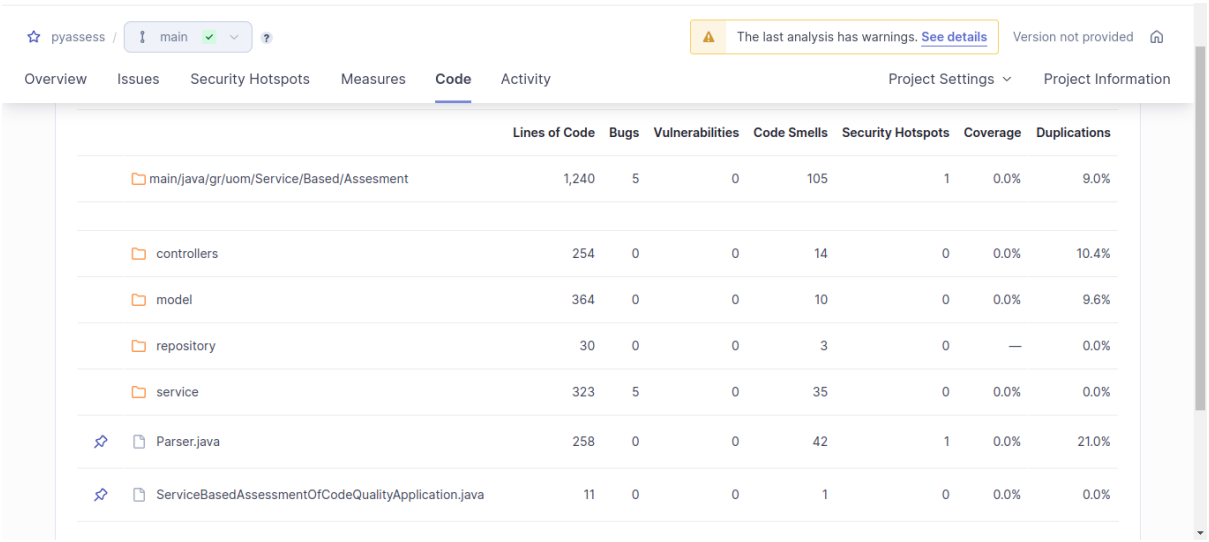
«Εντοπισμός Code Smells ή/και Παραβιάσεων Αρχών Σχεδίασης, Refactorings, Git»

Γεώργιος Δαυίδ Αποστολίδης
mai24002
Εργασία 1
Μάθημα: Τεχνολογία Συστημάτων Λογισμικού
Κατεύθυνση: Ανάπτυξη Λογισμικού και Νέφος

Αποστολίδης Γεώργιος Δαυίδ
mai240022
Εργασία 1
Τεχνολογία Συστημάτων Λογισμικού
Μεταπτυχιακό στην Εφαρμοσμένη Πληροφορική
Κατεύθυνση Ανάπτυξης Λογισμικού και Νέφος

Βήμα 1

Εκτελούμε την ανάλυση για τα Java αρχεία στο φάκελο *../backend/src* και παίρνουμε τα εξής αποτελέσματα:



	Lines of Code	Bugs	Vulnerabilities	Code Smells	Security Hotspots	Coverage	Duplications
main/java/gr/uom/Service/Based/Assesment	1,240	5	0	105	1	0.0%	9.0%
controllers	254	0	0	14	0	0.0%	10.4%
model	364	0	0	10	0	0.0%	9.6%
repository	30	0	0	3	0	—	0.0%
service	323	5	0	35	0	0.0%	0.0%
Parser.java	258	0	0	42	1	0.0%	21.0%
ServiceBasedAssessmentOfCodeQualityApplication.java	11	0	0	1	0	0.0%	0.0%

Όπως παρατηρείται εδώ, υπάρχουν πάνω από 100 code smell στον κώδικα. Ας εξετάσουμε 5 από αυτά.

1. ProjectController.java, γραμμή 179

Όπως παρατηρούμε, η μεταβλητή “analysis” δηλώνεται στην αρχή του βρόχου, αλλά δεν χρησιμοποιείται πουθενά εντός του βρόχου. Της ανατίθεται μια τιμή από τη συλλογή projectAnalysis σε κάθε επανάληψη του βρόχου, αλλά δεν εξυπηρετεί κανέναν σκοπό.

Μια αχρησιμοποίητη μεταβλητή όπως αυτή, προσθέτει περιττή ακαταστασία στον κώδικα. Μπορεί να κάνει τον κώδικα πιο δύσκολο να διαβαστεί και να κατανοηθεί, ειδικά για άλλους προγραμματιστές που μπορεί να αναρωτιούνται γιατί η μεταβλητή δηλώνεται αλλά δεν

χρησιμοποιείται. Επιπλέον, οι περιττές μεταβλητές μπορεί να επηρεάσουν τη συντηρησιμότητα του κώδικα. Κατά την αναθεώρηση ή την τροποποίηση του κώδικα στο μέλλον, οι προγραμματιστές μπορεί να σπαταλήσουν χρόνο προσπαθώντας να καταλάβουν τον σκοπό αυτών των μεταβλητών, γεγονός που μπορεί να οδηγήσει σε σύγχυση.

2. ProjectAnaylsisService.java, γραμμή 121

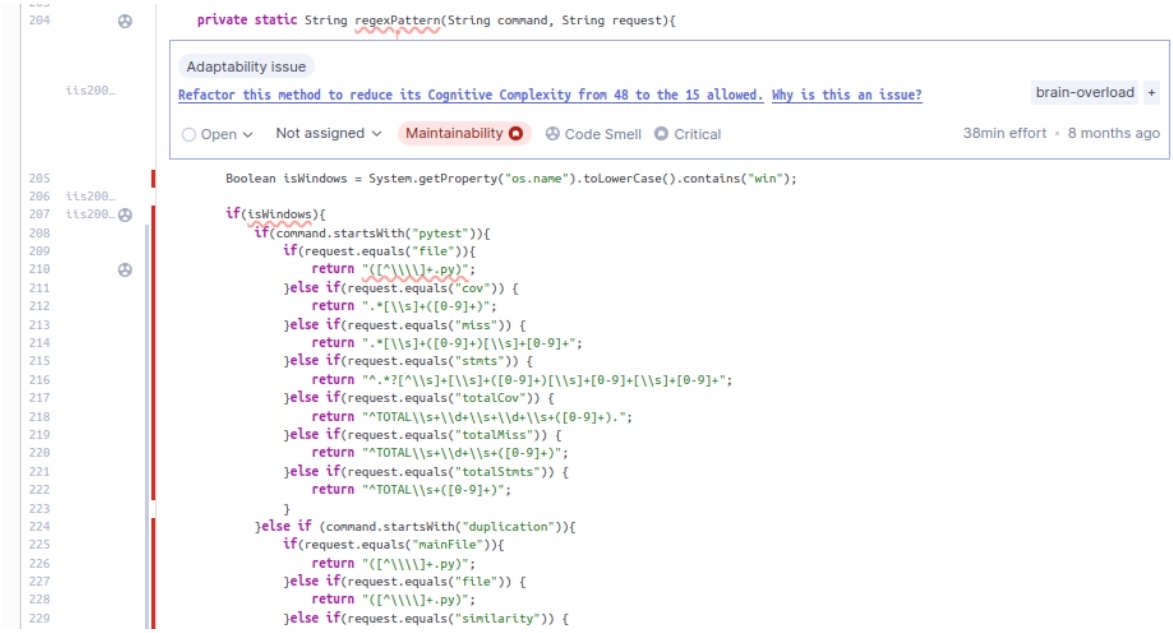
```
120
121 iis200…    public void executeCommand(ProjectAnalysis projectAnalysis, ArrayList<ProjectFile> fileList, String command, String destination) throws IOException, InterruptedException {
122 iis200…        ArrayList<String> similarityResponse = new ArrayList<>();
123                ArrayList<String> commentsResponse = new ArrayList<>();
124                Process p = Runtime.getRuntime().exec(command + destination);
125                InputStream is = p.getInputStream();
126                try (BufferedReader reader = new BufferedReader(new InputStreamReader(is))) {
127                    List<String> lines = reader.lines().collect(Collectors.toList());
128                    for (String line : lines) {
129 iis200…                if(command.startsWith("python3 -W ignore " + System.getProperty("user.dir")+ File.separator + "duplicate-code-detection-tool/duplicate_code_detection.py -d  ")){
130 iis200…                    if(line.contains("Code duplication probability for") || line.startsWith(projectAnalysis.getDirectory())){
131 iis200…                        similarityResponse.add(line);
132                            }
133                        }else if(command.startsWith("pylint")){
134 iis200…                    if(line.contains("Your code has been rated at") || line.startsWith("************* Module ") || line.startsWith(projectAnalysis.getName())) {
135 iis200…                        commentsResponse.add(line);
136                            }
137 iis200…                }else if(command.startsWith("pytest")){
138 iis200…                    if(line.startsWith(projectAnalysis.getName()) || line.startsWith("TOTAL")){
139 iis200…                        storeDataInObjects(projectAnalysis, fileList, line, command);
140 iis200…                    }
141 iis200…                }
142                        System.out.println(line);
143                    }
144                    if (similarityResponse.size()>0){
145 iis200…                storeSimilarity(similarityResponse, fileList, projectAnalysis);
146 iis200…                }
147                    if (commentsResponse.size()>0) {
148 iis200…                storeComments(commentsResponse, fileList, projectAnalysis);
149 iis200…                }
150                }
151
152            }
```

Το πρόβλημα στο παραπάνω code smell σύμφωνα με το SonarQube είναι: *Refactor this method to reduce its Cognitive Complexity from 19 to the 15 allowed.* Η γνωστική πολυπλοκότητα είναι ένα μέτρο του πόσο πολύπλοκη είναι η κατανόηση μιας μεθόδου. Σε αυτή τη μέθοδο, η γνωστική πολυπλοκότητα υπολογίζεται σε 19, η οποία είναι υψηλότερη από το επιτρεπόμενο όριο των 15 σύμφωνα με τα πρότυπα κωδικοποίησης του SonarQube.

Η μέθοδος φαίνεται να εκτελεί πολλαπλές εργασίες, συμπεριλαμβανομένης της εκτέλεσης μιας εξωτερικής εντολής, της ανάλυσης της εξόδου της και της αποθήκευσης δεδομένων. Αυτό παραβιάζει την Αρχή της Ενιαίας Ευθύνης (Single Responsibility Principle), μια αρχή σχεδιασμού από το SOLID, η οποία υποδηλώνει ότι μια μέθοδος θα πρέπει να έχει μόνο έναν λόγο αλλαγής. Σε αυτή την περίπτωση, υπάρχουν πολλοί λόγοι αλλαγής (π.χ. αν αλλάξει η μορφή της εντολής ή αν αλλάξει η λογική αποθήκευσης δεδομένων), γεγονός που καθιστά τον κώδικα λιγότερο συντηρήσιμο.

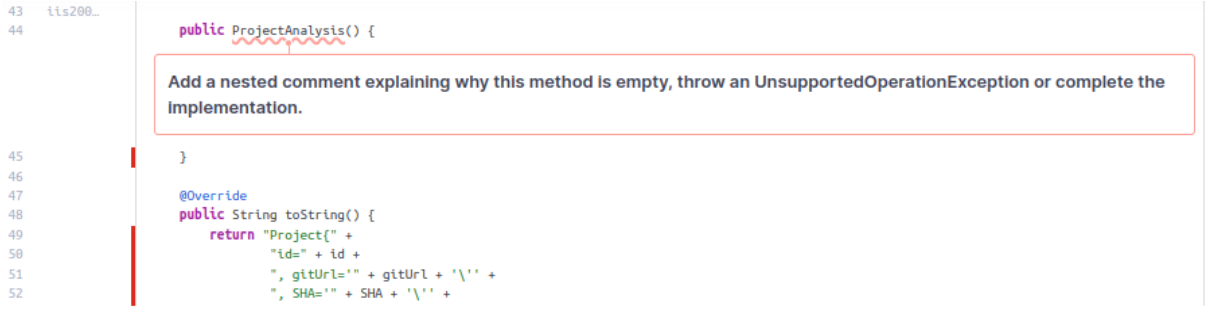
Αποστολίδης Γεώργιος Δαυίδ
mai240024
Εργασία 1
Τεχνολογία Συστημάτων Λογισμικού
Μεταπτυχιακό στην Εφαρμοσμένη Πληροφορική
Κατεύθυνση Ανάπτυξης Λογισμικού και Νέφος

3. Parser.java, γραμμή 204



Στο ίδιο μοτίβο, υπάρχει μία μέθοδος στο αρχείο Parser.java η οποία διαχειρίζεται τα αποτελέσματα που βγαίνουν από την κονσόλα. Επειδή το project διαχειρίζεται Python source code analyzers τα αποτελέσματα των εκτελέσεων του κώδικα βγαίνουν στην κονσόλα και σε ένα print line το οποίο μπορούμε να το δούμε μέσα στην Java. Όταν λοιπόν παίρνουμε μία μία την γραμμή στην οποία εκτελούνται αυτοί οι analyzers περνάει μέσα από αυτό το regex το οποίο φιλτράρει την γραμμή για να καταλάβει και να συλλάβει τις πληροφορίες που θέλει. Επειδή οι περισσότερες γραμμές είναι “σταθερές” δημιουργείται αυτή η μεγάλη μέθοδος στην οποία πρέπει να ελεγχθούν κάποιες συνθήκες, όπως ποια εντολή έχει χρησιμοποιηθεί, ποιο στοιχείο πρέπει να αποθηκεύσουμε σε κάθε περίπτωση κλπ ανεβάζοντας την γνωστική πολυπλοκότητα στο 48, νούμερο τρεις φορές μεγαλύτερο από αυτο που προτείνει το SonarQube.

4. ProjectAnalysis.java, γραμμή 44



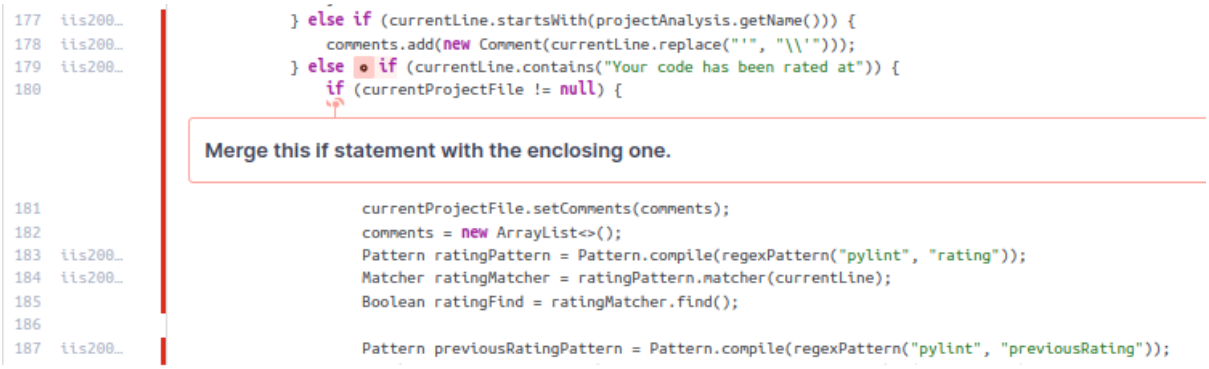
Αποστολίδης Γεώργιος Δαυίδ
mai240025
Εργασία 1
Τεχνολογία Συστημάτων Λογισμικού
Μεταπτυχιακό στην Εφαρμοσμένη Πληροφορική
Κατεύθυνση Ανάπτυξης Λογισμικού και Νέφος

Το πρόβλημα στον κώδικα σχετίζεται με την παρουσία ενός κενού κατασκευαστή στην κλάση ProjectAnalysis και η SonarQube προτείνει την προσθήκη ενός εμφωλευμένου σχολίου που εξηγεί γιατί η μέθοδος είναι κενή, την απόρριψη μιας UnsupportedOperationException ή την ολοκλήρωση της υλοποίησης.

Η κλάση ProjectAnalysis περιέχει έναν κενό κατασκευαστή, πράγμα που σημαίνει ότι δεν εκτελεί καμία ουσιαστική λειτουργία όταν δημιουργείται μια περίπτωση της κλάσης. Αυτό το

ζήτημα δεν σχετίζεται άμεσα με το σπάσιμο ενός συγκεκριμένου προτύπου σχεδίασης. Ωστόσο, μπορεί να επηρεάσει την ποιότητα του κώδικα και τη συντηρησιμότητα, που αποτελούν σημαντικές πτυχές του σχεδιασμού λογισμικού. Η παρουσία ενός κενού κατασκευαστή αφορά περισσότερο την ποιότητα κώδικα και τη συντηρησιμότητα παρά τα πρότυπα σχεδίασης.

5. Parser.java, γραμμή 179



Στον κώδικα, υπάρχει μια εμφωλευμένη δήλωση if, όπου η εξωτερική δήλωση if ελέγχει αν η τρέχουσα γραμμή αρχίζει με projectAnalysis.getName() και η εμφωλευμένη δήλωση if ελέγχει αν η τρέχουσα γραμμή περιέχει τη συγκεκριμένη συμβολοσειρά "Your code has been rated at".

Οι εμφωλευμένες εντολές if μπορούν να αυξήσουν την πολυπλοκότητα του κώδικα και να τον καταστήσουν πιο δυσνόητο. Προσθέτει ένα επιπλέον επίπεδο εσοχής, το οποίο μπορεί να προκαλέσει οπτική σύγχυση. Η δομή του κώδικα μπορεί να επηρεάσει αρνητικά την αναγνωσιμότητα του κώδικα. Οι προγραμματιστές μπορεί να χρειαστεί να παρακολουθούν τη λογική και των δύο συνθηκών και η εμφωλευμένη εντολή μπορεί να δυσχεράνει την παρακολούθησή της.

Αποστολίδης Γεώργιος Δαυίδ
mai240026
Εργασία 1
Τεχνολογία Συστημάτων Λογισμικού
Μεταπτυχιακό στην Εφαρμοσμένη Πληροφορική
Κατεύθυνση Ανάπτυξης Λογισμικού και Νέφος

Βήμα 2

Σε αυτό το σημείο θα κάνουμε τις κατάλληλες αναδομήσεις στον κώδικα για να μπορέσουμε να επιλύσουμε τα θέματα και τις οσμές που αναφέραμε στο Βήμα 1.

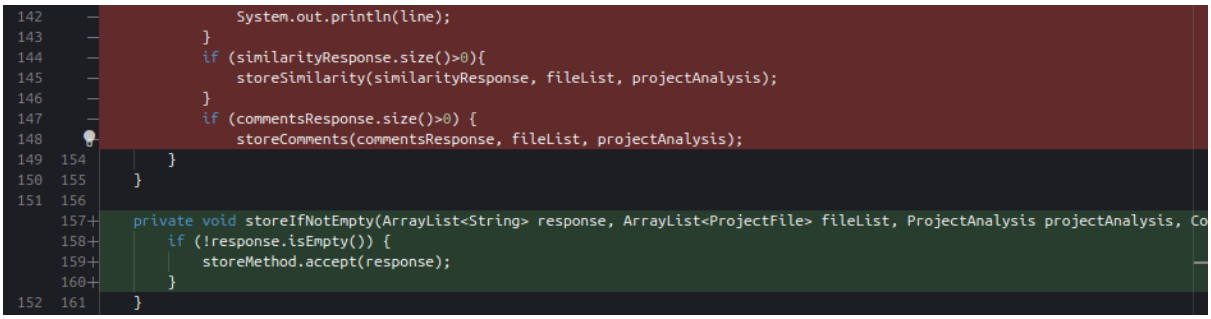
Στην **1ή περίπτωση** η μόνη αλλαγή που θα κάνουμε είναι να αφαιρέσουμε την πρώτη for η οποία δεν επηρεάζει σε καμία περίπτωση το υπόλοιπο μέρος του κώδικά. Οπότε οι αλλαγές θα είναι ως εξής:

```
178 178     List<ProjectFile> files = appProjectAnalysisService.getProjectFilesByName(project.get().getName());
179     -   for (ProjectAnalysis analysis : projectAnalysis) {
180     -       int number = 0;
181     -       for (ProjectFile file : files) {
182     -           if (file.getName().equals(fileName)) {
183     -               responseComFiles.add(file.getComments());
184     -               number++;
185     -           }
186     -       }
187     -       System.out.println(number);
    179 +   int number = 0;
    180 +   for (ProjectFile file : files) {
    181 +       if (file.getName().equals(fileName)) {
    182 +           responseComFiles.add(file.getComments());
    183 +           number++;
    184 +       }
    185 +   }
    186 +   System.out.println(number);
188     -   }
```

Στην **2η περίπτωση** η μέθοδος αναλύεται σε μικρότερες, πιο εστιασμένες μεθόδους για να μειωθεί η πολυπλοκότητα.
Εισάγεται η handleLine για τον χειρισμό της λογικής υπό συνθήκη για διαφορετικές εντολές, καθιστώντας την καλύτερη ανάγνωση. Ακόμη,εισάγεται μια μέθοδος storeIfNotEmpty για την αποθήκευση δεδομένων μόνο εάν η λίστα απαντήσεων δεν είναι κενή, βελτιώνοντας την αναγνωσιμότητα.
Αυτές οι αλλαγές καθιστούν τον κώδικα πιο σπονδυλωτό, ευανάγνωστο και ευκολότερο στη συντήρηση, ενώ μειώνουν τη γνωστική πολυπλοκότητα.

```
128 131         for (String line : lines) {
129     -           if(command.startsWith("python3 -W ignore " + System.getProperty("user.dir")+ File.separator + "duplicate-code-det
130     -               if(line.contains("Code duplication probability for") || line.startsWith(projectAnalysis.getDirectory())){
    132 +           handleLine(line, projectAnalysis, similarityResponse, commentsResponse, fileList);
    133 +           System.out.println(line);
    134 +       }
    135 +   }
    136 +
    137 +   storeIfNotEmpty(similarityResponse, fileList, projectAnalysis, this::storeSimilarity);
    138 +   storeIfNotEmpty(commentsResponse, fileList, projectAnalysis, this::storeComments);
    139 +}
    140 +
    141 +private void handleLine(String line, ProjectAnalysis projectAnalysis, ArrayList<String> similarityResponse, ArrayList<String>
    142 +   if (command.startsWith("python3 -W ignore " + System.getProperty("user.dir") + File.separator + "duplicate-code-detection
    143 +       if (line.contains("Code duplication probability for") || line.startsWith(projectAnalysis.getDirectory())) {
131 144             similarityResponse.add(line);
132 145         }
133     -       }else if(command.startsWith("pylint")){
134     -           if(line.contains("Your code has been rated at") || line.startsWith("************* Module ") || line.startsWit
    146 +   } else if (command.startsWith("pylint")) {
    147 +       if (line.contains("Your code has been rated at") || line.startsWith("************* Module ") || line.startsWith(proje
135 148             commentsResponse.add(line);
136 149         }
137     -       }else if(command.startsWith("pytest")){
138     -           if(line.startsWith(projectAnalysis.getName()) || line.startsWith("TOTAL")){
    150 +   } else if (command.startsWith("pytest")) {
    151 +       if (line.startsWith(projectAnalysis.getName()) || line.startsWith("TOTAL")) {
139 152             storeDataInObjects(projectAnalysis, fileList, line, command);
140 153         }
141     -       }
```

Αποστολίδης Γεώργιος Δαυίδ
mai240027
Εργασία 1
Τεχνολογία Συστημάτων Λογισμικού
Μεταπτυχιακό στην Εφαρμοσμένη Πληροφορική
Κατεύθυνση Ανάπτυξης Λογισμικού και Νέφος



```
142 —                System.out.println(line);
143 —            }
144 —            if (similarityResponse.size()>0){
145 —                storeSimilarity(similarityResponse, fileList, projectAnalysis);
146 —            }
147 —            if (commentsResponse.size()>0) {
148 —                storeComments(commentsResponse, fileList, projectAnalysis);
149 154      }
150 155  }
151 156
    157+  private void storeIfNotEmpty(ArrayList<String> response, ArrayList<ProjectFile> fileList, ProjectAnalysis projectAnalysis, Co
    158+      if (!response.isEmpty()) {
    159+          storeMethod.accept(response);
    160+      }
152 161  }
```

Στην **3η περίπτωση** χρησιμοποιείται ένας Χάρτης που ονομάζεται REGEX_PATTERNS για την αποθήκευση μοτίβων regex για διαφορετικούς συνδυασμούς εντολών και αιτήσεων. Τα μοτίβα οργανώνονται με βάση το λειτουργικό σύστημα (Windows ή Unix). Η μέθοδος regexPattern ελέγχει το λειτουργικό σύστημα και ανακτά τα κατάλληλα μοτίβα από το χάρτη REGEX_PATTERNS, απλοποιώντας την υπό συνθήκη λογική και μειώνοντας την πολυπλοκότητα.
Αυτή η αναδιοργάνωση απλοποιεί τον κώδικα και τον καθιστά πιο συντηρήσιμο, καθώς συγκεντρώνει τα μοτίβα regex και εξαλείφει την ανάγκη για εμφωλευμένες υπό συνθήκη εντολές.

```
private static final Map<String, Map<String, String>> REGEX_PATTERNS = new HashMap<>();

static {
    Map<String, String> windowsPatterns = new HashMap<>();
    Map<String, String> unixPatterns = new HashMap<>();

    windowsPatterns.put("file", "([^\\\\]+.py)");
    windowsPatterns.put("cov", ".*[\\s]+([0-9]+)");
    windowsPatterns.put("miss", ".*[\\s]+([0-9]+)[\\s]+[0-9]+");
    windowsPatterns.put("stmts", "^.*?[^\\s]+[\\s]+([0-9]+)[\\s]+[0-9]+[\\s]+[0-9]+");
    windowsPatterns.put("totalCov", "^TOTAL\\s+\\d+\\s+\\d+\\s+([0-9]+).");
    windowsPatterns.put("totalMiss", "^TOTAL\\s+\\d+\\s+([0-9]+)");
    windowsPatterns.put("totalStmts", "^TOTAL\\s+([0-9]+)");

    windowsPatterns.put("mainFile", "([^\\\\]+.py)");
    windowsPatterns.put("file", "([^\\\\]+.py)");
    windowsPatterns.put("similarity", "m([+-]?[0-9]*\\.?[0-9]+(?:[eE][+-]?[0-9]+)?)");

    windowsPatterns.put("file", "\\b(\\w+)$");
    windowsPatterns.put("rating", "at (\\b[0-9]+\\.[0-9]+)/10");
    windowsPatterns.put("previousRating", "run: (\\b[0-9]+\\.[0.9]+)/10");

    unixPatterns.put("file", "([^/]+.py)");
    unixPatterns.put("cov", ".*\\s+\\d+\\s+\\d+\\s+(\\d+)%");
    unixPatterns.put("miss", ".*\\s+\\d+\\s+(\\d+)\\s+\\d+%");
    unixPatterns.put("stmts", ".*\\s+(\\d+)\\s+\\d+\\s+\\d+%");
    unixPatterns.put("totalCov", "^TOTAL\\s+\\d+\\s+\\d+\\s+([0-9]+).");
    unixPatterns.put("totalMiss", "^TOTAL\\s+\\d+\\s+([0-9]+)");
```

Αποστολίδης Γεώργιος Δαυίδ
mai240028
Εργασία 1
Τεχνολογία Συστημάτων Λογισμικού
Μεταπτυχιακό στην Εφαρμοσμένη Πληροφορική
Κατεύθυνση Ανάπτυξης Λογισμικού και Νέφος

```java
private static String regexPattern(String command, String request) {
    Boolean isWindows = System.getProperty("os.name").toLowerCase().contains("win");
    Map<String, String> patterns = REGEX_PATTERNS.get(isWindows ? "Windows" : "Unix");

    if (patterns != null && patterns.containsKey(request)) {
        return patterns.get(request);
    }

    return "";
}
```

Στην **4η περίπτωση** για να αντιμετωπίσουμε το πρόβλημα, θα πρέπει να προσθέσουμε ένα ενσωματωμένο σχόλιο που να εξηγεί γιατί ο κατασκευαστής είναι άδειος. Αυτή η τεκμηρίωση παρέχει σαφήνεια σε άλλους προγραμματιστές και υποδεικνύει ότι ο άδειος κατασκευαστής είναι σκόπιμος. Είναι κοινή πρακτική να προσθέτουμε σχόλια σε τέτοιες περιπτώσεις για να αποφύγουμε τη σύγχυση.

```java
43
    George - David Apostolidis
44  public ProjectAnalysis() {
45      // This empty constructor is provided for Hibernate and should not be used for general object creation.
46  }
47
```

Στην **5η** και τελευταία **περίπτωση** χρειάστηκε να γίνει γενικότερη αλλαγή όλης της μεθόδου. Αναλυτικά, ο κώδικας για το χειρισμό των γραμμών "Module" μεταφέρθηκε σε μια ξεχωριστή μέθοδο handleModuleLine για να απλοποιηθεί ο κύριος βρόχος. Ο κώδικας για το χειρισμό των γραμμών "Rating" μεταφέρεται σε ξεχωριστή μέθοδο handleRatingLine. Αυτή η μέθοδος αναλαμβάνει επίσης τη ρύθμιση των αξιολογήσεων και των σχολίων για το τρέχον αρχείο έργου. Οι πρώιμες επιστροφές χρησιμοποιούνται για την αποφυγή εμφωλευμένων εντολών if και τη βελτίωση της αναγνωσιμότητας του κώδικα.

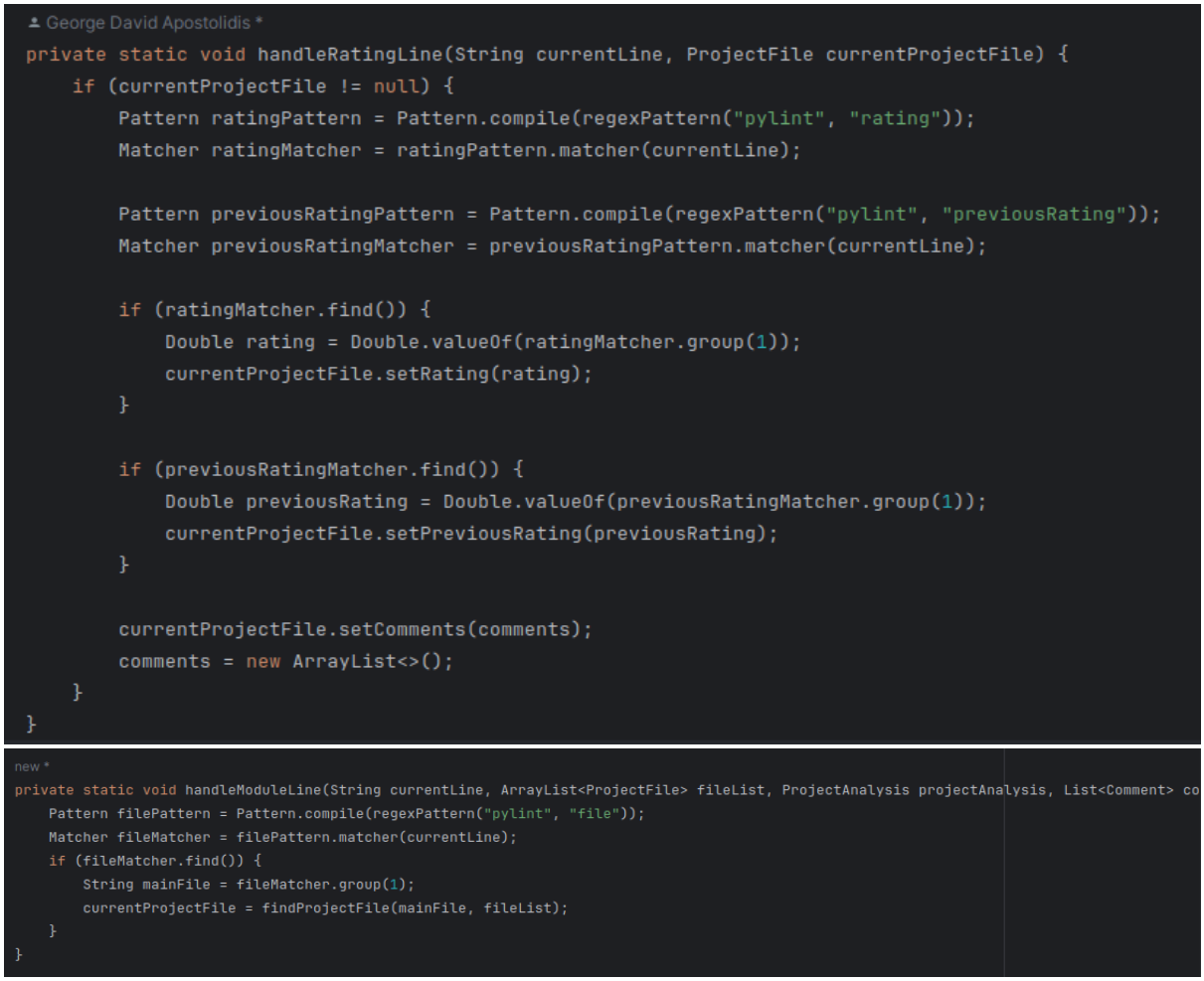
Αυτή η αναδιοργάνωση καθιστά τον κώδικα πιο κατανοητό με το σπασιμο του σε μικρότερες, πιο εστιασμένες μεθόδους και την αφαίρεση των εμφωλευμένων δηλώσεων υπό συνθήκη.

```java
George David Apostolidis +1 *
public static void storeComments(ArrayList<String> commentsResponse, ArrayList<ProjectFile> fileList, ProjectAnalysis projectAnalysis) {
    String mainFile = "";
    List<Comment> comments = new ArrayList<>();
    ProjectFile currentProjectFile = null;

    for (int i = 0; i < commentsResponse.size(); i++) {
        String currentLine = commentsResponse.get(i);

        if (currentLine.startsWith("************ Module")) {
            handleModuleLine(currentLine, fileList, projectAnalysis, comments, currentProjectFile);
        } else if (currentLine.startsWith(projectAnalysis.getName())) {
            comments.add(new Comment(currentLine.replace("'", "\\'")));
        } else if (currentLine.contains("Your code has been rated at")) {
            handleRatingLine(currentLine, currentProjectFile);
        }
    }
}
```

Αποστολίδης Γεώργιος Δαυίδ
mai240029
Εργασία 1
Τεχνολογία Συστημάτων Λογισμικού
Μεταπτυχιακό στην Εφαρμοσμένη Πληροφορική
Κατεύθυνση Ανάπτυξης Λογισμικού και Νέφος



```
George David Apostolidis *
private static void handleRatingLine(String currentLine, ProjectFile currentProjectFile) {
    if (currentProjectFile != null) {
        Pattern ratingPattern = Pattern.compile(regexPattern("pylint", "rating"));
        Matcher ratingMatcher = ratingPattern.matcher(currentLine);

        Pattern previousRatingPattern = Pattern.compile(regexPattern("pylint", "previousRating"));
        Matcher previousRatingMatcher = previousRatingPattern.matcher(currentLine);

        if (ratingMatcher.find()) {
            Double rating = Double.valueOf(ratingMatcher.group(1));
            currentProjectFile.setRating(rating);
        }

        if (previousRatingMatcher.find()) {
            Double previousRating = Double.valueOf(previousRatingMatcher.group(1));
            currentProjectFile.setPreviousRating(previousRating);
        }

        currentProjectFile.setComments(comments);
        comments = new ArrayList<>();
    }
}
```

```
new *
private static void handleModuleLine(String currentLine, ArrayList<ProjectFile> fileList, ProjectAnalysis projectAnalysis, List<Comment> co
    Pattern filePattern = Pattern.compile(regexPattern("pylint", "file"));
    Matcher fileMatcher = filePattern.matcher(currentLine);
    if (fileMatcher.find()) {
        String mainFile = fileMatcher.group(1);
        currentProjectFile = findProjectFile(mainFile, fileList);
    }
}
```

Βήμα 3

Για το συγκεκριμένο βήμα παραθέτω το link όπου φαίνονται οι 5 τελευταίες αλλαγές στο branch με όνομα “tss-assignment” και σας παραθέτω στον παρακάτω πίνακα τα sha με το όνομα τους.

Αποστολίδης Γεώργιος Δαυίδ
mai2400210
Εργασία 1
Τεχνολογία Συστημάτων Λογισμικού
Μεταπτυχιακό στην Εφαρμοσμένη Πληροφορική
Κατεύθυνση Ανάπτυξης Λογισμικού και Νέφος

Name	Commit
Revision: Solve Issue #1	7d0e1ead65e3784c9496db9ed0ba14967557d99b
Revision: Solve Issue #2	937950f490e994494fb20c0753e23834eee30dd2
Revision: Solve Issue #3	f304f826b3cee67f106fdd523f02b5eaeaa5f141
Revision: Solve Issue #4	b93cb80145341efe675e88d2cacf9e3722141d73
Revision: Solve Issue #5	71273f80e0692dbea92a42c3891308f3e36a1ab9

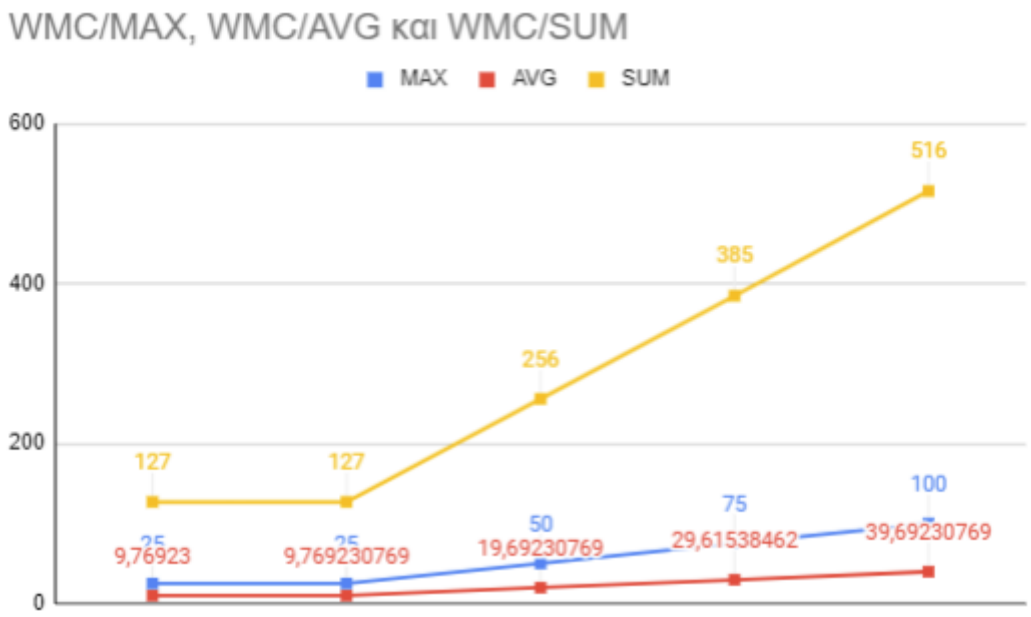
Βήμα 4

Στο βήμα 4 χρησιμοποιούμε το Metrics Calculator με GUI που μας κοινοποιήθηκε. Οι κατηγορίες που θα εξετάσουμε είναι: size, cohesion, complexity, inheritance, coupling.
Για το size, θα ακολουθήσουμε τη μετρική S*IZE1*, για το cohesion την μετρική *LCOM (Lack of Cohesion in Methods)*, για το complexity *WMC (Weighted Methods per Class)*, για το inheritance *ANA (Average Number of Ancestors)* και τέλος για το coupling *CBO (Coupling Between Objects)*.

Αποστολίδης Γεώργιος Δαυίδ
mai2400211
Εργασία 1
Τεχνολογία Συστημάτων Λογισμικού
Μεταπτυχιακό στην Εφαρμοσμένη Πληροφορική
Κατεύθυνση Ανάπτυξης Λογισμικού και Νέφος

	VERSIONS	1	2	3	4	5
METRICS						
WMC	*MAX*	25	25	50	75	100
	AVG	9,76923	9,769230769	19,69230769	29,61538462	39,69230769
	SUM	127	127	256	385	516
CBO	*MAX*	8	8	8	8	8
	AVG	3,230769	3,230769231	3,230769231	3,230769231	3,230769231
	SUM	42	42	42	42	42
LCOM	*MAX*	236	236	236	236	236
	AVG	51,30769231	51,30769231	53,23076923	53,23076923	54,53846154
	SUM	667	667	692	692	709
SIZE1	*MAX*	262	262	198	198	203
	SUM	81,84615385	82	77,46153846	77,46153846	77,84615385
	AVG	1064	1066	1007	1007	1012
ANA	*MAX*	0	0	0	0	0
	SUM	0	0	0	0	0
	AVG	0	0	0	0	0

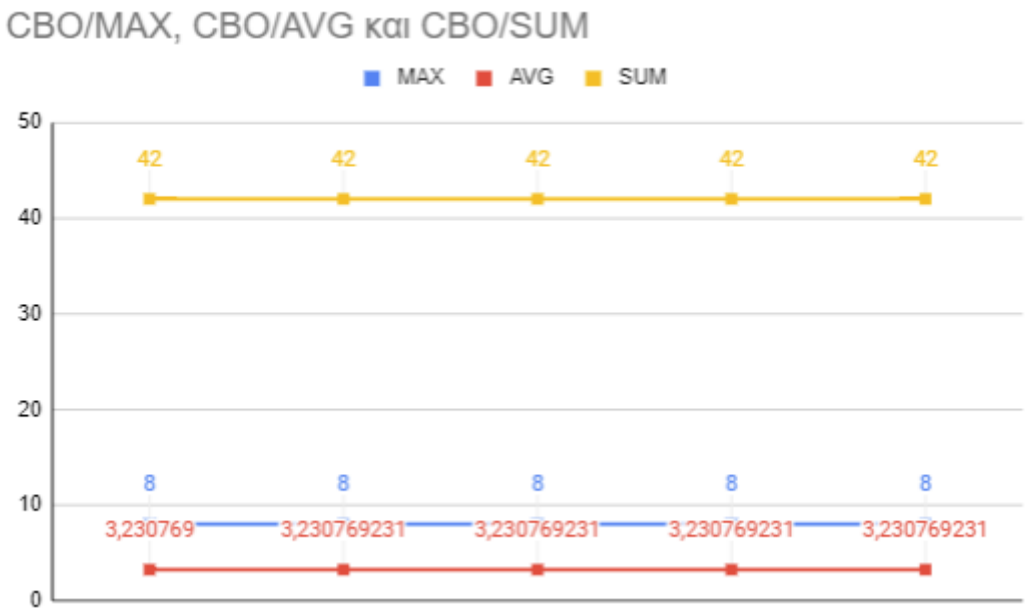
Με βάση τις αλλαγές που εφαρμόστηκαν στο σύστημα, φαίνεται ότι υπήρξε μια εξέλιξη στη δομή του κώδικα και στην πολυπλοκότητά του. Οι αλλαγές αυτές που έγιναν οδήγησαν σε μια σημαντική αύξηση του συνολικού κώδικα, όπως φαίνεται από τη μετρική SIZE 1 που αυξήθηκε ανά πέρασμα των εκδόσεων. Παράλληλα, παρατηρούμε μια σταθερότητα στο CBO, που υποδηλώνει ότι ο βαθμός σύνδεσης μεταξύ των κλάσεων παρέμεινε αμετάβλητος. Ωστόσο, η αύξηση του WMC και η πτώση του LCOM σε μερικές περιπτώσεις μπορεί να υποδεικνύει την εμφάνιση πιο πολύπλοκου κώδικα, ενώ η μετρική ANA παραμένει σταθερά στο μηδέν. Αυτές οι αλλαγές μπορούν να επηρεάσουν την αποδοτικότητα και τη συντήρηση του συστήματος και ενδέχεται να απαιτήσουν περαιτέρω επανεξέταση για τη διασφάλιση της σταθερότητας και της ευκολίας συντήρησής του.



Γράφημα 1 - WMC

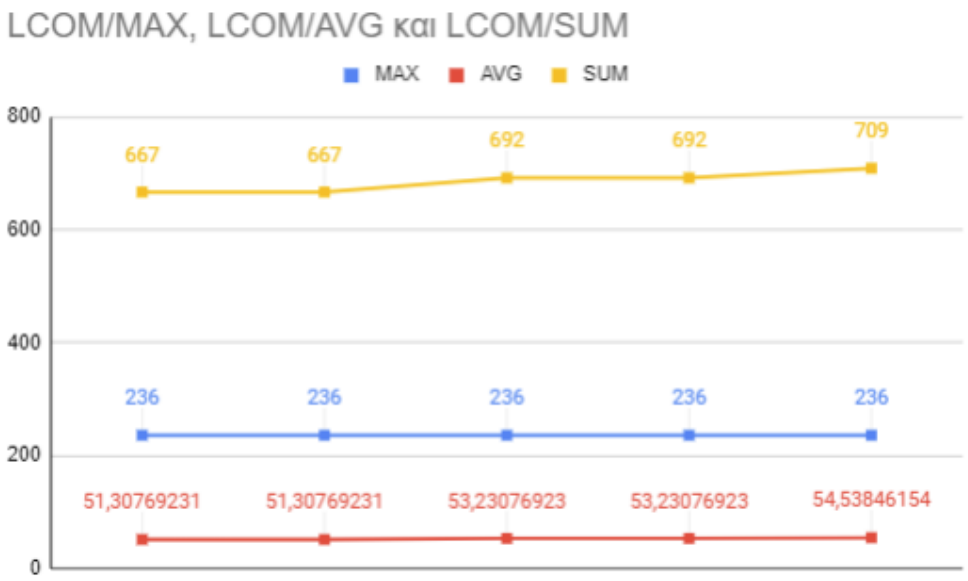
Στην πρώτη έκδοση, η μέση τιμή της WMC ήταν 9.77, με συνολικό άθροισμα 127. Ο αριθμός των μεθόδων ήταν περιορισμένος και η πολυπλοκότητά τους χαμηλή. Καθώς προχωρούσαμε στις επόμενες εκδόσεις και αλλαγές, τόσο ο μέσος όρος όσο και το άθροισμα αυξήθηκαν δραματικά, φτάνοντας στις 39.69 και 516 αντίστοιχα στην πέμπτη έκδοση. Αυτό υποδεικνύει ότι οι νέες μέθοδοι που προστέθηκαν έχουν υψηλότερη πολυπλοκότητα, πιθανόν λόγω της επιπλέον λειτουργικότητας ή άλλων παραμέτρων που εισήχθησαν στο σύστημα.

Εν ολίγοις, η WMC αυξήθηκε σημαντικά με τον χρόνο, υποδεικνύοντας μια αυξημένη πολυπλοκότητα και περισσότερες μεθόδους στις πιο πρόσφατες εκδόσεις του λογισμικού.



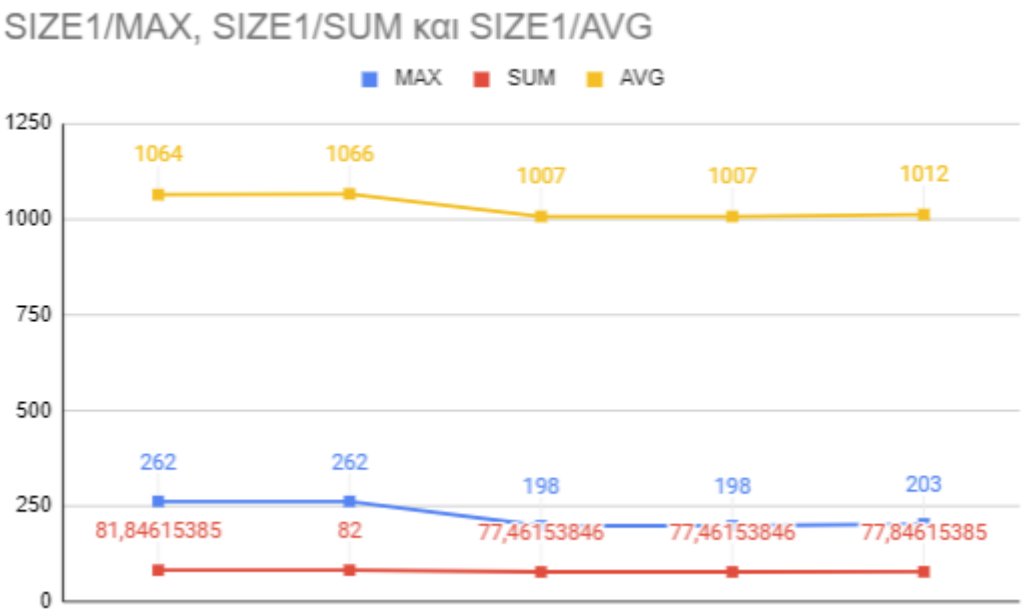
Γράφημα 2 - CBO

Το μέγιστο CBO σε κάθε έκδοση ήταν 8, ενώ ο μέσος όρος και το άθροισμα των τιμών ήταν πάντα 3,23 και 42 αντίστοιχα. Αυτό αποδεικνύει ότι η σύνδεση μεταξύ των αντικειμένων παραμένει σταθερή και σχετικά χαμηλή καθ' όλη την εξέλιξη του λογισμικού. Αυτό είναι θετικό καθώς δείχνει ότι η δομή και η οργάνωση του λογισμικού παραμένει σταθερή χωρίς μεγάλες αλλαγές στο επίπεδο συνδεσιμότητας των αντικειμένων.



Γράφημα 3 - LCOM

Σε αυτήν τη συγκεκριμένη μετρική, η τιμή MAX παραμένει σταθερή σε όλες τις εκδόσεις με τιμή 236, υποδηλώνοντας σταθερό επίπεδο έλλειψης συνοχής σε μεθόδους στον κώδικα. Ωστόσο, ο μέσος όρος αυτής της μετρικής αυξήθηκε από 51,30 σε 54,54. Αυτή η αύξηση μπορεί να υποδείξει μεγαλύτερη διασπορά μεθόδων σε διαφορετικές κλάσεις, προκαλώντας μεγαλύτερη ασυνέπεια στον τρόπο που αλληλεπιδρούν μεταξύ τους. Αυτό μπορεί να υποδείξει ανάγκη πρόσθετης προσοχής και επιμέλειας στη δομή του κώδικα για να διατηρηθεί η καλή οργάνωση και η συνοχή του.



Γράφημα 4 - SIZE1

Αποστολίδης Γεώργιος Δαυίδ
mai2400214
Εργασία 1
Τεχνολογία Συστημάτων Λογισμικού
Μεταπτυχιακό στην Εφαρμοσμένη Πληροφορική
Κατεύθυνση Ανάπτυξης Λογισμικού και Νέφος

Η μετρική SIZE1 παρακολουθεί το μέγεθος του κώδικα και στις πέντε εκδόσεις φαίνεται ότι η τιμή MAX παρέμεινε σταθερή, αλλά το SUM αυξήθηκε από 1064 σε 1012. Αυτό υποδηλώνει μια συνεχή μείωση στο συνολικό μέγεθος του κώδικα με το πέρασμα του χρόνου. Επιπλέον, η μέση τιμή αυτής της μετρικής μειώθηκε από 81,85 σε 77,85. Αυτό μπορεί να υποδεικνύει τη μείωση του μέσου μεγέθους των κλάσεων ή των αρχείων κώδικα, που μπορεί να οδηγήσει σε πιο διαχειρίσιμα και ευανάγνωστα μέρη του κώδικα. Αυτό μπορεί να ερμηνευτεί ως ενδεικτικό ότι η διαχείριση του κώδικα γίνεται με πιο αποδοτικό τρόπο καθώς εξελίσσεται το σύστημα λογισμικού.

Τέλος, η μετρική ANA έχει τιμές που παραμένουν σταθερές σε όλες τις εκδόσεις. Η MAX και η SUM παραμένουν μηδενικές σε όλες τις περιπτώσεις, ενώ η AVG διατηρείται επίσης στο μηδέν. Αυτό υποδηλώνει ότι δεν έχουν γίνει αλλαγές στον αριθμό των επικλήσεων ανά μέθοδο ή την ανάλυση συναρτήσεων. Συνεπώς, δεν έχουν γίνει αλλαγές στην αντίληψη της πολυπλοκότητας ή στον τρόπο διαχείρισης των μεθόδων στο σύστημα. Η χαμηλή ή μηδενική τιμή της μετρικής ANA μπορεί να υποδείξει ότι οι μέθοδοι δεν καλούνται συχνά ή καθόλου από άλλες μεθόδους. Αυτό μπορεί να οδηγήσει σε ένα σύστημα που έχει πολλές μικρές μεθόδους που δεν αλληλεπιδρούν σημαντικά μεταξύ τους. Αν και η μετρική ANA μπορεί να χαμηλώνει την πολυπλοκότητα, μια υπερβολικά χαμηλή τιμή ή μηδενική μπορεί να υποδείξει απομόνωση και έλλειψη αλληλεπίδρασης μεταξύ των τμημάτων του κώδικα. Αυτό μπορεί να δημιουργήσει προβλήματα στην κατανόηση και στη συντήρηση του κώδικα, επιδρώντας στην επεκτασιμότητα και στην αντιστοίχιση των λειτουργιών σε διάφορα μέρη του συστήματος. Συνήθως, μια ορθή ισορροπία μεταξύ υψηλών και χαμηλών τιμών ANA μπορεί να οδηγήσει σε ένα πιο εύρωστο και εύκολα συντηρήσιμο σύστημα. Για τη βελτίωση αυτής της μετρικής, μπορεί να είναι απαραίτητο να εξετάσουμε τη διάρθρωση των μεθόδων, την αναδιοργάνωση των καθηκόντων και την αναθεώρηση της σχεδίασης του κώδικα, προκειμένου να αυξηθεί η συνοχή και η αλληλεπίδραση μεταξύ τους.