Αναγνώριση Προτύπων

Εργασία μαθήματος – Περιγραφή Προβλήματος και Σετ Δεδομένων

1. Εισαγωγή

1.1. Ποιότητα Λογισμικού

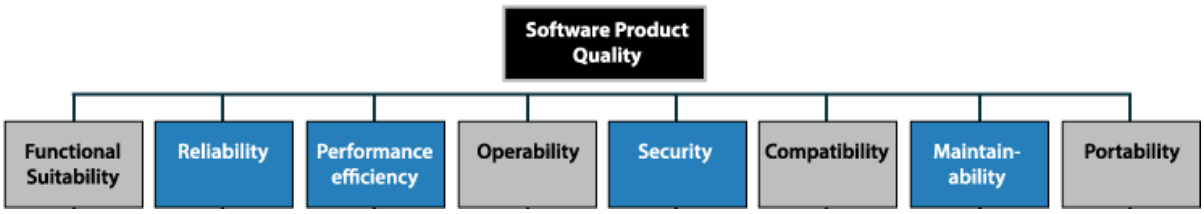
Στο αντικείμενο της τεχνολογίας λογισμικού, ο όρος "**ποιότητα λογισμικού**" (software quality) αναφέρεται σε δύο συναφείς και ταυτόχρονα διακριτές έννοιες, την "**λειτουργική**" και τη "**δομική**":

- Η "**λειτουργική**" ποιότητα λογισμικού (software functional quality) αποτιμά το κατάπόσο ένα έργο λογισμικού ανταποκρίνεται στις λειτουργικές του απαιτήσεις (functional requirements).
- Η "**δομική**" ποιότητα λογισμικού (software structural quality) αποτιμά το κατά πόσο ένα έργο λογισμικού ανταποκρίνεται στις μη λειτουργικές του απαιτήσεις (non-functional requirements), οι οποίες υπαγορεύονται από πολλούς παράγοντες όπως είναι το πεδίο εφαρμογής του έργου λογισμικού, η κατανομή πόρων, το επιχειρηματικό μοντέλο ανάπτυξης κτλ...

Η αξιολόγηση της ποιότητας ενός έργου λογισμικού αποτελεί ένα πολυπαραγοντικό πρόβλημα, το οποίο αποτελεί ένα ευρύτατο πεδίο επιστημονικού ενδιαφέροντος. Η συνθετότητά του έγκειται στο γεγονός ότι ο όρος "**ποιότητα**" αποτιμάται με διαφορετικό τρόπο από διαφορετικούς ανθρώπους και σχετίζεται άμεσα με το πεδίο εφαρμογής κάθε έργου λογισμικού. Η ανάγκη δημιουργίας ενός κοινού τόπου αναφοράς ανάμεσα στην επιστημονική κοινότητα οδήγησε στην τυποποίηση της "ποιότητας λογισμικού" μέσω της πρότασης **προτύπων αξιολόγησης ποιότητας**, τα οποία μοντελοποιούν την ποιότητα λογισμικού ως ένα σύνολο από δομικά χαρακτηριστικά. Πιο συγκεκριμένα, το διεθνές πρότυπο ISO/IEC 25010:2011 ορίζει ότι η ποιότητα ενός έργου λογισμικού αποτελείται από τα εξής οκτώ δομικά χαρακτηριστικά [1]:

- ✓ Λειτουργική καταλληλότητα (Functional suitability)
- ✓ Αξιοπιστία (Reliability)
- ✓ Επίδοση και Αποδοτικότητα (Performance and Efficiency)
- ✓ Ευχρηστία (Usability)
- ✓ Συντηρησιμότητα (Maintainability)
- ✓ Ασφάλεια (Security)
- ✓ Συμβατότητα (Compatibility)
- ✓ Μεταφερσιμότητα (Portability)

Στην Εικόνα 1 παρουσιάζονται τα δομικά χαρακτηριστικά ποιότητας λογισμικού, όπως αυτά διαμορφώθηκαν στο πρότυπο ISO-25010.



Εικόνα 1 - Χαρακτηριστικά ποιότητας λογισμικού, όπως αυτά ορίζονται στο πρότυπο ISO-25010

1.2. Έλεγχος Λογισμικού

Σύμφωνα με τους John E. Bentley, Wachovia Bank και Charlotte NC [2], ως έλεγχος λογισμικού ορίζεται η διαδικασία επαλήθευσης (verification) και επικύρωσης (validation) ότι ένα έργο λογισμικού καλύπτει τις παρακάτω προδιαγραφές:

- ✓ Ανταποκρίνεται πλήρως στα τεχνικά και επιχειρησιακά χαρακτηριστικά που προσδιορίστηκαν κατά την σχεδίαση και υλοποίησή του.
- ✓ Επιτελεί σωστά τη λειτουργία για την οποία υλοποιήθηκε.

Για τον έλεγχο λογισμικού μπορεί να εφαρμοστεί πληθώρα τεχνικών, οι οποίες χωρίζονται σε δύο βασικές κατηγορίες, οι οποίες διακρίνονται με βάση το αν προϋποθέτουν την εκτέλεση του πηγαίου κώδικα:

- ➢ Στατικός Έλεγχος (Static testing)
- ➢ Δυναμικός Έλεγχος (Dynamic testing)

Η διαδικασία του στατικού ελέγχου διενεργείται μέσω της στατικής ανάλυσης (Static Analysis) και το έργο λογισμικού δεν εκτελείται, αλλά ο κώδικάς του αναλύεται με αποτέλεσμα την εξαγωγή χρήσιμων χαρακτηριστικών. Στο σημείο αυτό αξίζει να σημειωθεί ότι η στατική ανάλυση δεν είναι σε θέση να εντοπίσει τα ελαττώματα του κώδικα, αλλά εντοπίζει τα πιθανά σημεία στα οποία μπορούν να εμφανιστούν. Τα στοιχεία/χαρακτηριστικά του κώδικα που μπορούν να προκύψουν από τη στατική ανάλυση είναι πολλά και καλύπτουν διάφορα πεδία ενδιαφέροντος, τα κυριότερα από τα οποία είναι:

- ✓ **Ανάλυση του στυλ του κώδικα (Code Style Analysis)**
  Η διαδικασία αυτή περιλαμβάνει ένα σύνολο από ελέγχους αναφορικά με το αν ο κώδικας ανταποκρίνεται σε μια σειρά από ενδεικνυόμενες πρακτικές στη συγγραφή του.
- ✓ **Μετρικές κώδικα (Static Analysis Metrics)**
  Οι μετρικές αποτελούν σημαντικά στατιστικά στοιχεία του κώδικα που αναφέρονται σε πληθώρα χαρακτηριστικών, όπως είναι η πολυπλοκότητα (complexity), το μέγεθος (size), η σύζευξη (coupling) κ.α., τα οποία έχουν άμεση συσχέτιση με τα χαρακτηριστικά ποιότητας.

1.3. Επαναχρησιμοποίηση Κώδικα

Σύμφωνα με τον Eric S. Raymond [10] ,

"ενώ οι καλοί προγραμματιστές ξέρουν τι να γράψουν, οι εξαιρετικοί προγραμματιστές ξέρουν τι να ξαναγράψουν (επαναχρησιμοποιήσουν)".

Από τη φράση αυτή μπορούμε να συμπεράνουμε τη σημασία της επαναχρησιμοποίησης κώδικα στην τεχνολογία λογισμικού. Η αξιοποίηση των πολυάριθμων έργων ανοικτού λογισμικού, τόσο από αποθήκες ανοικτού λογισμικού, όσο και με τη χρήση συμβατικών μηχανών αναζήτησης ή και εξειδικευμένων συστημάτων ανάκτησης κώδικα, διευκολύνει σημαντικά το έργο των προγραμματιστών. Ωστόσο, η επαναχρησιμοποίηση κώδικα αποφέρει οφέλη εφόσον ανταποκρίνεται στις ανάγκες του προγραμματιστή, και εφόσον πληροί κάποιες προδιαγραφές ποιότητας.

2. Εργασία

2.1. Το Πρόβλημα

Η παρούσα εργασία αναφέρεται στην αξιολόγηση της ποιότητας λογισμικού, όπως αυτή τοποθετείται στο πεδίο της επαναχρησιμοποίησης κώδικα, μεσω της χρήσης μετρικών στατικής ανάλυσης. Πιο συγκεκριμένα, το βασικό ζητούμενο του προβλήματος προς επίλυση αποτελεί η κατασκευή ενός συστήματος πρόβλεψης της ποιότητας ενός έργου λογισμικού, χρησιμοποιώντας ως ground truth (γενική αλήθεια) δεδομένα που σχετίζονται άμεσα με το χαρακτηριστικό της **επαναχρησιμοποίησης** (**reusability**). Τα δεδομένα αυτά είναι ο αριθμός των **stars** και ο αριθμός των **forks.** Μια σειρά από ερευνητικά ερωτήματα αναφορικά με το πεδίο αυτό είναι τα εξής:

- **Εύρεση μηχανισμού για τον καθορισμό του Target Set**

Για την κατασκευή ενός συστήματος πρόβλεψης ποιότητας απαιτείται η δημιουργία ενός μηχανισμού ποσοτικοποίησης του βαθμού επαναχρησιμοποίησης για κάθε method/class/package. Ως βάση για τον μηχανισμό αυτό θα αποτελέσουν τα stars και forks. Όμως, ο αριθμός των stars και forks ορίζεται **για ένα αποθετήριο** γεγονός που σημαίνει ότι θα πρέπει να κατακερματιστεί **για κάθε method/class/package.** Ο μηχανισμός αυτός αποτελεί ένα ερευνητικό ερώτημα.

Μια σειρά από ενδεικτικές προσεγγίσεις στο παραπάνω πρόβλημα είναι οι εξής:

- ✓ Κατακερματισμός των stars/forks με βάση τον αριθμό των methods/classes/packages που ανήκουν σε κάθε αποθετήριο.
- ✓ Χρήση των size metrics για τον κατακερματισμό των stars/forks. (π.χ Lines of Code).
- ✓ Εφόσον τα δεδομένα πάνω στα οποία θα διαμορφωθεί το target set αναφέρονται στο χαρακτηριστικό reusability, τότε θα μπορούσε ο κατακερματισμός των stars/forks να γίνει μέσω του αριθμού των public attributes.
- ✓ Συνδυασμός των παραπάνω.

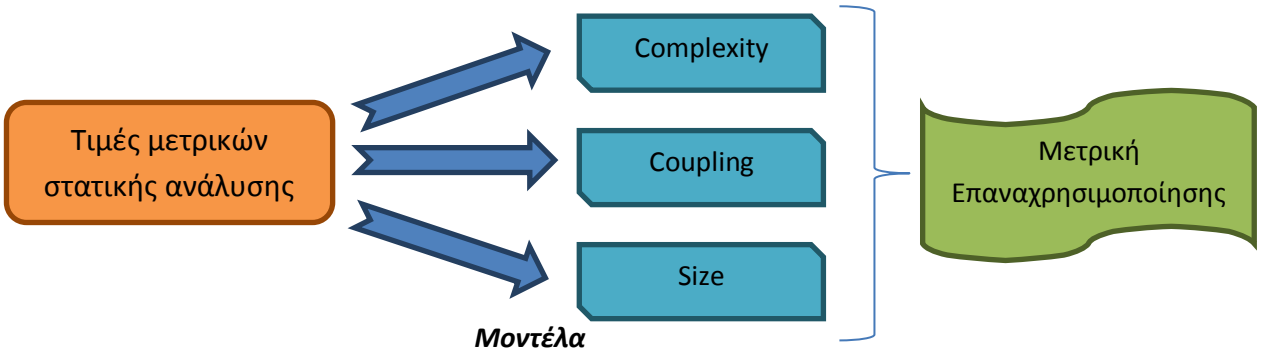
ΠΡΟΣΟΧΗ: Οι μετρικές που θα χρησιμοποιηθούν για την κατασκευή του target set **δεν πρέπει** να χρησιμοποιηθούν ως input στο μοντέλο πρόβλεψης ποιότητας.

- **Αξιολόγηση Επαναχρησιμοποίησης**

Εφόσον κατασκευαστεί το **target set** ακολουθεί η μεθοδολογία για την κατασκευή του μηχανισμού αξιολόγησης επαναχρησιμοποίησης ως χαρακτηριστικού ποιότητας. Η μεθοδολογία αυτή επιδέχεται πολλές προσεγγίσεις μερικές από τις οποίες είναι οι εξής:

1. Εκτίμηση μιας μετρικής ποιότητας με τη χρήση ενός συνόλου μετρικών, οι οποίες συνδέονται με όλα τα χαρακτηριστικά ποιότητας.

2. Εκτίμηση μιας μετρικής ποιότητας ως συνδυασμός εκτιμήσεων για κάθε χαρακτηριστικό ποιότητας.



2.2. Περιγραφή σετ δεδομένων

Τα δεδομένα τα οποία σας δίνονται είναι τα εξής:

- **Repositories_info.csv**

Στο αρχείο αυτό περιέχονται πληροφορίες αναφορικά με τα 100 most forked και most starred αποθετήρια. Πιο συγκεκριμένα σας δίνονται:

- Ο ιδιοκτήτης του αποθετηρίου (repository owner).
- Το όνομα του project (project name).
- Ο αριθμός των stars.
- Ο αριθμός των forks.
- Ο αριθμός των μεθόδων.

- Ο αριθμός των κλάσεων.
- Ο αριθμός των πακέτων.

Μέσα στα περιεχόμενα των φακέλων **Class, Method** και **Package** περιλαμβάνονται οι **μετρικές στατικής ανάλυσης (βλ. Appendix I)** καθώς και **ο αριθμός των παραβιάσεων (violations)** για μια σειρά από κατηγορίες **(βλ. Appendix II)** για όλα τα repositories που περιέχονται στο αρχείο **Repositories_info.csv**.

2.3. Ενδεικτική αντιμετώπιση

Στο αρχείο ***User-Perceived Source Code Quality Estimation based on Static Analysis Metrics.pdf*** μπορείτε να δείτε τον τρόπο με τον οποίο αντιμετωπίσαμε εμείς το παραπάνω πρόβλημα σε μια εργασία η οποία δημοσιεύτηκε στο συνέδριο "*IEEE International Conference on Software Quality, Reliability and Security (QRS), Vienna, Austria, 2016*". [4]

3. References

[1] "CISQ Code Quality Standards". [Retrieved July, 2016]. [Online]. Available: http://it-cisq.org/standards/
[2] John E. Bentley, Wachovia Bank, Charlotte NC. "Software Testing Fundamentals". (2010). Available: http://www2.sas.com/proceedings/sugi30/141-30.pdf.
[3] Eric S. Raymond."The Cathedral and the Bazaar". (1999), ISBN:1-565-92724-9
[4] M. Papamichail, T. Diamantopoulos and A. Symeonidis, "*User-Perceived Source Code Quality Estimation Based on Static Analysis Metrics*", 2016 IEEE International Conference on Software Quality, Reliability and Security (QRS), Vienna, 2016, pp. 100-107.

Appendix I – Μετρικές Στατικής Ανάλυσης

Category	Metric name	Abbrev.	Method	Class	Package
Cohesion metrics	Lack of Cohesion in Methods 5	LCOM5		X	
Complexity metrics	McCabe's Cyclomatic Complexity	McCC	X		
	Nesting Level	NL	X	X	
	Nesting Level Else-If	NLE	X	X	
	Weighted Methods per Class	WMC		X	
Coupling metrics	Coupling Between Object classes	CBO		X	
	Coupling Between Object classes Inverse	CBOI		X	
	Number of Incoming Invocations	NII	X	X	
	Number of Outgoing Invocations	NOI	X	X	
	Response set For Class	RFC		X	
Documentation metrics	API Documentation	AD		X	X
	Comment Density	CD	X	X	X
	Comment Lines of Code	CLOC	X	X	X
	Documentation Lines of Code	DLOC	X	X	
	Public Documented API	PDA		X	X
	Public Undocumented API	PUA		X	X
	Total API Documentation	TAD			X
	Total Comment Density	TCD	X	X	X
	Total Comment Lines of Code	TCLOC	X	X	X
	Total Public Documented API	TPDA			X
	Total Public Undocumented API	TPUA			X

Inheritance metrics	Depth of Inheritance Tree	DIT		X	
	Number of Ancestors	NOA		X	
	Number of Children	NOC		X	
	Number of Descendants	NOD		X	
	Number of Parents	NOP		X	
Size metrics	Lines of Code	LOC	X	X	X
	Logical Lines of Code	LLOC	X	X	X
	Number of Attributes	NA		X	X
	Number of Classes	NCL			X
	Number of Enums	NEN			X
	Number of Getters	NG		X	X
	Number of Interfaces	NIN			X
	Number of Local Attributes	NLA		X	
	Number of Local Getters	NLG		X	
	Number of Local Methods	NLM		X	
	Number of Local Public Attributes	NLPA		X	
	Number of Local Public Methods	NLPM		X	
	Number of Local Setters	NLS		X	
	Number of Methods	NM		X	X
	Number of Packages	NPKG			X
	Number of Parameters	NUMPAR	X		
	Number of Public Attributes	NPA		X	X
	Number of Public Methods	NPM		X	X
	Number of Setters	NS		X	X
	Number of Statements	NOS	X	X	
	Total Lines of Code	TLOC	X	X	X
	Total Logical Lines of Code	TLLOC	X	X	X

	Total Number of Attributes	TNA		X	X
	Total Number of Classes	TNCL			X
	Total Number of Directories	TNDI			X
	Total Number of Enums	TNEN			X
	Total Number of Files	TNFI			X
	Total Number of Getters	TNG		X	X
	Total Number of Interfaces	TNIN			X
	Total Number of Local Attributes	TNLA		X	
	Total Number of Local Getters	TNLG		X	
	Total Number of Local Methods	TNLM		X	
	Total Number of Local Public Attributes	TNLPA		X	
	Total Number of Local Public Methods	TNLPM		X	
	Total Number of Local Setters	TNLS		X	
	Total Number of Methods	TNM		X	X
	Total Number of Packages	TNPKG			X
	Total Number of Public Attributes	TNPA		X	X
	Total Number of Public Classes	TNPCL			X
	Total Number of Public Enums	TNPEN			X
	Total Number of Public Interfaces	TNPIN			X
	Total Number of Public Methods	TNPM		X	X
	Total Number of Setters	TNS		X	X
	Total Number of Statements	TNOS	X	X	X

Appendix II – Κατηγορίες Παραβιάσεων

Category	Method	Class	Package
Android Rules	X	X	X
Basic Rules	X	X	X
Brace Rules	X	X	X
Clone Metric Rules	X	X	
Code Size Rules	X	X	X
Complexity Metric Rules	X	X	
Controversial Rules	X	X	X
Coupling Metric Rules	X	X	
Design Rules	X	X	X
Documentation Metric Rules	X	X	
Empty Code Rules	X	X	X
Java Logging Rules	X	X	X
Migration Rules	X	X	X
Naming Rules	X	X	X
Optimization Rules	X	X	X
Security Code Guideline Rules	X	X	X
Size Metric Rules	X	X	
Strict Exception Rules	X	X	X
String and StringBuffer Rules	X	X	X
Type Resolution Rules	X	X	X
Unnecessary and Unused Code Rules	X	X	X
Vulnerability Rules	X	X	X
Cohesion Metric Rules		X	
Inheritance Metric Rules		X	

Και στις τρεις περιπτώσεις (Method, Class, Package) υπάρχουν επίσης και οι εξής μεταβλητές:

- **WarningBlocker**: Το σύνολο των παραβιάσεων που εντοπίστηκαν για κανόνες με Priority 1
- **WarningCritical:** Το σύνολο των παραβιάσεων που εντοπίστηκαν για κανόνες με Priority 2
- **WarningMajor:** Το σύνολο των παραβιάσεων που εντοπίστηκαν για κανόνες με Priority 3
- **WarningMinor:** Το σύνολο των παραβιάσεων που εντοπίστηκαν για κανόνες με Priority 4
- **WarningInfo:** Το σύνολο των παραβιάσεων που εντοπίστηκαν για κανόνες με Priority 5

ΠΡΟΣΟΧΗ: Όσο πιο μικρή είναι η τιμή του Priority, τόσο πιο σημαντικός είναι ο κανόνας.