Πτυχιακή εργασία του φοιτητή Ρούμπου Γεώργιου

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

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

"Εμπειρική μελέτη ανασκόπησης της εφαρμογής των Α/Σ μετρικών στις βιβλιοθήκες λογισμικού (APIs)"

Του φοιτητή
Ρούμπος Γιώργος
Αρ. Μητρώου: ΑΜ 063164

Επιβλέπων καθηγητής
Ιγνάτιος Δεληγίαννης

Θεσσαλονίκη 2015

Περιεχόμενα

Περίληψη

Η παρούσα πτυχιακή έχει ως αντικείμενο την ανασκόπηση των εμπειρικών μελετών ως προς την εφαρμογή και αποτελεσματικότητα των αντικειμενοστρεφών μετρικών στην ανάπτυξη ποιοτικών προγραμμάτων βιβλιοθηκών λογισμικού (Application Program Interfaces – APIs). Για την εκπόνηση της, αναζητήθηκαν και μελετήθηκαν επιστημονικές εργασίες για το διάστημα 2010-2015 που εξετάζουν: 1) τις Α/Σ μετρικές 2) τα APIs και 3) τις εμπειρικές μελέτες που αφορούν την εφαρμογή των μετρικών στην ανάπτυξη των APIs. Στα πλαίσια του ερευνητικού μέρους, πραγματοποιήθηκε εμπειρική μελέτη ανασκόπησης της εφαρμογής των Α/Σ μετρικών στις βιβλιοθήκες λογισμικού (APIs) και εξήχθησαν συμπεράσματα και προτεινόμενες προτάσεις σχετικά την αποτελεσματικότητα των μετρικών στην ποιότητα ανάπτυξης των APIs.

Abstract

This paper is intended to review the empirical studies on the implementation and effectiveness of object-oriented metrics to develop quality software library programs (Application Program Interfaces - APIs). Within the theoretical part, researched and studied scientific work for the period 2010-2015 to consider: 1) the A / S metric 2) the APIs and 3) empirical studies on the application of measures on development of APIs. As part of the research part, a empirical study review of the implementation of the A / V metrics on software libraries (APIs) and exported conclusions and proposed recommendations on the effectiveness of measures on the quality development of APIs.

Κεφάλαιο 1

1.1 Εισαγωγή

Η πτυχιακή αυτή έχει ως αντικείμενο την ανασκόπηση των εμπειρικών μελετών ως προς την εφαρμογή και αποτελεσματικότητα των αντικειμενοστρεφών μετρικών στην ανάπτυξη ποιοτικών προγραμμάτων βιβλιοθηκών λογισμικού (Application Program Interfaces – APIs).

Για τις ανάγκες της πτυχιακής εργασίας, θα πρέπει να αναζητηθούν και να μελετηθούν επιστημονικές εργασίες που εξετάζουν: 1) Τις Α/Σ μετρικές 2) Τα APIs, και 3) Εμπειρικές μελέτες που αφορούν την εφαρμογή των μετρικών στην ανάπτυξη των APIs. Ως αναμενόμενο αποτέλεσμα είναι να εξαχθούν συμπεράσματα και προτεινόμενες προτάσεις σχετικά την αποτελεσματικότητα των μετρικών στην ποιότητα ανάπτυξης των APIs.

1.2 Περίγραμμα Πτυχιακής

Η παρούσα πτυχιακή εργασία οργανώνεται στα παρακάτω κεφάλαια:

Στο κεφάλαιο 2 γίνεται μία σύντομη αναφορά στην έννοια της ποιότητας του λογισμικού. Αναλύεται, επίσης, ο όρος της «κρίσης του λογισμικού», γίνεται αναφορά στην αξιολόγηση και ποιότητας λογισμικού και παρουσιάζονται τα βασικότερα μοντέλα ποιότητας και προτύπων βάσει των οποίων μπορεί να αναπτυχθεί ποιοτικό λογισμικό.

Στο κεφάλαιο 3 γίνεται μια πρώτη γνωριμία με τις μετρήσεις και τις μετρικές. Πιο αναλυτικά, προσδιορίζεται ο σκοπός, για τον οποίο πραγματοποιούνται οι μετρικές, γίνεται εκτενή αναφορά στις μετρικές αντικειμενοστρεφούς λογισμικού και προσδιορίζονται οι αρχές του αντικειμενοστρεφούς προγραμματισμού. Ακόμη, στο τέλος αυτού του κεφαλαίου παρουσιάζονται οι μέθοδοι διεξαγωγής μετρήσεων ποιότητας λογισμικού.

Στο κεφάλαιο 4 ορίζεται η διεπαφή προγραμματισμού εφαρμογών (API) και
αναλύονται, ώστε να «αποκρυπτογραφηθεί» η λειτουργία της δομή της. Στην συνέχεια του κεφαλαίου θα παρατεθούν παραδείγματα γνωστών APIs, θα συγκριθούν και θα προσδιοριστεί πως γίνεται η επιλογή σωστού κώδικα διεπαφής προγραμματισμού εφαρμογών.

Στο πέμπτο κεφάλαιο θα αναφερθούν τα σημαντικότερα και δευτερεύοντα εργαλεία μετρικών.

Στο έκτο κεφάλαιο θα παρουσιαστούν εμπειρικές μελέτες που αφορούν την εφαρμογή των μετρικών στην ανάπτυξη των APIs.

Κεφάλαιο 2 - Ποιότητα Λογισμικού

2.1 Ορισμός

2.1.1 Η έννοια της ποιότητας

Η έννοια ποιότητα άρχισε να απασχολεί τον άνθρωπο από τα πρώτα βήματα της εμφάνισης του και με την πάροδο του χρόνου, το εμπόριο και την ανταλλαγή προϊόντων ενδυναμώθηκε στο μυαλό των ανθρώπων. Η έννοια της, σήμερα έχει την προέλευσή της κυρίως από το χώρο της οικονομίας και συνδέεται με το στόχο της αύξησης της αποτελεσματικότητας των παραγωγικών μονάδων ή των προσφερόμενων υπηρεσιών.

Η «ποιότητα» είναι ένας όρος πολυδιάστατος και δεν υπάρχει παγκόσμια κοινά αποδεκτός ορισμός για αυτήν, αφού ο κάθε υποψήφιος «πελάτης» έχει ένα διαφορετικό χαρακτηριστικό του προϊόντος / υπηρεσίας το οποία αποκαλεί «ποιότητα». Παρ' όλα αυτά, το κοινό σημείο όλων των ορισμών είναι ότι η ποιότητα συνεπάγεται την ικανοποίηση των αναγκών του πελάτη (Ζαβλανός, 2003).

Ο Διεθνής Οργανισμός Τυποποίησης ISO[1] έχει υιοθετήσει μια δική του ερμηνεία για
την ποιότητα η οποία αναφέρει ως «ποιότητα είναι η ολικότητα των στοιχείων και χαρακτηριστικών ενός προϊόντος ή υπηρεσίας για την εξυπηρέτηση μιας δοσμένης ανάγκης».

Συνοπτικά η ποιότητα έχει οριστεί ως:

[1] Ο Διεθνής Οργανισμός Τυποποίησης (αγγλ. International Organization for Standardization, διακριτική ονομασία: ISO), είναι μια διεθνής οργάνωση δημιουργίας και έκδοσης προτύπων που αποτελείται από αντιπροσώπους των εθνικών οργανισμών τυποποίησης. Ο οργανισμός ιδρύθηκε στις 23 Φεβρουαρίου του 1947 και παράγει τα παγκόσμια βιομηχανικά και εμπορικά πρότυπα, τα επονομαζόμενα πρότυπα ISO. http://www.iso.org/iso/home.html

1. Συμμόρφωση στις προσδοκίες των ανθρώπων
2. Συμμόρφωση με τις απαιτήσεις των ανθρώπων
3. Αποφυγή της ζημιάς
4. Ανταπόκριση στις προσδοκίες των πελατών
5. Υπεροχή και προστιθέμενη αξία.

Οι δύο τομείς που θεωρούνται πρωτοπόροι είναι ο στρατιωτικός και ο διαστημικός, δραστηριοποιούμενοι στα συστήματα διασφάλισης ποιότητας. Το 1987 ο Διεθνής Οργανισμός Τυποποίησης (I.S.O - International Standardisation Organization) εξέδωσε πρότυπα που σχετίζονται με τη διασφάλιση της ποιότητας δηλαδή τη σειρά ISO 9000[2]. Το επόμενο στάδιο, το οποίο έχει υιοθετηθεί από πολλές επιχειρήσεις και οργανισμούς είναι «Η Ολική Ποιότητα» και η «Διοίκηση Ολικής Ποιότητας» είναι το σύνολο των δραστηριοτήτων που υπόσχεται ποιοτικά προϊόντα με ταυτόχρονη μείωση κόστους, πλήρη αξιοποίηση και ανάπτυξη του διαθέσιμου προσωπικού, εφαρμογή καινοτομιών, συνεχή βελτίωση και πλήρη συμμετοχή στην προσπάθεια όλων των επιπέδων των εργαζομένων (Γεωργίου-Τσιότρα, 2002).

2.1.2 Ποιότητα Λογισμικού

Η ποιότητα του λογισμικού αποτελεί σήμερα ένα πολύ σημαντικό και ενδιαφέρον κεφάλαιο στη επιστήμη των υπολογιστών. Με το πέρασμα του χρόνου και με την εξέλιξη της τεχνολογίας η ανάγκη για ποιοτικότερα προϊόντα και τώρα πλέον, η ανάγκη για εξασφάλιση της ποιότητας γίνονται όλο και μεγαλύτερες και αποτελούν βασικές επιδιώξεις επιχειρήσεων, οργανισμών και προγραμματιστών (Μικρόπουλος, 2000).

Το λογισμικό είναι σε γενικά πλαίσια κι αυτό ένα παραγόμενο αγαθό, που διέπεται από τους ίδιους κανόνες και ελέγχους ποιότητας. Η δυσκολία στον ορισμό της ποιότητας έγκειται στο γεγονός ότι η ποιότητα του λογισμικού είναι μια πολυδιάστατη έννοια, η οποία συμπεριλαμβάνει το αντικείμενο του

[2] http://www.iso.org/iso/home/standards/management-standards/iso_9000.htm

ενδιαφέροντος, την οπτική γωνία προσέγγισης καθώς και τα χαρακτηριστικά γνωρίσματα που εξετάζονται. Το ISO/IEC 9126-1[3] ορίζει την ποιότητα λογισμικού ως: «Η ικανότητα ενός προϊόντος λογισμικού να επιτρέπει σε συγκεκριμένους επισκέπτες να επιτύχουν συγκεκριμένους στόχους με αποδοτικότητα, παραγωγικότητα, ασφάλεια και ικανοποίηση σε συγκεκριμένα περιβάλλοντα χρήσης του εν λόγω προϊόντος».

Πιο αναλυτικά, το παραπάνω πρότυπο στοχεύει στη διασφάλιση της ποιότητας των προϊόντων του λογισμικού, καθορίζοντας τα χαρακτηριστικά και τα υπό-χαρακτηριστικά της ποιότητας καθώς και τις συνδεδεμένες μετρικές:

- ✓ **Εξωτερικές μετρικές:** εφαρμόζονται στο λογισμικό όταν είναι σε λειτουργία.
- ✓ **Εσωτερικές μετρικές:** δεν στηρίζονται στην λειτουργία του λογισμικού, είναι συνήθως μετρικές κώδικα (στατικές μετρικές).
- ✓ **Μετρικές σε χρήση:** είναι διαθέσιμες όταν το λογισμικό χρησιμοποιείται σε πραγματικές συνθήκες.

Στην παρακάτω εικόνα φαίνεται το μοντέλο Εξωτερικής/Εσωτερικής Ποιότητας του προτύπου ISO/IEC 9126-1.

[3] http://en.wikipedia.org/wiki/ISO/IEC_9126

Εικόνα 2.1. Η Διαχείριση Ποιότητας Λογισμικού σύμφωνα με το πρότυπο ISO/IEC 9126-1. Διαθέσιμη στο : http://sce.umkc.edu/BIT/burrise/pl/software_quality_management/

Παρόλο λοιπόν που οι βασικές αρχές της ποιότητας λογισμικού παραμένουν ίδιες με αυτές του κλασσικού προϊόντος, υπάρχουν βασικές διαφορές στον τρόπο προσέγγισής της στο λογισμικό από την αντίστοιχη στην παραγωγή του κλασσικού υλικού προϊόντος.

Η ποιότητα του λογισμικού και η εξασφάλιση της ποιότητας είναι άμεσα συνδεδεμένη με τις μετρήσεις, ενώ σαν μέτρηση ορίζεται η διαδικασία με την οποία αριθμοί ή σύμβολα αντιστοιχίζονται σε ιδιότητες οντοτήτων του πραγματικού κόσμου έτσι ώστε να τις περιγράφουν σύμφωνα με καθορισμένους κανόνες. Δεν μετράμε το λογισμικό, αλλά ιδιότητες του λογισμικού. Η εξασφάλιση ποιότητας και η χρήση μετρήσεων έδωσε μεγάλη ώθηση στη βιομηχανία τις τελευταίες δεκαετίες.

Οι βασικές λειτουργίες του προγράμματος εξασφάλισης ποιότητας στη μαζική παραγωγή υλικών προϊόντων είναι ο καθορισμός των χαρακτηριστικών που θα μετρηθούν και των επιθυμητών ορίων, ο καθορισμός των διαδικασιών μέτρησης, ο εντοπισμός και η απόρριψη των προϊόντων που δεν ικανοποιούν τις ποιοτικές προδιαγραφές και η βελτίωση της διαδικασίας παραγωγής ώστε να ελαχιστοποιηθεί ο αριθμός των προϊόντων που απορρίπτονται (Κόμης, 2005).

Σαν εσωτερικά χαρακτηριστικά του λογισμικού ορίζονται τα χαρακτηριστικά για τα οποία υπάρχει απτή και φυσική αντίληψη και άμεσος τρόπος μέτρησης τους (π.χ. γραμμές τεκμηρίωσης). Αυτά είναι χαρακτηριστικά τα οποία μετρώνται εύκολα, αλλά δεν προσφέρουν πληροφορίες υψηλού επιπέδου που σχετίζονται με την ποιότητα του λογισμικού. Σαν εξωτερικά χαρακτηριστικά ορίζονται τα χαρακτηριστικά με υψηλό επίπεδο αφαίρεσης, τα οποία συνθέτουν την ποιότητα του λογισμικού (π.χ. μεταφερσιμότητα). Τα χαρακτηριστικά αυτά είναι δύσκολο έως αδύνατο να μετρηθούν άμεσα.

Αντίστοιχα με τα εσωτερικά και εξωτερικά χαρακτηριστικά ορίζονται και οι εσωτερικές και εξωτερικές μετρικές μέσω των οποίων 'μετρούνται' τα χαρακτηριστικά. Οι εξωτερικές μετρικές επειδή δεν μπορούν να βασιστούν σε μετρήσεις φυσικών ποσοτήτων, βασίζονται σε έρευνες της γνώμης των «πελατών», βασισμένες κυρίως σε ερωτηματολόγια και συνεντεύξεις. Επειδή λοιπόν η έννοια της ποιότητας λογισμικού είναι αρκετά αφηρημένη, διασπάστηκε σε επιμέρους χαρακτηριστικά (παράγοντες ποιότητας λογισμικού) και δημιουργήθηκαν κάποια μοντέλα, τα οποία περιγράφουν την συσχέτιση αυτών, τρία από τα σημαντικότερα μοντέλα ποιότητας λογισμικού που χρησιμοποιούνται μέχρι σήμερα είναι το μοντέλο του McCall (ή αλλιώς μοντέλο FCM), το μοντέλο του Boehm και το πρότυπο ISO 9126-1.

2.2. Ιστορική αναδρομή

Η ποιότητα λογισμικού και γενικά, η ποιότητα είναι ένα χαρακτηριστικό, το οποίο έχει απασχολήσει τις οργανωμένες κοινωνίες από την αρχαιότητα έως σήμερα, όταν υπάρχει υψηλή πολιτισμική στάθμη και συνοδεύεται πάντα από μία αναπτυγμένη τεχνολογία. Και, το πιο σημαντικό, η τελευταία λειτουργεί στη βάση ενός μηχανισμού που διασφαλίζει τον έλεγχο της ποιότητας και την προστασία του καταναλωτή.

Υπάρχουν πολλά παραδείγματα, τα οποία χρονολογούνται από την αρχαία Βαβυλώνα, την αρχαία Ρώμη, μέχρι και σήμερα. Ενδεικτικά, στην

αρχαία Βαβυλώνα, στα πλαίσια της ποιότητας, ανάμεσα στους νόμους του Hammurabi υπάρχει και ο αρχαιότερος κανονισμός σχετικά με την οικοδομή, όπου αν η οικοδομή δεν κτιστεί σύμφωνα με τους ισχύοντες κανονισμούς και πρότυπα, τότε ο οικοδόμος έπρεπε να συμπληρώσει εκείνος τα επιπλέον έξοδα, σε περίπτωση που χρειάζονταν κάποια αλλαγή. Παρόμοιος έλεγχος ποιότητας υπήρχε και στην Αρχαία Αίγυπτο.

Δε, στην κλασική Ελλάδα, υπήρχε ένας μηχανισμός ελέγχου ποιότητας, τυποποίησης και πιστοποίησης των παραγόμενων και προσφερόμενων προϊόντων στον τόπο παραγωγής, αλλά και στην αγορά. Οι αρχαίοι Έλληνες εφάρμοζαν πρότυπα με πολύ αυστηρές προδιαγραφές που κάλυπταν όλο το φάσμα των τότε παραγόμενων προϊόντων, όπως τα μέταλλα και τα κράματά τους, τα γεωργικά προϊόντα, τα τρόφιμα και τα ποτά, αλλά και οι κατασκευές, χαρακτηριστικό παράδειγμα, ο Παρθενώνας. Στην Αθηναϊκή Πολιτεία ορίζονταν με κλήρο δέκα μετρονόμοι που ευθύνονταν για όλα τα μέτρα και τα σταθμά της αγοράς και όφειλαν να μεριμνούν ώστε οι πωλητές να τα χρησιμοποιούν σωστά. Τέλος, η ποιότητα έπαιζε σημαντικό ρόλο και στην αρχαία Ρώμη.

2.3 Κρίση του λογισμικού

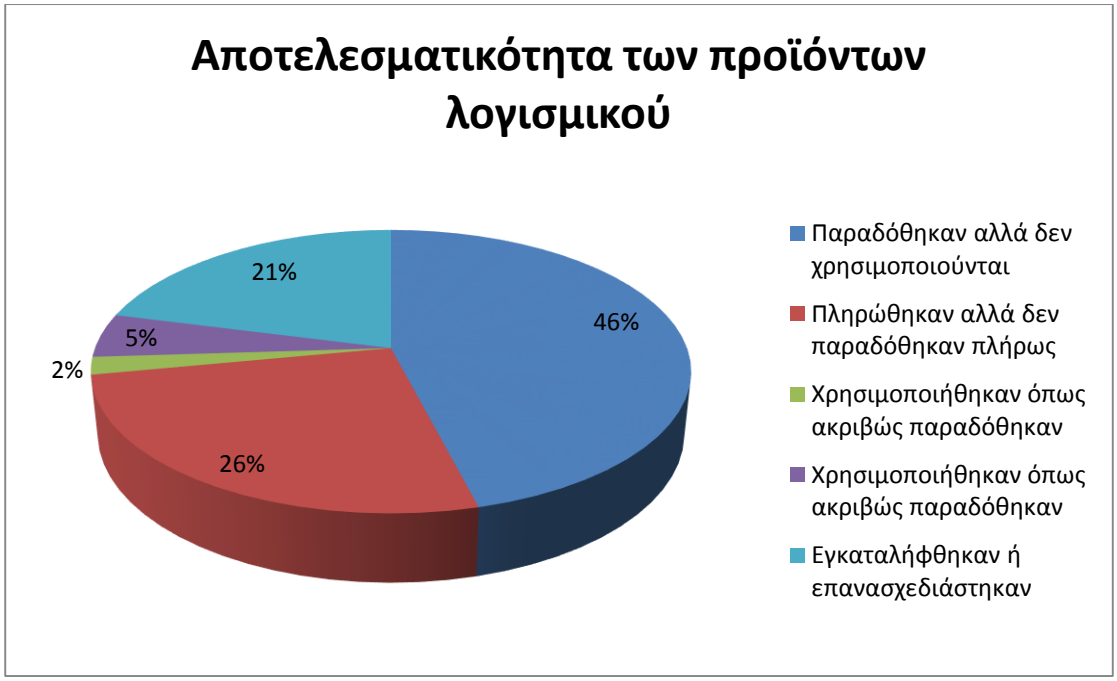
Στην λεγόμενη «κρίση του λογισμικού», η δυσκολία καθορισμού μετρήσιμων στόχων και διαδικασιών μέτρησης αποτέλεσε μείζον πρόβλημα στην εξασφάλιση ποιότητα λογισμικού. Ενώ ήταν σχετικό απλό να γίνει καθορισμός των επιθυμητών τιμών κατά την παραγωγή και σχεδίαση των υλικού – αγαθού, πχ σε ένα σκεύος, μπορούμε εύκολα να καθορίσουμε το ύφος, το πάχος και την διάμετρο βάσης, κάτι αντίστοιχο δεν είναι άμεσα εφαρμόσιμο στην περίπτωση προϊόντων λογισμικού.

Το προαναφερθέν πρόβλημα έχει ως αποτέλεσμα ότι δεν μπορούμε να έχουμε λογισμικό με υψηλή αξιοπιστία ή υψηλή λειτουργικότητα. Όπως αναφέρει και ο Gustafsson το 1993, «Όταν ένας εργάτης που δουλεύει σε μία γραμμή μαζικής παραγωγής κάνει ένα λάθος, τότε πετάει το εξάρτημα που έφτιαχνε. Αν το κάνει συχνά, σε λίγο ένας επιστάτης θα θέλει να μάθει γιατί η

στοίβα με τα πεταμένα εξαρτήματα είναι τόσο μεγάλη. Αυτό συμβαίνει συνέχεια με το λογισμικό, αλλά κανένας δεν μπορεί να δει τη στοίβα με τα πεταμένα εξαρτήματα».

Το υλικό σχετίστηκε αρκετά άμεσα με το λογισμικό καθώς ορθώς επικράτησε η άποψη ότι το υλικό δεν μπορεί να σταθεί χωρίς την υποστήριξη του λογισμικού. Αρχικά το λογισμικό αντιμετωπίζονταν ως μια ιδέα και δημιουργούνταν ανεξάρτητα από την φύση και τις απαιτήσεις του υλικού (Ξένος Μ. και Χριστοδουλάκης Δ, 1994). Έτσι δημιουργήθηκε η «κρίση του λογισμικού» καθώς τα περισσότερα λογισμικά δεν είχαν ουδεμία απήχηση και εφαρμοσιμότητα λόγω της έλλειψης της πληροφορικής (Sommerville I, 1989). Η λεγόμενη κρίση προέκυψε επίσης λόγω ότι, παράλληλα με την αυξανόμενη απαίτηση για καινούργια λογισμικά, οι εταιρίες παραγωγής λογισμικού δεν μπορούσαν να τηρήσουν τον αρχικό προϋπολογισμό στην υλοποίηση ενός λογισμικού είτε μη ικανοποίησης των απαιτήσεων του πελάτη.

Έρευνες που διεξήχθησαν έδειξαν ότι ένα πολύ μικρό ποσοστό, της τάξης του 2%, χρησιμοποιήθηκε όπως ακριβώς παραδόθηκε στον πελάτη (Bell, et al., 1992). Στα αποτελέσματα της έρευνας που φαίνεται παρακάτω, παρατηρούμε εξίσου ότι το ποσοστό των προϊόντων λογισμικού που τελικώς χρησιμοποιήθηκαν ύστερα από ορισμένες αλλαγές είναι επίσης πολύ μικρό (περίπου 5%).



Εικόνα 2.2.Αποτελεσματικότητα των προϊόντων λογισμικού

Με την πάροδο των ετών, η κρίση του λογισμικού παρέμενε και χαρακτηρίστηκε ως «χρόνια πάθηση» (Pressman, 1997). Τα λάθη και πολλές φορές η έλλειψη μίας συγκεκριμένης μεθοδολογίας κατά τη διάρκεια όλων των φάσεων του κύκλου ζωής λογισμικού έχουν ως αποτέλεσμα να απαιτείται μία τεράστια προσπάθεια (αναλογικά με τις άλλες φάσεις) για διορθωτική περιοδική συντήρηση. Η κρίση του λογισμικού φαινόταν ότι περιορίζεται με την πρόταση του DeMacro ο οποίος, πρότεινε στις εταιρίες λογισμικού να i) να υλοποιούν λιγότερο κώδικα, ii) να επιλέγουν πολύ προσεκτικότερα τι ακριβώς υλοποιούν και iii) να θέτουν ελαστικότερα όρια ολοκλήρωσης και παράδοσης των έργων τους (DeMacro, 1999).

Σήμερα, η κρίση του λογισμικού έχει αντιμετωπιστεί, καθώς δημιουργήθηκαν αρκετά πρότυπα, τα οποία καθορίζουν πως πρέπει να εφαρμοστεί ένα πρόγραμμα εξασφάλισης ποιότητας στη διαδικασία παραγωγής. Τα προαναφερθέντα πρότυπα είναι:

- ISO9000[4] ((ISO/IEC, 2000))
- Πρότυπα IEEE[5] ((IEEE, 1989))

[4] http://www.iso.org/iso/iso_9000

- Βραβεία Baldrige[6] ((Brown, 1991))
- Capability Maturity Model (CMM)[7]
- Capability Maturity Model Integration (CMMI)[8] ((Ahern, et al., 2004))

Έτσι εμφανίζονται οι μετρήσεις των λογισμικών χωρίς όμως να προτείνουν συγκεκριμένες λύσεις για τις μετρήσεις. Στην συνέχεια, με την εμφάνιση των μετρικών του λογισμικού, η ποιότητα του λογισμικού τέθηκε σε σωστότερες βάσεις. Βέβαια, το πρόβλημα που ανέκυψε είναι ότι για τα χαρακτηριστικά που επιθυμούμε να μετρήσουμε (και ενδιαφέρουν άμεσα το χρήστη) δεν υπάρχει μία δυνατότητα άμεσης μέτρησης.

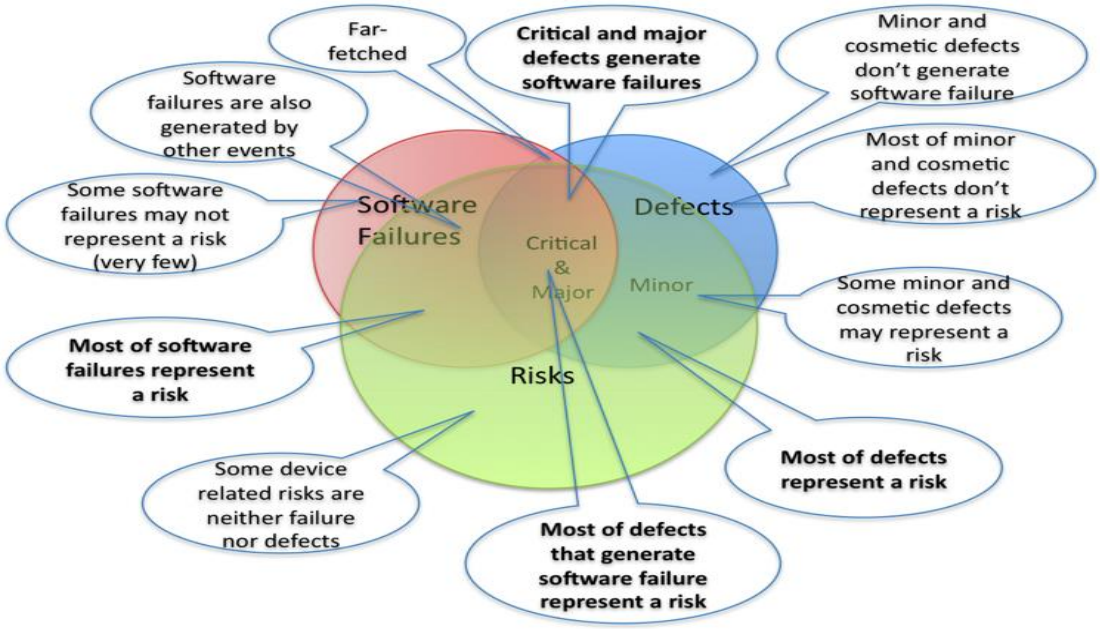
2.4 Αξιολόγηση και διασφάλιση ποιότητας λογισμικού

Όπως προαναφέραμε παραπάνω, η αξιολόγηση και η διασφάλιση ποιότητας του λογισμικού είναι ιδιαίτερα σημαντική καθώς τα λογισμικά των υπολογιστών γίνονται καθημερινά περισσότερα και πιο απαιτητικά σε υπολογιστική ισχύ και είναι οφθαλμοφανές ότι, πλέον το παραγόμενο λογισμικό κρίνεται με βάση την αξιοπιστίας του. Με άλλα λόγια μπορούμε να πούμε ότι η αξιοπιστία είναι ένα μέτρο, βάσει του οποίου προσδιορίζεται κατά πόσο το λογισμικό ανταποκρίνεται στις προδιαγραφές και στις απαιτήσεις του χρήστη. Μετρικές που έχουν αναπτυχτεί για την αξιοπιστία του λογισμικού, όπως ο «μέσος χρόνος μεταξύ των βλαβών», μπορούν να χρησιμοποιηθούν αλλά δεν συμπεριλαμβάνουν την επικείμενη φύση των αποτυχιών του λογισμικού.

Η αξιοπιστία του λογισμικό αποτελεί ένα δυναμικό χαρακτηριστικό του συστήματος, το οποίο έχει πληθώρα παραμέτρων. Μια βασική παράμετρος είναι ο αριθμός αποτυχιών του λογισμικού. Αποτυχία λογισμικού (Software

[5] http://standards.ieee.org/findstds/standard/730-2014.html
[6] http://en.wikipedia.org/wiki/Malcolm_Baldrige_National_Quality_Award
[7] http://en.wikipedia.org/wiki/Capability_Maturity_Model
[8] http://en.wikipedia.org/wiki/Capability_Maturity_Model_Integration

Failure) χαρακτηρίζεται ένα εκτελούμενο γεγονός, όπου το λογισμικό συμπεριφέρεται με μη αναμενόμενο τρόπο. Τα περισσότερα σφάλματα λογισμικού προέρχονται από ανθρώπινα λάθη ή σφάλματα που γίνονται είτε στον πηγαίο κώδικα είτε στον σχεδιασμό / αρχιτεκτονική του προγράμματος, και μερικά προέρχονται από την εσφαλμένη παραγωγή κώδικα από έναν μεταγλωττιστή. Αναφορές που λεπτομερώς καταγράφουν τα σφάλματα σε ένα πρόγραμμα αποκαλούνται συνήθως αναφορές «bugs», αναφορές σφαλμάτων, αναφορές προβλημάτων, αναφορές αλλαγών, και τα λοιπά.[9]



Εικόνα 2.3.Διαφοροποίηση και λόγοι εμφάνισης αποτυχιών λογισμικού, ελαττωμάτων λογισμικού, ρίσκων και άλλα. Διαθέσιμο στο: http://blog.cm-dm.com/

Μια άλλη παράμετρος, που διαφέρει «στα σημεία» με την αποτυχία λογισμικού είναι το ελάττωμα λογισμικού (Software Fault), το οποίο αποτελεί ένα στατικό χαρακτηριστικό του συστήματος. Τα ελαττώματα λογισμικού προκαλούνται όταν εκτελείται το κομμάτι του ελαττωματικού κώδικα με ένα συγκεκριμένο σύνολο εισόδων. Η βασική προγραμματιστική διαφορά μεταξύ των δύο προαναφερθέντων παραμέτρων είναι ότι η αποτυχία λογισμικού συμβαίνει όταν το λογισμικό δεν εκτελείται όπως ο χρήστης αναμένει ενώ το ελάττωμα λογισμικού είναι ένα κρυμμένο προγραμματιστικό λάθος. Το ελάττωμα λογισμικού καταλήγει σε αποτυχία λογισμικού μόνο όταν οι ακριβείς συνθήκες υπολογισμού πληρούνται και το ελαττωματικό κομμάτι του κώδικα

[9] http://en.wikipedia.org/wiki/Software_bug

εκτελείται στον επεξεργαστή και συνήθως συμβαίνει σε κανονική λειτουργία. Βέβαια, κάποιες φορές, τα ελαττώματα λογισμικού συμβαίνουν όταν το λογισμικό μεταφέρεται σε διαφορετική πλατφόρμα ή σε διαφορετικό μεταγλωττιστή[10].

Τα ελαττώματα λογισμικού δεν αποτελούν μόνο ελαττώματα εφαρμογής. Μη αναμενόμενη συμπεριφορά μπορεί να συμβεί σε περιπτώσεις όπου το λογισμικό προσαρμόζεται στις απαιτήσεις του, αλλά οι απαιτήσεις και οι προδιαγραφές από μόνες τους δεν είναι ολοκληρωμένες. Παραλείψεις στην τεκμηρίωση λογισμικού (Software Documentation[11]) μπορεί να οδηγήσουν σε αναμενόμενη συμπεριφορά και πολλές φορές αυτό συμβαίνει χωρίς να περιέχονται ελαττωματικά κομμάτια κώδικα.

Ο Littlewood (1990) αποδεικνύει ότι δεν υπάρχει αποδεδειγμένο εφαρμόσιμο μοντέλο ανάπτυξης αξιοπιστίας και εναλλακτικά προτείνει ότι οι προβλέψεις πρέπει να βασίζονται στο ταίριασμα παρατηρημένων δεδομένων σε διάφορα μοντέλα ανάπτυξης.

2.5 Μοντέλα ποιότητας λογισμικού και πρότυπα

Στην προσπάθεια να οριστεί και να αναγνωριστεί η ποιότητα στο λογισμικό ορίστηκαν διάφορα μοντέλα. Τα μοντέλα αυτά αναλύουν την ποιότητα στους επιμέρους παράγοντες που την συνθέτουν – στα εξόν συνετέθη – ώστε να γίνεται εύκολα αντιληπτή και φυσικά μετρήσιμη.

Οι παράγοντες ποιότητας είναι οι παρακάτω (Cavano J., McCall J.; 1978.):

- **Functionality (λειτουργικότητα).** Το λογισμικό παρέχει λειτουργίες που ικανοποιούν τις απαιτήσεις του χρήστη.

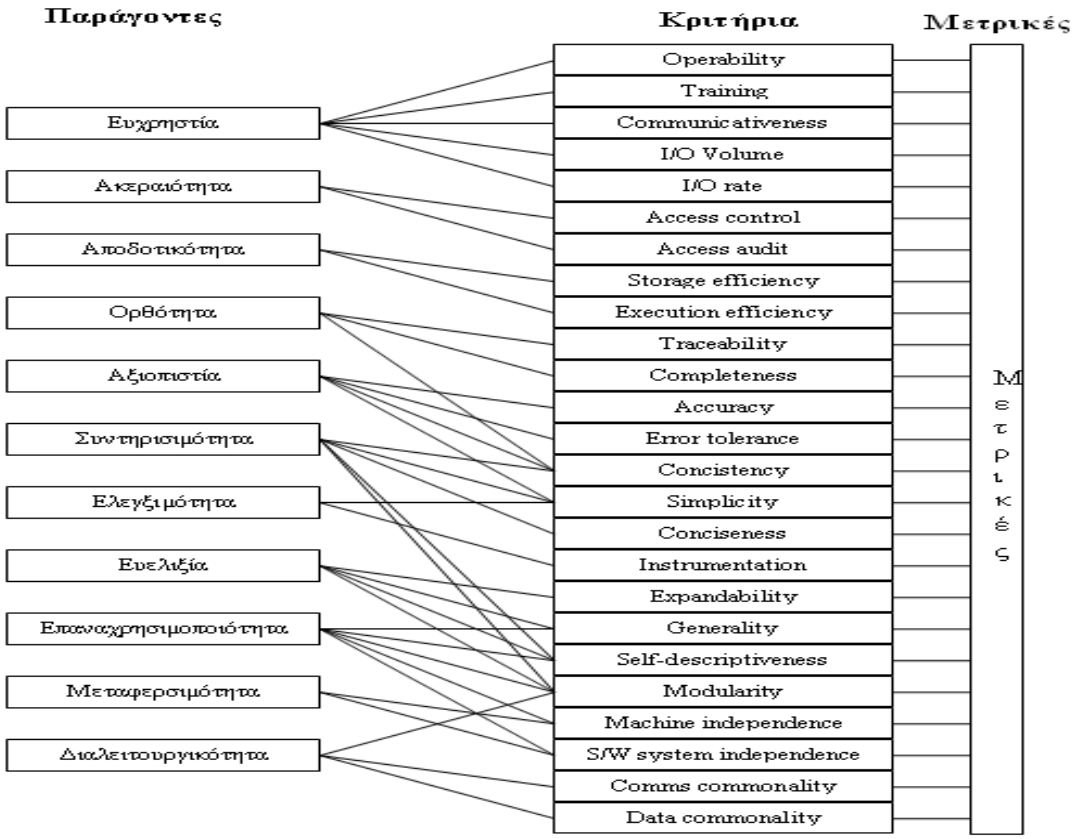
[10] http://www.robdavispe.com/free2/What-02102-is-the-difference-between-software-fault-and-software-failure.html
[11] http://en.wikipedia.org/wiki/Software_documentation

- **Usability (ευχρηστία).** Το λογισμικό παρέχει τις λειτουργίες του με τρόπο αποδοτικό και αποτελεσματικό διευκολύνοντας τους τελικούς χρήστες.
- **Correctness (ορθότητα)**. Το λογισμικό ικανοποιεί τις απαιτήσεις και καλύπτει τους αντικειμενικούς στόχους του χρήστη.
- **Integrity (ακεραιότητα).** Το λογισμικό προφυλάσσει τα δεδομένα και την λειτουργικότητα από μη εξουσιοδοτημένους χρήστες.
- **Maintainability (συντηρησιμότητα).** Το λογισμικό είναι κατά τέτοιο τρόπο σχεδιασμένο και υλοποιημένο ώστε να είναι δυνατή η διόρθωση, η τροποποίηση ή η προσθήκη λειτουργικών ή αρχιτεκτονικών χαρακτηριστικών του λογισμικού.
- **Testability (ελεγξιμότητα).** Το λογισμικό επιδέχεται ελέγχων και συγκρίσεων σχετικά με την λειτουργία του και τα αποτελέσματα των εργασιών του.
- **Reusability (επαναχρησιμοποίηση).** Αφορά την απαιτούμενη προσπάθεια ώστε να γίνει δυνατή η επαναχρησιμοποίηση μέρους ή τμήματος του λογισμικού.
- **Portability (μεταφερσιμότητα).** Το λογισμικό είναι δυνατόν να μεταφερθεί από ένα περιβάλλον σε άλλο.
- **Efficiency (αποδοτικότητα).** Το λογισμικό διατηρεί υψηλά επίπεδα επεξεργασίας και απόδοσης κάτω από συγκεκριμένες συνθήκες.
- **Reliability (αξιοπιστία).** Το λογισμικό παράγει αξιόπιστα αποτελέσματα.

Τα σημαντικότερα πρότυπα που επικράτησαν, όπως αναφέραμε και παρακάνω, τα οποία χαρακτηρίστηκαν ως τα πιο πληρέστατα είναι το μοντέλο του McCall (ή μοντέλο FCM), το μοντέλο του Boehm και το πρότυπο ISO 9126.

2.5.1 Το μοντέλο του McCall (ή μοντέλο FCM)

Ο McCall [Mcc77] προτείνει την τμηματοποίηση της ποιότητας σε παράγοντες ποιότητας. Επειδή τόσο η ίδια η ποιότητα, όσο και οι ποιοτικοί παράγοντες είναι εξαιρετικά αφηρημένες έννοιες, ο McCall πρότεινε επίσης την τμηματοποίηση των παραγόντων σε κριτήρια (criteria) που βρίσκονται σε χαμηλότερο επίπεδο αφαίρεσης και τα οποία μπορούν να μετρηθούν άμεσα με μετρικές (metrics). Ο McCall είχε προτείνει οι μετρήσεις για κάθε κριτήριο να προκύπτουν από απαντήσεις σε ερωτήσεις για το κριτήριο. Από τα ονόματα των τριών επιπέδων αφαίρεσης το μοντέλο αυτό ονομάστηκε FCM (Factors Criteria Metrics).



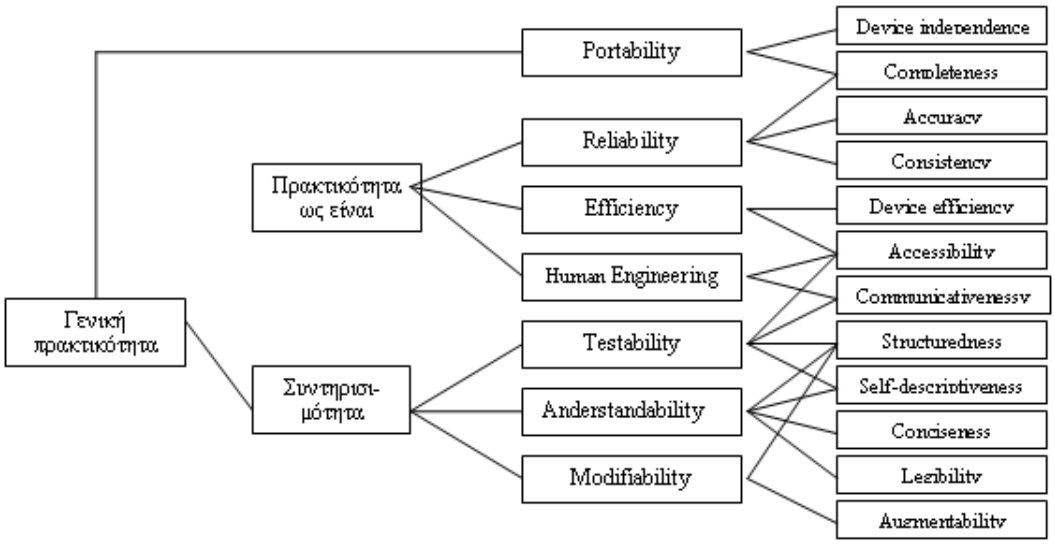
Εικόνα 2.4. Μοντέλο FCM

Το μοντέλο αυτό αποτέλεσε ένα από τα πιο ολοκληρωμένα μοντέλα της εποχής του κι έγινε βάση για το διεθνές πρότυπο ISO 9126. Παρά τα προβλήματα της υποκειμενικότητας των ερωτήσεων, της ύπαρξης περιορισμένης κλίμακας (το μοντέλο δεχόταν μόνο απαντήσεις «ΝΑΙ» και «ΟΧΙ») και της αδυναμίας συνδυασμού μετρικών, το μοντέλο αυτό γνώρισε ευρεία υποδοχή και ακόμη και σήμερα που το ISO 9126 κυριαρχεί, αρκετές επιχειρήσεις βασίζουν το σύστημα ποιότητάς τους στην τμηματοποίηση του

FCM μοντέλου. Ο βασικότερος λόγος για αυτό είναι ότι είναι καλύτερα προσαρμοσμένο στην ομάδα υλοποίησης και πιο αναλυτικό από το ISO 9126.

2.5.2 Μοντέλο Boehm

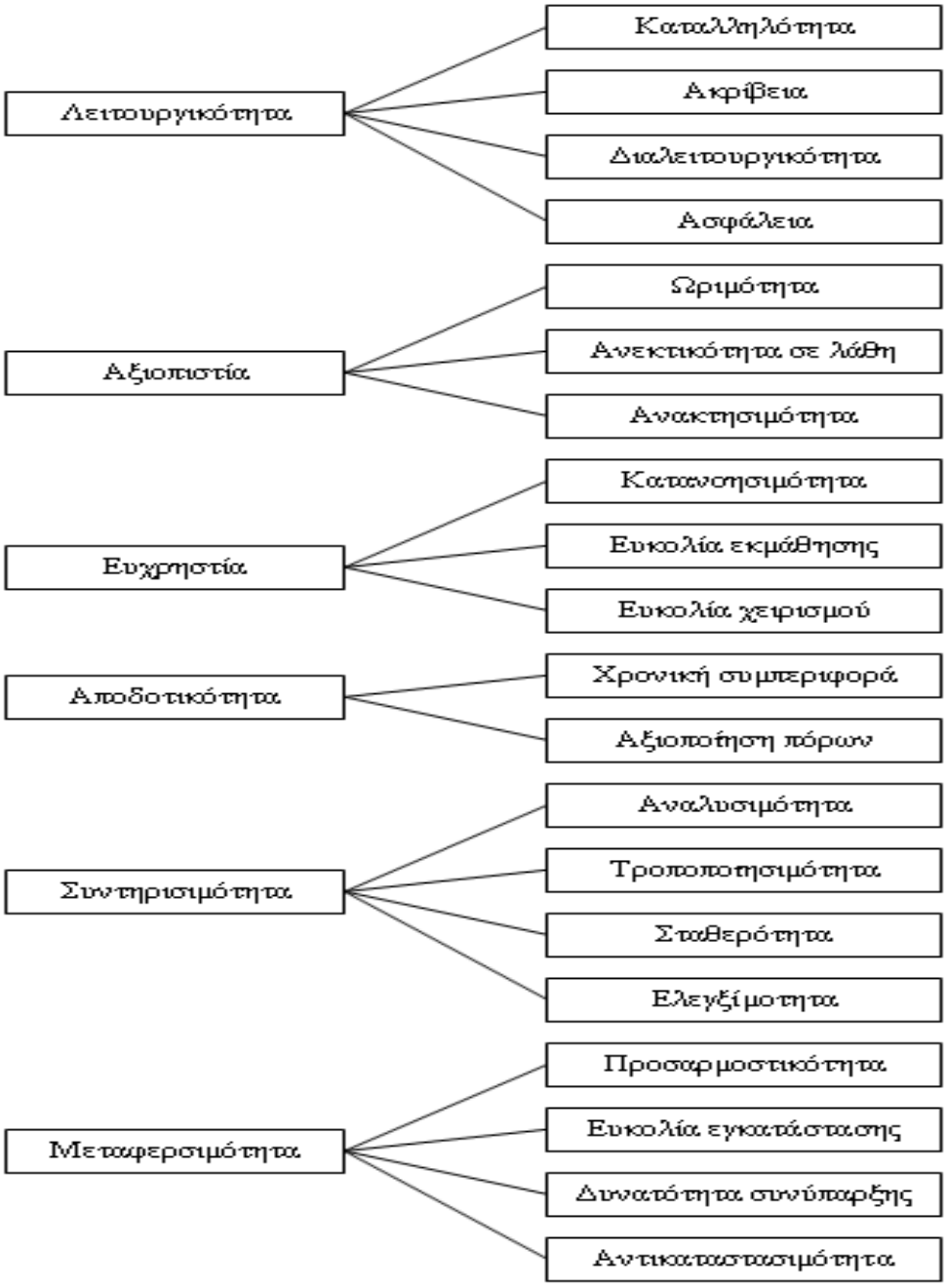
Το μοντέλο του Boehm, που έχει παρόμοια ιεραρχική δομή με το FCM μοντέλου, σύμφωνα με την οποία διασπά την ποιότητα του λογισμικού σε πρωταρχικές χρήσεις (primary uses) και αυτές σε ενδιάμεσες κατασκευές (intermediate constructs), ανάλογες με τα κριτήρια ποιότητας του μοντέλου FCM, όπως φαίνεται στο εικόνα 4. Οι ενδιάμεσες κατασκευές με τη σειρά τους διασπώνται σε πρωτογενείς κατασκευές (primitive constructs) οι οποίες μετρώνται άμεσα με μετρικές (metrics). (Boehm, 1978)



Εικόνα 2.5.Μοντέλο του Boehm

2.5.3 Πρότυπο ISO 9126

Το πρότυπο ISO 9126 αποτελεί ένα μοντέλο ποιότητας λογισμικού που εξελίχθηκε σε διεθνές πρότυπο από το διεθνή οργανισμό τυποποίησης ISO. Ως μοντέλο ποιότητας λογισμικού διαφέρει από τα προγενέστερα μοντέλα τόσο στην ορολογία όσο και στη δομή καθώς είναι απόλυτα ιεραρχικό.



Εικόνα 2.6.Το πρότυπο ISO 9126

Στο ISO 9126 κάθε χαρακτηριστικό ανήκει αυστηρά σ' ένα παράγοντα ποιότητας χωρίς να υπάρχουν επικαλύψεις. Επίσης, κάθε χαρακτηριστικό είναι ορατό στο χρήστη και δεν αποτελεί εσωτερικό χαρακτηριστικό του προϊόντος. Με αυτή τη λογική, το ISO 9126 αντικατοπτρίζει περισσότερο τη θεώρηση από την πλευρά του χρήστη, σε αντίθεση με τα δύο προγενέστερα μοντέλα που αντικατοπτρίζουν την προσέγγιση της ομάδας ανάπτυξης.

Αυτός είναι και ο βασικός λόγος για τον οποίο, αν και πρότυπο, πολλές επιχειρήσεις (οι ομάδες ανάπτυξης, δηλαδή) εμμένουν ακόμη και σήμερα στο FCM μοντέλο. Άλλοι λόγοι είναι ότι το ISO 9126 δεν ορίζει καθαρά πως μπορούν να μετρηθούν απευθείας τα επιμέρους χαρακτηριστικά του και η μη καλά ορισμένη διάσπαση των ποιοτικών χαρακτηριστικών σε επιμέρους υπο χαρακτηριστικά.

Κεφάλαιο 3 - Μετρικές Ποιότητας Λογισμικού

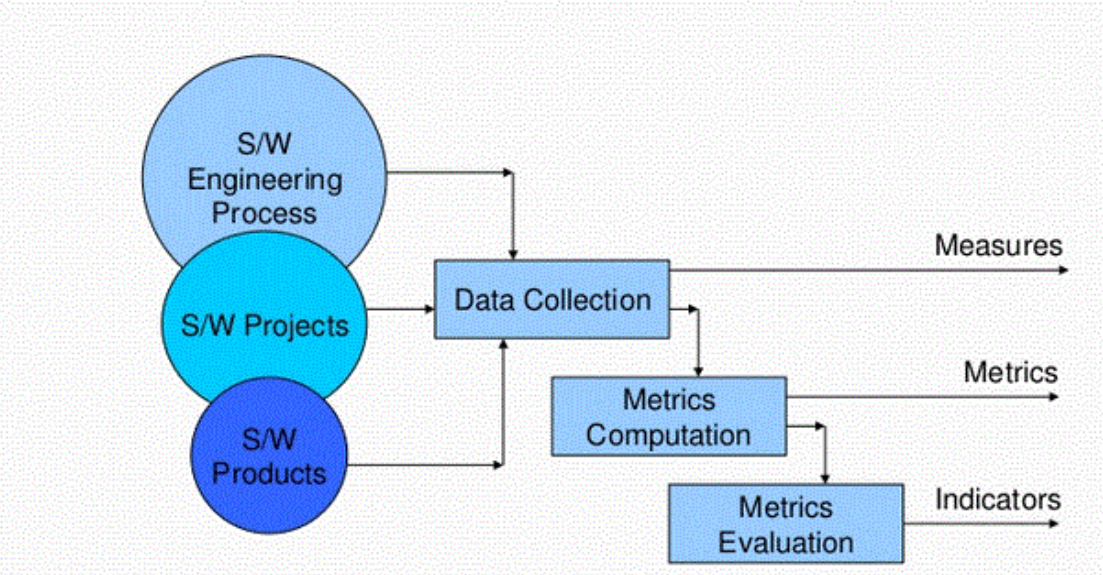
3.1 Εισαγωγή

Στο κεφάλαιο αυτό θα ορίσουμε την μέτρηση και την μετρική, πως σχετίζονται με την ποιότητα και πως είναι εφικτό να διασφαλιστεί η ποιότητα του λογισμικού με την χρήση των μετρήσεων και των μετρικών. Ακόμη, γίνεται παρουσίαση των σημαντικότερων μετρικών λογισμικού - συναρτησιακού και αντικειμενοστρεφής προγραμματισμού.

3.2 Μετρήσεις και Μετρικές

Στην τεχνολογία λογισμικού, οι όροι μέτρηση (Measure) και μετρική (Metric) είναι αλληλένδετοι και άμεσα εξαρτώμενοι. Μπορούμε να ορίσουμε ως Μέτρηση την αντιστοίχηση ενός μεγέθους με μια τιμή και την μετρική ως τη ποσοτική μέτρηση του βαθμού, στον οποίο ένα σύστημα ή τμήμα αυτού έχει ένα χαρακτηριστικό. Βέβαια οι ορισμοί είναι γενικοί αλλά στην τεχνολογία λογισμικού και ποιότητα λογισμικού ορίζεται πιο εμπεριστατωμένα, όπως θα δούμε παρακάτω [IEEE Software Engineering Standards, 1990].

Παρακάτω παραθέτονται κάποιοι ορισμοί ώστε να κατανοήσουμε καλύτερα την έννοια της μετρικής. Σύμφωνα με το Fenton, η μετρική ορίζεται ως μία εμπειρική και αντικειμενική αντιστοίχιση ενός αριθμού ή συμβόλου σε μία οντότητα με σκοπό να χαρακτηρίσει ένα συγκεκριμένο χαρακτηριστικό της, ενώ, σύμφωνα με το Edward Berard η μετρική ορίζεται ως μία μονάδα μέτρησης και αναφέρει ότι ο όρος χρησιμοποιείται για να δηλώσει ένα σύνολο μετρήσεων που πραγματοποιούνται πάνω σε συγκεκριμένο αντικείμενο ή διαδικασία. Τέλος, ο Tom De Marco (1982) είχε δηλώσει πολύ εύστοχα ότι "*You cannot control what you cannot measure*". Συμπερασματικά, μπορούμε να πούμε ότι μετρική είναι μία 'μεθοδολογία' μέτρησης που αντιστοιχεί μία τιμή σε κάποια προϋπάρχουσα ιδιότητα του αντικειμένου.



Εικόνα 3.1. Μετρήσεις και Μετρικές. Διαθέσιμο στο : http://www.slideshare.net/deepikashanti/13-software-metrics

Οι μετρικές χωρίζονται σε :

- ✓ **Μετρικές Διαδικασίας** (Διαδικασία: Άνθρωποι, Τεχνολογία, Προϊόν)
- ✓ **Μετρικές Έργου** (Π.χ. εκτίμηση διάρκειας και κόστους του έργου, COCOMO)
- ✓ **Μετρικές Προϊόντος** (Pressman, 1997)

Πιο αναλυτικά:

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

α) **Εσωτερικές**: αριθμός γραμμών (LOC), χρόνος εκτέλεσης, λάθη του κώδικα β) **Εξωτερικές** – λειτουργικότητα (functionality), ποιότητα (quality), πολυπλοκότητα (complexity),αποτελεσματικότητα (efficiency), αξιοπιστία (reliability), συντηρησι-μότητα (maintainability).

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

Οι μετρικές διαδικασίας μετρούν τη διαδικασία κατασκευής ενός προϊόντος, για παράδειγμα το σχεδιασμό, τη συγγραφή κώδικα, τους απαιτούμενους πόρους. (Pressman, 1997)

3.2.1 Εσωτερικές Μετρικές

Οι εσωτερικές μετρικές λογισμικού χρησιμοποιούνται για τη μέτρηση χαρακτηριστικών του λογισμικού για τα οποία υπάρχει απτή αντίληψη για τη φυσική τους σημασία και δυνατότητα άμεσης μέτρησης. Η χρήση εσωτερικών μετρικών λογισμικού ενθαρρύνεται από όλα τα διεθνή πρότυπα. Η μέτρηση εσωτερικών χαρακτηριστικών του λογισμικού μπορεί να γίνει πολύ εύκολα, γρήγορα και αυτόματα. διαδικασία μέτρησης μπορεί να πραγματοποιηθεί με ελάχιστη ανθρώπινη συμμετοχή, βασισμένη αποκλειστικά στον κώδικα του προγράμματος. Επομένως, η συχνότητα λαθών κατά τη διάρκεια αυτής της διαδικασίας είναι μηδαμινή και αμελητέα και τα αποτελέσματα των εσωτερικών μετρήσεων θεωρούνται έγκυρα και αντικειμενικά. Το κόστος διεξαγωγής εσωτερικών μετρήσεων είναι ασφαλώς πολύ μικρό, αφού το αντικείμενο της μέτρησης απαρτίζεται από απτά χαρακτηριστικά του κώδικα. Παρόλα αυτά, οι εσωτερικές μετρικές δεν προσφέρουν συνήθως πληροφορίες υψηλού επιπέδου που σχετίζονται με την ποιότητα του προϊόντος. Τα δεδομένα που προκύπτουν δεν είναι άμεσα αξιοποιήσιμα και ερμηνεύσιμα, αφού παρουσιάζουν μία αδυναμία σύνδεσης με τα εξωτερικά ποιοτικά χαρακτηριστικά του λογισμικού.

Διαδεδομένες Εσωτερικές Μετρικές

- Μετρικές της επιστήμης λογισμικού
- Μετρική κυκλωματικής πολυπλοκότητας
- Μετρική πολυπλοκότητας δομών δεδομένων

3.2.2 Εξωτερικές Μετρικές

Οι εξωτερικές μετρικές λογισμικού σχετίζονται άμεσα με τα εξωτερικά ποιοτικά χαρακτηριστικά του λογισμικού και χρησιμοποιούνται για τη μέτρηση χαρακτηριστικών του λογισμικού με υψηλό επίπεδο αφαίρεσης τα οποία σχετίζονται άμεσα με την ποιότητα του λογισμικού. Δίνουν δεδομένα άμεσα αξιοποιήσιμα και ερμηνεύσιμα, αφού μετρούν απευθείας τα ζητούμενα εξωτερικά ποιοτικά χαρακτηριστικά του λογισμικού. Εξυπηρετούν δηλαδή με άμεσο τρόπο τη δυνατότητα μέτρησης της ικανοποίησης των απαιτήσεων του πελάτη, η οποία αποτελεί βασικό στοιχείο κάθε ορισμού της ποιότητας. Τα αρνητικά στοιχεία των εξωτερικών μετρικών είναι: 1. βασίζονται σε υποκειμενικά δεδομένα και επομένως τα αποτελέσματά τους είναι πιθανόν να υπόκεινται σε αμφισβητήσεις και 2. η αδυναμία αυτοματοποίησης της διαδικασίας των εξωτερικών μετρήσεων.

3.3 Στόχοι των Μετρικών

Ο κυριότερος στόχων των μετρικών που εφαρμόζονται και διεξάγονται σε ένα λογισμικό είναι η διασφάλιση της ποιότητας. Εκ των πραγμάτων να διασφαλιστεί ότι το υλοποιηθέν πρόγραμμα παραγωγής είναι συμβατό και λειτουργικά αποδοτικό.

Παράλληλα με τις μετρικές που χρησιμοποιούνται, η διασφάλιση ποιότητας διακρίνεται σε:

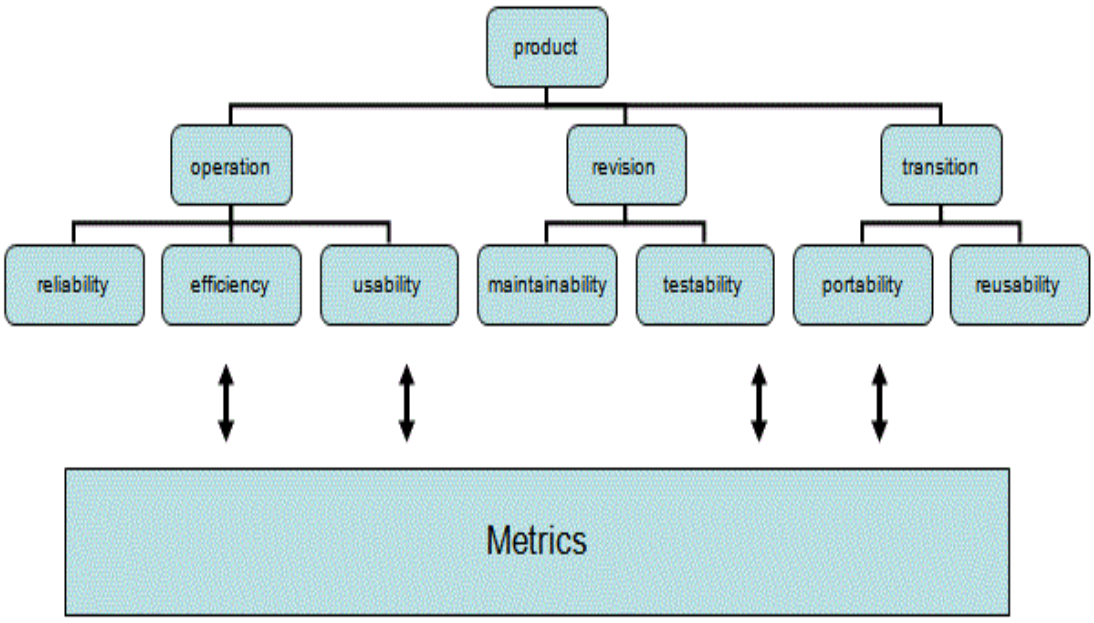
- Διασφάλιση ποιότητας προϊόντος (Product Quality Assurance): στόχος είναι η ποιότητα του προϊόντος και κατ’ επέκταση η ικανοποίηση του χρήστη.

- Διασφάλιση ποιότητας έργου (Project Quality Assurance): που στοχεύει α) στην αποτίμηση της κατάστασης του έργου, β) την επίλυση προβλημάτων πριν αυτά γίνουν ‘κρίσιμα’ και επηρεάσουν την πορεία του

έργου γ) τη ρύθμιση και το διαχωρισμό των εργασιών και δ) την εκτίμηση της ικανότητας των ομάδων να παράγουν ‘ποιοτικό’ λογισμικό.

- Διασφάλιση ποιότητας διαδικασιών (Process Quality Assurance): που βοηθά την εταιρεία να έχει γνώση για την αποτελεσματικότητα των διαδικασιών.

Με την χρήση των μετρικών γίνεται προσπάθεια να απαντηθούν μια σειρά από ερωτήματα, βάσει των οποίων, η μετρική αξιολογεί το πρόγραμμα παραγωγής. Έτσι, καταλυτική σημασία έχει η καλή “συγγραφή” του πηγαίου κώδικα, η δυνατότητα επαναχρησιμοποίηση του πηγαίου κώδικα σε άλλη εφαρμογή (Reusability), η συντηρησιμότητα του πηγαίου κώδικα (Maintainability), ο αριθμός των σφαλμάτων που εντοπίστηκαν κατά το έλεγχο του και τα πλαίσια βελτίωσης του πηγαίου κώδικα.



Εικόνα 3.2. Μοντέλο Ποιότητας Μετρικών

Με βάση τους στόχους για τους οποίους έχουν αναπτυχθεί οι μετρικές μπορούν να διαχωριστούν στις ακόλουθες κατηγορίες:

- Λειτουργικές μετρικές (Functional Metrics) που αποτελούνται από μετρικές καταλληλότητας (Suitability), ακρίβειας (Accuracy), διαλειτουργικότητας (interoperability), ελαστικότητας (Compliance) και ασφάλειας (Security) του λογισμικού.
- Μετρικές αξιοπιστίας (Reliability Metrics) που αποτελούνται από μετρικές ωριμότητας (Maturity), ανοχής λαθών (Fault Tolerance), ικανότητας ανάκτησης (Recoverability).

- Μετρικές χρησιμότητας (Usability metrics) που αποτελούνται από μετρικές "κατανοησιμότητας" (Understandability), ικανότητας μάθησης (Learn Ability) και λειτουργικότητας (Operability) του λογισμικού.

- Μετρικές αποδοτικότητας (Efficiency Metrics) που αποτελούνται από μετρικές συμπεριφοράς του συστήματος στο χρόνο (System Behavior over time) και χρήσης των πόρων (Usage of Resources).

- Μετρικές συντηρησιμότητας (Maintainability Metrics) που αποτελούνται από μετρικές ικανότητας ανάλυσης (Analysis Ability), τροποποιησιμότητας (Changeability), σταθερότητας (Stability) και ικανότητας ελέγχου (Testability).

- Μετρικές μεταφερσιμότητας (Portability Metrics) που αποτελούνται από μετρικές προσαρμοστικότητας (Adaptability), ικανότητας εγκατάστασης (Install Ability), συμμόρφωσης (Conformance) και ικανότητας αντικατάστασης (Replace Ability).

3.4 Μετρικές Λογισμικού

Οι μετρικές λογισμικού, που παρατίθενται στο συγκεκριμένο υποκεφάλαιο εφαρμόζονται γενικά στα λογισμικά παραγωγής. Ωστόσο, αρκετές από αυτές βρίσκουν εφαρμογή και σε αντικειμενοστρεφή λογισμικό, το οποίο είναι αποτέλεσμα μεταγλώττιση κώδικα κάποιας αντικειμενοστρεφής γλώσσας

(Java, JavaScript, PHP, C++, Python). Τέλος οι μετρικές λογισμικού κατατάσσονται στις εσωτερικές μετρικές του τελικού προϊόντος.

Σύμφωνα με τον Roger Pressman, οι μετρικές λογισμικού χωρίζονται σε κατηγορίες, ανάλογα με τη φάση ανάπτυξης του λογισμικού που εφαρμόζονται:

- ✓ Μετρικές για το μοντέλο της ανάλυσης απαιτήσεων
- ✓ Μετρικές για το μοντέλο του σχεδιασμού
- ✓ Μετρικές για τον πηγαίο κώδικα
- ✓ Μετρικές για τον έλεγχο του λογισμικού
- ✓ Μετρικές για τη συντήρηση (Pressman, 1997)

Στην συνέχεια, παρουσιάζονται επιγραμματικά πιο γνωστές μετρικές προϊόντος που
εφαρμόζονται στον τελικό κώδικα του λογισμικού.

- **Average Module Length**

Πρόκειται για μία μετρική της τμηματοποίησης ενός προγράμματος. Ορίζεται ως το μέσο μήκος των τμημάτων (modules) του (Fenton & Pfleeger, 1994).

- **Chen Metric**

Εξετάζει την εντροπία ενός προγράμματος. (Fenton & Pfleeger, 1994)

- **Decision Count**

Μετρά τη λογική δομή ενός προγράμματος. Ορίζεται από τον αριθμό των δηλώσεων ελέγχου (υπό-συνθήκη δηλώσεις και ανακυκλώσεις). (Fenton & Pfleeger, 1994) (Shepperd & Ince, 1993)

- **Executable Statements**

Μετρική μεγέθους. Ορίζεται ως τον αριθμό των εκτελέσιμων εντολών (executable statements) ενός προγράμματος. Μετρά τις διαφορετικές εντολές στην ίδια γραμμή ως διαφορετικές και αγνοεί τα σχόλια, τις δηλώσεις δεδομένων και τα headings**.** (Fenton & Pfleeger, 1994)

- **Extent Of Reuse**

Μετρά το ποσοστό επαναχρησιμοποίησης κώδικα σε ένα υπάρχον πρόγραμμα. Ορίζονται τέσσερα διαφορετικά επίπεδα: reused verbatim (κώδικας που επαναχρησιμοποιήθηκε χωρίς καμία αλλαγή), slightly modified (λιγότερο από 25% κώδικα έχει αλλαχθεί), extensively modified (25% και περισσότερο των γραμμών του κώδικα έχει αλλαχθεί) και new (εντελώς νέος κώδικας). Με τον τρόπο αυτό, για ένα συγκεκριμένο πρόγραμμα ορίζεται το μέγεθος (size) που τροποποιήθηκε σε σχέση με το συνολικό και, επομένως, το ποσοστό επαναχρησιμοποίησης στα τέσσερα επίπεδα. (Fenton & Pfleeger, 1994)

- **Function Count**

Ορίζεται από τον αριθμό των συναρτήσεων (functions) ενός προγράμματος. Συνάρτηση θεωρείται μια συλλογή εκτελέσιμων εντολών που πραγματοποιούν μια συγκεκριμένη εργασία και οι δηλώσεις των παραμέτρων που διαχειρίζονται οι εντολές αυτές. (Conte, et al., 1986) (Lorenz & Kidd, 1994)

- **Hausen Metric On Modularity**

Μετρική της τμηματοποίησης ενός προγράμματος που περιγράφει την ολική τμηματοποίηση σε σχέση με ειδικές της θεωρήσεις**.** (Fenton & Pfleeger, 1994)

- **Live Variables**

Στόχος της μετρικής αυτής είναι να χαρακτηρίσει τη ροή των δεδομένων μέσα σε ένα τμήμα (module). Τα δεδομένα (data items) κάθε εντολής ενός προγράμματος ονομάζονται ‘ζωντανές μεταβλητές’ (live variables). Η μετρική μετρά για κάθε γραμμή κώδικα πόσες μεταβλητές βρίσκονται σε χρήση και ορίζει LV = (αριθμός live variables) / (αριθμός εντολών) για κάθε διαδικασία. (Conte, et al., 1986)

- **Lines Of Code**

Μετρά τον αριθμό των γραμμών κώδικα. Παραλλαγές της είναι οι μετρικές που
μετράνε τον αριθμό των κενών γραμμών, του εκτελέσιμου κώδικα, των γραμμών του πηγαίου κώδικα, το ποσοστό σχολίων κλπ. (Fenton & Pfleeger, 1994), (Conte, et al., 1986), (Lorenz & Kidd, 1994)

- **McCabe’ s Essential Complexity Measure**

Μετράται το ποσό της δομής σε ένα πρόγραμμα. Για ένα πρόγραμμα με γράφο ροής G, η βασική πολυπλοκότητα ορίζεται ως ev(G) = v(G) - m,
όπου v(G) = e – n + 2 η κυκλωματική πολυπλοκότητα του γράφου ροής G
m ο αριθμός των υπογράφων ροής του γράφου G που είναι D-structured primes (είναι δηλαδή οι εντελώς στοιχειώδεις δομές D0, D1, D2 ή D3). (Fenton & Pfleeger, 1994)

- **Reach Ability**

Μετρά τη λογική δομή του προγράμματος.
Ορίζει την reachability ως R = πλήθος διαφορετικών τρόπων για να φτάσει κάποιος ένα κόμβο και R’ = (συνολικό πλήθος μονοπατιών) / (αριθμός κόμβων).
Η μετρική αυτή σχετίζεται με τον αριθμό των λαθών και τον χρόνο εντοπισμού τους. (Conte, et al., 1986)

- **Tree Impurity**

Η μετρική αυτή ορίζει τον αριθμό m(G) από το γράφο G του συστήματος. Ο m(G) ισούται με το ‘πόσο απέχει’ από ένα δένδρο ο γράφος αυτός. Όσο το m τείνει στο μηδέν, τόσο καλύτερος είναι ο σχεδιασμός του συστήματος. (Fenton & Pfleeger, 1994).

3.5 Συσχέτιση εσωτερικών και εξωτερικών μετρικών λογισμικού

Η εύρεση εσωτερικών μετρικών, οι οποίες ενσωματωμένες σε μια μέθοδο εσωτερικών μετρήσεων θα είχαν απόλυτη συσχέτιση με τις εξωτερικές μετρήσεις είναι πολύ δύσκολη. Πρακτικά, εάν υπήρχε ένα τέτοιο σύνολο μετρικών, δεν θα υπήρχε ανάγκη για εξωτερικές μετρήσεις, αφού η τήρηση των ορίων, τα οποία τίθενται από αυτές τις εσωτερικές μετρικές θα υποκαθιστούσε το πρόγραμμα ποιότητας. Με απλά λόγια, η τήρηση των ορίων στις εσωτερικές μετρικές θα συνεπάγονταν αυτόματα υψηλή ποιότητα του λογισμικού, οπότε δεν θα υπήρξε λόγος επιβεβαίωσης με την χρήση των εξωτερικών μετρικών.

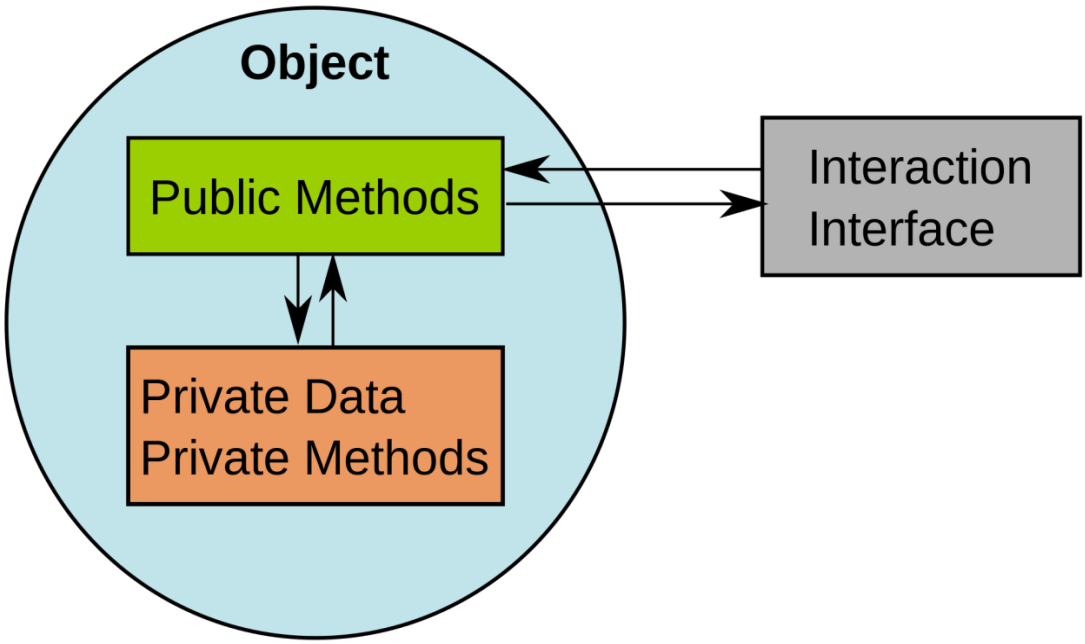
3.6 Μετρικές Αντικειμενοστρεφούς Λογισμικού

Οι αρχές του αντικειμενοστρεφούς προγραμματισμού διαφοροποιούνται σε σχέση με αυτές του συναρτησιακού. Ωστόσο, πολλές φορές, οι μετρικές που εφαρμόζονται στο λογισμικό, που είναι αποτέλεσμα αντικειμενοστρεφή προγραμματισμού, αποτελούν “παραλλαγές” των μη-αντικειμενοστρεφούς λογισμικού και εστιάζουν στα σημεία του πηγαίου κώδικα με αντικειμενοστρεφή χαρακτηριστικά.

3.6.1 Αρχές αντικειμενοστρεφούς σχεδίασης

Στον αντικειμενοστρεφή προγραμματισμό τον κύριο ρόλο έχουν οι κλάσεις και τα αντικείμενα. Κάθε αντικείμενο έχει μια συμπεριφορά, η οποία καθορίζεται από τις μεθόδους (συναρτήσεις) που αυτό διαθέτει. Μεταξύ των αντικειμένων και των κλάσεων δεν υπάρχουν απλώς σχέσεις καλούντος-καλούμενου, αλλά πλήθος άλλων σχέσεων, όπως ιεραρχίας, συνδέσμου κλπ.

Επίσης, δίνεται έμφαση στον σχεδιασμό των αντικειμένων και στην μεταξύ τους επικοινωνία, με την μορφή ανταλλαγής μηνυμάτων. Το σύστημα θεωρείται σαν ένα σύνολο αντικειμένων, που επικοινωνούν μέσω μηνυμάτων και εκτελούν ορισμένες λειτουργίες. Τα αντικείμενα που ενσωματώνουν μέσα τους δεδομένα και συναρτήσεις θεωρούνται κρίσιμα στοιχεία της εφαρμογής και πρέπει να ελέγχεται η σχεδίαση τους και η επικοινωνία τους με τα άλλα αντικείμενα της εφαρμογής[12].



Εικόνα 3.3. Τεχνική Αντικειμενοστρεφούς προγραμματισμού. Διαθέσιμο στο: http://en.wikibooks.org

Οι βασικές αρχές του Αντικειμενοστρεφούς Προγραμματισμού είναι:

[12] http://www.oop.esmartkid.com/

✓ **Η Αφαίρεση (Abstraction):** η αφαιρετική παρουσίαση μιας έννοιας του πραγματικού κόσμου παρουσιάζοντας μόνο τις λεπτομέρειες που δείχνουν πως θα χρησιμοποιήσουμε το αντικείμενο.

✓ **Η Ενθυλάκωση (Encapsulation)** – ή **Απόκρυψη Δεδομένων (Data Hiding)**: η ενσωμάτωση των χαρακτηριστικών (μεταβλητών) και τον μεθόδων (συναρτήσεων) σε κάθε αντικείμενο έτσι ώστε να εμφανίζονται μόνο οι πληροφορίες χρήσης του αντικειμένου και να κρύβονται οι πληροφορίες λειτουργίας του αντικειμένου.

✓ **Η Κληρονομικότητα (Inheritance)**: η ιδιότητα των αντικειμένων να κληρονομούν χαρακτηριστικά και μεθόδους από άλλα αντικείμενα.

✓ **Ο Πολυμορφισμός (Polymorphism):** η ιδιότητα των αντικειμένων να υλοποιούν με διαφορετικό τρόπο τις μεθόδους τους, που έχουν ίδιο όνομα με μεθόδους άλλων αντικειμένων.

3.6.2 Μετρικές Αντικειμενοστρεφούς Λογισμικού

Οι μετρικές Αντικειμενοστρεφούς λογισμικού, οι οποίες εκτιμούν την περιπλοκότητα και την ποσότητα μιας αντικειμενοστρεφούς σχεδίασης λογισμικού, χωρίζονται σε μετρικές κλάσης (class-oriented metrics) και μετρικές λειτουργίας (operation-oriented metrics).

- **Μετρικές κλάσης (class-oriented metrics)**

Οι μετρικές της κατηγορίας αυτής ασχολούνται με τις λειτουργίες (operations) και τα γνωρίσματα (attributes) μίας κλάσης, τις σχέσεις της κλάσης με τις υποκλάσεις και την κληρονομικότητα, τις σχέσεις της κλάσης με άλλες κλάσεις. Οι γνωστότερες μετρικές της κατηγορίας αυτής είναι:

α) Weighted Methods per Class, β) Depth of the Inheritance Tree, γ) Number of Children, δ) Coupling between Object Classes, ε) Response for a Class και στ) Lack of Cohesion in Methods. (Chidamber & Kemerer, 1994).

- **Μετρικές λειτουργίας (operation-oriented metrics)**

Οι σημαντικότερες μετρικές της κατηγορίας αυτής είναι: α) Average Operation Size, β) Operation Complexity και γ) Average Number of Parameter per operation (Lorenz & Kidd, 1994).

Στη συνέχεια παρουσιάζονται ορισμένες μετρικές αντικειμενοστρεφούς λογισμικού.

- **Weighted Methods per Class (WMC)**

Το ακέραιο πλήθος των μεθόδων που ορίζονται σε μία κλάση. Δεν υπάρχει ένα καθιερωμένο εύρος βέλτιστου WMC αλλά είναι γενικώς αποδεκτό ότι υπερβολικά μεγάλο WMC οδηγεί σε μεγαλύτερη πιθανότητα σφαλμάτων και προβλημάτων συντήρησης.

- **Depth of Inheritance Tree (DIT)**

Το μήκος του μέγιστου μονοπατιού κληρονομικότητας από την τρέχουσα κλάση έως τη ρίζα μιας ιεραρχίας κλάσεων. Μεγάλες τιμές DIT αυξάνουν την πολυπλοκότητα της σχεδίασης, την πιθανότητα σφαλμάτων αλλά και την επαναχρησιμοποίηση κώδικα λόγω της κληροδότησης μεθόδων. Μία βέλτιστη τιμή DIT θεωρείται το 5.

- **Number of Children (NOC)**

Το πλήθος των κλάσεων που κληρονομούν άμεσα από την τρέχουσα. Υψηλό NOC σημαίνει υψηλή επαναχρησιμοποίηση κώδικα αλλά πιθανόν να υποδεικνύει εννοιολογικά εσφαλμένη χρήση της κληρονομικότητας. Δεν

υπάρχει κάποιο γενικώς αποδεκτό βέλτιστο NOC, αφού αυτό εξαρτάται από την εκάστοτε κλάση, αλλά εν γένει οι κλάσεις οι ευρισκόμενες υψηλότερα στην ιεραρχία κληρονομικότητας είναι θεμιτό να έχουν υψηλότερο NOC από όσες τοποθετούνται χαμηλότερα αφού είναι λιγότερο εξειδικευμένες.

- **Coupling Between Object Classes (CBO)**

Το πλήθος των κλάσεων από τις οποίες εξαρτάται η τρέχουσα κλάση. Κάθε τέτοια εξάρτηση μπορεί να είναι αμφίδρομη ή μονόδρομη οποιασδήποτε κατεύθυνσης. Επομένως υψηλό CBO σημαίνει υψηλή σύζευξη (coupling) και δεν είναι επιθυμητό, καθώς αποτρέπει την επαναχρησιμοποίηση κώδικα και ζημιώνει τον αρθρωτό σχεδιασμό του προγράμματος. Όσο χαμηλότερο είναι το CBO μίας κλάσης τόσο πιθανότερο είναι να μπορεί η τελευταία να επαναχρησιμοποιηθεί ως μαύρο κουτί. Υπερβολικά υψηλές θεωρούνται οι τιμές CBO > 14.

- **Response for a Class (RFC και RFC')**

Η μετρική αυτή ισούται με το πλήθος των μεθόδων που μπορεί να εκτελεστούν ως απάντηση στη λήψη ενός μηνύματος / συμβάντος από την τρέχουσα κλάση (έστω Α), τόσο τοπικών της μεθόδων όσο και «απομακρυσμένων» μεθόδων άλλων κλάσεων οι οποίες καλούνται άμεσα από την Α. Η μετρική RFC' συμπεριλαμβάνει επιπλέον και τις απομακρυσμένες μεθόδους που καλούνται εμμέσως (π. χ. από άμεσα καλούμενες απομακρυσμένες μεθόδους ή, αναδρομικά, από άλλες έμμεσα καλούμενες απομακρυσμένες μεθόδους). Κάθε μέθοδος μετράται μόνο μία φορά στον υπολογισμό του RFC, ανεξαρτήτως του πόσες φορές καλείται. Υψηλή τιμή RFC υποδεικνύει υψηλές πιθανότητες σφάλματος και δυσκολία συντήρησης.

- **Average Operation Size**

Ορίζονται ως ο αριθμός των μηνυμάτων που στέλνονται από μία λειτουργία. Όταν ο αριθμός αυτός αυξάνεται, σημαίνει ότι δεν έχει γίνει καλή κατανομή των ‘υποχρεώσεων’ εσωτερικά στην κλάση.

- **LCOM1**

Μετρική έλλειψης συνοχής. Ισχύει LCOM1 = P - Q αν P > Q, διαφορετικά LCOM1 = 0. Τα P, Q υπολογίζονται με τον ακόλουθο αλγόριθμο (σε ψευδοκώδικα):

```
P = 0; Q = 0;
 Για κάθε ζεύγος μεθόδων της κλάσης
 do
 {
   Αν τα σύνολα των πεδίων που χρησιμοποιούν οι δύο τρέχουσες μέθοδοι είναι ξένα μεταξύ τους
      P = P + 1;
   Διαφορετικά
      Q = Q + 1;   }
```

Μία τιμή LCOM1 = 0 υποδεικνύει συνεκτική κλάση, ενώ αν LCOM1 > 0 η κλάση καλό είναι να διασπαστεί. Η μετρική LCOM1 παρουσιάζει κάποια προβλήματα, όπως π.χ. ότι δίνει τιμή 0 για κλάσεις πολύ διαφορετικές μεταξύ τους.

- **LCOM2**

Βελτιωμένη μετρική έλλειψης συνοχής. Ισχύει LCOM2 = 1 - sum(mA)/(m*a), όπου m το πλήθος των μεθόδων της κλάσης, a το πλήθος των πεδίων της, mA το πλήθος των μεθόδων που προσπελαύνουν ένα γνώρισμα και sum(mA) το άθροισμα των mA για όλα τα πεδία μίας κλάσης. Η μετρική LCOM2 αποτελεί τον μέσο όρο των ποσοστών των μεθόδων που δε χρησιμοποιούν κάθε γνώρισμα. LCOM2 = 0 (υψηλή συνοχή) σημαίνει πως όλες οι μέθοδοι της κλάσης χρησιμοποιούν όλα τα πεδία της, ενώ LCOM2 = 1 (καμία συνοχή) σημαίνει πως καμία μέθοδος δεν προσπελαύνει κανένα πεδίο.

- **LCOM3**

Εναλλακτική μετρική έλλειψης συνοχής. Ισχύει LCOM3 = (m - sum(mA)/a) / (m-1). Η μετρική LCOM3 λαμβάνει τιμές από 0 (υψηλή συνοχή) έως 2 (καμία συνοχή). Τιμές μεγαλύτερες του 1 σημαίνουν πως με βεβαιότητα υπάρχει τουλάχιστον ένα "νεκρό γνώρισμα", δηλαδή γνώρισμα το οποίο δεν προσπελαύνεται από καμία μέθοδο της κλάσης[13].

[13] http://en.wikipedia.org/wiki/Object-oriented_programming

Κεφάλαιο 4 - Διεπαφή Προγραμματισμού Εφαρμογών (Application Program Interfaces – APIs)

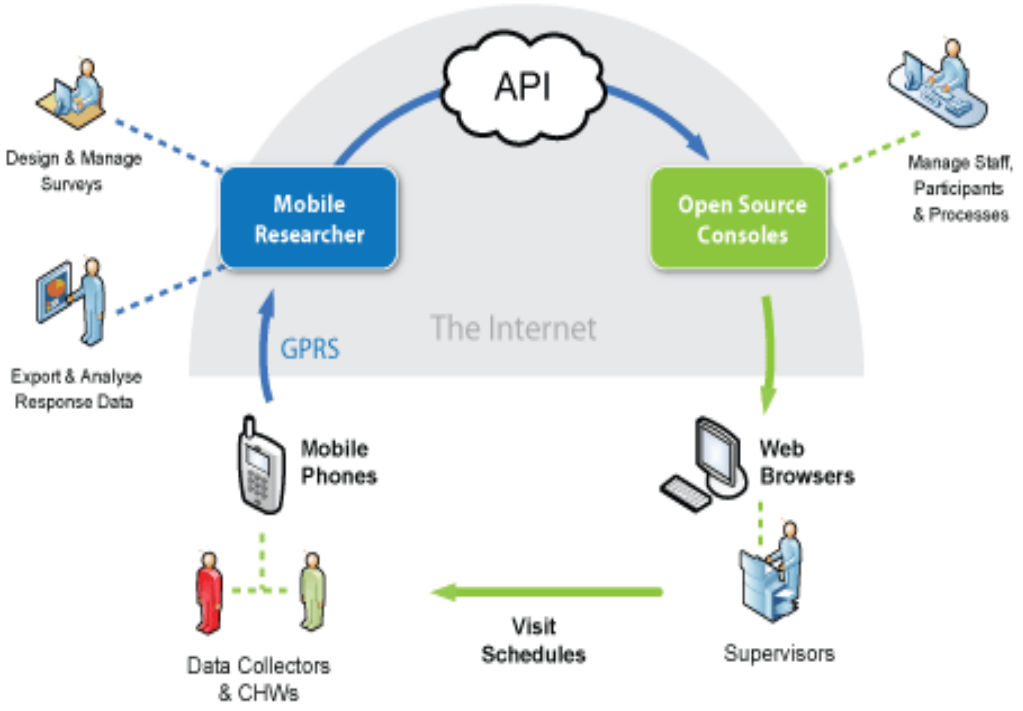
4.1 Ορισμός

Η διεπαφή Προγραμματισμού Εφαρμογών (αγγλικά API, από το Application Programming Interface), γνωστή και ως Διασύνδεση Προγραμματισμού Εφαρμογών ορίζεται ως η διεπαφή των προγραμματιστικών διαδικασιών που ένα λειτουργικό σύστημα, βιβλιοθήκη ή εφαρμογή παρέχει, προκειμένου να επιτρέπει να γίνονται προς αυτό αιτήσεις από άλλα προγράμματα ή / και ανταλλαγή δεδομένων[14].

Το Application Programming Interface (API) είναι ένα σύνολο από παραδοχές και ορισμούς, οι οποίες ορίζουν πώς μια υπηρεσία «καλείται» μέσα από ένα προγραμματιστικό περιβάλλον. Το API επιτρέπει στους προγραμματιστές να έχουν πρόσβαση στις λειτουργικότητες ενός προγράμματος μέσα από καλά καθορισμένες δομές δεδομένων. Με άλλα λόγια, το API περιγράφει ουσιαστικά τη λειτουργία και δομή ενός μέρους ή ολόκληρου προγράμματος και ορίζει τη διεπαφή αυτού με τον έξω κόσμο. Έχοντας την τεκμηρίωση ενός API είναι δυνατός ο προγραμματισμός ενός συστήματος χωρίς να έχουμε άλλου είδους πληροφορίες.

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

[14] http://en.wikipedia.org/wiki/Application_programming_interface



Εικόνα 4.1. Λειτουργία των APIs. Διαθέσιμο στο: http://www.mobenzi.com/researcher/Features/API

4.2 Λειτουργίες των APIs

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

Ένα API είναι μία λογισμικό-προς-λογισμικό διεπαφή και όχι μία διεπαφή χρήστη. Με τα API, οι εφαρμογές μιλούν μεταξύ τους χωρίς τη γνώση ή την παρέμβαση κάποιου χρήστη. Στη περίπτωση της αγοράς εισιτηρίων online και εισαγωγής των στοιχείων της πιστωτικής κάρτας, η ιστοσελίδα χρησιμοποιεί ένα API για να στείλει τα στοιχεία της πιστωτικής κάρτας σε μια απομακρυσμένη εφαρμογή που ελέγχει αν οι πληροφορίες είναι σωστές ή όχι. Μόλις επιβεβαιωθεί η πληρωμή, η απομακρυσμένη εφαρμογή στέλνει μια απάντηση πίσω στη ιστοσελίδα λέγοντας ότι είναι OK για την έκδοση των εισιτηρίων. Ο χρήστης μπορεί να δει μόνο ένα interface - την ιστοσελίδα - αλλά πίσω από τις σκηνές, πολλές εφαρμογές εργάζονται από κοινού με τη χρήση APIs. Αυτό το είδος της ενσωμάτωσης ονομάζεται απρόσκοπτη (seamless), μιας και ο χρήστης δεν καταλαβαίνει πότε οι λειτουργίες του λογισμικού χειρίζονται από μία εφαρμογή ή από την άλλη.

Ένα API μπορεί να είναι:

• **Language - Dependent** (εξαρτώμενο από τη γλώσσα), που σημαίνει ότι είναι διαθέσιμο μόνο με χρήση της σύνταξης και των στοιχείων μιας συγκεκριμένης προγραμματιστικής γλώσσας, με την οποία το API έχει υλοποιηθεί.

• **Language - Independent** (ανεξάρτητο από τη γλώσσα), οι εντολές μπορεί να είναι γραμμένες σε διάφορες γλώσσες προγραμματισμού. Αυτό είναι ένα επιθυμητό χαρακτηριστικό για ένα service-oriented API το οποίο δεν είναι δεσμευμένο με μια συγκεκριμένη διαδικασία ή σύστημα και μπορεί να παρέχεται ως απομακρυσμένες κλήσεις διαδικασίας (remote procedure calls) ή ως web services (Κοντοπρία, 2012).

4.3 Παραδείγματα γνωστών API

The PC BIOS call interface
Single UNIX Specification (SUS)
Windows API

Java Platform, Standard Edition API

Java Platform, Enterprise Edition API's

ASPI for SCSI device interfacing

Carbon and Cocoa for the Macintosh OS

IPhone API

OpenGL cross-platform 3D graphics API

DirectX for Microsoft Windows

Simple DirectMedia Layer (SDL)

Google Maps API

MediaWiki API

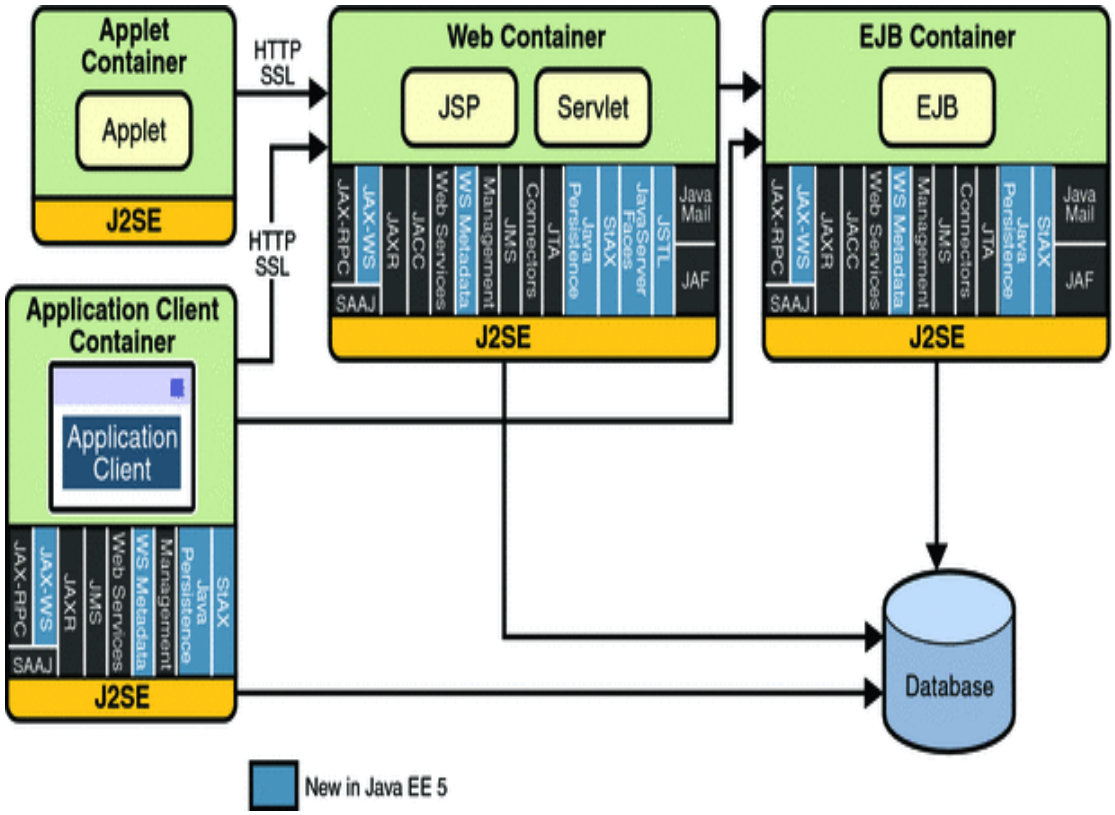
YouTube API

4.4 Παραδείγματα γνωστών Java API

Η **τυπική βιβλιοθήκη κλάσεων της Java** παρέχει ένα σετ κλάσεων Java API (Application Programming Interface), οι οποίες είναι διαθέσιμες σε κάθε Java περιβάλλον. Η βιβλιοθήκη κλάσεων οργανώνεται σε πακέτα. Στη JDK version 1.3 υπάρχουν περίπου 1000 κλάσεις σε 60 πακέτα (packages). Ένα πακέτο είναι μια συλλογή από κλάσεις. Κάποια από τα πακέτα που θα συναντήσετε συχνά είναι:

java.lang	**Κλάσεις που υποστηρίζουν τα βασικά χαρακτηριστικά της γλώσσας.**
	java.io	Κλάσεις για λειτουργίες εισόδου/εξόδου.
	java.util	Κλάσεις για επιτέλεση βασικών εργασιών σε δεδομένα
	java.awt	Κλάσεις που υποστηρίζουν το γραφικό περιβάλλον (GUI) της Java.
	java.swing	Πιο προχωρημένες GUI κλάσεις.
	java.applet	Κλάσεις για την εφαρμογή

		Java applets
	java.sql	Java Data Base Connectivity (JDBC) κλάσεις[15]



Εικόνα 4.2. Java EE Platform APIs.Διαθέσιμο στο: https://docs.oracle.com/javaee/5/tutorial/doc/bnacj.html

4.5 Τα API σε αντικειμενοστρεφή γλώσσες προγραμματισμού

Στην απλούστερη μορφή του, ένα αντικείμενο API είναι μια περιγραφή του, πώς τα αντικείμενα δουλεύουν σε ένα συγκεκριμένη αντικειμενοστρεφή γλώσσα - συνήθως εκφράζεται ως ένα σύνολο κατηγοριών που σχετίζονται με την λίστα των μεθόδων της κλάσης.

Για παράδειγμα, στη γλώσσα προγραμματισμού Java, αν ο κλάση Scanner πρόκειται να χρησιμοποιηθεί (μια κατηγορία που διαβάζει τα

[15] http://java.sun.com/j2se/1.3/docs/api/index.html

δεδομένα εισόδου από το χρήστη σε προγράμματα που βασίζονται σε κείμενο), είναι υποχρεωμένη να εισάγει τη βιβλιοθήκη java.util.Scanner, έτσι ώστε τα αντικείμενα του τύπου Scanner να μπορούν να χρησιμοποιηθούν με την επίκληση κάποιας από τις μεθόδους της κλάσης.

```
import java.util.Scanner;

public class Test {
  public static void main(String[] args) {
    System.out.println("Enter your name:");
    Scanner inputScanner = new Scanner(System.in);
    String name = inputScanner.nextLine();
    System.out.println("Your name is " + name + ".");
    inputScanner.close();  }   }
```

4.6 API Βιβλιοθήκες και Πλαίσια

Ένα API συνήθως σχετίζεται με μια βιβλιοθήκη λογισμικού: το API περιγράφει και προδιαγράφει την αναμενόμενη συμπεριφορά, ενώ η βιβλιοθήκη είναι μια πραγματική εφαρμογή του συνόλου των κανόνων. Ένα ενιαίο API μπορεί να έχει πολλαπλή εφαρμογή (ή καμία), με τη μορφή των διαφόρων βιβλιοθηκών να μοιράζονται την ίδια διεπαφή προγραμματισμού.

Ένα API μπορεί επίσης να συνδέεται με ένα πλαίσιο λογισμικού: ένα πλαίσιο μπορεί να βασίζεται σε αρκετές βιβλιοθήκες υλοποίησης αρκετών APIs, αλλά σε αντίθεση με την κανονική χρήση ενός API, η πρόσβαση στην συμπεριφορά, ενσωματωμένη στο πλαίσιο, προκαλείται από την επέκταση του περιεχομένου του, με νέες κατηγορίες, συνδεδεμένο στο ίδιο το πλαίσιο.

Επιπλέον, η συνολική ροή του προγράμματος ελέγχου μπορεί να είναι εκτός του ελέγχου του κληθέντος API [16] . (Mohamed & Douglas, 1997)

[16] http://martinfowler.com/bliki/InversionOfControl.html

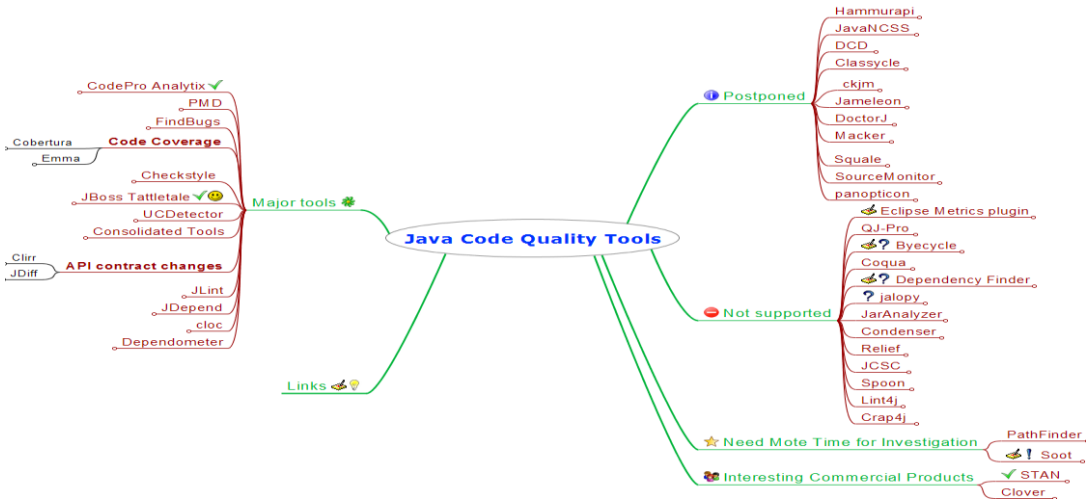
Κεφάλαιο 5 – Εργαλεία Μετρικών

5.1 Σημαντικότερα εργαλεία

5.1.1 Codepro Analytix

Είναι ένα μεγάλο εργαλείο (Eclipse plugin) για τη βελτίωση της ποιότητας του λογισμικού. Διαθέτει τα παρακάτω βασικά χαρακτηριστικά: Ανάλυση κώδικα, Δημιουργία δοκιμής JUnit (JUnit Test Generation), δομική επεξεργασίας JUnit (JUnit Test Editor), ανάλυση πανομοιότυπου κώδικα, μετρικές, επικάλυψη κώδικα και εξάρτηση ανάλυσης.

Το CodePro Analytix είναι το κορυφαίο εργαλείο δοκιμών λογισμικού Java για προγραμματισμό μέσω Eclipse, για προγραμματιστές που ενδιαφέρονται για τη βελτίωση της ποιότητας του λογισμικού και τη μείωση του κόστους υλοποίησης και των χρονοδιαγραμμάτων. Τα χαρακτηριστικά ελέγχου του εργαλείου καθιστούν απαραίτητη την εποπτεία του έργου με σκοπό την μείωση των σφαλμάτων, καθώς αναπτύσσεται ο κώδικας, διατηρώντας πρακτικές κωδικοποίησης, σύμφωνα με τις οργανωτικές κατευθυντήριες γραμμές.

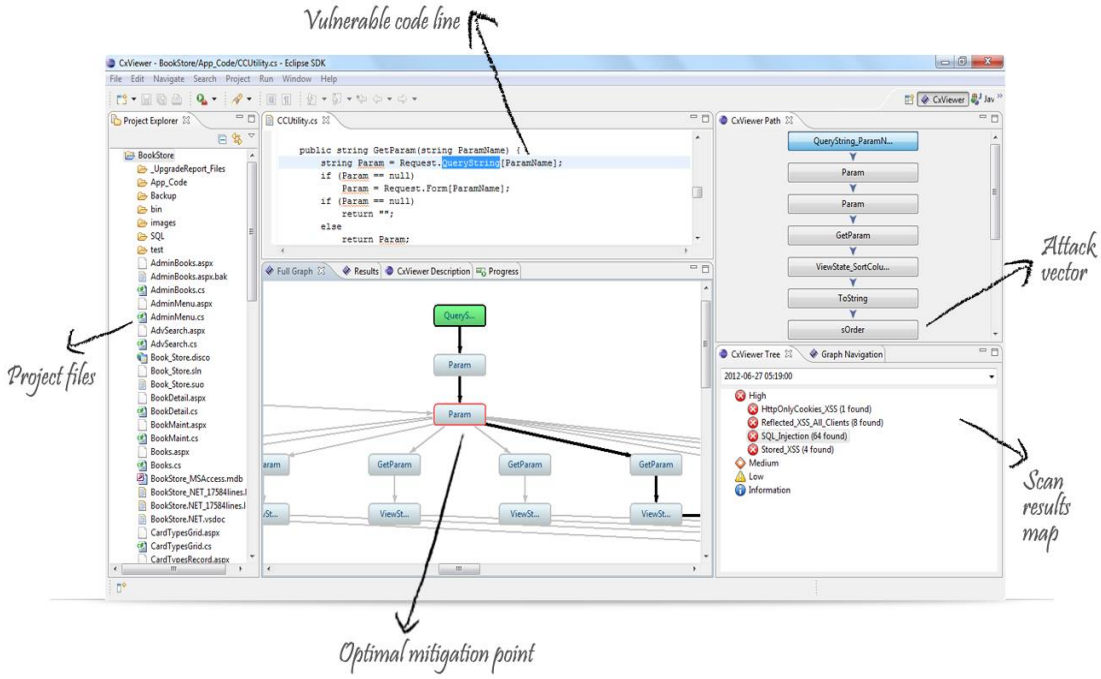


Εικόνα 5.1. Εργαλεία Μετρικών Java. Διαθέσιμο στο: http://2.bp.blogspot.com/-Z1kYo9ShOgU/UGi129KTi6I/ AAAAAAAAClw/t4cG6z65Gv8/s1600/Java+Code+Quality.png

Κύρια χαρακτηριστικά του Codepro Analytix

Ανάλυση κώδικα (Code Analysis)

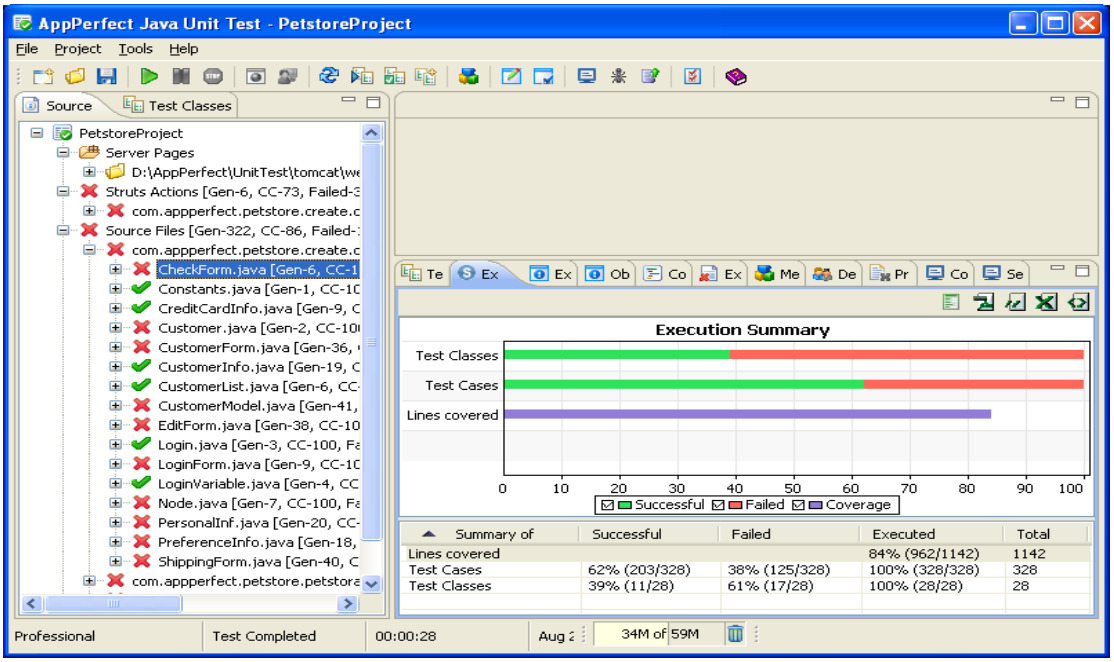
Κατά την ανάλυση του κώδικα, παρέχονται δυναμικά, επεκτάσιμα εργαλεία που ανιχνεύουν, εκθέτουν και «επισκευάζουν» αποκλίσεις ή μη συμμόρφωση με προκαθορισμένα πρότυπα κωδικοποίησης. Ακόμη παρέχονται δημοφιλή πλαίσια, ασφάλεια και οι συμβάσεις στυλ.



Εικόνα 5.2. Code Analysis - Codepro Analytix. Διαθέσιμο στο: https://www.checkmarx.com/

Δημιουργία δοκιμής JUnit (JUnit Test Generation)

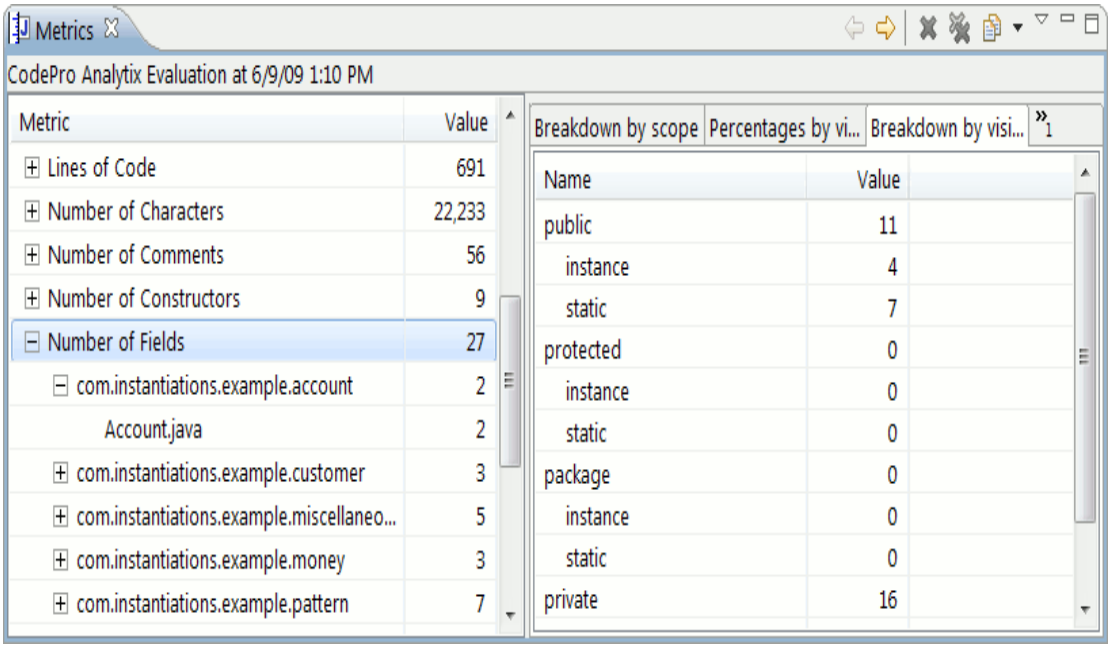
Παρέχεται εξοικονόμηση χρόνου που χρησιμοποιεί εξελιγμένες τεχνικές ανάλυσης διαδρομής ροής που αυτοματοποιεί τη δημιουργία των δοκιμών ολοκληρωμένης παλινδρόμησης JUnit.



Εικόνα 5.3. Code Analysis - JUnit Test Generation. Διαθέσιμο στο: http://www.appperfect.com/products/java-unit-test.html

Μετρικές (Metrics)

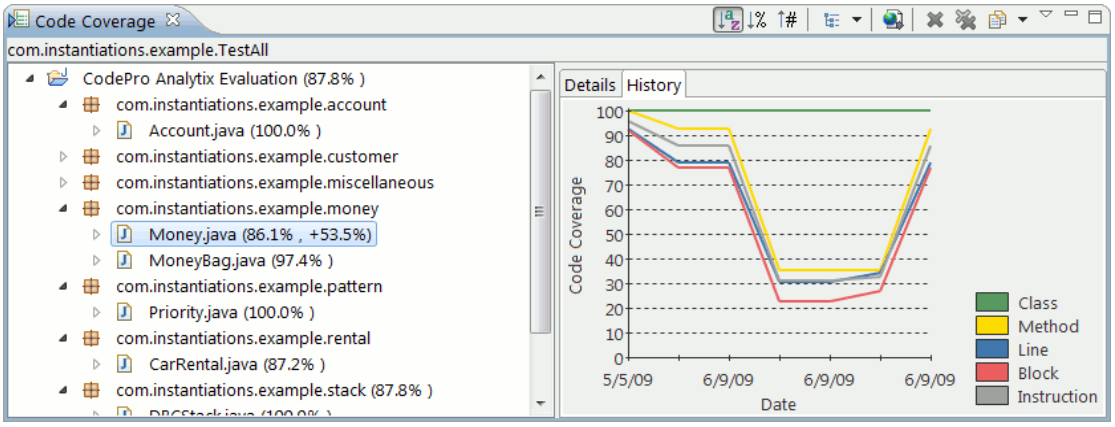
Παρέχονται αυτοματοποιημένα εργαλεία που μετρούν και δημιουργούν αναφορές σχετικά με τους βασικούς δείκτες ποιότητας σε ένα κύριο «σώμα» του κώδικα της Java.



Εικόνα 7. Code Analysis - Metrics. Διαθέσιμο στο: https://developers.google.com/java-dev-tools/codepro/doc/features/metrics/metrics

Επικάλυψη κώδικα (Code Coverage)

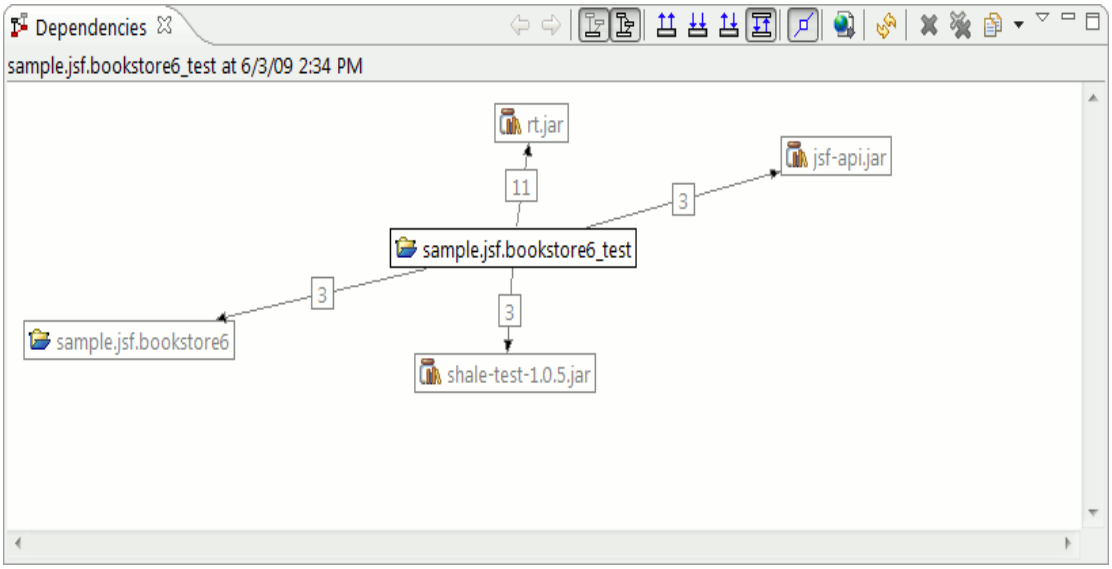
Ισχυρά εργαλεία που μετρούν το ποσοστό του εκτελέσιμου κώδικα χρησιμοποιώντας παραγόμενες περιπτώσεις δοκιμών ή χειροκίνητα σενάρια δοκιμών.



Εικόνα 5.4. Code Analysis - Code Coverage. Διαθέσιμο στο: https://developers.google.com/java-dev-tools/codepro/doc/features/codecoverage/code_coverage

Ανάλυση εξάρτησης (Dependency Analysis)

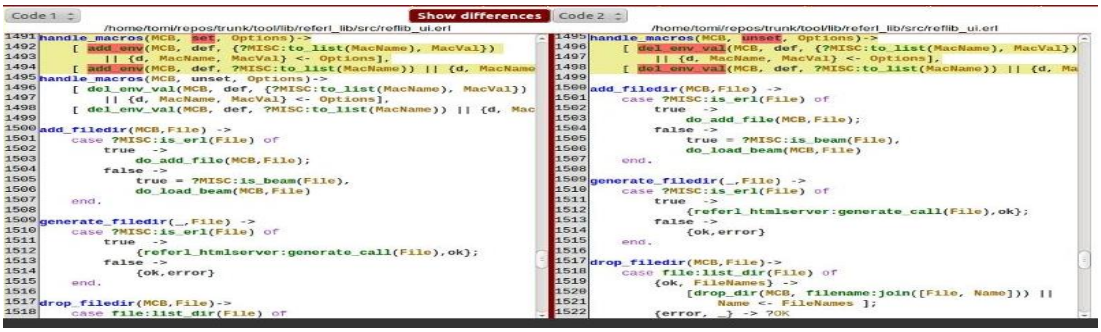
Αυτοματοποιημένα εργαλεία που αναλύουν και οπτικά απεικονίζουν τις εξαρτήσεις μεταξύ των έργων, των πακέτων και των ειδών



Εικόνα 5.5. Code Analysis - Code Coverage. Διαθέσιμο στο: https://developers.google.com/java-dev-tools/codepro/doc/features/dependencies/dependencies

- **Ανάλυση Πανομοιότυπου Κώδικα (Similar Code Analysis)**

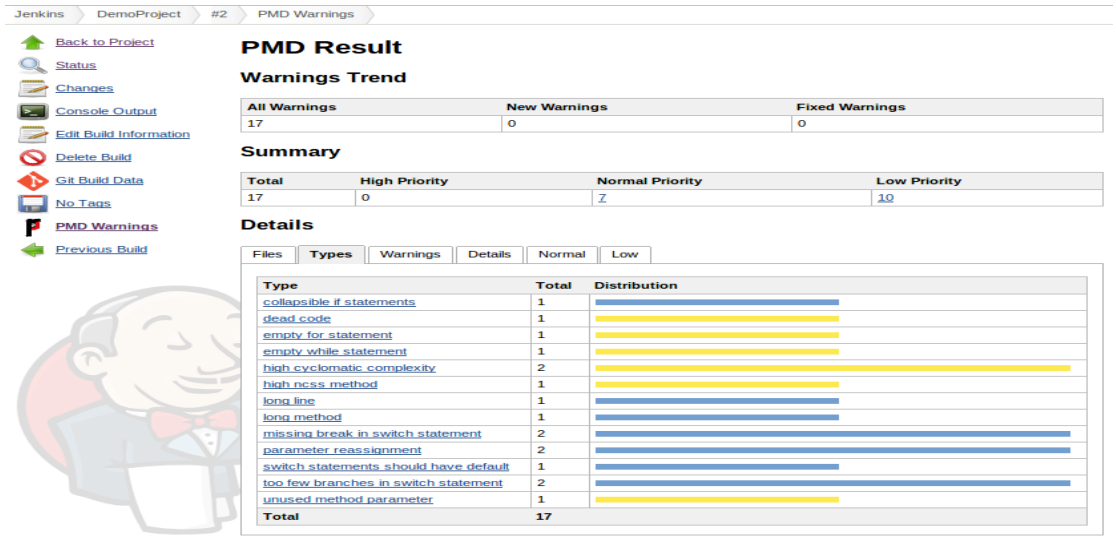
Εξετάζει αποτελεσματικά τον κώδικα της Java για να βρεί τα διπλά ή τα πολύ παρόμοια τμήματα του κώδικα που περιέχει σφάλματα αντιγραφής / επικόλλησης.



Εικόνα 5.6. Code Analysis - Similar Code Analysis. Διαθέσιμο στο: https://developers.google.com/java-dev-tools/codepro/html/what_is_sc

5.1.2 PMD

Το PMD είναι ένας αναλυτής πηγαίου κώδικα. Βρίσκει συνήθη σφάλματα προγραμματισμού όπως αχρησιμοποίητες μεταβλητές, κενών μπλοκ κώδικα και περιττή δημιουργία αντικειμένων. Υποστηρίζει Java, JavaScript, XML, XSL. Επίσης, περιλαμβάνει CPD, ανιχνευτή κομματιών copy-paste. Το εργαλείο CPD βρίσκει διπλές γραμμές κώδικα σε Java, C, C ++, C #, PHP, Ruby, Fortran, και JavaScript.



Εικόνα 5.7. PMD. Διαθέσιμο στο : http://blog.mgm-tp.com/2010/05/understanding-hudsons-code-analysis-plugins/

Χαρακτηριστικά

- Εργαλείο στατικής ανάλυσης για λογισμικό γραμμένο σε Java (επίσης, αρκετά διαδεδομένο)
- Ανοιχτού κώδικα (υπό ενός "BSD-style" license)
- Ανεξάρτητη κοινότητα ανάπτυξης, Υποστηρικτές: DARPA, ae.be, QA-Systems κ.ά.
- Εξετάζει κώδικα Java αναζητώντας παραβιάσεις (violations) έναντι ενός συνόλου κανόνων και πιο συγκεκριμένα:
    - πιθανά σφάλματα, «νεκρό» κώδικα, μη βέλτιστο κώδικα και υπερβολικά σύνθετες εκφράσεις
- Περιλαμβάνει επίσης εργαλείο εντοπισμού κλώνων μέσω string matching (αλγόριθμος Karp-Rabin)
- Site: http://pmd.sourceforge.net/
- Διατίθενται plug-ins για πολλά IDEs.

Ενσωμάτωση & Χρήση

- ✓ Εγκατάσταση ως Eclipse plug-in (υπάρχει, επίσης, αντίστοιχο update site)
- ✓ Στο Eclipse προστίθεται το PMD Perspective και τα Views:
    - Violations Outline: Δένδρο πλοήγησης παραβιάσεων
    - Violations Overview: Παραβιάσεις ανά κλάση
    - CPD View (για το εργαλείο εντοπισμού κλώνων) και Dataflow View
- ✓ Η εφαρμογή της ανάλυσης γίνεται στο σύνολο του project.
- ✓ Δεν απαιτείται επέμβαση στον κώδικα ή περαιτέρω ενέργειες από το χρήστη (εκτός ίσως από ρυθμίσεις των παραμέτρων της ανάλυσης)

Παράμετροι

- Δυνατότητα ρυθμίσεων ανά “working set”
- Ανάλυση

- Επιλογή των κανόνων έναντι των οποίων ο κώδικας θα ελεγχθεί για παραβιάσεις. Επιλέχθηκαν όλοι οι διαθέσιμοι. Συγκεκριμένα, στο PMD υπάρχουν «σύνολα κανόνων» (rule sets) τα οποία περιλαμβάνουν κανόνες παρόμοιων χαρακτηριστικών και διαφόρων επιπέδων προτεραιότητας. Κατά συνέπεια, οι παραβιάσεις που εντοπίζονται έχουν νόημα εφόσον ο κανόνας που επιλέγεται έχει νόημα για το project.

- Αναφορά
  - Οι παραβιάσεις οργανώνονται ανά πακέτο και κλάση.
  - Επιλέγονται τα επίπεδα προτεραιότητας που εμφανίζονται, τα οποία είναι: [1] error high, [2] error, [3] warning high, [4] warning, [5] information
  - Δυνατότητα υπολογισμού στατιστικών στοιχείων (# Violations, #Violations / LOC, #Violations / Method)

5.1.3 Findbugs

Εξετάζει τον κώδικα Java για σφάλματα. Μπορεί να ανιχνεύσει μια ποικιλία από κοινά λάθη κωδικοποίησης, συμπεριλαμβανομένων των προβλημάτων συγχρονισμού νήματος, κακή χρήση των μεθόδων API, κ.ά.

Πιο αναλυτικά, το FindBugs είναι ένα πρόγραμμα που εξετάζει τα μεταγλωττισμένα αρχεία java (δηλαδή τα αρχεία με κατάληξη .class) ή τα JAR/WAR/EAR πακέτα του προγράμματος και συγκρίνει τα bytecodes με διάφορα bug patterns. Αν βρει κάποιο bug θα το επισημάνει με απόλυτη ακρίβεια στον κώδικα.

Χαρακτηριστικά

- Ένα από τα πιο δημοφιλή εργαλεία στατικής ανάλυσης για έργα λογισμικού υλοποιημένα σε Java
- >700.000 downloads (Ιούλιος 2012)

- Ανοιχτού κώδικα (Lesser GNU Public License)
- Φορείς: U of Maryland, Υποστηρικτές: SureLogic, Google, Sun Microsystems, National Science Foundation των ΗΠΑ
- Αναλύει στατικά Java bytecode έναντι ενός συνόλου προτύπων σφαλμάτων
    - Σύμφωνα με το site, το ποσοστό των false warnings που δίνει είναι μικρότερο από 50%
- Site: http://pmd.sourceforge.net/
- Διατίθενται plug-ins για πολλά IDEs.

Ενσωμάτωση & Χρήση

- ✓ Εγκατάσταση ως Eclipse plug-in (υπάρχει αντίστοιχο update site)
- ✓ Στο Eclipse προστίθεται το FindBugs Perspective και τα Views:
    - Bug Explorer: Δένδρο πλοήγησης σφαλμάτων (σημαντικές δυνατότητες ομαδοποίησης & φιλτραρίσματος)
    - Bug User Annotations: Ο χρήστης μπορεί να κατηγοριοποιήσει και να προσθέσει επισημάνσεις για κάθε bug
- ✓ Η περιγραφή κάθε bug παρατίθεται στο Properties View.
- ✓ Η εφαρμογή της ανάλυσης γίνεται στο σύνολο του project.
- ✓ Δεν απαιτείται επέμβαση στον κώδικα (π.χ. annotations) ή γενικότερα περαιτέρω ενέργειες από τον χρήστη (πέρα ενδεχομένως από ρυθμίσεις των παραμέτρων της ανάλυσης)

Παράμετροι

- Δυνατότητα για project specific / workspace ρυθμίσεις
- Ανάλυση
    - Επίπεδο «προσπάθειας ανάλυσης»: minimum, default, maximum. Επιλέχθηκε το maximum
    - Επιλογή των bug detectors που θα χρησιμοποιηθούν. Επιλέχθηκαν όλοι οι διαθέσιμοι

- Κάθε bug detector αντιστοιχεί σε ένα σύνολο προτύπων σφαλμάτων (bug patterns) συχνά διαφορετικής προτεραιότητας & κατηγορίας.

- Αναφορά
  - Επίπεδο προτεραιότητας: Low, Normal, High. *Επιλέχθηκε Low*.
  - Κατηγορία: Malicious code vulnerability, Dodgy, Bad practice, Correctness, Internationalization, Performance, Security, Multithreaded correctness, Experimental, Bogus random noise.
- Αρχεία φίλτρων (include/exclude φίλτρα).

Εικόνα 5.8. Findbugs. Διαθέσιμο στο: http://findbugs.sourceforge.net/

5.1.4 Checkstyle

Είναι ένα εργαλείο ανάπτυξης για να βοηθήσει τους προγραμματιστές που γράφουν κώδικα Java, ο οποίος προσκολλάται σε ένα πρότυπο κωδικοποίησης. Το Checkstyle είναι εξαιρετικά παραμετροποιήσιμο και μπορεί να γίνει για να υποστηρίξει σχεδόν κάθε πρότυπο κωδικοποίησης. Τα παρεχόμενα αρχεία διαμόρφωσης υποστηρίζουν τις συμβάσεις Sun Κώδικα και το Google Java.

Το Checkstyle είναι πιο χρήσιμο αν ενσωματωθεί στη διαδικασία κατασκευής ή στο περιβάλλον ανάπτυξης. Η διανομή περιλαμβάνει:

- ✓ Apache Ant
- ✓ Ένα εργαλείο γραμμής εντολών (A command line)

Site: http://eclipse-cs.sourceforge.net/#!/

Προσθήκες Checkstyle

IDE / Build tool	Main/Initial Author	Available from
SCM-Manager		SCM-Manager Plugin Page
jGRASP	Larry Barowski	jGRASP Home Page
Sonar	Freddy Mallet (initial author)	Sonar Home Page
Eclipse/RAD/RDz	David Schneider	Eclipse-CS Home Page
Eclipse/RAD/RDz	Roman Ivanov	Project Page
Eclipse/RAD/RDz	Marco van Meegen	Checklipse Home Page
IntelliJ IDEA	Jakub Slawinski	QAPlug
IntelliJ IDEA	James Shiell	Checkstyle-idea Project Page
IntelliJ IDEA	Mark Lussier	JetStyle Project Page
NetBeans	Petr Hejl	Checkstyle Beans
NetBeans	Paul Goulbourn	nbCheckStyle
NetBeans		Software Quality Environment (SQE)
	jCoderZ	fawkeZ
BlueJ	Rick Giles	bluejcheckstyle home page
tIDE		Built in
Emacs JDE	Markus Mohnen	Part of the standard JDEE distribution
jEdit	Todd Papaioannou	JEdit CheckStylePlugin
Vim editor	Xandy Johnson	Plugin Homepage
Maven	Vincent Massol	Checkstyle supported out of the box
QALab	Benoit Xhenseval	QALab Home Page
JArchitect		JArchitect Home Page
Jenkins Checkstyle plug-in		Jenkins Checkstyle plug-in Home Page
Bamboo Checkstyle plug-in	Atlassian (formerly by Ross Rowe and Stephan Paulicke)	Bamboo Checkstyle plug-in Atlassian Marketplace page

Εικόνα 5.9. Προσθήκες Checkstyle

5.1.5 JBoss Tattletale

Το JBoss Tattletale είναι ένα εργαλείο που μπορεί να σας βοηθήσει να πάρετε μια γενική εικόνα του έργου που εργάζεστε ή ένα προϊόν που εξαρτώνται από αυτό. Το εργαλείο σαρώνει αναδρομικά έναν κατάλογο για τα αρχεία JAR και να δημιουργήσει σχηματοποιημένες HTML εκθέσεις.

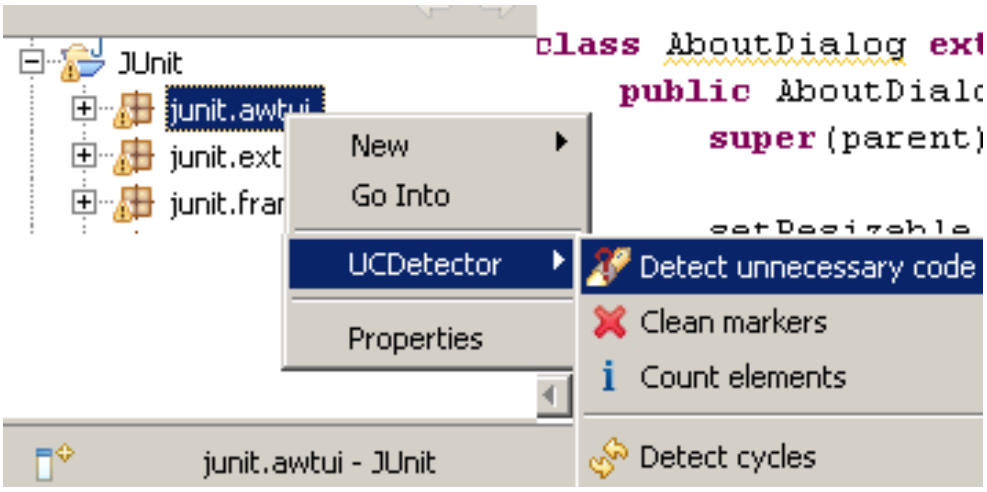
Εικόνα 5.10. JBoss Tattletale. Διαθέσιμο στο: http://www.ivankrizsan.se/2014/12/27/dependency-management-with-jboss-tattletale/

Site: http://tattletale.jboss.org/

5.1.6 UCDetector

Το UCDetector είναι ένα plugin για Eclipse το οποίο έχε σχεδιαστεί για να βοηθήσει το χρήστη να βρεί περιττά (νεκρά) δημόσια Java code. Το UCDetector δημιουργεί δείκτες για τα ακόλουθα προβλήματα:

- ✓ Άσκοπος (νεκρός) κώδικας
- ✓ Μέθοδοι πεδίων, οι οποίοι μπορεί να είναι οριστικοί[17]



Εικόνα 5.11. UCDetector. Διαθέσιμο στο: http://www.ucdetector.org/

Site: http://www.ucdetector.org/

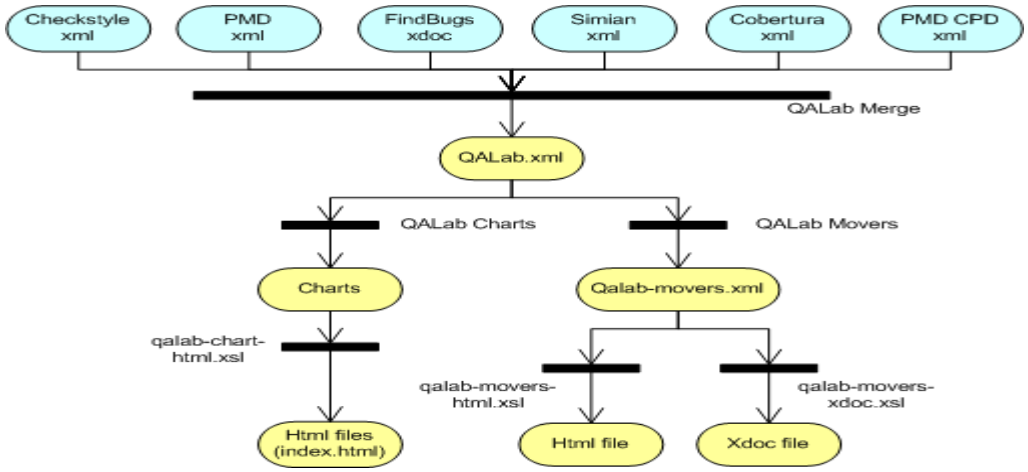
[17] http://www.ucdetector.org/

5.1.7 QALab

Το QALab ενοποιεί δεδομένα από Checkstyle, PMD, FindBugs και τα εμφανίζει σε μια ενοποιημένη άποψη. Το QALab διατηρεί το χρονοδιάγραμμα των αλλαγών ώστε να μπορεί ο χρήστης να δεί τις διαχρονικές τάσεις.

Τα ακόλουθα εργαλεία που υποστηρίζονται αυτήν τη στιγμή:

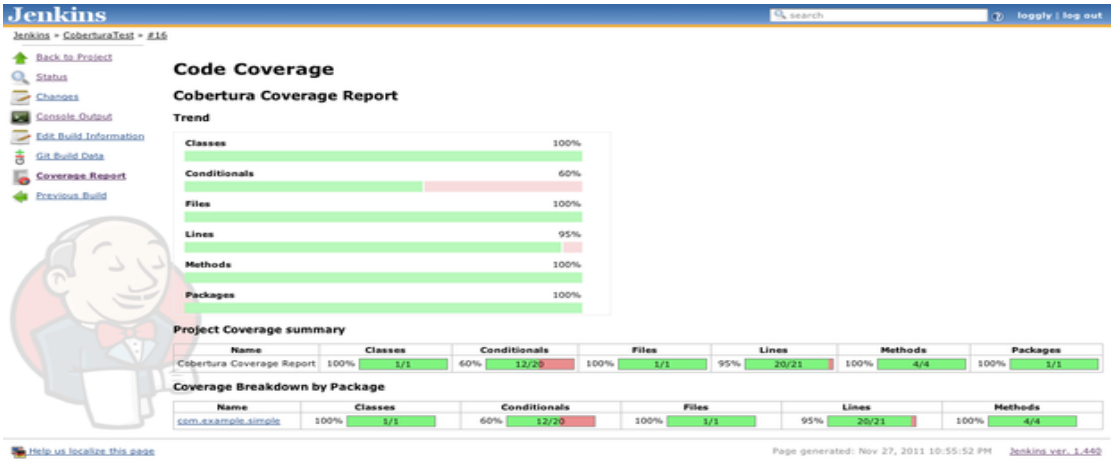
- ✓ Checkstyle
- ✓ PMD
- ✓ PMD CPD
- ✓ FindBugs
- ✓ Cobertura
- ✓ Simian



Εικόνα 5.12. QALab. Διαθέσιμο στο: http://qalab.sourceforge.net/

5.1.8 Cobertura

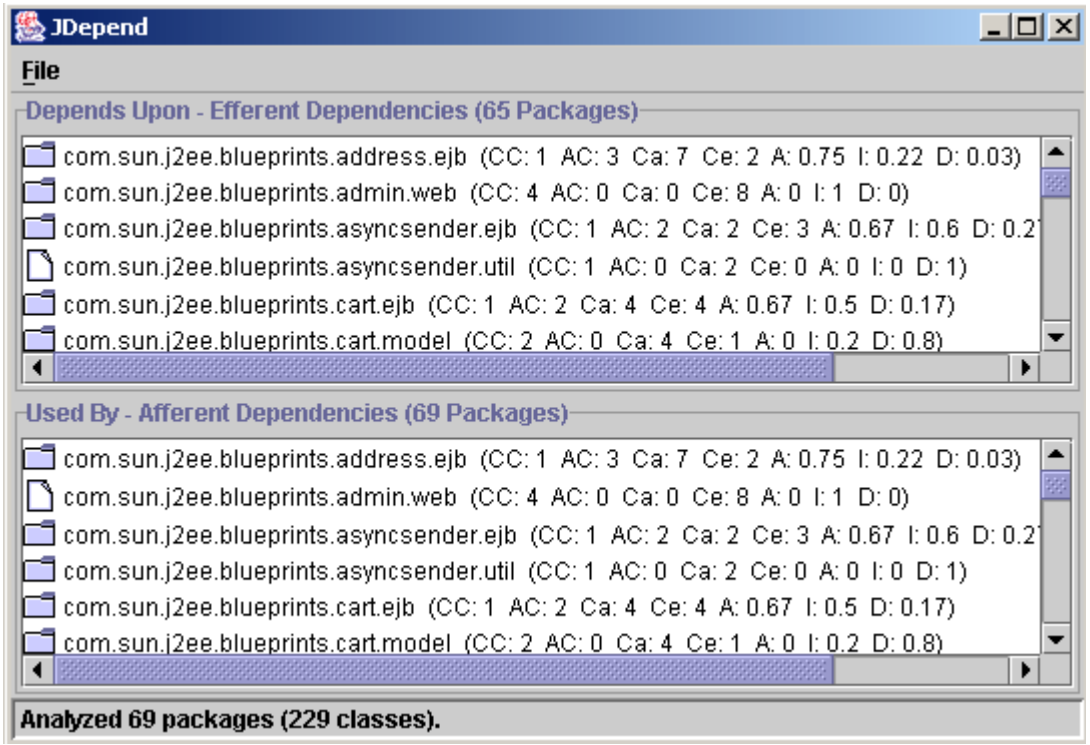
Το Cobertura είναι ένα δωρεάν εργαλείο Java που υπολογίζει το ποσοστό του κώδικα, το οποίο είναι προσβάσιμο από τις δοκιμές. Μπορεί να χρησιμοποιηθεί για να εντοπίσει ποια τμήματα του προγράμματος Java στερούνται κάλυψης δοκιμής. Βασίζεται σε jcoverage.



Εικόνα 5.13. Cobertura - Code Coverage – Διαθέσιμο στο: https://www.loggly.com/blog/java-code-coverage-with-cobertura-and-jenkins/

5.1.9 JDepend

Το JDepend διασχίζει καταλόγους αρχείων Java και δημιουργεί μετρήσεις ποιότητας σχεδιασμού για κάθε πακέτο Java. Επίσης, το JDepend επιτρέπει την αυτόματη μέτρηση της ποιότητας του σχεδιασμού, όσον αφορά την επεκτασιμότητα του, την επαναχρησιμοποίηση και την συντήρηση των πακέτων.

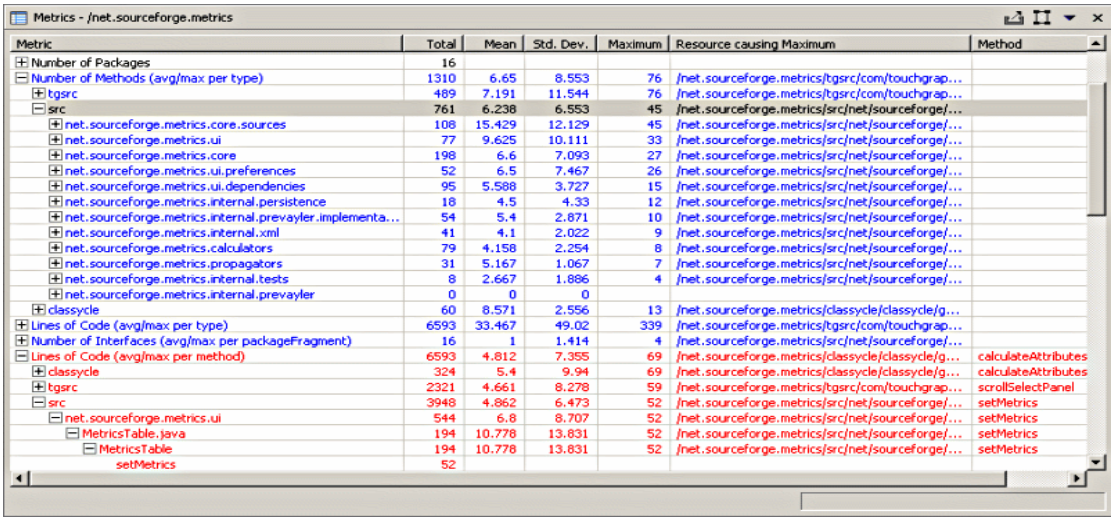


Εικόνα 5.14.JDepend - Διαθέσιμο στο: http://www.onjava.com/pub/a/onjava/2004/01/21/jdepend.html

5.2 Δευτερεύοντα Εργαλεία (Μη ευρέως χρησιμοποιούμενα)

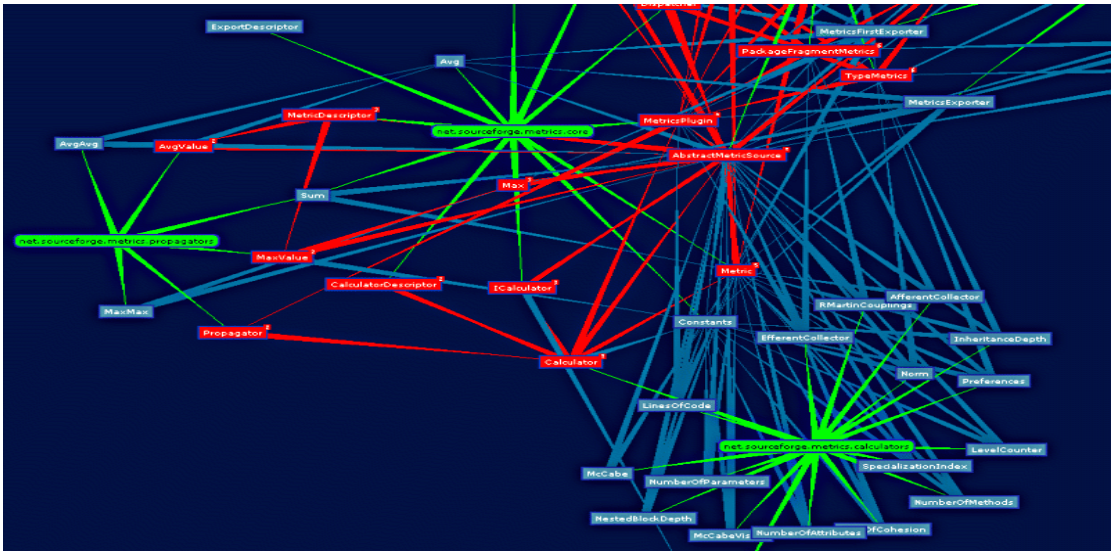
5.2.1 Eclipse Metrics plugin

Παρέχει μετρήσεις υπολογισμού και αναλυτή εξάρτησης plugin για την πλατφόρμα Eclipse. Ακόμα παρέχει διάφορες πληροφορίες για τις μετρικές, όπως μέσος όρος και τυπική απόκλιση, εντοπισμό κύκλων ανά πακέτο. Τέλος έχει την δυνατότητα να παράγει κάποια γραφήματα.



Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum	Method
Number of Packages	16					
Number of Methods (avg/max per type)	1310	6.65	8.553	76	/net.sourceforge.metrics/tgsrc/com/touchgrap...	
tgsrc	489	7.191	11.544	76	/net.sourceforge.metrics/tgsrc/com/touchgrap...	
src	761	6.238	6.553	45	/net.sourceforge.metrics/src/net/sourceforge/...	
net.sourceforge.metrics.core.sources	108	15.429	12.129	45	/net.sourceforge.metrics/src/net/sourceforge/...	
net.sourceforge.metrics.ui	77	9.625	10.111	33	/net.sourceforge.metrics/src/net/sourceforge/...	
net.sourceforge.metrics.core	198	6.6	7.093	27	/net.sourceforge.metrics/src/net/sourceforge/...	
net.sourceforge.metrics.ui.preferences	52	6.5	7.467	26	/net.sourceforge.metrics/src/net/sourceforge/...	
net.sourceforge.metrics.ui.dependencies	95	5.588	3.727	15	/net.sourceforge.metrics/src/net/sourceforge/...	
net.sourceforge.metrics.internal.persistence	18	4.5	4.33	12	/net.sourceforge.metrics/src/net/sourceforge/...	
net.sourceforge.metrics.internal.prevayler.implementa...	54	5.4	2.871	10	/net.sourceforge.metrics/src/net/sourceforge/...	
net.sourceforge.metrics.internal.xml	41	4.1	2.022	9	/net.sourceforge.metrics/src/net/sourceforge/...	
net.sourceforge.metrics.calculators	79	4.158	2.254	8	/net.sourceforge.metrics/src/net/sourceforge/...	
net.sourceforge.metrics.propagators	31	5.167	1.067	7	/net.sourceforge.metrics/src/net/sourceforge/...	
net.sourceforge.metrics.internal.tests	8	2.667	1.886	4	/net.sourceforge.metrics/src/net/sourceforge/...	
net.sourceforge.metrics.internal.prevayler	0	0	0			
classycle	60	8.571	2.556	13	/net.sourceforge.metrics/classycle/classycle/g...	
Lines of Code (avg/max per type)	6593	33.467	49.02	339	/net.sourceforge.metrics/tgsrc/com/touchgrap...	
Number of Interfaces (avg/max per packageFragment)	16	1	1.414	4	/net.sourceforge.metrics/src/net/sourceforge/...	
Lines of Code (avg/max per method)	6593	4.812	7.355	69	/net.sourceforge.metrics/classycle/classycle/g...	calculateAttributes
classycle	324	5.4	9.94	69	/net.sourceforge.metrics/classycle/classycle/g...	calculateAttributes
tgsrc	2321	4.661	8.278	59	/net.sourceforge.metrics/tgsrc/com/touchgrap...	scrollSelectPanel
src	3948	4.862	6.473	52	/net.sourceforge.metrics/src/net/sourceforge/...	setMetrics
net.sourceforge.metrics.ui	544	6.8	8.707	52	/net.sourceforge.metrics/src/net/sourceforge/...	setMetrics
MetricsTable.java	194	10.778	13.831	52	/net.sourceforge.metrics/src/net/sourceforge/...	setMetrics
MetricsTable	194	10.778	13.831	52	/net.sourceforge.metrics/src/net/sourceforge/...	setMetrics
setMetrics	52					

Εικόνα 5.15. Eclipse Metrics plugin - Διαθέσιμο στο: http://metrics.sourceforge.net/

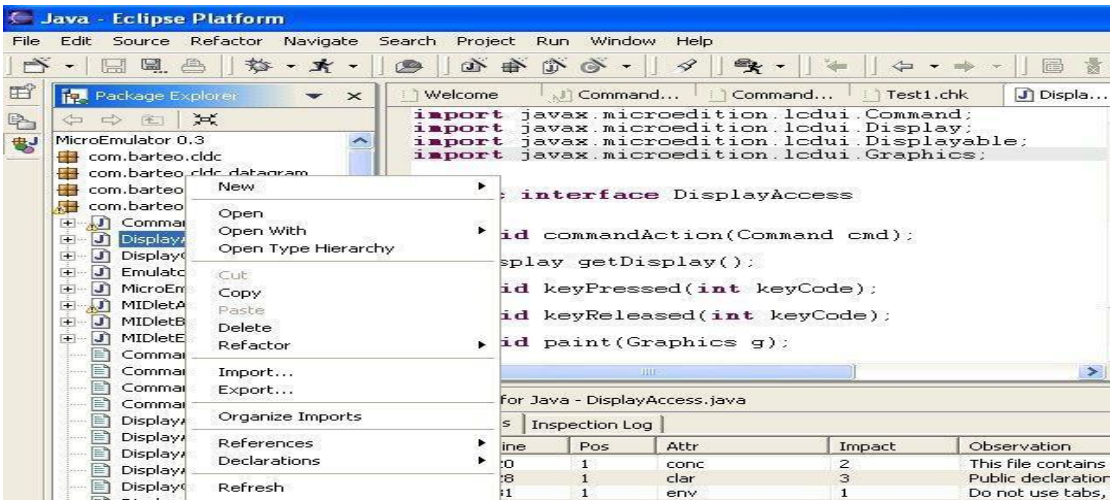


Εικόνα 5.16. Eclipse Metrics plugin – Γράφημα Συσχετίσεων. Διαθέσιμο στο: http://metrics.sourceforge.net/

5.2.2 QJ-Pro

Το QJ-Pro είναι ένα ολοκληρωμένο εργαλείο ελέγχου λογισμικού που στοχεύει στην ανάπτυξη του λογισμικού. Οι προγραμματιστές μπορούν να ελέγχουν αυτόματα τον κώδικα Java και να βελτιώσουν τις δεξιότητές τους, μέσω του προγραμματισμού Java.

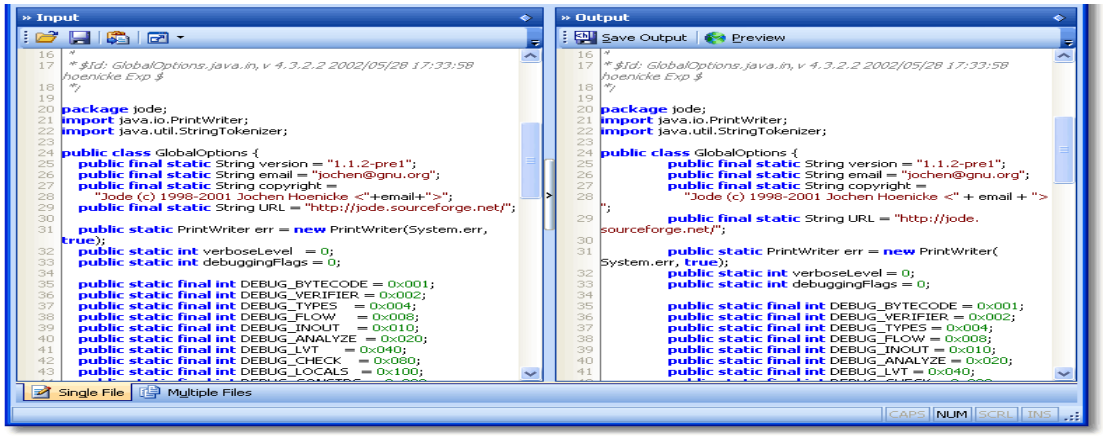
Το εργαλείο βασίζεται σε ποιοτικά χαρακτηριστικά όπως η αξιοπιστία, συντηρησιμότητα, δοκιμαστικότητα και φορητότητα, που συνήθως ορίζεται στο λεγόμενο πρότυπο κωδικοποίησης. Το QJ-Pro έρχεται με μια σειρά από καλές πρακτικές προγραμματισμού που μπορεί να εκτελεστεί αυτόματα.



Εικόνα 5.17. QJ-Pro. Διαθέσιμο στο: http://freecode.com/projects/qjpro/

5.2.3 Condenser

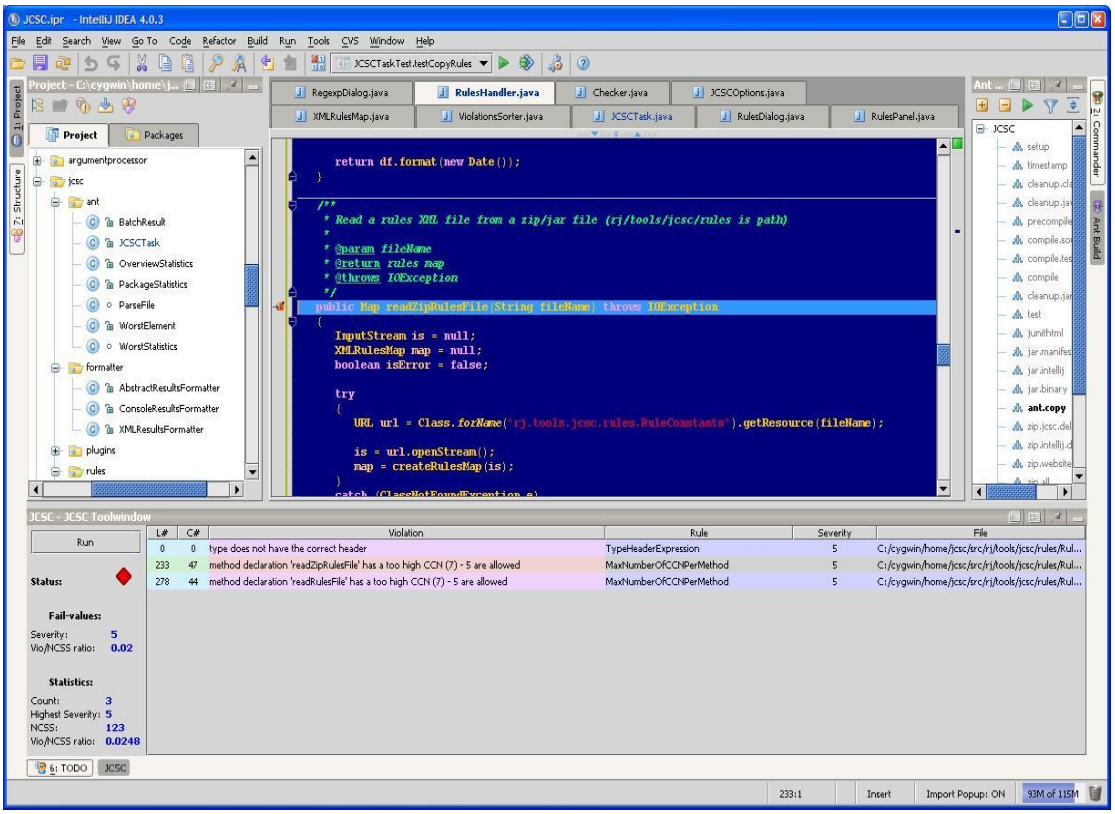
Το Condenser είναι ένα εργαλείο για την εύρεση και την αφαίρεση διπλού κώδικα Java. Σε αντίθεση με τα εργαλεία που εντοπίζουν μόνο διπλό κώδικα, ο σκοπός του Condenser είναι επίσης να αφαιρέσει αυτόματα τον διπλό κώδικα όπου είναι ασφαλές να το πράξει.



Εικόνα 5.18.Condenser. Διαθέσιμο στο: http://www.topshareware.com/Java-Code-Export-download-42963.htm

5.2.4 JCSC

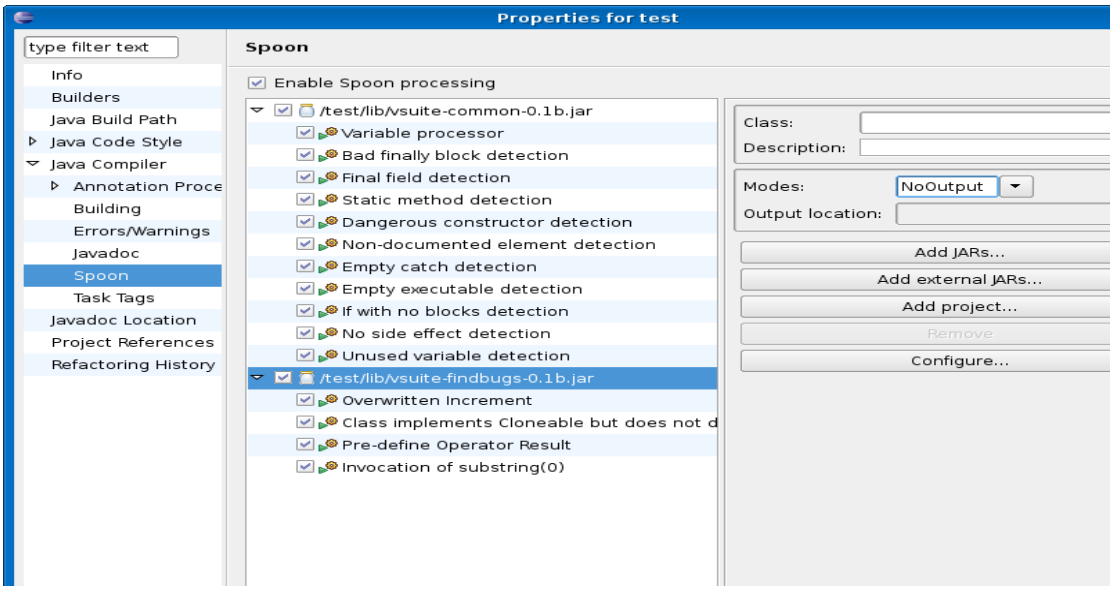
Το JCSC είναι ένα ισχυρό εργαλείο για να ελέγξετε τον πηγαίο κώδικα με ένα πολύ προσδιορίσιμο πρότυπο κωδικοποίησης. Το JCSC διαθέτει ένα γραφικό περιβάλλον εργασίας για εύκολη διαμόρφωση, μια διεπαφή γραμμής εντολών, ένα ant που παράγει μια ιστοσελίδα στυλ JavaDoc και είναι συμβατό με CruiseControl.



Εικόνα 5.19. JCSC. Διαθέσιμο στο : http://jcsc.sourceforge.net/www/screenshots.html#IntelliJPlugin

5.2.5 Spoon

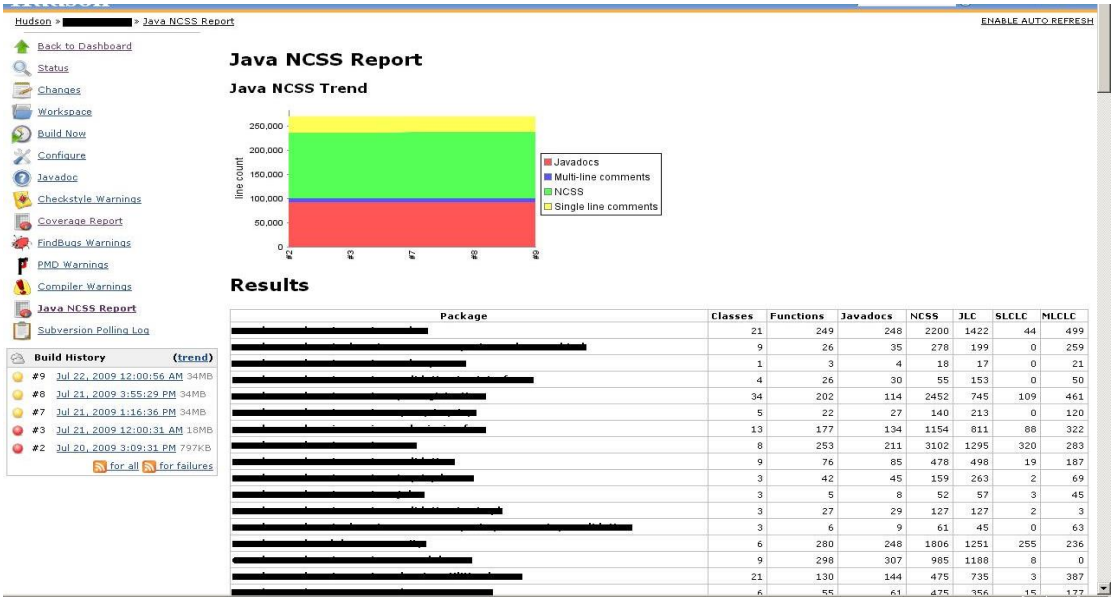
Το Spoon είναι ένας μεταγλωττιστής Java που υποστηρίζει πλήρως Java 5. Παρέχει πλήρες μεταμοντέλο Java όπου μπορεί να έχει πρόσβαση κάθε στοιχείο του προγράμματος (κατηγορίες, τις μεθόδους, τα πεδία, δηλώσεις, εκφράσεις ...) τόσο για την ανάγνωση και τροποποίηση. Ακόμα μπορεί να μπορεί να χρησιμοποιηθεί για το σκοπό της επικύρωσης, για να διασφαλιστεί ότι τα προγράμματα σας θα σέβονται κάποιες συμβάσεις προγραμματισμού ή κατευθυντήριες γραμμές, ή για τη μετατροπή του προγράμματος.



Εικόνα 5.20. Spoon. Διαθέσιμο στο: http://linux.softpedia.com/get/Programming/Interpreters/Spoon-for-Java-27016.shtml

5.2.6 JavaNCSS

Το JavaNCSS είναι ένα απλό εργαλείο γραμμής εντολών που μετρά δύο τυποποιημένες μετρήσεις κώδικα για τη γλώσσα προγραμματισμού Java. Οι μετρήσεις που συλλέγονται σε παγκόσμιο επίπεδο για κάθε κατηγορία ή / και για κάθε λειτουργία.



Εικόνα 5.21. JavaNCSS. Διαθέσιμο στο: https://java.net/projects/hudson/lists/dev/archive/2009-07/message/364

Κεφάλαιο 6 Εμπειρικές μελέτες που αφορούν την εφαρμογή των μετρικών στην ανάπτυξη των APIs

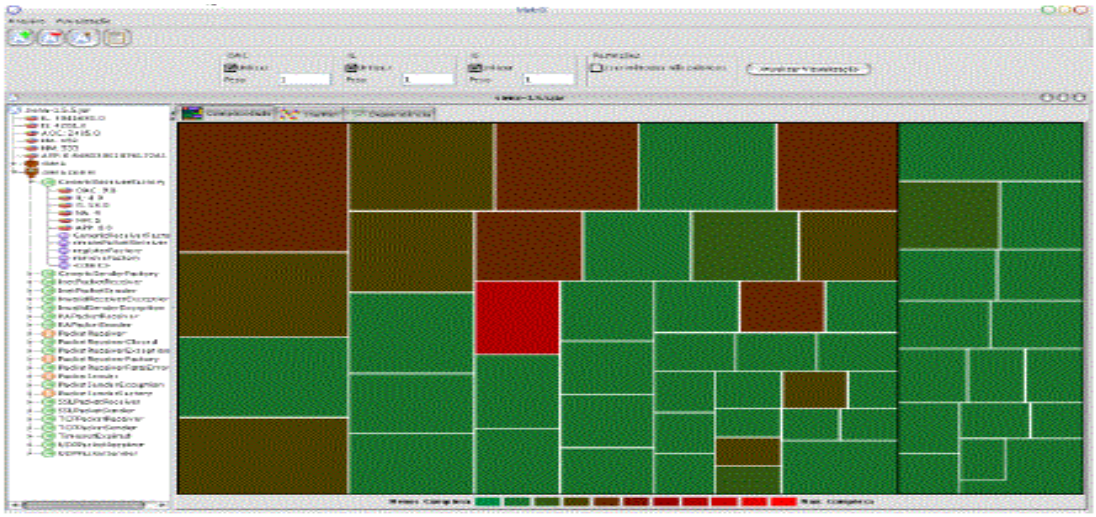
Στα πλαίσια της πτυχιακής εργασίας συγκεντρώθηκαν και κατεγράφησαν παρελθόντες μελέτες μετρικών στην ανάπτυξη των APIs. Μέσω αυτών, κατανοείται και εξακριβώνεται η εφαρμογή και η αποτελεσματικότητα των αντικειμενοστρεφών μετρικών στην ανάπτυξη ποιοτικών προγραμμάτων βιβλιοθηκών λογισμικού (Application Program Interfaces – APIs). Για την καλύτερη αξιολόγηση των παρελθόντων μελετών, χωρίστηκαν σε χρονικές περιόδους, ξεκινώντας από το 2010 έως σήμερα και παρουσιάζονται παρακάτω:

- ✓ **Κυριότερες μελέτες που υλοποιήθηκαν το 2010**

Οι Souza και Bentolila προσεγγίζουν το υπό εξέταση θέμα κάνοντας αυτόματη αξιολόγηση ευχρηστίας API χρησιμοποιώντας μετρήσεις πολυπλοκότητας και απεικονίσεις. Πιο αναλυτικά, παρουσιάζεται ένα εργαλείο, το οποίο επιτρέπει σε έναν εξυπηρετητή API να αξιολογήσει την χρησιμότητα ενός API. Επίσης, η συγκεκριμένη προσέγγιση πηγαίνει ένα βήμα μετά από τις μελέτες που ήδη έχουν γίνει, προβαίνοντας στην ανάλυση χρηστικότητας των APIs με εναλλακτικές μεθόδους.

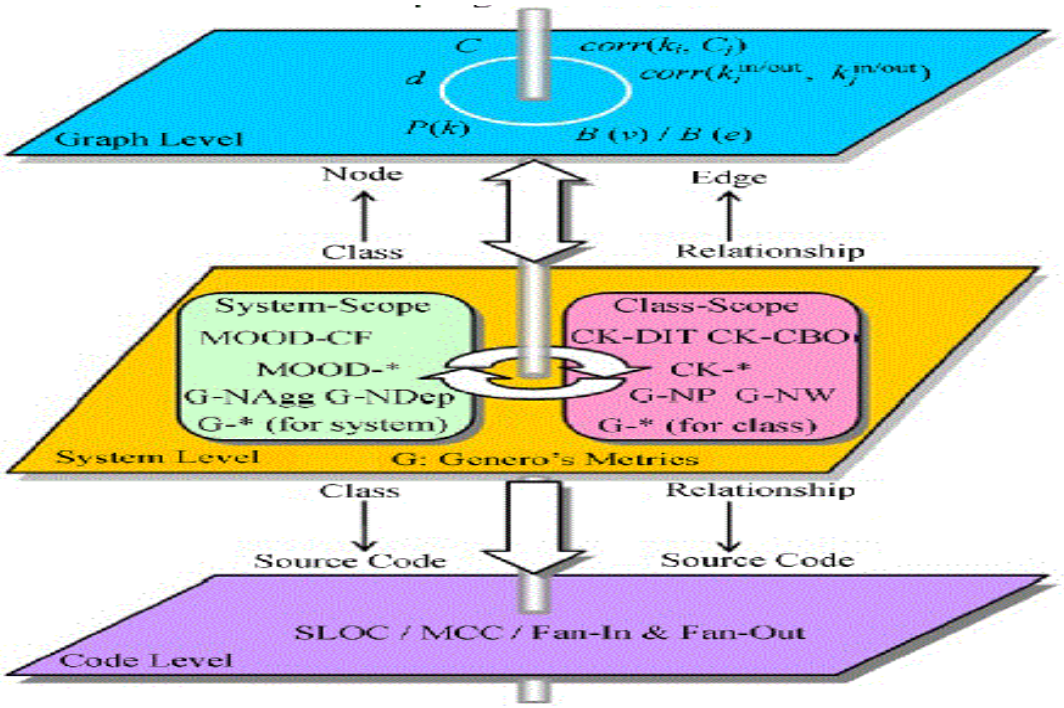
Η προσέγγιση των προαναφερθέντων βασίζεται σε δύο πτυχές, αφενός στη χρήση των μετρικών πολυπλοκότητας και αφετέρου στις τεχνικές απεικόνισης. Το εργαλείο που χρησιμοποιήθηκε ονομάζεται Metrix, το οποίο αναλύει τον ορισμό API και υπολογίζει την πολυπλοκότητα για κάθε μέθοδο του API, βάσει διαφόρων χρήσιμων πληροφοριών. Τέλος τα αποτελέσματα απεικονίζονται μέσω μια χρωματιστής κλίμακας απεικόνισης (Εικόνα 6.1) και ενός χάρτη, με μορφή δέντρου (Εικόνα 6.2).

Εικόνα 6.1. Χρηματιστή κλίμακα Απεικόνισης



Εικόνα 6.2. Χάρτης Απεικόνισης σε μορφή δέντρου

Στην συνέχεια, οι Yu-Tao Ma, Ke-Qing He, Bing Li, Jing Liu και Xiao-Yan Zhou παρουσιάζουν ένα υβριδικό σύνολο μετρικών πολυπλοκότητας για αντικειμενοστρεφής λογισμικό μεγάλης κλίμακας. Στην συγκεκριμένη μελέτη παρουσιάζονται και αξιολογούνται 12 συστήματα λογισμικού και API ανοιχτού κώδικα, αναλύοντας διάφορα επίπεδα διακριτότητας, δημιουργώντας παραγόμενες γραφικές παραστάσεις, με βάση την τάξη του αντικειμένου και τον πηγαίο κώδικα.



Εικόνα 6.3. Ιεράρχηση των μετρήσεων

Οι Khalid, Zehra και Arif πραγματοποιούν αντικειμενοστρεφής ανάλυση πολυπλοκότητας χρησιμοποιώντας μετρήσεις και μετρικές αντικειμενοστρεφούς σχεδιασμού. Η συγκεκριμένη μελέτη κάνει ανασκόπηση «δοκιμαστικότητας» και πολυπλοκότητας των συστημάτων λογισμικού στο επίπεδο του σχεδιασμού. Επίσης, για την εξαγωγή των συμπερασμάτων, τροποποιούνται προηγούμενα ερευνητικά μοντέλα ώστε να αναλυθεί λεπτομερώς η σχέση της πολυπλοκότητας και της δοκιμαστικότητας με την πρόβλεψη του επιπέδου της δοκιμαστικότητας.

Μια πολύ ενδιαφέρουσα αξιολόγηση στις παραδοσιακές και στις νέες μετρικές των αντικειμενοστρεφή συστημάτων πραγματοποιήθηκε στα μέσα του 2010 από τους Concas, Marchesi, Murgia, Pinna και Tonelli. Για τις ανάγκες της αξιολόγησης γίνεται μια εκτενή ανάλυση των μετρικών λογισμικών για 111 αντικειμενοστρεφή συστήματα, τα οποία είναι γραμμένα σε γλώσσα προγραμματισμού JAVA. Για ένα σύστημα, χρησιμοποιήθηκαν 18 παραδοσιακές μετρήσεις όπως LOC και Chidamber και Kemerer[18].

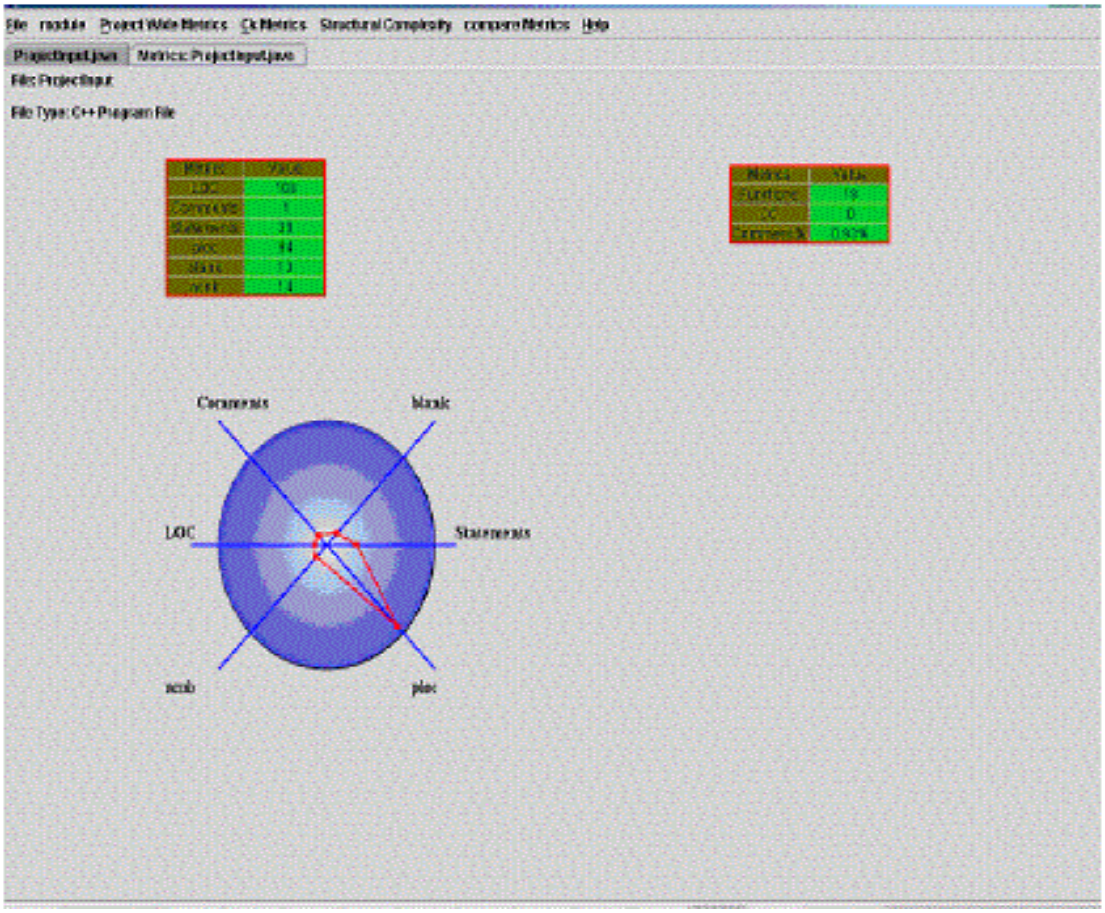
	LOCS	Fanin	Fanout	CBO	RFC	WMC	LCOM	NOC	DIT	REFF	EFFSZ	CSNS	DWRH	INFCY	SIZE	TIES	NWCP	BRKG
LOCS	1	0.23	0.74	0.57	0.81	0.75	0.56	0.04	-0.03	-0.06	0.35	0.05	0.27	-0.06	0.37	0.36	-0.14	0.23
Fanin	0.23	1	0.11	0.09	0.27	0.35	0.3	0.29	-0.09	-0.14	0.76	0.05	0.26	-0.06	0.76	0.74	-0.12	0.73
Fanout	0.74	0.11	1	0.75	0.8	0.58	0.4	0.02	0.1	-0.02	0.32	0.05	0.34	-0.08	0.35	0.36	-0.19	0.19
CBO	0.57	0.09	0.75	1	0.72	0.44	0.29	0.01	0.17	0.03	0.37	0.05	0.5	-0.12	0.41	0.4	-0.24	0.2
RFC	0.81	0.27	0.8	0.72	1	0.89	0.67	0.06	0.04	-0.05	0.43	0.06	0.38	-0.09	0.46	0.45	-0.2	0.29
WMC	0.75	0.35	0.58	0.44	0.89	1	0.8	0.08	-0.03	-0.08	0.41	0.06	0.28	-0.06	0.42	0.41	-0.14	0.3
LCOM	0.56	0.3	0.4	0.29	0.67	0.8	1	0.05	-0.02	-0.06	0.32	0.04	0.16	-0.04	0.32	0.33	-0.07	0.27
NOC	0.04	0.29	0.02	0.01	0.06	0.08	0.05	1	-0.03	-0.11	0.35	0.03	0.12	-0.03	0.34	0.28	-0.02	0.26
DIT	-0.03	-0.09	0.1	0.17	0.04	-0.03	-0.02	-0.03	1	0.15	-0.03	-0.01	0.2	-0.02	-0.01	-0.02	-0.02	-0.05
REFF	-0.06	-0.14	-0.02	0.03	-0.05	-0.08	-0.06	-0.11	0.15	1	-0.16	-0.04	0.38	-0.07	-0.16	-0.16	0.16	-0.12
EFFSZ	0.35	0.76	0.32	0.37	0.43	0.41	0.32	0.35	-0.03	-0.16	1	0.08	0.44	-0.1	0.99	0.91	-0.11	0.9
CSNS	0.05	0.05	0.05	0.05	0.06	0.06	0.04	0.03	-0.01	-0.04	0.08	1	-0.03	0.75	0.08	0.07	0.06	0.07
DWRH	0.27	0.26	0.34	0.5	0.38	0.28	0.16	0.12	0.2	0.38	0.44	-0.03	1	-0.28	0.47	0.38	0.07	0.28
INFCY	-0.06	-0.06	-0.08	-0.12	-0.09	-0.06	-0.04	-0.03	-0.02	-0.07	-0.1	0.75	-0.28	1	-0.11	-0.12	0.2	-0.07
SIZE	0.37	0.76	0.35	0.41	0.46	0.42	0.32	0.34	-0.01	-0.16	0.99	0.08	0.47	-0.11	1	0.93	-0.18	0.88
TIES	0.36	0.74	0.36	0.4	0.45	0.41	0.33	0.28	-0.02	-0.16	0.91	0.07	0.38	-0.12	0.93	1	-0.27	0.84
NWCP	-0.14	-0.12	-0.19	-0.24	-0.2	-0.14	-0.07	-0.02	-0.02	0.16	-0.11	0.06	0.07	0.2	-0.18	-0.27	1	-0.1
BRKG	0.23	0.73	0.19	0.2	0.29	0.3	0.27	0.26	-0.05	-0.12	0.9	0.07	0.28	-0.07	0.88	0.84	-0.1	1

Πίνακας 6.1. Πίνακα Συσχέτισης των μετρήσεων για κάθε σύστημα

[18] http://www.aivosto.com/project/help/pm-oo-ck.html

Συμπερασματικά, η αξιολόγηση καταλήγει ότι αφενός προκύπτουν συνεπείς συσχετίσεις μεταξύ ορισμένων μετρικών και αφετέρου στην πλειοψηφία τους, οι μετρικές είναι συσχετιζόμενες και πρακτικά είναι σχεδόν ισοδύναμες.

Συνεχίζοντας την καταγραφή των εμπειρικών μελετών για το 2010, οι Thirugnanam και Swathi παρουσιάζουν είναι χρήσιμο εργαλείο μετρικών λογισμικού για αντικειμενοστρεφή προγραμματισμό. Το προαναφερθέν εργαλείο υλοποιήθηκε για να καθορίσει τις διαφορές μετρικών λογισμικού και τα ποιοτικά χαρακτηριστικά του αντικειμενοστρεφούς προγραμματισμού, ειδικά των βιβλιοθηκών λογισμικού (APIs).



Εικόνα 6.4. Στιγμιότυπο του εργαλείου μετρικών βιβλιοθηκών λογισμικού (APIs)

Πιο αναλυτικά, το προτεινόμενο σύστημα αξιολογεί διάφορες μετρικές σχεδιασμού λογισμικού για αντικειμενοστρεφής γλώσσα προγραμματισμού όπως η C ++ και Java. Οι μετρικές θεωρούνται ως δείκτες της ποιότητας της

διαδικασίας ανάπτυξης λογισμικού. Οι μετρικές που χρησιμοποιήθηκαν είναι οι ευρέως χρησιμοποιούμενες και αξιολογούν τις κλάσεις / μεθόδους του κώδικα, την απόκριση σε μια κλάση (RFC), τον πολυμορφισμό του κώδικα, την ενθυλάκωση, τη σύζευξη μεταξύ αντικειμένων (CBO) και τέλος τον αριθμός των παιδιών (NOC).

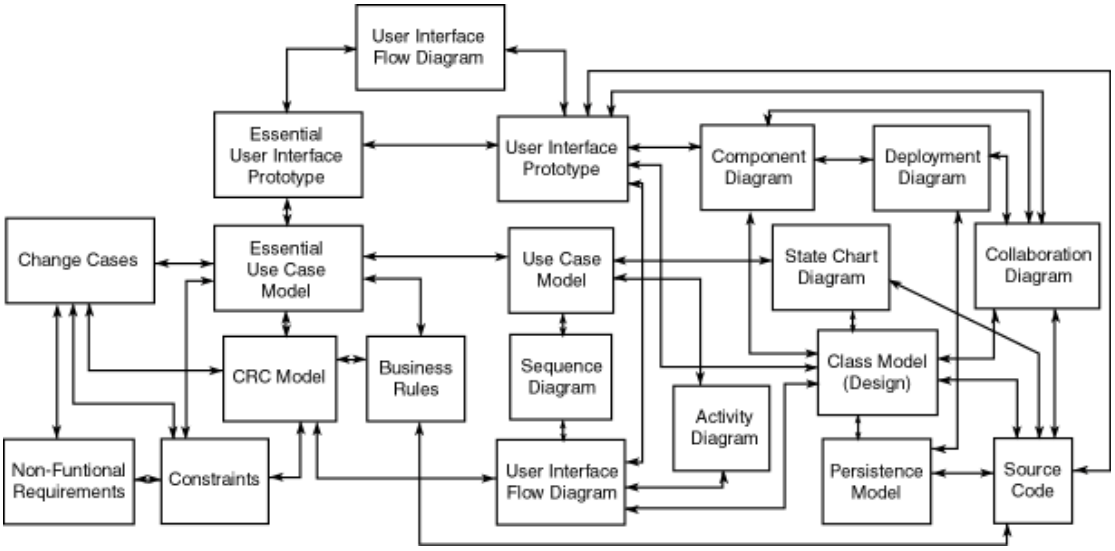
Στο συνέδριο της IEEE με θέμα «Οπτικές Γλώσσες Προγραμματισμού και Ανθρωποκεντρική Πληροφορική» παρουσιάστηκε η εργασία, στην οποία χρησιμοποιήθηκε συνδυασμός μετρικών για να βοηθήσουν τους χρήστες στην πλοήγηση της τεκμηρίωσης των APIs. Σύμφωνα με την παρούσα μελέτη, στην τελευταία πενταετία υπήρξε θεαματική αύξηση των βιβλιοθηκών λογισμικού εργαλείων, πλαισίων και προγραμματιστικών εργαλείων. Για τις ανάγκες της έρευνας, υλοποιήθηκε ένα εργαλείο, το Apatite, το οποίο δείχνει ποια στοιχεία ενός API είναι δημοφιλείς σε διαφορετικά πλαίσια και επιτρέπει την περιήγηση, επιλέγοντας μεθόδους και δράσεις σε κλάσεις και πακέτα.

Εικόνα 6.5. Εισαγωγή δεδομένων στο εργαλείο Apatite

- ✓ **Κυριότερες μελέτες που υλοποιήθηκαν το 2011**

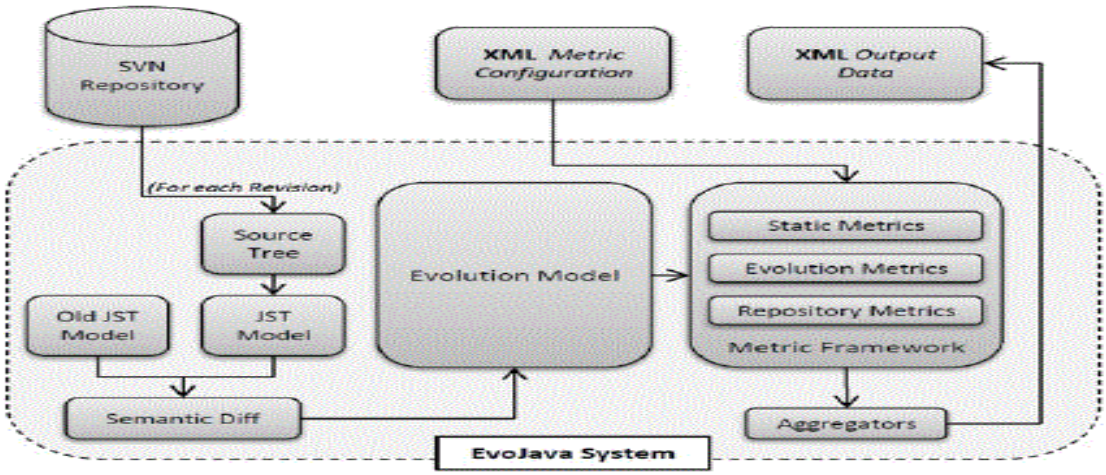
Οι Lakra, Kumar, Hooda και Bhardwaj παρουσιάζουν μια μετρική, η οποία μετρά την «ενεργητικότητα» ενός αντικειμένου μιας αντικειμενοστρεφής βιβλιοθήκης λογισμικού. Για πλαίσια της έρευνας αναπτύχθηκε ένας ειδικός

τύπος μετρικών λογισμικού που ονομάζεται Component Activeness Quotient(CAQ), όπου ορίζεται ως ο βαθμός ετοιμότητας του Object Oriented Component Library (OOCL).



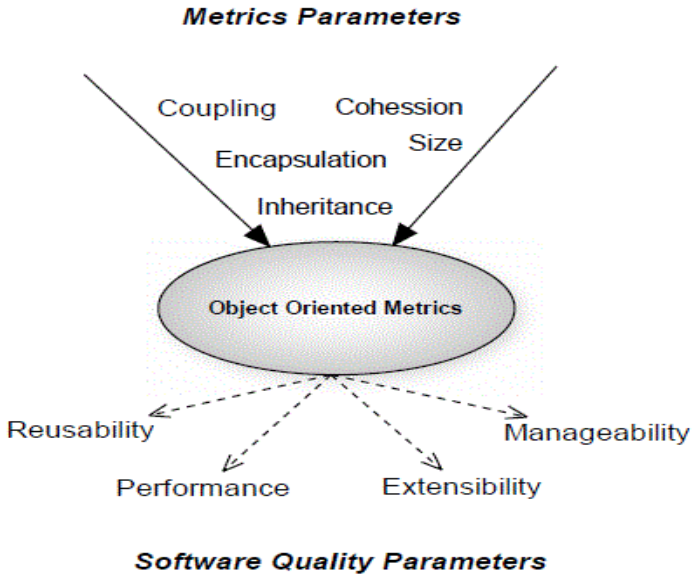
Εικόνα 6.6. Αρχιτεκτονική του Object Oriented Component Library (OOCL)

Στην συνέχεια, οι Oosterman, Irwin και Churcher παρουσιάζουν το EvoJava, ένα εργαλείο που εξάγει στατικές μετρικές αντικειμενοστρεφούς λογισμικού του πηγαίου κώδικα, γραμμένου σε Java. Πιο αναλυτικά, η εφαρμογή EvoJava δημιουργεί ένα ολοκληρωμένο μοντέλο των σημασιολογικών χαρακτηριστικών που περιγράφονται από τον κώδικα της Java (κλάσεις, μέθοδοι, επικλήσεις, κλπ), και παρακολουθεί την ταυτότητα αυτών των χαρακτηριστικών, όπως αυτά εξελίσσονται μέσω των διαδοχικών εκδόσεων.

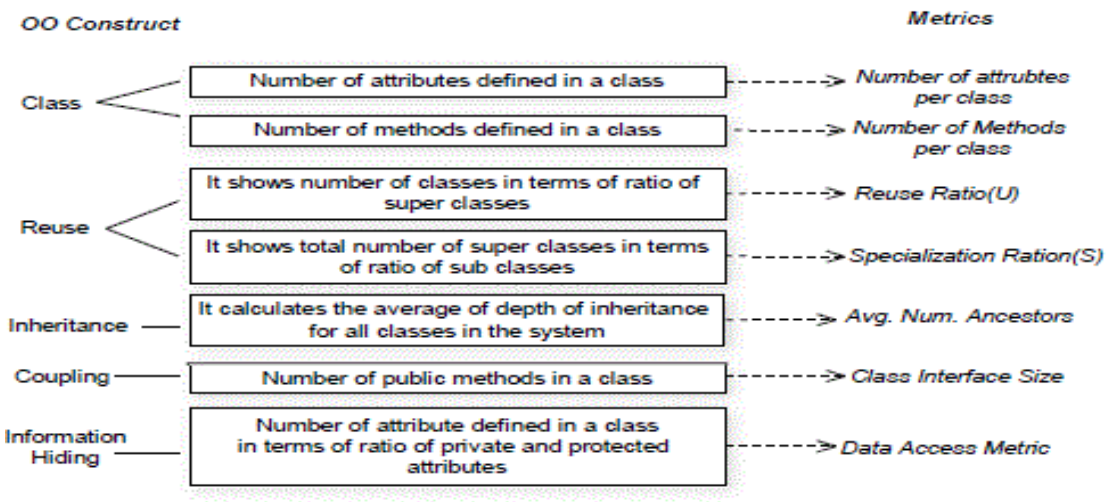


Εικόνα 6.7. Αρχιτεκτονική του Συστήματος EvoJava

Στην συνέχεια της καταγραφής των παρελθόντων ερευνών, οι Arora, Khanna, Tripathi, Sharma και Shukla κάνουν εκτίμηση της ποιότητας λογισμικού μέσω μετρικών αντικειμενοστρεφούς σχεδιασμού. Οι μετρήσεις λογισμικού που απαιτούνται για τη μέτρηση της ποιότητας αναφορικά με τις επιδόσεις και την αξιοπιστία του λογισμικού, σχετίζονται με διάφορα χαρακτηριστικά, όπως οι εξαρτήσεις, η σύζευξη και η συνοχή του λογισμικού. Παρέχει ένα τρόπο για τη μέτρηση της προόδου του κώδικα κατά τη διάρκεια της ανάπτυξης του και έχουν άμεση σχέση με το κόστος και το χρόνο που απαιτείται για τον σχεδιασμό και την ανάπτυξη του λογισμικού.

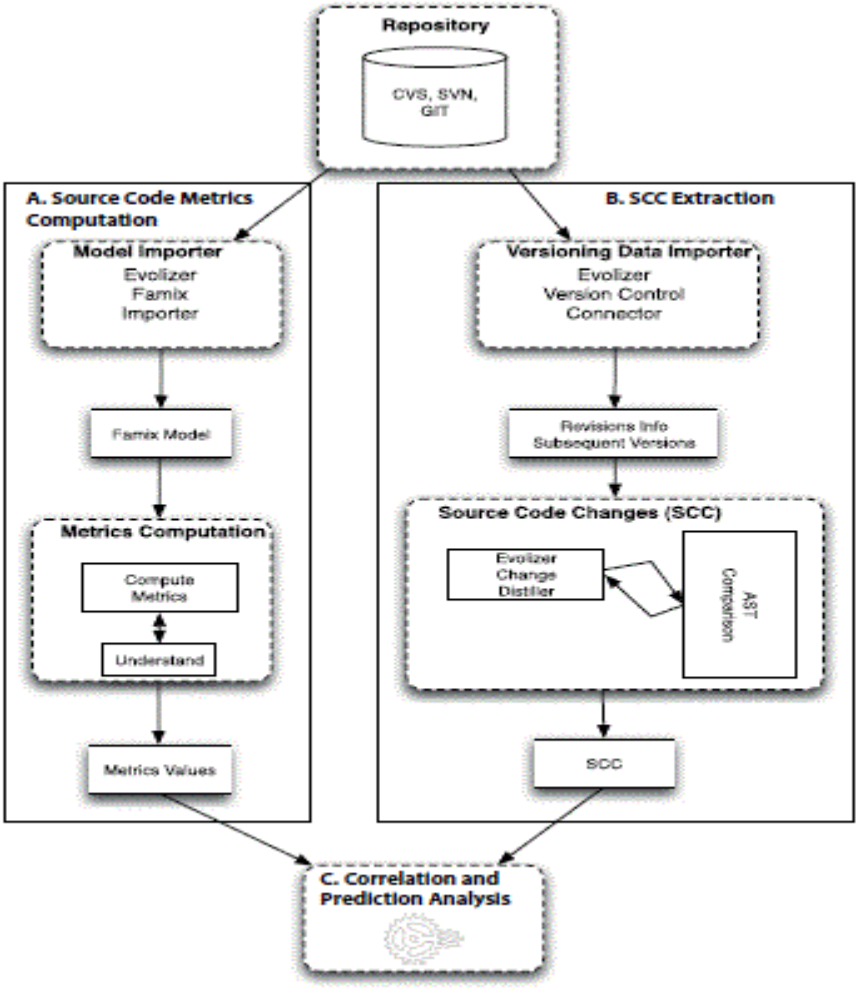


Εικόνα 6.8. Σχέση μεταξύ των μετρικών και των παραμέτρων ποιότητας



Εικόνα 6.9. Κατηγοριοποίηση Μετρικών

Τέλος, οι Romano και Pinzger χρησιμοποιούν τον πηγαίο κώδικα ώστε να προβλέψουν τις αλλαγές στις διεπαφές των Java εφαρμογών. Πρόσφατες εμπειρικές μελέτες έχουν διερευνήσει τη χρήση των μετρικών του πηγαίου κώδικα για να προβλέψουν την αλλαγή και την προδιάθεση σφάλματος στα αρχεία του πηγαίου κώδικα και των κλάσεων. Ενώ τα αποτελέσματα έδειξαν ισχυρές συσχετίσεις και καλή προβλεπτική ικανότητα αυτών των μετρήσεων, δεν κάνουν διάκριση μεταξύ διεπαφών και κλάσεων του κώδικα. Τα αποτελέσματα έδειξαν ότι οι μετρικές συνοχής της εξωτερικής διεπαφής παρουσιάζουν την ισχυρότερη συσχέτιση με τον αριθμό των αλλαγών του πηγαίου κώδικα. Αυτή η μέτρηση βελτιώνει επίσης την απόδοση των μοντέλων πρόβλεψης για την ταξινόμηση των διεπαφών Java.

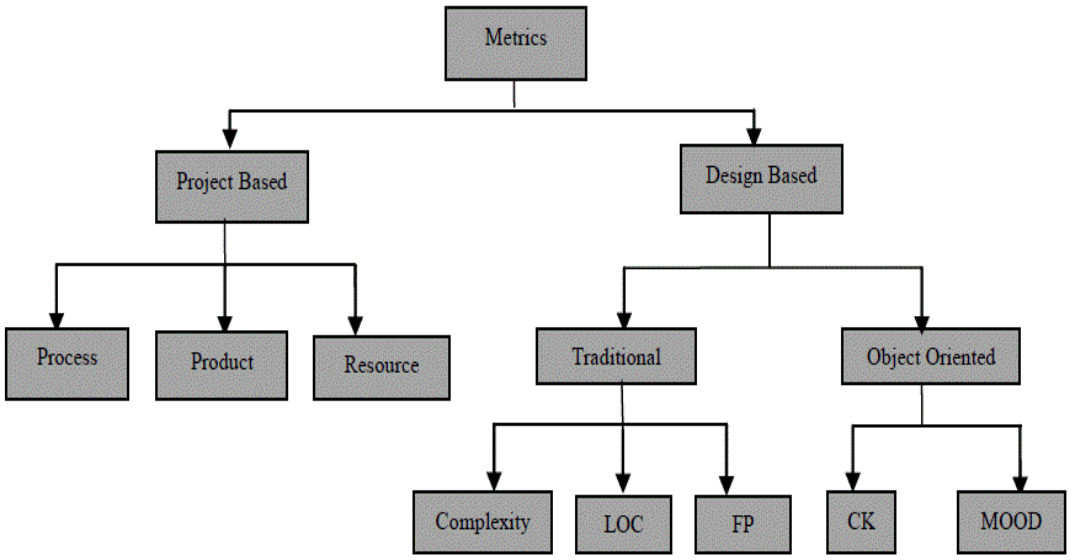


Εικόνα 6.10.Αλγόριθμος πρόβλεψης αλλαγών και προδιάθεσης σφαλμάτων διεπαφών Java

✓ **Κυριότερες μελέτες που υλοποιήθηκαν το 2012**

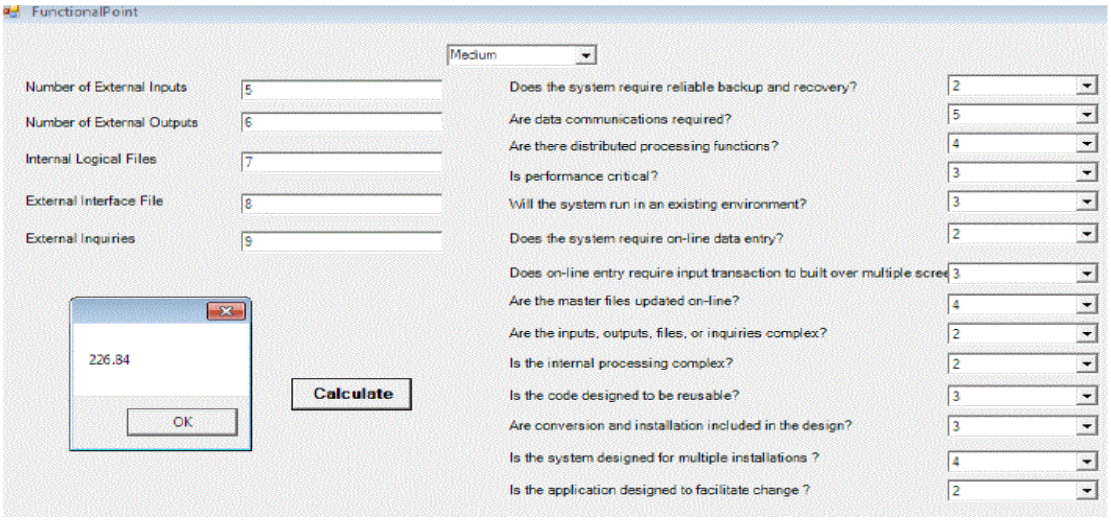
Το Διεθνές περιοδικό «Αναδυόμενες τεχνολογίες στην Υπολογιστική και Εφαρμοσμένες Επιστήμες (IJETCAS)» δημοσιεύει την εργασία των Dhawan και Kiran με θέμα «Ένα πλαίσιο για την αξιολόγηση των διαδικαστικών μετρικών και των μετρικών αντικειμενοστρεφούς λογισμικού».

Σύμφωνα με την συγκεκριμένη έρευνα, οι διαδικαστικές μετρικές μετρούν διαφορετικά χαρακτηριστικά ενός έργου ή μικρότερα κομμάτια του πηγαίου κώδικα. Για παράδειγμα, μια μέτρηση μπορεί να μετρήσει τον αριθμό των γραμμών κώδικα, την πολυπλοκότητα του κώδικα ή τον αριθμό των παρατηρήσεων (warnings). Το πλαίσιο που παρουσιάζεται στην εργασία συγκεντρώνει τις πιο επωφελής μετρικές για την μέτρηση και αξιολόγηση του πηγαίου κώδικα.



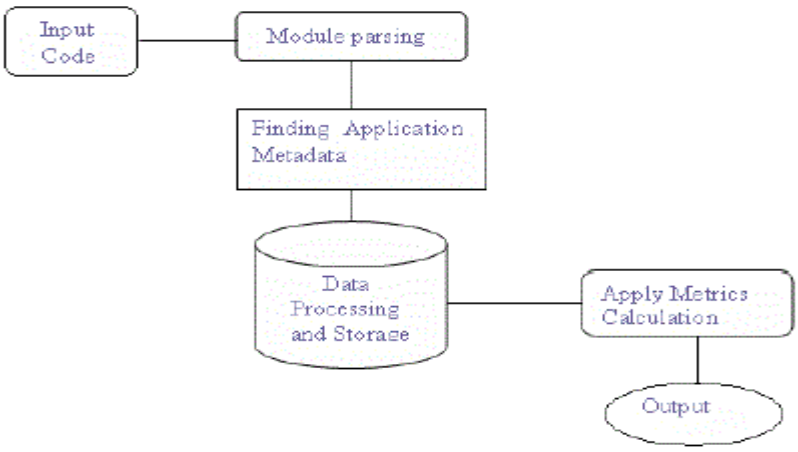
Εικόνα 6.11. Ταξινόμηση των μετρικών λογισμικού.

Το συγκεκριμένο πλαίσιο, όπως μπορούμε να δούμε και στην εικόνα 6.12, έχει την δυνατότητα φόρτωσης ενός αρχείου κώδικα και το υλοποιηθέν πρόγραμμα εξετάζει αν είναι API ή όχι και ακόμη δείχνει πώς οι ενότητες αλληλεπιδρούν μεταξύ τους μέσω της διεπαφής προγραμματισμού εφαρμογών (API).



Εικόνα 6.12. Λειτουργία ανάλυσης Σημείου διεπαφής αντικειμενοστρεφούς λογισμικού

Στην συνέχεια, οι Walde και Kulkarni παρουσιάζουν μετρικές για την μέτρηση των ενοτήτων ενός αντικειμενοστρεφούς λογισμικού. Η τρέχουσα έρευνα επικεντρώνεται στην περίπτωση της αναδιοργάνωσης της «κληρονομιάς» του λογισμικού που αποτελείται από εκατομμύρια γραμμές μη αντικειμενοστρεφή κώδικα. Στόχος είναι να αναλύσει ή να μετρήσει πόσο ο κώδικας πλαισιώνεται για το συγκεκριμένο λογισμικό και την εφαρμογή των μετρικών λογισμικού για να δείξει το επιθυμητό αποτέλεσμα. Ωστόσο, θεμελιώδη αρχή σε κάθε απόπειρα αναδιοργάνωσης του κώδικα είναι η διαίρεση του λογισμικού σε ενότητες, η δημοσίευση του API (Application Programming Interface) για τις μονάδες, και στη συνέχεια απαιτεί ότι οι μονάδες έχουν πρόσβαση σε πόρους του άλλου, μόνο μέσω των δημοσιευμένων διεπαφών.



Εικόνα 6.13. Διάγραμμα Ροής Δεδομένων

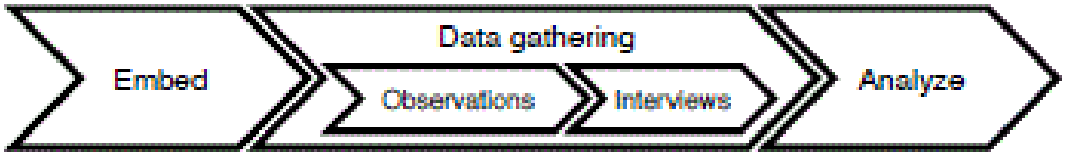
Στην συνέχεια, οι Shaik, Reddy και Damodaram παρουσιάζουν την τρέχουσα κατάσταση των μετρικών αντικειμενοστρεφούς λογισμικού, κάνοντας αξιολόγηση της κατάστασης. Πιο αναλυτικά, σύμφωνα με την έρευνα, οι προγραμματιστές λογισμικού απαιτούν ακριβείς μετρήσεις για την ανάπτυξη αποτελεσματικού συστήματος λογισμικού. Οι μετρικές και οι μετρήσεις αντικειμενοστρεφούς λογισμικού διαδραματίζουν σημαντικό ρόλο σχετικά με αυτό το θέμα, λόγω της σημασίας τους στην ανάπτυξη επιτυχημένων εφαρμογών λογισμικού. Η έρευνα παρουσιάζει εσωτερικά ακόμα διάφορες παρελθόντες έρευνες όπως: α) των Mohammad Alshayeb and Wei Li που έχουν δώσει 2 επαναληπτικές διαδικασίες για τη ρεαλιστική μελέτη των μετρικών αντικειμενοστρεφούς λογισμικού, β) των Ramanath Subramanyam and M.S. Krishnan, οι οποίοι αναφέρουν ότι οι περίπλοκες μετρικές αντικειμενοστρεφούς λογισμικού αποτελούν ένα σημαντικό ρόλο στην αναγνώριση των σφαλμάτων για την υλοποίηση του λογισμικού και γ) οι Hector M. Olague et al. εξέτασαν πειραματικά τις σουίτες σχετικά με την προδιάθεση σφαλμάτων (Chidamber and Kemerer (CK, MOOD και QMOOD).

Source	Metrics
Chidamber et al. [37]	WMC, RFC, LCOM, CBO, DIT, NOC
Lorenz et al [27]	Class size, Class inheritance, Class internal
Abreu [4]	MIF, AIF, MHF, AHF, POF, COF
Rosenberg et al. [21]	CC, LOC, CP, WMC, RFC, LCOM, CBO, DIT, NOC
Li W. et al.[43]	NAC, NLM, CMC, CMC, NDC, CTA, CTM
Bansiya et al. [10]	ANA, CAM, CIS, DAM, DCC, MOA, MFA, NOP, DSC, NOH, NOM

Εικόνα 6.14. Αντικειμενοστρεφής μετρικές από διάφορες πηγές

✓ **Κυριότερες μελέτες που υλοποιήθηκαν το 2013**

Οι Bouwers, Deursen, Visser, στην έρευνα τους αξιολογούν την χρησιμότητα των μετρικών στον αντικειμενοστρεφή λογισμικό και στις βιβλιοθήκες λογισμικού (APIs), παραθέτοντας μια αναφορά από βιομηχανική εμπειρία. Πιο αναλυτικά, η ερευνητική κοινότητα εξέτασε ένα ευρύ φάσμα μετρικών λογισμικού που στοχεύουν σε διάφορα επίπεδα αφαίρεσης και άλλα ποιοτικά χαρακτηριστικά. Για πολλές από αυτές τις μετρήσεις, η αξιολόγηση συνίσταται στην επαλήθευση των μαθηματικών ιδιοτήτων της μετρικής, ερευνά τη συμπεριφορά της μετρικής για τον αριθμό των συστημάτων ανοικτού κώδικα και συγκρίνει την τιμής της εκάστοτε μετρικής ενάντια άλλων μετρικών, αναφορικά με τα ποιοτικά χαρακτηριστικά.



Εικόνα 6.15. Μια διαδικασία τεσσάρων βημάτων για την αξιολόγηση των μετρικών λογισμικού στην πράξη

Μια εναλλακτική πρόταση παρουσιάζουν οι Jošt, Huber και Heričko με θέμα «Χρησιμοποιώντας Αντικειμενοστρεφής Μετρικές για ανάπτυξη εφαρμογών κινητών τηλεφώνων». Πιο αναλυτικά, η συγκεκριμένη έρευνα θεωρεί ότι το αντικειμενοστρεφές λογισμικό, η εκτίμηση του κόστους, ο εντοπισμός σφαλμάτων και η βελτιστοποίηση της απόδοσης μπορούν να επιτευχθούν με χρήση των μετρικών ενώ στην τεχνολογία των κινητών τα δεδομένα διαφοροποιούνται. Έτσι, η παρούσα εργασία έχει ως σκοπό να εξετάσει κατά πόσο οι παραδοσιακές μετρήσεις είναι κατάλληλες για τη μέτρηση του πηγαίου κώδικα των εφαρμογών φορητών συσκευών. Για την επίτευξη αυτού του στόχου, η εφαρμογή μικρής κλίμακας αναπτύχθηκε σε τρεις διαφορετικές πλατφόρμες (Android, iOS και Windows Phone). Ο κώδικας στη συνέχεια αξιολογήθηκε χρησιμοποιώντας τις παραδοσιακές μετρήσεις του λογισμικού. Μετά την εφαρμογή των μετρικών και ανάλυση του κώδικα, λάβαμε συγκρίσιμα αποτελέσματα, ανεξάρτητα από την πλατφόρμα.

Metrics	Android	Windows Phone	iOS
DIT (average)	2,237	0,5	0,033
NOC1 (average)	0,475	0,055	0,033
LCOM (average)	0,186	0,452	0,764
WMC (average)	7,242	15,174	4,494
NOM	249	143	191
NOC2	59	18	49
LOC	2707	554	3482
CC (average)	1,72	1,91	1,15

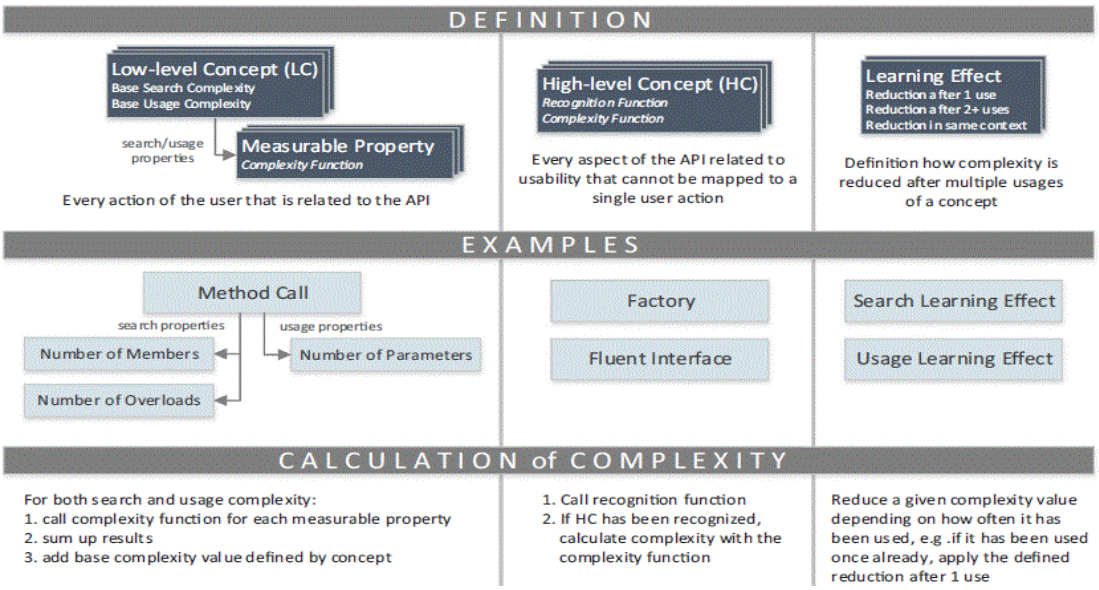
Εικόνα 6.16. Αποτελέσματα μετρικών εφαρμογών κινητών τηλεφώνων.

- ✓ **Κυριότερες μελέτες που υλοποιήθηκαν το 2014**

Το 2014 δεν παρουσιάστηκε κάποια μελέτη, σχετική με το θέμα που πραγματεύεται η παρούσα πτυχιακή εργασία.

- ✓ **Κυριότερες μελέτες που υλοποιήθηκαν το 2015**

Το 2015 παρουσιάστηκε η έρευνα των Scheller και Kühn με θέμα «Αυτοματοποιημένη μέτρηση της χρησιμότητας ενός API». Η παρούσα έρευνα έχει στόχο να κάνει την μέτρηση ευχρηστίας της διεπαφής προγραμματισμού εφαρμογών (API) ευκολότερη. Δεδομένου ότι θα ήταν αδύνατο να βρει και να ενσωματώσει όλους τους πιθανούς παράγοντες που επηρεάζουν την χρηστικότητα ενός API σε ένα βήμα, ο κύριος στόχος είναι να αποδειχθεί η σκοπιμότητα της εισαγόμενης προσέγγισης.

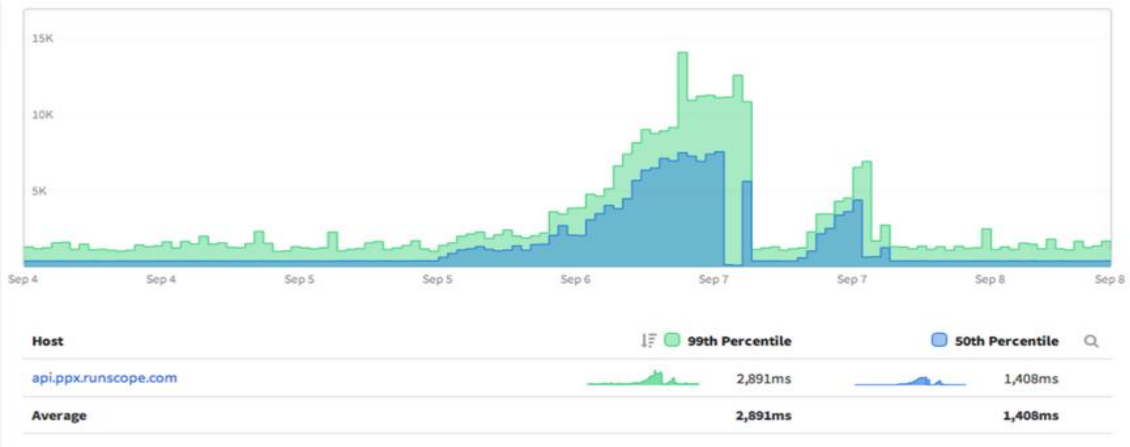


Εικόνα 6.17. Το εννοιολογικό πλαίσιο ενός API

Η τελευταία έρευνα της εμπειρικής μελέτης είναι από τους Rama και Kak και έχει ως αντικείμενο «Ορισμένα διαρθρωτικά μέτρα της χρηστικότητας ενός API». Πιο αναλυτικά, σε αυτή την εποχή της συνεργατικής ανάπτυξης λογισμικού, η σημασία της χρηστικότητας ενός API αναγνωρίζεται ευρέως. Υπάρχει ήδη μια πλούσια βιβλιογραφία που ασχολείται με την σχεδίαση των APIs. Ωστόσο, ακόμη δεν υπάρχουν μετρικές γενικής χρήσης που προβαίνουν σε μια πιο ποσοτική αξιολόγηση της χρηστικότητας ενός API. Η παρούσα έρευνα παρουσιάζει ένα σύνολο τύπων που εξετάζουν μεθόδους δηλώσεων ενός API από την πλευρά αρκετών διαδεδομένων αντιλήψεων, σχετικά με το τι κάνει ένα API δύσκολο στην χρήση.

Οι έρευνα κατέληξε ότι οι μετρήσεις χρηστικότητας ενός API κυμαίνονται από το χαμηλό επίπεδο στο υψηλό επίπεδο. Στο χαμηλό επίπεδο, ασχολήθηκαν με το πώς ονομάζονται οι μέθοδοι, πώς ομαδοποιούνται οι μέθοδοι, πώς οι παράμετροι ονομάζονται και πώς ομαδοποιούνται οι παράμετροι. Στο υψηλό επίπεδο, ασχολήθηκαν με ζητήματα που σχετίζονται με τα «νήματα» ασφάλειας - αυτό ισχύει μόνο για τα API που μπορεί να χρησιμοποιηθεί για τις πολυνηματικές εφαρμογές. Οι τιμές των μετρικών χρησιμεύουν ως δείκτες έγκαιρης προειδοποίησης μειωμένης ποιότητας API. Τέλος, αν επισημανθεί από τις μετρήσεις ότι ένα

API βρίσκεται σε χαμηλή ποιότητα και δεν μπορεί να βελτιωθεί η ποιότητα του, μπορεί να ενημερωθεί με βοηθητικές πληροφορίες για να προειδοποιήσει και να βοηθήσει τους χρήστες του API.



Εικόνα 6.18. Αποτελέσματα Μετρήσεων APIs, χαμηλό και υψηλό επίπεδο.

7 Συμπεράσματα

Η παρούσα πτυχιακή εργασία είχε ως αντικείμενο την ανασκόπηση των εμπειρικών μελετών ως προς την εφαρμογή και αποτελεσματικότητα των αντικειμενοστρεφών μετρικών στην ανάπτυξη ποιοτικών προγραμμάτων βιβλιοθηκών λογισμικού (Application Program Interfaces – APIs). Στο θεωρητικό μέρος αναλύθηκε η έννοια της τεχνολογίας λογισμικού, παρουσιάστηκαν οι μετρήσεις, οι μετρικές αντικειμενοστρεφούς λογισμικού και τα διαθέσιμα εργαλεία μετρικών βιβλιοθηκών λογισμικού JAVA. Στο ερευνητικό μέρος, πραγματοποιήθηκε εμπειρική μελέτη ανασκόπησης, όπου κατεγράφησαν και αναλύθηκαν 30 έρευνες, χρονολογούμενες από το 2010 έως το 2015, ενώ στο ερευνητικό μέρος οι 17 εξ αυτών αναλύθηκαν σχολαστικά και παρουσιάστηκαν στο έκτο κεφάλαιο.

Η βιβλιοθήκες λογισμικού αποτελούν πλέον είναι αναπόσπαστο τμήμα του δομημένου προγραμματισμού και αναπτύχθηκε παράλληλα με αυτόν καθώς περιέχουν υποβοηθητικό κώδικα και δεδομένα, παρέχοντας, με αυτόν τον τρόπο, υπηρεσίες σε προγράμματα. Έτσι, είναι οφθαλμοφανές ότι η λειτουργία τους είναι ιδιαίτερα σημαντική στην λειτουργία και χρησιμότητα των εφαρμογών και για τον λόγο αυτών η χρήση των μετρικών στα APIs είναι κομβικής σημασίας.

Σύμφωνα με τους προγραμματιστές εφαρμογών, οι εφαρμογές γίνονται καλύτερες και πιο αξιόπιστες, όσο βελτιώνονται οι διαδικασίες αποσφαλμάτωσης και οι μετρικές των εφαρμογών. Στο θεωρητικό μέρος της παρούσας πτυχιακής παρουσιάστηκαν τα σημαντικότερα εργαλεία μετρικών, τα οποία κατηγοριοποιήθηκαν και αναλύθηκαν. Η εμπειρική μελέτη ανασκόπησης έδειξε ότι πολλοί ερευνητές έχουν ασχοληθεί με την αποτελεσματικότητα των μετρικών στην ποιότητα ανάπτυξης των APIs.

Συμπερασματικά, η εμπειρική μελέτη ανασκόπησης έδειξε ότι σημαντικό θέμα στις μετρικές είναι η εμπειρία, καθώς όσο μεγαλύτερη εμπειρία έχουν οι μετρικές, τόσο καλύτερα αποτελέσματα εξάγονται. Σύμφωνα

με αυτήν την θεωρία, οι μετρικές που χρησιμοποιούνται στην ανάπτυξη του λογισμικού πρέπει να ξεκινούν από το χαμηλό επίπεδο, και πρέπει να συνεχίζουν στη μέτρηση των κλάσεων και τάξεων του πηγαίου κώδικα. Ακόμα, μια άλλη προσέγγιση έγινε, με σκοπό να συνδυαστούν οι μετρικές, σε μία ενιαία, συνοπτική μετρική. Αυτό ο τρόπος θα βοηθούσε τους προγραμματιστές ώστε να λαμβάνουν υπόψη μια ενιαία τιμή, η οποία θα υποδείκνυε το επίπεδο της ποιότητας σχεδιασμού. Βέβαια, το σκεπτικό αυτό, αυτόαπορρίφτηκε καθώς πολλοί προγραμματιστές ενδιαφερόντουσαν μόνο για κάποιες συγκεκριμένες μετρικές.

Τέλος, οι μετρικές των βιβλιοθηκών αντικειμενοστρεφούς προγραμματισμού, κερδίζουν ολοένα και περισσότερο «έδαφος» και αυτό έχει ως συνέπεια να υλοποιούνται νέες μετρικές και να αναβαθμίζονται οι υπάρχουσες. Μελλοντική προοπτική της εργασίας είναι αφενός, η διερεύνηση και ανάλυση περισσότερων μετρικών Α/Σ λογισμικού και αφετέρου η εμπειρική μελέτη ανασκόπησης από το 2015 και μετά.

Βιβλιογραφία

Ξένη Βιβλιογραφία

[1] Basili V.R., Briand L.C. and Melo W.L.,(1996) “A Validation of Object-Oriented Design Metrics as Quality Indicators”, IEEE Transactions on Software Engineering, SE-22, Volume 10

[2] Bellin D., Tyagi M. and Tyler M.,(1999) “Object Oriented Metrics: An Overview”, Web Publication

[3] Boehm B.W., Brown J.R., Kaspar J.R.,(1978) Lipow M., McCleod G.J. and Merrit M.J., “Characteristics of Software Quality”, North Holland,.

[4] Chidamber R.S. and Kemerer C.F.,(1991) “Towards a metrics suite for object–oriented design”, Proceedings of the OOPSLA ’91 Conference, pp. 197-211

[5] Dromey R.G., (1995) “A Model for Software Product Quality”, IEEE Transactions on Software Eng., Vol. 21, 2, pp. 146-162.

[6] Fenton N.E.,(1992) “Software Metrics. A Rigorous Approach”, Chapman & Hall, ISBN: 0-442-31355-1

[7] Hambling B.,(1996) “Managing Software Quality”, McGraw-Hill, London

[8] Hudli R., Hoskins C. and Hudli A.,(1994) “Software Metrics for Object Oriented Designs”, IEEE

[9] Ince D.,(1993) “An Introduction to Software Quality Assurance and Its Implementation”, McGraw-Hill, Maidenhaid

[10] Kafura D. and Reddy R.,(1987) “The Use of Software Complexity Metrics in Software Maintenance”, IEEE Trans. Software Engineering, vol. SE-13, no. 3, pp. 335-343

[11] Kan S., (1996), “Metrics and Models in Software Quality Engineering”, Addison-Wesley

[12] Kolewe R., (1993), “Metrics in Object-Oriented Design and Programming”, Software Development

[13] Lorenz M. and Kidd J.,(1994) “Object–Oriented Software Metrics”, Prentice Hall

[14] Weyuker E.,(1998) “Evaluating Software Complexity Measures”, IEEE Transactions on Software Engineering, SE-14 (9), pp. 1357-1365

[15] Conte, S., Dunsmore, V. & Shen, V. (1986). Software Engineering Metrics and Models. The Benjamin/Cummings Publishing Company

Ελληνική Βιβλιογραφία

[16] Σταυρινούδης Δ., (1995), “Πειραματική μελέτη του βαθμού συσχέτισης εσωτερικών και εξωτερικών μετρικών ποιότητας λογισμικού”, Διπλωματική Εργασία, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πανεπιστήμιο Πατρών

[17] Τσαλίδης Χ (1990), “Ολοκληρωμένη Μέτρηση Ποιοτικών Χαρακτηριστικών Λογισμικού Ανεξάρτητα από Γλώσσα Προγραμματισμού”, Διδακτορική Διατριβή, Πανεπιστήμιο Πατρών

[18] Ξένος Μ. και Χριστοδουλάκης Δ., (1994) “Τεχνολογία Λογισμικού. Αρχές και Μεθοδολογίες”, Εκδόσεις Πανεπιστημίου Πατρών

[19] Ξένος Μ.,(1996) ,“Μεθοδολογία Ελέγχου και Εξασφάλισης της Ποιότητας Λογισμικού Βασισμένη στις Μετρικές Προϊόντος και στα Εξωτερικά Ποιοτικά Χαρακτηριστικά του Λογισμικού”, Διδακτορική Διατριβή, Πανεπιστήμιο Πατρών

[20] Παπαδόπουλος Κ. (2002), “Κωδικοποίηση και εφαρμογή αντικειμενοστραφών μετρικών και τεχνικών μετρήσεων ποιότητας λογισμικού”, Διπλωματική εργασία Τμήματος Μηχανικών Η/Υ & Πληροφορικής

Παράρτημα Α

Οι υπό – εξέταση παρελθόντες έρευνες που χρησιμοποιήθηκαν στην εμπειρική μελέτη της παρούσας πτυχιακής.

International Journal of Computer Theory and Engineering (IJCTE)	**Quality Metrics Tool for Object Oriented Programming**
Information and Software Technology	**Automated Measurement of API Usability: The API Concepts Framework**
International Journal on Computer Science and Engineering	**Measuring the quality of object oriented software Modularization: Defining Metrics and Algorithm**
International Journal of Computer Science and Network Security	**Software Quality Estimation through Object Oriented Design Metrics**
IEEE Xplore Digital Library	**Metrics for Measuring the Quality of Modularization of Large-Scale Object-Oriented Software**
Wiley Online Library	**A suite of metrics for quantifying historical changes to predict future change-prone classes in object-oriented software**
ACM Digital Library	**Analysis of Object Oriented Complexity and Testability Using Object Oriented Design Metrics**
Cornell University Library	**A Metric For The Activeness Of An Object-Oriented Component Library**
Sabinet Reference	**An empirical validation of object oriented design metrics in object oriented systems**
ACM Digital Library	**An empirical analysis of a testability model for object-oriented programs**
Science Direct	**Identifying thresholds for object-oriented software metrics**
Springer	**A Hybrid Set of Complexity Metrics for Large-Scale Object-Oriented Software Systems**
International Journal of Advanced Research in Computer Science and Software Engineering	**Coupling Based Structural Metrics-An Quality Assessment of Software Modularization**
Wiley Online Library	**Some structural measures of API usability**
International Journal of Emerging Technologies in Computational and Applied Sciences	**A Framework for Evaluating the Procedural and Object Oriented Software Metrics**
IEEE Xplore Digital Library	**API-Based and Information-Theoretic Metrics for Measuring the Quality of Software Modularization**

Scientific Research	**Empirical Analysis of Object-Oriented Design Metrics for Predicting Unit Testing Effort of Classes**
International Journal of Advanced Research in Computer Science and Software Engineering	**Object Oriented Software Metrics and Quality Assessment: Current State of the Art**
International Journal of Management, IT and Engineering	**Software Reliability Measurement Based On API and Information Theoretic Metrics**
ACM Digital Library	**Assessing traditional and new metrics for object-oriented systems**
IEEE Xplore Digital Library	**Modularization Metrics: Assessing Package Organization in Legacy Large Object-Oriented Software**
Citeseerx	**Software Defects and Object Oriented Metrics – An Empirical Analysis**
IEEE Xplore Digital Library	**Using source code metrics to predict change-prone Java interface s**
International Journal of Computer Applications	**Metrics for measuring the quality of object oriented software modularization**
SQAMIA 2013-2nd Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications	**Using Object Oriented Software Metrics for Mobile Application Development**
School of Computer Science	**Using Association Metrics to Help Users Navigate API Documentation**
TU Delft	**Evaluating Usefulness of Software Metrics - an Industrial Experience Report -**
Conferences in Research and Practice in Information Technology -http://crpit.com/	**EvoJava: A Tool for Measuring Evolving Software**
IEEE Xplore Digital Library	**Automatic evaluation of API usability using complexity metrics and visualizations**