ) Ελαχιστοποίηση των παραγόντων σύγχυσης, (ε) Σχεδιασμό πλάνου για την μελέτη περίπτωσης, (στ) Παρακολούθηση της μελέτης περίπτωσης, (ζ) Ανάλυση και αναφορά των αποτελεσμάτων

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

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

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

Το τέταρτο βήμα αφορά την ελαχιστοποίηση των παραγόντων σύγχυσης. Τέτοιοι παράγοντες μπορεί να είναι η εκμάθηση του τρόπου χρήσης μιας μεθόδου ή ενός εργαλείου κατά τη διάρκεια της προσπάθειας αξιολόγησής του, η χρήση είτε πολύ ενθουσιώδους είτε πολύ δύσπιστου προσωπικού σχετικά με τη χρήση της

μεθόδου ή του εργαλείου, η σύγκριση διαφορετικών τύπων εφαρμογών, κ.α. Ορισμένες φορές, μπορούμε να ελέγξουμε τη σύγχυση, μετρώντας τον παράγοντα σύγχυσης και ρυθμίζοντας τα αποτελέσματα ανάλογα.

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

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

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

2 ΑΝΑΣΚΟΠΗΣΗ ΤΗΣ ΒΙΒΛΙΟΓΡΦΙΑΣ: ΠΡΟΤΥΠΑ ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΟΙΟΤΗΤΑ

Το κεφάλαιο αυτό, στοχεύει στο να συνοψίσει την υπάρχουσα μέχρι σήμερα ερευνητική δραστηριότητα στο πεδίο των προτύπων σχεδίασης και να χρησιμοποιήσει τις αποδείξεις που βασίζονται στο παράδειγμα της μηχανικής λογισμικού (Kitchenham et al., 2009) προκειμένου να ανακαλύψει την επίδραση της εφαρμογής των προτύπων σχεδίασης στην ποιότητα του λογισμικού. Πρώτο βήμα, για να επιτευχθεί αυτό, είναι η διεξαγωγή μιας συστηματικής βιβλιογραφικής ανασκόπησης. Πρόσφατα, έχουν αυξηθεί οι ερευνητικές προσπάθειες για συστηματικές ανασκοπήσεις της βιβλιογραφίας (Ampatzoglou and Stamelos, 2010, Cai and Card, 2008, Dyba and Dingsoyr, 2008, Glass et al., 2002, και Kitchenham et al., 2009)

Στην ενότητα 2.1, με βάση τη μεθοδολογία που παρουσιάζεται στο κεφάλαιο 1.4, ορίζονται τα ερευνητικά ερωτήματα που θα διερευνήσει η μελέτη αυτή. Η ενότητα 2.2, παρουσιάζει διάφορα στατιστικά (descriptive statistics) για τα ευρήματα της έρευνας. Η συζήτηση σχετικά με τα αποτελέσματα της μελέτης, χωρίζεται σε υποενότητες βάση των ερευνητικών ερωτημάτων που διατυπώνονται στην ενότητα 2.1.1. Τέλος παρουσιάζονται οι κίνδυνοι εγκυρότητας στο κεφάλαιο 2.4.

2.1 ΜΕΘΟΔΟΛΟΓΙΑ ΑΝΑΣΚΟΠΗΣΗΣ

Σύμφωνα με την μεθοδολογία που περιγράφεται στο κεφάλαιο 1.4 παρουσιάζουμε τα έξι βήματα της συστηματικής ανασκόπησης της βιβλιογραφίας που διεξαγάγαμε στα πλαίσια αυτής της μελέτης.

2.1.1 Τα ερωτήματα της έρευνας

Στη μελέτη αυτή, θα διερευνήσουμε διάφορα ζητήματα που αφορούν την μέχρι σήμερα ερευνητική δραστηριότητα πάνω στα αντικειμενοστραφή πρότυπα σχεδίασης. Τα βασικά ερευνητικά ερωτήματα που αποτυπώνονται στην μελέτη είναι:

RQ1: Ποιά είναι τα πιο δημοφιλή ερευνητικά θέματα σχετικά με τα πρότυπα σχεδίασης;

RQ2: Ποια είναι η επίδραση των προτύπων σχεδίασης στην εξωτερική ποιότητα του λογισμικού;

RQ3: Ποια είναι η επίδραση των προτύπων σχεδίασης στην εσωτερική ποιότητα του λογισμικού;

RQ4: Ποια πρότυπα σχεδίασης έχουν ισοδύναμες απλούστερες σχεδιαστικές λύσεις;

2.1.2 Η διαδικασία αναζήτησης

Η διαδικασία αναζήτησης της σχετικής βιβλιογραφίας για την έρευνα αυτή, έχει βασιστεί στη διαδικασία που περιγράφεται στο (Cai and Card, 2008), οπού οι συγγραφείς συγκέντρωσαν ως πεδίο έρευνας, εφτά περιοδικά και εφτά συνέδρια . Τα περιοδικά επιλέχθηκαν σύμφωνα με το impact factor και το cited half life. Από τη άλλη, τα συνέδρια επιλέχθηκαν με πιο υποκειμενικά κριτήρια. Για την μελέτη αυτή, διερευνήσαμε τα περιοδικά και τα συνέδρια που αναφέρονται στο (Cai and Card, 2008), ενώ αποφασίσαμε να διερευνήσουμε επιπλέον ένα περιοδικό, τέσσερα συνέδρια και δύο workshops, που ασχολούνταν με την αντίστροφη μηχανική, την επεκτασιμότητα, την επανασχεδίαση, τις μετρικές και γενικά τη μηχανική λογισμικού. Τα περιοδικά και τα συνέδρια που μελετήθηκαν κατά την διαδικασία αναζήτησης της σχετικής βιβλιογραφίας παρουσιάζονται στον Πίνακα 1.

Πίνακας 1:Περιοδικά και Συνέδρια που περιλαμβάνονται στην ανασκόπηση.

id	Name	Type
1	IEEE Transactions on Software Engineering (TSE)	Περιοδικό
2	ACM Transactions on Software Engineering (TOSEM)	Περιοδικό
3	IEEE Software (SW)	Περιοδικό
4	Software Testing, Verification and Reliability (STVR)	Περιοδικό
5	Empirical Software Engineering (ESE)	Περιοδικό
6	Information and Software Technology (IST)	Περιοδικό
7	Journal of Systems and Software (JSS)	Περιοδικό
8	Science of Computer Programming (SCP)	Περιοδικό
9	International Conference on Software Engineering (ICSE)	Συνέδριο
10	International Symposium in Software Testing and Analysis (ISSTA)	Συνέδριο
11	International Symposium on Software Reliability Engineering (ISSRE)	Συνέδριο
12	International Conference on Automated Software Engineering (ASE)	Συνέδριο
13	ACM SIGSOFT Symposium on Foundation of Software Engineering (FSE)	Συνέδριο
14	Annual Computer Software and Application Conference (COMPSAC)	Συνέδριο
15	International Symposium on Empirical Software Engineering (ISESE)	Συνέδριο
16	IEEE Working Conference on Reverse Engineering (WCRE)	Συνέδριο
17	European Conference on Software Maintenance and Reengineering (CSMR)	Συνέδριο
18	International Conference on Software Maintenance (ICSM)	Συνέδριο

id	Name	Type
19	IEEE Metrics Symposium (METRICS)	Συνέδριο
20	FSE Workshops	Workshop
21	ICSE Workshops	Workshop

Η διαδικασία αναζήτησης διεξήχθη μέσω τους ιστοχώρους των εξής πέντε ψηφιακών βιβλιοθηκών: ACM, IEEE, ScienceDirect, Springer και Wiley. Σαν λέξη κλειδί για την αναζήτηση, χρησιμοποιήθηκε μονάχα ο όρος «πρότυπο», όταν βρίσκονταν στον τίτλο κάποιας δημοσίευσης. Το σύνολο των άρθρων που επιστράφηκαν από την προαναφερθείσα αναζήτηση αντιστοιχεί σε 367 άρθρα. Ωστόσο, πολλά από τα άρθρα αυτά θεωρήσαμε ότι δεν σχετίζονταν ικανοποιητικά με τα πρότυπα σχεδίασης του (Gamma et al., 1995). Ο αποκλεισμός των μη σχετικών άρθρων διεξήχθη με το χέρι, σύμφωνα με τα κριτήρια συμπερίληψης ή αποκλεισμού που ορίζονται στο επόμενο υποκεφάλαιο.

2.1.3 Κριτήρια Συμπερίληψης και Αποκλεισμού.

Τα άρθρα που επιλέχθηκαν ως πρωταρχικές μελέτες στην ανασκόπηση, έπρεπε να σχετίζονται με τα πρότυπα σχεδίασης που περιγράφονται στο (Gamma et al., 1995). Σύμφωνα με το (Dyba and Dingsoyr, 2008), υπάρχουν τέσσερα στάδια φιλτραρίσματος του συνόλου των άρθρων, προκειμένου να προκύψει το σύνολο των πρωταρχικών μελετών. Τα βήματα αυτά είναι τα εξής:

- Εύρεση των σχετικών μελετών – αναζήτηση σε ψηφιακές βιβλιοθήκες. (με την ολοκλήρωση του βήματος αυτού το σύνολο των άρθρων ανέρχονταν στα 367 άρθρα.)
- Αποκλεισμός μελετών βάση του τίτλο τους. (με την ολοκλήρωση του βήματος αυτού το σύνολο των άρθρων ανέρχονταν στα 216 άρθρα.)
- Αποκλεισμός μελετών βάση της περίληψή τους. (με την ολοκλήρωση του βήματος αυτού το σύνολο των άρθρων ανέρχονταν στα 141 άρθρα.) 10 άρθρα που είχαν πολύ μικρές ή δεν είχαν περιλήψεις, θεωρήθηκε ότι πρέπει να περάσουν στην επόμενη φάση για μελέτη του πλήρους κειμένου.
- Προσεκτική μελέτη των άρθρων και επιλογή των πιο σχετικών με τα πρότυπα σχεδίασης, βάση του πλήρους κειμένου. (με την ολοκλήρωση του βήματος αυτού το τελικό σύνολο πρωταρχικών μελετών αποτελούνταν από 107 άρθρα.)

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

Το τελικό σύνολο πρωταρχικών μελετών αποτελείται από τα ερευνητικά άρθρα που παρουσιάζονται στο Παράρτημα Β. Όλα τα άρθρα που μελετήθηκαν κατά τη φάση επιλογής των πρωταρχικών μελετών, παρουσιάζονται στο Παράρτημα Α.

2.1.4 Ποιοτική αξιολόγηση

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

2.1.5 Συλλογή δεδομένων

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

[A1] Τύπο δημοσίευσης (περιοδικό, συνέδριο, workshop)

[A2] Τόπο δημοσίευσης (όνομα συνεδρίου ή περιοδικού)

[A3] Έτος δημοσίευσης

[A4] Αντικείμενο Έρευνας

Για τις μελέτες που ασχολούνταν με την Ποιότητα Λογισμικού, καταγράφονταν επιπλέον πληροφορίες:

[Q1] Τα πρότυπα που βρέθηκαν

[Q2] Τα χαρακτηριστικά εξωτερικής ποιότητας που βρέθηκαν

[Q3] Τα χαρακτηριστικά εσωτερικής ποιότητας που βρέθηκαν

[Q4] Τις μετρικές λογισμικού που χρησιμοποιούνταν

[Q5] Το είδος της εμπειρικής μελέτης

[Q6] Αν παραθέτονταν ή όχι απλούστερες σχεδιαστικές λύσεις

2.1.6 Ανάλυση Δεδομένων

Τα δεδομένα που συλλέχθηκαν από τις μεταβλητές [A1] – [A3] και [Q5] χρησιμοποιήθηκαν για να παρέχουν περιγραφικά στατιστικά για την έρευνα στο πεδίο των προτύπων σχεδίασης.. Η μεταβλητή [A4] χρησιμοποιήθηκε για να εντοπιστούν τα πιο δημοφιλή ερευνητικά θέματα που ασχολούνται με τα πρότυπα σχεδίασης καθώς και για να βοηθήσει στην περιγραφή της μέχρι τώρα ερευνητικής διαδικασίας σε κάθε πεδίο (αντιμετωπίζει το RQ1).

Όσο αφορά την επίδραση του κάθε προτύπου σχεδίασης στην εσωτερική ή εξωτερική ποιότητα λογισμικού (αφορά τα RQ2 και RQ3), αξιοποιούνται οι μεταβλητές [Q1], [Q2], [Q3] και [Q4]. Τέλος, η μεταβλητή [Q6] χρησιμοποιήθηκε για τη συλλογή και παρουσίαση όλων των διαθέσιμων απλούστερων σχεδιαστικών λύσεων που ισοδυναμούν με κάποιο πρότυπο σχεδίασης (αντιμετωπίζει το RQ4). Το σύνολο των δεδομένων, αναλύεται και οποτικοποιείται με τη βοήθεια στατιστικών μεθόδων και γραφημάτων.

2.2 ΑΠΟΤΕΛΕΣΜΑΤΑ

Στο κεφάλαιο αυτό παρουσιάζονται τα περιγραφικά στατιστικά που προέκυψαν από την ανάλυση του συνόλου των δεδομένων. Στον πίνακα 2, συγκεντρώνουμε τον αριθμό των δημοσιεύσεων κατηγοριοποιώντας τες σύμφωνα με το πού δημοσιεύτηκαν. Το πιο δημοφιλές περιοδικό όσο αφορά τα πρότυπα σχεδίασης, φαίνεται να είναι το «IEEE Transactions on Software Engineering and Information and Software Technology». Από την άλλη πλευρά, τα «ACM Transactions on Software Engineering and Methodology» και «Science of Computer Programming» δεν έχουν καμία δημοσίευση σχετική με το θέμα. Αντίστοιχα, τα συνέδρια «COMPSAC», «ICSE» και «ICSM» παρουσιάζονται ως τα πιο ενεργά, ενώ τα «ISSTA» και «ISESE» δεν έχουν δημοσιεύσει κανένα άρθρο σχετικό με αντικειμενοστραφή πρότυπα σχεδίασης.

Ο πίνακας 3, παρουσιάζει την ερευνητική δραστηριότητα σε πιο εξειδικευμένα θέματα που αφορούν τα πρότυπα σχεδίασης. Κατά τη διάρκεια της ανασκόπησης μας, βρήκαμε τέσσερεις βασικούς τομείς στην έρευνα προτύπων σχεδίασης, που είναι οι εξής: (α) Τυποποίηση προτύπων, (β) ανάκτηση προτύπων, (γ) πρότυπα και ποιότητα και (δ) εφαρμογή προτύπων. Όσα άρθρα δεν μπορούσαν να κατηγοριοποιηθούν σε μία από τις κατηγορίες αυτές, τοποθετούνταν σε μια πέμπτη γενικευμένη κατηγορία.

Τα αποτελέσματα των πινάκων, δείχνουν ότι ο ισχυρότερος τομέας έρευνας για τα πρότυπα, είναι εκείνος που εξετάζει την επίδραση της εφαρμογής των προτύπων σχεδίασης στην ποιότητα λογισμικού, ενώ ακολουθούν οι έρευνες για μεθόδους ανάκτησης σχεδιαστικών προτύπων και τεχνικών για την τυποποίηση τους.

Πίνακας 2: Τόποι Δημοσίευσης

α/α	Όνομα	Δημοσιεύσεις	
1	TSE	7	6,54%
2	TOSEM	0	0,00%
3	SW	5	4,67%
4	STVR	1	0,93%
5	ESE	2	1,87%
6	IST	7	6,54%
7	JSS	6	5,61%
8	SCP	0	0,00%
9	ICSE	12	11,21%
10	ISSTA	0	0,00%
11	ISSRE	1	0,93%
12	ASE	6	5,61%
13	FSE	2	1,87%
14	COMPSAC	13	12,15%
15	ISESE	0	0,00%
16	WCRE	8	7,48%
17	CSMR	11	10,28%
18	ICSM	11	10,28%
19	METRICS	3	2,80%
20	FSE Workshops	1	0,93%
21	ICSE	11	10,28%

Πίνακας 3: Ερευνητικά Θέματα

α/α	Όνομα	Δημοσιεύσεις	
1	Formalization	21	19,63%
2	Detection	28	26,17%
3	Quality	31	28,97%
4	Application	15	14,02%
5	Miscellaneous	12	11,21%

Πίνακας 4: Εμπειρικές Μέθοδοι

α/α	Όνομα	Δημοσιεύσεις	
1	Survey	2	8,00%
2	Case Studies	15	60,00%
3	Experiments	8	32,00%

Σχετικά με τις μελέτες που αξιολογούν την επιρροή της εφαρμογής σχεδιαστικών προτύπων στην ποιότητα λογισμικού, προκύπτει ότι το 80.61% εφαρμόζουν μια εμπειρική μέθοδο έρευνας. Στο (Glass et al,. 2002) αναφέρεται ότι γενικότερα στις έρευνες της μηχανικής λογισμικού, το ποσοστό των μελετών που

χρησιμοποιούν εμπειρικές μεθόδους εγκυρότητας είναι μικρότερο του 20%. Έτσι, μπορούμε να υποθέσουμε, ότι χρειάζονται νέες διαφορετικές ερευνητικές προσεγγίσεις στον τομέα των προτύπων σχεδίασης και της ποιότητας λογισμικού.

Τέλος, στον Πίνακα 4, παρουσιάζουμε τη συχνότητα εφαρμογής κάθε εμπειρικής μεθόδου, δηλαδή μελέτης πεδίου, μελέτης περίπτωσης και πειράματος (Wohlin et al., 2000), που χρησιμοποιείται για την αξιολόγηση των προτύπων σχεδίασης από τη σκοπιά της ποιότητας λογισμικού. Παρατηρείται ότι κυρίαρχη μεθοδολογία είναι η «Μελέτη Περίπτωσης», ακολουθούν τα «πειράματα» και τέλος οι «μελέτες πεδίου». Τα αποτελέσματα αυτά, έρχονται και πάλι σε αντίθεση με το (Glass et al,. 2002) που ασχολείται γενικά με την μηχανική λογισμικού, και παρουσιάζει ως κυρίαρχη προσέγγιση τα ελεγχόμενα πειράματα.

2.3 ΣΥΖΗΤΗΣΗ

Στο κεφάλαιο αυτό, συζητάμε τα αποτελέσματα της ανασκόπησής μας, σύμφωνα με ερωτήματα τη έρευνας που έχουν διατυπωθεί. Στο κεφάλαιο 2.3.1 παρουσιάζουμε την μέχρι τώρα ερευνητική δραστηριότητα σχετικά με τα αντικειμενοστραφή πρότυπα σχεδίασης. Στα κεφάλαια 2.3.2 και 2.3.3 συζητάμε τα αποτελέσματα για την επίδραση των προτύπων στην εσωτερική και εξωτερική ποιότητα λογισμικού και στο κεφάλαιο 2.3.4 παρουσιάζουμε τις απλούστερες σχεδιαστικές λύσεις που παρέχουν λειτουργικά ισοδύναμα υποκατάστατα των προτύπων σχεδίασης.

2.3.1 Η ερευνητική δραστηριότητα μέχρι τώρα

Η ερευνητική δραστηριότητα μέχρι σήμερα, σχετικά με τα αντικειμενοστραφή πρότυπα σχεδίασης, χωρίζεται σε πέντε ερευνητικούς τομείς, όπως προέκυψε από την ανάλυση των δεδομένων μας. Ο πρώτος τομέας, ονομάζεται Τυποποίηση Προτύπων Σχεδίασης, και στοχεύει στη δημιουργία οντολογιών, γλωσσών σήμανσης κ.τ.λ., που μπορούν αν χρησιμοποιηθούν για την περιγραφή προτύπων σχεδίασης. Ο δεύτερος τομέας, ονομάζεται Ανάκτηση Προτύπων Σχεδίασης και περιλαμβάνει μελέτες που ασχολούνται με μεθοδολογίες, εργαλεία και αλγορίθμους εξόρυξης προτύπων σχεδίασης από πηγαίο κώδικα. Τρίτος, έρχεται ο τομέας Πρότυπα Σχεδίασης και Ποιότητα Λογισμικού, που ασχολείται με άρθρα που ερευνούν την επίδραση της εφαρμογής προτύπων σχεδίασης στην ποιότητα του λογισμικού. Ο τομέας Εφαρμογή Προτύπων Σχεδίασης περιλαμβάνει άρθρα

που παρουσιάζουν μεθόδους για την εύρεση συστημάτων που χρειάζονται εφαρμογή προτύπων, ή μεθόδους και εργαλεία που αυτοματοποιούν ή βοηθούν την εφαρμογή προτύπων. Ο τελευταίος τομέας έρευνας, αποτελείται από έρευνες που δεν μπορούν να κατηγοριοποιηθούν σε κάποια από τις άλλες κατηγορίες και για αυτό ονομάζεται Γενικά ζητήματα, σχετικά με τα πρότυπα σχεδίασης. Ακολουθεί μια αναλυτική περιγραφή της δραστηριότητας καθενός από τους τομείς αυτούς.

2.3.1.1 Τυποποίηση προτύπων σχεδίασης

Στην πρώτη κατηγορία ανήκουν άρθρα που σχετίζονται με διαδικασίες τυποποίησης των προτύπων. Στα άρθρα αυτά εξετάζονται ζητήματα όπως η μοντελοποίηση αρχιτεκτονικών προτύπων, νέοι τρόποι αναπαράστασης προτύπων, νέοι τρόποι καταγραφής προδιαγραφών, η βελτίωση των ανεπίσημων περιγραφών των προτύπων σχεδίασης, η κατανόηση των προτύπων σχεδίασης και η σύγκρισή τους με άλλες μεθόδους αναπαράστασης. Σκοπός είναι άλλοτε η αποκόμιση εμπειριών για τη διαδικασία καταγραφής των προδιαγραφών και άλλοτε η εύρεση καινοτόμων ιδεών ή η ανάπτυξη εργαλείων που θα διευκολύνουν και θα βελτιώσουν την χρήση των προτύπων. Πιο συγκεκριμένα, στα (Zdun και Avegriou, 2008 , Kim et al., 2003 , France et al., 2004 και Dong et al., 2007) παρουσιάζονται προσεγγίσεις που στοχεύουν στην οπτική αναπαράσταση ή σχεδιαστικών προτύπων με χρήση της UML. Το (Zdun και Avegriou, 2008) εξετάζει την δυνατότητα μοντελοποίησης αρχιτεκτονικών προτύπων, όσο αφορά τα συστατικά και τη σύνδεσή τους, μέσω κάποιων αρχιτεκτονικών αρχών. Η προσέγγιση έχει επικυρωθεί μοντελοποιώντας τις αρχές σε επεκτάσεις της UML, περνώντας τες στο εργαλείο Eclipse/Octopus και εφαρμόζοντάς τες σε ορισμένες μελέτες περίπτωσης. Έτσι δημιουργήθηκε ένα νέο πρωτότυπο επικύρωσης μοντέλων που θα μπορεί να υποστηρίξει την βασιζόμενη σε μοντέλα ανάπτυξη. Η συμβολή της προσέγγισης είναι η δημιουργία μιας γενικής και επεκτάσιμης ιδέας για τη μοντελοποίηση αρχιτεκτονικών προτύπων που παρέχονται με τη μορφή αρχιτεκτονικών αρχών. Στο (Kim et al., 2003) παρουσιάζεται μια προσέγγιση για τον προσδιορισμό προτύπων σχεδίασης, μετά τη μοντελοποίηση των συστατικών, χρησιμοποιώντας ρόλους. Στόχος είναι μια προσέγγιση εύκολη στη χρήση και πρακτική για την ανάπτυξη εργαλείων που ενσωματώνουν πρότυπα σε μοντέλα

UML. Η χρήση ρόλων των μοντέλων απεικονίζεται με την προδιαγραφή μιας παραλλαγής του προτύπου σχεδίασης Observer. Οι εμπειρίες που αποκομίζονται από το άρθρο υποδεικνύουν ότι η διαδικασία της καταγραφής των προδιαγραφών των προτύπων σχεδίασης δεν είναι δυσκολότερη από την διαδικασία σχεδιασμού μοντέλων UML. Το (France et al., 2004) παρουσιάζει μια πρακτική τεχνική για την καταγραφή των προδιαγραφών των λύσεων με χρήση προτύπων σε γλώσσα UML. Στο άρθρο περιγράφεται μια λύση που χρησιμοποιεί πρότυπα από δύο διαφορετικές οπτικές. Από κατασκευαστικής πλευράς περιγράφεται από ένα διάγραμμα κλάσεων και από άποψη αλληλεπίδρασης περιγράφεται από διαγράμματα ακολουθίας. Η τεχνική χρησιμοποιήθηκε για τη δημιουργία προδιαγραφών διαφόρων λύσεων για διάφορα πρότυπα σχεδίασης και η χρήση της απεικονίζεται με την καταγραφή προδιαγραφών για τα πρότυπα Observer and Visitor. Η προσέγγιση αυτή, ανοίγει το δρόμο για την ανάπτυξη εργαλείων που θα μπορούν να υποστηρίξουν αυστηρή εφαρμογή προτύπων σχεδίασης σε γλώσσα UML. Το (Dong et al., 2007) παρουσιάζει ένα προφίλ που ορίζει νέα στερεότυπα, ετικέτες και περιορισμούς για την ανίχνευση προτύπων σχεδίασης σε UML διαγράμματα. Με βάση το προφίλ αυτό, αναπτύχθηκε ένα διαδικτυακό εργαλείο για την οπτικοποίηση προτύπων σχεδίασης. Η αξιολόγηση της προσέγγισης γίνεται μέσω μιας μελέτης περίπτωσης και ενός συγκριτικού πειράματος με υπάρχουσες προσεγγίσεις. Το (Boussaidi και Mili, 2007) ασχολείται επίσης με τη μοντελοποίηση προτύπων, αλλά από λίγο διαφορετική σκοπιά, αφού έχει ως στόχο την δημιουργία ενός αποθετηρίου επαναχρησιμοποιήσιμων μοντέλων που θα προδιαγράφουν πρότυπα και ενός περιβάλλοντος οδηγούμενου από κανόνες για τη χρησιμοποίηση των μοντέλων αυτών. Το κρίσιμο σημείο της προσέγγισης είναι η ρητή αναπαράσταση των προβλημάτων που λύνουν τα πρότυπα σχεδίασης. Προτείνεται ότι τα πρότυπα σχεδίασης πρέπει να ορίζονται λεπτομερώς ως επαναχρησιμοποιήσιμα τεχνουργήματα που χαρακτηρίζονται από ένα μοντέλο προβλημάτων , ένα μοντέλο λύσεων και ένα σύνολο κανόνων που θα περιγράφουν τη μετατροπή που θα γίνει κατά την εφαρμογή τους.

Τα άρθρα (Taibi και Ngo, 2003, Bayley και Zhu, 2010, Bayley και Zhu, 2008 , Alencar et al., 1999 , Dong , 2004, Blewitt et al., 2005 , Baniassad et al., 2003, Eden et al., 1997, Soundarajan και Hallstrom, 2004 και Mens και Tourwe, 2003) περιγράφουν νέους τρόπους αναπαράστασης προτύπων σχεδίασης και καταγραφής προδιαγραφών. Τα (Taibi και Ngo, 2003, Bayley και Zhu, 2010 και

Bayley και Zhu, 2008) ασχολούνται με την προδιαγραφή προτύπων σχεδίασης χρησιμοποιώντας κατηγορήματα πρώτης τάξης και καταγράφουν τόσο δομικά χαρακτηριστικά όσο και χαρακτηριστικά συμπεριφοράς. Το (Taibi και Ngo, 2003) προτείνει μια απλή Γλώσσα Ισορροπημένης Προδιαγραφής Προτύπων (BPSL) που μπορεί να χρησιμοποιηθεί για προδιαγράψει επίσημα συνδυασμούς προτύπων. Προκειμένου να τονιστεί το ευρύ φάσμα εφαρμογής της γλώσσας χρησιμοποιούνται δυο μελέτες περίπτωσης . Το (Bayley και Zhu, 2010) παρουσιάζει μια επίσημη προσέγγιση που εντοπίζει και καταγράφει τις παραλλαγές των προτύπων σε μια καλά δομημένη διάταξη, μετά τη φάση της μοντελοποίησης των συστατικών. Το άρθρο εκθέτει μια μελέτη περίπτωσης που συμπεριλαμβάνει την επίσημη προδιαγραφή των 23 προτύπων του (Gamma et al., 1995), δίνοντας τη δυνατότητα επίσημης αιτιολόγησης για τα πρότυπα, την σύνθεσή τους και τον μετασχηματισμό τους και παρουσιάζει ένα αυτοματοποιημένο εργαλείο για την υποστήριξη της εφαρμογής των προτύπων κατά τη φάση του σχεδιασμού. Τα πλεονεκτήματα της προσέγγισης φαίνονται μέσω παραδειγμάτων και δικαιολογούνται με την παράθεση μελετών περίπτωσης και πειραματικών αποτελεσμάτων. Το (Bayley και Zhu, 2008) επεκτείνει μια προγενέστερη προσέγγιση αναπαριστώντας σε διαγράμματα ακολουθίας τη δυναμική συμπεριφορά των προτύπων. Η μελέτη περίπτωσης που χρησιμοποιείται προβάλλει ότι η μέθοδος βελτιώνει την ακρίβεια και επάρκεια της επίσημης προδιαγραφής των προτύπων σχεδίασης σχεδόν για όλα τα πρότυπα στο βιβλίο (Gamma et al., 1995), με εξαίρεση το πρότυπο Flyweight. Η βασική συνεισφορά του άρθρου είναι η έρευνα των χαρακτηριστικών συμπεριφοράς των προτύπων. Το (Alencar et al., 1999) προτείνει μια προσέγγιση σχετικά με τη βασισμένη στα συστατικά τεχνολογία λογισμικού, που στηρίζεται σε μια επίσημη περιγραφή των προτύπων σχεδίασης. Οι πληροφορίες αρχιτεκτονικής σχεδίασης που περιλαμβάνονται στα πρότυπα σχεδίασης, αναπαριστούνται ρητά με ξεκάθαρο τρόπο χρησιμοποιώντας Prolog. Η χρησιμότητα αυτής της προσέγγισης απεικονίζεται μέσω μιας μελέτης περίπτωσης που συμπεριλαμβάνει διάφορα πρότυπα σχεδίασης. Στο (Dong , 2004) παρουσιάζονται κάποιοι καινούριοι συμβολισμοί για την ρητή αναπαράσταση των κατασκευαστικών πτυχών και της συμπεριφοράς κάθε προτύπου κατά την εφαρμογή του ή τη σύνθεσή του με άλλα πρότυπα σχεδίασης. Οι συμβολισμοί αυτοί εφιστούν δυνατή την ύπαρξη των σχετικών με πρότυπα πληροφοριών σε διαγράμματα κλάσεων και συνεργασίας,

που είναι ιδιαίτερα χρήσιμα για την επαναχρησιμοποίηση της σχεδιαστικής εμπειρίας, τη βελτίωση στην επικοινωνία, τη σύλληψη σχεδιαστικών αποφάσεων και την καταγραφή σχεδιαστικών εναλλακτικών σε διαφορετικές εφαρμογές. Για την απεικόνιση της προσέγγισης χρησιμοποιείται μια μελέτη περίπτωσης. Το (Blewitt et al., 2005) παρουσιάζει την Spine, μια γλώσσα προδιαγραφής προτύπων που επιτρέπει τον ορισμό προτύπων χρησιμοποιώντας τους περιορισμούς που υπάρχουν κατά την υλοποίηση τους σε Java. Η Spine μοιάζει σαν γλώσσα με την Prolog και επιτρέπει τον ορισμό των προτύπων χρησιμοποιώντας ενσωματωμένες συναρτήσεις και κατηγορήματα. Στο άρθρο παρουσιάζονται κάποια παραδείγματα προτύπων που δείχνουν τον τρόπο που τα πρότυπα υποβάλλονται σε επεξεργασία. Τα αποτελέσματα δείχνουν ότι είναι δυνατό να οριστούν λεπτομερώς πρότυπα χρησιμοποιώντας τους δηλωτικούς περιορισμούς και ότι τα πρότυπα μπορούν να χρησιμοποιηθούν για να επαληθεύσουν την υλοποίηση κώδικα πραγματικών συστημάτων. Το (Baniassad et al., 2003) αφορά μια προσέγγιση που χρησιμοποιεί λογικούς γράφους προτύπων σχεδίασης (DPRG) και παρουσιάζει ένα σχετικό εργαλείο. Και τα δυο αναπτύχθηκαν για να καταστήσουν ικανό τον αποτελεσματικό εντοπισμό των σχεδιαστικών στόχων που σχετίζονται με ένα πρότυπο. Τέλος παρουσιάζονται ορισμένες μελέτες περίπτωσης που δείχνουν ότι ένα DPRG μπορεί με πολύ χαμηλό κόστος να βοηθήσει έναν προγραμματιστή. Το (Eden et al., 1997) περιγράφει μια μέθοδο για τον ακριβή προσδιορισμό του τρόπου που εφαρμόζεται ένα πρότυπο σχεδίασης, διατυπώνοντας το σαν αλγόριθμο σε μια μεταπρογραμματιστική γλώσσα. Ακολουθεί η παρουσίαση ενός πρωτότυπου εργαλείου, που υποστηρίζει την προδιαγραφή προτύπων σχεδίασης και την αυτόματη εφαρμογή τους σε ένα δοθέν πρόγραμμα. Η μελέτη περίπτωσης που περιγράφεται στο άρθρο αυτό δείχνει τον τρόπο που οι σχεδιαστικές αρχές μπορούν να διατυπωθούν επίσημα σε μια γλώσσα προδιαγραφής προτύπων. Μεταπρογραμματιστική γλώσσα χρησιμοποιείται και στο (Mens και Tourwe, 2003), το οποίο παρουσιάζει μια πολλά υποσχόμενη τεχνική δηλωτικού μεταπρογραμματισμού, που προσφέρει συνύπαρξη μιας αντικειμενοστραφούς γλώσσας και μιας δηλωτικής μεταγλώσσας. Στόχος του είναι παράσχει αυτοματοποιημένη υποστήριξη για την εξέλιξη των προτύπων σχεδίασης. Αυτό το δηλωτικό πλαίσιο μπορεί να χρησιμοποιηθεί για να προδιαγράψει πρότυπα σχεδίασης, τους περιορισμούς τους και τους υψηλού επιπέδου εξελικτικούς

μετασχηματισμούς τους. Εκτός αυτών οι προδιαγραφές επιτρέπουν την ανίχνευση εξελικτικών συγκρούσεων λογισμικού με υψηλό επίπεδο αφαίρεσης. Κλείνοντας την υποκατηγορία αυτή το (Soundarajan και Hallstrom, 2004) αναπτύσσει μια προσέγγιση για τυποποίηση των προτύπων σχεδίασης. Η προσέγγιση σχεδιάστηκε για να επιβεβαιώσει τη διατήρηση της ελαστικότητας κατά την τυποποίηση των προτύπων και απεικονίζεται με την εφαρμογή του προτύπου Observer και την ανάπτυξη μια προδιαγραφής για το πρότυπο αυτό.

Τρία άρθρα της κατηγορίας αυτής, τα (Ziane, 2000 , Mikkonen, 1998 , Eide et al., 2002) ασχολούνται με η βελτίωση των ανεπίσημων περιγραφών των προτύπων σχεδίασης. Συγκεκριμένα το (Ziane, 2000) στοχεύει στη βελτίωση των ανεπίσημων περιγραφών των προτύπων σχεδίασης χρησιμοποιώντας μετασχηματιστικούς κανόνες, που περιγράφουν τον τρόπο που λειτουργούν οι θεμελιώδεις μηχανισμοί των προτύπων. Αναλύεται το βασικό πρόβλημα των κατασκευαστικών προτύπων, και προτείνεται η χρήση μετασχηματιστικών κανόνων, δηλαδή μηχανισμών που μπορούν να δημιουργήσουν λύσεις για τα δυο πιο σημαντικά κατασκευαστικά πρότυπα, το Prototype και το Factory Method βελτιώνοντας έτσι την κατανόηση των προτύπων σχεδίασης και προωθώντας την υποστήριξή τους από εργαλεία. Το (Mikkonen, 1998) δείχνει ότι η ρητή τεκμηρίωση ενός προτύπου μπορεί να γίνει ευκολότερη τυποποιώντας τις προσωρινές συμπεριφορές των προτύπων με υψηλού επιπέδου αφαιρέσεις στον τομέα της επικοινωνίας και χρησιμοποιώντας βελτιώσεις που διατηρούν τις ιδιότητες τους. Στο άρθρο χρησιμοποιείται η μέθοδος προδιαγραφής Disco, που σκοπεύει σε προδιαγραφές και μοντελοποιήσεις των αλληλεπιδράσεων σε ένα υψηλό επίπεδο αφαίρεσης. Η γλώσσα που χρησιμοποιείται για τη σύνθεση των προδιαγραφών είναι σε μορφή κειμένου, ωστόσο από το συναφές εργαλείο παρέχονται και γραφικά. Το (Eide et al., 2002) περιγράφει μια συμπληρωματική υλοποίηση των προτύπων σχεδίασης, στην οποία πολλά πρότυπα που συμμετέχουν αντιστοιχούν σε συνδεδεμένα συστατικά που δημιουργούνται στατικά. Στο άρθρο ορίζεται μια συστηματική μέθοδος για την εφαρμογή της προσέγγισης σε υπάρχοντα πρότυπα. Η προσέγγιση που παρουσιάζεται απεικονίζεται στα πλαίσια του OSKit, μιας συλλογής συστατικών λειτουργικών συστημάτων γραμμένα σε C. Η προσέγγιση βοηθά στην αναγνώριση σχεδιαστικών συναλλαγών και επιτυγχάνει την ισορροπία μεταξύ ευελιξίας κατά τη σχεδιαστική φάση και την φάση εκτέλεσης.

Τα τελευταία 3 άρθρα της κατηγορίας τυποποίησης προτύπων επιδιώκουν την κατανόηση των προτύπων σχεδίασης και τη σύγκρισή τους με άλλες μεθόδους αναπαράστασης. Το (Porras και Gueheneuc, 2010) παρουσιάζει μια εμπειρική μελέτη που συγκεντρώνει δεδομένα σχετικά με την επίδοση των προγραμματιστών σε βασικά καθήκοντα που σχετίζονται με την κατανόηση των προτύπων σχεδίασης, αξιολογεί την επίδραση τριών οπτικών αναπαραστάσεων (Dong, Gamma, και Schauer) και τις συγκρίνει με την αναπαράσταση σε γλώσσα UML. Η ανάλυση των συγκεντρωμένων δεδομένων δείχνει ότι τα ενισχυμένα με στερεότυπα UML διαγράμματα φαίνεται να είναι πιο αποδοτικά από τα διαγράμματα αλληλεπίδρασης για τον προσδιορισμό σύνθεσης και ρόλων και ότι οι αναπαραστάσεις σε UML καθώς και τα ενισχυμένα με πρότυπα UML διαγράμματα κλάσεων είναι πιο αποτελεσματικά για τον εντοπισμό των κλάσεων που συμμετέχουν σε ένα πρότυπο σχεδίασης. Το (Winn και Calder, 2002) επιδιώκει γενικά την κατανόηση της σημασίας των προτύπων και της καταγραφής των σημαντικών χαρακτηριστικών τους με σκοπό την αποσαφήνιση και την καλύτερη κατανόηση τους, γεγονός που έχει αντίκτυπο στην ικανότητα αναγνώρισης και χρήσης των προτύπων. Τέλος το (Mak et al., 2004) παρουσιάζει τις κατασκευαστικές ιδιότητες των προτύπων σχεδίασης που αποκαλύπτουν την πραγματικά αφαιρετική φύση της δομής τους σε σύγκριση με τα συμβατικά αντικειμενοστραφή μοντέλα, τα υποδείγματα κλάσεων και τα πλαίσια. Στο άρθρο χρησιμοποιούνται δύο μελέτες περίπτωσης για να απεικονίσουν τον τρόπο που μοντελοποιείται ένα πρότυπο σχεδίασης.

2.3.1.2 Ανίχνευση προτύπων σχεδίασης

27 άρθρα που μελετήθηκαν για την εργασία αυτή, ασχολούνται με την ανίχνευση προτύπων σχεδίασης . Δυο από αυτά ασχολούνται με την ανάκτηση προτύπων συμπεριφοράς συνδυάζοντας τη στατική και δυναμική ανάλυση (De Lucia et al., 2009 και Wendehals και Orso, 2006), ενώ τρία άρθρα (Antoniol et al., 1998 , Antoniol et al., 2001 και De Lucia et al., 2007) με την ανάκτηση κατασκευαστικών προτύπων. Στο (A. De Lucia et al., 2009) αναφέρεται σε μια προκαταρκτική μελέτη κατα την οποία εφαρμόστηκε η διαδικασία ανάκαμψης στη βιβλιοθήκη JHotDraw προκειμένου να αξιολογηθεί η αποτελεσματικότητα ανάκτησής προτύπων σχεδίασης συμπεριφοράς. Σύμφωνα με τα αποτελέσματα,

όλες οι περιπτώσεις προτύπων Observer που είχαν καταγραφεί, αναγνωρίστηκαν ενώ παρατηρήθηκε ακρίβεια 55%. Στο (Wendehals και Orso, 2006) αναλύεται επίσης μια προκαταρκτική μελέτη που αξιολογεί τη δυνατότητα πραγματοποίησης της προσέγγισης εφαρμόζοντάς την σε μια πραγματική, σύνθετη εφαρμογή. Στόχος της προσέγγισης είναι να επιτευχθεί μια ακριβέστερη αναγνώριση προτύπων σχεδίασης. Στα (Antoniol et al., 1998 και Antoniol et al., 2001) παρουσιάζεται μια προσέγγιση πολλών επιπέδων για την εξαγωγή κατασκευαστικών προτύπων σχεδίασης από αντικειμενοστραφή αντικείμενα, είτε σχέδια είτε κώδικα. Για την αξιολόγηση της αποτελεσματικότητας των προσεγγίσεων, και στις δυο μελέτες έχει αναπτυχθεί ένα φορητό εργαλείο σε Java, που επιτρέπει την απομακρυσμένη πρόσβαση μέσω οποιασδήποτε μηχανής αναζήτησης ιστού. Στο πρώτο άρθρο κώδικας και σχέδιο χαρτογραφούνται σε μια ενδιάμεση αναπαράσταση, αποκαλούμενη γλώσσα αφηρημένων αντικειμένων (Abstract Object Language), για να διατηρήσουν την ανεξαρτησία τους από τη γλώσσα προγραμματισμού και τα αντίστοιχα CASE εργαλεία. Προκειμένου να εξαχθούν τα structural πρότυπα σχεδίασης, χρησιμοποιούνται οι αντικειμενοστραφείς μετρικές λογισμικού και οι δομικές ιδιότητες. Με βάση αυτό το περιβάλλον που αναπτύχθηκε, εκθέτονται τα πειραματικά αποτελέσματα που επιτυγχάνονται σε ανοιχτό κώδικα και βιομηχανικά λογισμικά. Στο δεύτερο, παρουσιάζονται πειραματικά αποτελέσματα που προέκυψαν από 8 βιομηχανικά λογισμικά και 200.000 γραμμές κώδικα ανοιχτού κώδικα σε C++. Στο (De Lucia et al., 2007) παρουσιάζεται μια προσέγγιση δύο φάσεων για την ανάκτηση των κατασκευαστικών προτύπων σχεδίασης. Στην πρώτη φάση το διάγραμμα κλάσεων που εξάγεται από τον πηγαίο κώδικα αναλύεται για τον προσδιορισμό των σχεδιαστικών δομών που είναι υποψήφιες περιπτώσεις προτύπων. Αυτό ολοκληρώνεται χρησιμοποιώντας μια τεχνική ανάκτησης βασισμένη στην οπτική γλωσσική ανάλυση. Η δεύτερη φάση είναι η επέκταση μιας προηγούμενης προσέγγισης, που στοχεύει στη βελτίωση των αποτελεσμάτων της απόδοσης, της ακρίβειας και του χρόνου. Η διαδικασία ανάκτησης έχει αξιολογηθεί σε τέσσερα συστήματα λογισμικού διαφορετικού μεγέθους. Τα επιτευχθέντα αποτελέσματα δείχνουν ότι η προσέγγιση χαρακτηρίζεται από αξιόλογες τιμές αξιοπιστίας.

Μια μεγάλη μερίδα των άρθρων που μελετούνται σε αυτή την κατηγορία, ασχολούνται γενικότερα με την παρουσίαση νέων μεθόδων για την ανίχνευση προτύπων σχεδίασης. Το (Tsantalis et al., 2006) προτείνει μια μεθοδολογία, που

βασίζεται στην καταγραφή των ομοιοτήτων μεταξύ των κορυφών ορισμένων γραφικών παραστάσεων. Η αξιολόγηση σε τρία προγράμματα ανοιχτού λογισμικού κάνει εμφανή την ακρίβεια και την αποδοτικότητα της προτεινόμενης μεθόδου. Το (Costagliola et al., 2005) προτείνει μια αντικειμενοστραφή προσέγγιση ανάκτησης προτύπων σχεδίασης που χρησιμοποιεί μια βιβλιοθήκη προτύπων σχεδίασης, εκφράζεται από την σκοπιά οπτικών γραμματικών, και βασίζεται σε μια τεχνική οπτικής γλωσσικής ανάλυσης. Παρουσιάζεται επίσης ένα οπτικό περιβάλλον που υποστηρίζει τη διαδικασία, ανακτώντας αυτόματα τα πρότυπα σχεδίασης από τα εισαγόμενα διαγράμματα κλάσεων σε UML. Παρόμοια προσέγγιση ανάκτησης προτύπων παρουσιάζεται και στο (Costagliola et al., 2006) που επεκτείνει μια προηγούμενη πρόταση, συμπεριλαμβάνοντας τα αρνητικά κριτήρια στην γραμματική προδιαγραφής των προτύπων σχεδίασης. Και αυτή η προσέγγιση βασίζεται στη χρήση οπτικών τεχνικών γλωσσικής ανάλυσης, και υποστηρίζεται από ένα οπτικό περιβάλλον. Η τεχνική έχει εφαρμοστεί σε προγράμματα και βιβλιοθήκες με ενθαρρυντικά αποτελέσματα. Το (Zhu et al., 2009) παρουσιάζει ένα εργαλείο, αποκαλούμενο LAMBDES-DP, που χρησιμοποιεί τα διαγράμματα UML ως μοντέλα σχεδίασης λογισμικού. Το θεωρητικό υπόβαθρο του εργαλείου είναι μια περιγραφική μορφή της UML και τα πρότυπα σχεδίασης, τα οποία προδιαγράφονται επίσημα στην ίδια γλώσσα. Το εργαλείο χρησιμοποιεί το σύστημα LAMBDES για να μεταφράσει τα διαγράμματα UML και εμπλέκει το θεώρημα SPASS για να αποφασίσει εάν το σχέδιο προσαρμόζεται σε κάποιο πρότυπο. Μερικά πειράματα δείχνουν ότι το εργαλείο έχει σημαντικά χαμηλότερα ποσοστά λαθών έναντι των υπαρχόντων εργαλείων. Το (Amiot et al., 2001) παρουσιάζει ένα σύνολο εργαλείων και τεχνικών για να βοηθήσει στη σχεδίαση, κατανόηση, και επανασχεδίαση ενός τμήματος λογισμικού, χρησιμοποιώντας πρότυπα σχεδίασης. Δυο πρωτότυπα εργαλεία ορίζονται, και σε συνδυασμό με την υποστήριξη της συντήρησης τονίζουν τις ατέλειες ενός σχεδίου και προτείνουν την εφαρμογή διορθώσεων που βασίζονται στις ευρέως αποδεκτές λύσεις προτύπων σχεδίασης. Στο (Niere et al., 2002) έχει σχεδιαστεί μια προσέγγιση ειδικά για να υπερνικήσει τα προβλήματα εκλεξιμότητας που προκαλούνται από τις πολλές παραλλαγές των προτύπων σχεδίασης σε επίπεδο σχεδιασμού και εφαρμογής. Είναι βασισμένο σε έναν νέο αλγόριθμο αναγνώρισης που λειτουργεί επαυξητικά αντί να προσπαθεί να αναλύσει ένα μεγάλο σύστημα λογισμικού σε ένα μόνο πέρασμα χωρίς οποιαδήποτε ανθρώπινη επέμβαση. Στο άρθρο

παρουσιάζεται επίσης μια συγκριτική και ποσοτική αξιολόγηση της εφαρμογής της προσέγγισης στις βιβλιοθήκες AWT και JGL της Java, δείχνοντας ότι τόσο η προσέγγιση όσο και το αντίστοιχο εργαλείο επιτρέπουν την ανάλυση μεγάλων συστημάτων λογισμικού παράγοντας λογικά αποτελέσματα. Το άρθρο (Wang και Tzerpos, 2005) παρουσιάζει μια προσέγγιση για την ανίχνευση προτύπων σχεδίασης σε συστήματα Eiffel που ασχολούνται και με τη στατική δομή και με τη δυναμική συμπεριφορά των συνθετικών. Επίσης παρουσιάζεται το πρώτο εργαλείο αντίστροφης μηχανικής για την ανίχνευση προτύπων στα συστήματα αυτά, που ονομάζεται DPVK. Τα αποτελέσματα τριών διαφορετικών μελετών περίπτωσης δείχνουν ότι DPVK είναι αρκετά αποτελεσματικό. Στο (Dong και Zhao, 2007) παρουσιάζονται τέσσερα πειράματα για την ανακάλυψη προτύπων σχεδίασης σε συστήματα ανοιχτού λογισμικού χρησιμοποιώντας το εργαλείο, DP-Miner. Αυτά τα πειράματα ανακαλύπτουν τα πρότυπα Adapter, Bridge, Strategy, και Composite στα συστήματα Java.AWT, JUnit, JEdit, και JHotDraw. Τα πειραματικά αποτελέσματα δείχνουν ότι τα πρότυπα σχεδίασης έχουν εφαρμοστεί ευρέως σε αυτά τα συστήματα και μπορούν να ανακτηθούν. Επιπλέον, τα αποτελέσματα συγκρίνονται με εκείνα άλλων πειραμάτων και έτσι διαπιστώνονται διάφορες αποκλίσεις. Τέλος στο (Di Penta et al., 2007) παρουσιάζεται μια πρωτότυπη βασιζόμενη στον έλεγχο προσέγγιση για την ανίχνευση αρχιτεκτονικών πρότυπων συστημάτων που είναι προσανατολισμένα στις υπηρεσίες. Κάθε πρότυπο περιγράφεται από μία παραμετρική λογική, ενώ η επαλήθευση γίνεται σε ένα πρότυπο του συστήματος, το οποίο μπορεί να ανακτηθεί είτε μέσω κώδικα είτε μέσω ενός συνόλου ημερολογίων του συστήματος.

Τα (Kramer και Prechelt, 1996 , Balanyi και Ferenc, 2003 , Asencio et al., 2002 και Shi και Olsson, 2006) αναφέρονται σε μεθοδολογίες ανάκτησης απευθείας από πηγαίο κώδικα. Το (Kramer και Prechelt, 1996) παρουσιάζει μια προσέγγιση, αποκαλούμενη σύστημα Pat, το οποίο εξάγει σχεδιαστικές πληροφορίες απευθείας από C++ αρχεία επικεφαλίδων και τις αποθηκεύει. Τα πρότυπα εκφράζονται με κανόνες Prolog και οι σχεδιαστικές πληροφορίες μεταφράζονται σε γεγονότα. Έπειτα χρησιμοποιείται ένα απλό ερώτημα Prolog για την αναζήτηση όλων των προτύπων. Τέσσερις εφαρμογές έχουν εξεταστεί με το Pat. Με μερικούς περιορισμούς έχουν εντοπιστεί όλες οι περιπτώσεις προτύπων. Τα αποτελέσματα δείχνουν ότι το Pat είναι ένα χρήσιμο εργαλείο για την ανάκτηση

σχεδιαστικών πληροφοριών. Το (Balanyi και Ferenc, 2003) παρουσιάζει μια νέα μέθοδο για τα την ανακάλυψη προτύπων σχεδίασης μέσα σε πηγαίο κώδικα C++, που περιλαμβάνει την ανίχνευση μεταβίβασης κλήσεων μεθόδων, της δημιουργίας αντικειμένων και της υπερφόρτωσης συναρτήσεων και παρέχει μια ακριβή προδιαγραφή για το πώς λειτουργούν τα πρότυπα. Επίσης παρουσιάζεται μια νέα γλώσσα σήμανσης σχεδιαστικών προτύπων (DPML), που βασίζεται στην XML και παρέχει έναν εύκολο τρόπο στους χρήστες για να τροποποιήσουν τις περιγραφές των προτύπων ώστε να ανταποκρίνονται στις ανάγκες τους, ή ακόμα για να ορίσουν τα δικά τους πρότυπα ή να εντοπίσουν απλώς κλάσεις που συμμετέχουν σε ορισμένες σχέσεις. Η προσέγγιση έχει εξεταστεί σε τέσσερα συστήματα ανοιχτού λογισμικού, και έχει βρεθεί για να είναι αποτελεσματική στην ανακάλυψη στιγμιοτύπων προτύπων σχεδίασης. Στο (Asencio et al., 2002) αναπτύσσονται κάποια εργαλεία, που εξετάζουν στατικά τον πηγαίο κώδικα, αναγνωρίζουν αυτόματα τη χρήση προτύπων σχεδίασης και συσχετίζουν τη χρήση προτύπων με την ποιότητα λογισμικού, κωδικοποιώντας τους στόχους και τις προσδοκίες εφαρμοσμένης μηχανικής συστημάτων για τον πηγαίο κώδικα, και δίνοντας έτσι πρόσβαση για να αποκαλυφθούν σημαντικές σχεδιαστικές αποφάσεις. Το (Shi και Olsson, 2006) παρουσιάζει μια νέα, πλήρως αυτοματοποιημένη προσέγγιση ανίχνευσης προτύπων, που βασίζεται σε μια νέα επαναταξινόμηση των GoF προτύπων σύμφωνα με τις προθέσεις του καθενός, η οποία λέγεται ότι ταιριάζει καλύτερα στην αντίστροφη μηχανική. Ορίζεται επίσης το PINOT, ένα πρωτότυπο εργαλείο που χρησιμοποιείται για την εφαρμογή της προσέγγισης. Το PINOT ανιχνεύει όλα τα πρότυπα GoF που έχουν σαφείς ορισμούς καθοδηγούμενους από τη δομή του κώδικα ή τη συμπεριφορά του συστήματος και είναι ένα πλήρως αυτοματοποιημένο εργαλείο ανίχνευσης προτύπων που είναι γρηγορότερο, ακριβέστερο, και περιεκτικότερο από τα υπάρχοντα εργαλεία.

Υπάρχουν επίσης άρθρα που αναφέρονται στην ανάκτηση προτύπων σχεδίασης μετά τη φάση ολοκλήρωσης της ανάπτυξης λογισμικού. Το (Kaczor et al.,2006) παρουσιάζει μια προσαρμογή των bitvector αλγορίθμων της βιοπληροφορικής στο πρόβλημα συντήρησης λογισμικού λόγω της ανάγκης για αναγνώριση των προτύπων. Ο έμφυτος παραλληλισμός των bit-wise διαδικασιών χρησιμοποιείται για την παραγωγή ενός αποδοτικού bit-vector αλγόριθμου που βρίσκει ακριβή ή και κατά προσέγγιση στιγμιότυπα προτύπων σχεδίασης σε ένα πρόγραμμα. Η προσέγγιση εφαρμόστηκε σε τρία μικρομεσαία προγράμματα για

να παρουσιαστεί την αποδοτικότητά της. Τα αποτελέσματα συγκρίθηκαν με δύο υπάρχουσες βασισμένες στους περιορισμούς προσεγγίσεις. Μια παρόμοια προσέγγιση παρουσιάζεται στο (Kaczor et al., 2010). Το άρθρο παρουσιάζει την προσαρμογή δύο κλασσικών αλγορίθμων ταιριάσματος προτύπων στο πρόβλημα της συντήρησης λογισμικού ως προς την αναγνώριση σχεδιαστικών μοτίβων: αυτόματη προσομοίωση και bit-vector επεξεργασία. Προκειμένου να προσδιοριστεί ακριβώς και κατά προσέγγιση η ύπαρξη μοτίβων παρουσιάζονται δύο μελέτες περίπτωσης, που έχουν πραγματοποιηθεί για να μετρήσουν την απόδοση, την ακρίβεια, και ανάκληση των αλγορίθμων. Οι μελέτες δείχνουν ότι το κατά προσέγγιση ταίριασμα φράσεων που είναι βασισμένο στην bit-vector επεξεργασία παρέχει αποδοτικούς αλγορίθμους για την αναγνώριση μοτίβων σχεδίασης. Το (Chaabane, 2007) από την άλλη πλευρά στοχεύει στην ανίχνευση της πιθανής κακής χρήσης προτύπων σε πηγαίο κώδικα και στο να μετρηθούν οι αποδόσεις τους, προκειμένου να γίνουν ξεκάθαρα τα πλαίσια που πρέπει να αναδομηθούν. Η προσέγγιση επεξηγεί τον λόγο για τον οποίο πρέπει να εγκατασταθεί μια σύνδεση μεταξύ της στατικής ανάλυσης για την ανίχνευση προτύπων στον πηγαίο κώδικα και της δυναμικής ανάλυσης για τη μέτρηση της απόδοσης στις αντίστοιχες εκτελεσμένες οδηγίες, επιτρέποντας έτσι την αναγνώριση των προτύπων που πρέπει πραγματικά να αναδομηθούν. Το (Keller et al., 1999) παρουσιάζει ένα περιβάλλον για την επανασχεδίαση των σχεδιαστικών συστατικών που βασίζονται στις δομικές περιγραφές των προτύπων σχεδίασης. Σκοπός είναι η εισαγωγή της βασισμένης σε πρότυπα αντίστροφης μηχανικής ως μια πολύτιμη τεχνική για την κατανόηση λογισμικού ώστε να αλλάξει η κοινή γνώμη σύμφωνα με την οποία τα πρότυπα σχεδίασης είναι σημαντικά μόνο στην ευθεία μηχανική. Στο άρθρο, γίνεται μια επισκόπηση του περιβάλλοντος (SPOOL) ενώ επεξηγούνται επίσης τρεις μελέτες περίπτωσης, και συζητιέται ο τρόπος που η βασιζόμενη σε πρότυπα αντίστροφη μηχανική βοήθησε στην απόκτηση επίγνωσης της σχεδιαστικής λογικής κάποιων τμημάτων τριών μεγάλης κλίμακας C++ συστημάτων λογισμικού.

Το άρθρο (Gueheneuc και Antoniol, 2008) αποτελεί έναν συνδυασμό των προηγούμενων δυο κατηγοριών. Παρουσιάζει το DeMIMA, μια προσέγγιση που προσδιορίσει ημιαυτόματα τις μικροαρχιτεκτονικές που μοιάζουν με σχεδιαστικά μοτίβα στον πηγαίο κώδικα και εξασφαλίσει την ανιχνευσιμότητα τους μεταξύ της εφαρμογής και του σχεδιασμού. Το DeMIMA έχει εφαρμοστεί σε πέντε συστήματα ανοιχτού λογισμικού και, κατά μέσον όρο, παρατηρήθηκε 34% ακρίβεια για τα 12

σχεδιαστικά μοτίβα που εξετάστηκαν. Μέσω της χρήσης του βασισμένου σε επεξηγήσεις περιορισμών προγραμματισμού, εξασφαλίζει 100% επιτυχία στα πέντε συστήματα. Το DeMIMA έχει εφαρμοστεί επίσης σε 33 βιομηχανικά συστατικά.

Άρθρα που στοχεύουν στην βελτίωση της ακρίβειας κατά την ανάκτηση σχεδιαστικών προτύπων είναι τα (Huang et al., 2005 , Ferenc et al., 2005 και Pettersson, 2005). Η προσέγγιση που παρουσιάζεται στο (Huang et al., 2005) μετατρέπει τις επίσημες προδιαγραφές σε αναπαραστάσεις σε Prolog για να υποστηρίξουν την ανάκτηση προτύπων. Για την απεικόνιση του τρόπου καταγραφής των προδιαγραφών και της ανάκτησης προτύπων, παρουσιάζεται ένα παράδειγμα για κάθε κατηγορία προτύπων σχεδίασης. Για την επικύρωση της προσέγγισης, αναπτύχθηκε ένα εργαλείο που ονομάζεται PRAssistor ενώ αναλύθηκαν και δύο γνωστά πλαίσια ανοιχτού λογισμικού. Τα αποτελέσματα του πειράματος δείχνουν ότι τα περισσότερα από τα πρότυπα που εξετάζονται έχουν ανακτηθεί και θεωρείται ότι η προσέγγιση και το εργαλείο που παρουσιάζονται είναι χρήσιμα για την ανάκτηση σχεδιαστικών πληροφοριών σε πραγματικά συστήματα Στο (Ferenc et al., 2005) βασική ιδέα είναι να χρησιμοποιηθεί η μέθοδος μηχανικής μάθησης για να ενισχύει την εξόρυξη προτύπων φιλτράροντας όσο το δυνατόν περισσότερα λανθασμένα αποτελέσματα. Για να επιτευχθεί αυτό, τόσο τα αληθινά όσο και τα λανθασμένα πρότυπα διακρίνονται με τη βοήθεια μιας βάσης δεδομένων εκμάθησης που έχει δημιουργηθεί χειροκίνητα από ένα μεγάλο C++ σύστημα. Τα πειράματα που έγιναν έδειξαν ότι η προσέγγιση έχει τη δυνατότητα να ενισχύσει τις τρέχουσες μεθόδους εξόρυξης προτύπων σχεδίασης. Σε αντίθεση με τα δύο προηγούμενα άρθρα, το (Pettersson, 2005) ερευνά το πού χρειάζεται προσπάθεια για να επιτευχθεί υψηλή ακρίβεια κατά την ανίχνευση προτύπων στον κώδικα. Εξετάζονται βασικά οι σημαντικότερες παράμετροι για τη βελτίωση της ακρίβειας κατά τη διαδικασία της ανίχνευσης και τα αποτελέσματα δείχνουν ότι τα απλά πρωτόκολλα συμπεριφοράς διπλασιάζουν την ακρίβεια όταν λαμβάνεται κάλυψη 30% και ότι αποδίδουν με μεγάλη ακρίβεια όταν λαμβάνεται γενικά υψηλή κάλυψη. Στις περιπτώσεις αυτές υπάρχει ενδιαφέρον όταν η υψηλή κάλυψη δεν μπορεί να επιτευχθεί.

Τέλος, δυο άρθρα ασχολούνται με την σύγκριση των επιδόσεων των τεχνικών και των εργαλείων εξόρυξης προτύπων με σκοπό την αξιολόγηση τους. Το (Streitferdt et al., 2005) αναλύει τις υπάρχουσες προσεγγίσεις αναζήτησης

προτύπων και τις συγκρίνει με βάση τις τιμές ανάκλησης και ακρίβειας, μετρικές από τον τομέα ανάκτησης πληροφοριών αναπτύσσοντας νέους αλγορίθμους αναζήτησης προτύπων για τα 23 πρότυπα σχεδίασης που περιγράφονται από τον Gamma et al. Το άρθρο εξηγεί τα βασικά σημεία αυτής της αναζήτησης προτύπων και περιγράφει τα πρώτα αποτελέσματα των αλγορίθμων αναζήτησης που αναπτύχθηκαν ως Java plug-in για το Together IDE. Το (Fulop et al., 2008) παρουσιάζει την ανάπτυξη μιας συγκριτικής μέτρησης επιδόσεων για την αξιολόγηση των εργαλείων εξόρυξης προτύπων σχεδίασης και στιγμιοτύπων των προτύπων. Η προσέγγιση είναι ανεξάρτητη από τη γλώσσα προγραμματισμού, τα εργαλεία, τα πρότυπα και το λογισμικό που χρησιμοποιούνται, και είναι ελεύθερα διαθέσιμη στην κοινότητα. Η benchmark βάση δεδομένων περιέχει τα αποτελέσματα τριών εργαλείων. Μερικά ανακτημένα πρότυπα έχουν ήδη ελεγχθεί από πεπειραμένους προγραμματιστές. Αυτή η εργασία είναι το πρώτο βήμα στην δόμηση μιας μεγάλης βάσης δεδομένων προτύπων σχεδίασης που χρησιμοποιούνται στο ανοιχτό λογισμικό.

2.3.1.3 Πρότυπα σχεδίασης και Ποιότητα Λογισμικού

Η κατηγορία αυτή περιλαμβάνει άρθρα που ασχολούνται με τις συνέπειες της εφαρμογής προτύπων σχεδίασης στην ποιότητα του λογισμικού. Χαρακτηριστικά που αξιολογούν την ποιότητα του λογισμικού είναι η ευκολία κατά τη συντήρηση και την εφαρμογή αλλαγών, η δυνατότητα επαναχρησιμοποίησης και διενέργειας ελέγχων, η προσαρμοστικότητα, η επεκτασιμότητα, η σταθερότητα, η ευελιξία, η εύκολη κατανόηση, η χρηστικότητα κ.α. Με βάση αυτά τα ποιοτικά χαρακτηριστικά παρουσιάζονται τα 30 άρθρα αυτής της κατηγορίας.

Τα (Di Penta et al., 2008, Aversano et al., 2007, Vokac, 2004 και Gatrell et al., 2009) παρουσιάζουν εμπειρικές μελέτες εξετάζοντας το κατά πόσο η εφαρμογή προτύπων σχεδίασης διευκολύνει την εφαρμογή αλλαγών σε ένα λογισμικό. Το (Di Penta et al., 2008) στοχεύει στην κατανόηση σχετικά με το αν υπάρχουν ρόλοι που είναι πιο επιρρεπείς στις αλλαγές και αν υπάρχουν κάποιες αλλαγές που συμβαίνουν συχνότερα σε συγκεκριμένους ρόλους. Η προσέγγιση αφορά ένα σχεδιαστικό μοτίβο αναγνώρισης προτύπων ενώ περιγράφεται και η εξαγωγή των στοιχείων που απαιτούνταν για τη διεξαγωγή της μελέτης. Τα στοιχεία προέρχονται από αποθετήρια πηγαίου κώδικα τριών συστημάτων καθώς

και από 12 πρότυπα σχεδίασης. Τα αποτελέσματα επιβεβαιώνουν ότι κάποιοι ρόλοι στα σχεδιαστικά μοτίβα είναι πιο επιρρεπείς στις αλλαγές και προειδοποιούν για τη σημαντικότητα του κατάλληλου σχεδιασμού των τμημάτων ενός μοτίβου όταν συμβαίνουν συχνές αλλαγές. Τα αποτελέσματα μιας παρόμοιας μελέτης παρουσιάζει το (Aversano et al., 2007). Η μελέτη αφορά την εξέλιξη των προτύπων σχεδίασης σε τρία συστήματα ανοιχτού ,λογισμικού γραμμένα σε Java και αναλύει τη συχνότητα που τα πρότυπα τροποποιούνται, σε ποιες αλλαγές υποβάλλονται και ποιες κλάσεις αντικαθιστούνται με τα πρότυπα. Αξίζει να σημειωθεί η ταύτιση των αποτελεσμάτων των δυο άρθρων. Στο (Vokac, 2004) αναλύεται η εβδομαδιαία εξέλιξη και συντήρηση ενός μεγάλου εμπορικού προϊόντος σε διάρκεια τριών ετών, συγκρίνοντας τα επίπεδα λαθών για τις κλάσεις που συμμετείχαν σε επιλεγμένα πρότυπα σχεδίασης με τον κώδικα γενικά. Τα ποσοστά των προτύπων προκύπτει ότι έχουν σημαντικές διαφορές. Επίσης αναπτύσσεται ένα σύνολο νέων εργαλείων ικανό να εξάγει πληροφορίες για τα πρότυπα σχεδίασης. Με βάση μια ποιοτική ανάλυση του κώδικα και της φύσης των προτύπων, συμπεραίνεται ότι τα πρότυπα Observer και Singleton συσχετίζονται με μεγαλύτερες δομές κώδικα, και έτσι μπορούν να χρησιμοποιηθούν ως δείκτες στον κώδικα, γεγονός που απαιτεί ιδιαίτερη προσοχή. Αντίθετα το πρότυπο Factory είναι πιο συμπαγές και πιθανότατα εμφανίζει πιο χαλαρή σύζευξη, ενώ το Template Method φάνηκε να χρησιμοποιούνταν τόσο σε απλές όσο και σε πιο σύνθετες καταστάσεις. Το (Gatrell et al., 2009) καταγράφει μια μελέτη για την τροποποίηση προτύπων που εξήχθησαν από ένα μεγάλο εμπορικό σύστημα C# και προσπαθεί να αποφασίσει εάν υπάρχει σχέση μεταξύ της πιθανότητας τροποποίησης μιας κλάσης και του σχεδιαστικού περιεχομένου. Τα συγκεκριμένα πρότυπα σχεδίασης και η ροπή τους για αλλαγές προσδιορίζονται, δείχνοντας ότι τα πρότυπα σχεδίασης που συμμετέχουν έχουν μεγαλύτερη τάση προς αλλαγή από τις κλάσεις που δεν συμμετείχαν σε κάποιο πρότυπο σχεδίασης.

Η επόμενη υποκατηγορία άρθρων που μελετήθηκαν ασχολείται με την χρήση προτύπων ως προς την ευκολία συντήρησης λογισμικού. Το (Vokáč et al., 2004) επεκτείνει ένα προηγούμενο πείραμα σχετικά με την συντήρηση προτύπων σχεδίασης. Από την έρευνα αποδεικνύεται ότι κάθε πρότυπο από αυτά που εξετάστηκαν έχει τα δικά του χαρακτηριστικά και δεν είναι δυνατόν να τα κρίνουμε γενικά ως χρήσιμα ή επιβλαβή, αλλά πρέπει να χρησιμοποιούνται ανάλογα με το

πρόβλημα που υπάρχει και την εμπειρία του προγραμματιστή. Η τεκμηρίωση των πρότυπων σχεδίασης που χρησιμοποιούνται σε ένα πρόγραμμα μπορεί να βελτιώσει τόσο την ταχύτητα όσο και την ποιότητα των διαδικασιών συντήρησης του. Και τα επόμενα δυο άρθρα παρουσιάζουν πειράματα. Το (Prechelt et al., 2001) ερευνά σενάρια συντήρησης λογισμικού που χρησιμοποιούν διάφορα πρότυπα σχεδίασης και συγκρίνει σχεδιαστικές λύσεις με πρότυπα και απλούστερες εναλλακτικές λύσεις. Ως συμπέρασμα προκύπτει ότι συνήθως η χρήση πρότυπων είναι χρησιμότερη απ' ότι οι εναλλακτικές λύσεις. Το (Ng et al., 2006) ασχολείται με την συντήρηση του JHotDraw, ενός συστήματος ανοιχτού λογισμικού που έχει επεκταθεί με πολλαπλά πρότυπα.. Τα αποτελέσματα δείχνουν ότι, για να ολοκληρωθεί ένα καθήκον συντήρησης, ο χρόνος που ξοδεύτηκε ακόμη και από άπειρους συντηρητές ήταν πολύ πιο σύντομος στην επανασχεδιασμένη εκδοχή από αυτόν των πεπειραμένων συντηρητών στην αρχική έκδοση. Επιπλέον, η ποιότητα των παραδοτέων προγραμμάτων, από άποψη ακρίβειας, φαίνεται να είναι συγκρίσιμη. Το (Wendorff, 2001) από την άλλη πλευρά παρουσιάζει ένα μεγάλο εμπορικό πρόγραμμα όπου η ανεξέλεγκτη χρήση των προτύπων έχει συμβάλει σε σοβαρά προβλήματα συντήρησης. Στο άρθρο παρουσιάζονται χαρακτηριστικά προβλήματα που μπορούν να παρουσιαστούν στον πηγαίο κώδικα από την απερίσκεπτη χρήση των προτύπων και έπειτα παρουσιάζονται μια σειρά απλών βημάτων για την αξιολόγηση των προτύπων από μια προοπτική επανασχεδίασης. Τα αποτελέσματα δείχνουν ότι αν και η απομάκρυνση αυτών των προτύπων εμφανίζεται να είναι επιθυμητή, συχνά συνδέονται στενά με άλλα αντικείμενα του λογισμικού, με αποτέλεσμα η αφαίρεσή τους να μην είναι δυνατή.

Η συμβολή των προτύπων σχεδίασης στη δυνατότητα επέκτασης ενός λογισμικού παρουσιάζεται στα (Ampatzoglou και Chatzigeorgiou, 2007, Kouskouras et al., 2008, και Ng et al., 2006).Το (Ampatzoglou και Chatzigeorgiou, 2007) εξετάζει τον τρόπο που τα αντικειμενοστραφή πρότυπα σχεδίασης μπορούν να χρησιμοποιηθούν στα παιχνίδια για υπολογιστή και αξιολογεί τα οφέλη και τα μειονεκτήματά τους. Για το λόγο αυτό, αξιολογούνται δυο παιχνίδια ανοιχτού λογισμικού. Για την ποσοτική αξιολόγηση, τα προγράμματα αναλύονται με τεχνικές αντίστροφης μηχανικής και υπολογίζονται κάποιες μετρικές λογισμικού. Τα αποτελέσματα δείχνουν ότι τα πρότυπα μπορούν να είναι ευεργετικά όσον αφορά τη ευκολία συντήρησης, αφού φαίνεται να μειώνουν τη σύζευξη και την

πολυπλοκότητα και να αυξάνουν τη συνοχή του λογισμικού. Ως συμπέρασμα προκύπτει ότι στον προγραμματισμό παιχνιδιών πρέπει να ενθαρρύνεται η κατάλληλη εφαρμογή προτύπων σχεδίασης. Στο (Kouskouras et al., 2008) ερευνάται η συμπεριφορά μιας αντικειμενοστραφούς εφαρμογής λογισμικού σε ένα συγκεκριμένο σενάριο επέκτασης. Αρχικά αναλύεται το πλαίσιο αξιολόγησης και παρουσιάζονται τρείς εναλλακτικές λύσεις. Η πρώτη σχολιάζει τα αποτελέσματα των μετρικών και άλλες ποιοτικές παρατηρήσεις για μια απλή λύση του συστήματος. Η δεύτερη παρουσιάζει και εφαρμόζει το πρότυπο Registry και η τρίτη εφαρμόζει θεματοστρεφή προγραμματισμό (Aspect-Oriented Programming). Ως αποτέλεσμα, επιδιώκεται η αξιολόγηση των τριών εναλλακτικών λύσεων, σε ποιοτικό και σε ποσοτικό επίπεδο, προσδιορίζοντας τις πρόσθετες σχεδιαστικές επιπτώσεις που απαιτούνται τόσο για την επέκταση όσο και για την αξιολόγηση της επίδρασης της επέκτασης σε διάφορες ποιοτικά χαρακτηριστικά της εφαρμογής. Το (Ng et al. 2006) εξετάζει εάν υπάρχει οποιαδήποτε σχέση μεταξύ των προτύπων και της Αρχής της Ανοιχτής - Κλειστής σχεδίασης, η οποία υποστηρίζεται ότι μπορεί να καταφέρει να δημιουργήσει επεκτάσιμο αντικειμενοστραφές λογισμικό. Ένα πείραμα έχει πραγματοποιηθεί προτείνοντας ότι η ικανοποίηση των προτύπων γενικά οδηγεί στην προσαρμογή στην Αρχή Ανοιχτής – Κλειστής σχεδίασης. Εντούτοις, βρέθηκαν τρεις περιπτώσεις εξαιρέσεων.

Το (Baudry et al., 2001) παρουσιάζει δύο διαμορφώσεις ενός αντικειμενοστραφούς σχεδίου που μπορεί να αποδυναμώσει τη δυνατότητα δοκιμών. Συζητά τα προβλήματα που μπορούν να προκύψουν κατά τη διάρκεια ελέγχου είτε από κακό σχεδιασμό είτε από εφαρμογή μιας αντικειμενοστραφούς αρχιτεκτονικής. Επιπλέον απεικονίζει την έννοια της σύγκρουσης των δοκιμών χρησιμοποιώντας πρότυπα σχεδίασης και προτείνει μια λύση αποφυγής των συγκρούσεων. Το άρθρο εστιάζει στα πρότυπα σχεδίασης ως συνεπή υποσύνολα για την αρχιτεκτονική, εξηγεί πώς η χρήση τους μπορεί να παρέχει έναν τρόπο για την μείωση της πολυπλοκότητας των ελέγχων που αφορούν τις συγκρούσεις και τον περιορισμό της επίδρασής τους στις κλάσεις που περιλαμβάνονται στο πρότυπο. Η δυνατότητα δοκιμών στον αντικειμενοστραφή σχεδιασμό απασχολεί και το άρθρο (Baudry et al., 2003), που ξεκινά με μια περίληψη της μέτρησης της δυνατότητας δοκιμών και των στερεοτύπων που προτείνονται για την βελτίωση της δυνατότητας ελέγχου ενός διαγράμματος κλάσεων και μετά παρουσιάζει τα

πρότυπα σχεδίασης, και τον βασισμένο σε πρότυπα σχεδιασμό λογισμικού και απεικονίζει την ανάλυση της δυνατότητας δοκιμών με χρήση δύο παραδειγμάτων. Το άρθρο καταλήγει μελετώντας αφηρημένες αναπαραστάσεις προτύπων σχεδίασης μέσα σε UML metamodel, για να επιτευχθεί αυτόματη εφαρμογή των προτύπων σχεδίασης με έναν τρόπο που θα μπορεί να ελεγχθεί.

Το ποιοτικό χαρακτηριστικό της προσαρμοστικότητας του λογισμικού, ως προς τον αντίκτυπο των προτύπων σχεδίασης κατά την μεταφορά του σχεδίου σε άλλη γλώσσα προγραμματισμού μελετά το άρθρο (Nielson και Knutson, 2006). Το άρθρο προτείνει ένα κύκλο συντήρησης προτύπων σχεδίασης ώστε να ανταποκρίνονται στις εξελικτικές γλωσσικές αλλαγές και να προσαρμόζονται στις OO++ γλώσσες. Οι προτάσεις απεικονίζονται με την επέκταση του προτύπου Iterator σε μια OO++ παραλλαγή. Το άρθρο (Elish, 2006) εκθέτει με παραδείγματα τον αντίκτυπο τεσσάρων structural προτύπων σχεδίασης (adapter, bridge, composite and faqade) στη σταθερότητα των διαγραμμάτων κλάσεων. Τα παραδείγματα δείχνουν έναν θετικό αντίκτυπο των προτύπων που μελετούνται. Την ευκολία κατανόησης λογισμικού που κάνει χρήση προτύπων σχεδίασης παρουσιάζει το (Beck et al., 1996), περιγράφοντας εν συντομία βιομηχανικές εμπειρίες με πρότυπα. Η χρήση των προτύπων φαίνεται να ασκεί μεγάλη επίδραση στον τρόπο ανάπτυξης λογισμικού. Αφ' ενός, τα καλά πρότυπα είναι δύσκολο να χρησιμοποιηθούν. Είναι δύσκολο να βρεθεί μια ισορροπία μεταξύ των πλεονεκτημάτων και των μειονεκτημάτων, ειδικά όταν τα μετρήσιμα αποτελέσματα δεν είναι ακόμα διαθέσιμα. Το (Malloy και Power, 2006) ασχολείται με το ποιοτικό χαρακτηριστικό της ευελιξίας και περιγράφει τη χρήση δύο προτύπων σχεδίασης για την αυτοματοποίηση του ελέγχου των μη μεταβλητών κλάσεων μιας εφαρμογής σε C++. Αρχικά παρουσιάζεται μια επισκόπηση των προτύπων Visitor και Decorator μαζί με μια λεπτομερή περιγραφή της χρήσης τους για επικύρωση. Για την διερεύνηση της χρήσης των δυο προτύπων, παρουσιάζεται η μελέτη μιας υπάρχουσας, καλά-ελεγμένης εφαρμογής, που ονομάζεται keystone, a parser and front-end για τη γλώσσα C++. Παρουσιάζονται ακόμα τα ποσοτικά αποτελέσματα που μετρούν τον αντίκτυπο αυτών των προσεγγίσεων στη συγκεκριμένη μελέτη περίπτωσης. Τα αποτελέσματα δείχνουν ότι τα δύο πρότυπα που μελετούνται παρέχουν ευελιξία από άποψη της συχνότητας και του επιπέδου κατάτμησης για την επικύρωση σταθερών κλάσεων, που εκφράζονται σε OCL (Object Constraint Language). Το (Ellis et al., 2007) παρουσιάζει γιατί η δημιουργία αντικειμένων με

τη χρήση των προτύπων Factory που χρησιμοποιούνται σε APIs είναι πολύ πιο χρονοβόρα από τους κατασκευαστές, ανεξάρτητα από το πλαίσιο ή το επίπεδο εμπειρίας του προγραμματιστή που χρησιμοποιεί το API. Τέλος συζητούνται οι λόγοι που συμβαίνει αυτό, τα εμπόδια που εμφανίζονται και τα πιθανά εναλλακτικά πρότυπα που μπορούν να χρησιμοποιηθούν.

7 από τα άρθρα που μελετήθηκαν ασχολούνται με περισσότερες από μία από τις προαναφερθείσες κατηγορίες ποιοτικών χαρακτηριστικών, καθιστώντας αδύνατη τη κατάταξή τους σε μία μονό από αυτές. Το McNatt και Bieman, 2001) εξετάζει την έννοια της σύζευξης προτύπων («σφιχτής» ή «χαλαρής») για να ταξινομήσει τους τρόπους που τα σχέδια μπορούν να περιλάβουν συνδεμένα πρότυπα. Τα θετικά της σύζευξης προτύπων αξιολογούνται από τη σκοπιά των αποτελεσμάτων της στην ευκολία συντήρησης, την κατάτμηση, και την ικανότητα επαναχρησιμοποίησης σε περιπτώσεις σύζευξης προτύπων με διάφορους τρόπους. Το (Ng et al., 2007) μελετά εμπειρικά εάν οι συντηρητές λογισμικού χρησιμοποιούν τα ήδη εφαρμοσμένα πρότυπα σχεδίασης, και αν ναι, με ποιες λειτουργίες ασχολούνται συχνότερα. Τα πειράματα δείχνουν ότι σχεδόν όλοι ασχολούνται με την προσθήκη των νέων συμμετεχόντων, λιγότεροι ασχολούνται με ενέργειες που αφορούν τους πελάτες, ενώ ακόμη και λιγότεροι εκτελούν καθήκοντα που περιλαμβάνουν αφηρημένα την ομάδα των συμμετεχόντων. Επιπλέον, η χρήση των προτύπων σχεδίασης παρατηρείται στατιστικά να είναι συνδεμένη με την παράδοση των λιγότερο ελαττωματικών κωδίκων. Το (Ng και Cheung, 2005) παραθέτει τους σχεδιαστικούς περιορισμούς στην εφαρμογή των προτύπων για την επίτευξη σωστής ενθυλάκωσης. Τα πρώτα πειράματα δείχνουν ότι αυτή η προσέγγιση διευκολύνει τις τροποποιήσεις του προγράμματος σύμφωνα με πολλές σχεδιαστικές ιδέες. Αν και η προτεινόμενη μεθοδολογία απαιτεί ενδεχομένως μια επιβάρυνση για την εφαρμογή περισσότερων προτύπων, δεν απαιτεί τροποποίηση των επαναχρησιμοποιήσιμων κλάσεων για την επικοινωνία τους, καθώς το λογισμικό επεκτείνεται, διευθετώντας το πρόβλημα αυτό. Το (Prechelt et al., 2002) εστιάζει στη συντήρηση και καταγράφει τις πειραματικές δοκιμές που έγιναν για την ακόλουθη ερώτηση: Βοηθά τον συντηρητή όταν τα πρότυπα σχεδίασης που χρησιμοποιούνται σε ένα κώδικα προγράμματος είναι ρητά τεκμηριωμένα (χρησιμοποιώντας σχόλια στον πηγαίο κώδικα) έναντι ενός καλά-σχολιασμένου προγράμματος χωρίς ρητή αναφορά στα πρότυπα σχεδίασης; Η συμβολή αυτού του άρθρου είναι διπλή: Παρέχει τα πρώτα

ελεγχόμενα πειραματικά αποτελέσματα για τη χρήση προτύπων σχεδίασης και παρουσιάζει μια προσέγγιση, που είναι η λύση σε μια σημαντική κατηγορία σχεδιαστικών προβλημάτων σχετικά με την τεκμηρίωση, που αφορούν πειράματα. Το (Bieman et al., 2003) έχει στόχο να χρησιμοποιηθούν οι μελέτες περίπτωσης, τόσο από τη βιομηχανία όσο και από την κοινότητα ανοιχτού λογισμικού, για να ερευνηθεί η σχέση μεταξύ των σχεδιαστικών δομών στο αντικειμενοστραφές λογισμικό και της ανάπτυξης και των αλλαγών κατά τη συντήρηση. Τα αποτελέσματα παρέχουν διορατικότητα στο πώς τα πρότυπα σχεδίασης χρησιμοποιούνται πραγματικά, και μπορούν να βοηθήσουν στο να μάθουμε να αναπτύσσουμε σχέδια λογισμικού που προσαρμόζονται ευκολότερα. Ασχολείται επίσης με έννοιες όπως η δυνατότητα τροποποίησης, συντήρησης και επαναχρησιμοποίησης του λογισμικού. Το (Jain και Yang, 2003) εξετάζει ένα εξελισσόμενο αντικειμενοστραφές εμπορικό σύστημα και εξετάζει τη σχέση μεταξύ της σχεδιαστικής δομής και των αλλαγών λογισμικού. Η πιο σημαντική διαπίστωση είναι η ύπαρξη μιας ισχυρής σχέσης μεταξύ του μεγέθους των κλάσεων και του αριθμού των αλλαγών που συμβαίνουν. Δυο ακόμα μη-αναμενόμενες σχέσεις που προέκυψαν είναι ότι: (1) οι κλάσεις που συμμετέχουν στα πρότυπα σχεδίασης δεν είναι λιγότερο επιρρεπείς σε αλλαγές και (2) οι κλάσεις που επαναχρησιμοποιούνται, συνήθως μέσω κληρονομικότητας, τείνουν να είναι πιο επιρρεπής στις αλλαγές. Τέλος η εργασία (Khomh και Gueheneuce, 2008) μελετά τον αντίκτυπο των προτύπων σχεδίασης στα ποιοτικά χαρακτηριστικά, στα πλαίσια της συντήρησης λογισμικού και της εξέλιξης. Μελετώντας 10 ποιοτικά χαρακτηριστικά αποδεικνύεται ότι τα πρότυπα σχεδίασης δε βελτιώνουν πάντα την ποιότητα των συστημάτων και ότι πρέπει να χρησιμοποιούνται με σύνεση κατά τη διάρκεια της ανάπτυξης επειδή μπορούν να αποτελέσουν εμπόδιο στη συντήρηση και την εξέλιξη. Επίσης αποκαλύπτεται ότι οι αντικειμενοστραφείς αρχές δεν έχουν ως αποτέλεσμα απαραιτήτως συστήματα με καλή ποιότητα.

Τέλος, τα άρθρα που ακολουθούν παρουσιάζουν μελέτες και προσεγγίσεις που αφορούν γενικότερα την ποιότητα λογισμικού. Το (Hsueh et al., 2008) προτείνει μια προσέγγιση για την ανάλυση και τον έλεγχο προτύπων σχεδίασης από άποψη ποιότητας. Για τον έλεγχο της συνοχής μεταξύ του σκοπού και της δομής ενός προτύπου σχεδίασης, χρησιμοποιείται ένα αντικειμενοστραφές μοντέλο ποιότητας, που βοηθά στον έλεγχο της ποιότητας του προτύπου, γεγονός σημαντικό αφού τα πρότυπα χρησιμοποιούνται ευρέως στην αντικειμενοστραφή

ανάπτυξη. Επίσης, προτείνεται ένας τρόπος μέτρησης της αποτελεσματικής βελτίωσης της ποιότητας ενός προτύπου σχεδίασης. Τέλος παρουσιάζεται μια μελέτη περίπτωσης στην οποία αναπτύσσεται ένα εργαλείο λογισμικού για την υποστήριξη της προσέγγισης. Το (Huston, 2001) παρουσιάζει μεθόδους ανάλυσης που καταδεικνύουν τις επιδράσεις της εφαρμογής διάφορων προτύπων μέσω ορισμένων αποτελεσμάτων μετρικών. Στόχος είναι να παρουσιαστεί μια προσέγγιση για την δημιουργία ενός αμεσότερου συστήματος αξιολόγησης της απόδοσης των μετρικών καθώς και προτάσεων για την επιλογή μετρικών και την εφαρμογή κανόνων. Η μέθοδος που υιοθετείται είναι βασισμένη σε μια ανάλυση της αλληλεπίδρασης των σχεδιαστικών μετρικών με τις καθιερωμένες δομές προτύπων σχεδίασης. Το (Gustafsson et al., 2002) επιδεικνύει πώς οι μετρικές λογισμικού και τα αρχιτεκτονικά πρότυπα μπορούν να χρησιμοποιηθούν για τη διαχείριση της εξέλιξης λογισμικού. Η ποιότητα του τελικού συστήματος προβλέπεται μέσω της μελέτης της ιστορίας ανάπτυξης των προηγούμενων προγραμμάτων που είχαν παρόμοια σύνθεση χαρακτηριστικών, μετρικών και προτύπων. Η μέθοδος υποστηρίζεται από δύο ενσωματωμένα εργαλεία λογισμικού, ένα εργαλείο εξόρυξης προτύπων και μετρικών, το Maisa και το εργαλείο Columbus, που είναι ένα εργαλείο αντίστροφης μηχανικής. Στο (Geuheneuc et al., 2004) προτείνεται μια πειραματική μελέτη για τις κλάσεις που διαδραματίζουν τους ρόλους στα σχεδιαστικά μοτίβα. Για την μελέτη χρησιμοποιούνται μετρικές και ένας αλγόριθμος μηχανικής μάθησης για να επισημάνει τους ρόλους αυτούς. Οι τρόποι επισήμανσης επινοούνται πειραματικά χρησιμοποιώντας ένα αποθετήριο μικροαρχιτεκτονικών, παρόμοιων με τα σχεδιαστικά μοτίβα. Αποδεικνύεται ότι αυτοί οι τρόποι επισήμανσης βοηθούν στη μείωση του χρόνου αναζήτησης των μικροαρχιτεκτονικών χρησιμοποιώντας αποτελεσματικά το πρότυπο Composite και το πλαίσιο JHotDraw. Το (Muraki και Saeki, 2001) παρουσιάζει ένα είδος μετρικών λογισμικού που μας βοηθούν να εφαρμόσουμε τα πρότυπα σχεδίασης σε διαδικασίες επανασχεδίασης. Τα αντικειμενοστραφή σχέδια έχουν αναλυθεί και τα χαρακτηριστικά των σχεδίων χαμηλής ποιότητάς τους έχουν εξαχθεί. Με βάση αυτά τα χαρακτηριστικά, έχουν αναπτυχθεί 20 μετρικές για εναλλακτικές ροές και βάθος δέντρου ιεραρχίας. Αυτές οι μετρικές μας βοηθούν να αποφασίσουμε ποια πρότυπα σχεδίασης και πού πρέπει να εφαρμοστούν για να βελτιώσει το φτωχό σχεδιασμό. Τέλος (Khomh και Gueheneuc, 2009) παρουσιάζει μια περιγραφική και αναλυτική μελέτη των

κλάσεων που διαδραματίζουν μέχρι δύο ρόλους σε έξι διαφορετικά πρότυπα σχεδίασης. Τρεις ερευνητικές ερωτήσεις απαντιούνται δείχνοντας ότι στα προγράμματα υπάρχουν συχνά κλάσεις που διαδραματίζουν έναν ή δύο ρόλους, δεν είναι αμελητέες και υπάρχουν σημαντικές διαφορές μεταξύ των εσωτερικών και εξωτερικών χαρακτηριστικών τους.

2.3.1.4 Εφαρμογή προτύπων σχεδίασης

Στην τέταρτη κατηγορία άρθρων ανήκουν εκείνα που ασχολούνται με την εφαρμογή προτύπων σχεδίασης. Τα (Keepence και Mannion, 1999 και Bishop, 2008) βλέπουν τα πρότυπα σχεδίασης ως κλειδί για την πραγματοποίηση αφαίρεσης στην τεχνολογία λογισμικού. Το (Keepence και Mannion, 1999) χρησιμοποιεί πρότυπα σχεδίασης στοχεύοντας στην απλοποίηση των σύνθετων μοντέλων που θα υποστηρίζουν τη μεταβλητότητα. Το (Bishop, 2008) υποστηρίζει ότι τα πρότυπα σχεδίασης μπορούν να εφαρμοστούν με πολλούς τρόπους, και ότι όσο πιο υψηλό το επίπεδο των γλωσσικών χαρακτηριστικών γνωρισμάτων που χρησιμοποιούνται, τόσο ευκολότερη και πιο κατανοητή είναι οι εφαρμογή των προτύπων. Τα παραδείγματα που δίνονται αφορούν τα συντακτικά χαρακτηριστικά σε C# 3.0 καθώς επίσης και βιβλιογραφικές μεθόδους μέσω των οποίων οι εφαρμογή προτύπων γίνεται ευκολότερη Τέλος εξετάζεται η εφαρμογή και η επαναχρησιμοποίηση που προσφέρουν τα πρότυπα σχεδίασης.

Τα άρθρα (Briannd et al., 2006 και Meyer, 2006) ασχολούνται με την ανίχνευση προτύπων και αντιπροτύπων σε διαγράμματα UML και πηγαίο κώδικα αντίστοιχα. Στο (Briannd et al., 2006) παρουσιάζεται μια μεθοδολογία που στοχεύει στη χρήση των σωστών προτύπων από τους προγραμματιστές. Η προσέγγιση απεικονίζεται μέσω ενός παραδείγματος GoF προτύπων σχεδίασης. Επίσης αναπτύσσεται ένα εργαλείο πρωτοτύπων για να παρουσιάσει τη δυνατότητα πραγματοποίησης της προσέγγισης σε πρακτικές καταστάσεις, ενώ χρησιμοποιείται και μία μελέτη περίπτωσης, που έχει ενθαρρυντικά αποτελέσματα. Στο (Meyer, 2006) περιγράφεται μια προσέγγιση που υποστηρίζει την ανίχνευση των αντιπροτύπων που έχουν εφαρμοστεί σε πηγαίο κώδικα. Η προσέγγιση χωρίζεται σε τρία βασικά βήματα: την αναγνώριση αντιπροτύπων, τον μετασχηματισμό, και την επαλήθευση μετασχηματισμού, που υποστηρίζουν την επανασχεδίαση και μπορεί να χρησιμοποιηθεί για την αξιολόγηση υπάρχοντος.

Ένα άλλο θέμα που απασχολεί κάποια από τα άρθρα της κατηγορίας είναι η εφαρμογή προτύπων σχεδίασης με σκοπό να ωφελήσουν ή να ωφεληθούν από την αντικειμενοστραφή γλώσσα προγραμματισμού Java. Το (Palsberg και Jay, 1998) παρουσιάζει πώς μπορεί κανείς να προγραμματίσει visitors χωρίς να βασιστεί σε accept μεθόδους και χωρίς να γνωρίζει προκαταβολικά όλες τις κλάσεις των αντικειμένων. Έτσι προτείνεται η χρήση της κλάσης Walkabout της Java, που υποστηρίζει όλα τα visitors ως υποκλάσεις και έτσι μπορούν να χρησιμοποιηθούν χωρίς επιπτώσεις. Από την άλλη πλευρά, το (Cagnin et al., 2000) παρουσιάζει ένα πείραμα, στο οποίο μέρος του κατατετμημένου κώδικα μια προηγούμενης εργασίας επανασχεδιάστηκε σε αντικειμενοστραφή γλώσσα (Java) και τα αρχεία κειμένου που προηγουμένως χρησιμοποιούνταν αντικαταστάθηκαν από μια σχεσιακή βάση δεδομένων. Εδώ τα πρότυπα σχεδίασης χρησιμοποιούνται για να συνδέσουν τη βάση δεδομένων με τα αντικειμενοστραφή σενάρια σε Java. Στο άρθρο επιπλέον καταγράφεται ένα πείραμα συντήρησης, που συγκρίνει την ευκολία συντήρησης του παλιού συστήματος, το κατατετμημένο σύστημα και την αντικειμενοστραφή έκδοση.

Υπάρχουν άρθρα που προτείνουν νέα πρότυπα σχεδίασης. Το (Thu και Tran, 2007) προτείνει ένα σύνθετο πρότυπο σχεδίασης , που αποτελείται από τέσσερα πολύ γνωστά πρότυπα (adapter, strategy, abstract factory, and singleton), και σκοπεύει στην επίλυση του προβλήματος της προσαρμογής δυο ανεξάρτητων πτυχών. Το πρότυπο χρησιμοποιείται για την σύνδεση κλάσεων σε αντικειμενοστραφή πλαίσια, όταν κατασκευάζονται επαναχρησιμοποιήσιμες κλάσεις για συγκεκριμένες εργασίες λογισμικού. Πρόκειται για μια λύση στο πρόβλημα προσαρμογής σε δυο διαστάσεις. Το (Nelson, 1999) παρουσιάζει ένα πρότυπο σχεδίασης, το STESCA (Strategic-Tactical-Execution Software Control Architecture) που χρησιμοποιείται για τη δημιουργία συστημάτων ελέγχου λογισμικού. Το (MacDonald et al., 2002) περιγράφει μια νέα προσέγγιση για τη δημιουργία generative προτύπων σχεδίασης και συμπεριλαμβάνει τις εξής νέες ιδέες: Τα πρότυπα ορίζονται από ένα σύνολο παραμέτρων με συγκεκριμένες τιμές και ένα πρότυπο κώδικα που δημιουργεί πλαίσια των οποίων η δομή εξαρτάται από τους συνδυασμούς των τιμών για αυτές τις παραμέτρους. Η προσέγγιση απεικονίζεται με τη χρήση δυο εργαλείων, CO2P2S και Meta-CO2P2S, που υποστηρίζουν τη διαδικασία και χρησιμοποιούνται για την προσαρμογή και δημιουργία προτύπων σχεδίασης. Τέλος παρουσιάζονται 4 βήματα για να

μπορέσει ένας προγραμματιστής να χρησιμοποιήσει generative πρότυπα σχεδίασης καθώς και 4 βήματα για μπορέσει να δημιουργήσει generative πρότυπα σχεδίασης.

Αντίστοιχα, υπάρχουν και άρθρα που προτείνουν την ανάπτυξη μετασχηματισμών των προτύπων σχεδίασης. Στο (O Cinneide, 2000) παρουσιάζεται μια μεθοδολογία για την κατασκευή αυτοματοποιημένων μετασχηματισμών που παρουσιάζουν πρότυπα σχεδίασης. Αυτό επιτρέπει στους προγραμματιστές να καθυστερήσουν με ασφάλεια την εφαρμογή ενός προτύπου σχεδίασης μέχρι η ελαστικότητα που παρέχει να είναι η κατάλληλη. Βασικά συστατικά της προσέγγισης είναι η ικανότητα εφαρμογής των μετασχηματισμών στον υπάρχων κώδικα και η συντήρηση της συμπεριφοράς μετά από κάθε μετασχηματισμό. Ένα πρωτότυπο εργαλείο είναι υπό κατασκευή και τα αποτελέσματα μέχρι τώρα επιβεβαιώνουν την εγκυρότητα του άρθρου. Στο (O Cinneide και Nixon, 1999) παρουσιάζεται μια μεθοδολογία για την ανάπτυξη μετασχηματισμών των προτύπων σχεδίασης με τρόπο ώστε συντηρούν τη συμπεριφορά τους. Εξετάζονται τα ζητήματα του καθορισμού μιας αφετηρίας για το μετασχηματισμό, της αποσύνθεσης ενός προτύπου σε μικροπρότυπα (minipatterns) και της ανάπτυξης των αντίστοιχων μικρομετασχηματισμών, που μπορούν να εισαγάγουν αυτά τα minipatterns σε ένα πρόγραμμα. Συζητείται επίσης η αρχιτεκτονική ενός υπάρχοντος πρωτοτύπου λογισμικού και παρουσιάζονται τα αποτελέσματα της εφαρμογής αυτής της μεθοδολογίας για να αναπτύξουν έναν μετασχηματισμό του προτύπου Factory Method. Στο (O Cinneide και Nixon, 2001) αναπτύσσεται μια ακόμη μεθοδολογία για τη δημιουργία αυτοματοποιημένων μετασχηματισμών, οι οποίοι μπορούν να εφαρμόσουν ένα πρότυπο σχεδίασης σε ένα υπάρχον πρόγραμμα. Η μεθοδολογία έχει εφαρμοστεί στα πρότυπα σχεδίασης του Gamma et al. Τα αποτελέσματα είναι ελπιδοφόρα, αφού για τα περισσότερα πρότυπα θα μπορούσε να βρεθεί μια εφαρμόσιμη λύση, ενώ επιπλέον αποδείχθηκε και η εκτενής επαναχρησιμοποίηση των μικρομετασχηματισμών που αναπτύχθηκαν κατά τη διάρκεια αυτής της εργασίας.

Τέλος στην κατηγορία αυτή ανήκουν άρθρα που εφαρμόζουν πρότυπα σχεδίασης με σκοπό την απόκτηση προνομίων, όπως η βελτίωση της ποιότητας ή η ανέξοδη τεκμηρίωση σχεδιαστικών αποφάσεων. Το (Yau και Dong, 2000) παρουσιάζει μια προσέγγιση που εφαρμόζει πρότυπα σχεδίασης για την ολοκλήρωση των συστατικών. Τα πρότυπα σχεδίασης τυποποιούνται, για να

επιτρέψουν την αυτόματη δημιουργία στιγμιοτύπων. Ο συνδυασμός προτύπων σχεδίασης και βασιζόμενης στα συστατικά ανάπτυξης λογισμικού βελτιώνει την ποιότητα και παραγωγικότητα του λογισμικού. Το (Zdun et al., 2008) προτείνει μια βασιζόμενη σε πρότυπα διαδικασία αρχιτεκτονικής σεδίασης που στοχεύει στην ανέξοδη τεκμηρίωση των σχεδιαστικών αποφάσεων των αρχιτεκτονικών ιδεών παράλληλα με τη φυσική ροή του σχεδίου. Η προσέγγιση έχει αποδεχθεί στον τομέα των οδηγούμενων από τις διαδικασίες SOA για δύο αρχιτεκτονικές απόψεις: Συστατικό και σύνδεσμος και ροή διαδικασίας. Για την επικύρωση της προσέγγισης, έχει εφαρμοστεί ένα οδηγούμενο από μοντέλα εργαλείο ανάπτυξης λογισμικού. Με την τεκμηρίωση προτύπων σχεδίασης, που βασίζεται στην ανάλυση εννοιών, ασχολείται το άρθρο (Tonella και Antoniol, 1999). Η προσέγγιση αυτή εξάγει τις επαναλαμβανόμενες ομάδες κλάσεων που μοιράζονται κοινές δομικές σχέσεις καθώς επίσης και μη δομικές ιδιότητες, χωρίς να χρησιμοποιεί οποιασδήποτε προκαθορισμένη βιβλιοθήκη προτύπων, ενώ μπορεί να υιοθετηθεί τόσο σε επίπεδο κώδικα όσο και σε επίπεδο σχεδιασμού. Η ανάλυση εννοιών αποκάλυψε μια ισχυρή τεχνική ικανή να αναγνωρίσει άμεσα από τον κώδικα τις σημαντικές οργανώσεις κλάσεων. Η προσέγγιση εφαρμόστηκε σε μια C++ εφαρμογή. Το πρότυπο που προέκυψε από τη ανάλυση της εφαρμογής μπορεί να αποκτήσει νόημα και σημασία και να γίνει έτσι πολύτιμη πηγή πληροφόρησης για το σύστημα.

2.3.1.5 Γενικά ζητήματα σχετικά με τα πρότυπα σχεδίασης

Στη πέμπτη κατηγορία ανήκουν άρθρα γενικού περιεχομένου όσο αφόρα τα πρότυπα σχεδίασης. Τα άρθρα (Torchiano, 2002 και Sametinger και Riebish, 2002), αφορούν την τεκμηρίωση προτύπων με χρήση του εργαλείου Javadoc για τη δημιουργία τεκμηρίωσης σε HTML. Το πρώτο περιγράφει μια δομημένη προσέγγιση για την καταγραφή της χρήσης προτύπων στη γλώσσα Java. Η τεκμηρίωση που προκύπτει είναι καλά δομημένη, κάνοντας εύκολη την κατανόηση του προγράμματος. Στο δεύτερο παρουσιάζεται μια μεθοδολογία υποστήριξης αναπτυσσόμενων συστημάτων λογισμικού, μέσω μιας ομογενούς προσέγγισης τεκμηρίωσης για πρότυπα που μπορεί να χρησιμοποιηθούν για να δημιουργήσουν αναφορές για τα χαρακτηριστικά του σχεδίου και της ανιχνευσιμότητας.

Ένα άλλο θέμα που απασχολεί τα άρθρα (Schmidt και Buschmann, 2003 , Kerth και Cunningham, 1997 και Mellor και Johnson, 1997) της κατηγορίας είναι η σύγκριση παρόμοιων τεχνολογιών με σκοπό την αποσαφήνιση συγκεχυμένων όρων. Το (Schmidt και Buschmann, 2003) παρουσιάζει μια σύνοψη των προτύπων και των πλαισίων, περιγράφει πώς οι τεχνολογίες αυτές αλληλοσυμπληρώνουν η μία την άλλη για την καλύτερη επαναχρησιμοποίηση και παραγωγικότητα. Επιπλέον απεικονίζει τον τρόπο που τα πρότυπα εφαρμόζονται επιτυχώς στην πράξη για την βελτίωση της ποιότητας σύνθετων συστημάτων λογισμικού. Ο τρόπος συνεργασίας των τεχνολογιών παρουσιάζεται μέσω μιας μελέτης περίπτωσης για το πώς εφαρμόζονται στην πράξη τα πρότυπα και τα πλαίσια. Τα (Kerth και Cunningham, 1997 και Mellor και Johnson, 1997) σκοπεύουν στην αποσαφήνιση των όρων «αντικείμενο», «αρχιτεκτονική» και «πρότυπα», ορίζοντάς τα και παρουσιάζοντας τα από διαφορετικές σκοπιές. Μετά την ανάλυση των όρων παρουσιάζεται η φιλοσοφία βάση της οποίας οι όροι αυτοί χρησιμοποιούνται στα έργα.

Το (Tsai et al., 1999) εξετάζει τον έλεγχο προτύπων σχεδίασης σε ένα αντικειμενοστραφές πλαίσιο. Το άρθρο παρουσιάζει μια τεχνική για τη δημιουργία κάποιων σεναρίων προτύπων που μπορούν να χρησιμοποιηθούν για τη δημιουργία πολλών σεναρίων ελέγχου εφαρμογών που έχουν δημιουργηθεί με χρήση επεκτάσιμων προτύπων σχεδίασης και αντικειμενοστραφών πλαισίων. Η τεχνική ονομάζεται MfSS (Message Framework Sequence Specifications).Έπειτα απεικονίζονται πολλές διαφορετικές τεχνικές ελέγχου που βασίζονται στη χρήση τέτοιων σεναρίων προτύπων. Τέλος η προτεινόμενη τεχνική εφαρμόζεται για τον έλεγχο του πλαισίου μιας τράπεζας. Ως αποτέλεσμα τα σενάρια ελέγχου κατάφεραν να εντοπίσουν μεγάλο αριθμό λαθών που υπήρχαν στο πρόγραμμα. Το (Arcelli et al., 2008) εξετάζει αν η ανίχνευση προτύπων μπορεί να είναι χρήσιμη ως προς τη μεταφερσιμότητα. Το άρθρο επικεντρώνεται στον χρόνο που ξεκινά η διαδικασία μεταφοράς του συστήματος και θέτει το ερώτημα αν είναι εφικτό μια προσέγγιση βασισμένη στην ανίχνευση προτύπων, να δώσει χρήσιμες προτάσεις για την εύρεση συστατικών που θα χρησιμοποιηθούν ως υπηρεσίες. Το (Noda και Kishi, 2001) προτείνει έναν πιο ελαστικό τρόπο για την εφαρμογή προτύπων σχεδίασης, διαχωρίζοντας τα πρότυπα σχεδίασης από την κυρίως εφαρμογή που είναι υπεύθυνη για τις βασικές λειτουργίες. Την προσέγγιση υποστηρίζουν οι νέες τεχνολογίες υλοποίησης, όπως το Hyper/J™. Με τη χρήση ενός παραδείγματος,

φαίνεται ότι είναι δυνατή η ανταλλαγή προτύπων για την υποστήριξη διαφορετικών συμπεριφορών χωρίς καμία επιπλέον αλλαγή στην κυρίως εφαρμογή. Η προσέγγιση έχει ως πλεονέκτημα τη βελτίωση από άποψη επαναχρησιμοποίησης τόσο των προτύπων σχεδίασης όσο και των κυρίως εφαρμογών. Στο άρθρο (Tahvildari και Kontogiannis, 2002) παρουσιάζεται ένας νέος τρόπος αρχειοθέτησης των σχέσεων μεταξύ προτύπων σχεδίασης, που μπορούν να οδηγήσουν σε μια νέα κατηγοριοποίηση των προτύπων σε διαφορετικά στρώματα. Το άρθρο επίσης παρουσιάζει τον τρόπο που αυτό το σχήμα αρχειοθέτησης μπορεί να εφαρμοστεί για την επανασχεδίαση και αναδόμηση αντικειμενοστραφών συστημάτων. Το (Bortis και van der Hoek, 2008) παρουσιάζει την ιδέα pre-patterns στη μηχανική λογισμικού. Έχοντας ως έμπνευση την παρουσίαση των προτύπων σχεδίασης από τον Christopher Alexander, προτείνει μια νέα προσέγγιση για τη δημιουργία προτύπων που είναι ευρύτερα και μπορούν να εφαρμοστούν στις πρώτες φάσεις στης σχεδιαστικής διαδικασίας.

Τέλος, τα άρθρα (Buschmann et al., 2007 και Park et al., 2004) αναφέρονται γενικά στα πρότυπα σχεδίασης και την συμπεριφορά τους. Το (Park et al., 2004) προσπαθεί να προσεγγίσει στατιστικά την συμπεριφορά κάποιων προτύπων , που συμμετέχουν στη έρευνα, κατά τον χρόνο εκτέλεσης. Επίσης παρουσιάζεται ένα πείραμα που έγινε για το πρότυπο Command με τη χρήση του εργαλείου JHotDraw καθώς και οι πληροφορίες που ανιχνεύθηκαν, οι οποίες μπορούν να βοηθήσουν στη διαμόρφωση της δομής του προτύπου που υλοποιείται σε κώδικα .

2.3.2 Επίδραση των Προτύπων Σχεδίασης στην Εξωτερική Ποιότητα του Λογισμικού

Το κεφάλαιο αυτό, ανακεφαλαιώνει και συζητά τα αποτελέσματα σχετικά με την επίδραση της εφαρμογής αντικειμενοστραφών προτύπων σχεδίασης στην εξωτερική ποιότητα. Σύμφωνα με τα αποτελέσματα της έρευνάς μας, τα πιο συνηθισμένα ποιοτικά χαρακτηριστικά είναι η Επεκτασιμότητα, η Κατανόηση και η Επαναχρησιμοποίηση. Στον Πίνακα 5, παρουσιάζεται η επίδραση κάθε προτύπου σε κάθε ποιοτικό χαρακτηριστικό. Το σύμβολο (+) σημαίνει ότι το πρότυπο επιδρά θετικά στο συγκεκριμένο ποιοτικό χαρακτηριστικό, ενώ το (-) ότι επιδρά αρνητικά.

Τέλος, μέσα σε παρένθεση, υπάρχει η αναφορά από την οποία προέκυψε το κάθε σύμβολο.

Πίνακας 5: Επίδραση των Προτύπων Σχεδίασης στην εξωτερική ποιότητα λογισμικού

Ποιοτικό Χαρακτηριστικό	Abstract Factory	Builder	Factory Method	Prototype	Singleton	Adapter
Επεκτασιμότητα	+ [S055]	+ [S055]	+ [S055]	+ [S055]	- [S055]	+ [S055]
Πιθανότητα Αλλαγής					- [S043]	- [S043]
Επαναχρησιμοποίηση	+ [S055]	- [S055]	+ [S055]	+ [S055]	- [S055]	+ [S055]
Σταθερότητα	+ [S097]					+ [S038]
Κατανόηση	- [S098] - [S055]	+ [S055]	- [S055]	+ [S055]	+ [S055]	- [S055]
Ευχρηστία	- [S039]					
Κατάτμηση	+ [S055]					
Γενικότητα	+ [S055]					
Κλιμάκωση	- [S055]					
Ευρωστία	- [S055]					

Ποιοτικό Χαρακτηριστικό	Composite	Decorator	Facade	Flyweight	Bridge	Proxy
Επεκτασιμότητα	+ [S055]	+ [S098] + [S084] + [S055]	+ [S055]	- [S055]	+ [S055] + [S003]	- [S055] + [S059]
Πιθανότητα Αλλαγής						- [S043]
Επαναχρησιμοποίηση	+ [S055]	- [S055]	- [S055]	- [S055]	- [S055]	+ [S055]
Σταθερότητα		- [S084]	+ [S038]		+ [S038]	
Κατανόηση	- [S098] + [S055]	+ [S098] - [S084] - [S055]	+ [S055]	- [S055]	+ [S055]	+ [S101] - [S055]
Κατάτμηση	+ [S055]			- [S055]	+ [S101]	- [S101]
Γενικότητα	+ [S055]			- [S055]		
Κλιμάκωση	- [S055]			+ [S055]		
Ευρωστία	- [S055]			- [S055]		

Ποιοτικό Χαρακτηριστικό	Command	Interpreter	Iterator	Mediator	Memento	Observer
Επεκτασιμότητα	+ [S055]	+ [S055]	+ [S055]	+ [S055]	- [OS55]	+ [S055]
Πιθανότητα Αλλαγής	+ [S043]					
Επαναχρησιμοποίηση	- [S055]	+ [S055]	+ [S055]	- [S055]	- [S055]	+ [S055]
Σταθερότητα						- [S097]
Κατανόηση	- [S055]	+ [S055]	+ [S055]	+ [S055]	- [S055]	+ [S098] - [S084] - [S055]
Ευκολία Ελέγχου				- [S012]		- [S012]

Ποιοτικό Χαρακτηριστικό	State	Template Method	Strategy	Visitor	Chain of Responsibility
Επεκτασιμότητα	+ [S055] + [S003]	+ [S055]	+ [S055]	- [S098] + [S084] + [S055]	+ [S055]
Πιθανότητα Αλλαγής	- [S043]		- [S043]		

Ποιοτικό Χαρακτηριστικό	State	Template Method	Strategy	Visitor	Chain of Responsibility
Επαναχρησιμοποίηση	- [S055]	+ [S055]	- [S055]	- [S055]	+ [S055]
Σταθερότητα		+ [S097]			
Κατανόηση	+ [S055]	- [S055]	+ [S055]	- [S098] + [S055]	- [S055]
Ευκολία Ελέγχου				- [S012]	

Σύμφωνα με τα αποτελέσματα του Πίνακα 5 18 από τα 23 πρότυπα σχεδίασης, έχουν θετική επίδραση στην επεκτασιμότητα του λογισμικού. Τα αποτελέσματα σχετικά με την επαναχρησιμοποίηση είναι μοιρασμένα σχεδόν στη μέση, ωστόσο σχόλια πάνω στο θέμα καταγράφονται μόνο στο [S055]. Επιπλέον η κατανόηση των προτύπων, φαίνεται να είναι το πιο ασαφές χαρακτηριστικό αφού 6 πρότυπα σχεδίασης (Visitor, Composite, Decorator, Proxy, Observer and Abstract Factory) σε άλλες μελέτες φαίνεται να θεωρούνται εύκολα κατανοητά, ενώ σε άλλες δυσνόητα. Τέλος, διάφορα ποιοτικά χαρακτηριστικά, όπως η Ευχρηστία, η Κατάτμηση, η Γενίκευση, η Κλιμάκωση και η Ευρωστία, δεν έχουν ακόμα ερευνηθεί σε βάθος.

2.3.3 Επίδραση των Προτύπων Σχεδίασης στην Εσωτερική Ποιότητα του Λογισμικού

Στο κεφάλαιο αυτό, ανακεφαλαιώνουμε και συζητάμε την επίδραση της εφαρμογής προτύπων σχεδίασης στην εσωτερική ποιότητα του λογισμικού. Στον Πίνακα 6, το σύμβολο (+) σημαίνει ότι το πρότυπο επιδρά θετικά στο συγκεκριμένο ποιοτικό χαρακτηριστικό, ενώ το (-) ότι επιδρά αρνητικά. Ανάλογα με τη φύση κάθε χαρακτηριστικού το σύμβολο (+) μπορεί να σημαίνει αύξηση ή μείωση κάποιας μετρικής. Προκειμένου ένα πρότυπο να έχει θετική επιρροή οι μετρικές που υπολογίζουν πολυπλοκότητα, σύζευξη, κληρονομικότητα και μέγεθος πρέπει να μειώνονται, ενώ, οι μετρική της συνοχής αν αυξάνεται. Επιπλέον, σε κάθε κελί καταγράφεται η μετρική που χρησιμοποιείται για την αξιολόγηση του προτύπου και μέσα σε παρένθεση, υπάρχει η αναφορά από την οποία προέκυψε το κάθε σύμβολο.

Πίνακας 6: Επίδραση των Προτύπων Σχεδίασης στην εσωτερική ποιότητα λογισμικού

Πρότυπο Σχεδίασης	Πολυπλοκότητα	Συνοχή	Σύζευξη	Ενθυλάκωση
Abstract Factory			CF, + [S047]	
Adapter	WMPC2, + [S045]			

Πρότυπο Σχεδίασης	Πολυπλοκότητα	Συνοχή	Σύζευξη	Ενθυλάκωση
Bridge	AC, + [S003] WMPC1, + [S003] WMPC2, + [S003]	LCOM, + [S003]	CF, + [S003]	
Proxy		H, - [S059]	Ce, - [S059]	
Iterator				DAM, + [S047]
Mediator			CF, + [S049]	
Observer			CF, + [S047] CBO, + [S045]	
State	AC, - [S003] WMPC1, + [S003] WMPC2, + [S003]	LCOM, + [S003]	CF, + [S003]	
Visitor	WMPC1, - [S045] WMPC1, + [S049]		CBO, + [S045]	

Πρότυπο Σχεδίασης	Μέγεθος	Πολυμορφισμός	Κληρονομικότητα
Bridge	LOC, - [S101] LOC, - [S003] NOC, - [S003]		DIT, + [S049] NOC, + [S049]
Proxy	LOC, - [S101]		A, - [S059]
Command	LOC, - [S101]		
Observer	LOC, - [S101]		
State	LOC, + [S003]		
Strategy		NOP, + [S047]	
Visitor			NMI, + [S045]

Σύμφωνα με τα αποτελέσματα του Πίνακα 6, το πρότυπο Bridge ερευνάται αυστηρότερα από τα υπόλοιπα πρότυπα. Στις περιπτώσει εφαρμογής του, οι μετρικές συνοχής, σύζευξης, πολυπλοκότητας και κληρονομικότητας βελτιώνονται, ενώ οι μετρικές του μεγέθους επιδεινώνονται. Τα πιο συνηθισμένα χαρακτηριστικά εσωτερικής ποιότητας που εξετάζονται, είναι η συνοχή, το μέγεθος και η πολυπλοκότητα. Αντίθετα ο πολυμορφισμός και η ενθυλάκωση έχουν απασχολήσει μόνο μία μελέτη.

Γενικά, παρατηρήθηκε ότι η έρευνα των εσωτερικών ποιοτικών χαρακτηριστικών είναι λιγότερο εντατική από αυτή των εξωτερικών ποιοτικών χαρακτηριστικών, τόσο από άποψη αριθμού μελετών όσο και από άποψη αριθμού χαρακτηριστικών και ποικιλίας προτύπων. Το γεγονός αυτό, φανερώνει ένα κενό στην έρευνα των προτύπων σχεδίασης που μπορεί να συντελέσει στη δημιουργία ενός θέματος εντατικής έρευνας στο κοντινό μέλλον.

2.3.4 Απλούστερες Σχεδιαστικές Λύσεις

Το κεφάλαιο αυτό στοχεύει στη σύνοψη των πρωταρχικών μελετών που παρουσιάζουν απλούστερες σχεδιαστικές λύσεις από τη χρήση προτύπων σχεδίασης, που παρέχουν ωστόσο ισοδύναμη λειτουργικότητα. Οι λύσεις αυτές,

χρησιμοποιούνται στη βιβλιογραφία προκειμένου να αξιολογήσουν τα πρότυπα σχεδίασης και θεωρούμε ότι η δημιουργία ενός αποθετηρίου τέτοιων σχεδίων μπορεί να αποδειχθεί ιδιαίτερα ευεργετική σε μελλοντικές έρευνες. Ο Πίνακας 7, παρουσιάζει τα πρότυπα για τα οποία βρέθηκαν απλούστερες σχεδιαστικές λύσεις, οι μελέτες στις οποίες αναφέρονταν η ύπαρξή τους και ο τρόπος παρουσίασης της λύσης. Κάθε λύση μπορεί να περιγραφεί με τρείς τρόπους: (α) διάγραμμα κλάσης, (β) πηγαίος κώδικας και (γ) συνδυασμός και των δυο.

Πίνακας 7: Απλούστερες σχεδιαστικές λύσεις ισοδύναμες με Πρότυπα Σχεδίασης

Πρότυπο Σχεδίασης	Μελέτη	Διάγραμμα κλάσεων	Πηγαίος Κώδικας
Bridge	[S003]	yes	no
	[S049]	yes	no
State	[S003]	yes	no
	[S011]	no	yes
Abstract Factory	[S039]	no	yes
Decorator	[S047]	yes	no
Mediator	[S047]	yes	no
	[S049]	yes	no
Visitor	[S049]	yes	no
	[S084]	no	yes
	[S098]	no	yes
Proxy	[S059]	yes	no
Factory Method	[S069]	no	yes
Decorator	[S069]	yes	yes
	[S084]	no	yes
	[S098]	no	yes
Iterator	[S070]	yes	yes
Observer	[S084]	no	yes
	[S098]	no	yes
Composite	[S084]	no	yes
	[S098]	no	yes

Μέχρι σήμερα, έχουν προταθεί απλούστερες σχεδιαστικές λύσεις για μόνο 12 από τα 23 πρότυπα που περιγράφονται στο (Gamma et al., 1995). Όλες τους, περιγράφονται ικανοποιητικά, ώστε να μπορούν στο μέλλον να υποστηρίξουν ερευνητικές προσεγγίσεις. Στο [S003] οι συγγραφείς ερεύνησαν ένα έργο ανοιχτού λογισμικού που χρησιμοποιεί εμφωλευμένες εντολές if, στην κλάση πελάτη αντί να εφαρμόσει το πρότυπο. Στο [S049], οι συγγραφείς λένε ότι χρησιμοποιώντας ένα βαθύτερο δέντρο κληρονομικότητας μπορεί να υποκαταστήσει το πρότυπο σχεδίασης Bridge. Επιπλέον, τα [S003 και S011] προτείνουν ότι η χρήση του προτύπου State, μπορεί να αντικατασταθεί από ένα τμήμα κώδικα στην κλάση πελάτης, που χρησιμοποιεί πολλαπλές εναλλακτικές ροές, προκειμένου να υλοποιήσει τη συμπεριφορά συγκεκριμένων καταστάσεων. Παρομοίως, στα [S039 και S069] τα πρότυπα Abstract Factory και Factory Method μπορούν να

υλοποιηθούν με εμφωλευμένες εντολές if στην κλάση πελάτη, απορρίπτοντας τον πολυμορφισμό που χαρακτηρίζει το πρότυπο. Στα [S047, S069, S084 και S098] παρουσιάζεται μια εναλλακτική λύση του προτύπου Decorator, που χρησιμοποιεί ένα βαθύτερο δέντρο κληρονομικότητας, για να παράγει μια ισοδύναμη λύση με το πρότυπο σχεδίασης.

Όσο αφορά τον πρότυπο Mediator στα [S047 και S049] οι συγγραφείς λένε ότι μια απλή λύση που δεν χρησιμοποιεί το πρότυπο, αλλά προσφέρει ισοδύναμη λειτουργικότητα, μπορεί να σχεδιαστεί συνδέοντας όλες τις σχετικές κλάσεις άμεσα και όχι μέσω την κλάσης Mediator. Στα [S049, S084 και S098] προτείνεται μια εναλλακτική του προτύπου Visitor. Η απλούστερη λύση εφαρμόζει πολλαπλές λειτουργίες, αντί να «δέχεται» και να «επισκέπτεται» μεθόδους, και μία μόνο ιεραρχία κλάσεων. Η μεταβίβαση αρμοδιοτήτων, χειρίζεται μέσω μιας εμφωλευμένης, εντολής if,για κάθε υποκατηγορία της υπόλοιπης ιεραρχίας.

Στο [S070] παρουσιάζονται τέσσερεις διαφοροποιήσεις του προτύπου Iterator. Όσο αφορά τα πρότυπα σχεδίασης Observer και Composite, στα [S084 και S098] οι συγγραφείς παραλείπουν την απλούστερη σχεδιαστική λύση από το εγχειρίδιό, είναι όμως διαθέσιμη ώς συμπληρωματικό υλικό και ως τεχνική αναφορά αντίστοιχα. Και στις δυο μελέτες μελετάται το ίδιο σύνολο προτύπων και απλούστερων σχεδιαστικών λύσεων.

2.4 ΚΙΝΔΥΝΟΙ ΕΓΚΥΡΟΤΗΤΑΣ

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

3 ΕΜΠΕΙΡΙΚΗ ΜΕΛΕΤΗ ΧΡΗΣΗΣ ΠΡΟΤΥΠΩΝ ΣΧΕΔΙΑΣΗΣ ΣΤΟ ΑΝΟΙΧΤΟ ΛΟΓΙΣΜΙΚΟ

Η ενότητα αυτή, στοχεύει στην διερεύνηση της χρήσης αντικειμενοστραφών προτύπων σχεδίασης σε λογισμικό ανοιχτού κώδικα. Ειδικότερα χρησιμοποιήσαμε μια εμπειρική μέθοδο, δηλ μια μελέτη περίπτωσης ώστε να αξιολογήσουμε ποια πρότυπα χρησιμοποιούνται πιο συχνά στα παιχνίδια ανοιχτού λογισμικού, ποιες διαφορές εμφανίζονται μεταξύ των διάφορων κατηγοριών των παιχνιδιών και ποιοι είναι οι κυριότεροι παράγοντες που επηρεάζουν τη χρήση των προτύπων σχεδίασης.

3.1 ΜΕΘΟΔΟΛΟΓΙΑ

Σύμφωνα με την μεθοδολογία διεξαγωγής μιας μελέτης περίπτωσης που περιγράφεται αναλυτικά στο κεφάλαιο 1.5.2, παρουσιάζουμε την μεθοδολογία με την οποία δουλέψαμε, δηλαδή τα ερωτήματα της έρευνας, τη διαδικασία που ακολουθήθηκε κατά τη μελέτη περίπτωσης και τις μεθόδους ανάλυσης των δεδομένων

3.1.1 Τα ερωτήματα της έρευνας

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

- ***RQ1:*** Ποια πρότυπα σχεδίασης χρησιμοποιούνται συχνότερα στα παιχνίδια ανοιχτού λογισμικού;
- ***RQ2:*** Υπάρχουν διαφορές στα πρότυπα που χρησιμοποιούνται από κατηγορία σε κατηγορία;
- ***RQ3:*** Ποιοι είναι οι παράγοντες που επηρεάζουν κατά κύριο λόγο την χρήση των προτύπων σχεδίασης;

3.1.2 Πλάνο της μελέτης περίπτωσης

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

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

Στη μελέτη αυτή, οι εφαρμογές που επιλέξαμε είναι παιχνίδια ανοιχτού λογισμικού που χωρίζονται σε 10 επιμέρους κατηγορίες: 1) Role Playing Games 2) Simulation Games 3) Board Games 4) Arcade Games 5) Multi User Games 6) Turn-Based Strategy Games 7) Puzzle Games 8) First Person Shooter Games 9) Real-Time Strategy Games 10) Card Games. Οι κατηγορίες αυτές επιλέχθηκαν ως οι δημοφιλέστερες .

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

- είναι γραμμένα σε java, σύμφωνα με τους περιορισμούς του εργαλείου που χρησιμοποιούμε για την ανίχνευση των προτύπων (Tsantalis et al., 2006).
- διαθέτουν δυαδικό κώδικα, σύμφωνα με τους περιορισμούς του εργαλείου που χρησιμοποιούμε για την ανίχνευση των προτύπων
- πρόκειται για λογισμικά που έχουν τουλάχιστον 100 λήψεις, έτσι ώστε να μπορούμε να τα θεωρήσουμε ενεργά

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

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

προγραμματιστικής κοινότητας, η κατανομή εκείνων που έχουν προγραμματιστική ικανότητα και εμπειρία πλησιάζει κατά πολύ την κατανομή του πληθυσμού.

3.1.3 Μέθοδοι Ανάλυσης Δεδομένων

Το σύνολο των δεδομένων που προέκυψαν μετά την ανίχνευση των προτύπων αποτελεί μέρος των αριθμητικών μας δεδομένων. Ωστόσο κάποιες τεχνικές που χρησιμοποιήθηκαν κατά την ανάλυση των δεδομένων απαιτούσαν ανάλογα κατηγορικές ή δυαδικές μεταβλητές. Έτσι, όπως παρουσιάζεται και στο Παράρτημα Δ, έγιναν κάποιες μετατροπές στην μορφή των δεδομένων. Συμπληρώνοντας τη φάση της προεπεξεργασίας κάθε εφαρμογή χαρακτηρίζεται από 57 μεταβλητές:

1. αύξων αριθμός
2. όνομα
3. κατηγορία
4. έτος της τελευταίας έκδοσης
5. αριθμός ημερών που ήταν ενεργή
6. αριθμός λήψεων
7. αριθμός προγραμματιστών
8. αριθμός εκδόσεων
9. αριθμός κλάσεων
10. αριθμός στιγμιοτύπων του προτύπου factory
11. αριθμός στιγμιοτύπων του προτύπου prototype
12. αριθμός στιγμιοτύπων του προτύπου singleton
13. αριθμός στιγμιοτύπων των creational προτύπων
14. αριθμός στιγμιοτύπων του προτύπου adapter
15. αριθμός στιγμιοτύπων του προτύπου composite
16. αριθμός στιγμιοτύπων του προτύπου decorator
17. αριθμός στιγμιοτύπων του προτύπου proxy
18. αριθμός στιγμιοτύπων των structural προτύπων
19. αριθμός στιγμιοτύπων του προτύπου observer
20. αριθμός στιγμιοτύπων του προτύπου state-strategy
21. αριθμός στιγμιοτύπων του προτύπου template
22. αριθμός στιγμιοτύπων του προτύπου visitor
23. αριθμός στιγμιοτύπων των behavioural προτύπων

24. 20 κατηγορικές μεταβλητές για τις μεταβλητές 4-23

25. 14 δυαδικές μεταβλητές για τις μεταβλητές 10-23

Κατά τη διάρκεια της ανάλυσης στην έρευνά μας, χρησιμοποιήσαμε στατιστικές τεχνικές και τεχνικές clustering. Αυτές οι τεχνικές είναι:

- Descriptive statistics
- Independent sample t-test
- Paired sample t-test
- K-Means clustering
- Pearson x^2 test

Όσον αφορά το πρώτο ερώτημα έρευνας (*RQ1)*, χρησιμοποιήσαμε descriptive statistics και paired sample t-tests ώστε να συγκρίνουμε τις μέσες τιμές των δειγμάτων για κάθε πρότυπο σχεδίασης. Στην έρευνα σχετικά με το δεύτερο ερώτημα (*RQ2)*, για τους ίδιους λόγους χρησιμοποιήσαμε descriptive statistics και independent sample t-tests. Τέλος, σχετικά με το 3^o ερώτημα της έρευνας μας (*RQ3)*, εκτελέσαμε Pearson x^2 test και K-Means clustering. Η στατιστική ανάλυση και το two-step clustering έγιναν με τη χρήση SPSS©.

3.2 ΑΠΟΤΕΛΕΣΜΑΤΑ

Στην ενότητα αυτή, παρουσιάζουμε τα ευρήματα της εμπειρικής μας μελέτης. Στον Πίνακα 8, φαίνεται η μέση τιμή για όλα τα πρότυπα σχεδίασης, ο μέγιστος αριθμός προτύπων που εντοπίσθηκαν σε μία εφαρμογή και η τυπική απόκλιση για την κάθε μεταβλητή . Τα αποτελέσματα αφορούν ολόκληρο το σύνολο δεδομένων χωρίς διάκριση μεταξύ των κατηγοριών.

Πίνακας 8: Μέσος όρος χρήσης των προτύπων

	Maximum	**Mean**	**Std. Deviation**
Factory	12	0,71	1,87
Prototype	37	0,82	4,04
Singleton	60	6,11	12,03
Creational	*104*	*7,64*	*15,36*
Adapter	223	12,64	32,48
Composite	2	0,08	0,31
Decorator	24	1,15	3,88
Proxy	10	0,62	1,70
Structural	*232*	*14,49*	*35,63*

	Maximum	Mean	Std. Deviation
Observer	9	0,55	1,39
State	180	12,43	30,17
Template	16	1,74	3,24
Visitor	68	0,71	6,80
Behavioural	*229*	*15,43*	*36,79*

Τα αποτελέσματα του Πίνακα 8 παρέχουν ενδείξεις για το επίπεδο εφαρμογής του κάθε προτύπου στις εφαρμογές ανοιχτού λογισμικού. Προκειμένου να μπορέσουμε να συγκρίνουμε τις μέσες τιμές των μεταβλητών με έναν τρόπο που θα μας δώσει πιο ασφαλή συμπεράσματα, εκτελέσαμε 55 paired sample t-tests, δηλαδή, έναν έλεγχο για κάθε δυνατό συνδυασμό μεταξύ δυο προτύπων σχεδίασης. Τα αποτελέσματα ενός t-test μεταξύ δυο μεταβλητών μεταφράζεται με δυο νούμερα, τη μέση διαφορά (diff) και την σημαντικότητα του t-test (sig). Η μεταβλητή *diff* αντιπροσωπεύει τη διαφορά που προκύπτει, αφαιρώντας τη μέση τιμή της δεύτερης μεταβλητής από τη μέση τιμή της πρώτης, ενώ, η μεταβλητή *sig* αναπαριστά την πιθανότητα η μεταβλητή *diff* να μην είναι στατιστικά σημαντική. Στον Πίνακα 9, παρουσιάζουμε τις στατιστικά σημαντικές διαφορές στην εφαρμογή των προτύπων.

Πίνακας 9: Σημαντικά paired sample t-tests για τις διαφοροποιήσεις στη χρήση των προτύπων

	diff	Sig
Factory – Singleton	-5,40	0.00
Factory – Adapter	-11,93	0.00
Factory – Composite	0,63	0.00
Factory – State	-11,72	0.00
Factory – Template	-1,03	0.00
Prototype – Singleton	-5,29	0.00
Prototype – Adapter	-11,82	0.00
Prototype – State	-11,61	0.00
Prototype – Template	-0,92	0.02
Singleton – Adapter	-6,53	0.02
Singleton – Composite	6,03	0.00
Singleton – Decorator	4,96	0.00
Singleton – Proxy	5,49	0.00
Singleton – Observer	5,56	0.00
Singleton – State	-6,32	0.01
Singleton – Template	4,37	0.00
Singleton – Visitor	5,40	0.00

	diff	Sig
Adapter – Composite	12,56	0.00
Adapter – Decorator	11,49	0.00
Adapter – Proxy	12,02	0.00
Adapter – Observer	12,09	0.00
Adapter – Template	10,90	0.00
Adapter – Visitor	11,93	0.00
Composite – Decorator	-1,70	0.01
Composite – Observer	-0,54	0.00
Composite – Proxy	-0,47	0.00
Composite – State	-12,35	0.00
Composite – Template	-1,66	0.00
Decorator – State	-11,28	0.00
Decorator – Template	-0,59	0.04
Proxy – State	-11,81	0.00
Proxy – Template	-1,12	0.00
Observer – State	-11,88	0.00
Observer – Template	-1,19	0.00
State – Template	10,69	0.00
State – Visitor	11,72	0.00
Creational - Structural	-6,85	0,02
Creational - Behavioural	-7,79	0,01

Στον Πίνακα 10, παρουσιάζουμε τη μέση τιμή που προέκυψε από κάθε κατηγορία παιχνιδιών για κάθε πρότυπο σχεδίασης. Στον πίνακα αυτό, δε συμπεριλάβαμε τις μέγιστες τιμές και την τυπική απόκλιση καθαρά για λόγους εξοικονόμησης χώρου.

Πίνακας 10: Μέσος όρος χρήσης όλων των προτύπων ανά κατηγορία παιχνιδιού.

	Role Playing	Simulation	Board Game	Arcade	Multi -User	Turn Based	Puzzle	Shooter	Real Time Strategy	Card Game
Factory	0,40	0,10	1,40	0,10	1,10	1,00	0,30	0,50	2,10	0,10
Prototype	0,80	0,20	3,80	0,20	0,00	1,20	1,60	0,00	0,00	0,40
Singleton	13,60	2,00	11,80	1,90	4,70	13,70	1,60	1,10	8,90	1,80
Creational	14,80	2,30	17,00	2,20	5,80	15,90	3,50	1,60	11,00	2,30
Adapter	8,20	24,10	19,20	1,90	10,70	28,60	4,90	5,50	19,30	4,00
Composite	0,20	0,10	0,30	0,00	0,00	0,00	0,00	0,00	0,20	0,00
Decorator	1,60	0,50	2,10	0,00	2,60	0,60	0,40	0,40	3,30	0,00

	Role Playing	Simulation	Board Game	Arcade	Multi -User	Turn Based	Puzzle	Shooter	Real Time Strategy	Card Game
Proxy	1,60	0,30	0,90	0,00	0,30	0,80	0,00	0,40	1,70	0,20
Structural	11,60	25,00	22,50	1,90	13,60	30,00	5,30	6,30	24,50	4,20
Observer	0,60	0,00	0,60	0,10	1,20	1,40	0,10	1,00	0,40	0,10
State	9,90	5,60	25,10	2,10	15,40	19,00	2,80	8,00	30,80	5,60
Template	1,80	0,60	3,90	0,40	3,40	2,40	0,80	0,50	2,10	1,50
Visitor	0,00	0,00	0,00	0,00	0,30	0,00	0,00	0,00	6,80	0,00
Behavioural	12,30	6,20	29,60	2,60	20,30	22,80	3,70	9,50	40,10	7,20

Προκειμένου να επικυρώσουμε στατιστικά τα αποτελέσματα του προηγούμενου πίνακα εκτελέσαμε 630 Independent sample t-tests και 550 Paired Sample t-tests, δηλαδή, ένα Independent sample t-test για κάθε πρότυπο για όλα τα δυνατά ζευγάρια κατηγοριών και ένα Paired Sample t-test για όλα τα δυνατά ζευγάρια προτύπων σε κάθε κατηγορία.

Πιο συγκεκριμένα οι πίνακες 11.1- 11.11 επισημαίνουν εκείνες τις κατηγορίες παιχνιδιών ανά πρότυπο, που έχουν τιμές στατιστικά σημαντικές , χρησιμοποιούν δηλαδή το αντίστοιχο πρότυπο στατιστικά σημαντικά περισσότερες φορές συγκριτικά με τις υπόλοιπες κατηγορίες.

Πίνακας 11.1 : Independent sample t-test για το πρότυπο Factory

	Role Playing Games	Simulator	Board Games	Arcade	Multi-User	Turn Based Strategy	Puzzle	Shooter	Real Time Strategy	Card Games
Role Playing Games	-	0.04	0.27	0.36	0.42	0.32	0.79	0.85	0.21	0.36
Simulator	-	-	0.14	1.00	0.24	0.11	0.41	0.35	0.14	1.00
Board Games	-	-	-	0.14	0.79	0.68	0.22	0.34	0.64	0.14
Arcade	-	-	-	-	0.24	0.11	0.41	0.35	0.14	1.00
Multi-User	-	-	-	-	-	0.92	0.34	0.51	0.51	0.24
Turn Based Strategy	-	-	-	-	-	-	0.21	0.44	0.42	0.11
Puzzle	-	-	-	-	-	-	-	0.67	0.19	0.40
Shooter	-	-	-	-	-	-	-	-	0.25	0.35
Real Time Strategy	-	-	-	-	-	-	-	-	-	0.14
Card Games	-	-	-	-	-	-	-	-	-	-

Πίνακας 11.2 : Independent sample t-test για το πρότυπο Prototype

	Role Playing Games	Simulator	Board Games	Arcade	Multi-User	Turn Based Strategy	Puzzle	Shooter	Real Time Strategy	Card Games
Role Playing Games	-	0.17	0.43	0.17	0.05	0.69	0.56	0.05	0.05	0.47
Simulator	-	-	0.36	1.00	0.34	0.31	0.31	0.34	0.34	0.66
Board Games	-	-	-	0.36	0.33	0.50	0.58	0.33	0.33	0.37
Arcade	-	-	-	-	0.34	0.31	0.32	0.34	0.34	0.66
Multi-User	-	-	-	-	-	0.22	0.25	N/A	N/A	0.34
Turn Based Strategy	-	-	-	-	-	-	0.80	0.22	0.22	0.43
Puzzle	-	-	-	-	-	-	-	0.25	0.25	0.39
Shooter	-	-	-	-	-	-	-	-	N/A	0.34
Real Time Strategy	-	-	-	-	-	-	-	-	-	0.34
Card Games	-	-	-	-	-	-	-	-	-	-

Πίνακας 11.3 : Independent sample t-test για το πρότυπο Singleton

	Role Playing Games	Simulator	Board Games	Arcade	Multi-User	Turn Based Strategy	Puzzle	Shooter	Real Time Strategy	Card Games
Role Playing Games	-	0.03	0.82	0.03	0.17	0.99	0.03	0.02	0.46	0.03
Simulator	-	-	0.15	0.95	0.54	0.07	0.73	0.41	0.14	0.87
Board Games	-	-	-	0.15	0.36	0.82	0.12	0.12	0.70	0.14
Arcade	-	-	-	-	0.54	0.07	0.84	0.59	0.14	0.95
Multi-User	-	-	-	-	-	0.22	0.48	0.41	0.49	0.51
Turn Based Strategy	-	-	-	-	-	-	0.06	0.05	0.50	0.07
Puzzle	-	-	-	-	-	-	-	0.50	0.12	0.83
Shooter	-	-	-	-	-	-	-	-	0.09	0.40
Real Time Strategy	-	-	-	-	-	-	-	-	-	0.13
Card Games	-	-	-	-	-	-	-	-	-	-

Πίνακας 11.4 : Independent sample t-test για το πρότυπο Adapter

	Role Playing Games	Simulator	Board Games	Arcade	Multi-User	Turn Based Strategy	Puzzle	Shooter	Real Time Strategy	Card Games
Role Playing Games	-	0.49	0.42	0.06	0.76	0.22	0.38	0.53	0.28	0.18
Simulator	-	-	0.85	0.34	0.57	0.87	0.40	0.42	0.85	0.39
Board Games	-	-	-	0.22	0.58	0.65	0.30	0.32	1.00	0.26
Arcade	-	-	-	-	0.27	0.12	0.26	0.30	0.10	0.19
Multi-User	-	-	-	-	-	0.31	0.47	0.53	0.48	0.40
Turn Based Strategy	-	-	-	-	-	-	0.17	0.18	0.62	0.15
Puzzle	-	-	-	-	-	-	-	0.88	0.17	0.73
Shooter	-	-	-	-	-	-	-	-	0.19	0.66
Real Time Strategy	-	-	-	-	-	-	-	-	-	0.14
Card Games	-	-	-	-	-	-	-	-	-	-

Πίνακας 11.5 : Independent sample t-test για το πρότυπο Composite

	Role Playing Games	Simulator	Board Games	Arcade	Multi-User	Turn Based Strategy	Puzzle	Shooter	Real Time Strategy	Card Games
Role Playing Games	-	0.56	0.70	0.17	0.17	0.15	0.20	0.17	1.00	0.17
Simulator	-	-	0.41	0.34	0.34	0.34	0.34	0.34	0.56	0.34
Board Games	-	-	-	0.19	0.19	0.19	0.19	0.19	0.70	0.19
Arcade	-	-	-	-	N/A	N/A	N/A	N/A	0.17	N/A
Multi-User	-	-	-	-	-	N/A	N/A	N/A	0.17	N/A
Turn Based Strategy	-	-	-	-	-	-	N/A	N/A	0.15	N/A
Puzzle	-	-	-	-	-	-	-	N/A	0.17	N/A
Shooter	-	-	-	-	-	-	-	-	0.17	N/A
Real Time Strategy	-	-	-	-	-	-	-	-	-	0.17
Card Games	-	-	-	-	-	-	-	-	-	-

Πίνακας 11.6 : Independent sample t-test για το πρότυπο Decorator

	Role Playing Games	Simulator	Board Games	Arcade	Multi-User	Turn Based Strategy	Puzzle	Shooter	Real Time Strategy	Card Games
Role Playing Games	-	0.24	0.82	0.06	0.66	0.26	0.16	0.16	0.51	0.06
Simulator	-	-	0.45	0.34	0.34	0.88	0.87	0.87	0.28	0.34
Board Games	-	-	-	0.32	0.86	0.47	0.41	0.41	0.71	0.32
Arcade	-	-	-	-	0.24	0.17	0.22	0.22	0.20	N/A
Multi-User	-	-	-	-	-	0.36	0.32	0.31	0.83	0.24
Turn Based Strategy	-	-	-	-	-	-	0.70	0.70	0.30	0.17
Puzzle	-	-	-	-	-	-	-	1.00	0.26	0.22
Shooter	-	-	-	-	-	-	-	-	0.26	0.22
Real Time Strategy	-	-	-	-	-	-	-	-	-	0.20
Card Games	-	-	-	-	-	-	-	-	-	-

Πίνακας 11.7 : Independent sample t-test για το πρότυπο Proxy

	Role Playing Games	Simulator	Board Games	Arcade	Multi-User	Turn Based Strategy	Puzzle	Shooter	Real Time Strategy	Card Games
Role Playing Games	-	0.26	0.56	0.16	0.26	0.49	0.16	0.29	0.95	0.22
Simulator	-	-	0.32	0.34	1.00	0.27	0.34	0.79	0.23	0.79
Board Games	-	-	-	0.11	0.29	0.87	0.11	0.38	0.50	0.21
Arcade	-	-	-	-	0.19	0.04	N/A	0.10	0.14	0.34
Multi-User	-	-	-	-	-	0.22	0.19	0.75	0.22	0.74
Turn Based Strategy	-	-	-	-	-	-	0.04	0.32	0.43	0.14
Puzzle	-	-	-	-	-	-	-	0.10	0.14	0.34
Shooter	-	-	-	-	-	-	-	-	0.26	0.51
Real Time Strategy	-	-	-	-	-	-	-	-	-	0.19
Card Games	-	-	-	-	-	-	-	-	-	-

Πίνακας 11.8 : Independent sample t-test για το πρότυπο Observer

	Role Playing Games	Simulator	Board Games	Arcade	Multi-User	Turn Based Strategy	Puzzle	Shooter	Real Time Strategy	Card Games
Role Playing Games	-	0.05	1.00	0.11	0.45	0.21	0.11	0.67	0.53	0.11
Simulator	-	-	0.11	0.34	0.13	0.03	0.34	0.29	0.04	0.34
Board Games	-	-	-	0.19	0.46	0.23	0.19	0.68	0.60	0.19
Arcade	-	-	-	-	0.17	0.04	1.00	0.33	0.14	1,00
Multi-User	-	-	-	-	-	0.83	0.17	0.86	0.31	0.17
Turn Based Strategy	-	-	-	-	-	-	0.04	0.71	0.11	0.04
Puzzle	-	-	-	-	-	-	-	0.33	0.14	1.00
Shooter	-	-	-	-	-	-	-	-	0.52	0.33
Real Time Strategy	-	-	-	-	-	-	-	-	-	0.14
Card Games	-	-	-	-	-	-	-	-	-	-

Πίνακας 11.9 : Independent sample t-test για το πρότυπο State

	Role Playing Games	Simulator	Board Games	Arcade	Multi-User	Turn Based Strategy	Puzzle	Shooter	Real Time Strategy	Card Games
Role Playing Games	-	0.45	0.41	0.04	0.62	0.35	0.06	0.74	0.30	0.28
Simulator	-	-	0.30	0.46	0.40	0.20	0.56	0.72	0.22	1.00
Board Games	-	-	-	0.22	0.64	0.76	0.22	0.36	0.83	0.29
Arcade	-	-	-	-	0.22	0.09	0.69	0.26	0.16	0.16
Multi-User	-	-	-	-	-	0.79	0.25	0.53	0.48	0.37
Turn Based Strategy	-	-	-	-	-	-	0.10	0.29	0.58	0.17
Puzzle	-	-	-	-	-	-	-	0.31	0.17	0.30
Shooter	-	-	-	-	-	-	-	-	0.27	0.65
Real Time Strategy	-	-	-	-	-	-	-	-	-	0.21
Card Games	-	-	-	-	-	-	-	-	-	-

Πίνακας 11.10 : Independent sample t-test για το πρότυπο Template

	Role Playing Games	Simulator	Board Games	Arcade	Multi-User	Turn Based Strategy	Puzzle	Shooter	Real Time Strategy	Card Games
Role Playing Games	-	0.32	0.30	0.20	0.37	0.67	0.38	0.24	0.86	0.82
Simulator	-	-	0.10	0.76	0.10	0.13	0.79	0.89	0.32	0.38
Board Games	-	-	-	0.07	0.83	0.46	0.11	0.08	0.42	0.22
Arcade	-	-	-	-	0.07	0.07	0.45	0.82	0.24	0.22
Multi-User	-	-	-	-	-	0.57	0.11	0.08	0.52	0.26
Turn Based Strategy	-	-	-	-	-	-	0.15	0.08	0.86	0.48
Puzzle	-	-	-	-	-	-	-	0.60	0.37	0.45
Shooter	-	-	-	-	-	-	-	-	0.28	0.27
Real Time Strategy	-	-	-	-	-	-	-	-	-	0.71
Card Games	-	-	-	-	-	-	-	-	-	-

Πίνακας 11.11 : Independent sample t-test για το πρότυπο Visitor

	Role Playing Games	Simulator	Board Games	Arcade	Multi-User	Turn Based Strategy	Puzzle	Shooter	Real Time Strategy	Card Games
Role Playing Games	-	0.35	N/A	N/A	0.34	N/A	N/A	N/A	0.34	N/A
Simulator	-	-	N/A	N/A	0.34	N/A	N/A	N/A	0.34	N/A
Board Games	-	-	-	N/A	0.34	N/A	N/A	N/A	0.34	N/A
Arcade	-	-	-	-	0.34	N/A	N/A	N/A	0.34	N/A
Multi-User	-	-	-	-	-	0.34	0.34	0.34	0.36	0.34
Turn Based Strategy	-	-	-	-	-	-	N/A	N/A	0.34	N/A
Puzzle	-	-	-	-	-	-	-	N/A	0.34	N/A
Shooter	-	-	-	-	-	-	-	-	0.34	N/A
Real Time Strategy	-	-	-	-	-	-	-	-	-	0.34
Card Games	-	-	-	-	-	-	-	-	-	-

Στους πίνακες 12.1 – 12.10 απεικονίζεται για κάθε κατηγορία παιχνιδιού το αν υπάρχει κάποιο πρότυπο σχεδίασης το οποίο χρησιμοποιείται στατιστικά σημαντικά περισσότερο από ότι κάποιο άλλο πρότυπο. Στο σημείο αυτό πρέπει να σημειωθεί ότι λόγω του μικρού δείγματος της ερευνάς μας είμαστε λίγο ελαστικοί ως προς τα κριτήρια αποδοχής του ότι οι Μέσοι Όροι δυο μεταβλητών διαφέρουν στατιστικά σημαντικά.

Πίνακας 12.1 : Paired sample t-test για την κατηγορία Role Playing Games

	Factory	Prototype	Singleton	Adapter	Composite	Decorator	Proxy	Observer	State	Template	Visitor
Factory	-	0.48	0.02	0.01	0.59	0.11	0,27	0.66	0.01	0.15	0.22
Prototype	-	-	0.02	0.03	0.11	0.33	0.50	0.71	0.02	0.42	0.05
Singleton	-	-	-	0.33	0.02	0.04	0.04	0.02	0.52	0.03	0.02
Adapter	-	-	-	-	0.02	0.03	0.03	0.03	0.33	0.03	0.02
Composite	-	-	-	-	-	0.08	0.19	0.17	0.01	0.15	0.17
Decorator	-	-	-	-	-	-	1.00	0.24	0.02	0.85	0.06
Proxy	-	-	-	-	-	-	-	0.32	0.02	0.86	0.16
Observer	-	-	-	-	-	-	-	-	0.02	0.22	0.05
State	-	-	-	-	-	-	-	-	-	0.02	0.01
Template	-	-	-	-	-	-	-	-	-	-	0.11
Visitor	-	-	-	-	-	-	-	-	-	-	-

Πίνακας 12.2 : Paired sample t-test για την κατηγορία Simulation Games

	Factory	Prototype	Singleton	Adapter	Composite	Decorator	Proxy	Observer	State	Template	Visitor
Factory	-	0.68	0.07	0.30	N/A	0.34	0.34	0.34	0.25	0.34	0.34
Prototype	-	-	0.07	0.31	0.68	0.60	0.80	0.34	0.26	0.56	0.34
Singleton	-	-	-	0.33	0.07	0.09	0.07	0.07	0.39	0.11	0.07
Adapter	-	-	-	-	0.30	0.30	0.30	0.30	0.32	0.30	0.30
Composite	-	-	-	-	-	0.34	0.34	0.34	0.25	0.34	0.34
Decorator	-	-	-	-	-	-	0.34	0.34	0.24	0.34	0.34
Proxy	-	-	-	-	-	-	-	0.34	0.24	0.34	0.34
Observer	-	-	-	-	-	-	-	-	0.24	0.34	N/A
State	-	-	-	-	-	-	-	-	-	0.23	0.25
Template	-	-	-	-	-	-	-	-	-	-	0.34
Visitor	-	-	-	-	-	-	-	-	-	-	-

Πίνακας 12.3 : Paired sample t-test για την κατηγορία Board Games

	Factory	Prototype	Singleton	Adapter	Composite	Decorator	Proxy	Observer	State	Template	Visitor
Factory	-	0.46	0.09	0.19	0.25	0.64	0.21	0.35	0.19	0.03	0.12
Prototype	-	-	0.05	0.14	0.37	0.35	0.40	0.40	0.16	0.97	0.33
Singleton	-	-	-	0.38	0.10	0.06	0.09	0.10	0.31	0.13	0.09
Adapter	-	-	-	-	0.18	0.16	0.18	0.19	0.23	0.22	0.18
Composite	-	-	-	-	-	0.40	0.31	0.47	0.19	0.07	0.19
Decorator	-	-	-	-	-	-	0.46	0.47	0.17	0.16	0.32

	Factory	Prototype	Singleton	Adapter	Composite	Decorator	Proxy	Observer	State	Template	Visitor
Proxy	-	-	-	-	-	-	-	0.60	0.19	0.04	0.11
Observer	-	-	-	-	-	-	-	-	0.19	0.08	0.11
State	-	-	-	-	-	-	-	-	-	0.22	0.19
Template	-	-	-	-	-	-	-	-	-	-	0.05
Visitor	-	-	-	-	-	-	-	-	-	-	-

Πίνακας 12.4 : Paired sample t-test για την κατηγορία Arcade Games

	Factory	Prototype	Singleton	Adapter	Composite	Decorator	Proxy	Observer	State	Template	Visitor
Factory	-	0.46	0.09	0.19	0.25	0.64	0.21	0.35	0.19	0.03	0.12
Prototype	-	-	0.05	0.14	0.37	0.35	0.40	0.40	0.16	0.97	0.33
Singleton	-	-	-	0.38	0.10	0.06	0.09	0.10	0.31	0.13	0.09
Adapter	-	-	-	-	0.18	0.16	0.18	0.19	0.23	0.22	0.18
Composite	-	-	-	-	-	0.40	0.31	0.47	0.19	0.07	0.19
Decorator	-	-	-	-	-	-	0.46	0.47	0.17	0.16	0.32
Proxy	-	-	-	-	-	-	-	0.60	0.19	0.04	0.11
Observer	-	-	-	-	-	-	-	-	0.19	0.08	0.11
State	-	-	-	-	-	-	-	-	-	0.22	0.19
Template	-	-	-	-	-	-	-	-	-	-	0.05
Visitor	-	-	-	-	-	-	-	-	-	-	-

Πίνακας 12.5 : Paired sample t-test για την κατηγορία Multi-User Games

	Factory	Prototype	Singleton	Adapter	Composite	Decorator	Proxy	Observer	State	Template	Visitor
Factory	-	0.20	0.33	0.19	0.20	0.28	0.21	0.80	0.17	0.08	0.39
Prototype	-	-	0.30	0.19	N/A	0.24	0.19	0.13	0.74	0.04	0.34
Singleton	-	-	-	0.13	0.30	0.37	0.31	0.36	0.12	0.73	0.33
Adapter	-	-	-	-	0.19	0.18	0.19	0.19	0.22	0.31	0.20
Composite	-	-	-	-	-	0.24	0.19	0.13	0.17	0.04	0.34
Decorator	-	-	-	-	-	-	0.25	0.37	0.16	0.66	0.31
Proxy	-	-	-	-	-	-	-	0.16	0.17	0.05	1.00
Observer	-	-	-	-	-	-	-	-	0.18	0.11	0.31
State	-	-	-	-	-	-	-	-	-	0.24	0.18
Template	-	-	-	-	-	-	-	-	-	-	0.04
Visitor	-	-	-	-	-	-	-	-	-	-	-

Πίνακας 12.6 : Paired sample t-test για την κατηγορία Turn Based Strategy Games

	Factory	Prototype	Singleton	Adapter	Composite	Decorator	Proxy	Observer	State	Template	Visitor
Factory	-	0.85	0.05	0.11	0.07	0.17	0.69	0.46	0.06	0.18	0.07
Prototype	-	-	0.05	0.11	0.22	0.49	0.68	0.81	0.06	0.36	0.22
Singleton	-	-	-	0.19	0.04	0.04	0.04	0.04	025	0.05	0.04
Adapter	-	-	-	-	0.10	0.10	0.10	0.11	0.31	0.11	0.10
Composite	-	-	-	-	-	0.17	0.04	0.03	0.06	0.03	N/A
Decorator	-	-	-	-	-	-	0.66	0.05	0.06	0.07	0.17
Proxy	-	-	-	-	-	-	-	0.19	0.06	0.06	0.04
Observer	-	-	-	-	-	-	-	-	0.06	0.17	0.03
State	-	-	-	-	-	-	-	-	-	0.07	0.06
Template	-	-	-	-	-	-	-	-	-	-	0.03
Visitor	-	-	-	-	-	-	-	-	-	-	-

Πίνακας 12.7 : Paired sample t-test για την κατηγορία Puzzle Games

	Factory	Prototype	Singleton	Adapter	Composite	Decorator	Proxy	Observer	State	Template	Visitor
Factory	-	0.36	0.08	0.06	0.19	0.78	0.19	0.44	0.08	0.30	0.19
Prototype	-	-	1.00	0.27	0.25	0.27	0.25	0.28	0.57	0.42	0.25
Singleton	-	-	-	0.20	0.02	0.04	0.02	0.03	0.48	0.14	0.02
Adapter	-	-	-	-	0.07	0.09	0.07	0.07	0.18	0.11	0.07
Composite	-	-	-	-	-	0.22	N/A	0.34	0.08	0.10	N/A
Decorator	-	-	-	-	-	-	0.22	0.39	0.13	0.10	0.22
Proxy	-	-	-	-	-	-	-	0.34	0.08	0.10	N/A
Observer	-	-	-	-	-	-	-	-	0.09	0.13	0.34
State	-	-	-	-	-	-	-	-	-	0.18	0.08
Template	-	-	-	-	-	-	-	-	-	-	0.10
Visitor	-	-	-	-	-	-	-	-	-	-	-

Πίνακας 12.8 : Paired sample t-test για την κατηγορία First Person Shooter Games

	Factory	Prototype	Singleton	Adapter	Composite	Decorator	Proxy	Observer	State	Template	Visitor
Factory	-	0.24	0.26	0.14	0.24	0.84	0.78	0.64	0.14	1.00	0.24
Prototype	-	-	0.03	0.12	N/A	0.22	0.10	0.29	0.13	0.18	N/A
Singleton	-	-	-	0.19	0.03	0.15	0.13	0.91	0.17	0.30	0.03
Adapter	-	-	-	-	0.12	0.11	0.12	0.18	0.24	0.12	0.12
Composite	-	-	-	-	-	0.22	0.10	0.29	0.13	0.18	N/A
Decorator	-	-	-	-	-	-	1.00	0.50	0.13	0.68	0.22
Proxy	-	-	-	-	-	-	-	0.54	0.13	0.73	0.10
Observer	-	-	-	-	-	-	-	-	0.18	0.62	0.29
State	-	-	-	-	-	-	-	-	-	0.13	0.13
Template	-	-	-	-	-	-	-	-	-	-	0.18
Visitor	-	-	-	-	-	-	-	-	-	-	-

Πίνακας 12.9 : Paired sample t-test για την κατηγορία Real Time Strategy Games

	Factory	Prototype	Singleton	Adapter	Composite	Decorator	Proxy	Observer	State	Template	Visitor
Factory	-	0.13	0.12	0.08	0.12	0.35	0.75	0.18	0.14	1.00	0.43
Prototype	-	-	0.06	0.07	0.17	0.20	0.14	0.04	0.14	0.15	0.34
Singleton	-	-	-	0.12	0.06	0.23	0.05	0.07	0.23	0.12	0.80
Adapter	-	-	-	-	0.07	0.10	0.06	0.07	041	0.08	0.29
Composite	-	-	-	-	-	0.21	0.16	0.17	0.13	0.16	0.35
Decorator	-	-	-	-	-	-	0.50	0.24	0.13	0.31	0.46
Proxy	-	-	-	-	-	-	-	0.21	0.14	0.77	0.48
Observer	-	-	-	-	-	-	-	-	0.14	0.21	0.37
State	-	-	-	-	-	-	-	-	-	0.14	0.14
Template	-	-	-	-	-	-	-	-	-	-	0.43
Visitor	-	-	-	-	-	-	-	-	-	-	-

Πίνακας 12.10 : Paired sample t-test για την κατηγορία Card Games

	Factory	Prototype	Singleton	Adapter	Composite	Decorator	Proxy	Observer	State	Template	Visitor
Factory	-	0.50	0.04	0.01	0.34	0.34	0.68	1.00	0.03	0.09	0.34
Prototype	-	-	0.05	0.01	0.34	0.34	0.68	0.34	0.04	0.08	034
Singleton	-	-	-	0.01	0.03	0.03	0.07	0.03	0.50	0.69	0.03
Adapter	-	-	-	-	0.01	0.01	0.01	0.01	0.33	0.04	0.01
Composite	-	-	-	-	-	N/A	0.34	0.34	0.03	0.09	N/A
Decorator	-	-	-	-	-	-	0.34	0.34	0.03	0.09	N/A
Proxy	-	-	-	-	-	-	-	0.68	0.04	0.16	0.34
Observer	-	-	-	-	-	-	-	-	0.03	0.08	0.34
State	-	-	-	-	-	-	-	-	-	0.09	0.03
Template	-	-	-	-	-	-	-	-	-	-	0.09
Visitor	-	-	-	-	-	-	-	-	-	-	-

Στον Πίνακα 13, παρουσιάζουμε μερικά γενικά στατιστικά στοιχεία για κάθε κατηγορία παιχνιδιών που μελετάμε. Οι αριθμητικές μεταβλητές αντιπροσωπεύονται από την μέση τιμή τους, ενώ οι κατηγορικές από την επικρατέστερη τιμή τους μέσα στο δείγμα. Μετά τον έλεγχο εγκυρότητας μέσω 225 Independent Sample t -Tests οι μεταβλητές που φαίνεται να διαφέρουν στατιστικά παρουσιάζονται στον πίνακα 14.

Πίνακας 13: Γενικά στατιστικά στοιχεία για τις κατηγορίες των παιχνιδιών

	Release Year	Days Active	Downloads	Developers	Versions	Classes
Role Playing	2009	450,50	21603,20	4,50	6,50	243,10
Simulation	2009/2010	1072,60	8090,00	1,50	5,00	118,50
Board Game	2006/2008/2009	688,30	14563,80	2,80	9,20	201,50
Arcade	2007/2008/2009	82,90	1314,60	1,70	3,10	39,70
Multi -User	2003	337,00	1912,70	1,00	3,40	229,10
Turn Based	2010	1526,70	108930,00	8,30	8,90	295,60
Puzzle	2009	163,70	5116,30	1,10	2,60	59,30
Shooter	2005/2008/2009	149,20	1453,00	2,30	3,20	100,70
Real Time Strategy	2005	512,70	31063,40	3,40	8,00	222,90
Card Game	2008	483,00	3308,20	1,00	5,90	124,00

Πίνακας 14: Μεταβλητές που διαφέρουν στατιστικά σημαντικά

	Days Active	Downloads	Developers	Versions	Classes
Role Playing – Arcade	0,05	-	-	-	0,00
Role Playing –Turn Based	0,03	-	-	-	-
Role Playing – Puzzle	-	-	-	-	0,01
Simulation – Arcade	0,01	0,02	-	-	-
Simulation – Multi -User	0,04	0,04	0,05	-	-
Simulation – Puzzle	0,01	-	-	-	-
Simulation – Shooter	0,01	0,02	-	-	**-**
Simulation – Card Game	-	-	0,05	-	-
Board Games – Arcade	0,03	-	-	-	-
Board Games – Puzzle	0,05	-	-	-	-
Board Games –Shooter	0,05	-	-	-	-
Arcade – Turn Based	0,01	-	-	-	0,04
Arcade – Real Time Strategy	0,03	-	-	**-**	0,05
Arcade –Card Game	-	-	-	-	0,04
Multi -User – Turn Based	0,02	-	-	-	-
Turn Based – Puzzle	0,01	-	-	0,05	0,05
Turn Based – Shooter	0,01	-	-	-	-
Turn Based – Real Time Strategy	0,04	-	-	-	-
Turn Based – Card Game	0,04	-	-	-	-

Ο Πίνακας 15 δείχνει τις τιμές της σημαντικότητας για τις συσχετίσεις μεταξύ του αριθμού των προτύπων που εντοπίσθηκαν σε κάθε μία από τις τρεις κατηγορίες προτύπων, δηλαδή, creational, behavioural και structural, και όλες τις μεταβλητές που δεν σχετίζονται με τα πρότυπα , δηλαδή, την κατηγορία παιχνιδιού, το έτος έκδοσης, τον αριθμό των ημερών που η εφαρμογή ήταν ενεργή, τον αριθμό των λήψεων, τον αριθμό των εκδόσεων και το μέγεθος της εφαρμογής μετρημένο με βάση τον αριθμό των κλάσεων. Ο έλεγχος που κάναμε για να εντοπίσουμε συσχετίσεις μεταξύ κάθε τέτοιου ζεύγους μεταβλητών ήταν το Pearson chi-square test.

Πίνακας 15: Συσχετίσεις από το Person x^2-test

	Creational	**Structural**	**Behavioural**
Category	0,20	0,55	0,65
Release Year	0,77	0,89	0,16
Days Active	0,00	0,00	0,01
Downloads	0,00	0,00	0,03
Developers	0,00	0,00	0,00
Versions	0,00	0,00	0,00
Classes	0,00	0,00	0,00

Προκειμένου να ερευνήσουμε και να δημιουργήσουμε προφίλ για τη χρήση των προτύπων σχεδίασης, δημιουργήσαμε clusters with με τη βοήθεια του αλγορίθμου K-Means. Ο αλγόριθμος δημιούργησε 14 clusters τα οποία φαίνονται στον Πίνακα 16. Τα δεδομένα του πίνακα που αντιστοιχούν σε κατηγορικές μεταβλητές αναφέρονται στο Παράρτημα Δ.

Πίνακας 16: Προφίλ Λογισμικού για τη χρήση προτύπων

Cluster No	**[1]**	**[2]**	**[3]**	**[4]**	**[5]**	**[6]**	**[7]**	**[8]**	**[9]**	**[10]**	**[11]**	**[12]**	**[13]**	**[14]**
Release year	3	1	1	3	2	1	1	3	3	3	1	1	2	1
Days Active	4	1	1	4	4	1	2	4	3	4	2	4	1	1
Downloads	1	1	4	4	4	1	1	4	2	3	1	4	1	3
Developers	1	0	0	1	1	0	0	0	0	0	0	0	0	0
Versions	2	2	2	3	3	2	1	3	2	1	2	2	2	2
Classes	4	3	4	2	4	1	2	4	3	2	4	4	1	3
Factory	1	2	2	1	3	1	1	1	1	2	4	2	1	1

Cluster No	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]
Prototype	1	1	1	1	5	1	1	1	4	1	1	1	1	2
Singleton	5	2	4	1	5	4	1	1	3	2	3	5	1	3
Creational	5	3	5	1	5	4	1	1	5	3	5	5	1	3
Adapter	3	4	5	1	5	1	1	4	2	3	5	5	1	3
Composite	1	1	1	1	1	1	1	1	1	1	2	1	1	1
Decorator	1	1	3	1	5	1	1	1	2	1	5	2	1	1
Proxy	1	1	3	1	3	1	1	1	1	2	2	3	1	1
Structural	3	4	5	1	5	1	1	4	3	3	5	5	1	3
Observer	1	2	1	1	2	1	1	2	1	1	2	2	1	2
State	3	4	5	1	5	1	4	2	2	5	5	4	1	3
Template	1	1	3	1	5	1	4	1	2	1	4	2	1	3
Visitor	1	1	1	1	1	1	2	1	1	1	5	1	1	1
Behavioural	3	4	5	1	5	1	5	2	3	5	5	4	1	5
Category	0	0	0	1	2	3	4	5	6	7	8	8	9	9

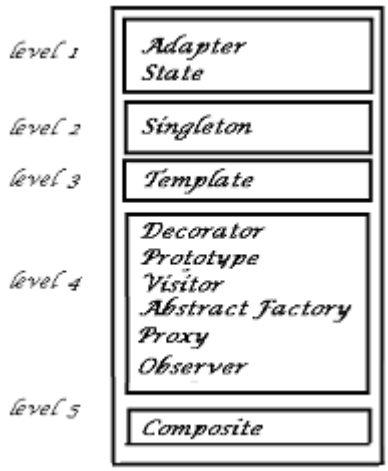
3.3 ΣΥΖΗΤΗΣΗ

Η ενότητα αυτή συζητά τα αποτελέσματα των στατιστικών τεχνικών και του clustering που εφαρμόστηκαν στο σύνολο των δεδομένων που συγκεντρώσαμε για την έρευνά μας. Η συζήτηση έχει οργανωθεί σε υποενότητες σύμφωνα με τα ερωτήματα της έρευνας, που αναφέρονται στην αρχή του άρθρου. Έτσι, το κεφάλαιο 5.1. discusses ποια πρότυπα σχεδίασης χρησιμοποιούνται συχνότερα στην ανάπτυξη παιχνιδιών ανοιχτού λογισμικού, το κεφάλαιο 5.2 συζητά σχετικά με τη χρήση κάθε προτύπου σχεδίασης που ανήκει σε μία από τις 10 κατηγορίες παιχνιδιών που μελετάμε και το κεφάλαιο 5.3 discusses για τους κρίσιμους παράγοντες που επηρεάζουν τη χρήση προτύπων στις εφαρμογές ανοιχτού λογισμικού.

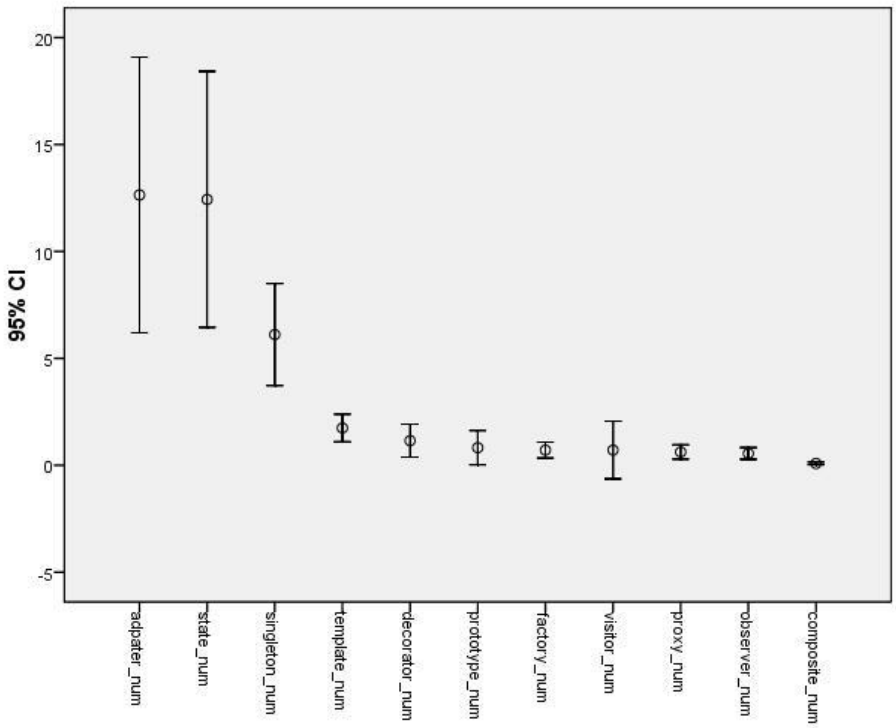
3.3.1 Εφαρμογή προτύπων σχεδίασης

Τα αποτελέσματα του Πίνακα 8, μας δείχνουν με την πρώτη ματιά ότι κάποια πρότυπα εφαρμόζονται περισσότερο από άλλα στα παιχνίδια ανοιχτού λογισμικού. Επιπλέον, στον Πίνακα 9 φαίνεται ότι η συχνότητα χρήσης των προτύπων τα κατηγοριοποιεί σε 5 κατηγορίες όπως φαίνεται στο Σχήμα 12. Το Σχήμα 13 αναπαριστά τις error bars των έντεκα προτύπων που μελετάμε.

Τα πρότυπα στην κορυφή του Σχήματος 12 χρησιμοποιούνται στατιστικά σημαντικά περισσότερες φορές σε εφαρμογές ανοιχτού λογισμικού απ' ότι αυτά που βρίσκονται χαμηλότερα στην κατάταξη.



Σχήμα 12: Επίπεδα χρήσης των προτύπων σχεδίασης



Σχήμα 13: Error Bars για τη χρήση προτύπων σχεδίασης

Κάποια από τα αποτελέσματα που παρουσιάζονται στα σχήματα 12 και 13 είναι λογικά και αναμενόμενα, ενώ κάποια άλλα προκαλούν έκπληξη. Κατ' αρχήν

ήταν αναμενόμενο το πρότυπο *Adapter* να χρησιμοποιείται συχνά, αφού οι κοινωνίες ανοιχτού λογισμικού βασίζονται ως λογική στην επαναχρησιμοποίηση κλάσεων άλλων προγραμματιστών. Σε τέτοιες περιπτώσεις, το πρότυπο adapter παρέχει έναν μηχανισμό αφομοίωσης της νέας κλάσης μέσα στο υπάρχον σύστημα χωρίς τροποποίηση του υπάρχοντος κώδικα. Επίσης, η λογική του Adapter ταιριάζει με τις βασικές αρχές του αντικειμενοστραφούς προγραμματισμού και έτσι πολλές φορές μπορεί να χρησιμοποιείται χωρίς πρόθεση από τους προγραμματιστές.

Για το πρότυπο *State* περιμέναμε επίσης μια υψηλή θέση στην κατάταξη, επειδή το μόνο υπόβαθρο που χρειάζεται για τη χρήση του, είναι η σωστή χρήση κληρονομικότητας. Αντίθετα, ενώ η δομή του προτύπου *Singleton* είναι αρκετά περίπλοκη, (Chatzigeorgiou, 2005) και θα περιμέναμε να μην είναι ένα δημοφιλές πρότυπο, κατατάσσεται στην $3^{η}$ θέση της λίστας των πιο συχνά χρησιμοποιούμενων προτύπων. Πιθανότατα, αυτό οφείλετε στο περιορισμένο πεδίο της έρευνας μας που ασχολείται αποκλειστικά με παιχνίδια ανοιχτού λογισμικού γραμμένα σε γλώσσα Java, όπου το πρότυπο Singleton εφαρμόζεται απλούστατα μέσω ενός μηχανισμού δημιουργίας στιγμιοτύπων.

Μια ακόμα παράξενη παρατήρηση, που προέκυψε από τα αποτελέσματά μας, είναι ότι το πρότυπο *Decorator* χρησιμοποιείται πιο συχνά απ’ ότι το πρότυπο *Composite*. Το πρότυπο Composite αποτελεί τα θεμέλια του Decorator και γι’ αυτό θα περιμέναμε να χρησιμοποιείται πιο συχνά. Τελικά, το πιο δυσνόητο πρότυπο φαίνεται να είναι το πρότυπο *Observer*, το οποίο δε χρησιμοποιείται συχνά από τους προγραμματιστές ανοιχτού λογισμικού.

Συνοψίζοντας τα παραπάνω, οι προγραμματιστές ανοιχτού λογισμικού, προτιμούν να χρησιμοποιούν τα πιο ευκολονόητα πρότυπα απ’ ότι εκείνα που δίνουν πιο ασφαλή αποτελέσματα. Πιθανότατα, ένας λόγος που συμβαίνει αυτό να είναι ότι γενικά οι προγραμματιστές δεν ξεκινούν από μια φάση αναλυτικής σχεδίασης προτού ξεκινήσουν τη συγγραφή κώδικα.

3.3.2 Πρότυπα Σχεδίασης και Κατηγορίες Παιχνιδιών

Όπως παρατηρούμε στον Πίνακα 10, η χρήση των προτύπων σχεδίασης σε κάθε κατηγορία ακολουθεί παρόμοια κατανομή με τη χρήση τους γενικότερα στην ανάπτυξη ανοιχτού λογισμικού. Παρόλα αυτά, συγκρίνοντας την εφαρμογή των προτύπων στις διάφορες κατηγορίες παιχνιδιών, βλέπουμε ότι σύμφωνα με τα

αποτελέσματα που προέκυψαν από την έρευνά μας, κάποια πρότυπα εφαρμόζονται συχνότερα σε κάποιες συγκεκριμένες κατηγορίες.

Στον πίνακα 11.2 παρατηρούμε ότι η κατηγορία Role Playing Games εμφανίζει περισσότερα στατιστικά σημαντικά στιγμιότυπα του προτύπου Prototype από ότι οι κατηγορίες Multi-User, Shooter και Real-Time Strategy. Επιπλέον, παρατηρούμε ότι η κατηγορία Board Games που φαίνεται να έχει τον μέγιστο μέσο όρο στιγμιοτύπων δε διαφέρει στατιστικά σημαντικά από καμία άλλη, πιθανότατα γιατί ο μέσος όρος της επηρεάζεται από κάποια ακραία τιμή. Στον πίνακα 11.3 οι κατηγορίες Role Playing Games και Turn Based Strategy έχουν τον μεγαλύτερο μέσο όρο στιγμιοτύπων για το πρότυπο Singleton. Η κατηγορία Role Playing Games διαφέρει στατιστικά σημαντικά από 5 κατηγορίες, ενώ η Turn Based Strategy, που έχει σχεδόν ίση τιμή με τα Role Playing Games διαφέρει στατιστικά σημαντικά μόνο από την κατηγορία Shooter, ενώ με άλλες τέσσερεις κατηγορίες διαφέρει οριακά και η διαφορά δε λογίζεται ως στατιστικά σημαντική.

Στον πίνακα 11.7 παρατηρούμε ότι η κατηγορία Turn Based Strategy εμφανίζει περισσότερα στατιστικά σημαντικά στιγμιότυπα του προτύπου Proxy και διαφέρει στατιστικά σημαντικά από 2 κατηγορίες, τις κατηγορίες Puzzle και Arcade. Στον πίνακα 11.8 φαίνεται ότι το πρότυπο Observer χρησιμοποιείται περισσότερο στην κατηγορία Turn Based Strategy, η οποία διαφέρει στατιστικά σημαντικά από 3εις κατηγορίες. Η εκτεταμένη χρήση του προτύπου στην κατηγορία Multi-user δεν επιβεβαιώνεται στατιστικά από τα t-tests.

Από τους πίνακες 11.1, 11.4, 11.5, 11.6, 11.9, 11.10 και 11.11 προκύπτει ότι τα πρότυπα Factory, Adapter, Composite, Decorator, State, Template και Visitor αντίστοιχα χρησιμοποιούνται εξίσου σε όλες τις κατηγορίες παιχνιδιών.

Σύμφωνα με τα αποτελέσματα του πίνακα 12.1 που απευθύνεται στην κατηγορία παιχνιδιών Role Playing Games μπορούμε να χωρίσουμε τα πρότυπα σε 3 επίπεδα σύμφωνα με το πόσο χρησιμοποιούνται κατά ανάπτυξη τέτοιων παιχνιδιών. Στο πρώτο επίπεδο, τοποθετούνται τα πιο συχνά χρησιμοποιούμενα πρότυπα, που από τον πίνακα φαίνεται να διαφέρουν στατιστικά σημαντικά από όλα τα άλλα πρότυπα αλλά όχι μεταξύ τους. Αυτά είναι τα Singleton, Adapter και State. Στο δεύτερο επίπεδο, τοποθετούμε τα πρότυπα Factory, Prototype, Composite, Decorator, Proxy, Observer και Template, ενώ στο τελευταίο επίπεδο τοποθετούμε το πρότυπο Visitor που παρουσιάζει μηδενικές τιμές. Στον πίνακα

12.2 η κατηγορία Simulation φαίνεται να μην παρουσιάζει στατιστικά σημαντικές διαφορές στη χρήση των προτύπων.

Από τον πίνακα 12.3 δεν μπορούμε να ξεχωρίσουμε επίπεδα χρήσης των προτύπων στην κατηγορία Board Games, αλλά παρατηρούμε ότι το πρότυπο Template, παρότι τέταρτο στην κατάταξη, διαφέρει στατιστικά σημαντικά από τρία αλλά πρότυπα, σε αντίθεση με τα State, Adapter και Singleton που προηγούνται και δεν διαφέρουν στατιστικά σημαντικά από περισσότερα από ένα πρότυπα. Τα ίδια ακριβώς συμπεράσματα προκύπτουν και από τον πίνακα 12.4 που απεικονίζει την χρήση των προτύπων στην κατηγορία Arcade Games, αλλά και από τον πίνακα 12.5 που απεικονίζει την κατάσταση για τα Multi-User Games, με μόνη διαφορά ότι τα πρότυπα State, Adapter και Singleton εδώ δεν διαφέρουν στατιστικά σημαντικά από κανένα πρότυπο.

Ο πίνακας 12.6 απεικονίζει τη χρήση προτύπων στην κατηγορία Turn Based Strategy Games και από τις τιμές του διαχωρίζονται 4 επίπεδα χρήσης προτύπων. Στην πρώτη κατηγορία, τοποθετείται το πρότυπο Singleton με τα State και Adapter να ακολουθούν σε ξεχωριστό επίπεδο. Στο τρίτο επίπεδο, τοποθετούμε τα πρότυπα Factory, Prototype, Decorator, Proxy, Observer και Template, ενώ στο τελευταίο επίπεδο τοποθετούμε τα πρότυπα Visitor και Composite που παρουσιάζουν μηδενικές τιμές. 3 επίπεδα χρήσης προτύπων ξεχωρίζουν και από τον πίνακα 12.7 όσο αφορά την κατηγορία Puzzle Games. Στο πρώτο επίπεδο ξεχωρίζει το πρότυπο Singleton ενώ στο δεύτερο επίπεδο τοποθετείται το πρότυπο Adapter που παρατηρούμε ότι δε διαφέρει στατιστικά σημαντικά στην τάξη του 0,05, αλλά εμφανίζει πολλές στατιστικά σημαντικές διαφορές στο επίπεδο του 0,1. Σε ένα ενιαίο επίπεδο ακολουθούν όλα τα υπόλοιπα πρότυπα χωρίς να διαφέρουν στατιστικά σημαντικά μεταξύ τους.

Από τα αποτελέσματα του πίνακα 12.8 δεν μπορούμε να ξεχωρίσουμε επίπεδα χρήσης των προτύπων στην κατηγορία First Person Shooter Games, αλλά παρατηρούμε ότι το πρότυπο Template, παρότι τέταρτο στην κατάταξη, διαφέρει στατιστικά σημαντικά από τρία αλλά πρότυπα, σε αντίθεση με τα State, Adapter και Singleton που προηγούνται και δεν διαφέρουν στατιστικά σημαντικά από κανένα πρότυπο. Στον πίνακα 12.9 για την κατηγορία Real time Strategy δεν παρατηρούνται στατιστικά σημαντικές διαφορές μεταξύ των προτύπων. Τέλος σύμφωνα με τις τιμές του πίνακα 12.10 για την κατηγορία Card Games προκύπτουν 3 επίπεδα χρήσης προτύπων, με τα State και Adapter να κυριαρχούν

ως τα πιο συχνά χρησιμοποιούμενα πρότυπα, το Singleton στο δεύτερο επίπεδο μόνο του και όλα τα υπόλοιπα πρότυπα σε ένα ενιαίο επίπεδο χωρίς να διαφέρουν στατιστικά σημαντικά μεταξύ τους.

3.3.3 Παράγοντες που επηρεάζουν την εφαρμογή προτύπων σχεδίασης

Σε αυτή την ενότητα συζητάμε τους πιο σημαντικούς παράγοντες που επηρεάζουν την εφαρμογή των προτύπων σχεδίασης. Σύμφωνα με τον Πίνακα 15 , οι ημέρες που ένα παιχνίδι είναι ενεργό, ο αριθμός λήψεων που έχει, ο αριθμός των προγραμματιστών που συνέβαλλαν στη δημιουργία του, ο αριθμός εκδόσεων και το μέγεθός του είναι οι πιο σημαντικοί παράγοντες που επηρεάζουν τη χρήση των προτύπων. Από την άλλη πλευρά, το έτος έκδοσης και η κατηγορία που ανήκει το κάθε παιχνίδι δε φαίνεται να είναι σημαντικά.

Μελετώντας τα δεδομένα του Πίνακα 16 παρατηρούμε ότι για την κατηγορία Role Playing Games, η οποία σύμφωνα με την έρευνά μας είναι η δημοφιλέστερη κατηγορία παιχνιδιών ανοιχτού λογισμικού, υπάρχουν 3 επίπεδα χρήσης προτύπων σχεδίασης, τα οποία περιγράφονται από τα παρακάτω χαρακτηριστικά.

Στο πρώτο επίπεδο ανήκουν παιχνίδια που εκδόθηκαν μετά το 2008 και έμειναν ενεργά για περισσότερα από 2 χρόνια. Τα παιχνίδια του επιπέδου αυτού έχουν μικρό αριθμό λήψεων που δεν ξεπερνάει τις 1.000 λήψεις, έχουν συνολικά 2 με 10 εκδόσεις, μέγεθος μεγαλύτερο από 201 κλάσεις και περισσότερους από έναν προγραμματιστές. Όσον αφορά τα πρότυπα σχεδίασης που εντοπίζονται στα παιχνίδια αυτά, παρατηρούμε ότι δεν βρίσκουμε στιγμιότυπα των προτύπων Factory, Prototype, Composite, Decorator, Proxy, Observer, Template και Visitor ενώ χρησιμοποιείται πολύ το πρότυπο Singleton με περισσότερα από 15 στιγμιότυπα. Σε λίγο μικρότερο βαθμό χρησιμοποιούνται τα πρότυπα Adapter και State με 5 έως 9 στιγμιότυπα.

Στο δεύτερο επίπεδο, ανήκουν παιχνίδια με έτος έκδοσης πριν το 2005. Τα παιχνίδια του δευτέρου επιπέδου χαρακτηρίζονται από μικρό αριθμό ημερών που παρέμειναν ενεργά (λιγότερες από 100 ημέρες) και μικρό αριθμό λήψεων που δεν υπερβαίνει τις 1.000. Επίσης έχουν δημιουργηθεί από έναν μόνο προγραμματιστή, έχουν 2 με 10 εκδόσεις και μέγεθος 101 με 200 κλάσεις. Μέγιστο αριθμό στιγμιοτύπων παρατηρούμε για τα πρότυπα Adapter και State (10-15 στιγμιότυπα) ενώ ακολουθούν τα Factory, Singleton και Observer με 1 έως 4 στιγμιότυπα.

Στιγμιότυπα για τα πρότυπα Prototype, Composite, Decorator, Proxy, Template, Visitor δεν υπάρχουν.

Στο τρίτο και τελευταίο επίπεδο χρήσης προτύπων, για την κατηγορία Role Playing Games, ανήκουν παιχνίδια που κατασκευάστηκαν πριν το 2005 και χαρακτηρίζονται από μεγάλο αριθμό λήψεων που ξεπερνούν τις 10.000 και έχουν μεγάλο μέγεθος της τάξης των 200 κλάσεων και άνω. Άλλα χαρακτηριστικά των παιχνιδιών αυτής της κατηγορίας είναι ότι έχουν κατασκευαστεί από ένα μόνο προγραμματιστή, ήταν ενεργά για λιγότερες από 100 ημέρες και έχουν 2 έως 10 εκδόσεις. Όσον αφορά τα πρότυπα σχεδίασης, στην κατηγορία αυτή παρατηρούμε έντονη χρήση και μεγάλη ποικιλία προτύπων. Ξεκινάμε από τα πρότυπα Adapter και State που βρίσκονται στην κορυφή με περισσότερα από 15 στιγμιότυπα και ακολουθεί το πρότυπο Singleton με αριθμό στιγμιοτύπων που κυμαίνεται από 10 μέχρι 14 στιγμιότυπα. Στην επόμενη θέση βρίσκονται τα πρότυπα Decorator, Proxy και Template για τα οποία σημειώνονται 5 με 9 στιγμιότυπα ενώ για το πρότυπο Factory μετράμε 1 με 4 στιγμιότυπα. Τέλος, δεν εμφανίζεται κανένα στιγμιότυπο για τα πρότυπα Visitor, Observer, Composite και Prototype.

Από τα δεδομένα του Πίνακα 16, τα clusters που μας παρέχουν περισσότερες πληροφορίες είναι τα cluster [4], [5], [7], [8], [11] και [15], τα οποία αποτελούν προφίλ των κατηγοριών Simulator, Board Games, Multi-User, Turn Based Strategy, Real Time Strategy και Card Games. Αρχικά παρατηρούμε ότι για τη δημιουργία παιχνιδιών Simulator [4] και Card Games [13], στη συνηθέστερη περίπτωση, δεν χρησιμοποιούνται πρότυπα σχεδίασης. Αντίθετα, στην κατηγορία Board Games [5] συναντάμε τη χρήση των πολλών στιγμιοτύπων προτύπων σχεδίασης. Επιπλέον, στην κατηγορία Real Time Strategy [11] παρατηρούμε ότι έχουμε τουλάχιστον ένα στιγμιότυπο από όλα τα πρότυπα σχεδίασης που μελετάμε. Επιπρόσθετα, για την κατηγορία Multi-User [7] βλέπουμε έντονη χρήση προτύπων συμπεριφοράς, δηλαδή συναντάμε στιγμιότυπα των προτύπων State, Template και Visitor που αθροιστικά ξεπερνούν τα 15 στιγμιότυπα, αλλά κανένα στιγμιότυπο προτύπου άλλης κατηγορίας. Τέλος, για την κατηγορία Turn Based Strategy [8] παρατηρούμε ότι κατά κύριο χρησιμοποιείται το πρότυπο Adapter και σε μικρότερο βαθμό δυο behavioral πρότυπα (Observer και State). Τα χαρακτηριστικά των παραπάνω clusters παρουσιάζονται στον Πίνακα 17.

Πίνακας 17: Κέντρα σημαντικότερων Cluster.

Cluster No	**[4]**	**[5]**	**[7]**	**[8]**	**[11]**	**[13]**
Release year	After 2008	2006-2007	Before 2005	After 2008	Before 2005	2006-2007
Days Active	More than 2 years	More than 2 years	Less than 1 year	More than 2 years	Less than 1 year	Less than 100 days
Downloads	More than 10,000	More than 10,000	Less than 1000	More than 10.000	Less than 1000	Less than 1000
Developers	More than one	More than one	Only one	Only one	Only one	Only one
Versions	More than 10 versions	More than 10 versions	Only one	More than 10 versions	2-10 versions	2-10 versions
Classes	101-200 classes	More than 201 classes	51-100 classes	More than 201 classes	More than 201 classes	1-50 classes
Factory	No pattern instance	5-9 pattern instances	No pattern instance	No pattern instance	10-14 pattern instances	No pattern instance
Prototype	No pattern instance	More than 15 pattern instances	No pattern instance	No pattern instance	No pattern instance	No pattern instance
Singleton	No pattern instance	More than 15 pattern instances	No pattern instance	No pattern instance	5-9 pattern instances	No pattern instance
Adapter	No pattern instance	More than 15 pattern instances	No pattern instance	10-14 pattern instances	More than 15 pattern instances	No pattern instance
Composite	No pattern instance	No pattern instance	No pattern instance	No pattern instance	1-4 pattern instances	No pattern instance
Decorator	No pattern instance	More than 15 pattern instances	No pattern instance	No pattern instance	More than 15 pattern instances	No pattern instance
Proxy	No pattern instance	5-9 pattern instances	No pattern instance	No pattern instance	1-4 pattern instances	No pattern instance
Observer	No pattern instance	1-4 pattern instances	No pattern instance	1-4 pattern instances	1-4 pattern instances	No pattern instance
State	No pattern instance	More than 15 pattern instances	10-14 pattern instances	1-4 pattern instances	More than 15 pattern instances	No pattern instance
Template	No pattern instance	More than 15 pattern instances	10-14 pattern instances	No pattern instance	10-14 pattern instances	No pattern instance
Visitor	No pattern instance	No pattern instance	1-4 pattern instances	No pattern instance	More than 15 pattern instances	No pattern instance
Category	Simulator	Board Games	Multi - User	Turn Based Strategy	Real Time Strategy	Card Games

3.4 ΚΙΝΔΥΝΟΙ ΕΓΚΥΡΟΤΗΤΑΣ

Η ενότητα αυτή ασχολείται με την παρουσίαση των εσωτερικών και εξωτερικών κινδύνων που μπορούν να επηρεάσουν την εγκυρότητα της έρευνάς μας. Κατ' αρχάς, εφόσον τα αντικείμενα μελέτης της δικής μας μελέτης περίπτωσης είναι παιχνίδια ανοιχτού λογισμικού, τα αποτελέσματα ίσως να μην αφορούν αντίστοιχες εφαρμογές κλειστού λογισμικού. Όσον αφορά την εσωτερική εγκυρότητα της εμπειρικής μελέτης, η ύπαρξη συγκεχυμένων παραγόντων αναλύθηκε στο κεφάλαιο 3.4. Είναι σημαντικό να αναφερθεί ότι το μέγεθος του δείγματος που πήραμε είναι αρκετά μικρό συγκριτικά με το συνολικό πλήθος των παιχνιδιών ανοιχτού λογισμικού και η γενίκευση των αποτελεσμάτων που προέκυψαν από το δείγμα για το τι ισχύει στο σύνολο του πληθυσμού δεν αποτελεί πολύ ασφαλή τρόπο όσον αφορά την εγκυρότητα των αποτελεσμάτων.

Επίσης, το σύνολο των δεδομένων που μελετήθηκαν ήταν αποκλειστικά παιχνίδια γραμμένα σε Java, επειδή το εργαλείο που χρησιμοποιήσαμε για την ανίχνευση προτύπων σχεδίασης μπορεί να χρησιμοποιηθεί μόνο σε δυαδικά αρχεία java. Επιπλέον, χρησιμοποιήθηκε μόνο ένα αποθετήριο κώδικα, που ονομάζεται Source forge. Ακόμα, το μέγεθος των εφαρμογών μετρήθηκε με βάση τον αριθμό των κλάσεων, (μέσω της μετρικής NOC) ,με τη λογική ότι τα πρότυπα σχεδίασης είναι συλλογές κλάσεων. Μια εναλλακτική θα ήταν να μετρήσουμε το φυσικό τους μέγεθος, με βάση το πλήθος των γραμμών κώδικά (μέσω της μετρικής LOC).

Τέλος, ένας πιθανός κίνδυνος για την εγκυρότητα των αποτελεσμάτων, είναι το γεγονός ότι ως μέγεθος της ομάδας που συμμετείχε στην δημιουργία του παιχνιδιού μετράμε τον απόλυτο αριθμό προγραμματιστών που συμμετείχαν και δεν λαμβάνουμε υπ' όψιν μας το κατά πόσο συνέβαλε ο καθένας με τις δραστηριότητές του π.χ. πόσες φορές συνέβαλε προσθέτοντας ή τροποποιώντας κώδικα.

4. ΣΥΜΠΕΡΑΣΜΑΤΑ

Η εργασία αυτή στοχεύει στη μελέτη των προτύπων σχεδίασης. Τα πρότυπα σχεδίασης προσεγγίστηκαν από δυο πλευρές. Αρχικά μέσω μιας συστηματικής ανασκόπησης της βιβλιογραφίας, προσπάθήσαμε να προσεγγίσουμε την επίδραση των προτύπων στη ποιότητα του λογισμικού. Στη συνέχεια μέσω μιας εμπειρικής μελέτης αξιολογήσαμε το βαθμό χρήσης προτύπων σχεδίασης στο ανοιχτό λογιμσικό.

Τα αποτελέσματα της συστηματικής ανασκόπησης της βιβλιογραφίας που διεξαγάγαμε, αξιολογούν ότι η επίδραση των προτύπων στην ποιότητα λογισμικού είναι το βασικό θέμα έρευνας, όσο αφορά τα πρότυπα σχεδίασης. Το μεγαλύτερο μέρος των δημοσιεύσεων στον τομέα αυτό είναι εμπειρικές μελέτες. Μέχρι σήμερα, η έρευνα για τα πρότυπα και την ποιότητα εστιάζει στα εξωτερικά χαρακτηριστικά ποιότητας των συστημάτων. Επίσης, σύμφωνα με τα αποτελέσματα, μόνο τα μισά GoF πρότυπα σχεδίασης φαίνεται να έχουν συγκριθεί με απλούστερες σχεδιαστικές λύσεις.

Σχετικά με τα αποτελέσματα της μελέτης μας για τη χρήση αντικειμενοστραφών προτύπων σχεδίασης σε 100 παιχνίδια ανοιχτού λογισμικού, επιβεβαιώνεται η εύκολη χρήση των προτύπων σχεδίασης , όπως τα Adapter, State και Singleton που εφαρμόζονται συχνότερα στο ανοιχτό λογισμικό. Οι πιο σημαντικοί παράγοντες που φαίνεται να επηρεάζουν την εφαρμογή προτύπων είναι: το μέγεθος και το κατά πόσο το πρόγραμμα είναι ενεργό, δηλαδή η διάρκεια, ο αριθμός λήψεων και το μέγεθος της ομάδας ανάπτυξης του λογισμικού. Αντίθετα, ο αριθμός των εκδόσεων του λογισμικού, δεν φαίνεται να επηρεάζει ιδιαίτερα. Τέλος, η συχνή εφαρμογή του προτύπου Adapter, μπορεί να σημαίνει, υψηλότερα επίπεδα επαναχρησιμοποίησης στον τομέα των παιχνιδιών ανοιχτού λογισμικού.

Μελλοντικοί στόχοι είναι η περεταίρω διερεύνηση του ερευνητικού αυτού πεδίου, για την καλύτερη κατανόηση του και συμπλήρωση των ερευνητικών κενών που εντοπίστηκαν στα πλαίσια αυτής της μελέτης.

ΑΝΑΦΟΡΕΣ

1. Alexander C., Ishikawa S., Silverstein M. (1977), “A Pattern Language – Town, Buildings, Construction”, *Oxford University Press*, New York.
2. Ampatzoglou A., Chatzigeorgiou A. (2007), “Evaluation of object-oriented design patterns in game development”, *In Information and Software Technology,* Elsevier, Vol.49, No 5, pp. 445-454
3. Ampatzoglou A., Stamelos I. (2010) “Software engineering research for computer games: A systematic review”, *Information and Software Technology,* Elsevier, Vol. 52, No 09, pp. 888-901.
4. Arnout K., Meyer B. (2006) “Pattern componentization: the factory example”, *In Innovations in Systems and Software Technology,* Springer, Vol. 2, No 2, pp. 65-79
5. Bansiya J., Davis C. (2002), “A Hierarchical Model for Object-Oriented Design Quality Assessment”, *In IEEE Transaction on Software Engineering*, IEEE Computer Society, Vol.28, No 1, pp. 4-17.
6. Basili V.R., Selby R.W., Hutchens D.H. (1986), “Experimentation in Software Engineering”, *In IEEE Transactions on Software Engineering,* IEEE Computer Society, Vol.13, No 07, pp. 733-743.
7. Bieman J.M., Jain D., Yang H.J. (2001), "OO design patterns, design structure, and program changes: an industrial case study", *In ICSM 2001, 17th International Conference on Software Maintenance*, IEEE Computer Society, Florence, Italy, November 07-09, 2001, pp. 580
8. Brereton P., Kitchenham B., Budgen D., Turner M., Khalil M. (2007), “Lessons from applying the systematic literature review process within the software engineering domain”, *Journal of Systems and Software*, Elsevier, Vol. 80, No. 4, pp 571-583.
9. Cai K. Y., Card D. (2008), “An analysis of topics in software engineering - 2006”, *Journal of Systems and Software*, Elsevier, Vol. 81, No 6, pp. 1051 – 1058.
10. Chatzigeorgiou A. (2005), “Object-Oriented Design: UML, Principles, Patterns and Heuristics”, *Kleidarithmos*, Athens, 1st edition.

11. Chidamber S. R., Kemerer C. F. (1994), “A Metrics Suite for Object Oriented Design”, *IEEE Transactions on Software Engineering*, IEEE, Vol. 20, No 6, pp.476-493.
12. Di Penta M., Cerulo L., Gueheneuc Y. G., Antoniol G. (2008), “An Empirical Study of Relationships between Design Pattern Roles and Class Change Proneness”, *In ICSM 2008, 24th International Conference on Software Maintenance,* IEEE Computer Society, Beijing, China, September 28 – October 4, 2008, pp. 217-226
13. Dyba T., Dingsoyr T. (2008), “Empirical studies of agile software development: A systematic review”, *Information and Software Technology*, Elsevier, Vol 50, No 9-10, pp. 833-859.
14. Feller J., Fitzgerald B. (2002), “Understanding open source software development”, *Addison-Wesley Longman*, Boston, MA, 1st edition
15. Gamma E, Helms R, Johnson R, Vlissides J. (1995), “Design Patterns: Elements of Reusable Object-Oriented Software”, *Addison-Wesley Professional*, Reading, MA, 1st edition.
16. Glass R. L., Vessey I., Ramesh V. (2002), “Research in software engineering: An analysis of the literature”, *Information and Software Technology*, Elsevier, Vol 44, No 8, pp. 491-506.
17. Hsueh N.L., Chu P.H., Chu W. (2008), “A quantitative approach for evaluating the quality of design patterns”, *In Journal of Systems and Software*, Elsevier, Vol. 81, No 8, pp. 1430-1439.
18. Huston B. (2001), “The effects of design pattern application on metric scores”, *In Journal of Systems and Software*, Elsevier, Vol. 58, No 03, pp. 261-269.
19. Jung H. W. , Kim S. G. , Chung C. S. (2004), “Measuring Software Product Quality: A Survey of ISO/iec 9126”, *IEEE Softeware*, IEEE, Vol. 21, No 05, pp. 88-92.
20. Khomh F., Gueheneuc Y.G. (2008), “Do design patterns impact software quality positively?”, *In CSMR 2008, 12th European Conference on Software Maintenance and Reengineering,* , IEEE Computer Society, Athens, Greece, April 01-04, 2008, pp. 274-278

21. Kitchenham B. (2007), “Procedures for undertaking systematic literature reviews”, *Joint Technical Report*, Computer Science Department, Keele University.
22. Kitchenham B., Brereton O. P., Budgen D., Turner M., Bailey J., Linkman S. (2009), “Systematic literature reviews in software engineering – A systematic literature review”, *Information and Software Technology*, Elsevier, Vol 51, No 1, pp 7-15.
23. Kitchenham B., Pfleeger S.L.(1996), “Software Quality: The Elusive Target”, *IEEE Software*, IEEE Computer Society, Vol, 13, No 1, pp. 12-21.
24. Kitchenham B., Pickard L., Pfleeger S.L. (1995), “Case Studies for Method and Tool Evaluation”, *In IEEE Software*, Vol. 12, No 04, pp. 52-62.
25. McShaffry M. (2003), “Game Coding Complete”, *Paraglyph Press*, Arizona.
26. Meyer B., Arnout K. (2006), “Componentization: The Visitor Example”, *In IEEE Computer*, IEEE Computer Society, Vol. 39, No 07, pp. 23-30.
27. Prechelt L., Unger B., Tichy W. F., Brossler P., Votta L. G. (2001), “A controlled experiment in maintenance comparing design patterns to simpler solutions”, *In IEEE Transactions on Software Engineering,* , IEEE Computer Society, Vol. 27, No 12, pp. 1134-1144.
28. Samoladas I., Stamelos I, Angelis L., Oikonomou A. (2004), “Open source software development should strive for even greater code maintainability”, *In Communications of the ACM,* Association of Computing Machinery, Vol. 47, No 10, pp. 83-87.
29. Sowe S.K., Angelis L., Stamelos I., Manolopoulos Y. (2007), “Using Repository of Repositories (RoRs) to Study the Growth of F/OSS Projects: A Meta-Analysis Research Approach”, *In OSS 2007*, *Open Source Software Conference,* Springer, Limeric, Ireland, June 11-14 2007,pp. 147-160.
30. Tsantalis N., Chatzigeorgiou V, Stephanides G., Halkidis S. T. (2006). "Design Pattern Detection using Similarity Scoring", *In IEEE Transaction on Software Engineering,* IEEE Computer Society, Vol. 32, No 11, pp. 896-909.
31. Vokác M., Tichy W., Sjøberg D.I.K., Arisholm E., Aldrin M. (2003), “A Controlled Experiment Comparing the Maintainability of Programs Designed

with and without Design Patterns - A Replication in a Real Programming Environment", *In Empirical Software Engineering,* Springer, Vol. 09, No 03, pp. 149-195.

32. Wendorff P. (2001), "Assessment of Design Patterns during Software Reengineering: Lessons Learned from a Large Commercial Project", *In CSMR 2001, 5th European Conference on Software Maintenance and Reengineering,* , IEEE Computer Society, Lisbon, Portugal, March 14-16, 2001, pp. 77
33. Wohlin C., Runeson P., Host M., Ohlsson M.C., Regnell B., Wesslen A. (2000), "Experimentation in Software Engineering", *Kluwer Academic Publishers*, Boston/ Dordrecht/ London, 1st edition.

ΠΑΡΑΡΤΗΜΑ Α - Μελέτες που αναφέρονται στην ανασκόπηση της βιβλιογραφίας

		Passed Exclusion Phase?		
	Title	**Title**	**Abstract**	**Full Text**
1.	Cloning considered harmful: patterns of cloning in software	no		
2.	A brief summary of cognitive patterns for program comprehension	yes	no	
3.	A catalog of architectural primitives for modeling architectural patterns	yes	yes	yes
4.	A Composite Design Pattern for Object Frameworks	yes	yes	yes
5.	A Controlled Experiment Comparing the Maintainability of Programs Designed with and without Design Patterns—A Replication in a Real Programming Environment	yes	yes	yes
6.	A Controlled Experiment in Maintenance Comparing Design Patterns to Simpler Solutions	yes	yes	yes
7.	A decision-making pattern for guiding the enterprise knowledge development process	no		
8.	A Declarative Evolution Framework for Object-Oriented Design Patterns	yes	yes	yes
9.	A Design Pattern based Approach to Generating Synchronization Adaptors from Annotated IDL	yes	N/A	no
10.	A Design Pattern for Autonomous Vehicle Software Control Architectures	yes	yes	yes
11.	A Dictionary-Based Compressed Pattern Matching Algorithm	yes	no	
12.	A Dynamic Indexing Structure for Searching Time-Series Patterns	no		
13.	A Formal Pattern Language for Refactoring of Lisp Programs	no		
14.	A framework and patterns for the specification of reactive systems	no		
15.	A Framework for Source Code Search Using Program Patterns	no		
16.	A Graph Pattern Matching Approach to Software Architecture Recovery	no		
17.	A Methodology for the Automated Introduction of Design Patterns	yes	yes	yes
18.	A Metric-Based Approach to Enhance Design Quality through Meta-pattern Transformations	yes	no	
19.	A model-driven framework for representing and applying design patterns	yes	yes	yes
20.	A pattern language for designing e-business architecture	yes	no	
21.	A Pattern Language Model for Framework Development	no		
22.	A Pattern Recognition Approach for Software Engineering Data Analysis	yes	no	
23.	A pattern system for the development of collaborative applications	no		
24.	A pattern-based application generator for building simulation	yes	N/A	no
25.	A pattern-based approach to protocol mediation for web services composition	no		
26.	A Pattern-Based Approach to Structural Design Composition	yes	yes	yes
27.	A pattern-based object-linking mechanism for component-based software development environments	yes	no	
28.	A pattern-based outlier detection method identifying abnormal attributes in software project data	no		
29.	A Pattern-Based Technique for Developing UML Models of Access Control Systems	yes	no	
30.	A practical pattern recovery approach based on both structural and behavioral analysis	yes	yes	yes
31.	A Process for Framework Construction Based on a Pattern Language	no		
32.	A Productivity Metric Based on Statistical Pattern Recognition	no		
33.	A Quality Verification Model for Design Pattern	yes	yes	no
34.	A quantitative approach for evaluating the quality of design patterns	yes	yes	yes
35.	A Role-Based Meta modeling Approach to Specifying Design	yes	yes	yes

	Patterns			
36.	A Semantic-Aware Publish/Subscribe System with RDF Patterns	no		
37.	A Simple Tree Pattern Matching Algorithm for Code Generator	no		
38.	A skewed distributed indexing for skewed access patterns on the wireless broadcast	no		
39.	A static reference flow analysis to understand design pattern behavior	yes	yes	yes
40.	A string pattern—matching algorithm	no		
41.	A Temporal Semantics for Workflow Control Patterns	no		
42.	A Transformational Viewpoint on Design Patterns	yes	yes	yes
43.	A Two Phase Approach to Design Pattern Recovery	yes	yes	yes
44.	A UML-Based Pattern Specification Technique	yes	yes	yes
45.	Adding pattern related information in structural and behavioral diagrams	yes	yes	yes
46.	Agent System Deployment method based on agent patterns	no		
47.	Agent-oriented software patterns for rapid and affordable robot programming	no		
48.	An Analysis of the Security Patterns Landscape	no		
49.	An approach to mining call-usage patterns with syntactic context	no		
50.	An architectural pattern for non-functional dependability requirements	yes	no	
51.	An Efficient Implementation of Static String Pattern Matching Machines	no		
52.	An empirical analysis on distribution patterns of software maintenance effort	no		
53.	An empirical study of the relationships between design pattern roles and class change proneness	yes	yes	yes
54.	An empirical study on the efficiency of different design pattern representations in UML class diagrams	yes	yes	yes
55.	An empirical study on the evolution of design patterns	yes	yes	yes
56.	An Empirical Study Using Task Assignment Patterns to Improve the Accuracy of Software Effort Estimation	no		
57.	An index organization for applications with highly skewed access patterns	no		
58.	An Intelligent Control Architecture for Adaptive Service-Based Software Systems with Workflow Patterns	no		
59.	Analysis of Long Term File Reference Patterns for Application to File Migration Algorithms	no		
60.	Analysis of signature change patterns	yes	no	
61.	Announcing a new Quarterly IEEE Transactions ... IEEE Transactions on Pattern Analysis and Machine Intelligence	no		
62.	Another view of computer science ethics: Patterns of responses among computer scientists	no		
63.	Anti Patterns in software architecture	yes	N/A	no
64.	Application of Design Patterns for the Self-Development of a Java Preprocessor	no		
65.	Architecting as decision making with patterns and primitives	yes	yes	yes
66.	Architectural Organizational Patterns	no		
67.	Architectural Styles, Design Patterns, and Objects	yes	no	
68.	Architecture-Centric Software Evolution by Software Metrics and Design Patterns	yes	yes	yes
69.	Artificial Patterns	no		
70.	Assessment of Design Patterns during Software Reengineering: Lessons Learned from a Large Commercial Project	yes	yes	yes
71.	Assurance patterns for distributed real-time embedded systems	no		
72.	Augmenting pattern-based architectural recovery with flow analysis: Mosaic-a case study	yes	no	
73.	Automated Identification of LTL Patterns in Natural Language Requirements	no		
74.	Automated refactoring to introduce design patterns	yes	yes	yes
75.	Automated software evolution towards design patterns	yes	yes	yes

76.	Automated verification of security pattern compositions	no		
77.	Automatic verification of design patterns in Java	yes	yes	yes
78.	Behavioral Pattern Identification through Visual Language Parsing and Code Instrumentation	yes	yes	yes
79.	Bridging patterns: An approach to bridge gaps between SE and HCI	yes	yes	no
80.	Building intrusion pattern miner for Snort network intrusion detection system	no		
81.	Building Systems Using Analysis Patterns	yes	no	
82.	Business patterns and viewpoints	yes	N/A	no
83.	Can design pattern detection be useful for legacy system migration towards SOA?	yes	yes	yes
84.	Case Studies of Visual Language Based Design Patterns Recovery	yes	yes	yes
85.	Characteristics of user file-usage patterns	no		
86.	Code web: data mining library reuse patterns	no		
87.	Communication patterns in geographically distributed software development and engineers' contributions to the development effort	no		
88.	Comparing Approaches to Mining Source Code for Call-Usage Patterns	yes	yes	no
89.	Component mining: a process and its pattern language	no		
90.	Composing pattern-based components and verifying correctness	yes	no	
91.	Composition of Software Architectures from Reusable Architecture Patterns	yes	yes	no
92.	Composition Patterns: an approach to designing reusable aspects	yes	no	
93.	Concern patterns and analysis	no		
94.	ConcernMorph: metrics-based detection of crosscutting patterns	yes	no	
95.	Concurrency design patterns, software quality attributes and their tactics	yes	yes	no
96.	Consolidating and applying the SDL-pattern approach: a detailed case study	no		
97.	Coupling of Design Patterns: Common Practices and Their Benefits	yes	yes	yes
98.	Coupling Patterns in the Effective Reuse of Open Source Software	yes	yes	no
99.	Cultural patterns in software process mishaps: incidents in global projects	no		
100.	Data Mining Library Reuse Patterns in User-Selected Applications	yes	no	
101.	Data mining library reuse patterns using generalized association rules	no		
102.	Defect Frequency and Design Patterns: An Empirical Study of Industrial Code	yes	yes	yes
103.	DeMIMA: A Multilayered Approach for Design Pattern Identification	yes	yes	yes
104.	Description templates for agent-oriented patterns	no		
105.	Design dysphasia and the pattern maintenance cycle	yes	yes	yes
106.	Design pattern concerns for software evolution	yes	yes	yes
107.	Design pattern detection in Eiffel systems	yes	yes	yes
108.	Design Pattern Detection Using Similarity Scoring	yes	yes	yes
109.	Design Pattern Mining Enhanced by Machine Learning	yes	yes	yes
110.	Design Pattern Rationale Graphs linking design to source	yes	yes	yes
111.	Design Pattern Recovery by Visual Language Parsing	yes	yes	yes
112.	Design pattern recovery through visual language parsing and source code analysis	yes	yes	no
113.	Design Patterns and Change Proneness A Replication Using Proprietary C# Software	yes	yes	yes
114.	Design Patterns and Change Proneness: An Examination of Five Evolving Systems	yes	yes	yes
115.	Design Patterns between programming and software design	yes	no	
116.	Design patterns for developing dynamically adaptive systems	yes	yes	no
117.	Design patterns for monitoring adaptive ULS systems	yes	yes	no
118.	Design patterns for object-oriented software development-tutorial	yes	no	
119.	Design recovery by automated search for structural design patterns in object-oriented software	yes	yes	yes
120.	Design Reuse through Frameworks and Patterns	yes	yes	no

121.	Designing embedded systems using patterns: A case study	yes	no	
122.	Detecting higher-level similarity patterns in programs	no		
123.	Detecting implicit collaboration patterns	yes	yes	no
124.	Detecting large number of infeasible paths through recognizing their patterns	no		
125.	Digging into UML models to remove performance anti patterns	yes	yes	no
126.	Discovering New Change Patterns in Object-Oriented Systems	yes	no	
127.	Discovering Patterns of Change Types	no		
128.	Discovery of SOA patterns via model checking	yes	yes	yes
129.	Discriminative pattern mining in software fault detection	yes	no	
130.	Do Design Patterns Impact Software Quality Positively?	yes	yes	yes
131.	Do Maintainers Utilize Deployed Design Patterns Effectively	yes	yes	yes
132.	Do Structural Design Patterns Promote Design Stability?	yes	yes	yes
133.	Documenting Maintenance Tasks Using Maintenance Patterns	yes	no	no
134.	Documenting Pattern Use in Java Programs	yes	yes	yes
135.	Does the "Refactor to Understand" Reverse Engineering Pattern Improve Program Comprehension?	no		
136.	Dynamic analysis of java applications for multithreaded anti patterns	yes	no	
137.	DynaMine: finding common error patterns by mining software revision histories	no		
138.	Effect of Fault Distribution and Execution Patterns on Fault Exposure in Software: A Simulation Study.	no		
139.	Efficient data mining for web navigation patterns	no		
140.	Efficient Identification of Design Patterns with Bit-vector Algorithm	yes	yes	yes
141.	Efficient mining and prediction of user behavior patterns in mobile web systems	no		
142.	Efficient mining of frequent XML query patterns with repeating-siblings	no		
143.	Efficient Monitoring of Parametric Context-Free Patterns	no		
144.	Efficient Partial Multiple Periodic Patterns Mining without Redundant Rules	no		
145.	Efficient storage and querying of sequential patterns in database systems	no		
146.	Embedded behavior pattern languages: A contribution to a taxonomy of case languages	no		
147.	Empirical study of exchange patterns during software peer review meetings	no		
148.	Encapsulating windows-based software applications into reusable components with design patterns	yes	no	
149.	Energy-efficient real-time object tracking in multi-level sensor networks by mining and predicting movement patterns	no		
150.	Enhancing class commutability in the deployment of design patterns	yes	yes	yes
151.	Enhancing usability testing through data mining techniques: A novel approach to detecting usability problem patterns for a context of use	no		
152.	ESP^3: A Language for Pattern Description and a System for Pattern Recognition	no		
153.	Evaluating Pattern catalogs the computer games experience	yes	yes	no
154.	Evaluation experiments on the detection of programming patterns using software metrics	yes	yes	no
155.	Evaluation of object-oriented design patterns in game development	yes	yes	yes
156.	Event-Based Input Validation Using Design-by-Contract Patterns	no		
157.	Evolution Support by Homogeneously Documenting Patterns, Aspects and Traces	yes	yes	yes
158.	Exception handling patterns for processes	yes	no	
159.	Experiments on Design Pattern Discovery	yes	yes	yes
160.	Exploiting design patterns to automate validation of class invariants.	yes	yes	yes
161.	Expressing and organizing real-time specification patterns via temporal logics	no		
162.	Extracting Change-patterns from CVS Repositories	yes	no	

163.	Facilitating software extension with design patterns and Aspect-Oriented Programming	yes	yes	yes
164.	Facilitator Agent Design Pattern of Procurement Business Systems	no		
165.	Fault patterns in Matlab	no		
166.	Finding synchronization defects in java programs: extended static analyses and code patterns	no		
167.	Fingerprinting design patterns	yes	yes	yes
168.	Flexible Self-Management Using the Model-View-Controller Pattern	no		
169.	Formal refinement patterns for goal-driven requirements elaboration	yes	no	
170.	Formal specification of design pattern combination using BPSL	yes	yes	yes
171.	Formal specification of the variants and behavioral features of design patterns	yes	yes	yes
172.	Formalizing design patterns	yes	yes	yes
173.	From security patterns to implementation using Petri nets	no		
174.	Generating a Pattern-Based Application Development Environment for Enterprise JavaBeans	yes	no	
175.	Generative Design Patterns	yes	yes	yes
176.	Graph-based mining of multiple object usage patterns	yes	no	
177.	Guest Editorial Data Structures and Pattern Recognition	no		
178.	Guest Editors' Introduction: Software Patterns	no		
179.	Guiding the Application of Design Patterns Based on UML Models	yes	yes	yes
180.	HCI pattern semantics in XML: a pragmatic approach	no		
181.	Hiding Sensitive Patterns in Association Rules Mining	no		
182.	Identification of design motifs with pattern matching algorithms	yes	yes	yes
183.	Idioms and Patterns as Architectural Literature	yes	no	
184.	IEEE Announcing New Quarterly IEEE Transactions... IEEE Transactions on Pattern Analysis and Machine Intelligence	no		
185.	Implementing an Agent System Using N-tier Pattern-Based Framework	no		
186.	Implementing protocols via declarative event patterns	no		
187.	Incremental pattern matching in the viatra model transformation system	no		
188.	Industrial experience with design patterns	yes	yes	yes
189.	Inferring structural patterns for concern traceability in evolving software	no		
190.	Instantiating and Detecting Design Patterns: Putting Bits and Pieces Together	yes	yes	yes
191.	Integrated design patterns for database applications	no		
192.	Integrating in-process software defect prediction with association mining to discover defect pattern	no		
193.	Integration in Component-Based Software Development Using Design Patterns	yes	yes	yes
194.	Interactive Pattern Recognition: A System and Data Structure	no		
195.	Is This a Pattern?	yes	yes	yes
196.	Knowledge centered assessment pattern: an effective tool for assessing safety concerns in software architecture	no		
197.	Language features meet design patterns: raising the abstraction bar	yes	yes	yes
198.	License integration patterns: Addressing license mismatches in component-based development	no		
199.	Linking usability to software architecture patterns through general scenarios	yes	no	
200.	Looking Beyond Software to Understand Software Design Patterns	yes	no	
201.	Making design patterns explicit in FACE: a frame work adaptive composition environment	yes	N/A	no
202.	Matching attack patterns to security vulnerabilities in software-intensive system designs	no		
203.	Measuring and Improving Design Patterns Testability	yes	yes	yes
204.	Measuring precision for static and dynamic design pattern recognition as a function of coverage	yes	yes	yes

205.	Metrics for applying GOF design patterns in Refactoring processes	yes	yes	yes
206.	Micro pattern evolution	yes	yes	no
207.	Migrating COBOL systems to the Web by using the MVC design pattern	no		
208.	Mining API patterns as partial orders from source code: from usage scenarios to specifications	yes	no	
209.	Mining Coding Patterns to Detect Crosscutting Concerns in Java Programs	yes	no	
210.	Mining Design Patterns from C++ Source Code	yes	yes	yes
211.	Mining frequent patterns in image databases with 9D-SPA representation	no		
212.	Mining Sequential Patterns Using Graph Search Techniques	yes	no	
213.	Model-based user interface engineering with design patterns	yes	yes	no
214.	Modeling behavioral design patterns of concurrent objects	no		
215.	Modeling Requirements Patterns with a Goal and PF Integrated Analysis Approach	no		
216.	Models, techniques, and algorithms for finding, selecting, and displaying patterns in strings and other discrete objects	no		
217.	Naming: Design Pattern and Framework	yes	N/A	no
218.	Native patterns	yes	yes	no
219.	Notable design patterns for domain-specific languages	no		
220.	Object Analysis Patterns for Embedded Systems	no		
221.	Object Oriented Design Pattern Inference	yes	yes	yes
222.	Object-oriented design patterns recovery	yes	yes	yes
223.	Object Oriented Reengineering Patterns	yes	no	
224.	Observations on patterns of development in open source software projects	no		
225.	On 'A Framework for Source Code Search Using Program Patterns'	no		
226.	On Identifying Bug Patterns in Aspect-Oriented Programs	no		
227.	On the Complexity of Finding Emerging Patterns	yes	no	
228.	On the design of more secure software-intensive systems by use of attack patterns	no		
229.	On the Role of Design Patterns in Quality-Driven Re-engineering	yes	yes	yes
230.	On the uniformity of software evolution patterns	yes	no	
231.	OO Design Patterns, Design Structure, and Program Changes: An Industrial Case Study	yes	yes	yes
232.	Organizing Security Patterns	no		
233.	Package Patterns for Visual Architecture Recovery	yes	no	
234.	Parsing Languages by Pattern Matching	no		
235.	Past, Present, and Future Trends in Software Patterns	yes	yes	yes
236.	PAT: A pattern classification approach to automatic reference oracles for the testing of mesh simplification programs	no		
237.	Pattern and Policy Driven Log Analysis for Software Monitoring	no		
238.	Pattern based reverse-engineering of design components	yes	yes	yes
239.	Pattern matching and pattern-directed invocation in systems programming languages	no		
240.	Pattern matching for design concept localization	no		
241.	Pattern-based design recovery of Java software	yes	no	
242.	Pattern-based Reengineering of Software Systems	yes	yes	yes
243.	Pattern-oriented distributed system architectures	yes	N/A	no
244.	Patterns	yes	N/A	no
245.	Patterns and performance of distributed real-time and embedded publisher/subscriber architectures	no		
246.	Patterns and technologies for enabling supply chain traceability through collaborative e-business	no		
247.	Patterns in Effective Distributed Software Development	yes	no	
248.	Patterns in Postmortems	yes	no	
249.	Patterns in the analysis, design and implementation of frameworks	yes	no	
250.	Patterns of conflict among software components	no		

251.	Patterns Topology for Performance Evaluation	no		
252.	patterns, frameworks, and middleware: their synergistic relationships	yes	yes	yes
253.	Patterns-based evaluation of open -source BPM systems: The cases of jBPM, OpenWFE, and Enhydra Shark	yes	no	
254.	Performance of circuit-switched interconnection networks under no uniform traffic patterns	no		
255.	Piecemeal Migration of a Document Archive System with an Architectural Pattern Language	yes	no	
256.	Playing roles in design patterns: An empirical descriptive and analytic study	yes	yes	yes
257.	Poor Performing Patterns of Code: Analysis and Detection	yes	yes	yes
258.	Practice patterns for architecture reconstruction	yes	no	
259.	Precise Modeling of Design Patterns in UML	yes	yes	yes
260.	Precise specification and automatic application of design patterns	yes	yes	yes
261.	Precise Specification to Compound Patterns with ExLePUS	no		
262.	Proactive Views on Concrete Aspects: A Pattern Documentation Approach for Software Evolution	no		
263.	Project-specific deletion patterns	no		
264.	Properties of Signature Change Patterns	yes	no	
265.	Quality-Attribute Based Economic Valuation of Architectural Patterns	yes	yes	no
266.	Rapid Embedded System Testing Using Verification Patterns	no		
267.	Rapid Verification of Embedded Systems Using Patterns	yes	no	
268.	Patterns in property specifications for finite-state verification	no		
269.	Real-Time Specification Patterns	no		
270.	Recognizing behavioral patterns at runtime using finite automata	yes	yes	yes
271.	Reconciling usability and interactive system architecture using patterns	yes	no	
272.	Recovering Interaction Design Patterns in Web Applications	no		
273.	Reducing search space of auto-tuners using parallel patterns	no		
274.	Reengineering using design patterns	yes	yes	yes
275.	Refinement patterns for rapid development of dependable systems	yes	no	
276.	Refreshing Patterns	yes	no	
277.	Relating expectations to automatically recovered design patterns	yes	yes	yes
278.	Release Pattern Discovery via Partitioning: Methodology and Case Study	yes	yes	no
279.	Release Pattern Discovery: A Case Study of Database Systems	yes	no	
280.	Responsibilities and Rewards: Specifying Design Patterns	yes	yes	yes
281.	Revealer: a lexical pattern matcher for architecture recovery	yes	no	
282.	Reverse Engineering of Design patterns from Java Source Code	yes	yes	yes
283.	Round Trip engineering with design patterns, UML, Java and C++	yes	N/A	no
284.	Scenario evolution in requirements elicitation processes: scenario pattern and framework approach	yes	no	
285.	ScriptEase: Generative Design Patterns for Computer Role Playing Games	yes	no	
286.	Searching Design Patterns in Source Code	yes	yes	yes
287.	Sharing Requirements Engineering Experience Using Patterns	no		
288.	Soft goal Traceability Patterns	no		
289.	Software Architecture Recovery based on Pattern Matching	yes	no	
290.	Software Design Improvement through Anti-patterns Identification	yes	yes	no
291.	Software Evolution, Volatility and Lifecycle Maintenance Patterns: A Longitudinal Analysis	yes	no	
292.	Software Library Usage Pattern Extraction Using a Software Model Checker	no		
293.	Software pre-patterns as architectural knowledge	yes	yes	yes
294.	Software project management anti-patterns	yes	no	
295.	Software Test Selection Patterns and Elusive Bugs	no		
296.	Some Domain Patterns in Web Application Framework	no		
297.	Specialization Patterns	no		
298.	Specification and design of component-based coordination systems by	no		

	integrating coordination patterns			
299.	Specification Patterns for probabilistic quality properties	no		
300.	Specifying Behavioral Features of Design Patterns in First Order Logic	yes	yes	yes
301.	SPI Patterns: Learning from Experience	yes	no	
302.	SQUIRE: Sequential pattern mining with quantities	no		
303.	Static and dynamic structure in design patterns	yes	yes	yes
304.	Structured Document Framework for Design Patterns Based on SGML	no		
305.	Survival Patterns in Fast-Moving Software Organizations	no		
306.	Systems reengineering patterns	yes	no	
307.	Taming coincidental correctness: Coverage refinement with context patterns to improve fault localization	no		
308.	Temporal moving pattern mining for location-based service	no		
309.	Testing Extensible Design Patterns in Object-Oriented Frameworks through Scenario Templates	yes	yes	yes
310.	The correlation between parallel patterns and multi-core benchmarks	yes	no	
311.	The Dublo Architecture Pattern for Smooth Migration of Business Information Systems An Experience Report	no		
312.	The effects of design pattern application on metric scores	yes	yes	yes
313.	The Essence of the Visitor Pattern	yes	yes	yes
314.	The factory pattern in API design: A usability Evaluation	yes	yes	yes
315.	The Growing Divide in the Patterns World	no		
316.	The maintenance and evolution of resource-constrained embedded systems created using design patterns	yes	no	
317.	The Origins of Pattern Theory: The Future of the Theory, and the Generation of a Living World	yes	N/A	no
318.	The value of a usability-supporting architectural pattern in software architecture design: a controlled experiment	yes	yes	no
319.	Timed Automata Patterns	no		
320.	Tool Support for Design Pattern Recognition at Model Level	yes	yes	yes
321.	Toward an understanding of bug fix patterns	no		
322.	Toward effective deployment of design patterns for software extension: a case study	yes	yes	yes
323.	Toward Exception-Handling Best Practices and Patterns	no		
324.	Towards a Benchmark for Evaluating Design Pattern Miner Tools	yes	yes	yes
325.	Towards a Problem Oriented Engineering Theory of Pattern-Oriented Analysis and Design	no		
326.	Towards a 'Safe' Use of Design Patterns to Improve OO Software Testability	yes	yes	yes
327.	Towards a systematic approach to the capture of patterns within a business domain	no		
328.	Towards pattern-based design recovery	yes	yes	yes
329.	Towards Reusable Measurement Patterns	yes	no	
330.	Towards software process patterns: An empirical analysis of the behavior of student teams	no		
331.	Towards the determination of typical failure patterns	yes	no	
332.	Transformation of Legacy Software into Client/Server Applications through Pattern-Based Re Architecturing	no		
333.	Tutorial: mastering design patterns	yes	no	
334.	Two Controlled Experiments Assessing the Usefulness of Design Pattern Documentation in Program Maintenance	yes	yes	yes
335.	Type-check elimination: two object-oriented reengineering patterns	no		
336.	Understanding and Aiding Code Evolution by Inferring Change Patterns	no		
337.	Understanding bug fix patterns in verilog	no		
338.	Understanding the Power of Abstraction in Patterns	yes	no	
339.	Usability Supporting Architectural Patterns	yes	yes	no
340.	Usability-enabling guidelines: a design pattern and software plug-in	yes	yes	no

	solution			
341.	Using Architectural Patterns and Blueprints for Service-Oriented Architecture	yes	no	
342.	Using concept analysis to detect co-change patterns	yes	no	
343.	Using Metrics to Identify Design Patterns in Object-Oriented Software	yes	yes	yes
344.	Using Pattern-Join and Purchase-Combination for Mining Web Transaction Patterns in an Electronic Commerce Environment	no		
345.	Using patterns for the refinement and translation of UML models: A controlled experiment	yes	no	
346.	Using Patterns to Capture Architectural Decisions	yes	no	
347.	Using Patterns To Create Component Documentation	yes	yes	no
348.	Using Patterns to Design Rules in Workflows	no		
349.	Using Patterns to Improve Our Architectural Vision	yes	yes	yes
350.	Using Patterns to Model Variability in Product Families	yes	yes	yes
351.	Using social networking and semantic web technology in software engineering – Use cases, patterns, and a case study	no		
352.	Using use case patterns to estimate reusability in software systems	yes	no	
353.	Visual identification of software evolution patterns	no		
354.	Visualizing Design Patterns in Their Applications and Compositions	yes	yes	yes
355.	Work Experience versus Refactoring to Design Patterns: A controlled Experiment	yes	yes	yes
356.	Why Explore Object Methods, Patterns, and Architectures?	yes	yes	yes
357.	A new taxonomy of sub linear right-to-left scanning keyword pattern matching algorithms	yes	no	
358.	A Boyer–Moore-style algorithm for regular expression pattern matching	no		
359.	Theory and practice of unparsed patterns for meta compilation	no		
360.	Sequences as a basis for pattern language composition	no		
361.	A taxonomy of sub linear multiple keyword pattern matching algorithms	yes	no	
362.	Refined compilation of pattern-matching for functional languages	no		
363.	Parallel program analysis and restructuring by detection of point-to-point interaction patterns and their transformation into collective communication constructs	no		
364.	Combinations of abstract domains for logic programming: open product and generic pattern construction	no		
365.	Generating function versions with rational strictness patterns	no		
366.	Playing with patterns, searching for strings	yes	no	
367.	Formal derivation of a pattern matching algorithm	yes	no	

ΠΑΡΑΡΤΗΜΑ Β - Μελέτες που συμπεριλαμβάνονται στην ανασκόπηση της βιβλιογραφίας

[S1] P. S. C. Alencar, D. D. Cowan, J. Dong and C. J. P. de Lucena, "A Pattern-Based Approach to Structural Design Composition", *23rd International Computer Software and Applications Conference(COMPSAC'99),* IEEE, pp. 160-165, Phoenix, Arizona, 25-26 October 1999

[S2] H. A. Amiot, P. Cointe, Y. G. Gueheneuc. and N. Jussien, "Instantiating and Detecting Design Patterns: Putting Bits and Pieces Together", *Proceedings of the 16th IEEE international conference on Automated software engineering,* ACM, pp.166, San Diego, California, 26-29 November 2001

[S3] A. Ampatzoglou and A. Chatzigeorgiou, "Evaluation of object-oriented design patterns in game development", *Information and Software Technology,* Elsevier, 49 (5), pp.445-454, May 2007.

[S4] G. Antoniol, G. Casazza, M .Di Penta and R .Fiutem, "Object-Oriented design patterns recovery", *Journal of Systems and Software,* Elsevier, 59 (2), pp.181-196, November 2001

[S5] G. Antoniol, R. Fiutem and L. Christoforetti, "Using Metrics to Identify Design Patterns in Object-Oriented Software", *Proceedings of the 5th International Symposium on Software Metrics,* IEEE, pp. 23, Bethesda, Maryland, 20-21 March 1998

[S6] F. Arcelli, C. Tosi and M. Zanoni, "Can design pattern detection be useful for legacy systemmigration towards SOA?", *Proceedings of the 2nd international workshop on Systems development in SOA environments(ICSE '08),* IEEE, pp. 63-68, Leipzig, Germany, 10-18 May 2008.

[S7] A. Asencio, S. Cardman, D. Harris and E. Laderman, "Relating Expectations to Automatically Recovered Design Patterns", *Proceedings of the Ninth Working Conference on Reverse Engineering (WCRE'02),* pp. 87, Richmond, Virginia, 29 October – 01 November 2002.

[S8] L. Aversano , G. Canfora, L. Cerulo, C. D. Grosso and M. Di Penta, "An empirical study on the evolution of design patterns", *Foundations of Software Engineering (FSE' 07),* ACM, pp. 385-394, Dubrovnik, Croatia, 3-7 September 2007

[S9] Z. Balanyi and R. Ferenc, "Mining Design Patterns from C++ Source Code", *Proceedings of the International Conference on Software Maintenance,* IEEE, Amsterdam, The Netherlands, 22-26 September 2003

[S10] E. L. A. Baniassad, G. C. Murphy and C .Schwanninger, "Design Pattern Rationale Graphs: linking design to source", *Proceedings of the 25th International Conference on Software Engineering,* IEEE, pp. 352-362, Portland, Oregon, 03-10 May 2003

[S11] B. Baudry, Y. Le Sunye and J. M. Jezequel, "Towards a 'Safe' Use of Design Patterns to Improve OO Software Testability", *Proceedings of the 12th International Symposium on Software Reliability Engineering,* IEEE, pp. 324, Hong Kong, China, 27-30 November 2001

[S12] B. Baudry, Y. Le Traon, G. Sunye and J. M. Jezequel, "Measuring and Improving Design Patterns Testability", *Proceedings of the 9th International Symposium on Software Metrics ,* IEEE, pp. 50, Sydney, Australia, 03-05 September 2003

[S13] I. Bayley and H. Zhu, "Formal specification of the variants and behavioural features of design patterns", *Journal of Systems and Software,* Elsevier, 83 (2), pp. 209-221, February 2010.

[S14] I. Bayley and H. Zhu, “Specifying Behavioural Features of Design Patterns in First Order Logic”, *Proceedings of the 2008 32nd Annual IEEE International Computer Software and Applications Conference (COMPSAC ‘08),* IEEE, pp. 203-210, Turku, Finland, 28 July-01 August 2008

[S15] K. Beck, R. Crocker, G. Meszaros, J. Vlissides, J. O. Coplien, L. Dominic and F. Paulisch, “Industrial experience with design patterns”, *Proceedings of the 18th international conference on Software engineering (ICSE’96),* IEEE, pp. 103-114, Berlin, Germany, 25-29 March 1996

[S16] J. M. Bieman, G. Straw, H. Wang, P. W. Munger and R. T. Alexander, “Design Patterns and Change Proneness: An Examination of Five Evolving Systems”, *Proceedings of the 9th International Symposium on Software Metrics,* IEEE, pp. 40, Sydney, Australia, 03-05 September 2003

[S17] J. Bishop, “Language features meet design patterns: raising the abstraction bar”, *Proceedings of the 2nd international workshop on The role of abstraction in software engineering (ICSE’08),* IEEE, pp. 1-7, Leipzig, Germany, 10-18 May 2008.

[S18] A. Blewitt, A. Bundy and I. Stark, “Automatic verification of design patterns in Java”, *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering,* ACM, pp. 224-232, Long Beach, CA, 07-11 November 2005

[S19] G. Bortis and A. van der Hoek, “Software pre-patterns as architectural knowledge”, *Proceedings of the 3rd international workshop on Sharing and reusing architectural knowledge (ICSE ’08),* IEEE, pp. 19-22, Leipzig, Germany, 10-18 May 2008

[S20] *G. E. Boussaidi and H. Mili,* “A model driven framework for representing and applying design patterns”, *31st Annual International Computer Software and Applications Conference (COMPSAC’07),* IEEE, pp 97-100, Beijing, China, 24-27 July 2007

[S21] L. C. Briannd, Y. Labiche and A. Sauve, “Guiding the Application of Design Patterns Based on UML Models”, *Proceedings of the 22nd IEEE International Conference on Software Maintenance,* IEEE, pp. 234-243, Philadelphia, Pennsylvania, 24-27 September 2006

[S22] F. Buschmann, K. Henney and D. C .Schmidt, “Past, Present and Future in Software Patterns”, *IEEE Software,* IEEE, 24 (4), pp. 31-37, July 2007

[S23] M. I. Cagnin, R. T. V. Braga, P. C. Masiero, I. Usp, R. Penteado, and Dc UFSCar, “Reengineering using Design Patterns”, *Proceedings of the Seventh Working Conference on Reverse Engineering (WCRE'00*), pp.118, Brisbane, Australia, 23-25 November 2000

[S24] R. Chaabane, “Poor performing patterns of code: Analysis and Detection”, *Proceedings of the IEEE International Conference on Software Maintenance (ICSM'07),* IEEE, pp. 501-502, Paris, France, 02-05 October 2007

[S25] M. O. Cinneide, “Automated refactoring to introduce design patterns”, *Proceedings of the 22nd international conference on Software engineering (ICSE’00),* IEEE, pp.722-724, Limeric, Ireland, 4-11 June 2000

[S26] M. O Cinneide and P. Nixon, “Automated software evolution towards design patterns”, *Proceedings of the 4th International Workshop on Principles of Software Evolution (ICSE’01*), IEEE, pp.162-165, Vienna, Austria, 12-19 May 2001

[S27] G. Costagliola, A. De Lucia, V. Deufemia, C. Gravino and M. Risi, “Case Studies of Visual Language Based Design Patterns Recovery”, *Proceedings of the Conference on Software Maintenance and Reengineering,* IEEE, pp. 165-174, Bari, Italy, 22-24 March 2006

[S28] G. Costagliola, A. De Lucia, V. Deufemia, C. Gravino and M. Risi, “Design Pattern Recovery by Visual Language Parsing”, *Proceedings of the 29th international conference on Software Engineering,* IEEE, pp 102-111, Manchester, UK, 21-23 March 2005

[S29] A. De Lucia, V. Deufemia, C. Gravino and M. Risi, “A Two Phase Approach to Design Pattern Recovery”, *Proceedings of the 11th European Conference on Software Maintenance and Reengineering,* IEEE, pp. 297-306, Amsterdam, the Netherlands, 21-23 March 2007

[S30] A. De Lucia, V. Deufemia, C. Gravino and M. Risi, “Behavioral Pattern Identification through Visual Language Parsing and Code Instrumentation”, *Proceedings of the 2009 European Conference on Software Maintenance and Reengineering,* IEEE, pp. 99-108, Kaiserslautern, Germany, 24-27 March 2009

[S31] M. Di Penta, L. Cerulo, Y. G. Gueheneuc and G. Antoniol, “An empirical study of the relationships between design pattern roles and class change proneness” , *Proceedings of the IEEE International Conference on Software Maintenance (ICSM'08),* IEEE, pp.217-226, Beijing, China, 28 September – 04 October 2008

[S32] M. Di Penta, A. Santone and M. L. Villani, “Discovery of SOA patterns via model checking”, *2nd international workshop on Service oriented software engineering: in conjunction with the 6th ESEC/FSE joint meeting,* ACM, pp. 8-14, Dubrovnik, Croatia, 26-28 March 2007.

[S33] J. Dong , “Adding pattern related information in structural and behavioural diagramms”, *Information and Software Technology*, Elsevier, 46 (5), pp.293-300, 15 April 2004

[S34] J. Dong, S. Yang and K. Zhang, “Visualizing Design Patterns in Their Applications and Compositions”, *IEEE Transactions on Software Engineering,* IEEE, 33 (7), pp. 433-453, July 2007

[S35] J.Dong and Y. Zhao, “Experiments on Design Pattern Discovery”, *Proceedings of the Third International Workshop on Predictor Models in Software Engineering (ICSE’07),* IEEE, pp. 12, Minneapolis, Minnesota, 23-25 May 2007

[S36] A. H. Eden, A. Yehudai, and J. Gil, “Precise specification on automatic application of design patterns”, *Proceedings of the 12th international conference on Automated software engineering (formerly: KBSE),* ACM, pp. 143, Lake Tahoe, CA, 02-05 November 1997.

[S37] E. Eide, A. Reid, J. Regehr and J. Lepreau, “Static and Dynamic structure in design patterns”, *Proceedings of the 24th International Conference on Software Engineering (ICSE’02),* IEEE, pp. 208-218, Orlando, Florida, 19-25 May 2002

[S38] M. Elish, “Do Structural Design Patterns Promote Design Stability?”, *Proceedings of the 30th Annual International Computer Software and Applications Conference - Volume 01 (COMPSAC’06),* IEEE, pp 215-220, Chicago, Illinois, 17-21 September 2006

[S39] B. Ellis, J. Stylos and B. Myers, “The Factory Pattern in API Design: A Usability Evaluation”, *Proceedings of the 29th international conference on Software Engineering,* IEEE, pp. 302-312, Minneapolis, Minnesota, 20-26 May 2007

[S40] R. Ferenc, A. Beszedes, L. Fulop and J. Lele, “Design Pattern Mining Enhanced by Machine Learning”, *Proceedings of the 21st IEEE International Conference on Software Maintenance*, IEEE, pp. 295-304, Budapest, Hungary , 25-30 September 2005

[S41] R. B. France, D. K. Kim, S. Ghosh and E. Song, “A UML-Based Pattern Specification Technique”, *IEEE Transactions on Software Engineering,* IEEE, 30 (3), pp.193-206, March 2004.

[S42] L. J. Fulop, R. Ferenc and T. Gyimothy, “Towards a Benchmark for Evaluating Design Pattern Miner Tools”, *Proceedings of the 2008 12th European Conference on Software Maintenance and Reengineering,* IEEE, pp. 143-152, Athens, Greece, 01-04 April 2008

[S43] M. Gatrell, S. Counsell and T. Hall, “Design Patterns and Change Proneness: A Replication Using Proprietary C# Software”, *Proceedings of the 2009 16th Working Conference on Reverse Engineering,* pp. 160-164, Lille, France, 13-16 October 2009.

[S44] Y. G. Gueheneuc and G. Antoniol, “DeMIMA: A Multilayered Approach foe Design Pattern Identification”, *IEEE Transaction of Software Engineering,* IEEE, 34 (5), pp. 667-684, September 2008.

[S45] Y. G. Geuheneuc, H. Sahraoui and F. Zaidi, “Fingerprinting Design Patterns”, *Proceedings of the 11th Working Conference on Reverse Engineering,* pp. 172-181, Delft, The Netherlands, 08-12 November 2004.

[S46] J. Gustafsson, J. Paakki, L. Nenonen and A. I. Verkamo, “Architecture-Centric Software Evolution by Software Metrics and Design Patterns”, *Proceedings of the Sixth European Conference on Software Maintenance and Reengineering,* IEEE, pp. 108, Budapest, Hungary, 11-13 March 2002

[S47] N. L. Hsueh , P. H. Chu and W. Chu, “A quantitative approach for evaluating the quality of design patterns”, Journal of Systems and Software, Elsevier, 81 (8), pp. 1430-1439, August 2008.

[S48] H. Huang, S. Zhang, J. Cao, Y. Duan, “A practical pattern recovery approach based on both structural and behavioral analysis”, *Journal of Systems and Software*, Elsevier, 75 (1-2), pp. 69-87, February 2005.

[S49] B. Huston, “The effects of design pattern application on metric scores”, Journal of Systems and Software, Elsevier, 58 (3), pp.261-269, September 2001.

[S50] D. Jain and H. J. Yang, “OO Design Patterns, Design Structure, and Program Changes: An Industrial Case Study” Proceedings of the IEEE International Conference on Software Maintenance (ICSM'01), IEEE, pp. 580, Florence, Italy, 07-09 November 2003

[S51] O. Kaczor, Y. G. Gueheneuc and S. Hamel, “Identification of design motifs with pattern matching algorithms”, *Information and Software Technology,* Elsevier, 52 (2), pp.152-168, February 2010.

[S52] B. Keepence and M. Mannion, “Using Patterns to Model Variability in Produst Families”, *IEEE Software,* IEEE, 16 (4), pp. 102-108, July 1999.

[S53] R. K. Keller, R. Schauer, S. Robitaille and P. Page, “Pattern-based reverse-engineering of design components”, *Proceedings of the 21st international conference on Software engineering (ICSE’99),* IEEE, pp. 226-235, Los Angeles, California, 16-22 May 1999

[S54] N. L. Kerth and W. Cunningham, “Using Patterns to Improve Our Architectural Vision”, *IEEE Software,* 14 (1), pp. 53-59, January 1997

[S55] F. Khomh and Y. G. Gueheneuce, “Do Design Patterns Impact Software Quality Positively?**”,** *Proceedings of the 2008 12th European Conference on Software Maintenance and Reengineering,* IEEE, pp. 274-278, Athens, Greece, 01-04 April 2008

[S56] F. Khomh and Y. G. Gueheneuc, “Playing roles in design patterns: An empirical descriptive and analytic study”, *Proceedings of the 25th IEEE International Conference on Software Maintenance,* IEEE, pp 83-92, Edmonton, Canada, 20-26 September 2009

[S57] D. K. Kim, R. France, S. Ghosh and E. Song, “A Role-Based Metamodeling Approach to Specifying Design Patterns”, *Proceedings of the 27th Annual International Conference on Computer Software and Applications,* IEEE, pp. 452, Dallas, Texas, 03-06 November 2003

[S58] O. Kaczor, Y.K. Gueheneuc, and S. Hamel, “Efficient Identification of Design Patterns with Bit-vector Algorithm”, *IEEE Proceedings of the 10th Conference on Software Maintenance and Reengineering, (CSMR’06),* IEEE Computer Society, pp. 175-184, 22 – 24 March 2006.

[S59] K. Kouskouras, A. Chatzigeorgiou and G. Stephanides, “Facilitating software extension with design patterns and Aspect-Oriented Programming”, *Journal of Systems and Software,* Elsevier, 81 (10), pp 1725-1737, October 2008.

[S60] C. Kramer and L. Prechelt, “Design Recovery by Automated Search for Structural Design Patterns in Object-Oriented Software”, *Proceedings of the 3rd Working Conference on Reverse Engineering (WCRE '96),* IEEE, pp. 208, Monterey, CA, 8-10 November 1996.

[S61] S. MacDonald, D. Szafron, J. Schaeffer, J. Anvik, S. Bromling and K. Tan, “Generative Design Patterns”, *Proceedings of the 17th IEEE international conference on Automated software engineering,* ACM, pp. 23, Edinburgh, UK, 23-27 September 2002

[S62] J. K. H. Mak, C. S. T. Choy and D. P. K. Lun, “Precise Modeling of Design Patterns in UML”, *Proceedings of the 26th International Conference on Software Engineering (ICSE’04),* IEEE, pp. 252-261, Edimburg, Scotland, 23-28 May 2004

[S63] B. A. Malloy and J. F .Power, “Exploiting design patterns to automate validation of class invariants: Research articles”, *Software Testing Verification & Reliability,* Wiley Interscience, 16 (2), pp. 71-95, June 2006

[S64] W. B. McNatt and J. M. Bieman, “Coupling of Design patterns: Common Practices and Their Benefits”, *25th Annual International Computer Software and Applications Conference (COMPSAC'01),* IEEE, pp.574, Chicago, Illinois, 08-12 October 2001

[S65] S. J. Mellor and R. Johnson, “Why Explore Object Methods, Patterns and Architectures?”, *IEEE Software,* 14 (1), pp. 27-30, January 1997.

[S66] T. Mens and T. Tourwe, “A Declarative Evolution Framework for Object-Oriented Design Patterns”, *Proceedings of the IEEE International Conference on Software Maintenance (ICSM'01),* IEEE, pp. 570, Florence, Italy, 07-09 November 2003

[S67] M. Meyer, “Pattern-based Reengineering of Software Systems”, *Proceedings of the 13th Working Conference on Reverse Engineering,* pp.305-306, Benevento, Italy, 23-27 October 2006

[S68] T. Mikkonen, “Formalizing design patterns”, *Proceedings of the 20th international conference on Software engineering (ICSE’98*), IEEE, pp. 115-124, Kyoto, Japan, 19-25 April 1998

[S69] T. Muraki and M. Saeki, “Metrics for applying GOF design patterns in refactoring processes”, *Proceedings of the 4th International Workshop on Principles of Software Evolution,* IEEE, pp.27-36, Vienna, Austria, 10-11 September 2001

[S70] M. L. Nelson, “A Design Pattern for Autonomus Vihicle Software Control Architectures”, *23rd International Computer Software and Applications Conference (COMPSAC’99),* IEEE, pp 172, Phoenix, Arizona, 25-26 October 1999

[S71] T. H. Ng and S. C. Cheung, “Enhancing class commutability in the deployment of design patterns”, *Information and Software Technology,* Elsevier, 47 (12), pp.797-804, September 2005

[S72] T. H. Ng, S. C. Cheung, W. K. Chan and Y. T. Yu, “Do Maintainers Utilize Deployed Design Patterns Effectively?”, *International Conference on Software Engineering,* IEEE, pp.168-177, Minneapolis, Minnesota, 20-26 May 2007

[S73] T. H. Ng, S. C. Cheung, W. K. Chan and Y. T. Yu, “Toward effective deployment of design patterns for software extension: a case study”, *Proceedings of the 2006 international workshop on Software quality (ICSE’06*), IEEE, pp.51-56, Shanghai, China, 21 May 2006.

[S74] T. H. Ng , S. C. Cheung, W. K. Chan and Y. T. Yu, “Work experience versus refactoring to design patterns: a controlled experiment” *Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering,* ACM, pp. 12-22, Portland, Oregon, 5-11 November 2006.

[S75] S .J. Nielson and C. D. Knutson, “Design dysphasia and the pattern maintenance cycle”, *Information and Software Technology,* Elsevier, 48 (8), pp. 660-675, August 2006

[S76] J. Niere, W. Schafer, J. P. Wadsack, L. Wendehals and J. Welsh, “Towards pattern-based design recovery”, *Proceedings of the 24th International Conference on Software Engineering,,* IEEE, pp. 338-348, Orlando, Florida, 19-25 May 2002 .

[S77] N. Noda and T. Kishi, “Design pattern concerns for software evolution”, *Proceedings of the 4th International Workshop on Principles of Software Evolution,* IEEE, pp.158-161, Vienna, Austria, 10-11 September 2001

[S78] M. O Cinneide and P. Nixon, “A Methodology for the Automated Introduction of Design Patterns”, *Proceedings of the 15th IEEE International Conference on Software Maintenance,* IEEE, pp. 463, Oxford, England, 30 August – 03 September 1999

[S79] J. Palsberg and C. B. Jay, “The Essence of the Visitor Pattern”, *Proceedings of the 22nd International Computer Software and Applications Conference,* IEEE, pp. 9-15, Vienna, Austria, 17-21 August 1998.

[S80] C. Park, Y. Kang, C. Wu and K. Yi, “A Static Reference Flow Analysis to Understand Design Pattern Behavior” *11th Working Conference on Reverse Engineering (WCRE 2004),* pp. 300-301, Delft, The Netherlands, 08-12 November 2004.

[S81] N. Pettersson, “Measuring precision for static and dynamic design pattern recognition as a function of coverage”, *Proceedings of the third international workshop on Dynamic analysis (ICSE’05*), IEEE, pp. 1-7, St. Louis, Missouri, 17 May 2005

[S82] G. C. Porras and Y. G. Gueheneuc, “An Empirical Study on the Efficiency of Different Design Pattern Representations in UML Class Diagramms”, *Empirical Software Engineering*, Springer, January 2010

[S83] L. Prechelt, B.Unger-Lamprecht, M. Philipsen and W. F. Tichy, “Two Controlled Experiments Assessing the Usefulness of Design Pattern Documentation in Program Maintenance”, *IEEE Transactions on Software Engineering*, IEEE, 28 (6), pp. 595-606, June 2002.

[S84] L. Prechelt, B. Unger-Lamprecht, W .F. Tichy, P. Brossler and L. G. Votta, "A controlled experiment in maintenance comparing design patterns to simpler solutions", *IEEE Transactions on Software Engineering,* IEEE, 27 (3), pp 1134 -1144 , December 2001

[S85] J. Sametinger and M. Riebish, "Evolution Support by Homogeneously Documenting Patterns, Aspects and Traces", *Proceedings of the Sixth European Conference on Software Maintenance and Reengineering* , IEEE, pp. 134, Budapest, Hungary , 11-13 March 2002

[S86] D. C. Schmidt and F. Buschmann, "Patterns, frameworks and middleware: their synergistic relationships *Proceedings of the 25th International Conference on Software Engineering (ICSE'03),* IEEE, pp. 694-704, Portland, Oregon, 03-10 May 2003.

[S87] N. Shi and R. A. Olsson, "Reverse Engineering of Design Patterns from Java Source Code", *Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering,* ACM, pp. 123-134, Tokyo, Japan, 18-22 September 2006

[S88] N. Soundarajan and J. O. Hallstrom, "Responsibilities and Rewards: Specifying Design patterns", *Proceedings of the 26th International Conference on Software Engineering (ICSE'04),* IEEE, pp. 666-675, Edinburgh, UK, 23-28 May 2004

[S89] D. Streitferdt, C. Heller and I. Philippow, "Searching Design Patterns in Source Code", *Proceedings of the 29th Annual International Computer Software and Applications Conference - Volume 02 (COMPSAC'05),* IEEE, pp. 33-34, Edinburgh, UK, 26-28 July 2005

[S90] L. Tahvildari and K. Kontogiannis, "On the Role of Design Patterns in Quality-Driven Re-engineering", *Proceedings of the Sixth European Conference on Software Maintenance and Reengineering* , IEEE, pp. 134, Budapest, Hungary , 11-13 March 2002

[S91] T. Taibi and D .C. L. Ngo, "Formal specification of design pattern combination using BPSL", *Information and Software Technology,* Elsevier, 45 (3), pp. 157-170, March 2003

[S92] T. D. Thu and H. T. B. Tran, "A Composite Design Pattern for Object Frameworks", *31st Annual International Computer Software and Applications Conference (COMPSAC'07),* IEEE, pp.521-526, Beijing, China, 24-27 July 2007

[S93] P. Tonella and G. Antoniol, "Object Oriented Design Pattern Inference", *Proceedings of the 15th IEEE International Conference on Software Maintenance,* IEEE, pp. 230, Oxford, England, 30 August – 03 September 1999

[S94] M. Torchiano, "Documenting Pattern Use in Java Programs**",** *Proceedings of the International Conference on Software Maintenance (ICSM'02),* IEEE, pp.230, Montreal, Canada, 03-06 October 2002

[S95] W. T. Tsai, Y. Tu, W. Shao and E. Ebner, "Testing Extensible Design Patterns in Object Oriented Frameworks through Scenario Templates", *23rd International Computer Software and Applications Conference (COMPSAC'99),* IEEE, pp.166-171, Phoenix, Arizona, 25-26 October 1999.

[S96] N. Tsantalis, A. Chatzigeorgiou, G. Stephanides and S. T. Halkidis, "Design Pattern Detection Using Similarity Scoring", *IEEE Transaction of Software Engineering,* IEEE, 32 (11), pp.896-909, November 2006

[S97] M. Vokac, "Defect Frequency and Design Patterns: An Empirical Study of Industrial Code", *IEEE Transactions on Software Engineering,* IEEE, 30(12), pp. 904-917, December 2004.

[S98] M. Vokáč, W. Tichy, D. I. K. Sjøberg , E. Arisholm and M. Aldrin, "A Controlled Experiment Comparing the Maintainability of Programs Designed with and without Design Patterns—A Replication in a Real Programming Environment", *Empirical Software Engineering, Springer,* 9(3), pp 149-195, September 2004

[S99] W. Wang and V. Tzerpos, "Design Pattern Detection in Eiffel Systems", *Proceedings of the 12th Working Conference on Reverse Engineering*, pp.165-174, Pittsburgh, Pennsylvania, 07-11 November 2005

[S100] L. Wendehals and A.Orso, "Recognizing behavioural patterns at run time using finite automata", *Proceedings of the 2006 international workshop on Dynamic systems analysis (ICSE'06*), IEEE, pp.33-40, Shanghai, China, 23 May 2006

[S101] P. Wendorff, "Assessment of Design Patterns during Software Reengineering: Lessons Learned from a Large Commercial Project", Proceedings of the Fifth European Conference on Software Maintenance and Reengineering, IEEE, pp. 77, Lisbon, Portugal , 14-16 March 2001

[S102] T. Winn and P. Calder, "Is This a Pattern?" , *IEEE Software*, IEEE, 19 (1), pp. 59-66, January 2002

[S103] S. S. Yau and N. Dong, "Integration in Component-Based Software Developmant Using Design Patterns", *24th International Computer Software and Applications Conference (COMPSAC'00),* IEEE,, pp.369, Taipei, Taiwan, 25-28 October 2000

[S104] U. Zdun, P. Alexiou, C. Hentrich and S. Dustdar, "Architecting as decision making with patterns and primitives", *Proceedings of the 3rd international workshop on Sharing and reusing architectural knowledge (ICSE '08),* IEEE, pp. 11-18, Leipzig, Germany, 10-18 May 2008

[S105] U. Zdun and P. Avegriou, **"**A catalog of architectural primitives for modeling architectural patterns**",** *Information and Software Technology*, Elsevier, 50 (9-10), pp 1003-1034, August 2008

[S106] H. Zhu, I. Bayley, L. Shan and R. Amphlett, "Tool Support for Design Pattern Recognition at Model Level", *Proceedings of the 2009 33rd Annual IEEE International Computer Software and Applications Conference*, IEEE, pp. 228-233, Seattle, Washington, 20-24 July 2009

[S107] M. Ziane, "A Transformational Viewpoint on Design Patterns", Proceedings of the 15th IEEE international conference on Automated software engineering, ACM, pp. 273, Grenoble, France, 11 – 15 September 2000.

ΠΑΡΑΡΤΗΜΑ Γ – Λογισμικό που χρησιμοποιήθηκε στη μελέτη περίπτωσης

Κατηγορίες Παιχνιδιών	**Ονόματα Παιχνιδιών**
Role Playing Games	Dr. Scenario
	Stigma - The Game
	Shoddy Battle
	Fabled Lands App
	NitsLoch
	KGarten
	Herculeum
	Spice Trade
	Tyrant - Java Roguelike
	Kloben
Simulation	tXtFL
	TGame2
	XHSI - High resolution HSI for X-Plane
	CarDriving
	Dioscuri - modular emulator
	CarDriving2D
	miniTraff
	The Tao of Soccer
	Arcus - Rubik's Cube Simulator
	pittrainer
Board Games	GoGrinder
	Flash Chess & Mobile Chess
	jBubbleBreaker
	Solitaire Settlers of Catan ComputerGame
	Computer Squad Leader
	RobotChase
	Capa chess
	The Bawo Project
	FireMox
	Xoridor
Arcade	SuperSnake
	Motherload Unlimited
	Batiscafo
	Jippy Snake
	JMario
	Aspirin
	MazeRunner
	JMaze
	Yet Another Tetris Implementation
	jLodeRunner
Multi User	JCrossClient
	Passenger
	IVJ MUD
	GraphMud
	Java Mu
	JLaby - The MMORP Java Project
	Cosmos Game

	GNU MMORPG for Java
	D5.2
	The Grid
Turn Based Strategy	FreeCol
	FreeLords
	Panzer Combat II
	Domination
	Colossus
	JSettlers2 - Java Settlers of Catan
	jWrestling
	Gomoku
	Marauroa
	Clippers
Puzzle Games	Mobile Sudoku Solver
	Mazix
	Mobile Picross
	JRicochet
	Stone's Throw
	sudoku player
	Blocks Game/BrickMonkey
	Dominoes on Acid
	Java-Sudoku
	Polytope Tetris
First Person Shooter	battlephone
	Ruin
	Roboman PC Game
	Vamp Busters
	Doom77
	PiTaBOT
	The Revenge of John Sage
	Arabian Flights
	Memories of Mordor
	Codename: X-Phase
Real Time Strategy	Risk
	Summer Project gone crazy!
	FreeRails
	Rescue! Max
	Hunt for Gold
	JFlag
	Tourist Camping Tycoon
	Fate - A Space Trading and Combat Game
	Codename Alpha
	Chaotic Domain
Card Games	PokerApp
	xUNO ME
	JSkat
	Truco! o algo...
	Jokers
	JGames
	TrickTakingGame
	JMhing
	JiBi's Hold'Em
	Magic Assistant

ΠΑΡΑΡΤΗΜΑ Δ – Κατηγορικές Μεταβλητές Μελέτης Περίπτωσης

Κατηγορικές μεταβλητές

Για τη μεταβλητή (3)

0 – Role Playing Games

1 – Simulation

2 – Board Games

3 – Arcade

4 – Multi User Games

5 – Turn Based Strategy

6 – Puzzle

7 – First Person Shooter

8 – Real Time Strategy

9 – Card Games

Για τη μεταβλητή (4)

1 – Εκδόθηκε πριν το 2005

2 – Εκδόθηκε μεταξύ 2006 - 2007

3 – Εκδόθηκε μετά 2008

Για τη μεταβλητή (5)

1 – Ενεργό για λιγότερο από 100 μέρες

2 – Ενεργό για λιγότερο από 1 χρόνο

3 – Ενεργό για λιγότερο από 2 χρόνια

4 – Ενεργό για περισσότερο από 2 χρόνια

Για τη μεταβλητή (6)

1 – λιγότερες από 1.000 λήψεις

2 – λιγότερες από 2.000 λήψεις

3 – λιγότερες από 10.000 λήψεις

4 – περισσότερες από 10.000 λήψεις

Για τη μεταβλητή (8)

1 – Μία μόνο έκδοση

2 – Μεταξύ 2 – 10 εκδόσεων

3 – Περισσότερες από 10 εκδόσεις

Για τη μεταβλητή (9)

1 – 1 με 50 κλάσεις

2 – 51 με 100 κλάσεις

3 – 101 με 200 κλάσεις

4 – Περισσότερες από 201 κλάσεις

Για τις μεταβλητές (10-22)

1 – Κανένα στιγμιότυπο προτύπου

2 – 1 με 4 στιγμιότυπα προτύπων

3 – 5 με 9 στιγμιότυπα προτύπων

4 – 10 με 14 στιγμιότυπα προτύπων

5 – περισσότερα από 15 στιγμιότυπα προτύπων

Δυαδικές μεταβλητές

Για τη μεταβλητή (7)

0 – Μόνο ένας προγραμματιστής

1 – Περισσότεροι από ένας προγραμματιστές

Για τις μεταβλητές (10-22)

1 – Η εφαρμογή δεν έχει κανένα στιγμιότυπο προτύπων σχεδίασης

2 – Η εφαρμογή έχει τουλάχιστον ένα στιγμιότυπο προτύπων σχεδίασης