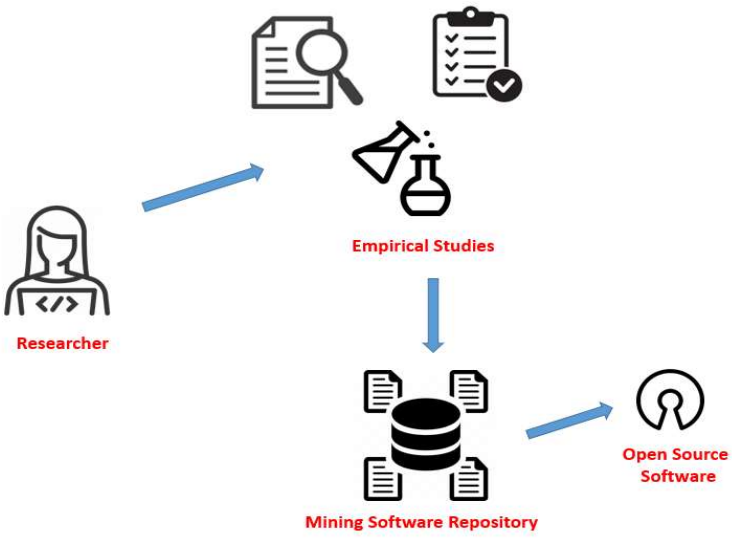
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Κατευθυντήριες γραμμές σχετικά με την επιλογή συλλογών από κώδικα ανοιχτού λογισμικού σε μια εμπειρική μελέτη»



Της φοιτήτριας
ΖΗΒΩΝΗ ΑΝΝΑ
Αρ. Μητρώου: 154605

Επιβλέπουσα
ΕΛΒΙΡΑ ΜΑΡΙΑ ΑΡΒΑΝΙΤΟΥ

ΘΕΣΣΑΛΟΝΙΚΗ 2023

Τίτλος Δ.Ε. **Κατευθυντήριες γραμμές σχετικά με την επιλογή συλλογών από κώδικα ανοιχτού λογισμικού σε μια εμπειρική μελέτη**

Κωδικός Π.Ε. **22145**

Ονοματεπώνυμο φοιτήτριας **Ζηβώνη Άννα**

Ονοματεπώνυμο εισηγητή **Αρβανίτου Ελβίρα Μαρία**

Ημερομηνία ανάληψης Π.Ε. **12/3/2022**

Ημερομηνία περάτωσης Π.Ε. **26/5/2023**

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Πρόλογος

Η παρούσα εργασία εκπονήθηκε από την φοιτήτρια του Τμήματος Πληροφορικής Τ.Ε του ΔΙΠΑΕ , Ζηβώνη Άννα τη χρονική περίοδο από το Μάρτιο του 2022 έως τον Μάιο του 2023.

Στο χώρο της Τεχνολογίας Λογισμικού τα τελευταία χρόνια οι ερευνητές χρησιμοποιούν εμπειρικές μελέτες ανοιχτού λογισμικού με σκοπό την επιβεβαίωση της μεθοδολογίας τους. Σκοπός της παρούσας πτυχιακής είναι η παροχή κατευθυντήριων γραμμών στους ερευνητές εμπειρικών μελετών και η δημιουργία ιστοσελίδας με όλες τις συλλογές από τα ανοιχτά λογισμικά των μελετών που συλλέξαμε.

Στο σημείο αυτό θα ήθελα να εκφράσω τις ευχαριστίες μου προς την κυρία Αρβανίτου Ελβίρα Μαρία που μου έδωσε την ευκαιρία να εμβαθύνω τις γνώσεις μου στις έννοιες της εμπειρικής μελέτης μέσα από την εκπόνηση αυτής της πτυχιακής εργασίας. Επίσης θα ήθελα να την ευχαριστήσω για την βοήθεια που μου πρόσφερε καθώς και για τις κατευθυντήριες γραμμές που μου έδωσε και ελπίζω αυτή η συνεργασία να συνεχιστεί μέχρι την πραγματοποίηση του επόμενου μας στόχου, την δημοσίευση αυτής της εργασίας σε επιστημονικό περιοδικό.

Θεσσαλονίκη, Μάϊος 2023

Περίληψη

Η παρούσα πτυχιακή εργασία έχει ως σκοπό αρχικά τη βιβλιογραφική ανασκόπηση των εμπειρικών μελετών της Τεχνολογίας Λογισμικού και κατά δεύτερον την δημιουργία ενός εργαλείου, όπου ο ερευνητής θα έχει τη δυνατότητα να αναζητά πληροφορίες για μελλοντικές έρευνες. Για τον παραπάνω λόγο, εξετάστηκαν άρθρα από ένα εύρος περιοδικών (Information & Software Technology , ACM Transactions on Software Engineering and Methodology (TOSEM) και Empirical Software Engineering) προκειμένου να καταγραφούν: (α) οι στόχοι των εμπειρικών μελετών; (β) τα κριτήρια επιλογής των ανοιχτών λογισμικών (open source) σε εμπειρικές μελέτες; (γ) τα διαθέσιμα ανοιχτά λογισμικά; καθώς και (δ) αν οι εμπειρικές μελέτες που πραγματοποιήθηκαν τα τελευταία χρόνια διαθέτουν τα δεδομένα τους ανοιχτά για τους ερευνητές ή προγραμματιστές. Για τους σκοπούς της έρευνας, η αναζήτηση των χρήσιμων μελετών έγινε μέσω των ψηφιακών βιβλιοθηκών Springerlink, Sciencedirect και ACM και η ανάκτηση των μελετών έγινε με την χρήση διαφόρων στοιχείων αναζήτησης (search string, κριτήρια επιλογής). Η μέθοδος αυτή είχε ως αποτέλεσμα να ανακτηθούν 394 εμπειρικές μελέτες και στην συνέχεια έγινε ανάλυση των δεδομένων που συλλέχθηκαν. Η ανασκόπηση αφορούσε μελέτες που δημοσιεύτηκαν τα δυο τελευταία έτη, έως το 2022. Οι πληροφορίες των άρθρων που συλλέχθηκαν χρησιμοποιήθηκαν στην δημιουργία μίας ιστοσελίδας ,όπου οι ερευνητές έχουν την δυνατότητα αναζήτησης των συγκεκριμένων μελετών καθώς και την προσθήκη νέων.

«Guidelines for selecting collections from open source software in an empirical study»

«Anna Zivoni»

Abstract

The purpose of this thesis is firstly the bibliographic review of the empirical studies of Software Technology and secondly the creation of a tool, where the researcher will have the possibility to search for information for future research. For the above reason, articles from a range of journals (Information & Software Technology, ACM Transactions on Software Engineering and Methodology (TOSEM) and Empirical Software Engineering) were examined in order to record: (a) the aims of the empirical studies; (b) the selection criteria of open source software in empirical studies; (c) available open source software; and (d) whether empirical studies conducted in recent years make their data open to researchers or developers. For the purposes of the research, the search for useful studies was done through the digital libraries Springerlink, Sciencedirect and ACM and the retrieval of the studies was done using various search elements (search string, selection criteria). This method resulted in the retrieval of 394 empirical studies and then an analysis of the collected data was performed. This review concerned studies published in the last two years, up to 2022. The information of the collected articles was used in the creation of a website, where researchers have the ability of searching for the specific studies as well as adding new ones.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

API(S)	Application Programming Interface(s)
DL	Digital Library
GNU	GNU is not Unix
GQM	Goal Question Metric
IBM	International Business Machines Corporation
MSR	Mining Software Repositories
MSRConf	Working Conference on Mining Software Repositories
OSS	Open Source Software
SMS	Systematic Mapping Study
CVS	Concurrent Versions System
BSD	Berkeley Software Distribution
GPL	General Public License
RQ	Research Question
IEE	Independent Educational Evaluation
ACM	Association for Computing Machinery
IC	Incusion Criteria
EC	Exclusion Criteria
HTML	HyperText Markup Language
DOM	Document Object Model
CSS	Cascading Style Sheets
SPL	Software Product Lines
SBSE	Search Based Software Engineering
CIA	Change Impact Analysis
CBSE	Component Based Software Engineering
SOA	Service Oriented Architectures
DSL	Domain Specific Languages
CI/CD	Continuous Integration and Continuous Delivery
JS	Javascript
GUI	Graphical user interface

Κεφάλαιο 1ο: Εισαγωγή

Στον τομέα της μηχανικής λογισμικού, οι εμπειρικές μελέτες αποτελούν το θεμέλιο της επιστημονικής έρευνας. Πιο συγκεκριμένα, η ερευνητική μελέτη ορίζεται ως μια ερευνητική μέθοδος που χρησιμοποιεί άμεση παρατήρηση ή εμπειρία για τη συλλογή πληροφοριών και την εξαγωγή συμπερασμάτων σχετικά με ένα συγκεκριμένο φαινόμενο ή πρόβλημα [1]. Αυτός ο τύπος μελέτης βασίζεται σε δεδομένα που συλλέγονται μέσω συστηματικών και αντικειμενικών μεθόδων, όπως πειράματα, έρευνες, συνεντεύξεις ή παρατηρήσεις, και αναλύονται χρησιμοποιώντας στατιστικές ή άλλες ποσοτικές μεθόδους. Χρησιμοποιώντας τις συγκεκριμένες μελέτες, ένας ερευνητής μπορεί να επικυρώσει και επαληθεύσει τα αποτελέσματα του. Τα αποτελέσματα των εμπειρικών μελετών δημοσιεύονται συνήθως σε επιστημονικά περιοδικά και αναμένεται να συμβάλουν στην πρόοδο της γνώσης στους αντίστοιχους τομείς τους [1].

1.1 Τύποι Εμπειρικών Μελετών

Με βάση τον [1], οι εμπειρικές μελέτες χωρίζονται σε: (α) ελεγχόμενα πειράματα, (β) μελέτες περιπτώσεων, (γ) έρευνες, και (δ) μεταγενέστερη ανάλυση, οι οποίες αναλύονται παρακάτω. :

- Ελεγχόμενα Πειράματα (Experiments)
  Στο συγκεκριμένο τύπο ο ερευνητής ελέγχει την πρόοδο της μελέτης και παρακολουθεί την εκτέλεση των εργασιών και τις περισσότερες φορές εκτελούνται σε εργαστηριακό περιβάλλον. Ένα πλεονέκτημα της συγκεκριμένης εμπειρικής μελέτης είναι ο προγραμματισμός και ο σχεδιασμός με στόχο την υψηλή αξιοπιστία, ενώ το μειονέκτημα της είναι το περιορισμένο εύρος των πειραμάτων.
  Συχνά τα πειράματα χρησιμοποιούνται για την σύγκριση διαφορετικών τεχνικών, μεθόδων, διαδικασιών εργασιών κλπ. Ένα παράδειγμα ελεγχόμενου πειράματος στη μηχανική λογισμικού είναι η σύγκριση δύο διαφορετικών μεθόδων για περαιτέρω έλεγχο, όπου εφαρμόζονται στατιστικές μέθοδοι με σκοπό να αναδειχθεί με στατιστική σημασία ότι η μία μέθοδος είναι καλύτερη από την άλλη.
  Αναλυτικότερα ένα πείραμα περιλαμβάνει τα εξής στάδια:
  1. Τον Σχεδιασμό: Ο σχεδιασμός χρειάζεται να προγραμματιστεί με ιδιαίτερη λεπτομέρεια και ο πληθυσμός του δείγματος πρέπει να είναι τυχαίος αλλά αντιπροσωπευτικός.
  2. Την Λειτουργία: Η λειτουργία του πειράματος περιλαμβάνει τη δέσμευση των συμμετεχόντων (participants), όπου γίνεται ανάληψη καθηκόντων.
  3. την Προετοιμασία των οργάνων μέτρησης, όπου περιλαμβάνονται οι γραπτές οδηγίες προς τους συμμετέχοντες και οι μορφές που πρέπει να χρησιμοποιούν οι συμμετέχοντες κατά τη διάρκεια των δοκιμών. Και τέλος,
  4. Την Εκτέλεση, η οποία περιλαμβάνει την ανάλυση των δεδομένων και την ερμηνεία του αποτελέσματος.
- Μελέτες περιπτώσεων (Case study)
  Μια μελέτη περίπτωσης πραγματοποιείται για να ερευνήσει ένα ενιαίο σύνολο ή ένα συγκεκριμένο φαινόμενο. Η συλλογή των πληροφοριών είναι λεπτομερής και κατά τη διάρκειά της εφαρμόζονται διάφορες τεχνικές συλλογής.

Οι μελέτες περιπτώσεων είναι κατάλληλες για την αξιολόγηση και την παρακολούθηση έργων, δραστηριοτήτων ή εργασιών της μηχανικής λογισμικού, διότι αποτρέπεται η όξυνση των προβλημάτων.

Τα στάδια μίας μελέτης περίπτωσης είναι τα εξής: [2] :

1. Σχεδιασμός μελέτης περίπτωσης: καθορίζονται οι στόχοι και σχεδιάζεται η μελέτη.
2. Προετοιμασία για συλλογή δεδομένων: ορίζονται διαδικασίες και πρωτόκολλα για τη συλλογή δεδομένων
3. Συλλογή δεδομένων μελέτης
4. Ανάλυση και ερμηνεία των δεδομένων.
5. Συμπεράσματα

Ο εύκολος προγραμματισμός είναι από τα βασικότερα πλεονεκτήματα αυτού του τύπου μελέτης, αλλά εμφανίζει δυσκολία στη γενίκευση και ερμηνεία των αποτελεσμάτων, καθώς εφάπτεται σε μια συγκεκριμένη κατάσταση.

Στη μηχανική λογισμικού οι μελέτες περιπτώσεων δεν πρέπει να χρησιμοποιούνται για την αξιολόγηση μόνο του τρόπου και αιτίας κάποιων φαινομένων, αλλά και των διαφορών τους.

Επιπρόσθετα, η μελέτη περίπτωσης δύναται να εφαρμοστεί και ως μία έρευνα με στοιχεία στρατηγικής σύγκρισης, η οποία παραθέτει και συγκρίνει τα αποτελέσματα της χρήσης μίας μεθόδου ή τα ελεγχόμενα αποτελέσματα μίας άλλης προσέγγισης. Η αποτροπή της μεροληψίας και η διασφάλιση της εγκυρότητας μπορούν να επιτευχθούν με τη δημιουργία σταθερής βάσης αξιολόγησης των αποτελεσμάτων. Συγκεκριμένα, υπάρχουν τρεις τρόποι για να διευκολυνθεί η μελέτη [1]:

1. Η σύγκριση των αποτελεσμάτων από τη χρήση της νέας μεθόδου έναντι μιας εταιρικής βάσης. Η εταιρεία θα πρέπει να συλλέγει δεδομένα από τυπικά έργα και να υπολογίζει χαρακτηριστικά όπως η μέση παραγωγικότητα και το ποσοστό ελαττωμάτων. Στη συνέχεια, είναι δυνατή η σύγκριση των αποτελεσμάτων από τη περιπτωσιακή μελέτη με τα στοιχεία της βάσης.
2. Ένα παρόμοιο- αδελφικό έργο μπορεί να επιλεγεί ως βάση. Το υπό μελέτη έργο χρησιμοποιεί τη νέα μέθοδο και το αδελφικό έργο την τρέχουσα. Και τα δύο έργα θα πρέπει να έχουν τα ίδια χαρακτηριστικά, δηλαδή τα έργα πρέπει να είναι συγκρίσιμα.
3. Εάν η μέθοδος ισχύει για μεμονωμένα χαρακτηριστικά, θα μπορούσε να εφαρμοστεί τυχαία σε ορισμένα και όχι σε άλλα. Αυτό μοιάζει πολύ με ένα ελεγχόμενο πείραμα, αλλά εφόσον τα έργα δεν προέρχονται τυχαία από τον πληθυσμό όλων των έργων, δεν είναι πείραμα.

Οι μελέτες περιπτώσεων παρουσιάζουν πλεονεκτήματα και μειονεκτήματα. Στα πλεονεκτήματα, πέρα από αυτό που προαναφέρθηκε, είναι η ενσωμάτωση ιδιοτήτων που σε ένα πείραμα δεν μπορούν να αναδειχθούν, όπως είναι η περιπλοκότητα και η κλίμακα [1]. Επιπλέον, η μελέτη περίπτωσης δεν ελέγχει πλήρως της ενέργειες των ερευνητών με αποτέλεσμα την ευελιξία τους σε περίπτωση έκτακτων αλλαγών. Όμως, ο περιορισμένος έλεγχος μπορεί να οδηγήσει σε προβλήματα λόγω σύγχυσης και έλλειψης συντονισμού [1].

- Έρευνες (Surveys)

  Μία έρευνα λαμβάνει χώρα όταν ένα εργαλείο ή μία τεχνική έχει ολοκληρωθεί ή ετοιμάζεται να δημοσιευτεί[1]. Μία έρευνα μπορεί να χαρακτηριστεί ως ένα στιγμιότυπο της τρέχουσας κατάστασης και να χρησιμοποιηθεί για δημοσκοπήσεις και έρευνες αγοράς.

  Οι έρευνες στοχεύουν στην κατανόηση του αντιπροσωπευτικού πληθυσμού, μέσω ερωτηματολογίων ή/και συνεντεύξεων, και παρέχουν ένα μεγάλο αριθμό δεδομένων για περαιτέρω αξιολόγηση.

Ως σκοπός της έρευνας μπορεί να καθοριστεί ένας από τους παρακάτω [1]:

1. Περιγραφή (Descriptive): Η περιγραφική έρευνα ελέγχει διάφορους ισχυρισμούς για ορισμένο πληθυσμό και καθορίζει την κατανομή συγκεκριμένων χαρακτηριστικών ή ιδιοτήτων.
2. Επεξήγηση (Explanatory): Η επεξηγηματική έρευνα ελέγχει συγκεκριμένους επεξηγηματικούς ισχυρισμούς σχετικά με τον πληθυσμό, καθώς αιτιολογούνται οι λόγοι επιλογής μεθόδων, τεχνικών κλπ. από άλλες προσεγγίσεις.
3. Εξερεύνηση (Explorative): Η έρευνα εξερεύνησης χρησιμοποιείται ως μία μελέτη, η οποία προηγείται της ενδελεχούς έρευνας και εξαλείφει τον κίνδυνο παράλειψης σημαντικών ζητημάτων.

Για την πραγματοποίηση μίας έρευνας, σπουδαίο ρόλο παίζει η συλλογή των δεδομένων, η οποία επιτυγχάνεται πιο συχνά μέσω ερωτηματολογίων και συνεντεύξεων. Πιο συγκεκριμένα τα στάδια που ακολουθεί μία τέτοια έρευνα, είναι τα εξής [3] :

1. Προσδιορισμός στόχου : Προσδιορισμός του είδους των ερωτήσεων και πληροφοριών που πρέπει να συλλεχθούν.
2. Προσδιορισμός συμμετεχόντων στην έρευνα: Καθορισμός των χαρακτηριστικών ή των δημογραφικών στοιχείων του κοινού-στόχου, όπως η ηλικία, το φύλο, η τοποθεσία ή το επάγγελμα.
3. Επιλογή του τύπου της έρευνας: Επιλογή της μεθόδου (έντυπη, ηλεκτρονική ή συνέντευξη) που ταιριάζει καλύτερα στο κοινό-στόχο.
4. Σχεδιασμός ερωτήσεων : Δημιουργία λίστας σαφών, συνοπτικών και αμερόληπτων ερωτήσεων που θα βοηθήσουν στην συγκέντρωση των επιθυμητών πληροφοριών.
5. Πιλοτική μελέτη: Πολιτική δοκιμή με μια μικρή ομάδα ατόμων που είναι παρόμοια με το κοινό-στόχο σας. Αυτό βοηθά στον εντοπισμό τυχόν προβλημάτων με το σχεδιασμό της έρευνας, τη σαφήνεια της ερώτησης ή τεχνικά προβλήματα.
6. Διανομή της έρευνας: Εφαρμογή της επιλεγμένης μεθόδου έρευνας και διανομή της στο κοινό-στόχο.
7. Ανάλυση των απαντήσεων και αποτελέσματα: Ανάλυση των δεδομένων με στατιστικά εργαλεία ή λογισμικό και εξαγωγή συμπερασμάτων.

Οι συνεντεύξεις είναι παρόμοιες με τα ερωτηματολόγια με τη διαφορά ότι οι ερωτήσεις γίνονται από τον ίδιο τον ερευνητή. Επιτρέποντας στους ερευνητές να χειριστούν τις ερωτήσεις (τηλεφωνικά ή πρόσωπο με πρόσωπο) αντί για τους ίδιους τους ερωτηθέντες, προσφέρει μια σειρά από πλεονεκτήματα: (α) επιτυγχάνονται υψηλότερα ποσοστά ανταπόκρισης; (β)μειώνεται ο αριθμός των απαντήσεων «δεν ξέρω» και «δεν απαντώ»; Και (γ) είναι δυνατό για τον ερευνητή να παρατηρεί και να κάνει ερωτήσεις. Το μειονέκτημα των συνεντεύξεων είναι ότι το κόστος και ο χρόνος εξαρτώνται άμεσα από το μέγεθος του δείγματος και τις προθέσεις της έρευνας.

- Μεταγενέστερη ανάλυση (Post-mortem analysis)

Τέλος, η μεταγενέστερη ανάλυση αποτελεί μία μέθοδο που μελετά μια παρελθοντική μελέτη και εστιάζει στην τυπική κατάσταση. Είναι παρόμοια με την μελέτη περίπτωσης ως προς το εύρος και την έρευνα, ωστόσο διαφέρουν στην χρονολογική διερεύνηση. Ο στόχος της μεταγενέστερης ανάλυσης είναι να έρθει σε επαφή με τη γνώση και την εμπειρία από μία συγκεκριμένη περίπτωση ή δραστηριότητα μετά το πέρας αυτής.

Υπάρχουν δύο τύποι μεταγενέστερης ανάλυσης: μία *γενική*, η οποία συλλέγει όλες τις διαθέσιμες πληροφορίες από μία δραστηριότητα και μία *εστιασμένη* σε μία συγκεκριμένη δραστηριότητα. Οι μεταγενέστερες αναλύσεις στοχεύουν κυρίως σε μεγάλα έργα μηχανικής

λογισμικού, προκειμένου να μάθουν από την επιτυχία τους ή να ανακάμψουν από την αποτυχία τους.

Μία μεταγενέστερη ανάλυση περιλαμβάνει τα παρακάτω στάδια [1]:

1. Έρευνα του έργου: Ο στόχος είναι να χρησιμοποιηθεί μια έρευνα για τη συλλογή πληροφοριών σχετικά με το έργο από τους συμμετέχοντες. Μέσω της έρευνας εξασφαλίζεται η εμπιστευτικότητα..
2. Συλλογή Αντικειμενικών Πληροφοριών: Σε αυτό το βήμα, συλλέγονται αντικειμενικές πληροφορίες σχετικά με την ευρυθμία του έργου, όπως σφάλματα(bugs) και ανθρωποώρες που χρειάστηκαν.
3. Ενημερωτική συνάντηση: Η συνάντηση αυτή πραγματοποιείται για την καταγραφή των ζητημάτων που δεν καλυφθήκαν από την έρευνα και για να εκφράσουν οι συμμετέχοντες την άποψη του..
4. Ιστορική μέρα του έργου: Η ιστορική μέρα πραγματοποιείται με ένα επιλεγμένο υποσύνολο ατόμων και εξετάζονται τα γεγονότα και τα δεδομένα του έργου.
5. Δημοσίευση των αποτελεσμάτων: Η έκθεση των αποτελεσμάτων επικεντρώνεται στα διδάγματα και στη χρήση τους για την καθοδήγηση της οργανωτικής βελτίωσης.

Υπάρχει μία πιο "ελαφριά" προσέγγιση της μεταγενέστερης ανάλυσης, προκειμένου να βοηθηθούν μικρές και μεσαίες επιχειρήσεις, η οποία επικεντρώνεται σε λίγες ζωτικές δραστηριότητες και υποστηρίζει ότι:

- Η μεταγενέστερη ανάλυση πρέπει να είναι "ανοιχτή " προς όλα τα μέλη της ομάδας και τους ενδιαφερόμενους.
- Οι στόχοι μπορούν να χρησιμοποιηθούν ως κεντρικό θέμα συζήτησης χωρίς αυτό να είναι απαραίτητο
- Η μεταγενέστερη ανάλυση αποτελείται από τρείς φάσεις: την προετοιμασία, την συλλογή και την ανάλυση των δεδομένων.

Γενικά θα λέγαμε ότι οι μεταγενέστερες αναλύσεις είναι ένας ευέλικτος τύπος ανάλυσης παρόλο που το πραγματικό αντικείμενο (ένα έργο ή συγκεκριμένη δραστηριότητα) που μελετούν και το είδος των ερωτημάτων που τίθενται είναι πολύ εξαρτημένα από την πραγματική κατάσταση και τους στόχους της ανάλυσης.

Οι εμπειρικές μελέτες μπορούν επίσης να διαχωριστούν ανάλογα με την φύση τους σε δύο τύπους ερευνητικών παραδειγμάτων, την ***ποιοτική*** και την ***ποσοτική*** έρευνα. Η ***ποιοτική*** έρευνα ασχολείται με τη μελέτη αντικειμένων στο φυσικό τους περιβάλλον, όπου ο ερευνητής στοχεύει στην ερμηνεία ενός φαινομένου βασιζόμενος στην ανθρώπινη επεξήγηση . Το βασικό στοιχείο είναι η αποδοχή της ύπαρξης διαφορετικών τρόπων ερμηνείας. Η ***ποσοτική*** έρευνα ασχολείται κυρίως με την ποσοτικοποίηση μιας σχέσης ή σύγκρισης δύο ή περισσότερων ομάδων. Ο στόχος είναι η εύρεση της σχέσης αιτίου-αποτελέσματος. Η ποσοτική έρευνα πραγματοποιείται συχνότερα μέσω ελεγχόμενων πειραμάτων ή μελετών περιπτώσεων και έχει ως πλεονέκτημα την ικανότητα της σύγκρισης και της στατιστικής ανάλυσης.

Πίνακας 1.1 : Σχέση τύπου και φύσης μελέτης [1].

Τύπος μελέτης	Φύση μελέτης
Ελεγχόμενα Πειράματα	Ποσοτική
Μελέτες περιπτώσεων	Ποιοτική και Ποσοτική
Έρευνες	Ποιοτική και Ποσοτική
Μεταγενέστερη ανάλυση	Ποιοτική και Ποσοτική

1.2 Ανοιχτό Λογισμικό

Οι πρώτες αναφορές του ανοιχτού λογισμικού (Open Source Software-OSS) εντοπίζονται στις δεκαετίες του 1960 και του 1970[10], όταν οι υπολογιστές ήταν ακόμα σχετικά νέοι και μόνο ένας μικρός αριθμός ανθρώπων είχε πρόσβαση σε αυτούς. Στη δεκαετία του 1960, οι πηγαίοι κώδικες ήταν ελεύθερα διαθέσιμοι και προσβάσιμοι σε προγραμματιστές σε όλο τον κόσμο. Χαρακτηριστικό παράδειγμα ήταν η ομάδα χρηστών SHARE, η οποία δημιουργήθηκε το 1955 για να μοιράζεται λογισμικό για μεγάλους υπολογιστές IBM. Το 1965 η IBM σταμάτησε να παρέχει πηγαίους κώδικες λογισμικού στα λειτουργικά συστήματα των υπολογιστών της.

Στη δεκαετία του 1970, οι προγραμματιστές υπολογιστών άρχισαν να συνειδητοποιούν ότι η ανάπτυξη λογισμικού ήταν κερδοφόρα και μέσω συμφωνιών αδειών χρήσης απαγόρευαν ή περιόριζαν την διάδοση του λογισμικού. Ο Richard Stallman πίστευε ότι το λογισμικό πρέπει να είναι ελεύθερο και ανοιχτό και δημιούργησε το Έργο GNU το 1984 με στόχο τη δημιουργία ενός ελεύθερου και ανοιχτού κώδικα λειτουργικού συστήματος και την ελεύθερη συνεργασία μεταξύ των προγραμματιστών. Επίσης, ο Stallman παρείχε τη Γενική Δημόσια Άδεια GNU (GPL) για την προστασία της ελευθερίας του λογισμικού.

Το Linux[11], δημιουργήθηκε το 1991 ως λειτουργικό σύστημα που μοιάζει με Unix για προσωπικούς υπολογιστές. Κυκλοφόρησε τον πηγαίο κώδικα για το Linux με τη Γενική Δημόσια Άδεια GNU, η οποία επέτρεπε σε άλλους να συνεισφέρουν και να τροποποιήσουν τον κώδικα. Με τον καιρό, το Linux απέκτησε δημοτικότητα και υποστήριξη από μια μεγάλη κοινότητα προγραμματιστών και έγινε ένα από τα πιο ευρέως χρησιμοποιούμενα λειτουργικά συστήματα στον κόσμο. Μαζί με το BSD (Berkeley Software Distribution) θεωρούνται δύο από τα πιο δημοφιλή λειτουργικά συστήματα ανοιχτού κώδικα που χρησιμοποιούνται σήμερα.

Τέλος, το GitHub, το οποίο ιδρύθηκε το 2008 [13], έχει γίνει ένα ουσιαστικό εργαλείο για τη σύγχρονη συνεργασία στη βιομηχανία λογισμικού και έχει παίξει βασικό ρόλο στην ανάπτυξη του λογισμικού ανοιχτού κώδικα. Παρέχοντας μια πλατφόρμα στους προγραμματιστές να μοιράζονται κώδικα, να συνεργάζονται σε έργα και να δημιουργούν κοινότητες γύρω από την εργασία τους, το GitHub βοήθησε στον εκδημοκρατισμό της ανάπτυξης λογισμικού και διευκολύνει οποιονδήποτε να συνεισφέρει και να επωφεληθεί από έργα ανοιχτού κώδικα. [12].

Παρακολουθώντας την εξέλιξη των ανοιχτών κώδικα λογισμικών, παρατηρείται ότι το OSS έχει κερδίσει σημαντική δημοτικότητα τα τελευταία χρόνια, λόγω της αξιοπιστίας, της αποτελεσματικότητας και τη λειτουργικότητάς του [4]. Το OSS αναφέρεται σε λογισμικό, του οποίου ο πηγαίος κώδικας ή ο κώδικας προγραμματισμού που βρίσκεται κάτω από το λογισμικό διατίθεται στο κοινό και είναι ανοιχτά προσβάσιμος για τροποποίηση και διανομή. Αυτό επιτρέπει σε οποιονδήποτε να δει, να χρησιμοποιήσει, να τροποποιήσει και να διανείμει το λογισμικό δωρεάν ή επί πληρωμή [4], [10].

Η χρήση του OSS συναντάται όλο και περισσότερο σε μελέτες περιπτώσεων, έρευνες και ελεγχόμενα πειράματα. Πιο συγκεκριμένα :

- Στα ελεγχόμενα πειράματα, το OSS χρησιμοποιείται στον σχεδιασμό του πειράματος και στην εκτέλεση των πειραμάτων, βοηθώντας στην ανάλυση των δεδομένων [5], [6],[9].
- Στις μελέτες περιπτώσεων μπορεί χρησιμοποιηθεί στη συλλογή και στην ανάλυση δεδομένων, καθώς προσφέρει πρόσβαση σε πληθώρα δεδομένων και δυνατότητα στατιστικής ανάλυσης [7],[8].
- Στις έρευνες μπορεί να χρησιμοποιηθεί στην ανάλυση των απαντήσεων[9].

1.3 Κίνητρο Έρευνας

Η αφθονία των έργων ανοιχτού λογισμικού καθιστά δύσκολο για τους ερευνητές να επιλέξουν μια κατάλληλη συλλογή λογισμικού (dataset) για τις εμπειρικές τους μελέτες.

Παρακάτω περιγράφονται κάποιες από τις ποιο σημαντικές προκλήσεις που πρέπει να αντιμετωπίσουν οι ερευνητές όταν επιλέγουν κατάλληλες συλλογές OSS για τις μελέτες τους:

- **Διασφάλιση ποιότητας (Quality assurance):** Με τόσες πολλές διαθέσιμες επιλογές OSS, μπορεί να είναι δύσκολο να καθοριστεί ποιες είναι αξιόπιστες και καλής ποιότητας. Οι ερευνητές πρέπει να είναι προσεκτικοί στην επιλογή λογισμικού που έχει δοκιμαστεί και επαληθευτεί από άλλους στην ερευνητική κοινότητα [15], [17], [18]
- **Συμβατότητα (Compatibility):** Το OSS μπορεί να μην είναι πάντα συμβατό με τα υπάρχοντα εργαλεία λογισμικού ή τις μορφές δεδομένων του ερευνητή. Τα ζητήματα συμβατότητας μπορεί να προκαλέσουν καθυστερήσεις στην ερευνητική διαδικασία και μπορεί να απαιτήσουν επιπλέον χρόνο και πόρους για την επίλυσή τους [14], [17],[21], [27].
- **Καμπύλη μάθησης (Learning Curve):** Οι ερευνητές μπορεί να χρειαστεί να επενδύσουν χρόνο και προσπάθεια για να μάθουν πώς να χρησιμοποιούν αποτελεσματικά το λογισμικό. Αυτό μπορεί να είναι ιδιαίτερα δύσκολο εάν το λογισμικό απαιτεί σημαντική τεχνική εξειδίκευση ή εάν ο ερευνητής δεν είναι εξοικειωμένος με τη γλώσσα προγραμματισμού που χρησιμοποιείται από το λογισμικό. Η καμπύλη μάθησης αναφέρεται στον χρόνο και την προσπάθεια που απαιτείται για να μάθουν οι ερευνητές πώς να χρησιμοποιείτε ένα εργαλείο λογισμικού αποτελεσματικά και αποδοτικά [20].
- **Συντήρηση και υποστήριξη (Maintenance and support):**Το OSS ενδέχεται να μην έχει το ίδιο επίπεδο συντήρησης και υποστήριξης με το εμπορικό λογισμικό, γεγονός που μπορεί να οδηγήσει σε προβλήματα με σφάλματα(bugs), ασφάλειας και ενημερώσεων. Οι ερευνητές πρέπει να είναι προετοιμασμένοι να χειριστούν αυτά τα ζητήματα οι ίδιοι ή να αναζητήσουν βοήθεια από την κοινότητα. Η συντήρηση αναφέρεται στις δραστηριότητες που απαιτούνται για τη διατήρηση της αποτελεσματικής και αποδοτικής λειτουργίας του λογισμικού, ενώ η υποστήριξη αναφέρεται στις δραστηριότητες που απαιτούνται για την αντιμετώπιση προβλημάτων και την παροχή βοήθειας στους χρήστες [15],[14],[17],[22],[27].
- **Έλλειψη τεκμηρίωσης (Poor documentation)**: Η διαθεσιμότητα και η ποιότητα της τεκμηρίωσης για το λογισμικό είναι σημαντικές για την κατανόηση της λειτουργικότητας και των δυνατοτήτων του και για την αντιμετώπιση προβλημάτων. Η τεκμηρίωση αναφέρεται στο επίπεδο λεπτομέρειας και σαφήνειας των πληροφοριών που παρέχονται σχετικά με τη συλλογή, συμπεριλαμβανομένων των οδηγών χρήσης, της τεκμηρίωσης API και του δείγματος κώδικα [14-17],[19],[27]. Η τεκμηρίωση είναι σημαντική γιατί μπορεί να επηρεάσει την ευκολία χρήσης και κατανόησης της συλλογής από τους ερευνητές. Μια συλλογή με ολοκληρωμένη και σαφή τεκμηρίωση μπορεί να διευκολύνει τους ερευνητές να κατανοήσουν και να χρησιμοποιήσουν τη συλλογή, μειώνοντας την πιθανότητα σφαλμάτων και αυξάνοντας την αποτελεσματικότητα της ερευνητικής διαδικασίας. Η αξιολόγηση της τεκμηρίωσης μπορεί να είναι δύσκολη, καθώς η ποιότητα και το επίπεδο λεπτομέρειας που παρέχονται στην τεκμηρίωση μπορεί να διαφέρουν πολύ μεταξύ των συλλογών. Επιπλέον, η τεκμηρίωση μπορεί να μην είναι πάντα ενημερωμένη ή να μην αντικατοπτρίζει αλλαγές στη συλλογή. Όταν επιλέγουν συλλογές λογισμικού ανοιχτού κώδικα για εμπειρικές μελέτες, οι ερευνητές θα πρέπει να λαμβάνουν υπόψη την τεκμηρίωση της συλλογής και να διασφαλίζουν ότι είναι κατάλληλη για τον ερευνητικό σκοπό. Οι ερευνητές μπορούν να αξιολογήσουν την τεκμηρίωση εξετάζοντας το επίπεδο λεπτομέρειας και σαφήνειας

που παρέχονται στους οδηγούς χρήσης, την τεκμηρίωση API και το δείγμα κώδικα και δοκιμάζοντας τη συλλογή για να διασφαλίσουν ότι η τεκμηρίωση αντικατοπτρίζει με ακρίβεια τη λειτουργικότητα και τη συμπεριφορά της συλλογής. Επιπλέον, οι ερευνητές μπορούν να συμβουλευτούν ειδικούς στον τομέα για να λάβουν καθοδήγηση σχετικά με την αξιολόγηση της τεκμηρίωσης και τον εντοπισμό πιθανών ζητημάτων τεκμηρίωσης.

Από τις παραπάνω δυσκολίες προκύπτει ότι υπάρχει ανάγκη για ***κατευθυντήριες γραμμές*** προκειμένου να βοηθηθούν οι ερευνητές ώστε να ξεπεράσουν τις προκλήσεις αυτές και να επιλέξουν κατάλληλες συλλογές λογισμικού για τις μελέτες τους.

Η σημασία των κατευθυντήριων γραμμών στις μελέτες αναλύεται παρακάτω:

- **Αποδοτικότητα (Efficiency)** [23]:
  Με τόσες πολλές διαθέσιμες επιλογές λογισμικού ανοιχτού κώδικα, οι ερευνητές μπορούν να σπαταλήσουν πολύ χρόνο και πόρους αξιολογώντας και δοκιμάζοντας λογισμικό που δεν είναι κατάλληλο για τις ερευνητικές τους ανάγκες. Οι κατευθυντήριες γραμμές μπορούν να βοηθήσουν τους ερευνητές να εντοπίσουν γρήγορα τις πιο σχετικές επιλογές, δίνοντάς τους τη δυνατότητα να επικεντρωθούν στα ερευνητικά τους καθήκοντα πιο αποτελεσματικά.
- **Αναπαραγωγιμότητα (Reproducibility)** [23]:
  Η αναπαραγωγιμότητα είναι απαραίτητη για τη διασφάλιση της αξιοπιστίας και της αξιοπιστίας της έρευνας, και το λογισμικό ανοιχτού κώδικα μπορεί να διαδραματίσει κρίσιμο ρόλο στη διευκόλυνση της αναπαραγωγιμότητα. Οι ερευνητές πρέπει να διασφαλίζουν ότι η έρευνά τους είναι αναπαραγώγιμη, πράγμα που σημαίνει ότι άλλοι ερευνητές μπορούν να αναπαράγουν τα ίδια αποτελέσματα ακολουθώντας τις ίδιες μεθόδους και χρησιμοποιώντας το ίδιο λογισμικό. Οι κατευθυντήριες γραμμές μπορούν να βοηθήσουν τους ερευνητές να εντοπίσουν και να επιλέξουν λογισμικό που έχει δοκιμαστεί και επαληθευτεί από την ερευνητική κοινότητα, προωθώντας την αναπαραγωγιμότητα
- **Ποιοτικός έλεγχος (Quality control)** [23]:
  Ο ποιοτικός έλεγχος είναι σημαντικός παράγοντας κατά την επιλογή συλλογών λογισμικού ανοιχτού κώδικα για ερευνητικούς σκοπούς. Χωρίς οδηγίες για την αξιολόγηση των επιλογών λογισμικού, οι ερευνητές μπορούν να επιλέξουν λογισμικό που είναι αναξιόπιστο, ασταθές ή κακής απόδοσης, το οποίο μπορεί να θέσει σε κίνδυνο την ποιότητα και την αξιοπιστία των ερευνητικών τους αποτελεσμάτων. Οι κατευθυντήριες γραμμές μπορούν να βοηθήσουν τους ερευνητές να επιλέξουν λογισμικό που έχει ελεγχθεί και δοκιμαστεί από την κοινότητα. Αυτό μπορεί να προωθήσει τον ποιοτικό έλεγχο και να δώσει τη δυνατότητα στους ερευνητές να αναπτύξουν λύσεις λογισμικού που είναι αξιόπιστες και αποτελεσματικές.
- **Διαλειτουργικότητα (Interoperability)** [23]**:**
  Η διαλειτουργικότητα είναι ένα σημαντικό πλεονέκτημα της χρήσης λογισμικού ανοιχτού κώδικα σε ερευνητικές μελέτες. Η ικανότητα διαφορετικών συστημάτων λογισμικού να συνεργάζονται απρόσκοπτα είναι ζωτικής σημασίας για τους ερευνητές που πρέπει να συνδυάσουν δεδομένα από πολλαπλές πηγές ή να ενσωματώσουν διαφορετικά στοιχεία λογισμικού για τις μελέτες τους. Ωστόσο, η επίτευξη διαλειτουργικότητας μπορεί να είναι πρόκληση, ειδικά με τον αυξανόμενο αριθμό διαθέσιμων έργων λογισμικού ανοιχτού κώδικα.
  Οι κατευθυντήριες γραμμές για την επιλογή συλλογών λογισμικού ανοιχτού κώδικα μπορούν να παροτρύνουν τους ερευνητές να ξεπεράσουν τις προκλήσεις διαλειτουργικότητας προτείνοντας κριτήρια για την επιλογή λογισμικού. Τέτοιες οδηγίες ενδέχεται να τονίσουν τη σημασία της επιλογής λογισμικού που υποστηρίζει τυπικές γλώσσες κωδικοποίησης, API και

μορφές δεδομένων. Αυτό μπορεί να βοηθήσει το λογισμικό να ενσωματωθεί εύκολα με άλλα συστήματα, μειώνοντας τον χρόνο και την προσπάθεια που απαιτείται για την ενοποίηση. Μπορεί επίσης να προτείνουν στους ερευνητές να δίνουν προτεραιότητα σε έργα λογισμικού ανοιχτού κώδικα που έχουν μια ισχυρή κοινότητα προγραμματιστών και χρηστών. Αυτό μπορεί να βοηθήσει να διασφαλιστεί ότι το λογισμικό ενημερώνεται και συντηρείται τακτικά, καθώς και ότι υπάρχει καλή τεκμηρίωση και υποστήριξη για διαλειτουργικότητα και ενοποίηση.
Για να διασφαλιστεί η διαλειτουργικότητα, οι οδηγίες μπορεί να συστήνουν στους ερευνητές να χρησιμοποιούν λογισμικό που παρέχει υποστήριξη για μεταφορά δεδομένων μεταξύ διαφορετικών συστημάτων και υποστηρίζει την ενοποίηση του συστήματος με άλλα στοιχεία λογισμικού. Έτσι, τα δεδομένα μπορούν να μεταφερθούν και να συνδυαστούν εύκολα και τα στοιχεία λογισμικού μπορούν να ενσωματωθούν εύκολα για να δημιουργήσουν ένα απρόσκοπτο περιβάλλον έρευνας.
Τέλος, οι κατευθυντήριες γραμμές μπορεί να προτείνουν στους ερευνητές να συνεργαστούν με την κοινότητα λογισμικού ανοιχτού κώδικα για να λάβουν υποστήριξη και συμβουλές σχετικά με τον τρόπο επίτευξης διαλειτουργικότητας. Αυτό μπορεί να περιλαμβάνει τη συμμετοχή σε φόρουμ(forum), τη συμμετοχή σε συνέδρια και τη συνεργασία με άλλους ερευνητές που έχουν εμπειρία με το λογισμικό.

- **Συνεργασία (Collaboration)** [24]:
  Η συνεργασία είναι μια κρίσιμη ανάγκη στις ερευνητικές μελέτες και οι οδηγίες μπορούν να βοηθήσουν στη διευκόλυνση της συνεργασίας μεταξύ ερευνητών που χρησιμοποιούν λογισμικό ανοιχτού κώδικα. Οι κατευθυντήριες γραμμές μπορούν να παρέχουν ένα πλαίσιο για αποτελεσματική συνεργασία, βοηθώντας να διασφαλιστεί ότι οι ερευνητές μπορούν να συνεργαστούν αποτελεσματικά για την επίτευξη των ερευνητικών τους στόχων.

Παρόλες τις προκλήσεις που μπορεί να προκύπτουν και την επίκτητη ανάγκη κατευθυντήριων γραμμών για την επίλυση τους, το OSS είναι ιδιαίτερα ωφέλιμο σε εμπειρικές μελέτες και έρευνες για τους εξής λόγους:

- **Αναπαραγωγιμότητα (Reproducibility)** [25],[29]: Το λογισμικό ανοιχτού κώδικα επιτρέπει την εύκολη κοινή χρήση κώδικα και δεδομένων, γεγονός που διευκολύνει τους ερευνητές να αναπαράγουν ο ένας την εργασία του άλλου. Αυτό όχι μόνο συμβάλλει στη διασφάλιση της ακρίβειας της έρευνας, αλλά προάγει επίσης τη διαφάνεια και τη λογοδοσία. Η αναπαραγωγιμότητα οδηγεί στην αυξημένη διαφάνεια και αξιοπιστία , τη βελτιωμένη συνεργασία, την διευκόλυνση της επικύρωση και της αναπαραγωγής ,καθώς και την εξοικονόμηση χρόνου και πόρων.
- **Προσαρμογή (Customization)**[28],[29]**:** Το λογισμικό ανοιχτού κώδικα μπορεί να προσαρμοστεί για να ταιριάζει στις συγκεκριμένες ανάγκες ενός ερευνητικού έργου. Οι ερευνητές μπορούν να τροποποιήσουν το λογισμικό για να προσθέσουν νέες δυνατότητες, να βελτιστοποιήσουν την απόδοση ή να το ενσωματώσουν με άλλα εργαλεία για να ταιριάζουν καλύτερα στους ερευνητικούς τους στόχους. Τα οφέλη της προσαρμογής είναι οι εξατομικευμένες λύσεις, το μειωμένο κόστος, την ευελιξία , την καινοτομία και την υποστήριξη της κοινότητας.
- **Συνεργασία (Collaboration)**[25],[27],[29]**:** Το OSS ενθαρρύνει τη συνεργασία και την ανταλλαγή γνώσεων μεταξύ των ερευνητών. Συνεργαζόμενοι για την ανάπτυξη και τη βελτίωση του λογισμικού, οι ερευνητές μπορούν να επωφεληθούν από την τεχνογνωσία του άλλου και να βασιστούν ο ένας στη δουλειά του άλλου. Η συνεργασία ωφελεί στην κοινή γνώμη, την

αυξημένη παραγωγικότητα, στο μειωμένο κόστος, την καινοτομία και την υποστήριξη της κοινότητας.

- **Κόστος (Cost),** [26 - 29]**:** Το OSS είναι συχνά δωρεάν ή τουλάχιστον σημαντικά λιγότερο ακριβό από το ιδιόκτητο λογισμικό. Αυτό μπορεί να είναι ιδιαίτερα σημαντικό για ερευνητές με περιορισμένους προϋπολογισμούς. Από την εξοικονόμηση κόστους λόγω χρήσης OSS προκύπτει μειωμένο κόστος ανάπτυξης, απουσία τελών αδειοδότησης, πρόσβαση σε μεγάλη κοινότητα και χαμηλό συνολικό κόστος ιδιοκτησίας.
- **Καινοτομία (Innovation),** Wheeler, D. [26 - 29]**:** Το OSS βρίσκεται συχνά στην πρώτη γραμμή της καινοτομίας, με νέες δυνατότητες και βελτιώσεις που αναπτύσσονται και μοιράζονται τακτικά. Οι ερευνητές που χρησιμοποιούν λογισμικό ανοιχτού κώδικα μπορούν να επωφεληθούν από αυτές τις καινοτομίες και να παραμένουν ενημερωμένοι με τα πιο πρόσφατα εργαλεία και τεχνικές.

  Η καινοτομία είναι ένα σημαντικό πλεονέκτημα από τη χρήση λογισμικού ανοιχτού κώδικα σε ερευνητικές μελέτες. Το λογισμικό ανοιχτού κώδικα παρέχει στους ερευνητές πρόσβαση σε μια μεγάλη κοινότητα χρηστών και προγραμματιστών που συμβάλλουν στην ανάπτυξη και τη βελτίωση του λογισμικού. Αυτό μπορεί να προωθήσει την καινοτομία και να δώσει τη δυνατότητα στους ερευνητές να αναπτύξουν νέες λύσεις λογισμικού που είναι βελτιστοποιημένες για συγκεκριμένες ερευνητικές απαιτήσεις.

Επομένως, στόχος της συγκεκριμένης εργασίας είναι να περιγράφουν κατευθυντήριες γραμμές που θα βοηθήσουν τους ερευνητές να ξεπεράσουν τις προκλήσεις και να επιλέξουν τις κατάλληλες συλλογές λογισμικού για τις μελέτες τους. Για την επίτευξη του στόχου, πρώτον εφαρμόσαμε μια συστηματική βιβλιογραφική χαρτογράφηση εμπειρικών μελετών ανοιχτού λογισμικού δημοσιευμένων σε ένα εύρος περιοδικών την περίοδο 2021-2022. Δεύτερον, δημιουργήσαμε ένα εργαλείο ,ευρετήριο μελετών, όπου οι ερευνητές θα μπορούν να αναζητούν και να προσθέτουν εμπειρικές μελέτες.

Η υπόλοιπη εργασία είναι οργανωμένη ως εξής: στο κεφάλαιο 2 παρουσιάζεται η σχετική βιβλιογραφία, στο κεφάλαιο 3 αναλύεται η μεθοδολογία που χρησιμοποιήθηκε στη συστηματική βιβλιογραφική χαρτογράφηση και στην ανάπτυξη του εργαλείου. Στο κεφάλαιο 4 παρουσιάζονται τα αποτελέσματα. Τέλος, στο κεφάλαιο 5 παρουσιάζονται τα συμπεράσματα ,καθώς και πιθανές μελλοντικές εργασίες.

Κεφάλαιο 2ο: Βιβλιογραφική Ανασκόπηση

2.1 Εξόρυξη Δεδομένων Από Αποθετήρια Λογισμικού (MSR)

Το Mining Software Repositories (MSR) είναι ένα ερευνητικό πεδίο που εστιάζει στην εξαγωγή πολύτιμων πληροφοριών και γνώσεων από αποθετήρια λογισμικού. Τα αποθετήρια λογισμικού περιέχουν τεράστιες ποσότητες δεδομένων που παράγονται κατά τη διαδικασία ανάπτυξης λογισμικού, συμπεριλαμβανομένων συστημάτων ελέγχου εκδόσεων (π.χ. το Concurrent Versions System (CVS)), συστημάτων παρακολούθησης σφαλμάτων (π.χ. Bugzilla) και καναλιών επικοινωνίας (π.χ. e-mail). [30]

Ο πρωταρχικός στόχος του MSR είναι να εφαρμόσει τεχνικές εξόρυξης δεδομένων, [31] για την ανάλυση αυτών των δεδομένων και την απόκτηση γνώσεων σχετικά με διάφορες πτυχές της ανάπτυξης, συντήρησης και εξέλιξης λογισμικού. Αναλύοντας τα δεδομένα που είναι αποθηκευμένα σε αποθετήρια, οι ερευνητές μπορούν να αποκαλύψουν μοτίβα, τάσεις και σχέσεις που μπορούν να ενημερώσουν τη λήψη αποφάσεων, να βελτιώσουν την ποιότητα του λογισμικού και να βελτιώσουν τις διαδικασίες ανάπτυξης. Η συστηματική μελέτη που πραγματοποίησαμε είχε ως δευτερεύουσες σχετικές μελέτες, έρευνες που εστίασαν σε μια γενική ανάλυση μελετών στο MSR πεδίο.

Στην πρώτη μελέτη [30] παρουσιάζονται δύο ενότητες . Η πρώτη είναι μια ολοκληρωμένη έρευνα (comprehensive literature survey) των προσεγγίσεων MSR, στο πλαίσιο της εξέλιξης λογισμικού, και η δεύτερη είναι μια ταξινόμηση αυτών των προσεγγίσεων, ειδικά στο πλαίσιο της εξέλιξης του λογισμικού. Επιδιώκει να παρέχει μια επισκόπηση των διαφορετικών τεχνικών και μεθοδολογιών εξόρυξης που χρησιμοποιούνται για την εξαγωγή πολύτιμων πληροφοριών από τα αποθετήρια λογισμικού για την κατανόηση της εξέλιξης του λογισμικού και των σχετικών διαδικασιών.

Η έρευνα καλύπτει ένα ευρύ φάσμα θεμάτων, συμπεριλαμβανομένων τεχνικών συλλογής δεδομένων, αλγορίθμων εξόρυξης και μεθόδων ανάλυσης που χρησιμοποιούνται στο ερευνητικό πεδίο του MSR. Κατηγοριοποιεί τις προσεγγίσεις σε διαφορετικές κατηγορίες με βάση τους στόχους και τα χαρακτηριστικά τους, παρέχοντας μια δομημένη ταξινόμηση στους ερευνητές και τους επαγγελματίες για την αποτελεσματική αναζήτηση στο πεδίο. Μέσα από βιβλιογραφική ανασκόπηση το άρθρο υπογραμμίζει τα μοτίβα εξέλιξης, τις τάσεις και τις προκλήσεις που αντιμετωπίζει η ανάπτυξη λογισμικού και διερευνά τον τρόπο που οι τεχνικές εξόρυξης μπορούν να συμβάλουν στη συντήρηση λογισμικού, στην πρόβλεψη σφαλμάτων και στη βελτίωση της ποιότητας του λογισμικού.

Τα ευρήματα αυτής της έρευνας χρησιμεύουν ως πολύτιμη πηγή για ερευνητές, μηχανικούς λογισμικού και επαγγελματίες που ενδιαφέρονται να αξιοποιήσουν τα αποθετήρια λογισμικού για την κατανόηση της εξέλιξής του. Στην ενότητα της ταξινόμησης που παρέχεται στο άρθρο προσφέρει ένα πλαίσιο σύγκρισης και αξιολόγησης διαφορετικών προσεγγίσεων, βοηθώντας στην επιλογή των κατάλληλων μεθόδων για συγκεκριμένες ερευνητικές ή πρακτικές ανάγκες. Η περίοδος των ερευνών που χρησιμοποιήθηκαν ήταν από το 1994-2004. Συνολικά ταξινομήθηκαν 80 προσεγγίσεις.

Στη δεύτερη μελέτη, η διερευνητική μελέτη (mapping study) [31], αναφερόταν στις πρόσφατες βελτιώσεις λογισμικού καθώς και στον τρόπο των αναβαθμίσεων του. Το άρθρο ενημερώνει τα ευρήματα MSR του αρχικού MSR CookBook, για την εξαγωγή και ανάλυση της τελευταίας τεχνολογίας και στη συνέχεια προτείνει μια εκτεταμένη έκδοση του CookBook. Η συλλογή δεδομένων πραγματοποιήθηκε στο διάστημα Σεπτεμβρίου-Νοεμβρίου του 2020. Σε αυτήν συμπεριλαμβανόταν μελέτες ,οι οποίες σχετίζονταν με τα Αποθετήρια Λογισμικού(Software Repositories) και χρησιμοποιούσαν τεχνικές εξόρυξης (Mining). Αντίθετα , μελέτες οι οποίες α) δημοσιεύτηκαν πριν το

2013, β) δεν ήταν γραμμένες στα αγγλικά, γ) ήταν παρόμοιες με την συγκεκριμένη έρευνα, δ) ήταν ανασκοπήσεις και ε) δεν σχετίζονταν με Αποθετήρια Λογισμικού ή σχετίζονταν αλλά δεν χρησιμοποιούσαν τεχνικές εξόρυξης. Αυτός ο διαχωρισμός οδήγησε σε μία τελική λίστα 112 επιλεγμένων μελετών, μέσω, των οποίων αποκαλύφθηκε αυξημένο ενδιαφέρον για τις τεχνικές εξόρυξης και την ανάπτυξη εργαλείων, οι οποίες χρησιμοποιήθηκαν για την δημιουργία ενός βελτιωμένου CookBook ως συνεισφορά στην πρακτική και την έρευνα σε διάφορους τομείς, καθώς συμβάλλει στη συνεχή ανάπτυξη του τομέα.. Το βελτιωμένο CookBook μπορεί να υποστηρίξει νέα ερευνητικά έργα, καθώς περιλαμβάνει νέες ανανεωμένες συστάσεις και διδάγματα τα οποία είναι διαθέσιμα μέσω των εργαλείων.

Τέλος, στη μελέτη από τους De F. Farias et al. [32], πραγματοποιείται μία αναζήτηση σε μία σειρά άρθρων του Working Conference on Mining Software Repositories (MSRConf), με σκοπό να δοθεί η ανάλυση και κατηγοριοποίηση των υπαρχόντων ερευνών του MSRConf στον τομέα αυτό. Η μελέτη στοχεύει στον εντοπισμό των τάσεων, των ερευνητικών θεμάτων, των μεθοδολογιών και των προκλήσεων που αντιμετωπίζονται στο MSR. Η παροχή αυτών των πολύτιμων πληροφοριών για την τρέχουσα κατάσταση της έρευνας του MSR, μπορεί να βοηθήσει τους ερευνητές και τους επαγγελματίες να περιηγηθούν στη βιβλιογραφία και να εντοπίσουν πιθανές ανακρίβειες ή πιθανούς τομείς για περαιτέρω διερεύνηση.

Η διερευνητική μελέτη (mapping study) διεξήχθη από το 2010 έως το 2014. Σε αυτή, οι συγγραφείς ταξινόμησαν τα άρθρα , συμπεριλαμβάνοντας όσα α) έχουν μία ολοκληρωμένη μελέτη, δημοσιευμένη στο MSRConf τη συγκεκριμένη περίοδο και β) διερευνούν μία θεωρία, πρακτική ή προσέγγιση σχετική με το MSR και τον τομέα της μηχανικής λογισμικού και απορρίπτοντας όσα α) δεν αφορούσαν το MSR και τον τομέα της μηχανικής λογισμικού και β) βασίζονταν σε πειραματικές μελέτες . Το σύνολο των άρθρων προς μελέτη ήταν 107. Μέσω τη διερεύνησης μελετών MSR με λανθασμένο αποτέλεσμα, κατέληξαν ότι η «κατανόηση των ελαττωμάτων» (defects) και ο «κώδικας» ήταν ο πιο χρησιμοποιούμενος στόχος και αντικείμενο ανάλυσης σε αυτόν τον τομέα. Τέλος μέσω ταξινόμησης των δεδομένων, αντιλήφθηκαν ότι τα δομημένα αποθετήρια διερευνώνται περισσότερο από τα μη δομημένα, αλλά ο αριθμός των προσεγγίσεων που χρησιμοποιούν μη δομημένες πηγές δεδομένων αυξάνεται τα τελευταία.

2.2 Κατευθυντήριες Γραμμές Για Διενέργεια Εμπειρικών Μελετών

Καθώς ο ερευνητικός τομέας ωριμάζει, συχνά αυξάνεται απότομα ο αριθμός των αναφορών και των αποτελεσμάτων που διατίθενται, και καθίσταται σημαντικό να συνοψίζεται και να παρέχεται επισκόπηση. Πολλά ερευνητικά πεδία έχουν συγκεκριμένες μεθοδολογίες για τέτοιες δευτερεύουσες μελέτες και έχουν χρησιμοποιηθεί εκτενώς.

Χαρακτηριστικό παράδειγμα , αποτελεί η έρευνα [33], η οποία παρέχει μια ολοκληρωμένη επισκόπηση των βασικών αρχών και πρακτικών της έρευνας. Στο συγκεκριμένο άρθρο εφαρμόστηκαν τα βήματα της έρευνας (survey) [3] που αναφέραμε στο κεφάλαιο 1.1 , ως εξής :

1. *Προσδιορισμός στόχου*: Η πρώτη αρχή που συζητήθηκε είναι η σαφήνεια του σκοπού. Οι συγγραφείς τονίζουν τη σημασία του καθορισμού των στόχων και της ανάπτυξης συγκεκριμένων ερευνητικών ερωτημάτων της έρευνας. Τονίζουν την ανάγκη να συντονιστεί ο σχεδιασμός της έρευνας με τους ερευνητικούς στόχους για να διασφαλιστεί ότι τα δεδομένα που συλλέγονται θα είναι χρήσιμα και σχετικά.

2. *Προσδιορισμός συμμετεχόντων στην έρευνα*: Η δεύτερη αρχή είναι η επιλογή δείγματος. Τονίζουν την ανάγκη επιλογής ενός αντιπροσωπευτικού δείγματος που αντικατοπτρίζει με ακρίβεια τον πληθυσμό-στόχο.
3. *Επιλογή του τύπου της έρευνας (έντυπη, ηλεκτρονική ή συνέντευξη)*: Το άρθρο καλύπτει επίσης μεθόδους συλλογής δεδομένων, συμπεριλαμβανομένων ερωτηματολογίων, τηλεφωνικών συνεντεύξεων και διαδικτυακών ερευνών. Υπογραμμίζει τα πλεονεκτήματα και τα μειονεκτήματα κάθε μεθόδου και παρέχει πληροφορίες για τη βελτιστοποίηση των ποσοστών απόκρισης.
4. *Σχεδιασμός ερωτήσεων*: Το άρθρο εξετάζει το σχεδιασμό του ερωτηματολογίου. Τονίζει τη σημασία της δημιουργίας σαφών, συνοπτικών και αμερόληπτων ερωτήσεων για να διασφαλιστεί ότι οι ερωτώμενοι τις κατανοούν και τις ερμηνεύουν σωστά. Οι συγγραφείς παρέχουν κατευθυντήριες γραμμές για τη διατύπωση και τη σειρά των ερωτήσεων και τις επιλογές απαντήσεων για την ελαχιστοποίηση του σφάλματος και της μεροληπτικής απόκρισης.
5. *Πιλοτική μελέτη*: Τονίζουν την αξία της διεξαγωγής μικρής κλίμακας δοκιμών για τον εντοπισμό πιθανών προβλημάτων, την αξιολόγηση της σαφήνειας των ερωτήσεων και τη βελτίωση της διαχείρισης της έρευνας πριν από την έναρξη της μελέτης.
6. *Διανομή της έρευνας*: Η διανομή της έρευνας πραγματοποιείται με τρόπο ανάλογο του τύπου της έρευνας (π.χ. η διανομή του ηλεκτρονικού τύπου μπορεί να γίνει μέσω mail).
7. *Ανάλυση των απαντήσεων και αποτελέσματα*: Αφού επιστραφούν απαντημένα τα ερωτηματολόγια και ολοκληρωθούν οι συνεντεύξεις, αναλύονται οι απαντήσεις και εξάγονται τα αντίστοιχα αποτελέσματα.

Η μελέτη [34] παρέχει πολύτιμες οδηγίες για τη διεξαγωγή και την αναφορά αποτελεσμάτων της μελέτης περίπτωσης στον τομέα της μηχανικής λογισμικού, ώστε να προσφέρει ένα πλαίσιο που βοηθά τους ερευνητές να διεξάγουν αυστηρές και διαφανείς μελέτες περιπτώσεων και να αποκτήσουν γνώσεις σχετικά με τις πραγματικές πρακτικές, τις διαδικασίες και τα αποτελέσματα ανάπτυξης λογισμικού. Τα στάδια που αναγράφονται στο [2] περιγράφονται αναλυτικά παρακάτω:

1. *Σχεδιασμός μελέτης περίπτωσης:* Το στάδιο αυτό εστιάζει στην παροχή καθοδήγησης σχετικά με τον τρόπο σχεδίασης μιας μελέτης περίπτωσης (ιδεολογία, σκοπός μελέτης και ερευνητικά ερωτήματα). Οι συγγραφείς τονίζουν τη σημασία της διατύπωσης σαφών ερευνητικών ερωτημάτων και του καθορισμού των ορίων και του πεδίου της μελέτης.
2. *Προετοιμασία για συλλογή δεδομένων:* Στο σημείο αυτό παρέχεται καθοδήγηση σχετικά με το πρωτόκολλο και τον τρόπο συλλογής σχετικών και αξιόπιστων δεδομένων για μια μελέτη περίπτωσης στη μηχανική λογισμικού (στρατηγική και μέθοδος επιλογής δεδομένων). Οι συγγραφείς τονίζουν την ανάγκη για μια συστηματική και καλά τεκμηριωμένη προσέγγιση στη συλλογή δεδομένων.
3. *Συλλογή δεδομένων μελέτης:*Η συλλογή δεδομένων πραγματοποιείται συνήθως μέσω συνεντεύξεων, αρχειοθετημένων στοιχείων και μετρικών – ποσοτικών δεδομένων.
4. *Ανάλυση και ερμηνεία των δεδομένων:*Οι συγγραφείς παρέχουν καθοδήγηση σχετικά με την οργάνωση, τη διαχείριση, την κωδικοποίηση και την κατηγοριοποίηση των δεδομένων και τη χρήση διαφόρων αναλυτικών τεχνικών όπως η αντιστοίχιση προτύπων, η οικοδόμηση επεξηγήσεων και η σύνθεση διασταυρούμενων περιπτώσεων. Επιπλέον, αναφέρονται στον τρόπο ερμηνείας των αποτελεσμάτων με βάση τα δεδομένα που αναλύθηκαν.

5. *Συμπεράσματα:*Οι συγγραφείς παρέχουν κατευθυντήριες γραμμές για τη δόμηση της αναφοράς μελέτης περίπτωσης, συμπεριλαμβανομένων των απαραίτητων στοιχείων όπως η εισαγωγή, το ιστορικό, η μεθοδολογία, τα ευρήματα και τα συμπεράσματα

Στο πεδίο της βιβλιογραφικής ανασκόπησης, οι [35] πραγματοποίησαν μία συστηματική διερευνητική μελέτη (systematic mapping study-SMS). Οι συγγραφείς περιγράφουν τη διαδικασία διεξαγωγής μίας SMS, η οποία περιλαμβάνει τον καθορισμό ερευνητικών ερωτημάτων, την επιλογή και αναζήτηση σχετικής βιβλιογραφίας μέσω αναζήτησης λέξεων-κλειδιών, την εφαρμογή κριτηρίων ένταξης και αποκλεισμού και την ανάλυση και ερμηνεία των ευρημάτων. Επιπλέον, στο [35] εξετάζονται τα οφέλη των SMS στην έρευνα της μηχανικής λογισμικού. Αρχικά, η SMS επιτρέπει στους ερευνητές να εντοπίσουν τα κύρια θέματα, τις τάσεις και τα κενά στην υπάρχουσα βιβλιογραφία, δίνοντάς τους τη δυνατότητα να λαμβάνουν τεκμηριωμένες αποφάσεις σχετικά με τις μελλοντικές κατευθύνσεις της έρευνας. Επιπλέον, συμβάλλει στον προσδιορισμό του εύρους και των ορίων του ερευνητικού πεδίου, παρέχοντας τη βάση για περαιτέρω εις βάθος μελέτες ή συστηματικές ανασκοπήσεις. Τέλος, η SMS επιτρέπει στους ερευνητές να εντοπίζουν ερευνητικές προκλήσεις, θεωρητικά πλαίσια, μεθοδολογίες και ερευνητικές μεθόδους που χρησιμοποιούνται στο πεδίο.

Στον αντίποδα, η συγκεκριμένη μελέτη αναγνωρίζει ορισμένους περιορισμούς της SMS, όπως την πιθανότητα μεροληψίας στην επιλογή και την ερμηνεία των μελετών και την έλλειψη επίσημης ποιοτικής αξιολόγησης. Ωστόσο, τονίζει ότι τα SMS είναι μια πολύτιμη προσέγγιση για την εξερεύνηση του υπάρχοντος γνωστικού πεδίου και την παροχή μιας τεκμηριωμένης βάσης για περαιτέρω έρευνα.

Με τη διεξαγωγή των SMS, οι ερευνητές μπορούν να αποκτήσουν γνώσεις για την υπάρχουσα βιβλιογραφία, να εντοπίσουν ερευνητικές ανακρίβειες και να αναβαθμίσουν μελλοντικές ερευνητικές κατευθύνσεις.

Επιπρόσθετα, στο [36] μελετήθηκε η μέθοδος Snowball, η οποία χρησιμοποιείται για τον εντοπισμό σχετικών μελετών ακολουθώντας αναδρομικά ίχνη παραπομπών από πρωτογενείς μελέτες σε σχετικές δημοσιεύσεις. Το άρθρο παρουσιάζει ένα σύνολο κατευθυντήριων γραμμών που παρέχουν μια συστηματική διαδικασία για τη διεξαγωγή της μεθόδου snowball συμπεριλαμβανομένων των σταδίων σχεδιασμού και εκτέλεσης. Αρχικά, η αναζήτηση των άρθρων περιλαμβάνει τον προσδιορισμό λέξεων-κλειδιών και τη διατύπωση σειρών χαρακτήρων προς αναζήτηση. Έπειτα, γίνεται έλεγχος των άρθρων με

- *Backward Snowballing*, κατά το οποίο ελέγχεται η λίστα αναφορών για τον εντοπισμό νέων εγγράφων που πρέπει να συμπεριληφθούν και εξαιρούνται τα έγγραφα που δεν πληρούν τα βασικά κριτήρια
- *Forward Snowballing*, όπου πραγματοποιείται ο εντοπισμός νέων εγγράφων με βάση τις βιβλιογραφικές αναφορές των εγγράφων που εξετάζονται. Η προσέγγιση για τον έλεγχο των νέων εγγράφων είναι παρόμοια με αυτή του Backward Snowballing.

Οι διαδικασίες αυτές συνεχίζονται ώσπου δεν εντοπίζονται καινούρια έγγραφα. Εναλλακτικές προσεγγίσεις είναι η επικοινωνία με τον συγγραφέα του άρθρου για την παροχή πρόσθετων δεδομένων και η αναζήτηση σε συγκεκριμένα περιοδικά ή συνέδρια που είναι πιθανό να περιλαμβάνουν περισσότερες εργασίες για το θέμα. Η εξαγωγή δεδομένων πρέπει να διεξάγεται σύμφωνα με τα ερευνητικά ερωτήματα που τέθηκαν στη συστηματική βιβλιογραφική μελέτη. Ακολουθώντας αυτές τις οδηγίες, οι ερευνητές μπορούν να βελτιώσουν την πληρότητα των ανασκοπήσεων της βιβλιογραφίας τους και να εξασφαλίσουν τη συμπερίληψη σχετικών μελετών που μπορεί να χαθούν μέσω των παραδοσιακών μεθόδων αναζήτησης.

Κεφάλαιο 3ο: Μεθοδολογία

Στο συγκεκριμένο κεφάλαιο παρουσιάζεται η μεθοδολογία την οποία ακολουθήσαμε για να εντοπίσουμε και να προτείνουμε κατευθυντήριες γραμμές προκειμένου να βοηθηθούν οι ερευνητές ώστε να επιλέξουν τις κατάλληλες συλλογές λογισμικού για τις μελέτες τους. Συγκεκριμένα, στο Κεφάλαιο 3.1 παρουσιάζεται η συστηματική βιβλιογραφική χαρτογράφηση που πραγματοποιήθηκε με σκοπό τη κατανόηση και τη συλλογή δεδομένων (π.χ. κριτήρια επιλογής έργων λογισμικού, επιλεγμένα έργα λογισμικού). Η συγκεκριμένη συστηματική ανασκόπηση όχι μόνο μας επέτρεψε να αποκτήσουμε πολύτιμες γνώσεις για τις πολύπλευρες πτυχές του λογισμικού ανοιχτού κώδικα, αλλά χρησίμευσε επίσης ως βάση για την ανάπτυξη ενός απαραίτητου εργαλείου το οποίο παρουσιάζεται στο Κεφάλαιο 3.2. Στόχος του συγκεκριμένου εργαλείου είναι να βοηθήσει τους ερευνητές στις μελλοντικές τους έρευνες, το οποίο μπορεί να χρησιμοποιηθεί ως καθοδηγητική πυξίδα, παρέχοντας πληθώρα πόρων και γνώσεων που θα τους επιτρέψουν να περιηγηθούν στο εκτεταμένο πεδίο του ανοιχτού λογισμικού με άνεση και ακρίβεια. Συγχωνεύοντας τα οφέλη της συστηματικής μας ανασκόπησης με τη δύναμη της τεχνολογίας, δημιουργήσαμε ένα πολύτιμο πλεονέκτημα που αναμφίβολα θα ενισχύσει ερευνητικές προσπάθειες σε αυτόν τον συνεχώς εξελισσόμενο τομέα.

3.1 Μεθοδολογία της Συστηματικής Βιβλιογραφικής Χαρτογράφησης

Σε αυτό το κεφάλαιο, παρουσιάζουμε το πρωτόκολλο της μελέτης συστηματικής χαρτογράφησης, με βάση τις οδηγίες που περιγράφονται από τον Petersen et al. [35]. Ένα πρωτόκολλο αποτελεί ένα σχέδιο που περιγράφει τα ερευνητικά ερωτήματα που διερευνήθηκαν και τον τρόπο με τον οποίο διεξήχθη η μελέτη χαρτογράφησης. Πιο συγκεκριμένα, το πρωτόκολλο περιλαμβάνει τρεις δραστηριότητες, και συγκεκριμένα: (α) καθορισμό ερευνητικών στόχων και ερωτημάτων —βλ. Ενότητα 3.1.1, (β) καθορισμός της διαδικασίας αναζήτησης και επιλογής άρθρων—βλ. Ενότητα 3.1.2 και (γ) ορισμός εξαγωγής δεδομένων, στρατηγική ανάλυσης και σύνθεσης—βλέπε Ενότητα 3.1.3

3.1.1 Στόχοι και ερευνητικά ερωτήματα

Ο στόχος αυτής της μελέτης, που εκφράζεται στη μορφή Goal-QuestionMetrics (GQM) [37], είναι να: *αναλύσει* υπάρχουσες MSR εμπειρικές μελέτες σχετικά με τον κώδικα ανοιχτού λογισμικού *με σκοπό* τον χαρακτηρισμό και την αξιολόγηση *σε σχέση με*: (α) τους στόχους των μελετών MSR; (β) τα κριτήρια επιλογής των έργων λογισμικού; και (γ) την επιλογή των έργων λογισμικού προς μελέτη, *από τη πλευρά του* ερευνητή. Με βάση αυτόν τον στόχο, ορίζουμε τα ακόλουθα ερευνητικά ερωτήματα (Research Questions—RQ):

RQ_1: Ποιοι είναι οι στόχοι των μελετών MSR που χρησιμοποιούν συλλογές έργων μέσω εξόρυξης από ανοιχτά αποθετήρια;

Το RQ_1 διερευνά τους στόχους των πρωτογενών μελετών MSR που χρησιμοποιούν συλλογές έργων που εξορύσσονται από ανοιχτά αποθετήρια, αναδεικνύοντας τη σημασία τους για τη βελτίωση των πρακτικών ανάπτυξης λογισμικού και τη δυνατότητα εμπειρικής έρευνας. Οι βασικοί στόχοι έχουν ανακτηθεί από τα ερευνητικά ερωτήματα των πρωτογενών μελετών (π.χ., testing, version control, refactoring).

RQ_2: Ποια είναι τα πιο κοινά κριτήρια επιλογής έργων ανοιχτού λογισμικού;

Το RQ_2 αναφέρεται στα πιο κοινά κριτήρια επιλογής έργων ανοιχτού λογισμικού, τονίζοντας τη σημασία τους για την εξασφάλιση επιτυχημένων μελετών. Για να διερευνήσουμε περαιτέρω αυτό το

ερώτημα, διερευνήσαμε ποια είναι τα πιο συχνά κριτήρια επιλογής έργων ανοιχτού λογισμικού με βάση τους στόχους μου έχουν τεθεί στις πρωτογενείς μελέτες.

RQ_3: Ποια είναι τα πιο συχνά χρησιμοποιούμενα έργα προς μελέτη;

Το RQ_3 αναφέρεται στα έργα που χρησιμοποιούνται πιο συχνά ως υποκείμενα στην έρευνα ανοιχτού κώδικα, τονίζοντας τη σημασία τους για την κατανόηση της ανάπτυξης λογισμικού και την ανακάλυψη πληροφοριών. Επιπλέον, για περαιτέρω διερεύνηση, ερευνήσαμε ποια είναι τα πιο συχνά έργα ανοιχτού λογισμικού με βάση τους στόχους μου έχουν τεθεί στις πρωτογενείς μελέτες.

3.1.2 Διαδικασία αναζήτησης και επιλογής άρθρων

Στη βιβλιογραφία μπορεί κάποιος να εντοπίσει δύο στρατηγικές για την αναζήτηση και επιλογή άρθρων: (α) την αυτοματοποιημένη αναζήτηση σε ψηφιακές βιβλιοθήκες, και (β) την χειροκίνητη αναζήτηση σε επιλεγμένα περιοδικά και συνέδρια. Λαμβάνοντας υπόψιν το πλήθος των MSR πρωτογενών μελετών και την ανάγκη εξόρυξης συγκεκριμένης πληροφορίας (π.χ., ερευνητικά ερωτήματα, κριτήρια επιλογής), θεωρήσαμε ότι είναι σκοπιμότερο και αποτελεσματικότερο να στοχεύσουμε σε επιστημονικά περιοδικά αναγνωρισμένου κύρους, όπως παρουσιάζονται και στον πίνακα 3.1. Η επιλογή περιοδικών έγινε με βάση τη μελέτη Wong et al. [38]. Τα άρθρα που συλλέχθηκαν σε πρώτο στάδιο ήταν 1492.

Πίνακας 3.1 Επιλεγμένοι τόποι δημοσίευσης και ψηφιακές βιβλιοθήκες

Venue	DL
Transactions on Software Engineering (TSE)	IEE
Information and Software Technology (IST) Journal of Systems and Software (JSS)	ScienceDirect
Transactions on Software Engineering and Methodology (TOSEM)	ACM
Empirical Software Engineering (ESE)	Springer

Στη συνέχεια για το σύνολο των άρθρων εφαρμόστηκε αναζήτηση μιας σειράς από λέξεις (βλ. παρακάτω πλαίσιο) στο πλήρες κείμενο των πρωτογενών μελετών, που δημοσιεύτηκαν στους προαναφερθέντες τόπους τα δύο τελευταία χρόνια. Ο στόχος αυτού του βήματος ήταν να επιστρέψει εμπειρικές μελέτες που σχετίζονται με το ανοιχτό λογισμικό.

(“open source” OR “open-source” OR “open projects”) AND (“empirical study” OR “experimental study” OR “case study”)

Μέσω της αναζήτησης που αναφέρθηκε παραπάνω καταλήξαμε σε 475 άρθρα, πουπληρούν τα κριτήρια ένταξης (Inclusion Criteria – IC) και αποκλεισμού (Exclusion Criteria – EC) .

Τα Κριτήρια Ένταξης της μελέτης μας είναι:

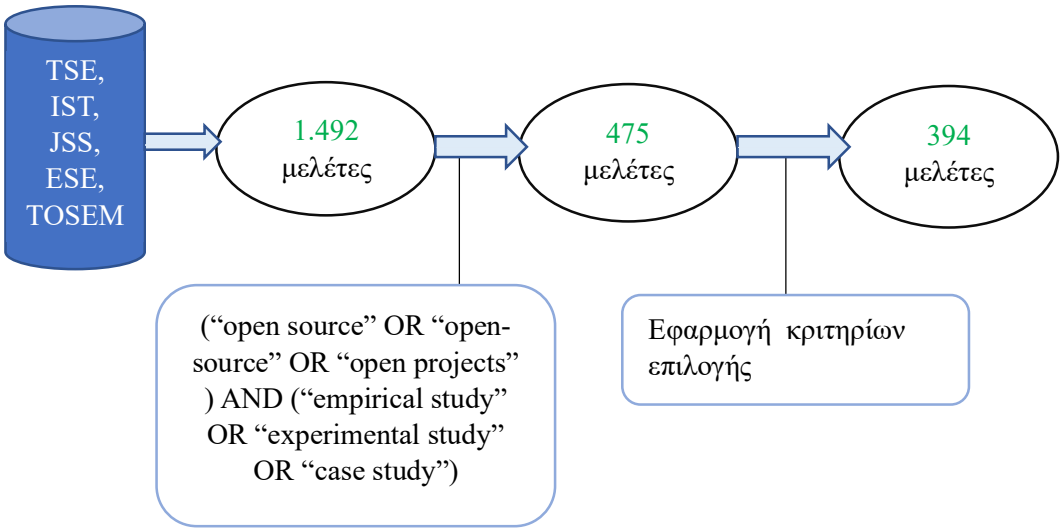
IC1—Η μελέτη ασχολείται με έργα ανοιχτού λογισμικού.
IC2 — Η μελέτη είναι μια εμπειρική μελέτη (δηλαδή, μελέτη περίπτωσης, έρευνα κτλ.).

Τα Κριτήρια Αποκλεισμού της μελέτης μας είναι:

EC1—Η μελέτη είναι γραμμένη σε γλώσσα άλλη από Αγγλικά

EC2—Η μελέτη είναι ένα δημοσιογραφικό άρθρο παράρτημα, κεντρική ομιλία/διάλεξη, σεμινάριο, αφίσα.

Στο τελικό σύνολο δεδομένων, διατηρήσαμε μελέτες που ικανοποιούσαν τόσο το IC1 όσο και το IC2 και δεν ικανοποιούσαν κανένα Κριτήριο Αποκλεισμού (EC). Η διαδικασία επιλογής άρθρων διεκπεραιώθηκε αρχικά από τη φοιτήτρια, και στη συνέχεια επιβεβαιώθηκε η επιλογή τους και από την επιβλέπουσα, χρησιμοποιώντας μια απλοποιημένη έκδοση της μεθόδου ψηφοφορίας, που προτάθηκε από τους Farhoodi et al. [39]. Συγκεκριμένα, επιθεώρησαμε το πλήρες κείμενο της πρωτογενής μελέτης και αναθέσαμε μια τιμή (ψήφος) συμπερίληψης σε κλίμακα Likert 4 βαθμών (4: ισχυρή συμπερίληψη, 1: αποκλεισμός συμβολοσειρών)—με μέγιστη βαθμολογία 8 βαθμούς. Στη μελέτη έχουν συμπεριληφθεί μελέτες με τιμές μεγαλύτερες του 6. Η συμπερίληψη των υπόλοιπων πρωτογενών μελετών συζητήθηκαν διεξοδικά. Συνολικά, δεδομένου ότι το επίπεδο σαφήνειας για την ένταξη/αποκλεισμό ήταν υψηλό, συζητήθηκαν μόνο 15 άρθρα. Στο Σχήμα 1, παρουσιάζεται μια επισκόπηση της διαδικασίας αναζήτησης και συλλογής μαζί με τον αριθμό των μελετών σε κάθε φάση. Στο τέλος, απέμειναν 394 άρθρα που θα συμπεριληφθούν σε αυτήν τη μελέτη (βλ. Σχετικές Βιβλιογραφικές Αναφορές) και πραγματοποιήθηκε η συλλογή δεδομένων.



Σχήμα 3.1 Επισκόπηση της διαδικασίας αναζήτησης και επιλογής άρθρων

3.1.3 Εξαγωγή, Ανάλυση και Σύνθεση Δεδομένων

Σε αυτή την ενότητα παρουσιάζουμε τη διαδικασία εξαγωγής, ανάλυσης και σύνθεσης δεδομένων που χρησιμοποιήσαμε για να απαντήσουμε στα ερευνητικά ερωτήματα της πτυχιακής. Η προτεινόμενη διαδικασία βασίζεται σε μεγάλο βαθμό σε μεθόδους σύνθεσης και μετα-ανάλυσης που εφαρμόζονται στον τομέα της μηχανικής λογισμικού, όπως παρουσιάζονται από τους Cruzes και Dyba [40], dos Santos και Travassos, [41] και Kitchenham et al. [42]. Αρχικά, καταγράψαμε όλα τα ερευνητικά ερωτήματα από τις πρωτογενείς μελέτες, φτιάχνοντας μια λίστα από τις ερωτήσεις (π.χ., «Ποια είναι η βάση, όσον αφορά τους τύπους πρωτογενών μελετών, στις συστηματικές ανασκοπήσεις της Μηχανικής Λογισμικού;», «Ποια είναι τα χαρακτηριστικά αυτών των μελετών όσον αφορά τις μεθόδους που χρησιμοποιούνται για πειραματικό σχεδιασμό και ανάλυση;»). Σημειώνεται ότι τα ερευνητικά ερωτήματα καταγράφηκαν ακριβώς όπως εμφανίζονται στη πρωτογενή μελέτη, χωρίς καμία παρέμβαση

από εμάς. Σε περίπτωση που μια πρωτογενή μελέτη δεν είχε ρητά διατυπωμένα ερευνητικά ερωτήματα, το πεδίο παρέμενε κενό. Επιπλέον, σημειώνεται ότι εφόσον στην εργασία αυτή βασιζόμαστε στη διαδικασία μελέτης συστηματικής χαρτογράφησης, δεν έχει γίνει αξιολόγηση ποιότητας. Για παράδειγμα, η χρήση του DARE θα είχε αποκλείσει μελέτες χωρίς ερευνητικά ερωτήματα. [43]. Στη συνέχεια, εφαρμόσαμε θεματική ανάλυση για να ενοποιηθεί ένας κατάλογος από στόχους του ερευνητικού ενδιαφέροντος. Για την επίτευξη του, εφαρμόσαμε τη μεθοδολογία Open Card Sorting, introduced by Spencer [44]. Αρχικά μελετώντας τα ερευνητικά ερωτήματα, ορίσαμε στόχους υψηλού επιπέδου με τη μορφή υπερκατηγοριών (για παράδειγμα από τον στόχο «πρόβλεψη αριθμού λαθών στο λογισμικό» δημιουργήσαμε τον στόχο υψηλού επιπέδου «λάθη λογισμικού»). Στη συνέχεια, ομαδοποιήσαμε κατηγορίες υψηλού επιπέδου με παρόμοια ή ταυτόσημα νοήματα (π.χ. ενώσαμε τις έννοιες “bug” , “defect” και “fault”). Τέλος, προσδώσαμε ένα όνομα στις ενοποιημένες κατηγορίες (στο παραπάνω παράδειγμα δώσαμε το όνομα “bug/fault/defect”). Η συγκεκριμένη διαδικασία εκπονήθηκε από τη φοιτήτρια σε συνεργασία με την επιβλέπουσα η οποία επικύρωνε τα αποτελέσματα. Κατά τη διαδικασία, καταγράφηκε το ποσοστό διαφωνιών που περιορίστηκε σε 8%. Η συγκεκριμένη μεθοδολογία εφαρμόστηκε και σε άλλα πεδία τα οποία συλλέχθηκαν για τις πρωτογενείς μελέτες: κριτήρια επιλογής και όνομα έργου ανοιχτού λογισμικού.

Ως μέρος της εξαγωγής δεδομένων, για όλες τις συμπεριλαμβανόμενες μελέτες, έχουμε ορίσει ένα σύνολο μεταβλητών που περιγράφουν κάθε πρωτογενή μελέτη [45]. Έτσι, για κάθε μελέτη, έχουμε καταγράψει τις τιμές των παρακάτω μεταβλητών :

[V1] **Τίτλος**: Τίτλος άρθρου
[V2] **Συγγραφέας:** Λίστα συγγραφέων του άρθρου
[V3] **Έτος:** Έτος έκδοσης του άρθρου
[V4] **Τόπος δημοσίευσης**: Όνομα του περιοδικού
[V5] **Στόχοι:** Οι στόχοι του κάθε άρθρου (ενοποιημένα από τα ερευνητικά ερωτήματα)
[V6] **Κριτήρια επιλογής έργων:** Λίστα με τα κριτήρια επιλογής των έργων (ενοποιημένη μεταβλητή)
[V7] **Ονόματα έργων:** Λίστα των έργων ανοιχτού λογισμικού που χρησιμοποίησε το άρθρο (ενοποιημένη μεταβλητή)
[V8] **Διαθέσιμη συλλογή δεδομένων (Ν/Ο):** Αν υπάρχει διαθέσιμη συλλογή δεδομένων ή όχι.

Οι μεταβλητές [V1] – [V4] έχουν χρησιμοποιηθεί για σκοπούς τεκμηρίωσης. Οι υπόλοιπες μεταβλητές χρησιμοποιήθηκαν για την απάντηση των ερευνητικών ερωτημάτων και την περιγραφή του πλαισίου της μελέτης. Μια επισκόπηση της ανάλυσης δεδομένων, που παρουσιάζει την αντιστοίχιση μεταξύ των επιλεγμένων μεταβλητών, των ερευνητικών ερωτημάτων και των μεθόδων ανάλυσης δεδομένων, παρέχεται στον Πίνακα 3.2.

Πίνακας 3.2 Επισκόπηση ανάλυσης δεδομένων

Ερευνητική Ερώτηση	Μεταβλητή	Μέθοδος Ανάλυσης
RQ_1	[V5]	Πίνακες Συχνοτήτων
RQ_2	[V5], [V6]	Πίνακες Συχνοτήτων, Πίνακες Διασταυρώσεων
RQ_3	[V5], [V7]	Πίνακες Συχνοτήτων, Πίνακες Διασταυρώσεων

3.2 Μεθοδολογία Ανάπτυξης Εργαλείου

Μετά την ολοκλήρωση της συλλογής και ανάλυσης δεδομένων, ακολούθησε η ανάπτυξη εργαλείου-ιστοσελίδας. Αρχικά, προσδιορίστηκαν οι απαιτήσεις τις ιστοσελίδας, δηλαδή καταγράφηκαν οι ανάγκες χρήστη και συστήματος και έπειτα καθορίστηκε ο σχεδιασμός και οι τεχνολογίες για την δημιουργία της.

3.2.1 Απαιτήσεις ιστοσελίδας

Οι απαιτήσεις της ιστοσελίδας είναι οι περιγραφές των υπηρεσιών που παρέχονται από την ιστοσελίδα και οι λειτουργικοί περιορισμοί της. Οι απαιτήσεις αυτές αντανακλούν τις ανάγκες των χρηστών της ιστοσελίδας (επισκέπτες και συνδεδεμένοι χρήστες), για ένα σύστημα που βοηθάει στην ανεύρεση πληροφοριών ή στην περίπτωση μας εμπειρικών μελετών [46].
Για την κατασκευή της ιστοσελίδας μας είναι αναγκαίο να γίνει μια ανάλυση των απαιτήσεων. Οι απαιτήσεις αυτές χωρίζονται στις:

- *Απαιτήσεις Χρήστη* [46]. Οι υπηρεσίες που θα παρέχει η ιστοσελίδα είναι:
    - Η δυνατότητα εύρεσης κατάλληλων εμπειρικών μελετών από τους ερευνητές με βάση τον τίτλο, το όνομα έργου, το κριτήριο και το στόχο των μελετών.
    - Η δυνατότητα προσθήκης νέων εμπειρικών μελετών.
- *Απαιτήσεις Συστήματος* [46]. Όπου περιγράφουν με περισσότερη λεπτομέρεια τις υπηρεσίες και τις λειτουργίες που παρέχει η ιστοσελίδα, δηλαδή οι απαιτήσεις συστήματος είναι επεκτάσεις των απαιτήσεων χρήστη. Στη περίπτωση του εργαλείου-ιστοσελίδας και με βάση τις απαιτήσεις χρήστη που προαναφέρθηκαν οι απαιτήσεις συστήματος είναι οι εξής:
    - Η δημιουργία μπάρας αναζήτησης για την εύρεση εμπειρικών μελετών.
    - Η δημιουργία υποβολής φόρμας για την προσθήκη νέων μελετών.
    - Η ύπαρξη μίας βάσης δεδομένων εμπειρικών μελετών, που περιέχει όλες τις πληροφορίες για τις εκάστοτε εμπειρικές μελέτες.

Η κατασκευή της ιστοσελίδας έχει ως σκοπό να βοηθήσει τους ερευνητές να αναζητήσουν εμπειρικές μελέτες βάση των χαρακτηριστικών τους, και να επιλέξουν τις κατάλληλες συλλογές έργων ανοιχτού λογισμικού για την υλοποίηση των δικών τους μελετών.

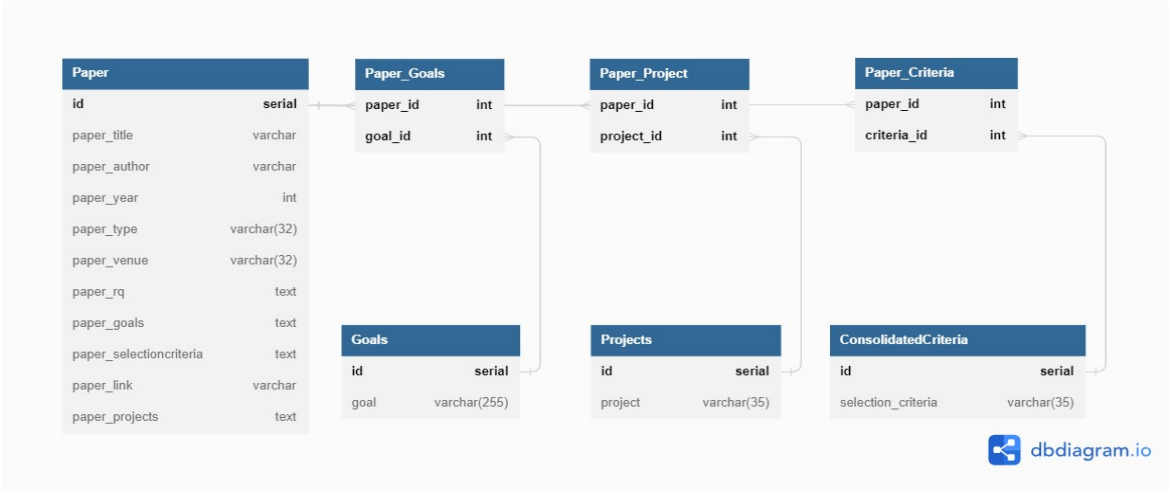
3.2.2 Σχεδιαστικές αποφάσεις

Οι σχεδιαστικές αποφάσεις έχουν άμεση σχέση με τις απαιτήσεις της ιστοσελίδας. Σε ότι αφορά το οπτικό μέρος (front-end) οι αποφάσεις που πάρθηκαν ήταν:

- Εμφάνιση των εμπειρικών μελετών της βάσης.
- Ύπαρξη μπάρας αναζήτησης.
- Τοποθέτηση κουμπιού για σύνθετη αναζήτηση.
- Τοποθέτηση κουμπιού για την εμφάνιση πληροφοριών.
- Ύπαρξη καρτελών για την παροχή στατιστικών στοιχείων.
- Δημιουργία φόρμας σύνδεσης.
- Τοποθέτηση κουμπιού για προσθήκη νέων μελετών.

Για το κρυμμένο από τον χρήστη μέρος (back-end) πάρθηκαν οι εξής αποφάσεις:

- Δημιουργία σχήματος βάσης δεδομένων – βλ. Σχήμα 2.
- Δημιουργία της βάσης σε PostgreSQL.
- Επιλογή πλατφόρμας Docker για την ανάπτυξη της ιστοσελίδας.
- Επικοινωνία του client με τη βάση με rest API.



Σχήμα 3.2 Παρουσίαση σχήματος βάσης δεδομένων

3.2.3 Περιγραφή τεχνολογιών

Ο σχεδιασμός του front-end διαδραματίζει κρίσιμο ρόλο στη δημιουργία ελκυστικών και φιλικών προς τον χρήστη εμπειριών. Για να επιτύχουμε μια αισθητικά και οπτικά ελκυστική διεπαφή χρήστη (user interface), χρησιμοποιήσαμε έναν συνδυασμό ισχυρών εργαλείων και πλαισίων. Για να παρέχουμε μία βέλτιστη εμπειρία στο χρήστη ενσωματώσαμε τις παρακάτω τεχνολογίες.

jQuery[47], [48]: Το jQuery είναι μια ισχυρή και ευρέως διαδεδομένη βιβλιοθήκη JavaScript που έχει φέρει επανάσταση στον τρόπο που οι προγραμματιστές αλληλοεπιδρούν με τις ιστοσελίδες. Παρέχει μια απλοποιημένη και αποτελεσματική προσέγγιση για το χειρισμό εγγράφων HTML, το χειρισμό συμβάντων, τη δημιουργία κινούμενων εικόνων και την υποβολή αιτημάτων AJAX. Λόγω της σύνταξης και της ευρείας επιλογής λειτουργιών, το jQuery έχει γίνει βασικό εργαλείο για σχεδιαστές ιστοσελίδων και προγραμματιστές που επιδιώκουν να βελτιώσουν τη διαδραστικότητα και τη λειτουργικότητα των ιστοσελίδων.

Ένα από τα βασικά πλεονεκτήματα του jQuery είναι η ικανότητά του να απλοποιεί τον χειρισμό DOM. Παραδοσιακά, ο χειρισμός του μοντέλου εγγράφου αντικειμένου (DOM) με χρήση απλού κώδικα JavaScript μπορεί να είναι επίπονος και χρονοβόρος. Το jQuery απλοποιεί αυτή τη διαδικασία παρέχοντας ένα σύνολο έξυπνων λειτουργιών και μεθόδων που αφαιρούν την πολυπλοκότητα της διέλευσης και του χειρισμού DOM. Η επιλογή στοιχείων, η τροποποίηση ιδιοτήτων CSS, η προσθήκη ή η αφαίρεση κλάσεων και ο χειρισμός περιεχομένου γίνονται με ευκολία λόγω της βελτιωμένης σύνταξης του jQuery.

Το jQuery υπερέχει επίσης στη διαχείριση συμβάντων. Είτε ανταποκρίνεται σε ένα κλικ κουμπιού, ανίχνευση εισόδου χρήστη σε μια φόρμα είτε εφαρμόζει ομαλή κύλιση. Επίσης, προσφέρει μια σειρά από μεθόδους συμβάντων που απλοποιούν τη σύνδεση και την ανάθεση συμβάντων. Αυτό επιτρέπει στους προγραμματιστές να δημιουργούν διαδραστικές και ανταποκρίσιμες ιστοσελίδες χωρίς την ανάγκη πολύπλοκων γραμμών κώδικα ή περίπλοκης διαχείρισης συμβάντων.

React Fragment [49]: Το React Fragment επιτρέπει στους προγραμματιστές να ομαδοποιούν πολλά στοιχεία μαζί χωρίς την ανάγκη πρόσθετου στοιχείου αναδίπλωσης. Αυτή η δυνατότητα αντιμετωπίζει το ζήτημα της προσθήκης περιττών div ή άλλων στοιχείων αποκλειστικά για την επιστροφή ενός μόνο γονικού στοιχείου σε ένα στοιχείο React (React Component).

Ένα από τα κύρια πλεονεκτήματα του React Fragment είναι η βελτιωμένη οργάνωση κώδικα και η δυνατότητα συντήρησης. Με τα Fragments, οι προγραμματιστές μπορούν να αποφύγουν να γεμίσουν

το DOM με περιττά στοιχεία, διατηρώντας παράλληλα μια καθαρή και σημασιολογική δομή HTML. Εξαλείφοντας την ανάγκη για επιπλέον στοιχεία αναδίπλωσης, ο κώδικας που προκύπτει γίνεται πιο συνοπτικός και ευανάγνωστος, μειώνοντας την περιττή πολυπλοκότητα. Ένα άλλο πλεονέκτημα του React Fragment είναι η βελτιωμένη απόδοση. Κατά την απόδοση ενός στοιχείου, το React Fragment δεν δημιουργεί πρόσθετο κόμβο DOM. Αντίθετα, λειτουργεί ως ένα ελαφρύ περιτύλιγμα που ομαδοποιεί τα στοιχεία. Αυτό σημαίνει ότι η χρήση Fragments δεν εισάγει επιπλέον επιβάρυνση, με αποτέλεσμα την βελτιωμένη απόδοση για την εφαρμογή.

Τέλος, το React Fragment συμβάλλει επίσης στην επαναχρησιμοποίηση κώδικα. Με την ενσωμάτωση πολλαπλών στοιχείων σε ένα Fragment, οι προγραμματιστές μπορούν να δημιουργήσουν πιο λεπτομερή στοιχεία που εύκολα επαναχρησιμοποιούνται σε διαφορετικά μέρη της εφαρμογής.

Babel [50]: Το Babel είναι ένας ευρέως χρησιμοποιούμενος μεταγλωττιστής JavaScript που επιτρέπει στους προγραμματιστές να γράφουν σύγχρονο κώδικα JavaScript διασφαλίζοντας ταυτόχρονα τη συμβατότητα με παλαιότερα περιβάλλοντα και προγράμματα περιήγησης. Καθώς η γλώσσα JavaScript εξελίσσεται με νέες δυνατότητες, η Babel λειτουργεί ως γέφυρα, επιτρέποντας στους προγραμματιστές να αξιοποιήσουν αυτές τις σύγχρονες γλωσσικές δυνατότητες χωρίς να θυσιάζουν τη συμβατότητα.

Ένα από τα κύρια πλεονεκτήματα του Babel είναι η ικανότητά του να μετασχηματίζει και να μεταφέρει τον σύγχρονο κώδικα JavaScript σε μια παλαιότερη σύνταξη που μπορεί να γίνει κατανοητή από παλαιότερα προγράμματα περιήγησης. Αυτό είναι ιδιαίτερα σημαντικό σε έργα που πρέπει να υποστηρίζουν ένα ευρύ φάσμα εκδόσεων προγράμματος περιήγησης.

Επιπλέον, η Babel ενσωματώνεται με δημοφιλή εργαλεία και πλαίσια ανάπτυξης. Είτε πρόκειται για ένα έργο React, Angular ή Vue.js, το Babel μπορεί εύκολα να ενσωματωθεί στη διαδικασία κατασκευής, διασφαλίζοντας ότι ο κώδικάς μετατρέπεται και είναι έτοιμος για παραγωγή.

Semantic UI [51]: Το Semantic UI είναι ένα ολοκληρωμένο πλαίσιο ανάπτυξης front-end που εστιάζει στη δημιουργία οπτικά ελκυστικών και προσβάσιμων διεπαφών χρήστη. Με την έμφαση στις σημασιολογικές συνθήκες και τα αισθητικά μοτίβα σχεδίασης, το Semantic UI απλοποιεί τη διαδικασία δημιουργίας σύγχρονων εφαρμογών ιστού.

Ένα από τα βασικά πλεονεκτήματα του Semantic UI είναι η εκτεταμένη βιβλιοθήκη με προσχεδιασμένα στοιχεία. Αυτά τα στοιχεία καλύπτουν ένα ευρύ φάσμα στοιχείων διεπαφής χρήστη, όπως κουμπιά, φόρμες, μενού κ.α. Αξιοποιώντας αυτά τα έτοιμα προς χρήση στοιχεία, οι προγραμματιστές μπορούν γρήγορα να πρωτοτυπήσουν και να δημιουργήσουν διεπαφές χρήστη χωρίς την ανάγκη εκτεταμένης προσαρμοσμένης κωδικοποίησης. Αυτό μειώνει σημαντικά τον χρόνο και την προσπάθεια ανάπτυξης κώδικα, επιτρέποντας στους προγραμματιστές να επικεντρωθούν στις βασικές λειτουργίες.

Ένα άλλο πλεονέκτημα του Semantic UI είναι οι δυνατότητες λειτουργικότητας στο σχεδιασμό του. Το πλαίσιο παρέχει μια σειρά λειτουργικού πλέγματος και ρευστών στοιχείων που προσαρμόζονται αυτόματα σε διαφορετικά μεγέθη οθόνης και συσκευής. Αυτή η ανταπόκριση εξασφαλίζει μια ευχάριστη εμπειρία χρήστη σε πλατφόρμες υπολογιστών, tablet και φορητών υπολογιστών.

Επιπλέον, το Semantic UI προσφέρει μια ποικιλία επιλογών θεμάτων, επιτρέποντας στους προγραμματιστές να προσαρμόσουν την εμφάνιση και την αίσθηση των εφαρμογών τους ώστε να ταιριάζουν με τις συγκεκριμένες απαιτήσεις σχεδίασης. Το πλαίσιο παρέχει μια ισχυρή μηχανή δημιουργίας θεμάτων και μια εκτενή συλλογή προκαθορισμένων θεμάτων, καθιστώντας εύκολη τη δημιουργία οπτικά εντυπωσιακών διεπαφών που συμβαδίζουν με την επιθυμητή αισθητική.

SweetAlert [52]: Το SweetAlert Popup είναι μια βιβλιοθήκη JavaScript που βελτιώνει την εμπειρία του χρήστη αντικαθιστώντας τα τυπικά παράθυρα ειδοποίησης και επιβεβαίωσης του προγράμματος περιήγησης με οπτικά ελκυστικά και προσαρμόσιμα αναδυόμενα παράθυρα.

Η βιβλιοθήκη αυτή προσφέρει ένα ευρύ φάσμα επιλογών στυλ, κινούμενων εικόνων και παραμέτρων προσαρμογής, επιτρέποντας στους προγραμματιστές να σχεδιάζουν αναδυόμενα παράθυρα που να

συντονίζονται με τη συνολική αισθητική της διαδικτυακής εφαρμογής ή της ιστοσελίδας. Επίσης, οι προγραμματιστές μπορούν να προσαρμόσουν το περιεχόμενο του παραθύρου διαλόγου, συμπεριλαμβάνοντας κείμενο, εικονίδια και κουμπιά, ώστε να ταιριάζει στις επιθυμίες τους.
Το SweetAlert Popup προσφέρει ένα απλό και αισθητικό API, καθιστώντας εύκολη την ενσωμάτωση σε υπάρχοντα έργα. Η βιβλιοθήκη υποστηρίζει διάφορες γλώσσες προγραμματισμού, συμπεριλαμβανομένων των JavaScript, jQuery και React, καθιστώντας την προσβάσιμη σε προγραμματιστές που χρησιμοποιούν διαφορετικά πλαίσια ή τεχνολογίες.

Κεφάλαιο 4ο: Αποτελέσματα

4.1 Αποτελέσματα Βιβλιογραφικής Ανασκόπησης

Σε αυτή την ενότητα παρουσιάζουμε τα αποτελέσματα της μελέτης οργανωμένα ανά ερευνητική ερώτηση. Στην Ενότητα 4.1.1, συζητάμε τα ευρήματά μας σχετικά με τους στόχους μελετών MSR που κάνουν χρήση συλλογών έργων ανοιχτού λογισμικού **(RQ_1)**. Στην Ενότητα 4.1.2, παρουσιάζουμε τα πιο χρησιμοποιημένα κριτήρια επιλογής των έργων ανοιχτού λογισμικού ανά στόχο μελέτης **(RQ_2)**. Τέλος, στην Ενότητα 4.1.3, παρουσιάζουμε αποτελέσματα που σχετίζονται με τα πιο μελετημένα έργα από τους ερευνητές ανά στόχο μελέτης **(RQ_3)**.

4.1.1 Στόχοι μελετών MSR (RQ_1)

Στον Πίνακα 4.1 παρουσιάζουμε τους πιο συχνά εμφανιζόμενους στόχους στις πρωτογενείς μελέτες που εξάγουν δεδομένα από αποθετήρια ανοιχτού λογισμικού. Παρατηρώντας τον Πίνακα 4.1, ο συχνότερος στόχος των MSR μελετών είναι η μελέτη «σφάλματα/ελαττώματα/λάθη» με συχνότητα 84 (12,63%) και στη συνέχεια ακολουθεί ο «έλεγχος» με συχνότητα 70 (10,53%). Στόχοι όπως «ανωμαλίες λογισμικού», «αξιοπιστία», «συμπεριφορά χρήστη» αποτελούν τους λιγότερο πιθανούς στόχους με συχνότητα 1 (0,15%).
Επικεντρώνοντας στη κατηγορία «σφάλματα/ελαττώματα/λάθη» μπορούμε να αναγνωρίσουμε μελέτες οι οποίες στοχεύουν:

- στη μελλοντική πρόβλεψη του πλήθους σφαλμάτων/ελαττωμάτων/λαθών μέσω άλλων παραγόντων. Η συγκεκριμένη πρόβλεψη είναι πολύ χρήσιμη για τις επιχειρήσεις καθώς μπορούν να εκτιμήσουν το κόστος της μελλοντικής συντήρησης.
- Στην αναγνώριση των τμημάτων κώδικα (hotspots) τα οποία είναι πιθανότερο να συσσωρεύσουν στο μέλλον σφάλματα/ελαττώματα/λάθη. Η συγκεκριμένη πληροφορία μπορεί να χρησιμοποιηθεί προληπτικά για τη βελτίωση της σχεδίασης και της κατανοησιμότητας των συγκεκριμένων τμημάτων ώστε τα λάθη να διορθωθούν ευκολότερα στο μέλλον.

Στο σημείο αυτό πρέπει να τονίσουμε ότι η αποσφαλμάτωση (debugging) και η αυτόματη διόρθωση προγραμμάτων (program repair) αφορά διαφορετικό στόχο με ποσοστό εμφάνισης 3,81%. Παραμένοντας στη συγκεκριμένη φάση ανάπτυξης μπορούμε να παρατηρήσουμε ότι πολλές μελέτες στοχεύουν στον έλεγχο (testing) λογισμικού. Οι μελέτες αυτές χρησιμοποιούν περιπτώσεις ελέγχου (test cases) ή ελέγχους μονάδα υλοποιημένες σε κώδικα (unit tests). Αναμενόμενα στη συνέχεια παρατηρούμε μελέτες που στοχεύουν στην ασφάλεια (security) και στην ανίχνευση τρωτοτήτων (vulnerability). Η άνοδος αυτών των μελετών θεωρείται αναμενόμενο με τη μετάβαση των συστημάτων σε περιβάλλοντα διαδικτύου, νέφους ή υπηρεσιών.

Στη συνέχεια, οι πιο δημοφιλείς στόχοι αφορούν στη ποιότητα λογισμικού, και πιο συγκεκριμένα στο πως αυτή μπορεί να παρακολουθηθεί στην εξέλιξη του λογισμικού και να βελτιωθεί. Ο συγκεκριμένος κλάδος της τεχνολογίας λογισμικού είναι ιδιαίτερα σημαντικός για τις βιομηχανίες καθώς το κόστος συντήρησης αγγίζει το 75% του συνολικού κόστους ανάπτυξης λογισμικού [53]. Στη κατηγορία αυτή ανήκουν μελέτες που στοχεύουν στη διαχείριση εκδόσεων (version control), στις αναδομήσεις (refactoring), στην αναγνώριση και απαλοιφή κακών οσμών (bad smells), στην ανάλυση εξαρτήσεων (dependency analysis), και στη διαχείριση χαρακτηριστικών ποιότητας εκτέλεσης (run-time quality attributes) όπως απόδοση / διαχείριση πόρων / χρόνος απόκρισης (performance / resource management / time behaviour).

Τέλος, σημαντικό ποσοστό της σχετικής αρθρογραφίας ασχολείται με την αυτόματη παραγωγή κώδικα εφαρμογής και ελέγχου (code/test generation). Η αυτόματη παραγωγή κώδικα μπορεί να συνεισφέρει σημαντικά στη μείωση του χρόνου παραγωγής λογισμικού ο οποίος είναι ιδιαίτερα πιεστικός στη σύγχρονη βιομηχανία ανάπτυξης λογισμικού με αρνητικά αποτελέσματα όπως το τεχνικό χρέος (technical debt). Το τεχνικό χρέος αναφέρεται ως διακριτός στόχος του 1,96% των μελετών. Ολοκληρώνοντας άξια αναφοράς κρίνεται το γεγονός ότι 2,11% των μελετών στοχεύουν στον ανθρώπινο παράγοντα (human factors) και τις δυσκολίες διαχείρισης μιας ομάδας ανάπτυξής που αποτελείται από διαφορετικούς ανθρώπους, με διαφορετικά χαρακτηριστικά και διαφορετικές προσδοκίες από την εργασία και την εξέλιξη τους.

Πίνακας 4.1 Συχνότητα τελικών στόχων

Goal	Freq.
Bugs / Defects / Faults	84
Testing	70
Security	37
Localization	28
Vulnerabilities	26
Program Repair	20
Version Control	18
Refactoring	18
Features / Requirements / Business Models	18
Bad Smells	17
Code/Test Generation	15
Dependency Analysis	15
Performance / Resource Management / Time Behaviour	15
Human Factors	14
Code Review	14
Application Programming Interfaces (APIs)	14
Technical Debt	13
Software Change	12
Documentation	12
Quality Metrics	11
Architecture	11
Static Analysis	11
Data / Information	10
Management	8
Cost Analysis	8
Code Clones	8
Execution Traces	8
Software Libraries	7
Traceability	7
Logs	7
Debugging / Bug Fixing	6
Comments	6
Software Product Lines (SPL) / Reconfigurable Systems	6
Software Design	5
Maintenance	5
GUI	4
Search Based Software Engineering (SBSE)	4
Software Quality Assurance Processes	4
Change Impact Analysis (CIA)	5

Goal	Freq.
Software Patterns	4
Component Based Software Engineering (CBSE)	4
Software Crashes	4
Effort Estimation	4
Complexity	4
Network	4
Software Build	3
User Reviews	3
Privacy	3
Modularity	3
Code Transformation / Compilers	3
CI/CD	3
Team Management	3
Service Oriented Architectures (SOA)	3
User Experience	3
Aesthetics	2
Slicing	2
Agile Methodologies	2
Reuse	2
Dynamic Analysis	2
Testability	2
Program Comprehension	2
Exception Handling	2
Parallelization	2
Cloud-Based Software	2
Business Parameters	2
Domain Specific Languages (DSL)	1
Software Analytics	1
Software Anomalies	1
Deployment	1
Code Annotations	1
Application Domain Analysis	1
Reliability	1
User Behaviour	1

4.1.2 Κοινά κριτήρια επιλογής έργων στις εμπειρικές μελέτες (RQ_2)

Όσο αναφορά τα κριτήρια μελέτης επιλογής έργων, ο Πίνακας 4.2 παρουσιάζει τα κριτήρια που συλλέξαμε από την έρευνα μας και ο αριθμός συχνότητας αυτών. Τα αποτελέσματα που προκύπτουν είναι ότι οι ερευνητές θέτουν συνήθως σαν κριτήριο επιλογής έργων τη χρήση της «Java» ως γλώσσα προγραμματισμού 78 φορές (10,17%) και έπειτα ακολουθεί η «δημοτικότητα» με συχνότητα 67 (8,74%). Η επιλογή της Java αποτελεί μία αναμενόμενη απόφαση στη σχεδίαση της έρευνας λόγω της πληθώρας εργαλείων που είναι διαθέσιμα για την ανάλυση κώδικα Java. Η απόφαση σχετικά με τη δημοτικότητα έχει τις ρίζες της στην διάθεση των ερευνητών να επιλέξουν έργα τα οποία θα είναι γνωστά στον αναγνώστη της δικής τους έρευνας καθώς και θα παρέχει κάποια εχέγγυα για τη διασφάλιση της ενεργής ανάπτυξης των έργων και της αποδοχής τους από την κοινότητα. Αντίθετα, «ανενεργά έργα» και «τυχαία επιλογή» έργων (4 φορές—0,52%), είναι λιγότερο πιθανό να αποτελούν κριτήριο επιλογής έργων. Στο σημείο αυτό, πρέπει να διευκρινίσουμε ότι ο όρος «τυχαία επιλογή» είναι διακριτός από την έννοια της διαφορετικότητας (π.χ. διαφορετικό μέγεθος, ιστορία) καθώς ο δεύτερος όρος υποδεικνύει συστηματική (όχι τυχαία) επιλογή έργων.

Επικεντρώνοντας στις ***γλώσσες προγραμματισμού***, η Java είναι η γλώσσα που αναφέρεται πιο συχνά με συχνότητα 78 (10,17%), ακολουθούμενη από γλώσσες της οικογένειας της C (C / C++ / C#) με συχνότητα 21 (2,74%), τη Python με συχνότητα 7 και τη JS/Javascript με συχνότητα 5. Συνολικά, οι γλώσσες προγραμματισμού συγκεντρώνουν ποσοστό 15,91%.

Τα κριτήρια που σχετίζονται με τα ***χαρακτηριστικά διαχείρισης του έργου*** με συνολική συχνότητα 14,73%. Τα κριτήρια αυτής της κατηγορίας είναι σημαντικά σε μελέτες που θέλουν να εξάγουν αυτοματοποιημένα την πληροφορία με χρήση κάποιου εργαλείου που ήδη χρησιμοποιείται από την ομάδα ανάπτυξης του έργου (π.χ. Version control, Issue Tracker, Bug Tracker). Στη κατηγορία αυτή βλέπουμε να ξεχωρίζουν τα κριτήρια: «Πληροφορίες ελέγχου έκδοσης» (Version control Information) με συχνότητα 43 (5,61%) και «Πληροφορίες για την κοινότητα της ανάπτυξης λογισμικού» (Development Community Information) με συχνότητα 22 (2,87%). Το κριτήριο «Πληροφορίες παρακολούθησης προβλημάτων» (Issue Tracker Information) εμφανίζεται 13 φορές (1,69%).

Τα κριτήρια που σχετίζονται με το ***μέγεθος και τη δραστηριότητα του έργου***. Από τη μία πλευρά το μέγεθος αποτελεί αναπόσπαστο κριτήριο των εμπειρικών μελετών για την επιλογή έργων ανοικτού κώδικα, καθώς ο στόχος του κριτηρίου είναι η διασφάλιση ότι πολύ μικρά έργα τα οποία δεν θα ήταν συγκρίσιμα με βιομηχανικά έργα θα αποκλειστούν. Από την άλλη μεριά, η δραστηριότητα του έργου είναι ιδιαίτερα σημαντική για άρθρα που στοχεύουν στη μελέτη της εξέλιξης του λογισμικού με αποτέλεσμα να χρειάζονται ικανό αριθμό εκδόσεων του λογισμικού για την επίτευξη στατιστικής ανάλυσης. Συγκεκριμένα, τα κριτήρια σχετικά με το «μέγεθος σε άλλες μετρικές» (size information In Other Metrics) εμφανίζονται με τη μεγαλύτερη συχνότητα 41 (5,35%) και ακολουθούν, ενώ πιο συγκεκριμένα κριτήρια όπως το «μέγεθος σε γραμμές κώδικα» (Size Information In LoC) και το «μέγεθος σε κλάσεις/πακέτα/συστατικά» (Size Information In Modules) εμφανίζονται σε 29 και 18 μελέτες αντίστοιχα. Οι πληροφορίες σχετικά με την «ηλικία και ωριμότητα του έργου» (Project Age / Maturity) εμφανίζονται σε 39 μελέτες (5,08%), και το κριτήριο «ένταση δραστηριότητας ανάπτυξης» (Intensity Of Development Activity) ακολουθεί με 28 μελέτες. Εν αντιθέσει, πληροφορίες για τη «μη δραστηριότητα έργου» (Inactive Projects) εμφανίζουν την λιγότερη συχνότητα με 4 φορές (0,52%), καθώς είναι χρήσιμο σε μελέτες που στοχεύουν στη διερεύνηση παραγόντων για το τερματισμό των έργων. Η ομάδα αυτή συγκεντρώνει τη μεγαλύτερη συνολική συχνότητα με ποσοστό 23,99%.

Στην συνέχεια, παρατηρούμε ότι το κριτήριο επιλογής με ***βάση προηγούμενη βιβλιογραφία*** σημειώνει συνολική συχνότητα 15,78%, με τα επιμέρους κριτήρια «Δημοτικότητα» (Widely Used / Popular) και «Χρήση σε προηγούμενες μελέτες»(Used In Previous Studies) να έχουν συχνότητα 67 (8,74%) και 54 (7,04%) αντίστοιχα. Αυτό το κριτήριο στοχεύει στην επανάληψη (replication) εμπειρικών μελετών αλλά και στη δυνατότητα σύγκρισης αποτελεσμάτων.

Επιπρόσθετα, στην ομάδα της ***ποικιλομορφίας*** εντοπίζονται τα κριτήρια «Διάφοροι Τομείς» (Various Domains) με συχνότητα 39 (5,08%), «διαφορετικότητα σε μέγεθος» (Diversity In Size) με συχνότητα 25, ακολουθούμενα από το κριτήριο «Τυχαία επιλογή» (Random Selection) με συχνότητα 4 (0,52%). Η ομάδα της ποικιλομορφίας σε αντίθεση με αυτή που έχει ως βάση τη προηγούμενη βιβλιογραφία σημειώνει την λιγότερη συνολική συχνότητα με ποσοστό 5,60%. Παρ'όλα αυτά κρίνεται ιδιαίτερα σημαντική για τη γενίκευση (generalization/external validity) των αποτελεσμάτων καθώς μειώνει την επίδραση των συγκεχυμένων παραγόντων (confounding factors).

Σχετικά με τις ***τεχνολογίες***, παρατηρείται ότι το πιο συχνό κριτήριο είναι η «Χρήση των καλύτερων πρακτικών» (Use Of Best Practices) με συχνότητα 54 (7,04%) και ακολουθεί ο «Συγκεκριμένος τομέας εφαρμογής ή τεχνολογίας» (Specific Application Domain Or Technology) με συχνότητα 45 (5,87%). Το κριτήριο σχετικά με την «λειτουργικότητα, τα χαρακτηριστικά και τις απαιτήσεις» συγκεντρώνει

την λιγότερη συχνότητα των τεχνολογιών με μόλις 11 φορές εμφάνισης (1,43%). Η συνολική συχνότητα εμφάνισης των τεχνολογιών ανέρχεται στο 16,04%. Η συγκεκριμένη ομάδα κριτηρίων λαμβάνεται υπόψιν από τους ερευνητές στις περιπτώσεις που το άρθρο στοχεύει σε κάποια συγκεκριμένη τεχνολογία (π.χ. πρότυπα σχεδίασης, κακές οσμές), ή σε κάποιο συγκεκριμένο πεδίο εφαρμογής (π.χ. διαδίκτυο, υπηρεσία, νέφος). Τέλος, τα κριτήρια που αφορούν τη ***δυνατότητα ελέγχου***, όπως τα «Δοκιμασμένα συστήματα» (Tested Systems) και τα «Συστήματα δημιουργίας λογισμικού» (Buildable Systems), έχουν συχνότητα 7,95%. Τα συγκεκριμένα κριτήρια είναι εφαρμόσιμα στις περιπτώσεις που: (α) η μελέτη σχετίζεται με τον έλεγχο λογισμικού, ή (β) η μελέτη χρησιμοποιεί εργαλεία για τη συλλογή δεδομένων που απαιτούν την εκτέλεση του κώδικα.

Πίνακας 4.2 Συχνότητα επιλογής κριτηρίων

Selected Criteria	Freq.
Java Programming Language	78
Widely Used / Popular	67
Use Of Best Practices	56
Used In Previous Studies	54
Specific Application Domain Or Technology	45
Tested Systems	43
Version Control Information	43
Size Information (In Other Metric)	41
Project Age / Maturity	39
Various Domains	39
Size Information (In Loc)	29
Intensity Of Development Activity	28
Diversity In Size	25
Development Community Information	22
C-Family Programming Language	21
Bug Tracker Information	20
Buildable Systems	18
Size Information (In Modules)	18
Hosted In Git/Github	15
Issue Tracker Information	13
Other Programming Language	11
Service-Oriened Software	11
Feature / Functionality / Requirements Criteria	11
Python Programming Language	7
Js/Javascript Programming Language	5
Inactive Projects	4
Random Selection	4

Καθοδήγηση ερευνητών στην επιλογή κριτηρίων. Με βάση τα παραπάνω, στο Πίνακα 4.3 παρουσιάζονται οι διασταυρώσεις συχνοτήτων μεταξύ στόχων και κριτηρίων. Ο πίνακας αυτός είναι ιδιαίτερα σημαντικός κατά τη φάση σχεδίασης εμπειρικών μελετών όπου ο κάθε ερευνητής με βάση τον στόχο του μπορεί να εντοπίσει τα κριτήρια που θα πρέπει να τον απασχολήσουν. Ο Πίνακας είναι ταξινομημένος ανά στόχο με φθίνουσα συχνότητα.

Πίνακας 4.3 Διασταύρωση Συχνοτήτων τελικού στόχου με κριτήρια επιλογής

Final Goal	Criteria																									
	Bug Tracker Information	Buildable Systems	C-Family Programming	Development	Diversity In Size	Feature / Functionality / Requirements Criteria	Hosted In Git/Github	Inactive Projects	Intensity Of Development Activity	Issue Tracker	Java Programming	Js/Javascript	Other Programming	Project Age / Maturity	Python Programming	Random Selection	Service-Oriened	Size (In Loc)	Size (In Modules)	Size (In Other Metric)	Specific Application	Tested Systems	Use Of Best Practices	Used In Previous Studies	Various Domains	Version Control	Widely Used / Popular
Bugs / Defects / Faults	14	1	6	4	6	1	3		7	6	17	1		10	2			6	6	12	6	15	4	16	8	9	12
Testing	1	5	7	4	7	2	5		1		9		2	7		1	1	8	2	4	12	21	5	13	7	2	9
Security	3	2	4		2			1	3		6		1	2	1			1		4	6	1	9	5	1	4	5
Localization	5	1	2	1	3	1	1		3	2	5			1				2	1	2	2	7	1	5	4		4
Vulnerabilities	2	2	3						1		5		1	1	1						3	1	8	3	1	2	4
Program Repair	2	3			2				5		4		1	2			1	1	1	1	4	2	3	2	1		2
Version Control	1		1	2	2	1	2	1	3	2	7			2	1					2	3	1	3	3		7	2
Refactoring				1	1				2		6	2		2				1	1	2			5	1		4	5
Features / Req. / Business Models					3	2				1	3			1		1		3		3	1	4	7	3	4	1	2

Final Goal	Criteria																									
	Bug Tracker Information	Buildable Systems	C-Family Programming	Development	Diversity In Size	Feature / Functionality / Requirements Criteria	Hosted In Git/Github	Inactive Projects	Intensity Of Development Activity	Issue Tracker	Java Programming	Js/Javascript	Other Programming	Project Age / Maturity	Python Programming	Random Selection	Service-Oriened	Size (In Loc)	Size (In Modules)	Size (In Other Metric)	Specific Application	Tested Systems	Use Of Best Practices	Used In Previous Studies	Various Domains	Version Control	Widely Used / Popular
Bad Smells		2	2	1	2		2		2		5			2				3	1		1		3	3	2		3
Code/Test Generation	1	1					2				4		1	1			1	1	2		2	2			2	1	3
Dependency Analysis	1	1			1	1		1	1		6		1		1		1	1		2	3	2	5		1	3	4
Performance / Resource Management / Time Behaviour		2	1			1	1				1	1		1		1	1	1	1	1	4	3	3	1	2	1	3
Human Factors	2			4	1		1		1	1				3				1	1	1	1		3		2	3	3
Code Review				1				1	3		1			1					1	2	1		5	2			2
Application Programming Interfaces (APIs)		1		1		1			2	1	5		1	1			5	3			1	1	4	2	1	1	4
Technical Debt		1	1	2				1	1	3	3			3	1			2	2	1	3		2		1	4	1
Software Change		1			3	1			2		3		2	2						2	1	1	2		3	2	3

Final Goal	Criteria																								
	Bug Tracker Information	Buildable Systems	C-Family Programming	Development	Diversity In Size	Feature / Functionality / Requirements Criteria	Hosted In Git/Github	Inactive Projects	Intensity Of Development Activity	Issue Tracker	Java Programming	Js/Javascript	Other Programming	Project Age / Maturity	Python Programming	Random Selection	Service-Oriened	Size (In Loc)	Size (In Modules)	Size (In Other Metric)	Specific Application	Tested Systems	Use Of Best Practices	Used In Previous Studies	Various Domains	Version Control	Widely Used / Popular
Documentation	2			2					1	2	3		1	2				1	1	2			1	1	2		1
Quality Metrics	1	1		1	2	1	2		2	1	5			2					1		1	2		3	1	1	2
Architecture		1			1						3						1		1	1	3		1	1	1		1
Static Analysis		1	1			1	1		1	1	3			2						1	1	1	3	1		2	
Data / Information	1								1	2	3			1	1			2					1	4		2	2
Management	1		1				1	1	2	1	1		1	3					1	1	1		2		1	4	3
Cost Analysis			1		2				1		2			1				1			1	1		2	4		1
Code Clones				1	1						3		1	1						1		1	4	1	2	1	2
Execution Traces			1						1						1	1				1	1			1	1		2
Softw. Libraries				1		1		1	2					2		1				2	3		1			2	1
Traceability				1	1						2							1		1	2		2		1	1	1

Final Goal	Criteria																										
	Bug Tracker Information	Buildable Systems	C-Family Programming	Development	Diversity In Size	Feature / Functionality / Requirements Criteria	Hosted In Git/Github	Inactive Projects	Intensity Of Development Activity	Issue Tracker	Java Programming	Js/Javascript	Other Programming	Project Age / Maturity	Python Programming	Random Selection	Service-Oriened	Size (In Loc)	Size (In Modules)	Size (In Other Metric)	Specific Application	Tested Systems	Use Of Best Practices	Used In Previous Studies	Various Domains	Version Control	Widely Used / Popular
Logs	1			1							2			2				1		4			3	3	2		1
Debugging / Bug Fixing	1		1						1		1		1	1												2	
Comments	1			2	1						2		1	1					1				1		1	1	1
SPL / Reconfigurable Systems					1	1								1						1				1	1		1
Software Design			1		2				2		3														3		
Maintenance					1						2			1				1			1		1	2	2		
GUI														1			1				2	1	1	1			
Search Based Software Engineering											2						1										2

Final Goal	Criteria																										
	Bug Tracker Information	Buildable Systems	C-Family Programming	Development	Diversity In Size	Feature / Functionality / Requirements Criteria	Hosted In Git/Github	Inactive Projects	Intensity Of Development Activity	Issue Tracker	Java Programming	Js/Javascript	Other Programming	Project Age / Maturity	Python Programming	Random Selection	Service-Oriened	Size (In Loc)	Size (In Modules)	Size (In Other Metric)	Specific Application	Tested Systems	Use Of Best Practices	Used In Previous Studies	Various Domains	Version Control	Widely Used / Popular
Software Quality Assurance Processes		2							1				1	2					1	1		1	1	1	1	2	
Change Impact Analysis (Cia)					1				1		2		1							1		1	2		3	1	1
Software Patterns		1			2						3									1			1	1	1		
Component Based Software Engineering																			1	1	1		1	1			1
Software Crashes	1	1														1		1		1		2					
Effort Estimation						1			1		1									1	1		1	1			
Complexity													1	1						1			1				1
Network			1		1						1			1			1			1	3				1		2
Software Build		2									1	1		1							1	1		1			

Final Goal	Criteria																										
	Bug Tracker Information	Buildable Systems	C-Family Programming	Development	Diversity In Size	Feature / Functionality / Requirements Criteria	Hosted In Git/Github	Inactive Projects	Intensity Of Development Activity	Issue Tracker	Java Programming	Js/Javascript	Other Programming	Project Age / Maturity	Python Programming	Random Selection	Service-Oriened	Size (In Loc)	Size (In Modules)	Size (In Other Metric)	Specific Application	Tested Systems	Use Of Best Practices	Used In Previous Studies	Various Domains	Version Control	Widely Used / Popular
User Reviews	1				1						1							1		1	1		2		1		
Privacy																					1		2				
Modularity														1												1	1
Code Transformation / Compilers			1								1													1			
CI/CD							1		1		2			1						1				2		2	
Team Management				1	1									1						1					1	1	
Service Oriented Architectures																	1		1	1	1		1	1			
User Experience																					2		1		1	1	1
Aesthetics											1						1										1
Slicing			1															1				1					

Final Goal	Criteria																										
	Bug Tracker Information	Buildable Systems	C-Family Programming	Development	Diversity In Size	Feature / Functionality / Requirements Criteria	Hosted In Git/Github	Inactive Projects	Intensity Of Development Activity	Issue Tracker	Java Programming	Js/Javascript	Other Programming	Project Age / Maturity	Python Programming	Random Selection	Service-Oriened	Size (In Loc)	Size (In Modules)	Size (In Other Metric)	Specific Application	Tested Systems	Use Of Best Practices	Used In Previous Studies	Various Domains	Version Control	Widely Used / Popular
Agile Methodologies									1												1		1				1
Reuse		1		1							1			1	1												2
Dynamic Analysis																						1	1				
Testability																	1	1		1							
Program Comprehension											1							1					1				
Exception Handling											1							1	1	1							
Parallelization			1															1				1					
Cloud-Based Software																	1				1					1	
Business Parameters				1							1								1	1					1	1	

Final Goal	Criteria																										
	Bug Tracker Information	Buildable Systems	C-Family Programming	Development	Diversity In Size	Feature / Functionality / Requirements Criteria	Hosted In Git/Github	Inactive Projects	Intensity Of Development Activity	Issue Tracker	Java Programming	Js/Javascript	Other Programming	Project Age / Maturity	Python Programming	Random Selection	Service-Oriened	Size (In Loc)	Size (In Modules)	Size (In Other Metric)	Specific Application	Tested Systems	Use Of Best Practices	Used In Previous Studies	Various Domains	Version Control	Widely Used / Popular
Domain Specific Languages																											
Software Analytics																											
Software Anomalies																											1
Deployment																	1									1	
Code Annotations							1				1									1							
Application Domain Analysis				1				1																			
Reliability																											
User Behaviour															1						1						1

Για παράδειγμα, αν κάποιος ερευνητής επιθυμεί να διενεργήσει μια μελέτη σχετικά με σφάλματα/ελαττώματα/λάθη (Bugs/Defects/Faults) θα πρέπει να λάβει υπόψιν του ότι η πλειονότητα της βιβλιογραφίας έχει χρησιμοποιήσει έργα γραμμένα σε Java, που επαναλαμβάνονται σε διαδοχικές μελέτες (Used In Previous Studies) που έχουν επιτυχώς ελεγχθεί (tested systems) και που διαθέτουν ένα bug trucking system. Ομοίως, αν κάποιος ερευνητής επιθυμεί να διεξάγει μια μελέτη σχετικά με την ασφάλεια (Security) θα πρέπει να λάβει υπόψιν του ότι η πλειονότητα της βιβλιογραφίας έχει επιλέξει έργα που έχουν χρησιμοποιήσει καλές πρακτικές (Use Of Best Practices), σε συγκεκριμένο τομέα εφαρμογής ή τεχνολογίας (Specific Application Domain Or Technology), καθώς και είναι γραμμένα σε γλώσσα προγραμματισμού Java.

4.1.3 Έργα που χρησιμοποιούνται ως υποκείμενα προς εμπειρικές μελέτες

Από τον Πίνακα 4.4, καταλαβαίνουμε ότι τα έργα apache-camel και apache-commons-lang είναι τα πιο μελετημένα έργα στις εμπειρικές μελέτες που ερευνήσαμε με συχνότητα 48 (2,74%) και 43 (2,74%). Αντίθετα, υπάρχουν πάνω από 1000 έργα που έχουν μελετηθεί μόνο μια φορά. Ο Πίνακας 4.4 παρουσιάζει έργα που κατ'ελάχιστον έχουν μελετηθεί σε 4 άρθρα. Τα 10 έργα με την μεγαλύτερη συχνότητα παρουσιάζονται παρακάτω :

- ***Apache Camel*[54]:** Το Apache Camel είναι ένα πλαίσιο ενοποίησης ανοιχτού κώδικα που παρέχει μια μηχανή δρομολόγησης και διαμεσολάβησης που βασίζεται σε κανόνες. Επιτρέπει στους χρήστες να ορίζουν και να εφαρμόζουν διάφορα πρότυπα διαμόρφωσης για να διευκολύνουν την ενοποίηση διαφορετικών συστημάτων και εφαρμογών.
- ***Apache Commons Lang* [55]**: Το Apache Commons Lang είναι μια βιβλιοθήκη που παρέχει ένα σύνολο κλάσεων και μεθόδων για κοινές εργασίες προγραμματισμού στην Java. Περιλαμβάνει βοηθητικά προγράμματα για τον χειρισμό συμβολοσειρών, την σειριοποίηση αντικειμένων, τον χειρισμό ενός πίνακα και πολλά άλλα.
- ***Apache Commons Math* [56]**: Το Apache Commons Math είναι μια βιβλιοθήκη που παρέχει μαθηματικά και στατιστικά στοιχεία για εφαρμογές Java. Περιλαμβάνει κλάσεις και μεθόδους για αριθμητική ανάλυση, γραμμική άλγεβρα, κατανομές πιθανοτήτων, βελτιστοποίηση και στατιστική ανάλυση.
- ***Apache Ant* [57]**: Το Apache Ant είναι ένα εργαλείο αυτόματης κατασκευής που χρησιμοποιείται κυρίως για έργα Java. Επιτρέπει στους προγραμματιστές να ορίζουν και να αυτοματοποιούν τις διαδικασίες κατασκευής, δοκιμής και ανάπτυξης των εφαρμογών τους.
- ***Apache Lucene* [58]**: Το Apache Lucene είναι μια βιβλιοθήκη αναζήτησης πλήρους κειμένου υψηλής απόδοσης. Παρέχει δυνατότητες εύρεσης και αναζήτησης για έγγραφα που βασίζονται σε κείμενο και χρησιμοποιείται ευρέως σε εφαρμογές που απαιτούν λειτουργίες αναζήτησης, όπως μηχανές αναζήτησης, συστήματα διαχείρισης περιεχομένου και εργαλεία ανάλυσης δεδομένων.
- ***Apache Log4j* [59]**: Το Apache Log4j είναι ένα πλαίσιο καταγραφής που παρέχει έναν ευέλικτο και διαμορφώσιμο τρόπο καταγραφής μηνυμάτων σε εφαρμογές Java. Υποστηρίζει διάφορα επίπεδα καταγραφής, εναλλαγή και προσαρμόσιμες μορφές αρχείων καταγραφής και καταγραφή σε πολλούς προορισμούς .
- ***jEdit* [60]**: Το jEdit είναι ένα πρόγραμμα επεξεργασίας κειμένου γραμμένο σε Java. Έχει σχεδιαστεί για προγραμματιστές και προσφέρει δυνατότητες όπως επισήμανση σύνταξης, αναδίπλωση κώδικα, υποστήριξη προσθηκών και εκτενείς επιλογές προσαρμογής.

- ***JFreeChart*** **[61]**: Το JFreeChart είναι μια βιβλιοθήκη Java για τη δημιουργία γραφημάτων. Παρέχει ένα ευρύ φάσμα τύπων γραφημάτων, συμπεριλαμβανομένων γραμμικών γραφημάτων, πίτας και άλλα, και υποστηρίζει την εξατομίκευση και τις διαδραστικές λειτουργίες.
- ***Spring Framework*** **[62]**: Το Spring Framework είναι ένα δημοφιλές πλαίσιο Java για τη δημιουργία εταιρικών εφαρμογών. Παρέχει ένα ολοκληρωμένο μοντέλο προγραμματισμού και διαμόρφωσης που απλοποιεί την ανάπτυξη ισχυρών, επεκτάσιμων και συντηρήσιμων εφαρμογών.
- ***Apache HBase*** **[63]**: Το Apache HBase είναι μια κατανεμημένη, επεκτάσιμη και συνεπής βάση δεδομένων NoSQL που χτίστηκε με βάση το Apache Hadoop. Παρέχει πρόσβαση ανάγνωσης και εγγραφής μεγάλων συνόλων δεδομένων σε πραγματικό χρόνο και έχει σχεδιαστεί για να χειρίζεται μεγάλο φόρτο δεδομένων.

Από τον Πίνακα 4.4 και από την παραπάνω συζήτηση μπορούμε να παρατηρήσουμε εύκολα ότι τα περισσότερα έργα ανοικτού λογισμικού προέρχονται από το οικοσύστημα της apache. Ενδιαφέρον εδώ είναι να τονίσουμε ότι πολλές μελέτες θέτουν το οικοσύστημα προέλευσης έργων (π.χ. apache, eclipse, google) ως κριτήριο επιλογής. Η συγκεκριμένη απόφαση στοχεύει στο να διασφαλίσει πολλά από τα κριτήρια που αναφέρθηκαν στο κεφάλαιο 4.1.2 όπως η χρήση καλών πρακτικών, η μεγάλη αποδοχή από τους χρήστες, το ευρύ ιστορικό, το μέγεθος κ.α.

Πίνακας 4.4 Συχνότητα έργων

Project	Freq.	Project	Freq.
apache-camel	48	weka	6
apache-commons-lang	43	curl	6
apache-commons-math	39	apache-commons configuration	6
apache-ant	35	apache-commons-net	6
apache-lucene	33	apache-kylin	6
apache-log4j	28	apache-avro	6
jedit	27	jenkins	6
jfreechart	25	netty	6
spring framework	24	apache-beam	6
apache-hbase	24	apache-flume	6
apache-cassandra	24	jetty	6
apache-xalan	23	petclinic	5
apache-hive	23	totinfo	5
apache-hadoop	23	replace	5
apache-commons-io	21	libtiff	5
apache-xerces	21	apache-spark	5
apache-commons-closure	21	antennapod	5
apache-wicket	19	couchbase	5
apache-poi	19	bugzilla	5
eclipse-jdt	19	columba	5
eclipse-core	19	postgresql	5
apache-ivy	18	equinox	5
apache-commons-collections	17	apache-deltaspike	5
apache-derby	16	apache-giraph	5
apache-activemq	16	apache-jspwiki	5

Project	Freq.	Project	Freq.
apache-tomcat	16	apache-knox	5
github organization	16	apache-nutch	5
apache-commons-mockito	16	apache-opennlp	5
joda-time	16	apache-santuario	5
apache-zookeeper	16	jabref	5
apache-velocity	15	netflix organization	5
chart	14	apache-httpcomponents	5
apache-flink	14	apache-struts	5
elasticsearch	13	hsqldb	5
google-guava	13	apache-jena	5
apache-synapse	13	pmd	5
apache-commons-codec	13	coreutils	5
apache-storm	12	wireshark	5
pde	12	libreoffice	5
argouml	12	tensorflow	5
apache-jmeter	12	apache-phoenix	5
apache-groovy	12	gcc	5
junit	11	apache-hdfs	5
apache-mahout	11	wordpress	5
apache-jackrabbit	10	notepad	5
gzip	10	chromium	4
qt	10	apache-qpid	4
google-gson	10	sed	4
apache-cxf	10	eq	4
hibernate	10	space	4
apache-maven	10	ansible	4
apache-kafka	10	geronimo	4
jackson-core	10	apache-mesos	4
apache-ambari	9	sentry	4
openstack	9	apache-zeppelin	4
reactivex-rxjava	9	jgit	4
apache-mylyn	9	connectbot	4
apache-cayenne	9	javaparser-organization	4
zxing	9	apache-jxpath	4
alibaba-fastjson	9	okhttp	4
apache-commons-bcel	9	busybox	4
mozilla-organization	9	nova	4
ffmpeg	9	apache-commons-digester	4
apache-jruby	9	apache-commons-vfs	4
apache-organization	9	apache-lens	4
time	8	apache-manifoldcf	4
checkstyle	8	tika	4
apache-commons-cli	8	apache-tez	4
jsoup	8	gitlab	4

Project	Freq.	Project	Freq.
linux	8	rhino	4
apache-calcite	8	zipkin	4
apache-commons-compress	8	matplotlib	4
apache-accumulo	8	react	4
apache-dubbo	8	apache-drill	4
django	8	apache-jclouds	4
pandas	7	colt	4
k-9 mail	7	eclipse-emf	4
apache-openjpa	7	fresco	4
apache-pig	7	php	4
jhotdraw	7	wget	4
android	7	apache-druid	4
apache-ignite	7	asterisk	4
apache-commons-beanutils	7	promise	4
apache-commons-dbcp	7	apache-aries	4
apache-commons-validator	7	keras	4
apache-pdfbox	7	gerrit	4
openssl	7	firefox	4
python	7	vlc	4
swt	6	qemu	4
eclipse	6	deeplearning4j	4
apache-karaf	6	aspectj	4
apache-thrift	6	quantum	4
apache-flex	6	apache-commons-csv	4
grep	6	apache-bookkeeper	4
squirrel	6	printtoken	4
freemind	6	tcas	4

Καθοδήγηση ερευνητών στην επιλογή κριτηρίων. Με βάση τα παραπάνω, στο Πίνακα 4.5 παρουσιάζονται οι διασταυρώσεις συχνοτήτων μεταξύ στόχων και έργων. Ο πίνακας αυτός είναι ιδιαίτερα σημαντικός κατά τη φάση σχεδίασης εμπειρικών μελετών όπου ο κάθε ερευνητής με βάση τον στόχο του μπορεί να εντοπίσει τα έργα ανοιχτού λογισμικού που θα πρέπει να τον απασχολήσουν εφόσον ο στόχος του είναι να επαναλάβει προηγούμενες μελέτες δουλεύοντας στο ίδιο σύνολο δεδομένων (dataset). Η συγκεκριμένη ενέργεια μπορεί να ωφελήσει τους ερευνητές ως προς τη συγκρισιμότητα των αποτελεσμάτων και την επαναχρησιμοποίηση μεταβλητών και τιμών τους για τα συγκεκριμένα έργα. Ο Πίνακας 4.5 είναι ταξινομημένος ανά στόχο με φθίνουσα συχνότητα.

Για παράδειγμα, ένας ερευνητής που θα ήθελε να μελετήσει για «σφάλματα/ελαττώματα/λάθη» (Bugs/Defects/Faults) και να συλλέξει δεδομένα για 10 έργα, θα μπορούσε να επιλέξει τα παρακάτω έργα ανοικτού λογισμικού: apache-camel, apache-lucene, apache-commons-lang, apache-commons-math, apache-xalan, apache-poi, apache-ivy, apache-log4j, eclipse-jdt, και apache-ant και να επαναχρησιμοποιήσει το υπάρχον dataset και να συγκρίνει τα αποτελέσματα του με αυτά των προηγούμενων μελετών.

Πίνακας 4.5 Διασταύρωση Συχνοτήτων τελικού στόχου με τα έργα της εμπειρικής μελέτης

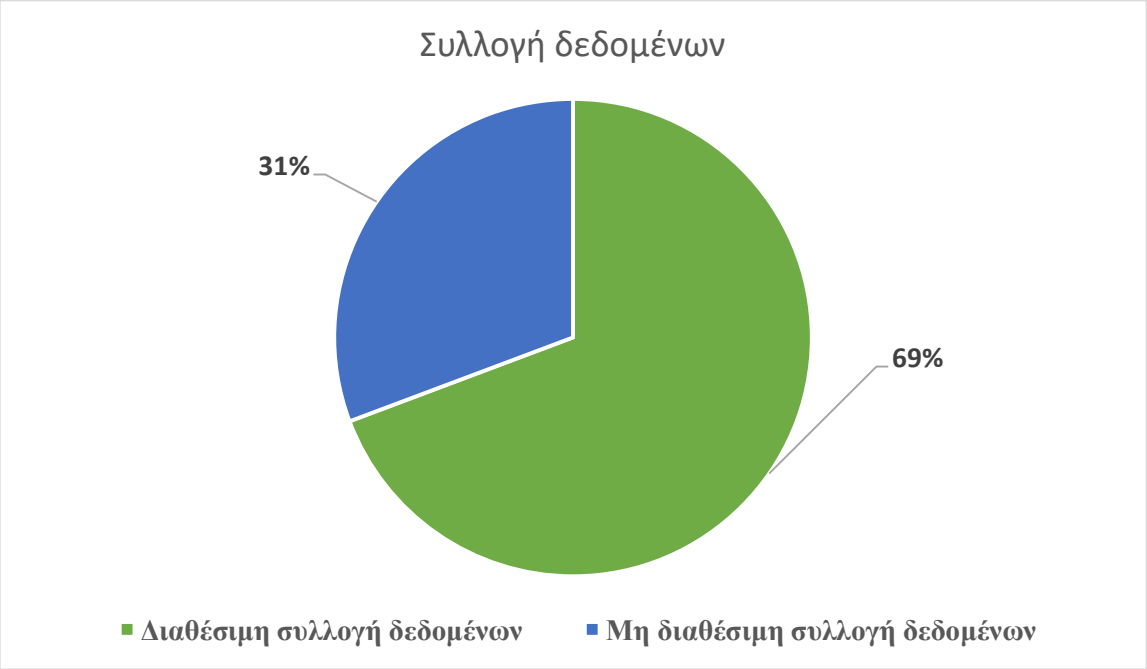
Final Goal	Projects	Freq.
Bugs / Defects / Faults	apache-camel	29
	apache-lucene	19
	apache-commons-lang	19
	apache-commons-math	19
	apache-xalan	17
	apache-poi	16
	apache-ivy	16
	apache-log4j	15
	eclipse-jdt	15
	apache-ant	14
	jedit	14
	apache-velocity	14
	apache-xerces	13
	apache-commons-closure	13
	pde	12
	apache-hbase	12
	apache-hive	12
	apache-synapse	12
	apache-activemq	11
	apache-derby	10
	apache-tomcat	9
	apache-commons-collections	9
	chart	8
	apache-commons-mockito	8
	apache-mylyn	7
	mozilla-organization	7
	apache-commons-codec	7
	apache-commons-io	7
	apache-wicket	7
	time	6
	apache-commons-compress	6
	apache-zookeeper	6
	apache-groovy	6
	apache-storm	6
	jackson-core	6
	swt	5
	zxing	5
	joda-time	5
	jfreechart	5
	postgresql	5
	equinox	5
	apache-calcite	5
	apache-commons-dbcp	5
	apache-kylin	5
	apache-jruby	5
	eclipse-core	5
	apache-hadoop	5
	eclipse	4
	apache-ambari	4
	apache-cayenne	4

Final Goal	Projects	Freq.
	eq	4
	apache-commons-bcel	4
	apache-commons-beanutils	4
	apache-commons configuration	4
	apache-commons-net	4
	apache-commons-validator	4
	apache-commons-vfs	4
	apache-mahout	4
	apache-avro	4
	apache-tez	4
	apache-commons-cli	4
	apache-flink	4
	apache-ignite	4
	aspectj	4
	quantum	4
	gzip	4
	apache-openjpa	3
	lc	3
	ml	3
	safe	3
	bugzilla	3
	apache-archiva	3
	apache-commons-digester	3
	apache-commons-jcs	3
	apache-commons-jexl	3
	apache-deltaspike	3
	apache-giraph	3
	apache-jspwiki	3
	apache-knox	3
	apache-lens	3
	apache-nutch	3
	apache-parquet	3
	apache-santuario	3
	apache-accumulo	3
	apache-beam	3
	apache-httpcomponents	3
	apache-jxpath	3
	apache-cassandra	3
	apache-cxf	3
	apache-kafka	3
	apache-commons-csv	3
	elasticsearch	3
	geronimo	3
	printtoken	3
	tcas	3
	totinfo	3
	apache-flex	3
	apache-aries	3
	apache-organization	3
	roslyn	3
Localization	apache-commons-lang	14
	apache-commons-math	14

Final Goal	Projects	Freq.
	apache-commons-closure	12
	chart	8
	apache-commons-mockito	8
	time	6
	jackson-core	6
	apache-hive	5
	apache-camel	4
	joda-time	4
	jfreechart	4
	apache-commons-compress	4
	apache-commons-collections	4
	apache-hbase	4
	apache-commons-codec	3
	swt	3
	aspectj	3
	eclipse-jdt	3
	apache-commons-csv	3
	gzip	3
	printtoken	3
	tcas	3
	totinfo	3
Testing	apache-commons-lang	13
	apache-commons-math	12
	joda-time	7
	gzip	5
	grep	5
	apache-commons-io	5
	jfreechart	5
	apache-hadoop	5
	google-guava	5
	apache-flex	4
	sed	4
	replace	4
	apache-commons-closure	4
	apache-commons-collections	4
	alibaba-fastjson	4
	google-gson	4
	apache-ant	4
	apache-cassandra	4
	apache-flink	4
	apache-hbase	4
	petclinic	4
	apache-jmeter	4
	apache-log4j	4
	totinfo	3
	make	3
	weka	3
	apache-camel	3
	apache-hive	3
	apache-karaf	3
	restcountries	3
	apache-commons-codec	3

Final Goal	Projects	Freq.
	colt	3
Code Review	qt	8
	openstack	5
	eclipse-core	5
	couchbase	4
	android	3
Program Repair	apache-commons-closure	7
	apache-commons-lang	7
	apache-commons-math	7
	chart	4
	time	3
	apache-commons-mockito	3
	libtiff	3
	jfreechart	3
	joda-time	3
	apache-camel	3
Security	github organization	6
	ffmpeg	5
	coreutils	3
	linux	3
	wireshark	3
Logs	apache-hadoop	6
	apache-zookeeper	6
	apache-hdfs	3
	apache-activemq	3
Statistic Analysis	apache-cassandra	5
	apache-ant	3
Bad Smells	jfreechart	5
	ganttproject	4
	apache-ant	4
	apache-xerces	4
	apache-cassandra	4
	checkstyle	3
	jasperreports	3
	apache-lucene	3
	squirrel	3
	eclipse-core	3
	junit	3
	apache-camel	3
	apache-hive	3
Version Control	apache-hadoop	5
	elasticsearch	4
	apache-activemq	3
	apache-camel	3
	apache-derby	3
	apache-hbase	3
	apache-openjpa	3
	spring framework	3
Vulnerabilities	github organization	5
	ffmpeg	4
	coreutils	3
Quality Metrics	apache-commons-lang	4

Final Goal	Projects	Freq.
	joda-time	4
	eclipse-jdt	4
	apache-lucene	4
	apache-cassandra	4
	apache-hbase	4
	apache-commons-math	3
	apache-commons-closure	3
	jfreechart	3
	apache-commons-mockito	3
	checkstyle	3
	apache-commons-io	3
	junit	3
	pde	3
	apache-ant	3
	apache-camel	3
	apache-hive	3
	apache-wicket	3
	netty	3
	apache-xerces	3
Technical Dept	quantum	4
	apache-beam	3
	apache-dubbo	3
	apache-pdfbox	3
	apache-ant	3
	argouml	3
	columba	3
	eclipse-emf	3
	hibernate	3
	jedit	3
	jfreechart	3
	apache-jmeter	3
	apache-jruby	3
	squirrel	3
Code/Test Generation	freemind	4
	weka	3
Dependency Analysis	github organization	4
	apache-thrift	3
Comments	elasticsearch	3
	google-guava	3
Application Programming Interfaces	apache-log4j	3
Documentation	hibernate	3
Effort Estimation	spring framework	3
Human Factors	apache-lucene	3



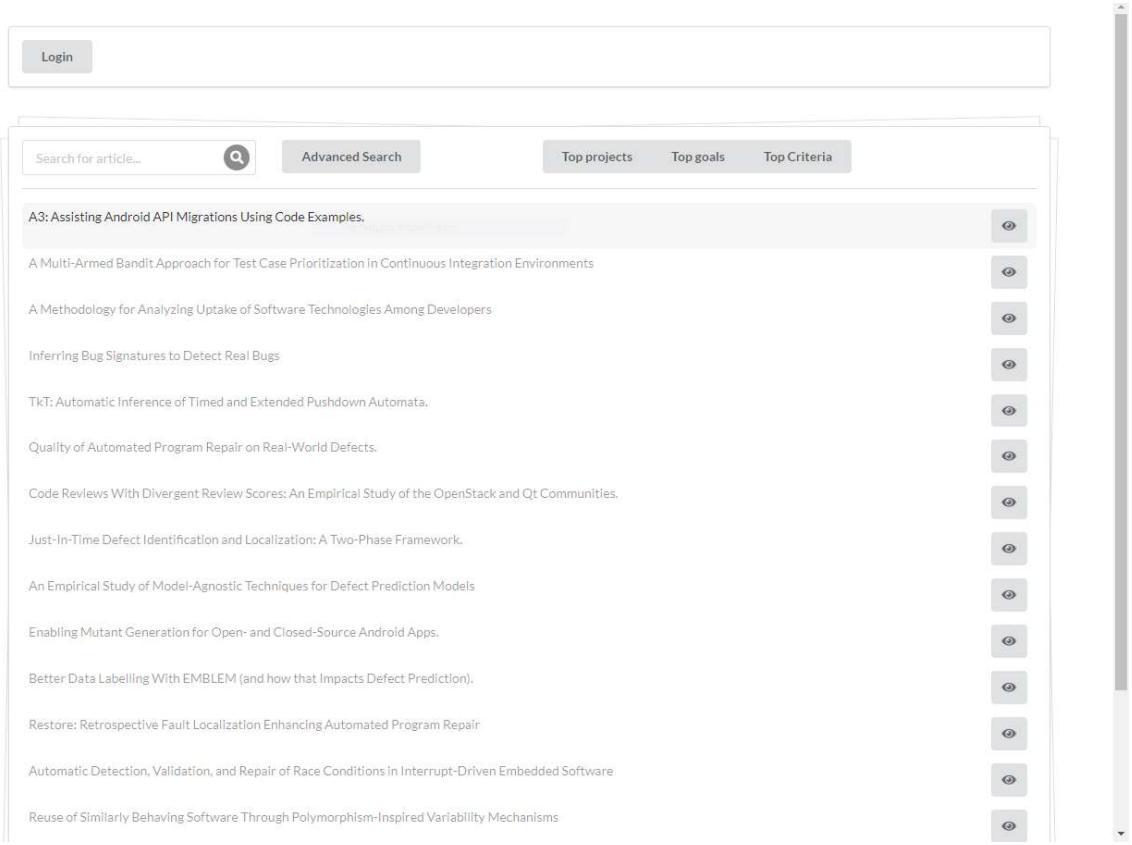
Σχήμα 4.1 Συχνότητα συλλογής δεδομένων

Τέλος, από το Σχήμα 4.1 προκύπτει ότι οι ερευνητές τείνουν να κάνουν διαθέσιμη τη συλλογή δεδομένων τους. Πιο συγκεκριμένα, από τις 394 εμπειρικές μελέτες που ερευνήσαμε , στις 273 μελέτες (69%) η συλλογή δεδομένων είναι διαθέσιμη ενώ στις 121 δεν είναι διαθέσιμη.

4.2 Παρουσίαση εργαλείου

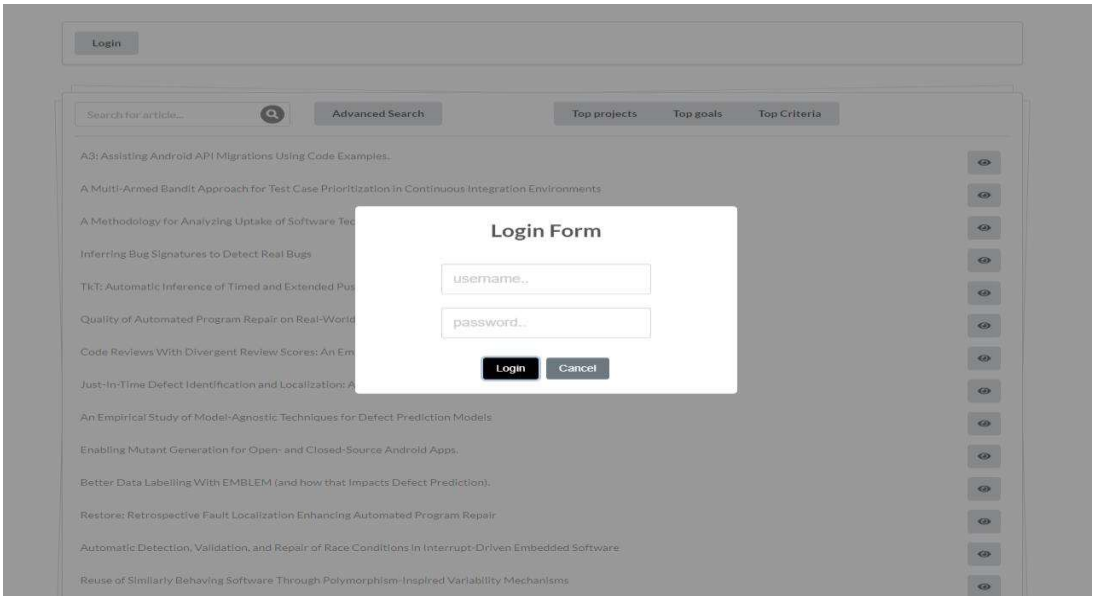
Σε αυτή την ενότητα παρουσιάζουμε την ιστοσελίδα που δημιουργήσαμε . Ο ιστότοπος της είναι ο εξής: https://users.iee.ihu.gr/~it154605/Thesis/ .

Στο Σχήμα 4.2, παρουσιάζεται η αρχική σελίδα του εργαλείου και αποτελεί την πρώτη επαφή της ιστοσελίδας με τον χρήστη-ερευνητή. Σε αυτήν απεικονίζεται η λίστα με τις ήδη υπάρχουσες μελέτες μαζί με ένα κουμπί εμφάνισης δεξιά (view), το κουμπί σύνδεσης «Login», μία μπάρα αναζήτησης άρθρων «Search for article…», ένα κουμπί σύνθετης αναζήτησης «Advanced Search» καθώς και τρία κουμπιά που αναφέρονται στα χαρακτηριστικά των μελετών (Top projects, Top goals, Top Criteria). Σε αυτό το σημείο ο ερευνητής μπορεί μόνο να αναζητήσει και να πληροφορηθεί για τις εμπειρικές μελέτες που υπάρχουν στη βάση.

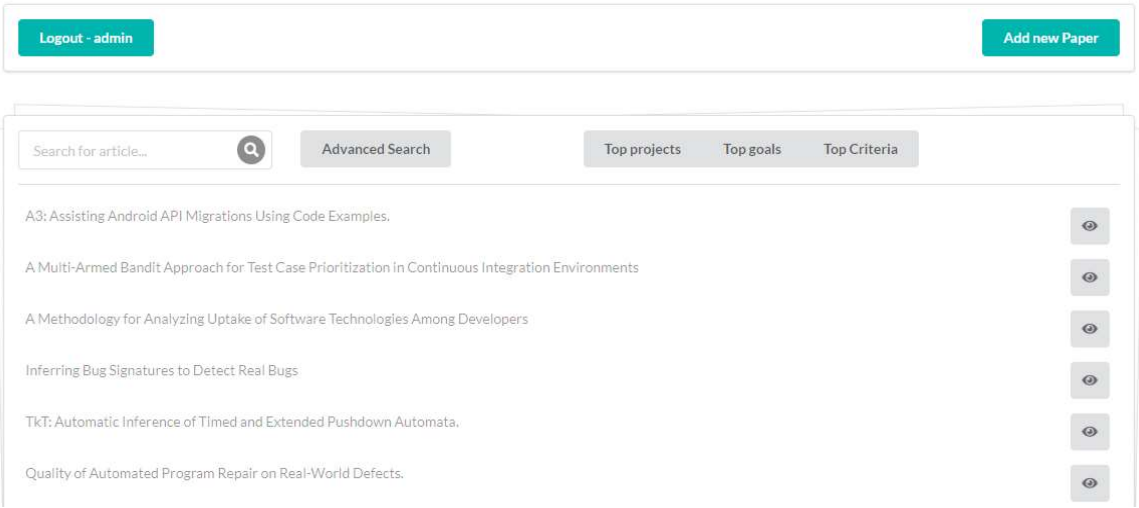


Σχήμα 4.2 Αρχική σελίδα της ιστοσελίδας

Στο Σχήμα 4.3, εμφανίζεται η φόρμα σύνδεσης του χρήστη . Σε αυτό το σημείο συμπληρώνονται τα στοιχεία σύνδεσης (credentials) .

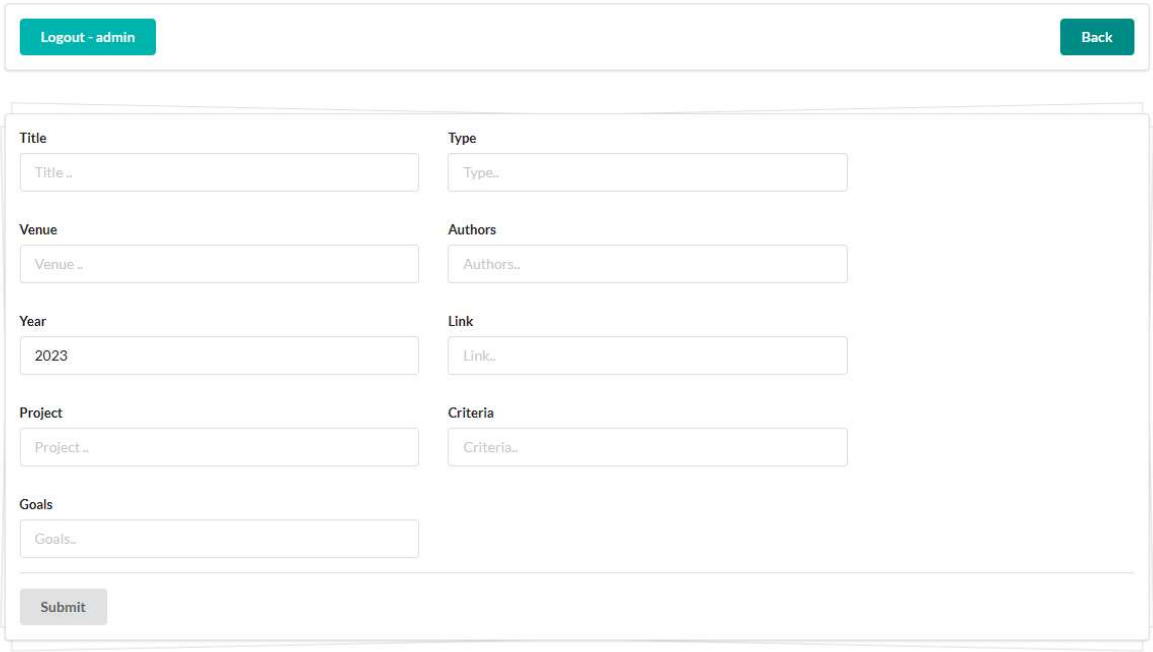


Σχήμα 4.3 Φόρμα σύνδεση χρήστη-ερευνητή



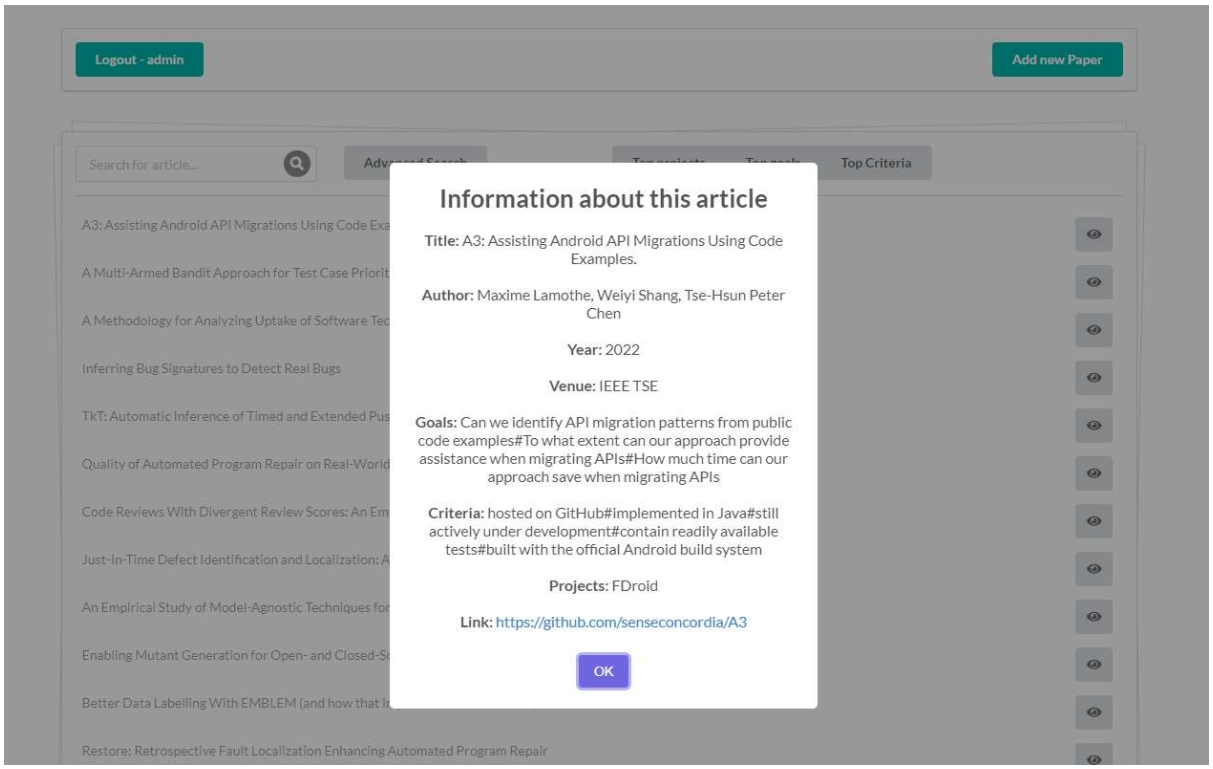
Σχήμα 4.4 Αρχική σελίδα συνδεδεμένου χρήστη

Στο Σχήμα 4.4, προβάλλεται η αρχική σελίδα που συναντά ένας συνδεδεμένος χρήστης. Ως παράδειγμα χρησιμοποιείται ένας χρήστης «admin» ,όπως μπορεί να παρατηρηθεί από το κουμπί αποσύνδεσης «Logout» πάνω αριστερά. Σε αυτό το σημείο ο ερευνητής έχει την δυνατότητα να προσθέσει καινούργιες εμπερικές μελέτες στο σύστημα. Για το λόγο αυτό δημιουργήθηκε το κουμπί «Add new Paper», πάνω δεξιά.



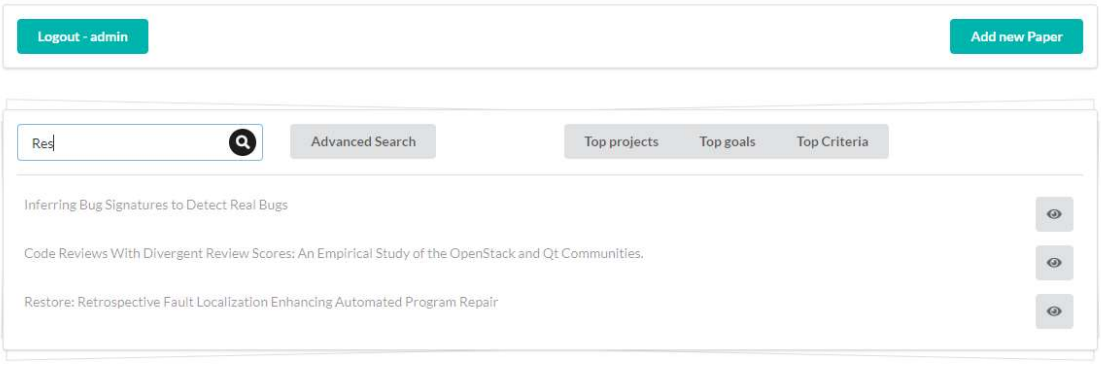
Σχήμα 4.5 Προσθήκη νέας εμπειρικής μελέτης.

Στο Σχήμα 4.5 απεικονίζεται φόρμα προσθήκης νέας εμπειρικής μελέτης στη βάση. Ο ερευνητής, πρέπει για λόγους ευχρηστίας και ευκολίας μελλοντικών ερευνητών να συμπληρώσει σημαντικά στοιχεία σχετικά με την μελέτη που θέλει να εντάξει. Τα χαρακτηριστικά προς συμπλήρωση με σειρά από πάνω προς τα κάτω είναι: Τίτλος μελέτης, Είδος δημοσίευσης, Τόπος δημοσίευσης, Συγγραφείς, Ημερομηνία Έκδοσης, Σύνδεσμος κατεύθυνσης στη συλλογή δεδομένων, Ονόματα των έργων που χρησιμοποιήθηκαν, Κριτήρια επιλογής των δεδομένων καθώς και Στόχων της μελέτης.



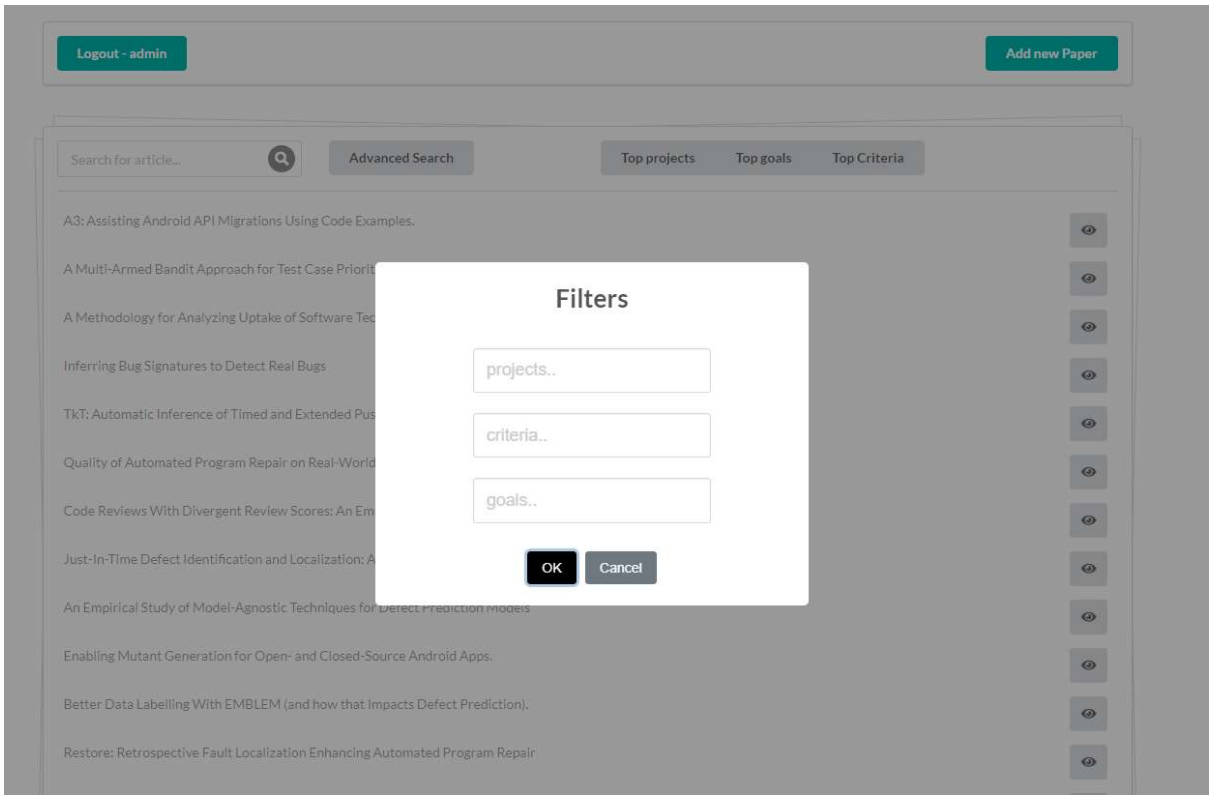
Σχήμα 4.6 Πληροφορίες άρθρων

Στο Σχήμα 4.6, προβάλλεται το πλαίσιο πληροφορίας που εμφανίζεται με το πάτημα του κουμπιού εμφάνισης (view) . Το κουμπί εμφάνισης βρίσκεται στα δεξιά του τίτλου της μελέτης και χρησιμοποιεί έναν χαρακτήρα μάτι. Οι πληροφορίες που περιλαμβάνει το πλαίσιο αυτό είναι ο τίτλος μελέτης, τα ονόματα των συγγραφέων, η ημερομηνία έκδοσης, ο τόπος δημοσίευσης, οι στόχοι της μελέτης, τα κριτήρια επιλογής, τα ονόματα των έργων και ο σύνδεσμος της συλλογής δεδομένων.



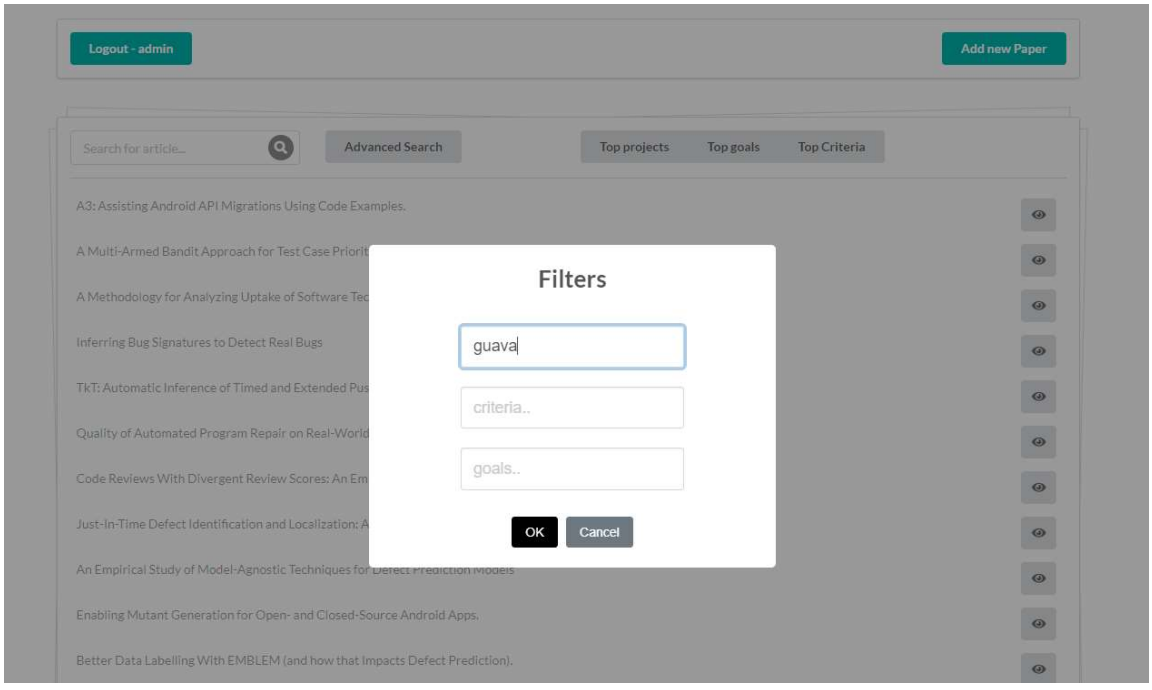
Σχήμα 4.7 Μπάρα αναζήτησης μελετών.

Στο Σχήμα 4.7, εμφανίζεται η χρήση της μπάρας αναζήτησης εμπειρικών μελετών. Σε αυτή, ο ερευνητής μπορεί να αναζητά μελέτες με βάση τον τίτλο. Πληκτρολογώντας τις λέξεις που επιθυμεί , εμφανίζονται αποτελέσματα που περιέχουν τις λέξεις αυτές.



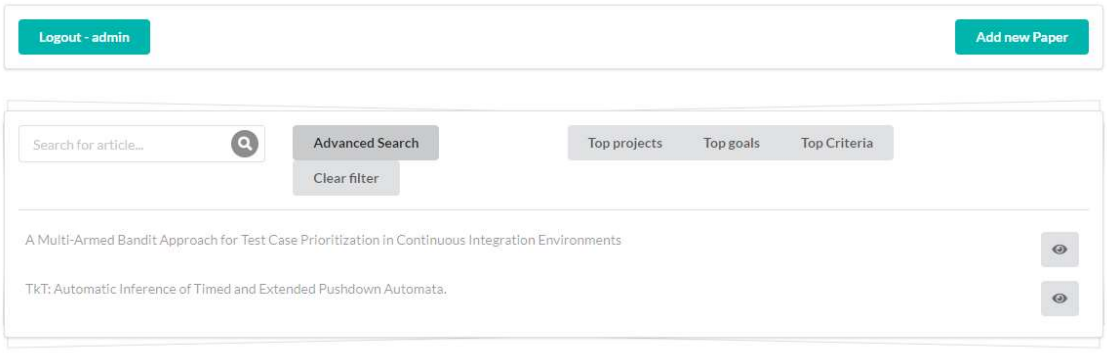
Σχήμα 4.8 Σύνθετη αναζήτηση.

Στο Σχήμα 4.8, εμφανίζεται το παράθυρο της σύνθετης αναζήτησης, που προκύπτει από το πάτημα του κουμπιού σύνθετης αναζήτησης (Advanced Search). Το παράθυρο αυτό δίνει την δυνατότητα στον ερευνητή να προσθέσει φίλτρο στην αναζήτηση μελετών. Τα φίλτρα περιλαμβάνουν τα έργα, τα κριτήρια επιλογής και τους στόχους που περιέχονται στις εμπειρικές μελέτες. Παραδείγματα εφαρμογής φίλτρων παραθέτονται στις παρακάτω εικόνες.



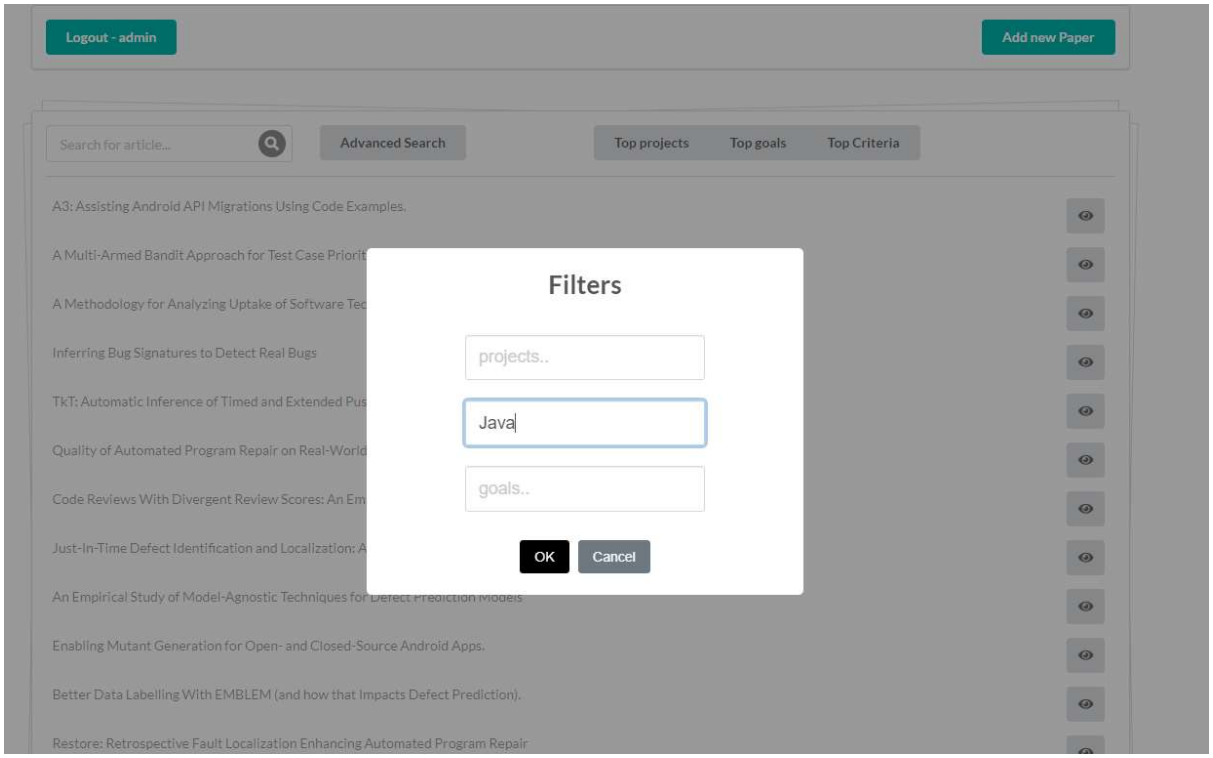
Σχήμα 4.9 Αναζήτηση με βάση το έργο.

Στο Σχήμα 4.9, ο ερευνητής προσθέτει φίλτρο αναζήτησης μελέτης με βάση το όνομα του έργου που περιλαμβάνει η μελέτη «guava».



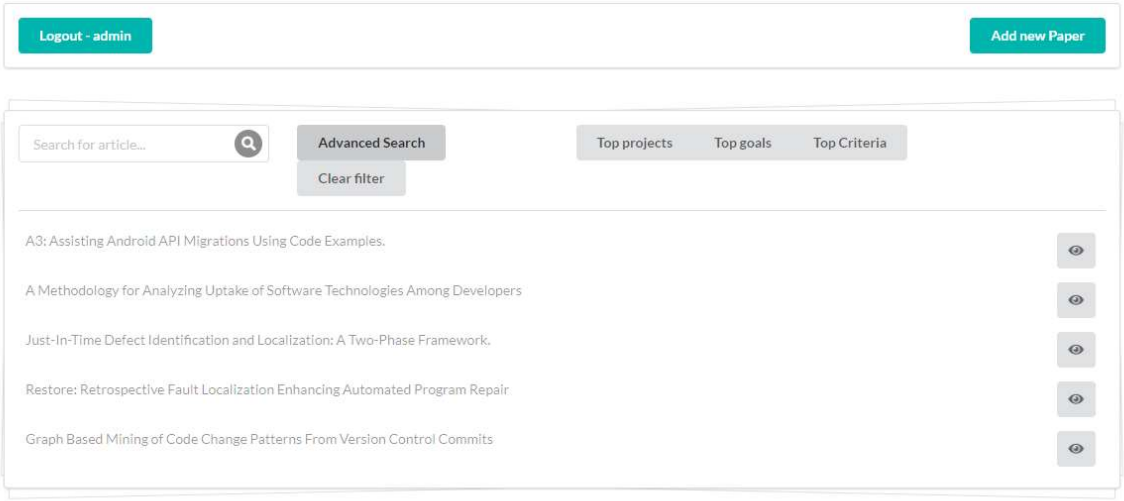
Σχήμα 4.10 Αποτελέσματα αναζήτησης.

Στο Σχήμα 4.10, παρουσιάζονται τα αποτελέσματα της αναζήτησης με βάση το έργο της εμπειρικής μελέτης. Επιπλέον , παρατηρείται το κουμπί καθαρισμού φίλτρων (Clear filter) ,το οποίο «καθαρίζει» την επιλογή των φίλτρων.



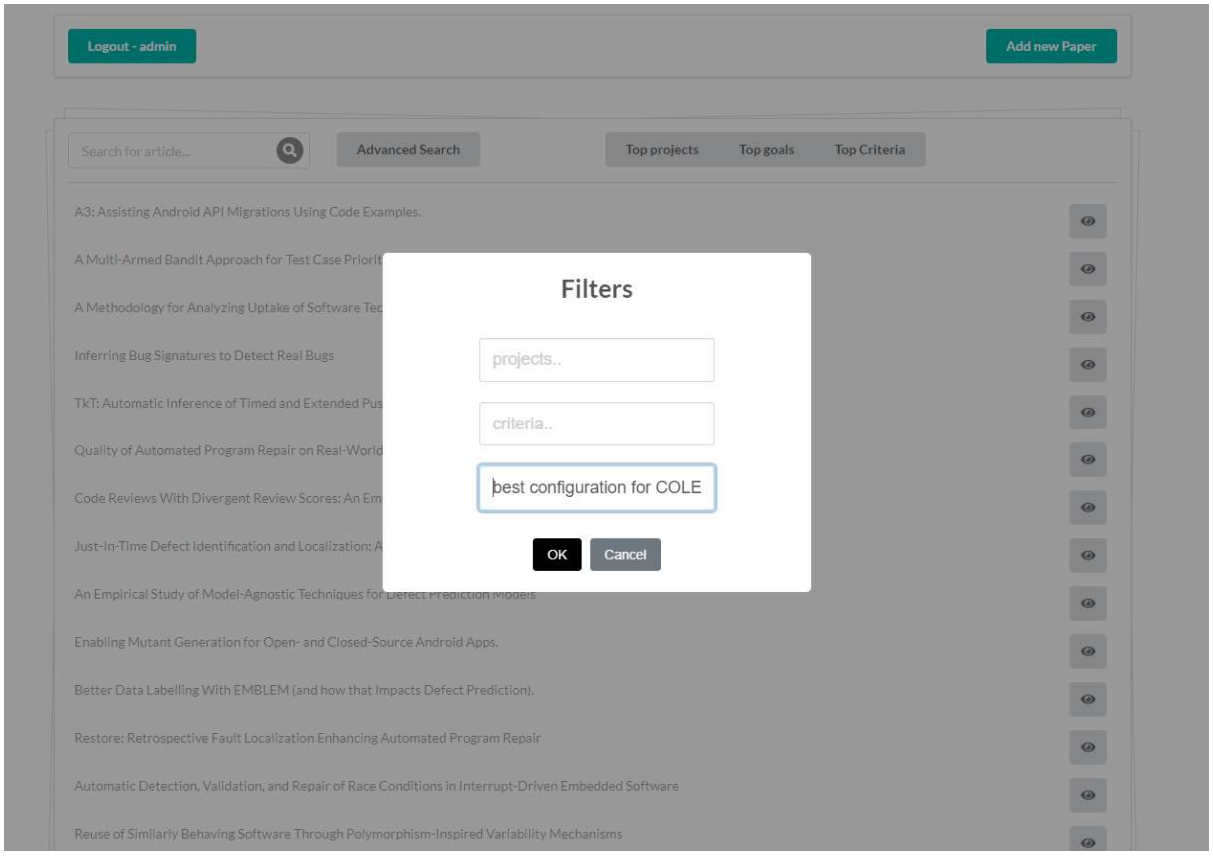
Σχήμα 4.11 Αναζήτηση με βάση το κριτήριο επιλογής .

Στο Σχήμα 4.11, ο ερευνητής προσθέτει φίλτρο αναζήτησης μελέτης με βάση το κριτήριο επιλογής έργων της μελέτης, στο παράδειγμα το κριτήριο επιλογής που ψάχνει ο ερευνητής είναι τα έργα να έχουν σαν γλώσσα προγραμματισμού τη Java.



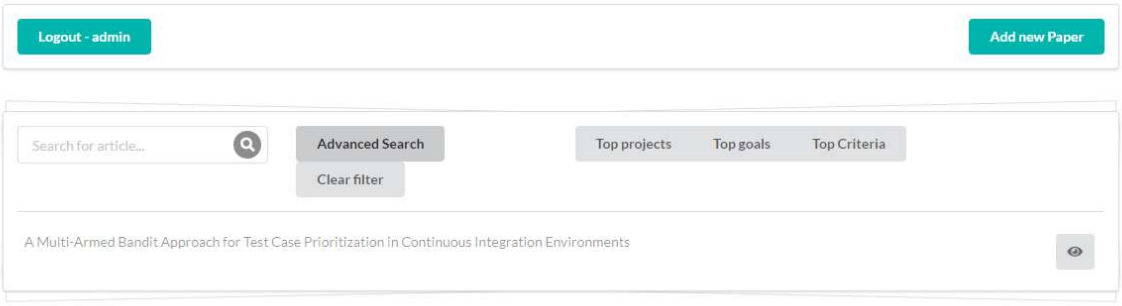
Σχήμα 4.12 Αποτελέσματα της αναζήτησης με βάση το κριτήριο επιλογής που θέσαμε.

Στο Σχήμα 4.12, εμφανίζονται η λίστα των εμπειρικών μελετών που χρησιμοποιούν ως κριτήριο επιλογής τη λέξη-κλειδί «Java».



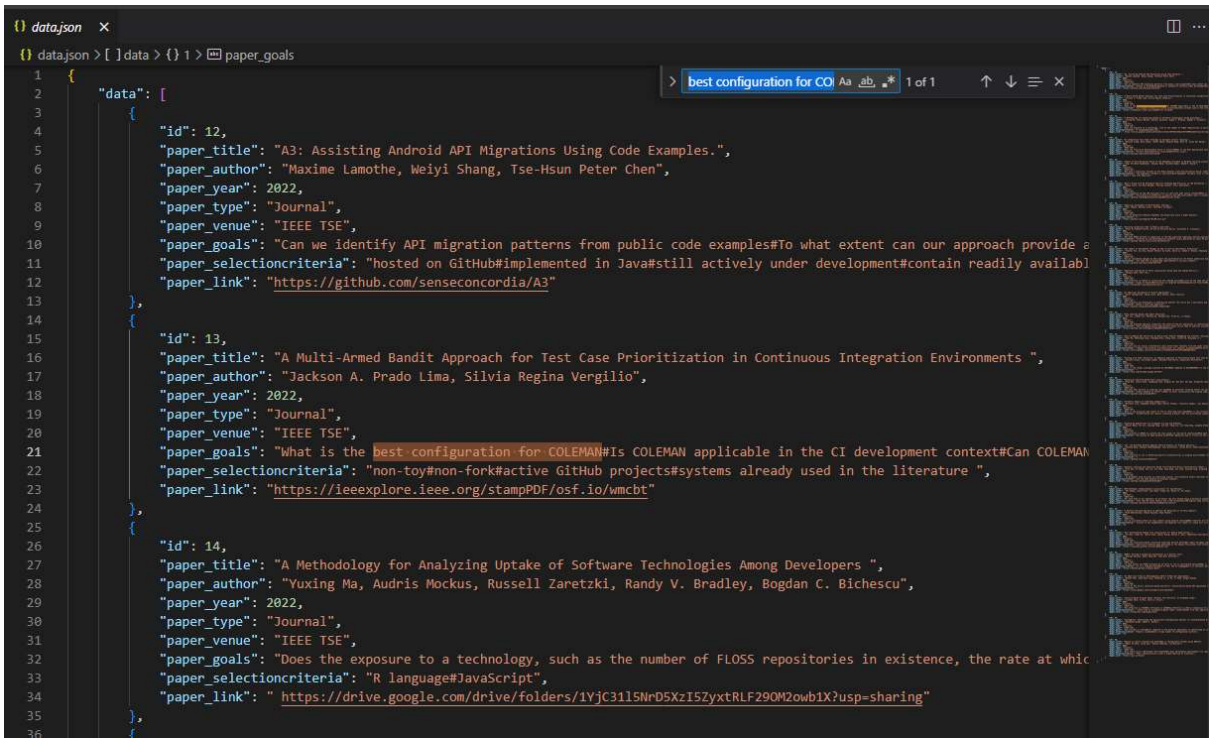
Σχήμα 4.13 Αναζήτηση με βάση τον στόχο της μελέτης.

Στο Σχήμα 4.13, ο ερευνητής ορίζει ως φίλτρο αναζήτησης μελέτης τον στόχο που επιθυμεί να έχει η εμπειρική μελέτη. Στο παράδειγμα θέσαμε ως αναζήτηση την καλύτερη διαμόρφωση για την μέθοδο COLEMAN.



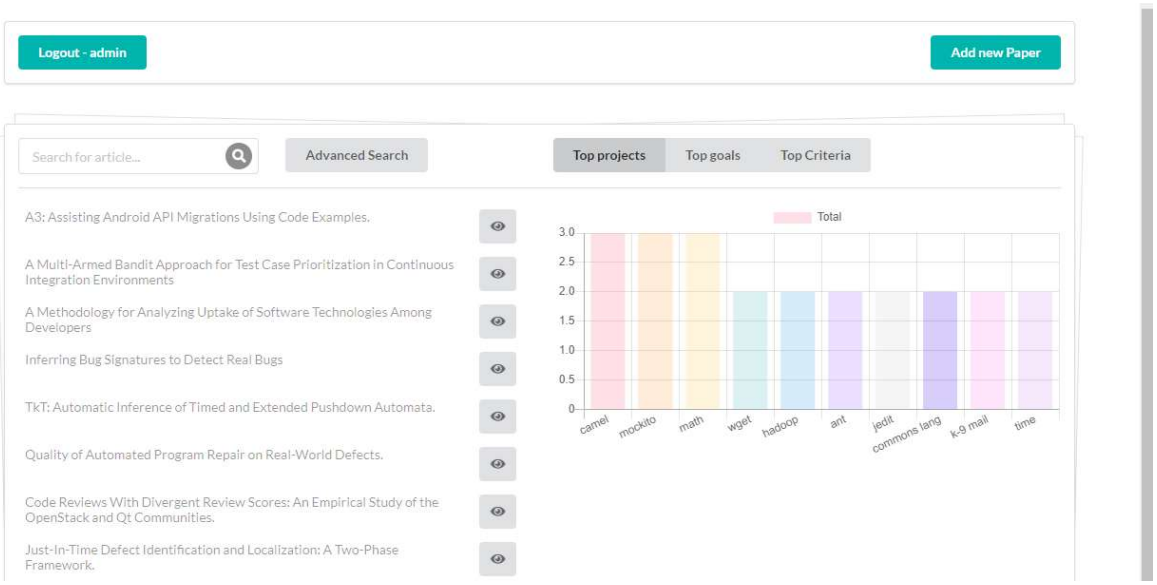
Σχήμα 4.14 Αποτελέσματα φίλτρου αναζήτησης με βάση το στόχο.

Στο Σχήμα 4.14, παρουσιάζονται οι εμπειρικές μελέτες που έχουν σαν στόχο το φίλτρο που ορίσαμε στην Εικόνα 12.



Σχήμα 4.15 Επιβεβαίωση αναζήτησης στα δεδομένα.

Στο Σχήμα 4.15, παρουσιάζεται η ανάγνωση των δεδομένων από τη βάση σε μορφή JSON. Για την επιβεβαίωση των αποτελεσμάτων που λάβαμε στο Σχήμα 4.14, πραγματοποιήθηκε αναζήτηση στα δεδομένα της βάσης. Παρατηρείται ότι πράγματι για την καλύτερη διαμόρφωση της μεθόδου COLEMAN αντιστοιχεί μόνο μία εμπειρική μελέτη.



Σχήμα 4.16 Στατιστικά στοιχεία.

Ο ερευνητής πατώντας ένα από τα κουμπιά των χαρακτηριστικών που συλλέξαμε από τις μελέτες μπορεί να έχει πρόσβαση στα στατιστικά στοιχεία τους προκειμένου να μπορεί να πληροφορηθεί για τα πιο διαδεδομένα χαρακτηριστικά. Στο παράδειγμα, παρατηρούμε ότι στα στατιστικά που αφορούν τα πιο διαδομένα έργα στις εμπειρικές μελέτες είναι το apache-camel, το apache-commons-mockito και το apache-commons-math.

Κεφάλαιο 5ο: Συμπεράσματα και Μελλοντική Έρευνα

Στην παρούσα πτυχιακή εργασία παρουσιάσαμε τα αποτελέσματα μιας βιβλιογραφικής συστηματικής χαρτογράφησης σχετικά με τα έργα ανοιχτού λογισμικού σε εμπειρικές μελέτες θέτοντας τους τρεις παρακάτω στόχους: (α) να παρέχει μια επισκόπηση των στόχων που ορίζουν οι MSR μελέτες, που παίρνουν τη συλλογή δεδομένων τους μέσω εξόρυξης, (β) να διερευνήσει τα κριτήρια επιλογής που βάζουν οι ερευνητές στις εμπειρικές τους μελέτες, και (γ) να ανιχνεύσει τα περισσότερο μελετημένα έργα στις μελέτες αυτές. Για την επίτευξη αυτών των στόχων, εξερευνήσαμε περισσότερα από 1400 άρθρα, από τα οποία προχωρήσαμε στην εξαγωγή δεδομένων για τα 394. Όσον αφορά τον πρώτο στόχο, η μελέτη αποκάλυψε ότι οι μελέτες MSR, συχνά προσπαθούσαν να αποκαλύψουν πληροφορίες σχετικά με τον εντοπισμό σφαλμάτων λογισμικού, έλεγχο λογισμικού καθώς και με την ασφάλεια των έργων ανοικτού λογισμικού. Όσον αφορά τα κοινά κριτήρια επιλογής, η μελέτη προσδιόρισε αρκετά κοινά κριτήρια που χρησιμοποιούνται σε μελέτες OSS. Αυτά τα κριτήρια περιλάμβαναν διάφορες διαστάσεις, όπως η Java ,ως γλώσσα προγραμματισμού, και η δημοτικότητα του έργου. Η εφαρμογή αυτών των κριτηρίων επέτρεψε στους ερευνητές να αξιολογήσουν την ασφάλεια, την ανακατασκευή και τη δοκιμή έργων OSS. Τέλος, η μελέτη διερεύνησε τη χρήση κοινών έργων ως υποκείμενα σε εμπειρικές μελέτες. Εξέτασε την επιλογή συγκεκριμένων έργων για εις βάθος ανάλυση και εξερεύνηση. Εστιάζοντας σε αυτά τα κοινά έργα OSS, οι ερευνητές μπόρεσαν να αποκτήσουν ολοκληρωμένες γνώσεις σχετικά με την τοπική προσαρμογή, την ποιότητα του κώδικα και τις δοκιμές.

Επιπρόσθετα, με βάση τα παραπάνω αποτελέσματα μπορέσαμε να θέσουμε κάποιους βασικούς άξονες για να καθοδηγήσουμε τους ερευνητές σε μελλοντικές MSR μελέτες. Καθώς η χρήση των πινάκων διασταυρώσεων (βλ. Πίνακας 4.3, Πίνακας 4.5) για τη καθοδήγηση των ερευνητών δεν είναι ιδιαίτερα εύχρηστοι, αποφασίσαμε να αναπτύξουμε σχετικό εργαλείο/ιστοσελίδα όπου ο ερευνητής θα μπορεί να δώσει το πεδίο έρευνας που τον ενδιαφέρει και να κατατοπιστεί σχετικά με τα πιο συχνά κριτήρια επιλογής έργων, τα πιο συχνά έργα και να πάρει πρόσβαση σε πιθανά ανοιχτές συλλογές δεδομένων.

Θεωρούμε ότι η συγκεκριμένη έρευνα μπορεί να συμβάλει σημαντικά στη βελτίωση της ποιότητας των μελλοντικών εμπειρικών μελετών, να μειώσει τις απειλές της εγκυρότητας τους, και να αυξήσει τη συστηματική επαναληψιμότητα τους δίνοντας δυνατότητα μετα-ανάλυσης και σύνθεσης.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Ι. ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΑΝΑΦΟΡΕΣ

[1] Claes Wohlin, Martin Höst, Kennet Henningsson ' Empirical Research Methods in Web and Software Engineering ' Web Engineering pp 409-430, 2006 .

[2] Runeson, P., Höst, M. Guidelines for conducting and reporting case study research in software engineering. *Empir Software Eng* **14**, 131–164 (2009). https://doi.org/10.1007/s10664-008-9102-8 .

[3] Jones TL, Baxter MA, Khanduja V. A quick guide to survey research. Ann R Coll Surg Engl. 2013 Jan;95(1):5-7. doi: 10.1308/003588413X13511609956372. PMID: 23317709; PMCID: PMC3964639

[4] Stol, Klaas-Jan & Ali Babar, Muhammad & Russo, Barbara & Fitzgerald, Brian. (2009). The use of empirical methods in Open Source Software research: Facts, trends and future directions. Emerging Trends in FLOSS Research and Development, International Workshop on. 19-24. 10.1109/FLOSS.2009.5071355.

[5] Weber, T., Georgii, R., & Böni, P. (2016). Takin: An open-source software for experiment planning, visualisation, and data analysis. SoftwareX, 5, 121–126. doi:10.1016/j.softx.2016.06.002 .

[6] Gureckis, T.M., Martin, J., McDonnell, J. *et al.* psiTurk: An open-source framework for conducting replicable behavioral experiments online. *Behav Res* **48**, 829–842 (2016). https://doi.org/10.3758/s13428-015-0642-8.

[7] Paulson, James & Succi, Giancarlo & Eberlein, Armin. (2004). An empirical study of open-source and closed-source software products. Software Engineering, IEEE Transactions on. 30. 246 - 256. 10.1109/TSE.2004.1274044.

[8] Panagiotis Barlas Ivor Lanning Cathal Heavey , (2015),"A survey of open source data science tools", International Journal of Intelligent Computing and Cybernetics, Vol. 8 Iss 3 pp. 232 - 261 Permanent link to this document: http://dx.doi.org/10.1108/IJICC-07-2014-0031

[9] Rademacher, J.D.M., Lippke, S. Dynamic online surveys and experiments with the free open-source software*dynQuest . Behavior Research Methods* **39**, 415–426 (2007). https://doi.org/10.3758/BF03193011

[10] Mohamed Sarrab, Osama M. Hussain Rehman,Empirical study of open source software selection for adoption, based on software quality characteristics,Advances in Engineering Software,Volume 69,2014,Pages 1-11,ISSN 0965-9978,https://doi.org/10.1016/j.advengsoft.2013.12.001 .

[11] Tirole, Jean and Lerner, Josh, The Simple Economics of Open Source (March 2000). NBER Working Paper No. w7600, Available at SSRN: https://ssrn.com/abstract=214311

[12] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. 2014. The promises and perils of mining GitHub. In Proceedings of the 11th Working Conference on Mining Software Repositories (MSR 2014). Association for Computing Machinery, New York, NY, USA, 92–101. https://doi.org/10.1145/2597073.2597074

[13] Wahyu Nur Hidayat, Online Git Tutorial Application for Pair Programming Learning that Supports the Coworking Space Concept, Proceedings of the International Joint Conference on Science and Engineering 2021 (IJCSE 2021), Atlantis Press, doi:10.2991/aer.k.211215.132

[14] Morgan, L., Finnegan, P. (2007). Benefits and Drawbacks of Open Source Software: An Exploratory Study of Secondary Software Firms. In: Feller, J., Fitzgerald, B., Scacchi, W., Sillitti, A. (eds) Open Source Development, Adoption and Innovation. OSS 2007. IFIP — The International Federation for Information Processing, vol 234. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-72486-7_33

[15] Xiaozhou Li, Sergio Moreschini, Zheying Zhang, Davide Taibi,Exploring factors and metrics to select open source software components for integration: An empirical study,Journal of Systems and Software,Volume 188,2022,111255,ISSN 0164-1212, https://doi.org/10.1016/j.jss.2022.111255 .

[16] S. S. Gokhale, T. Smith and R. McCartney, "Integrating Open Source Software into software engineering curriculum: Challenges in selecting projects," 2012 First International Workshop on Software Engineering Education Based on Real-World Experiences (EduRex), Zurich, Switzerland, 2012, pp. 9-12, doi: 10.1109/EduRex.2012.6225697

[17] Simon Butler, Jonas Gamalielsson, Björn Lundell, Christoffer Brax, Anders Mattsson, Tomas Gustavsson, Jonas Feist, Bengt Kvarnström, Erik Lönroth,Considerations and challenges for the adoption of open source components in software-intensive businesses,Journal of Systems and Software,Volume 186,2022,111152,ISSN 0164-1212, https://doi.org/10.1016/j.jss.2021.111152 .

[18] Σπυράκης Παύλος, Ημερίδα σχετικά με «Ελεύθερο Λογισμικό / Λογισμικό Ανοιχτού Κώδικα: Στην Εκπαίδευση, στη Δημόσια Διοίκηση και στις Επιχειρήσεις. Η διεθνής εμπειρία και η ελληνική πραγματικότητα».

[19] Ρήγας Πολύκαρπος, Σιδέρης Νίκος (2008), Προγράμματα Ανοιχτού Κώδικα, Αθήνα.

[20] Bogue L. Robert, TechRepublic (2005), The problems with open source, Daily Newsletters, ZDNet UK's daily newsletter.

[21] Τόγιας Κωνσταντίνος, DAISy Group - Ερευνητικό Ακαδημαϊκό Ινστιτούτο Τεχνολογίας Υπολογιστών, Ελεύθερο Λογισμικό / Λογισμικό Ανοικτού Κώδικα

[22] Επιτήδειος Γιώργος (2002), Μια επιχειρηματική προσέγγιση του Open Source και του Free Software.

[23] Jadhav, A.S., Sonar, R.M., 2009. Evaluating and Selecting Software Packages: A review. Information and Software Technology, 51(3), 555-563.

[24] Scott, Stacey & Grant, Karen & Mandryk, Regan. (2003). System Guidelines for Co-located, Collaborative Work on a Tabletop Display. 159-178. 10.1007/978-94-010-0068-0_9 .

[25] L. A. Barba, "Defining the Role of Open Source Software in Research Reproducibility," in Computer, vol. 55, no. 8, pp. 40-48, Aug. 2022, doi: 10.1109/MC.2022.3177133.

[26] Wheeler, D. (2005) Why Open Source Software/Free Sofware (OSS/FL, FLOSS, or FOSS)? Look at the Numbers! http:/ /www.dwheelcr.com/ oss _fs _ why.html.

[27] Morgan, L., Finnegan, P. (2007). Benefits and Drawbacks of Open Source Software: An Exploratory Study of Secondary Software Firms. In: Feller, J., Fitzgerald, B., Scacchi, W., Sillitti, A. (eds) Open Source Development, Adoption and Innovation. OSS 2007. IFIP — The International Federation for Information Processing, vol 234. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-72486-7_33

[28] AlMarzouq, M., Zheng, L., Rong, G., & Grover, V. (2005). Open Source: Concepts, Benefits, and Challenges. Communications of the Association for Information Systems, 16, pp-pp. https://doi.org/10.17705/1CAIS.01637

[29] Succi, Giancarlo. (2003). Toward an Empirical Assessment of the Benefits of Open Source Software.

[30] Kagdi, H.H., Collard, M.L., & Maletic, J.I. (2007). A survey and taxonomy of approaches for mining software repositories in the context of software evolution. J. Softw. Maintenance Res. Pract., 19, 77-131.

[31] D'Angelo R. Barros, Daniel & Horita, Flavio & Wiese, Igor & Silva, Kanan. (2021). A Mining Software Repository Extended Cookbook: Lessons learned from a literature review. 1-10. 10.1145/3474624.3474627.

[32] Mário André de F. Farias, Renato Novais, Methanias Colaço Júnior, Luís Paulo da Silva Carvalho, Manoel Mendonça, and Rodrigo Oliveira Spínola. 2016. A systematic mapping study on mining software repositories. In Proceedings of the 31st Annual ACM Symposium on Applied Computing (SAC '16). Association for Computing Machinery, New York, NY, USA, 1472–1479. https://doi.org/10.1145/2851613.2851786

[33] Shari Lawrence Pfleeger and Barbara A. Kitchenham. 2001. Principles of survey research: part 1: turning lemons into lemonade. SIGSOFT Softw. Eng. Notes 26, 6 (November 2001), 16–18. https://doi.org/10.1145/505532.505535

[34] Runeson, P.,Host, M., Rainer, A., Regnell, B. ,*Case Study Research in Software Engineering: Guidelines and Examples*,Wiley,2012. isbn: 9781118181003

[35] Kai Petersen, Robert Feldt and Shahid Mujtaba et al. Systematic Mapping Studies in Software Engineering. 2008. DOI: 10.14236/ewic/EASE2008.8

[36] Claes Wohlin. 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (EASE '14). Association for Computing Machinery, New York, NY, USA, Article 38, 1–10. https://doi.org/10.1145/2601248.2601268

[37] Basili, V.R., Caldiera, G., Rombach, H.D., 1994. Goal Question Metric Paradigm, Encyclopedia of Software Engineering. John Wiley & Sons, pp. 528–532.

[38] W. Eric Wong, Nikolaos Mittas, Elvira Maria Arvanitou, Yihao Li, A bibliometric assessment of software engineering themes, scholars and institutions (2013–2020), Journal of Systems and Software, Volume 180, 2021, 111029, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2021.111029.

[39] Farhoodi, R., Garousi, V., Pfahl, D. and Sillito, J., "Development of Scientific Software: A Systematic Mapping, a Bibliometrics Study, and a Paper Repository", International Journal of Software Engineering and Knowledge Engineering, 23 (4), 2013.

[40] D. S. Cruzes and T. Dybå, "Research synthesis in software engineering: A tertiary study", Information and Software Technology, 53(5), pp. 440-455, 2011.

[41] P. Santos and G. H. Travassos, “Research Synthesis in Software Engineering”, Contemporary Empirical Methods in Software Engineering, pp. 443-474, 2020

[42] B. Kitchenham, L. Madeyski, and P. Brereton, “Meta-analysis for families of experiments in software engineering: a systematic review and reproducibility and validity assessment”, Empirical Software Engineering, 25, pp. 353–401, 2020

[43] B. Kitchenham, T. Dyba, and M. Jorgensen, “Evidence-based software engineering”, 26th International Conference on Software Engineering, Edinburgh, UK, 28 May 2004.

[44] D. Spencer, “Card Sorting: Designing Usable Categories”, Rosenfeld Media, April 2009.

[45] Kitchenham, B., Brereton, P., Turner, M., Niazi, M., Linkman, S., Pretorius, R., Budgen, D., 2009b. The impact of limited search procedures for systematic literature reviews a participant-observer case study. In: 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM’09). IEEE Computer Society, USA.

[46] Sommerville, I., *Software Engineering*, Addison-Wesley, 2007, isbn: 9780321313799

[47] Chaffer, J., *Learning jQuery - Fourth Edition*, Packt Pub., 2013, isbn: 9781782163152

[48] McFarland, D.S., *JavaScript & JQuery: The Missing Manual*, O'Reilly Media, 2011, isbn: 9781449320461

[49] Boduch, A.,Derks, R., React and React Native: A complete hands-on guide to modern web and mobile development with React.js, Packt Publishing, 2020 ISBN 9781839212437

[50] Epperly, Thomas & Kumfert, Gary & Dahlgren, Tamara & Ebner, Dietmar & Leek, Jim & Prantl, Adrian & Kohn, Scott. (2012). High-Performance language interoperability for scientific computing through Babel. International Journal of High Performance Computing Applications - IJHPCA. 26. 260-274. 10.1177/1094342011414036.

[51] Khriyenko, O. Semantic UI: Automated Creation of Semantically Personalized User Interface. GSTF J Comput 4, 16 (2015). https://doi.org/10.7603/s40601-014-0016-6

[52] Edwards, T., “Documentation, SweerAlert”, No date. [Online]. Available at: https://sweetalert.js.org/

[53] H. van Vliet, "Software Engineering: Principles and Practice (3rd Edition)",Wiley & Sons, Chichester,England, 1993.)

[54] The Apache Software Foundation , “Apache Camel”, No date. [Online]. Available at: https://camel.apache.org/

[55] The Apache Software Foundation, ”Commons Lang”, No date. [Online]. Available at: https://commons.apache.org/proper/commons-lang/

[56] The Apache Software Foundation, ”Commons Math: The Apache Commons Mathematics Library”, No date. [Online]. Available at: https://commons.apache.org/proper/commons-math/

[57] The Apache Software Foundation, ”Apache Ant”, No date. [Online]. Available at: https://ant.apache.org/

[58] The Apache Software Foundation, ”Apache Lucene”, No date. [Online]. Available at: https://lucene.apache.org/

[59] The Apache Software Foundation, ”Apache Log4j”, No date. [Online]. Available at: https://logging.apache.org/log4j/2.x/

[60] jEdit developers, “jEdit”, 2020. [Online]. Available at: http://www.jedit.org/

[61] David Gilber, “JFreeChart”, No date. [Online]. Available at: https://www.jfree.org/jfreechart/

[62] VMware, Inc. “Spring Framework”, No date. [Online]. Available at: https://spring.io/projects/spring-framework

[63] The Apache Software Foundation,”Apache HBase”, No date. [Online]. Available at: https://hbase.apache.org/

II. ΑΝΑΦΟΡΕΣ ΤΗΣ ΣΥΣΤΗΜΑΤΙΚΗΣ ΜΕΛΕΤΗΣ ΧΑΡΤΟΓΡΑΦΗΣΗΣ

[PS1] Aatira Anum Ahmad, Abdul Rafae Noor, Hashim Sharif, Usama Hameed, Shoaib Asif, Mubashir Anwar, Ashish Gehani, Fareed Zaffar, Junaid Haroon Siddiqui ,"Trimmer: An Automated System for Configuration-Based Software Debloating. ", IEEE Transactions on Software Engineering, 48 (09) pp, 3485-3505,2022

[PS2] Aayush Garg, Renzo Degiovanni, Matthieu Jimenez, Maxime Cordy, Mike Papadakis, Yves Le Traon ,"Learning from what we know: How to perform vulnerability prediction using noisy historical data" , Empirical Software Engineering , 27 (07) pp. 169 ,2022

[PS3] Abdul Razzaq, Anthony Ventresque, Rainer Koschke, Andrea De Lucia, Jim Buckley , "The Effect of Feature Characteristics on the Performance of Feature Location Techniques " , IEEE Transactions on Software Engineering, 48 (06) pp. 2066-2085 ,2022

[PS4] Abu Naser Masud ,"Efficient computation of minimal weak and strong control closure", Journal of Systems and Software, 184 (02) pp. 111140 ,2022

[PS5] Afsoon Afzal, Manish Motwani, Kathryn T. Stolee, Yuriy Brun, Claire Le Goues, "SOSRepair: Expressive Semantic Search for Real-World Program Repair", IEEE Transactions on Software Engineering, 47 (10) pp.2162-2181 ,2021

[PS6] Ahmed Zerouali, Tom Mens, Alexandre Decan, Coen De Roover,"On the impact of security vulnerabilities in the npm and RubyGems dependency networks.", Empirical Software Engineering , 27 (05) pp.107 ,2022

[PS7] Aleem Khalid Alvi, Mohammad Zulkernine, "A security pattern detection framework for building more secure software" , Journal of Systems and Software, 171 (01) pp. 110838 ,2021

[PS8] Alexandre Decan, Tom Mens , "What Do Package Dependencies Tell Us About Semantic Versioning?" , IEEE Transactions on Software Engineering, 47 (06) pp.1226-1240 ,2021

[PS9] Alexandre Perez, Rui Abreu, Arie van Deursen , "A Theoretical and Empirical Analysis of Program Spectra Diagnosability." , IEEE Transactions on Software Engineering, 47 (02) pp,412-431 ,2021

[PS10] Alexi Turcotte, Ellen Arteca, Ashish Mishra, Saba Alimadadi, Frank Tip, "Stubbifier: debloating dynamic server-side JavaScript applications" , Empirical Software Engineering , 27 (07) pp. 161 ,2022

[PS11] Aline Brito, Andre Hora, and Marco Tulio Valente, Characterizing refactoring graphs in Java and JavaScript projects, Empirical Software Engineering, 26(6), 2021

[PS12] Alireza Aghamohammadi, Seyed-Hassan Mirian-Hosseinabadi, and Sajad Jalali, "Statement frequency coverage: A code coverage criterion for assessing test suite effectiveness", Information and Software Technology, Volume 129, 2021

[PS13] Alvin Jian Jia Tan, Chun Yong Chong, Aldeida Aleti ,"E-SC4R: Explaining Software Clustering for Remodularisation", Journal of Systems and Software, 186 (04) pp. 111162 ,2022

[PS14] Amine Barrak, Ellis E. Eghan, Bram Adams, Foutse Khomh, "Why do builds fail? - A conceptual replication study" , Journal of Systems and Software, 177 (07) pp. 110939 ,2021

[PS15] Amjad AbuHassan, Mohammad Alshayeb, and Lahouari Ghouti, Prioritization of model smell refactoring using a covariance matrix-based adaptive evolution algorithm, Information and Software Technology, 146,2022

[PS16] Amjed Tahir, Kwabena E. Bennin, Xun Xiao, and Stephen G. MacDonell, Does class size matter? An in-depth assessment of the effect of class size in software defect prediction, Empirical Software Engineering, 26(5), 2021

[PS17] Amritanshu Agrawal, Xueqi Yang, Rishabh Agrawal, Rahul Yedida, Xipeng Shen, Tim Menzies ,"Simpler Hyperparameter Optimization for Software Analytics: Why, How, When? ", IEEE Transactions on Software Engineering, 48 (08) pp, 2939-2954 ,2022

[PS18] An Ran Chen, Tse-Hsun (Peter) Chen, and Shaowei Wang, Demystifying the challenges and benefits of analyzing user-reported logs in bug reports, Empirical Software Engineering, 26 (1), 2021

[PS19] An Ran Chen, Tse-Hsun Chen, Shaowei Wang ,"Pathidea: Improving Information Retrieval-Based Bug Localization by Re-Constructing Execution Paths Using Logs " , IEEE Transactions on Software Engineering, 48 (08) pp,2905-2919 ,2022

[PS20] André de S. Landi, Daniel San Martín, Bruno Marinho Santos, Warteruzannan Soyer Cunha, Rafael S. Durelli, Valter Vieira de Camargo,"Architectural conformance checking for KDM-represented systems." , 183 (01) pp. 111116 ,2022

[PS21] Andrea Arcuri and Juan P. Galeotti. 2021. Enhancing Search-based Testing with Testability Transformations for Existing APIs. ACM Trans. Softw. Eng. Methodol.

[PS22] Andrea Di Sorbo, and Sebastiano Panichella, Exposed! A case study on the vulnerability-proneness of Google Play Apps, Empirical Software Engineering, 26(4), 2021

[PS23] Andrea Romdhana, Alessio Merlo, Mariano Ceccato, and Paolo Tonella. 2022. Deep Reinforcement Learning for Black-box Testing of Android Apps. ACM Trans. Softw. Eng. Methodol.

[PS24] Andreas Dann, Henrik Plate, Ben Hermann, Serena Elisa Ponta, Eric Bodden, "Identifying Challenges for OSS Vulnerability Scanners - A Study & Test Suite " , IEEE Transactions on Software Engineering, 48 (09) pp, 3613-3625 ,2022

[PS25] Ankur Tagra, Haoxiang Zhang, Gopi Krishnan Rajbahadur, Ahmed E. Hassan ,"Revisiting reopened bugs in open source software systems.", Empirical Software Engineering , 27 (04) pp. 92 ,2022

[PS26] Annibale Panichella, "A Systematic Comparison of search-Based approaches for LDA hyperparameter tuning", Information and Software Technology, Volume 130, 2021

[PS27] Annibale Panichella, Sebastiano Panichella, Gordon Fraser, Anand Ashok Sawant, Vincent J. Hellendoorn ,"Test smells 20 years later: detectability, validity, and reliability.",Empirical Software Engineering , 27 (07) pp. 170 ,2022

[PS28] Antonia Bertolino, Breno Miranda, Roberto Pietrantuono, Stefano Russo ,"Adaptive Test Case Allocation, Selection and Generation Using Coverage Spectrum and Operational Profile.", IEEE Transactions on Software Engineering, 47 (05) pp. 881-898 ,2021

[PS29] Antonios Gkortzis, Daniel Feitosa, Diomidis Spinellis , "Software reuse cuts both ways: An empirical analysis of its relationship with security vulnerabilities" , Journal of Systems and Software, 172 (02) pp. 110653 ,2021

[PS30] Aoi Takahashi, Natthawute Sae-Lim, Shinpei Hayashi, Motoshi Saeki, "An extensive study on smell-aware bug localization" , Journal of Systems and Software, 178 (08) pp. 110986 ,2021

[PS31] Arianna Blasi, Alessandra Gorla, Michael D. Ernst, Mauro Pezzè, Antonio Carzaniga,"MeMo: Automatically identifying metamorphic relations in Javadoc comments for test automation" ,Journal of Systems and Software, 181 (11) pp. 111041 ,2021

[PS32] Arianna Blasi, Nataliia Stulova, Alessandra Gorla, Oscar Nierstrasz,"RepliComment: Identifying clones in code comments" ,Journal of Systems and Software, 182 (12) pp. 111069 ,2021

[PS33] Arif Nurwidyantoro, Mojtaba Shahin, Michel R.V. Chaudron, Waqar Hussain, Rifat Shams, Harsha Perera, Gillian Oliver, and Jon Whittle, Human values in software development

artefacts: A case study on issue discussions in three Android applications, Information and Software Technology,141,2022

[PS34] Arthur Kamienski, and Cor-Paul Bezemer, An empirical study of Q&A websites for game developers, Empirical Software Engineering, 26(6), 2021

[PS35] Arthur Vitui, and Tse-Hsun (Peter) Chen, MLASP: Machine learning assisted capacity planning, Empirical Software Engineering, 26(5), 2021

[PS36] Baicai Sun, Dunwei Gong, Tian Tian, Xiangjuan Yao , "Integrating an Ensemble Surrogate Model's Estimation into Test Data Generation. " , IEEE Transactions on Software Engineering, 48 (04) pp.1336-1350 ,2022

[PS37] Bailey Vandehei, Daniel Alencar Da Costa, and Davide Falessi. 2021. Leveraging the Defects Life Cycle to Label Affected Versions and Defective Classes. ACM Trans. Softw. Eng. Methodol.

[PS38] Béla Vancsics, Ferenc Horváth, Attila Szatmári, Árpád Beszédes" ,Fault localization using function call frequencies." , Journal of Systems and Software, 193 (11) pp. 111429 ,2022

[PS39] Benjamin Loriot, Fernanda Madeiral, Martin Monperrus, "Styler: learning formatting conventions to repair Checkstyle violations.", Empirical Software Engineering , 27 (06) pp. 149 ,2022

[PS40] Beyza Eken, Ayse Tosun, "Investigating the performance of personalized models for software defect prediction" , Journal of Systems and Software, 181 (11) pp. 111038 ,2021

[PS41] Biruk Asmare Muse, Csaba Nagy, Anthony Cleve, Foutse Khomh, Giuliano Antoniol ,"FIXME: synchronize with database! An empirical study of data access self-admitted technical debt." , Empirical Software Engineering , 27 (06) pp. 130 ,2022

[PS42] Bo Lin, Shangwen Wang, Ming Wen, and Xiaoguang Mao. 2022. Context-Aware Code Change Embedding for Better Patch Correctness Assessment. ACM Trans. Softw. Eng. Methodol.

[PS43] Bodin Chinthanet, Raula Gaikovina Kula, Shane McIntosh, Takashi Ishio, Akinori Ihara, and Kenichi Matsumoto, Lags in the release, adoption, and propagation of npm vulnerability fixes, Empirical Software Engineering, 26(3), 2021

[PS44] Boqin Qin, Tengfei Tu, Ziheng Liu, Tingting Yu, Linhai Song , "Algorithmic Profiling for Real-World Complexity Problems " , IEEE Transactions on Software Engineering, 48 (07) pp, 2680-2694 ,2022

[PS45] Brent van Bladel, Serge Demeyer ,"A comparative study of test code clones and production code clones" , Journal of Systems and Software, 176 (06) pp. 110940 ,2021

[PS46] Camilo Escobar-Velásquez, Alejandro Mazuera-Rozo, Claudia Bedoya, Michael Osorio-Riaño, Mario Linares-Vásquez, Gabriele Bavota ,"Studying eventual connectivity issues in Android apps" , Empirical Software Engineering , 27 (01) pp. 22 ,2022

[PS47] Camilo Escobar-Velásquez, Mario Linares-Vásquez, Gabriele Bavota, Michele Tufano, Kevin Moran, Massimiliano Di Penta, Christopher Vendome, Carlos Bernal-Cárdenas, Denys Poshyvanyk, "Enabling Mutant Generation for Open- and Closed-Source Android Apps. ", IEEE Transactions on Software Engineering, 48 (02) pp.186-208 ,2022

[PS48] Catarina Costa, Jair Figueiredo, João Felipe Pimentel, Anita Sarma, Leonardo Murta ,"Recommending Participants for Collaborative Merge Sessions" , IEEE Transactions on Software Engineering, 47 (06) pp. 1198-1210 ,2021

[PS49] César Soto-Valero, Nicolas Harrand, Martin Monperrus, and Benoit Baudry, A comprehensive study of bloated dependencies in the Maven ecosystem, Empirical Software Engineering, 26(3), 2021

[PS50] Cezar Sas, Andrea Capiluppi ,”Antipatterns in software classification taxonomies” , Journal of Systems and Software, 190 (08) pp. 111343 ,2022

[PS51] Chaima Abid, Marouane Kessentini, Vahid Alizadeh, Mouna Dhaouadi, Rick Kazman , “How Does Refactoring Impact Security When Improving Quality? A Security-Aware Refactoring Approach “ , IEEE Transactions on Software Engineering, 48 (03) pp. 864-878 ,2022

[PS52] Chang-Ai Sun, Baoli Liu, An Fu, Yiqiang Liu, Huai Liu ,”Path-directed source test case generation and prioritization in metamorphic testing” , Journal of Systems and Software, 183 (01) pp. 111091 ,2022

[PS53] Changqing Wei, Xiangjuan Yao, Dunwei Gong,and Huai Liu, Spectral clustering based mutant reduction for mutation testing, Information and Software Technology, Volume 132, 2021

[PS54] Chao Liu, Xin Xia, David Lo, Zhiwe Liu, Ahmed E. Hassan, and Shanping Li. 2021. CodeMatcher: Searching Code Based on Sequential Semantics of Important Query Words. ACM Trans. Softw. Eng. Methodol.

[PS55] Chao Ni, Xin Xia, David Lo, Xiang Chen, Qing Gu , “Revisiting Supervised and Unsupervised Methods for Effort-Aware Cross-Project Defect Prediction “, IEEE Transactions on Software Engineering, 48 (03) pp. 786-802,2022

[PS56] Chi Chen, Xin Peng, Bihuan Chen, Jun Sun, Zhenchang Xing, Xin Wang, Wenyun Zhao, “"More Than Deep Learning": post-processing for API sequence recommendation.” , Empirical Software Engineering , 27 (01) pp. 15 ,2022

[PS57] Christoph Laaber, Harald C. Gall, and Philipp Leitner, Applying test case prioritization to software microbenchmarks, Empirical Software Engineering, 26(6), 2021

[PS58] Christophe Rezk, Yasutaka Kamei, Shane McIntosh,”The Ghost Commit Problem When Identifying Fix-Inducing Changes: An Empirical Study of Apache Projects “, IEEE Transactions on Software Engineering, 48 (09) pp, 3297-3309,2022

[PS59] Chunying Zhou, Peng He, Cheng Zeng, and Ju Ma, Software defect prediction with semantic and structural information of codes based on Graph Neural Networks, Information and Software Technology, 152,2022

[PS60] Cuiyun Gao, Jichuan Zeng, Federica Sarro, David Lo, Irwin King, Michael and R. Lyu, Do users care about ad’s performance costs? Exploring the effects of the performance costs of in-app ads on user experience, Information and Software Technology, Volume 132, 2021

[PS61] Daming Zou, Jingjing Liang, Yingfei Xiong, Michael D. Ernst, Lu Zhang , “An Empirical Study of Fault Localization Families and Their Combinations” , IEEE Transactions on Software Engineering, 47 (02) pp, 332-347 ,2021

[PS62] Danilo Dominguez Perez, Wei Le ,”Specifying Callback Control Flow of Mobile Apps Using Finite Automata.” , IEEE Transactions on Software Engineering, 47 (02) pp, 379-392 ,2021

[PS63] Danilo Silva, João Paulo da Silva, Gustavo Jansen de Souza Santos, Ricardo Terra, Marco Túlio Valente ,”RefDiff 2.0: A Multi-Language Refactoring Detection Tool.” , IEEE Transactions on Software Engineering, 47 (12) pp. 2786-2802 ,2021

[PS64] Daoyuan Wu, Debin Gao, and David Lo, Scalable online vetting of Android apps for measuring declared SDK versions and their consistency with API calls, Empirical Software Engineering, 26(1), 2021

[PS65] David Binkley, Leon Moonen, and Sibren Isaacman, Featherweight assisted vulnerability discovery, Information and Software Technology, 146,2022

[PS66] Davide Falessi, Aalok Ahluwalia, and Massimiliano DI Penta. 2021. The Impact of Dormant Defects on Defect Prediction: A Study of 19 Apache Projects. ACM Trans. Softw. Eng. Methodol.

[PS67] Dayi Lin, Chakkrit Tantithamthavorn, Ahmed E. Hassan ,"The Impact of Data Merging on the Interpretation of Cross-Project Just-In-Time Defect Models " , IEEE Transactions on Software Engineering, 48 (08) pp, 2969-2986,2022

[PS68] Debolina Ghosh,and Jagannath Singh, Spectrum-based multi-fault localization using Chaotic Genetic Algorithm, Information and Software Technology, Volume 133, 2021

[PS69] Devika Sondhi, Mayank Jobanputra, Divya Rani, Salil Purandare, Sakshi Sharma, Rahul Purandare ,"Mining Similar Methods for Test Adaptation. " , IEEE Transactions on Software Engineering, 48 (07) pp. 2262-2276 ,2022

[PS70] Dhia Elhaq Rzig, Foyzul Hassan, and Marouane Kessentini, An empirical study on ML DevOps adoption trends, efforts, and benefits analysis, Information and Software Technology,152,2022

[PS71] Diego Costa, Cor-Paul Bezemer, Philipp Leitner, Artur Andrzejak ,"What's Wrong with My Benchmark Results? Studying Bad Practices in JMH Benchmarks." , IEEE Transactions on Software Engineering, 47 (07) pp. 1452-1467 ,2021

[PS72] Diego Elias Costa, Suhaib Mujahid, Rabe Abdalkareem, Emad Shihab ,"Breaking Type Safety in Go: An Empirical Study on the Usage of the unsafe Package " , IEEE Transactions on Software Engineering, 48 (07) pp, 2277-2294 ,2022

[PS73] Dilini Rajapaksha, Chakkrit Tantithamthavorn, Jirayus Jiarpakdee, Christoph Bergmeir, John Grundy, Wray L. Buntine,"SQAPlanner: Generating Data-Informed Software Quality Improvement Plans. " , IEEE Transactions on Software Engineering, 48 (08) pp,2814-2835,2022

[PS74] Diomidis Spinellis, Paris Avgeriou, "Evolution of the Unix System Architecture: An Exploratory Case Study.", IEEE Transactions on Software Engineering, 47 (06) pp. 1134-1163 ,2021

[PS75] Dipesh Pradhan, Shuai Wang, Shaukat Ali, Tao Yue, Marius Liaaen ,"CBGA-ES+: A Cluster-Based Genetic Algorithm with Non-Dominated Elitist Selection for Supporting Multi-Objective Test Optimization", IEEE Transactions on Software Engineering, 47 (01) pp, 86-107 ,2021

[PS76] Dong Wang, Tao Xiao, Patanamon Thongtanunam, Raula Gaikovina Kula, and Kenichi Matsumoto, Understanding shared links and their intentions to meet information needs in modern code review, Empirical Software Engineering, 26(5), 2021

[PS77] Dong Jae Kim, Tse-Hsun (Peter) Chen, and Jinqiu Yang, The secret life of test smells - an empirical study on test smell evolution and maintenance, Empirical Software Engineering, 26(5), 2021

[PS78] Dong Wang, Raula Gaikovina Kula, Takashi Ishio, and Kenichi Matsumoto, Automatic patch linkage detection in code review using textual content and file location features, Information and Software Technology,139,2021

[PS79] Donghoon Jeon, Minseok Jeon,and Hakjoo Oh, A practical algorithm for learning disjunctive abstraction heuristics in static program analysis, Information and Software Technology, Volume 135, 2021

[PS80] Donghwan Shin, Domenico Bianculli, Lionel C. Briand , "PRINS: scalable model inference for component-based system logs", Empirical Software Engineering , 27 (04) pp. 87 ,2022

[PS81] Dongliang Mu, Yunlan Du, Jianhao Xu, Jun Xu, Xinyu Xing, Bing Mao, Peng Liu ,"POMP++: Facilitating Postmortem Program Diagnosis with Value-Set Analysis." , IEEE Transactions on Software Engineering, 47 (09) pp. 1929-1942,2021

[PS82] Eliane Maria De Bortoli Fávero, Dalcimar Casanova, and Andrey Ricardo Pimentel, SE3M: A model for software effort estimation using pre-trained embedding models, Information and Software Technology,147,2022

[PS83] Elijah Zolduoarrati, and Sherlock A. Licorish, On the value of encouraging gender tolerance and inclusiveness in software engineering communities, Information and Software Technology,139,2021

[PS84] Eman Abdullah AlOmar, Mohamed Wiem Mkaouer, Ali Ouni ,"Toward the automatic classification of Self-Affirmed Refactoring" , Journal of Systems and Software, 171 (01) pp. 110821 ,2021

[PS85] Emre Doğan, and Eray Tüzün, Towards a taxonomy of code review smells, Information and Software Technology,142,2022

[PS86] Emre Sülün, Eray Tüzün,and Uğur Doğrusöz, "RSTrace+: Reviewer suggestion using software artifact traceability graphs", Information and Software Technology, Volume 130, 2021

[PS87] Enrico Fregnan, Fernando Petrulio, Alberto Bacchelli ,"The evolution of the code during review: an investigation on review changes" , Empirical Software Engineering , 27 (07) pp. 177 ,2022

[PS88] Ezekiel O. Soremekun, Esteban Pavese, Nikolas Havrikov, Lars Grunske, Andreas Zeller , "Inputs From Hell" , IEEE Transactions on Software Engineering, 48 (04) pp. 1138-1153,2022

[PS89] Ezekiel Soremekun, Lukas Kirschner, Marcel Böhme, and Andreas Zeller, Locating faults with program slicing: an empirical analysis, Empirical Software Engineering, 26(3), 2021

[PS90] Fabiano Pecorelli, Fabio Palomba, and Andrea De Lucia, The Relation of Test-Related Factors to Software Quality: A Case Study on Apache Systems,Empirical Software Engineering,26(2), 2021

[PS91] Fabiano Pecorelli, Savanna Lujan, Valentina Lenarduzzi, Fabio Palomba, Andrea De Lucia, "On the adequacy of static analysis warnings with respect to code smell prediction." , Empirical Software Engineering , 27 (03) pp. 64 ,2022

[PS92] Fábio de Almeida Farzat, Márcio de Oliveira Barros, Guilherme H. Travassos ."Evolving JavaScript Code to Reduce Load Time" , IEEE Transactions on Software Engineering, 47 (08) pp. 1544-1558 ,2021

[PS93] Fabio Palomba, Damian Andrew Tamburri ,"Predicting the emergence of community smells using socio-technical metrics: A machine-learning approach" ,Journal of Systems and Software, 171 (01) pp. 110847 ,2021

[PS94] Fabio Palomba, Damian Andrew Tamburri, Francesca Arcelli Fontana, Rocco Oliveto, Andy Zaidman, Alexander Serebrenik, "Beyond Technical Aspects: How Do Community Smells Influence the Intensity of Code Smells?", IEEE Transactions on Software Engineering, 47 (01) pp, 108-129 ,2021

[PS95] Fabrizio Pastore, Daniela Micucci, Michell Guzmán, Leonardo Mariani,"TkT: Automatic Inference of Timed and Extended Pushdown Automata. ",IEEE Transactions on Software Engineering, 48 (02) pp. 617-636 ,2022

[PS96] Fanlong Zhang,and Siau-cheng Khoo, An empirical study on clone consistency prediction based on machine learning, Information and Software Technology,136,2021

[PS97] Faria Huq, Masum Hasan, Md Mahim Anjum Haque, Sazan Mahbub, Anindya Iqbal, and Toufique Ahmed, Review4Repair: Code review aided automatic program repairing, Information and Software Technology, 143,2022

[PS98] Ferenc Horváth, Árpád Beszédes, Béla Vancsics, Gergõ Balogh, László Vidács, Tibor Gyimóthy , "Using contextual knowledge in interactive fault localization" , Empirical Software Engineering , 27 (06) pp.150 ,2022

[PS99] Filipe R. Cogo, Gustavo A. Oliva, Cor-Paul Bezemer, and Ahmed E. Hassan, An empirical study of same-day releases of popular packages in the npm ecosystem, Empirical Software Engineering, 26(5), 2021

[PS100] Fiorella Zampetti, Saghan Mudbhari, Venera Arnaoudova, Massimiliano Di Penta, Sebastiano Panichella, Giuliano Antoniol, “Using code reviews to automatically configure static analysis tools.”, Empirical Software Engineering , 27 (01) pp. 28 ,2022

[PS101] Flavio Corradini, Fabrizio Fornari, Andrea Polini, Barbara Re, Francesco Tiezzi, Andrea Vandin, “A formal approach for the analysis of BPMN collaboration models.”, Journal of Systems and Software, 180 (10) pp. 111007 ,2021

[PS102] Francesco Lomio, Emanuele Iannone, Andrea De Lucia, Fabio Palomba, Valentina Lenarduzzi, “Just-in-time software vulnerability detection: Are we there yet”, Journal of Systems and Software, 188 (06) pp. 111283 ,2022

[PS103] Francesco Lomio, Sergio Moreschini, Valentina Lenarduzzi ,”A machine and deep learning analysis among SonarQube rules, product, and process metrics for fault prediction.” , Empirical Software Engineering , 27 (07) pp. 189 ,2022

[PS104] Francis Palma, Tobias Olsson, Anna Wingkvist, Javier Gonzalez-Huerta , “Assessing the linguistic quality of REST APIs for IoT applications” ,Journal of Systems and Software, 191 (09) pp. 111369 ,2022

[PS105] Francisco Handrick da Costa, Ismael Medeiros, Thales Menezes, João Victor da Silva, Ingrid Lorraine da Silva, Rodrigo Bonifácio, Krishna Narasimhan, Márcio Ribeiro, “Exploring the use of static and dynamic analysis to improve the performance of the mining sandbox approach for android malware identification” , Journal of Systems and Software, 183 (01) pp. 111092 ,2022

[PS106] Frolin S. Ocariza Jr. ,”On the Effectiveness of Bisection in Performance Regression Localization.”, Empirical Software Engineering , 27 (04) pp. 95 ,2022

[PS107] Gabriela Karoline Michelon, David Obermann, Wesley K. G. Assunção, Lukas Linsbauer, Paul Grünbacher, Stefan Fischer, Roberto E. Lopez-Herrejon, Alexander Egyed, “Evolving software system families in space and time with feature revisions”, Empirical Software Engineering , 27 (05) pp. 112 ,2022

[PS108] Gede Artha Azriadi Prana, Abhishek Sharma, Lwin Khin AU - Shar, Darius Foo, Andrew E. Santosa, AsankhayaSharma, and David Lo, Out of sight, out of mind? How vulnerable dependencies affect open-source projects, Empirical Software Engineering, 26(4), 2021

[PS109] George Digkas, Alexander Chatzigeorgiou, Apostolos Ampatzoglou, Paris Avgeriou , “Can Clean New Code Reduce Technical Debt Density? “ , IEEE Transactions on Software Engineering, 48 (05) pp. 1705-1721 ,2022

[PS110] Gerardo Canfora, Andrea Di Sorbo, Sara Forootani, Matias Martinez, and Corrado A. Visaggio, Patchworking: Exploring the code changes induced by vulnerability fixing activities, Information and Software Technology, 142,2022

[PS111] Giovani Guizzo, Federica Sarro, Jens Krinke, Silvia R. Vergilio ,”Sentinel: A Hyper-Heuristic for the Generation of Mutant Reduction Strategies “ ,IEEE Transactions on Software Engineering, 48 (03) pp. 803-818 ,2022

[PS112] Giovanni Grano, Christoph Laaber, Annibale Panichella, Sebastiano Panichella, “Testing with Fewer Resources: An Adaptive Approach to Performance-Aware Test Case Generation”, IEEE Transactions on Software Engineering, 47 (11) pp. 2332-2347,2021

[PS113] Giovanni Grano, Fabio Palomba, Harald C. Gall , “Lightweight Assessment of Test-Case Effectiveness Using Source-Code-Quality Indicators.” ,IEEE Transactions on Software Engineering, 47 (04) pp. 758-774 ,2021

[PS114] Giovanni Liva, Muhammad Taimoor Khan, Martin Pinzger, Francesco Spegni, Luca Spalazzi ,"Automatic Repair of Timestamp Comparisons.", IEEE Transactions on Software Engineering, 47 (11) pp.2369-2381 ,2021

[PS115] Gopi Krishnan Rajbahadur, Shaowei Wang, Gustavo Ansaldi Oliva, Yasutaka Kamei, Ahmed E. Hassan, "The Impact of Feature Importance Methods on the Interpretation of Defect Classifiers. " , IEEE Transactions on Software Engineering, 48 (07) pp. 2245-2261 ,2022

[PS116] Goran Piskachev, Johannes Späth, Ingo Budde, Eric Bodden ,"Fluently specifying taint-flow queries with fluentTQL", Empirical Software Engineering , 27 (05) pp. 104 ,2022

[PS117] Guanhua Wang, Sudipta Chattopadhyay, Ivan Gotovchits, Tulika Mitra, Abhik Roychoudhury , "oo7: Low-Overhead Defense Against Spectre Attacks via Program Analysis." , IEEE Transactions on Software Engineering, 47 (11) pp. 2504-2519 ,2021

[PS118] Guoli Cheng, Shi Ying, Bingming Wang, "Tuning configuration of apache spark on public clouds by combining multi-objective optimization and performance prediction model ", Journal of Systems and Software, 180 (10) pp. 111028 ,2021

[PS119] Gustav Bergstr, Arif Nurwidyantoro, Michel R. V. Chaudronöm, Fadhl Hujainah, Truong Ho-Quang, Rodi Jolak, Satrio Adi Rukmono ",Evaluating the layout quality of UML class diagrams using machine learning.", Journal of Systems and Software, 192 (10) pp. 111413 ,2022

[PS120] Ha Thanh Le, Lwin Khin Shar, Domenico Bianculli, Lionel Claude Briand, Cu Duy Nguyen ,"Automated reverse engineering of role-based access control policies of web applications", Journal of Systems and Software, 184 (02) pp. 111109 ,2022

[PS121] Hadhemi Jebnoun, Md. Saidur Rahman, Foutse Khomh, Biruk Asmare Muse, "Clones in deep learning code: what, where, and why?", Empirical Software Engineering , 27 (04) pp. 84 ,2022

[PS122] Hadi Jahanshahi, Mucahit Cevik, José Navas-Sú, Ayse Basar, Antonio González Torres ,"Wayback Machine: A tool to capture the evolutionary behavior of the bug reports and their triage process in open-source software systems" ,Journal of Systems and Software, 189 (07) pp. 111308 ,2022

[PS123] Haijun Wang, Yun Lin, Zijiang Yang, Jun Sun, Yang Liu, Jin Song Dong, Qinghua Zheng, Ting Liu ,"Explaining Regressions via Alignment Slicing and Mending". IEEE Transactions on Software Engineering, 47 (11) pp. 2421-2437 ,2021

[PS124] Hamid Bagheri, Jianghao Wang, Jarod Aerts, Negar Ghorbani, and Sam Malek, Flair: efficient analysis of Android inter-component vulnerabilities in response to incremental changes, Empirical Software Engineering, 26(3), 2021

[PS125] Hamidreza Ahmadi, Mehrdad Ashtiani, Mohammad Abdollahi Azgomi, and Raana Saheb-Nassagh, A DQN-based agent for automatic software refactoring, Information and Software Technology,147,2022

[PS126] Hamzeh Eyal Salman, Feature-based insight for forks in social coding platforms, Information and Software Technology,140,2021

[PS127] Hanefi Mercan, Atakan Aytar, Giray Coskun, Dilara Müstecep, Gülsüm Uzer, Cemal Yilmaz ,"CIT-daily: A combinatorial interaction testing-based daily build process",Journal of Systems and Software, 190 (08) pp. 111353 ,2022

[PS128] Hanyu Pei, Beibei Yin, Min Xie,and Kai-Yuan Cai, Dynamic random testing with test case clustering and distance-based parameter adjustment, Information and Software Technology, Volume 131, 2021

[PS129] Hao Zhong, Xiaoyin Wang, Hong Mei ,"Inferring Bug Signatures to Detect Real Bugs" ,IEEE Transactions on Software Engineering, 48 (02) pp. 571-584 ,2022

[PS130] Haonan Tong, Bin Liu, Shihai Wang, "Kernel Spectral Embedding Transfer Ensemble for Heterogeneous Defect Prediction." , IEEE Transactions on Software Engineering, 47 (09) pp. 1886-1906 ,2021

[PS131] Haonan Tong, Wei Lu, Weiwei Xing, Bin Liu, and Shihai Wang, SHSE: A subspace hybrid sampling ensemble method for software defect number prediction, Information and Software Technology,142,2022

[PS132] Haonan Zhang, Yiming Tang, Maxime Lamothe, Heng Li, Weiyi Shang, "Studying logging practice in test code", Empirical Software Engineering , 27 (04) pp. 83 ,2022

[PS133] Haowen Chen, Xiao-Yuan Jing, Yuming Zhou, Bing Li, and Baowen Xu, Aligned metric representation based balanced multiset ensemble learning for heterogeneous defect prediction, Information and Software Technology,147,2022

[PS134] Haowen Chen, Xiao-Yuan Jing, Zhiqiang Li, Di Wu, Yi Peng, Zhiguo Huang , "An Empirical Study on Heterogeneous Defect Prediction Approaches" , IEEE Transactions on Software Engineering, 47 (12) pp. 2803-2822 ,2021

[PS135] Hayyan Hasan, Behrouz Tork Ladani, and Bahman Zamani, MEGDroid: A model-driven event generation framework for dynamic android malware analysis, Information and Software Technology, Volume 135, 2021

[PS136] He Ye, Jian Gu, Matias Martinez, Thomas Durieux, Martin Monperru, "Automated Classification of Overfitting Patches With Statically Extracted Code Features " , IEEE Transactions on Software Engineering, 48 (08) pp, 2920-2938 ,2022

[PS137] He Ye, Matias Martinez, and Martin Monperrus, Automated patch assessment for program repair at scale, Empirical Software Engineering,26(2), 2021

[PS138] Héctor D. Menéndez, David Clark ,"Hashing Fuzzing: Introducing Input Diversity to Improve Crash Detection. ", IEEE Transactions on Software Engineering, 48 (09) pp, 3540-3553 ,2022

[PS139] Héctor D. Menéndez, Gunel Jahangirova, Federica Sarro, Paolo Tonella, and David Clark. 2021. Diversifying Focused Testing for Unit Testing. ACM Trans. Softw. Eng. Methodol.

[PS140] Hetong Dai, Heng Li, Che-Shao Chen, Weiyi Shang, Tse-Hsun Chen, "Logram: Efficient Log Parsing Using nn-Gram Dictionaries." , IEEE Transactions on Software Engineering, 48 (03) pp. 879-892 ,2022

[PS141] Hoa Khanh Dam, Truyen Tran, Trang Pham, Shien Wee Ng, John Grundy, Aditya Ghose ,"Automatic Feature Learning for Predicting Vulnerable Software Components.", IEEE Transactions on Software Engineering, 47 (01) pp, 67-85 ,2021

[PS142] Hong Jin Kang, David Lo , "Active Learning of Discriminative Subgraph Patterns for API Misuse Detection. " IEEE Transactions on Software Engineering, 48 (08) pp, 2761-2783 ,2022

[PS143] Huangzhao Zhang, Zhiyi Fu, Ge Li, Lei Ma, Zhehao Zhao, Hua'an Yang, Yizhe Sun, Yang Liu, and Zhi Jin. 2022. Towards Robustness of Deep Program Processing Models—Detection, Estimation, and Enhancement. ACM Trans. Softw. Eng. Methodol.

[PS144] Hui Liu, Jiahao Jin, Zhifeng Xu, Yanzhen Zou, Yifan Bu, Lu Zhang ,"Deep Learning Based Code Smell Detection", IEEE Transactions on Software Engineering, 47 (09) pp. 1811-1837 ,2021

[PS145] Hui Liu, Mingzhu Shen, Jiaqi Zhu, Nan Niu, Ge Li, Lu Zhang ,"Deep Learning Based Program Generation From Requirements Text: Are We There Yet? ", IEEE Transactions on Software Engineering, 48 (04) pp. 1268-1289 ,2022

[PS146] Huy Tu, Tim Menzies, "DebtFree: minimizing labeling cost in self-admitted technical debt identification using semi-supervised learning", Empirical Software Engineering , 27 (04) pp. 80 ,2022

[PS147] Huy Tu, Zhe Yu, Tim Menzies , "Better Data Labelling With EMBLEM (and how that Impacts Defect Prediction). " , IEEE Transactions on Software Engineering, 48 (02) pp.278-294 ,2022

[PS148] Ilaria Pigazzini, Francesca Arcelli Fontana, Bartosz Walter, "A study on correlations between architectural smells and design patterns", Journal of Systems and Software, 178 (08) pp. 110984 ,2021

[PS149] Information and Software Technology,151,2022

[PS150] Islem Saidani, Ali Ouni, Mohamed Wiem Mkaouer, and Fabio Palomba, On the impact of Continuous Integration on refactoring practice: An exploratory study on TravisTorrent, Information and Software Technology,138,2021

[PS151] Ivan Pashchenko, Henrik Plate, Serena Elisa Ponta, Antonino Sabetta, Fabio Massacci, "Vuln4Real: A Methodology for Counting Actually Vulnerable Dependencies " , IEEE Transactions on Software Engineering, 48 (05) pp. 1592-1609 ,2022

[PS152] Jackson A. Prado Lima, Silvia Regina Vergilio ,"A Multi-Armed Bandit Approach for Test Case Prioritization in Continuous Integration Environments " , IEEE Transactions on Software Engineering, 48 (02) pp. 453-465 ,2022

[PS153] Jahanshahi, Hadi, and Mucahit Cevik. S-DABT: Schedule and Dependency-Aware Bug Triage in Open-Source Bug Tracking Systems,Information and Software Technology ,151, 2022

[PS154] James Callan, Oliver Krauss, Justyna Petke, Federica Sarro ,"How do Android developers improve non-functional properties of software?" ,Empirical Software Engineering , 27 (05) pp. 113 ,2022

[PS155] Javaria Imtiaz, Muhammad Zohaib Iqbal, Muhammad Uzair Khan , "An automated model-based approach to repair test suites of evolving web applications" , Journal of Systems and Software, 171 (01) pp. 110841 ,2021

[PS156] Jeongju Sohn, Shin Yoo, "Empirical Evaluation of Fault Localisation Using Code and Change Metrics.", IEEE Transactions on Software Engineering, 47 (08) pp. 1605-1625 ,2021

[PS157] Jhih-Sin Lin, Chin-Yu Huang, Chih-Chiang Fang , "Analysis and assessment of software reliability modeling with preemptive priority queueing policy", Journal of Systems and Software, 187 (05) pp. 111249 ,2022

[PS158] Jhonny Mertz, Ingrid Nunes, "Tigris: A DSL and framework for monitoring software systems at runtime." , Journal of Systems and Software, 177 (07) pp. 110963 ,2021

[PS159] Jiakun Liu, Qiao Huang, Xin Xia, Emad Shihab, David Lo,and Shanping Li.An exploratory study on the introduction and removal of different types of technical debt in deep learning frameworks,Empirical Software Engineering ,26(2),2021

[PS160] Jiaming Ye, Mingliang Ma, Yun Lin, Lei Ma, Yinxing Xue, Jianjun Zhao , "Vulpedia: Detecting vulnerable ethereum smart contracts via abstracted vulnerability signatures.", Journal of Systems and Software, 192 (10) pp. 111410 ,2022

[PS161] Jian Gao, Yu Jiang, Zhe Liu, Xin Yang, Cong Wang, Xun Jiao, Zijiang Yang, Jiaguang Sun ,"Semantic Learning and Emulation Based Cross-Platform Binary Vulnerability Seeker.", IEEE Transactions on Software Engineering, 47 (11) pp. 2575-2589 ,2021

[PS162] Jianyi Zhou, Junjie Chen, and Dan Hao. 2021. Parallel Test Prioritization. ACM Trans. Softw. Eng. Methodol.

[PS163] Jiaojiao Bai, Jingdong Jia, and Luiz Fernando Capretz, A three-stage transfer learning framework for multi-source cross-project software defect prediction, Information and Software Technology,150,2022

[PS164] Jiaojiao Yu, Kunsong Zhao, Jin Liu, Xiao Liu, Zhou Xu, Xin Wang ,"Exploiting gated graph neural network for detecting and explaining self-admitted technical debts", Journal of Systems and Software, 187 (05) pp. 111219 ,2022

[PS165] Jie Tan, Daniel Feitosa, and Paris Avgeriou, Does it matter who pays back Technical Debt? An empirical study of self-fixed TD, Information and Software Technology, 143,2022

[PS166] Jinfu Chen, Weiyi Shang, Emad Shihab , "PerfJIT: Test-Level Just-in-Time Prediction for Performance Regression Introducing Commits " , IEEE Transactions on Software Engineering, 48 (05) pp.1529-1544 ,2022

[PS167] Jing Jiang, Qiudi Wu, Jin Cao, Xin Xia, and Li Zhang, "Recommending tags for pull requests in GitHub", Information and Software Technology , 129, 2021

[PS168] Jirat Pasuksmit, Patanamon Thongtanunam, Shanika Karunasekera ,"Story points changes in agile iterative development." , Empirical Software Engineering , 27 (06) pp. 156 ,2022

[PS169] Jirayus Jiarpakdee, Chakkrit Kla Tantithamthavorn, Hoa Khanh Dam, John C. Grundy , "An Empirical Study of Model-Agnostic Techniques for Defect Prediction Models " , IEEE Transactions on Software Engineering, 48 (02) pp. 166-185 ,2022

[PS170] Jirayus Jiarpakdee, Chakkrit Tantithamthavorn, Ahmed E. Hassan , "The Impact of Correlated Metrics on the Interpretation of Defect Models" , IEEE Transactions on Software Engineering, 47 (02) pp, 320-331 ,2021

[PS171] Joseph Hejderup, Georgios Gousios ,"Can we trust tests to automate dependency updates? A case study of Java Projects", Journal of Systems and Software, 183 (01) pp. 111097 ,2022

[PS172] Joshua Garcia, Ehsan Kouroshfar, Negar Ghorbani, Sam Malek ,"Forecasting Architectural Decay From Evolutionary History. " , IEEE Transactions on Software Engineering, 48 (07) pp,2439-2454 ,2022

[PS173] Juan Manuel Florez, Laura Moreno, Zenong Zhang, Shiyi Wei, Andrian Marcus ,"An empirical study of data constraint implementations in Java" , Empirical Software Engineering , 27 (05) pp. 119 ,2022

[PS174] Jun Wang, Xiaofang Zhang, Lin Chen, and Xiaoyuan Xie, Personalizing label prediction for GitHub issues, Information and Software Technology, 145,2022

[PS175] K. Ayberk Tecimer, Eray Tüzün, Cansu Moran, and Hakan Erdogmus, Cleaning ground truth data in software task assignment, Information and Software Technology, 149,2022

[PS176] Kaiyuan Yang, Junfeng Wang, Zhiyang Fang, Peng Wu, and Zihua Song, Enhancing software modularization via semantic outliers filtration and label propagation, Information and Software Technology, 145,2022

[PS177] Katerina Paltoglou, Vassilis E. Zafeiris, N. A. Diamantidis, Emmanouel A. Giakoumakis, "Automated refactoring of legacy JavaScript code to ES6 modules." , Journal of Systems and Software, 181 (11) pp. 111049 ,2021

[PS178] Kewen Peng, Tim Menzies , "Defect Reduction Planning (Using TimeLIME). " , IEEE Transactions on Software Engineering, 48 (07) pp, 2510-2525,2022

[PS179] Khairul Islam, Toufique Ahmed, Rifat Shahriyar, Anindya Iqbal, and Gias Uddin, Early prediction for merged vs abandoned code changes in modern code reviews, Information and Software Technology, 142,2022

[PS180] Khaled Sellami, Ali Ouni, Mohamed Aymen Saied, Salah Bouktif, and Mohamed Wiem Mkaouer, Improving microservices extraction using evolutionary search, Information and Software Technology,151,2022

[PS181] Khalid Alkharabsheh, Sadi Alawadi, Victor R. Kebande, Yania Crespo, Manuel Fernández-Delgado, and José A. Taboada, A comparison of machine learning algorithms on design smell detection using balanced and imbalanced dataset: A study of God class, Information and Software Technology, 143,2022

[PS182] Khashayar Etemadi, Niloofar Tarighat, Siddharth Yadav, Matias Martinez, Martin Monperrus ,"Estimating the potential of program repair search spaces with commit analysis.".Journal of Systems and Software, 188 (06) pp. 111263 ,2022

[PS183] Krishna Patel, Robert M. Hierons, and David Clark, An information theoretic notion of software testability, Information and Software Technology, 143,2022

[PS184] Kui Liu, Dongsun Kim, Tegawendé F. Bissyandé, Shin Yoo, Yves Le Traon, "Mining Fix Patterns for FindBugs Violations." , IEEE Transactions on Software Engineering, 47 (01) pp, 165-188 ,2021

[PS185] Kui Liu, Li Li, Anil Koyuncu, Dongsun Kim, Zhe Liu, Jacques Klein, Tegawendé F. Bissyandé , "A critical review on the evaluation of automated program repair systems" , Journal of Systems and Software, 171 (01) pp. 110817 ,2021

[PS186] Kunsong Zhao, Zhou Xu, Meng Yan, Tao Zhang, Dan Yang, and Wei Li, A comprehensive investigation of the impact of feature selection techniques on crashing fault residence prediction models, Information and Software Technology,139,2021

[PS187] Laerte Xavier, João Eduardo Montandon, Fabio Ferreira, Rodrigo Brito, Marco Túlio Valente, "On the documentation of self-admitted technical debt in issues.", Empirical Software Engineering , 27 (07) pp. 163 ,2022

[PS188] Lam Nguyen Tung, Hoang-Viet Tran, Khoi Nguyen Le, and Pham Ngoc Hung, An automated test data generation method for void pointers and function pointers in C/C++ libraries and embedded projects, Information and Software Technology, 145,2022

[PS189] Le Yu, Xiapu Luo, Jiachi Chen, Hao Zhou, Tao Zhang, Henry Chang, Hareton K. N. Leung, "PPChecker: Towards Accessing the Trustworthiness of Android Apps' Privacy Policies.", IEEE Transactions on Software Engineering, 47 (02) pp,221-242 ,2021

[PS190] Lin Jiang, Hui Liu, He Jiang, Lu Zhang, Hong Mei ,"Heuristic and Neural Network Based Prediction of Project-Specific API Member Access " , IEEE Transactions on Software Engineering, 48 (04) pp. 1249-1267,2022

[PS191] Liu Wang, Ren He, Haoyu Wang, Pengcheng Xia, Yuanchun Li, Lei Wu, Yajin Zhou, Xiapu Luo, Yulei Sui, Yao Guo, and Guoai Xu, Beyond the virus: a first look at coronavirus-themed Android malware, Empirical Software Engineering, 26(4), 2021

[PS192] Liushan Chen, Yu Pei, Carlo A. Furia , "Contract-Based Program Repair Without The Contracts: An Extended Study" , IEEE Transactions on Software Engineering, 47 (12) pp. 2841-2857,2021

[PS193] Long Zhang, Brice Morin, Philipp Haller, Benoit Baudry, Martin Monperrus ,"A Chaos Engineering System for Live Analysis and Falsification of Exception-Handling in the JVM." ,IEEE Transactions on Software Engineering, 47 (11) pp. 2534-2548 ,2021

[PS194] Lu Xiao, Yuanfang Cai, Rick Kazman, Ran Mo, Qiong Feng , “Detecting the Locations and Predicting the Maintenance Costs of Compound Architectural Debts “ , IEEE Transactions on Software Engineering, 48 (09) pp, 3686-3715 ,2022

[PS195] Luan P. Lima, Lincoln S. Rocha, Carla I. M.Bezerra, and Matheus Paixao, Assessing exception handling testing practices in open-source libraries, Empirical Software Engineering, 26(5), 2021

[PS196] Luca Traini, Daniele Di Pompeo, Michele Tucci, Bin Lin, Simone Scalabrino, Gabriele Bavota, Michele Lanza, Rocco Oliveto, and Vittorio Cortellessa. 2021. How Software Refactoring Impacts Execution Time. ACM Trans. Softw. Eng. Methodol.

[PS197] Luis Cruz, Rui Abreu , “On the Energy Footprint of Mobile Testing Frameworks” , IEEE Transactions on Software Engineering, 47 (10) pp. 2260-2271 ,2021

[PS198] Luiz Alberto Ferreira Gomes, Ricardo da Silva Torres, and Mario Lúcio Côrtes, On the prediction of long-lived bugs: An analysis and comparative study using FLOSS projects, Information and Software Technology, Volume 132, 2021

[PS199] Luiz Laerte Nunes da Silva, Troy Costa Kohwalter, Alexandre Plastino, and Leonardo Gresta Paulino Murta, Sequential coding patterns: How to use them effectively in code recommendation, Information and Software Technology, 140,2021

[PS200] M. Nosrati, H. Haghighi, and M. Vahidi Asl, Test data generation using genetic programming, Information and Software Technology, Volume 130, 2021

[PS201] Majid Hatamian, Samuel Wairimu, Nurul Momen, and Lothar Fritsch, A privacy and security analysis of early-deployed COVID-19 contact tracing Android apps, Empirical Software Engineering, 26(3), 2021

[PS202] Maleknaz Nayebi, Guenther Ruhe, Thomas Zimmermann ,”Mining Treatment-Outcome Constructs from Sequential Software Engineering Data”, IEEE Transactions on Software Engineering, 47 (02) pp, 393-411 ,2021

[PS203] Malinda Dilhara, Ameya Ketkar, and Danny Dig. 2021. Understanding Software-2.0: A Study of Machine Learning Library Usage and Evolution. ACM Trans. Softw. Eng. Methodol.

[PS204] Man Zhang, Bogdan Marculescu, and Andrea Arcuri, Resource and dependency based test case generation for RESTful Web services, Empirical Software Engineering, 26(4), 2021

[PS205] Manish Motwani, Mauricio Soto, Yuriy Brun, René Just, Claire Le Goues, “Quality of Automated Program Repair on Real-World Defects.” , IEEE Transactions on Software Engineering, 48 (02) pp. 637-661 ,2022

[PS206] Manuel Ohrndorf, Christopher Pietsch, Udo Kelter, Lars Grunske, and Timo Kehrer. 2021. History-based Model Repair Recommendations. ACM Trans. Softw. Eng. Methodol.

[PS207] Marc-André Laverdière, Karl Julien,and Ettore Merlo, RBAC protection-impacting changes identification: A case study of the security evolution of two PHP applications, Information and Software Technology,139,2021

[PS208] Maria Kechagia, Sergey Mechtaev, Federica Sarro, Mark Harman ,”Evaluating Automatic Program Repair Capabilities to Repair API Misuses. “ , IEEE Transactions on Software Engineering, 48 (07) pp, 2658-2679,2022

[PS209] Maria Ulan, Welf Löwe, Morgan Ericsson, and Anna Wingkvist, Weighted software metrics aggregation and its application to defect prediction, Empirical Software Engineering, 26(5), 2021

[PS210] Marianna Di Gregorio, Dario Di Nucci, Fabio Palomba, Giuliana Vitiello ,”The making of accessible Android applications: an empirical study on the state of the practice” , Empirical Software Engineering , 27 (06) pp. 145 ,2022

[PS211] Mario Janke, Patrick Mäder ,"Graph Based Mining of Code Change Patterns From Version Control Commits " , IEEE Transactions on Software Engineering, 48 (03) pp. 848-863 ,2022

[PS212] Martina Iammarino, Fiorella Zampetti, Lerina Aversano, Massimiliano Di Penta, "An empirical study on the co-occurrence between refactoring actions and Self-Admitted Technical Debt removal" , Journal of Systems and Software, 178 (08) pp. 110976 ,2021

[PS213] Maryam Raiyat Aliabadi, Mojtaba Vahidi-Asl, Ramak Ghavamizadeh , "ARTINALI++: Multi-dimensional Specification Mining for Complex Cyber-Physical System Security", Journal of Systems and Software, 180 (10) pp. 111016 ,2021

[PS214] Masanari Kondo, Yutaro Kashiwa, Yasutaka Kamei, Osamu Mizuno , "An empirical study of issue-link algorithms: which issue-link algorithms should we use?" , Empirical Software Engineering , 27 (06) pp. 136 ,2022

[PS215] Mateus Lopes, André C. Hora ,"How and why we end up with complex methods: a multi-language study" , Empirical Software Engineering , 27 (05) pp. 115 ,2022

[PS216] Matheus Paixão, Jens Krinke, DongGyun Han, Chaiyong Ragkhitwetsagul, Mark Harman ,"The Impact of Code Review on Architectural Changes." , IEEE Transactions on Software Engineering, 47 (05) pp. 1041-1059 ,2021

[PS217] Mathias Hedenborg, Jonas Lundberg, Welf Löwe, "Memory efficient context-sensitive program analysis", Journal of Systems and Software, 177 (07) pp. 110952 ,2021

[PS218] Mathieu Nassif, Alexa Hernandez, Ashvitha Sridharan, Martin P. Robillard ,"Generating Unit Tests for Documentation ", IEEE Transactions on Software Engineering, 48 (09) pp, 3268-3279,2022

[PS219] Matteo Biagiola and Paolo Tonella. 2022. Testing the Plasticity of Reinforcement Learning-based Systems. ACM Trans. Softw. Eng. Methodol.

[PS220] Maxime Lamothe, Heng Li, Weiyi Shang, "Assisting Example-Based API Misuse Detection via Complementary Artificial Examples ", IEEE Transactions on Software Engineering, 48 (09) pp, 3410-3422,2022

[PS221] Md Hasan Ibrahim, Mohammed Sayagh, and Ahmed E. Hassan, A study of how Docker Compose is used to compose multi-component systems, Empirical Software Engineering, 26(6), 2021

[PS222] Md. Imran Alam, Raju Halder, Jorge Sousa Pinto, "A deductive reasoning approach for database applications using verification conditions" , Journal of Systems and Software, 175 (05) pp. 110903 ,2021

[PS223] Md. Nadim, Manishankar Mondal, Chanchal K. Roy, Kevin A. Schneider ,"Evaluating the performance of clone detection tools in detecting cloned co-change candidates." ,Journal of Systems and Software, 187 (05) pp. 111229 ,2022

[PS224] Mehrdad Abdi, Henrique Rocha, Serge Demeyer, Alexandre Bergel ,"Small-Amp: Test amplification in a dynamically typed language" , Empirical Software Engineering , 27 (06) pp. 128 ,2022

[PS225] Meng Yan, Xin Xia, Yuanrui Fan, Ahmed E. Hassan, David Lo, Shanping Li , "Just-In-Time Defect Identification and Localization: A Two-Phase Framework." , IEEE Transactions on Software Engineering, 48 (02) pp. 82-101 ,2022

[PS226] Mengshi Zhang, Yaoxian Li, Xia Li, Lingchao Chen, Yuqun Zhang, Lingming Zhang, Sarfraz Khurshid, "An Empirical Study of Boosting Spectrum-Based Fault Localization via PageRank", IEEE Transactions on Software Engineering, 47 (06) pp. 1089-1113 ,2021

[PS227] Miao Zhang, Jacky Wai Keung, Yan Xiao, and Md Alamgir Kabir, "Evaluating the effects of similar-class combination on class integration test order generation", Information and Software Technology,129,2021

[PS228] Min Kyung Shin, Sudipto Ghosh, Leo R. Vijayasarathy ,"An empirical comparison of four Java-based regression test selection techniques.", Journal of Systems and Software, 186 (04) pp. 111174 ,2022

[PS229] Ming Wen, Junjie Chen, Yongqiang Tian, Rongxin Wu, Dan Hao, Shi Han, Shing-Chi Cheung, "Historical Spectrum Based Fault Localization" , IEEE Transactions on Software Engineering, 47 (11) pp. 2348-2368 ,2021

[PS230] Minxue Pan, Tongtong Xu, Yu Pei, Zhong Li, Tian Zhang, Xuandong Li , "GUI-Guided Test Script Repair for Mobile Apps. " , IEEE Transactions on Software Engineering, 48 (03) pp.910-929 ,2022

[PS231] Mitchell Joblin and Sven Apel. 2022. How Do Successful and Failed Projects Differ? A Socio-Technical Analysis. ACM Trans. Softw. Eng. Methodol.

[PS232] Mohamed Lamine Kerdoudi, Tewfik Ziadi, Chouki Tibermacine, Salah Sadou, "A novel approach for Software Architecture Product Line Engineering",Journal of Systems and Software, 186 (04) pp. 111191 ,2022

[PS233] Mohammad Javad Beheshtian, Amir Hossein Bavand, Peter C. Rigby , "Software Batch Testing to Save Build Test Resources and to Reduce Feedback Time. " , IEEE Transactions on Software Engineering, 48 (08) pp,2784-2801 ,2022

[PS234] Mohammad Masudur Rahman, Foutse Khomh, Marco Castelluccio ,"Works for Me! Cannot Reproduce - A Large Scale Empirical Study of Non-reproducible Bugs" , Empirical Software Engineering , 27 (05) pp. 111 ,2022

[PS235] Mohammad Masudur Rahman, Foutse Khomh, Shamima Yeasmin, and Chanchal K.Roy, The forgotten role of search queries in IR-based bug localization: an empirical study, Empirical Software Engineering, 26(6), 2021

[PS236] Mohammed Sayagh, Ahmed E. Hassan,"ConfigMiner: Identifying the Appropriate Configuration Options for Config-Related User Questions by Mining Online Forums" , IEEE Transactions on Software Engineering, 47 (12) pp. 2907-2918,2021

[PS237] Mojtaba Bagherzadeh, Nafiseh Kahani, Lionel C. Briand ,"Reinforcement Learning for Test Case Prioritization ", IEEE Transactions on Software Engineering, 48 (08) pp, 2836-2856,2022

[PS238] Morena Barboni, Andrea Morichetta, Andrea Polini ,"SuMo: A mutation testing approach and tool for the Ethereum blockchain" ,Journal of Systems and Software, 193 (11) pp. 111445 ,2022

[PS239] Moses Openja, Mohammad Mehdi Morovati, Le An, Foutse Khomh, Mouna Abidi, "Technical debts and faults in open-source quantum software systems: An empirical study",Journal of Systems and Software, 193 (11) pp. 111458 ,2022

[PS240] Mouna Abidi, Md Saidur Rahman, Moses Openja, and Foutse Khomh. 2021. Are Multi-Language Design Smells Fault-Prone? An Empirical Study. ACM Trans. Softw. Eng. Methodol.

[PS241] Navid Teymourian, Habib Izadkhah, Ayaz Isazadeh ,"A Fast Clustering Algorithm for Modularization of Large-Scale Software Systems. " , IEEE Transactions on Software Engineering, 48 (04) pp. 1451-1462 ,2022

[PS242] Nemania Borovits, Indika Kumara, Dario Di Nucci, Parvathy Krishnan, Stefano Dalla Palma, Fabio Palomba, Damian A. Tamburri, Willem-Jan van den Heuvel ,"FindICI: Using

machine learning to detect linguistic inconsistencies between code and natural language descriptions in infrastructure-as-code", Empirical Software Engineering , 27 (07) pp. 178 ,2022

[PS243] Nezih Sunman, Yigit Soydan, Hasan Sözer ,"Automated Web application testing driven by pre-recorded test case" ,Journal of Systems and Software, 193 (11) pp. 111441 ,2022

[PS244] Nicolas Harrand, Amine Benelallam, César Soto-Valero, François Bettega, Olivier Barais, Benoit Baudry, "API beauty is in the eye of the clients: 2.2 million Maven dependencies reveal the spectrum of client-API usages" ,Journal of Systems and Software, 184 (02) pp. 111134 ,2022

[PS245] Nima Miryeganeh, Sepehr Hashtroudi, Hadi Hemmati ,"GloBug: Using global data in Fault Localization." ,Journal of Systems and Software, 177 (07) pp. 110961 ,2021

[PS246] Olivier Nourry, Yutaro Kashiwa, Yasutaka Kamei, and Naoyasu Ubayashi, Does shortening the release cycle affect refactoring activities: A case study of the JDT Core, Platform SWT, and UI projects, Information and Software Technology,139,2021

[PS247] Osamah AlDhafer, Irfan Ahmad, and Sajjad Mahmood, An end-to-end deep learning system for requirements classification using recurrent neural networks, Information and Software Technology, 147,2022

[PS248] Oumayma Hamdi, Ali Ouni, Mel Ó Cinnéide, and Mohamed Wiem Mkaouer, A longitudinal study of the impact of refactoring in android applications, Information and Software Technology, 140,2021

[PS249] Patrick Keller, Abdoul Kader Kaboré, Laura Plein, Jacques Klein, Yves Le Traon, and Tegawendé F. Bissyandé. 2021. What You See is What it Means! Semantic Representation Learning of Code based on Visualization and Transfer Learning. ACM Trans. Softw. Eng. Methodol.

[PS250] Pattara Leelaprute, and Sousuke Amasaki, A comparative study on vectorization methods for non-functional requirements classification, Information and Software Technology,150,2022

[PS251] Paul Muntean, Martin Monperrus, Hao Sun, Jens Grossklags, Claudia Eckert , " IntRepair: Informed Repairing of Integer Overflows" , IEEE Transactions on Software Engineering, 47 (10) pp.2225-2241 ,2021

[PS252] Paul Temple, Mathieu Acher, Jean-Marc Jézéquel ,"Empirical Assessment of Multimorphic Testing." , IEEE Transactions on Software Engineering, 47 (07) pp. 1511-1527 ,2021

[PS253] Peipei Wang, Chris Brown, Jamie A. Jennings, Kathryn T. Stolee, "Demystifying regular expression bugs.", Empirical Software Engineering , 27 (01) pp. 21 ,2022

[PS254] Peng Zhang, Yang Wang, Xutong Liu, Yanhui Li, Yibiao Yang, Ziyuan Wang, Xiaoyu Zhou, Lin Chen, and Yuming Zhou. 2022. Mutant Reduction Evaluation: What is There and What is Missing? ACM Trans. Softw. Eng. Methodol.

[PS255] Peng Zhang, Yanhui Li, Wanwangying Ma, Yibiao Yang, Lin Chen, Hongmin Lu, Yuming Zhou, Baowen Xu , "CBUA: A Probabilistic, Predictive, and Practical Approach for Evaluating Test Suite Effectiveness " , IEEE Transactions on Software Engineering, 48 (03) pp. 1067-1096,2022

[PS256] Phuong T. Nguyen, Juri Di Rocco, Claudio Di Sipio, Davide Di Ruscio, Massimiliano Di Penta ,"Recommending API Function Calls and Code Snippets to Support Software Development " , IEEE Transactions on Software Engineering, 48 (07) pp, 2417-2438,2022

[PS257] Pooja Rani, Sebastiano Panichella, Manuel Leuenberger, Andrea Di Sorbo, Oscar Nierstrasz, “How to identify class comment types? A multi-language approach for class comment classification”, Journal of Systems and Software, 181 (11) pp. 111047 ,2021

[PS258] Pooja Rani, Sebastiano Panichella, Manuel Leuenberger, Mohammad Ghafari, and Oscar Nierstrasz, What do class comments tell us? An investigation of comment evolution and practices in Pharo Smalltalk, Empirical Software Engineering, 26(6), 2021

[PS259] Qiang Hu, Yuejun Guo, Maxime Cordy, Xiaofei Xie, Lei Ma, Mike Papadakis, and Yves Le Traon. 2022. An Empirical Study on Data Distribution-Aware Test Selection for Deep Learning Enhancement. ACM Trans. Softw. Eng. Methodol.

[PS260] Qianqian Zhu, Andy Zaidman, Annibale Panichella, “How to kill them all: An exploratory study on the impact of code observability on mutation testing” , Journal of Systems and Software, 173 (03) pp. 110864 ,2021

[PS261] Quang-Hung Luu, Man Fai Lau, Sebastian P. H. Ng, Tsong Yueh Chen, “Testing multiple linear regression systems with metamorphic testing”.Journal of Systems and Software, 182 (12) pp. 111062 ,2021

[PS262] Quanjun Zhang, Chunrong Fang, Weisong Sun, Shengcheng Yu, Yutao Xu, Yulei Liu , “Test case prioritization using partial attention” , Journal of Systems and Software, 192 (10) pp. 111419 ,2022

[PS263] Quanyi Zou, Lu Lu, Zhanyu Yang, Xiaowei Gu,and Shaojian Qiu, Joint feature representation learning and progressive distribution matching for cross-project defect prediction, Information and Software Technology,137,2021

[PS264] Qun Mao, Weiwei Wang, Feng You, Ruilian Zhao, Zheng Li, “User behavior pattern mining and reuse across similar Android apps” , Journal of Systems and Software, 183 (01) pp. 111085 ,2022

[PS265] Rabe Abdalkareem, Suhaib Mujahid, Emad Shihab , “A Machine Learning Approach to Improve the Detection of CI Skip Commits” , IEEE Transactions on Software Engineering, 47 (12) pp. 2740-2754 ,2021

[PS266] Rafael Caballero, Enrique Martin-Martin, Adrián Riesco,and Salvador Tamarit, "A unified framework for declarative debugging and testing", Information and Software Technology,129,2021

[PS267] Rafi Almhana, Marouane Kessentini,and Wiem Mkaouer, Method-level bug localization using hybrid multi-objective search, Information and Software Technology, Volume 131, 2021

[PS268] Rahul Krishna, Chong Tang, Kevin J. Sullivan, Baishakhi Ray , “ConEx: Efficient Exploration of Big-Data System Configurations for Better Performance “ , IEEE Transactions on Software Engineering, 48 (03) pp. 893-909 ,2022

[PS269] Rahul Krishna, Vivek Nair, Pooyan Jamshidi, Tim Menzies, “Whence to Learn? Transferring Knowledge in Configurable Systems Using BEETLE” , IEEE Transactions on Software Engineering, 47 (12) pp. 2956-2972 ,2021

[PS270] Rahul Yedida, Tim Menzies , “On the Value of Oversampling for Deep Learning in Software Defect Prediction “ , IEEE Transactions on Software Engineering, 48 (08) pp, 3103-3116 ,2022

[PS271] Ran Mo, Shaozhi Wei, Qiong Feng, Zengyang Li, An exploratory study of bug prediction at the method level, Information and Software Technology, 144,2022

[PS272] Ran Mo, Yuanfang Cai, Rick Kazman, Lu Xiao, Qiong Feng ,"Architecture Anti-Patterns: Automatically Detectable Violations of Design Principles", IEEE Transactions on Software Engineering, 47 (05) pp. 1008-1028 ,2021

[PS273] Ratnadira Widyasari, Gede Artha Azriadi Prana, Stefanus Agus Haryono, Shaowei Wang, David Lo ,"Real world projects, real faults: evaluating spectrum based fault localization techniques on Python projects" , Empirical Software Engineering , 27 (06) pp. 147 ,2022

[PS274] Rezvan Mahdavi-Hezaveh, Nirav Ajmeri, and Laurie Williams, Feature toggles as code: Heuristics and metrics for structuring feature toggles, Information and Software Technology, 145,2022

[PS275] Riccardo Coppola, Luca Ardito, Marco Torchiano, Emil Alégroth ,"Translation from layout-based to visual android test scripts: An empirical evaluation.", Journal of Systems and Software, 171 (01) pp. 110845 ,2021

[PS276] Richárd Szalay, Ábel Sinkovics, Zoltán Porkoláb, "Practical heuristics to improve precision for erroneous function argument swapping detection in C and C++." ,Journal of Systems and Software, 181 (11) pp. 111048 ,2021

[PS277] Ridhi Jain, Rahul Purandare, and Subodh Sharma. 2022. BiRD: Race Detection in Software Binaries under Relaxed Memory Models. ACM Trans. Softw. Eng. Methodol.

[PS278] Ridwan Salihin Shariffdeen, Shin Hwei Tan, Mingyuan Gao, Abhik Roychoudhury. 2021. Automated Patch Transplantation. ACM Trans. Softw. Eng. Methodol.

[PS279] Robert White, Jens Krinke ,"TCTracer: Establishing test-to-code traceability links using dynamic and static techniques." , Empirical Software Engineering , 27 (03) pp. 67 ,2022

[PS280] Roberto Natella , "StateAFL: Greybox fuzzing for stateful network servers" , Empirical Software Engineering , 27 (07) pp. 190 ,2022

[PS281] Rohit Gheyi, Márcio Ribeiro, Beatriz Souza, Marcio Guimarães, Leo Fernandes, Marcelo d'Amorim, Vander Alves, Leopoldo Teixeira, and Baldoino Fonseca, Identifying method-level mutation subsumption relations using Z3, Information and Software Technology, Volume 132, 2021

[PS282] Roland Kretschmer, Djamel Eddine Khelladi, Alexander Egyed , "Transforming abstract to concrete repairs with a generative approach of repair values" , Journal of Systems and Software, 175 (05) pp. 110889 ,2021

[PS283] Rômulo Manciola Meloca, Ingrid Nunes ,"A comparative study of application-level caching recommendations at the method level, Empirical Software Engineering , 27 (04) pp. 88 ,2022

[PS284] Ruben Heradio, David Fernández-Amorós, José A. Galindo, David Benavides, Don S. Batory ,"Uniform and scalable sampling of highly configurable systems.", Empirical Software Engineering , 27 (02) pp. 44 ,2022

[PS285] Sandra L. Ramírez-Mora, Hanna Oktaba, Helena Gómez-Adorno,and Gerardo Sierra, Exploring the communication functions of comments during bug fixing in Open Source Software projects, Information and Software Technology,136,2021

[PS286] Sarra Habchi, Naouel Moha, Romain Rouvoy, "Android code smells: From introduction to refactoring." , Journal of Systems and Software, 177 (07) pp. 110964 ,2021

[PS287] Saurabh Malgaonkar, Sherlock A. Licorish, and Bastin Tony Roy Savarimuthu, Prioritizing user concerns in app reviews – A study of requests for new features, enhancements and bug fixes, Information and Software Technology,144,2022

[PS288] Sebastiano Panichella, Gerardo Canfora, and Andrea Di Sorbo, “Won’t We Fix this Issue?” Qualitative characterization and automated identification of wontfix issues on GitHub, Information and Software Technology,139,2021

[PS289] Seonah Lee, Rongxin Wu, Shing-Chi Cheung, Sungwon Kang , “Automatic Detection and Update Suggestion for Outdated API Names in Documentation.” ,IEEE Transactions on Software Engineering, 47 (04) pp,653-675 ,2021

[PS290] Shaiful Alam Chowdhury, Reid Holmes, Andy Zaidman, Rick Kazman , “Revisiting the debate: Are code metrics useful for measuring maintenance effort?” , Empirical Software Engineering , 27 (06) pp. 158 ,2022

[PS291] Shangqing Liu, Cuiyun Gao, Sen Chen, Lun Yiu Nie, Yang Liu , “ATOM: Commit Message Generation Based on Abstract Syntax Tree and Hybrid Ranking.” , IEEE Transactions on Software Engineering, 48 (05) pp. 1800-1817 ,2022

[PS292] Shayan Zamani, and Hadi Hemmati, A pragmatic approach for hyper-parameter tuning in search-based test case generation, Empirical Software Engineering, 26(6), 2021

[PS293] Shiran Liu, Zhaoqiang Guo, Yanhui Li, Hongmin Lu, Lin Chen, Lei Xu, Yuming Zhou, and Baowen Xu, Prioritizing code documentation effort: Can we do it simpler but better? , Information and Software Technology,140,2021

[PS294] Shivashree Vysali, Shane McIntosh, Bram Adams ,”Quantifying, Characterizing, and Mitigating Flakily Covered Program Elements” , IEEE Transactions on Software Engineering, 48 (03) pp.1018-1029 ,2022

[PS295] Shiwen Yu, Ting Wang, Ji Wang ,”Data Augmentation by Program Transformation” ,Journal of Systems and Software, 190 (08) pp. 111304 ,2022

[PS296] Shiyue Rong, Weisheng Wang, Umme Ayda Mannan, Eduardo Santana de Almeida, Shurui Zhou, and Iftekhar Ahmed, An empirical study of emoji use in software development communication, Information and Software Technology,148,2022

[PS297] Shouvick Mondal, Rupesh Nasre, “Hansie: Hybrid and consensus regression test prioritization” , Journal of Systems and Software, 172 (02) pp. 110850 ,2021

[PS298] Shujuan Jiang, Miao Zhang, Yanmei Zhang, Rongcun Wang, Qiao Yu, Jacky Wai Keung, “An Integration Test Order Strategy to Consider Control Coupling.” , IEEE Transactions on Software Engineering, 47 (07) pp. 1350-1367 ,2021

[PS299] Shuo Feng, Jacky Keung, Peichang Zhang, Yan Xiao, and Miao Zhang, The impact of the distance metric and measure on SMOTE-based techniques in software defect prediction, Information and Software Technology,142,2022

[PS300] Shuo Feng, Jacky Keung, Xiao Yu, Yan Xiao, and Miao Zhang, Investigation on the stability of SMOTE-based oversampling techniques in software defect prediction, Information and Software Technology,139,2021

[PS301] Shuo Feng, Jacky Keung, Xiao Yu, Yan Xiao, Kwabena Ebo Bennin, Md Alamgir Kabir,and Miao Zhang, "COSTE: Complexity-based OverSampling TEchnique to alleviate the class imbalance problem in software defect prediction", Information and Software Technology,129,2021

[PS302] Sicong Cao, Xiaobing Sun, Lili Bo, Ying Wei,and Bin Li, BGNN4VD: Constructing Bidirectional Graph Neural-Network for Vulnerability Detection, Information and Software Technology,136,2021

[PS303] Simone Scalabrino, Antonio Mastropaolo, Gabriele Bavota, and Rocco Oliveto. 2021. An Adaptive Search Budget Allocation Approach for Search-Based Test Case Generation. ACM Trans. Softw. Eng. Methodol.

[PS304] Sofia Reis, Rui Abreu, and Luis Cruz, Fixing vulnerabilities potentially hinders maintainability, Empirical Software Engineering, 26(6), 2021

[PS305] Sofien Boutaib, Maha Elarbi, Slim Bechikh, Fabio Palomba, Lamjed Ben Said , “Handling uncertainty in SBSE: a possibilistic evolutionary approach for code smells detection” , Empirical Software Engineering , 27 (06) pp. 124 ,2022

[PS306] Solm Salimi, Mehdi Kharrazi ,”VulSlicer: Vulnerability detection through code slicing” ,Journal of Systems and Software, 193 (11) pp. 111450 ,2022

[PS307] Song Wang, Chetan Bansal,and Nachiappan Nagappan, "Large-scale intent analysis for identifying large-review-effort code changes", Information and Software Technology,130,2021

[PS308] Sooyoung Cha, Seongjoon Hong, Jiseong Bak, Jingyoung Kim, Junhee Lee, Hakjoo Oh , “Enhancing Dynamic Symbolic Execution by Automatically Learning Search Heuristics “, IEEE Transactions on Software Engineering, 48 (09) pp, 3640-3663 ,2022

[PS309] Sophia Quach, Maxime Lamothe, Yasutaka Kamei, and Weiyi Shang, An empirical study on the use of SZZ for identifying inducing changes of non-functional bugs, Empirical Software Engineering, 26(4), 2021

[PS310] Stefanus A. Haryono, Ferdian Thung, David Lo, Lingxiao Jiang, Julia Lawall, Hong Jin Kang, Lucas Serrano, Gilles Muller, “AndroEvolve: automated Android API update with data flow analysis and variable denormalization.”, Empirical Software Engineering , 27 (03) pp. 73 ,2022

[PS311] Steffen Herbold, Alexander Trautsch, Fabian Trautsch, Benjamin Ledel ,”Problems with SZZ and features: An empirical study of the state of practice of defect prediction data collection.” , Empirical Software Engineering , 27 (02) pp. 42 ,2022

[PS312] Steffen Tunkel, Steffen Herbold,”Exploring the relationship between performance metrics and cost saving potential of defect prediction models.” , Empirical Software Engineering , 27 (07) pp. 182 ,2022

[PS313] Steven Locke, Heng Li, Tse-Hsun Peter Chen, Weiyi Shang, Wei Liu , “LogAssist: Assisting Log Analysis Through Log Summarization “ , IEEE Transactions on Software Engineering, 48 (09) pp, 3227-3241 ,2022

[PS314] Supatsara Wattanakriengkrai, Patanamon Thongtanunam, Chakkrit Tantithamthavorn, Hideaki Hata, Kenichi Matsumoto ,”Predicting Defective Lines Using a Model-Agnostic Technique. “ , IEEE Transactions on Software Engineering, 48 (05) pp.1480-1496 ,2022

[PS315] Suppawong Tuarob, Noppadol Assavakamhaenghan, Waralee Tanaphantaruk, Ponlakit Suwanworaboon, Saeed-Ul Hassan, and Morakot Choetkiertikul, Automatic team recommendation for collaborative software development, Empirical Software Engineering, 26(4), 2021

[PS316] Suyu Ma, Zhenchang Xing, Chunyang Chen, Cheng Chen, Lizhen Qu, Guoqiang Li ,”Easy-to-Deploy API Extraction by Multi-Level Feature Embedding and Transfer Learning” , IEEE Transactions on Software Engineering, 47 (10) pp.2296-2311 ,2021

[PS317] Taher Ahmed Ghaleb, Daniel Alencar da Costa, Ying Zou, Ahmed E. Hassan , “Studying the Impact of Noises in Build Breakage Data” , IEEE Transactions on Software Engineering, 47 (09) pp. 1998-2011 ,2021

[PS318] Tao Zhang, Jiachi Chen, Xian Zhan, Xiapu Luo, David Lo, He Jiang, “Where2Change: Change Request Localization for App Reviews.”, IEEE Transactions on Software Engineering, 47 (11) pp. 2590-2616 ,2021

[PS319] Thazin Win Win Aung, Yao Wan, Huan Huo, Yulei Sui ,"Multi-triage: A multi-task learning framework for bug triage" , Journal of Systems and Software, 184 (02) pp. 111133 ,2022

[PS320] Thierry Titcheu Chekam, Mike Papadakis, Maxime Cordy, and Yves Le Traon. 2021. Killing Stubborn Mutants with Symbolic Execution. ACM Trans. Softw. Eng. Methodol.

[PS321] Thomas Bock, Angelika Schmid, and Sven Apel. 2021. Measuring and Modeling Group Dynamics in Open-Source Software Development: A Tensor Decomposition Approach. ACM Trans. Softw. Eng. Methodol.

[PS322] Tianpei Xia, Rui Shu, Xipeng Shen, Tim Menzies , "Sequential Model Optimization for Software Effort Estimation ", IEEE Transactions on Software Engineering, 48 (06) pp.1994-2009 ,2022

[PS323] Tobias Olsson, Morgan Ericsson, Anna Wingkvist, "To automatically map source code entities to architectural modules with Naive Bayes.",Journal of Systems and Software, 183 (01) pp. 111095 ,2022

[PS324] Tongtong Xu, Liushan Chen, Yu Pei, Tian Zhang, Minxue Pan, Carlo A. Furia, "Restore: Retrospective Fault Localization Enhancing Automated Program Repair " , IEEE Transactions on Software Engineering, 48 (02) pp.309-326 ,2022

[PS325] Toshiki Hirao, Shane McIntosh, Akinori Ihara, Kenichi Matsumoto ,"Code Reviews With Divergent Review Scores: An Empirical Study of the OpenStack and Qt Communities. ", IEEE Transactions on Software Engineering, 48 (02) pp. 69-81,2022

[PS326] Truong Ho-Quang, Arif Nurwidyantoro, Satrio Adi Rukmono, Michel R. V. Chaudron, Fabian Fröding, Duy Nguyen Ngoc, "Role stereotypes in software designs and their evolution" , Journal of Systems and Software, 189 (07) pp. 111296 ,2022

[PS327] Valentina Lenarduzzi, Vili Nikkola, Nyyti Saarimäki, Davide Taibi , "Does code quality affect pull request acceptance? An empirical study." , Journal of Systems and Software, 171 (01) pp. 110806 ,2021

[PS328] Valeria Pontillo, Fabio Palomba, Filomena Ferrucci ,"Static test flakiness prediction: How Far Can We Go?", Empirical Software Engineering , 27 (07) pp. 187 ,2022

[PS329] Van-Thuan Pham, Marcel Böhme, Andrew E. Santosa, Alexandru Razvan Caciulescu, Abhik Roychoudhury, "Smart Greybox Fuzzing" , IEEE Transactions on Software Engineering, 47 (09) pp. 1980-1997 ,2021

[PS330] Veronika Dashuber, and Michael Philippsen, Trace visualization within the Software City metaphor: Controlled experiments on program comprehension, Information and Software Technology,150,2022

[PS331] Vijay Walunj, Gharib Gharibi, Rakan Alanazi, Yugyung Lee ,"Defect prediction using deep learning with Network Portrait Divergence for software evolution", Empirical Software Engineering , 27 (05) pp. 118,2022

[PS332] Vu Nguyen,and Bach Le, RLTCP: A reinforcement learning approach to prioritizing automated user interface tests, Information and Software Technology,136,2021

[PS333] Wei Zheng, Tianren Shen, Xiang Chen, Peiran Deng ,"Interpretability application of the Just-in-Time software defect prediction model" ,Journal of Systems and Software, 188 (06) pp. 111245 ,2022

[PS334] Weifeng Pan, Ming Hua, Carl K. Chang, Zijiang Yang, Dae-Kyoo Kim , "ElementRank: Ranking Java Software Classes and Packages using a Multilayer Complex Network-Based Approach" , IEEE Transactions on Software Engineering, 47 (10) pp. 2272-2295 ,2021

[PS335] Wolfgang Mauerer, Mitchell Joblin, Damian A. Tamburri, Carlos V. Paradis, Rick Kazman, Sven Apel, “In Search of Socio-Technical Congruence: A Large-Scale Longitudinal Study “, IEEE Transactions on Software Engineering, 48 (08) pp, 3159-3184 ,2022

[PS336] Wuxia Jin, Ting Liu, Yuanfang Cai, Rick Kazman, Ran Mo, Qinghua Zheng,”Service Candidate Identification from Monolithic Systems Based on Execution Traces”, IEEE Transactions on Software Engineering, 47 (05) pp. 987-1007,2021

[PS337] Xi Xiao, Yuqing Pan, Bin Zhang, Guangwu Hu, Qing Li,and Runiu Lu, ALBFL: A novel neural ranking model for software fault localization via combining static and dynamic features, Information and Software Technology,139,2021

[PS338] Xiajing Wang, Changzhen Hu, Rui Ma, Donghai Tian, and Jinyuan He, CMFuzz: context-aware adaptive mutation for fuzzers, Empirical Software Engineering, 26(1), 2021

[PS339] Xiang Chen, Yanzhou Mu, Ke Liu, Zhanqi Cui, and Chao Ni, “Revisiting heterogeneous defect prediction methods: How far are we?” , Information and Software Technology, Volume 130, 2021

[PS340] Xiang Gao, Bo Wang, Gregory J. Duck, Ruyi Ji, Yingfei Xiong, and Abhik Roychoudhury. 2021. Beyond Tests: Program Vulnerability Repair via Crash Constraint Extraction. ACM Trans. Softw. Eng. Methodol.

[PS341] Xiangjuan Yao, Gongjie Zhang, Feng Pan, Dunwei Gong, Changqing Wei , “Orderly Generation of Test Data via Sorting Mutant Branches Based on Their Dominance Degrees for Weak Mutation Testing “ , IEEE Transactions on Software Engineering, 48 (04) pp. 1169-1184,2022

[PS342] Xiangying Dang, Dunwei Gong, Xiangjuan Yao, Tian Tian, Huai Liu , “Enhancement of Mutation Testing via Fuzzy Clustering and Multi-Population Genetic Algorithm “ , IEEE Transactions on Software Engineering, 48 (06) pp. 2141-2156 ,2022

[PS343] Xiao Cheng, Haoyu Wang, Jiayi Hua, Guoai Xu, and Yulei Sui. 2021. DeepWukong: Statically Detecting Software Vulnerabilities Using Deep Graph Neural Network. ACM Trans. Softw. Eng. Methodol.

[PS344] Xiao Ling, Rishabh Agrawal, Tim Menzies ,”How Different is Test Case Prioritization for Open and Closed Source Projects? “ , IEEE Transactions on Software Engineering, 48 (07) pp, 2526-2540,2022

[PS345] Xiaobo Yan, Bin Liu, Shihai Wang , “A Test Restoration Method based on Genetic Algorithm for effective fault localization in multiple-fault programs”, Journal of Systems and Software, 172 (02) pp. 110861 ,2021

[PS346] Xiaofei Xie, Tianlin Li, Jian Wang, Lei Ma, Qing Guo, Felix Juefei-Xu, and Yang Liu. 2022. NPC: Neuron Path Coverage via Characterizing Decision Logic of Deep Neural Networks. ACM Trans. Softw. Eng. Methodol.

[PS347] Xiaofeng Han, Amjed Tahir, Peng Liang, Steve Counsell, Kelly Blincoe, Bing Li, Yajing Luo ,”Code smells detection via modern code review: a study of the OpenStack and Qt communities” , Empirical Software Engineering , 27 (06) pp.127 ,2022

[PS348] Xiaoqin Fu, Haipeng Cai, Wen Li, and Li Li. 2021. SEADS: Scalable and Cost-effective Dynamic Dependence Analysis of Distributed Systems via Reinforcement Learning. ACM Trans. Softw. Eng. Methodol.

[PS349] Xiaoxue Ma, Jacky Keung, Zhen Yang, Xiao Yu, Yishu Li, and Hao Zhang, CASMS: Combining clustering with attention semantic model for identifying security bug reports, Information and Software Technology,147,2022

[PS350] Xiaoxue Ma, Shangru Wu, Ernest Pobee, Xiupei Mei, Hao Zhang, Bo Jiang, and Wing-Kwong Chan. 2021. RegionTrack: A Trace-Based Sound and Complete Checker to Debug Transactional Atomicity Violations and Non-Serializable Traces. ACM Trans. Softw. Eng. Methodol.

[PS351] Xiaoxue Wu, Wei Zheng, Xiang Chen, Yu Zhao, Tingting Yu,and Dejun Mu, Improving high-impact bug report prediction with combination of interactive machine learning and active learning, Information and Software Technology, Volume 133, 2021

[PS352] Xiaoyu Sun, Li Li, Tegawendé F. Bissyandé, Jacques Klein, Damien Octeau, and John Grundy. 2021. Taming Reflection: An Essential Step Toward Whole-program Analysis of Android Apps. ACM Trans. Softw. Eng. Methodol.

[PS353] Xiuting Ge, Chunrong Fang, Meiyuan Qian, Yu Ge, and Mingshuang Qing, Locality-based security bug report identification via active learning, Information and Software Technology, 147,2022

[PS354] Xueqi Yang, Jianfeng Chen, Rahul Yedida, Zhe Yu, and Tim Menzies, Learning to recognize actionable static code warnings (is intrinsically easy), Empirical Software Engineering, 26(3), 2021

[PS355] Yan Xiaobo, Liu Bin, Wang Shihai, An Dong, Zhu Feng,and Yang Yelin, Efilter: An effective fault localization based on information entropy with unlabelled test cases, Information and Software Technology, Volume 134, 2021

[PS356] Yanjie Zhao, Li Li, Xiaoyu Sun, Pei Liu,and John Grundy, Icon2Code: Recommending code implementations for Android GUI components, Information and Software Technology,138,2021

[PS357] Yaqin Zhou, Jing Kai Siow, Chenyu Wang, Shangqing Liu, and Yang Liu. 2021. SPI: Automated Identification of Security Patches via Commits. ACM Trans. Softw. Eng. Methodol.

[PS358] Yida Tao, Shan Tang, Yepang Liu, Zhiwu Xu, and Shengchao Qin. 2021. Speeding Up Data Manipulation Tasks with Alternative Implementations: An Exploratory Study. ACM Trans. Softw. Eng. Methodol.

[PS359] Yikun Hu, Hui Wang, Yuanyuan Zhang, Bodong Li, Dawu Gu , “A Semantics-Based Hybrid Approach on Binary Code Similarity Comparison” , IEEE Transactions on Software Engineering, 47 (06) pp. 1241-1258 ,2021

[PS360] Yilin Yang, Tianxing He, Zhilong Xia, and Yang Feng, A comprehensive empirical study on bug characteristics of deep learning frameworks, Information and Software Technology,151,2022

[PS361] Yin Liu, Siddharth Dhar, Eli Tilevich, “Only pay for what you need: Detecting and removing unnecessary TEE-based code.” ,Journal of Systems and Software, 188 (06) pp. 111253 ,2022

[PS362] Yingfei Xiong and Bo Wang. 2022. L2S: A Framework for Synthesizing the Most Probable Program under a Specification. ACM Trans. Softw. Eng. Methodol.

[PS363] Yingzhe Lyu, Heng Li, Mohammed Sayagh, Zhen Ming (Jack) Jiang, and Ahmed E. Hassan. 2021. An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions. ACM Trans. Softw. Eng. Methodol.

[PS364] Youngjoo Ko, Bin Zhu, Jong Kim ,”Fuzzing with automatically controlled interleavings to detect concurrency bugs.” ,Journal of Systems and Software, 191 (09) pp. 111379 ,2022

[PS365] Yu Nong, Haipeng Cai, Pengfei Ye, Li Li,and Feng Chen, Evaluating and comparing memory error vulnerability detectors, Information and Software Technology,137,2021

[PS366] Yu Qu, Jianlei Chi,and Heng Yin, Leveraging developer information for efficient effort-aware bug prediction, Information and Software Technology, 137,2021

[PS367] Yu Qu, Qinghua Zheng, Jianlei Chi, Yangxu Jin, Ancheng He, Di Cui, Hengshan Zhang, Ting Liu, “Using K-core Decomposition on Class Dependency Networks to Improve Bug Prediction Model's Practical Performance” , IEEE Transactions on Software Engineering, 47 (02) pp,348-366 ,2021

[PS368] Yu Wang, Fengjuan Gao, Linzhang Wang, Tingting Yu, Jianhua Zhao, Xuandong Li , “Automatic Detection, Validation, and Repair of Race Conditions in Interrupt-Driven Embedded Software “,IEEE Transactions on Software Engineering, 48 (02) pp.346-363,2022

[PS369] Yu Zhou, Xinying Yang, Taolue Chen, Zhiqiu Huang, Xiaoxing Ma, Harald C. Gall ,”Boosting API Recommendation With Implicit Feedback “ ,IEEE Transactions on Software Engineering, 48 (06) pp. 2157-2172 ,2022

[PS370] Yu Zhou, Yanqi Su, Taolue Chen, Zhiqiu Huang, Harald C. Gall, Sebastiano Panichella, “User Review-Based Change File Localization for Mobile Applications.” , IEEE Transactions on Software Engineering, 47 (12) pp. 2755-2770 ,2021

[PS371] Yuan Huang, Jinyu Jiang, Xiapu Luo, Xiangping Chen, Zibin Zheng, Nan Jia, Gang Huang ,”Change-Patterns Mapping: A Boosting Way for Change Impact Analysis “, IEEE Transactions on Software Engineering, 48 (07) pp, 2376-2398 ,2022

[PS372] Yuan Huang, Xingjian Liang, Zhihao Chen, Nan Jia, Xiapu Luo, Xiangping Chen, Zibin Zheng, Xiaocong Zhou ,”Reviewing rounds prediction for code patches”, Empirical Software Engineering , 27 (01) pp. 7 ,2022

[PS373] Yuanrui Fan, Xin Xia, Daniel Alencar da Costa, David Lo, Ahmed E. Hassan, Shanping Li ,”The Impact of Mislabeled Changes by SZZ on Just-in-Time Defect Prediction”, IEEE Transactions on Software Engineering, 47 (08) pp. 1559-1586 ,2021

[PS374] Yusuf Sulistyo Nugroho, Syful Islam, Keitaro Nakasai, Ifraz Rehman, Hideaki Hata, Raula Gaikovina Kula, MeiyappanNagappan, and Kenichi Matsumoto, How are project-specific forums utilized? A study of participation, content, and sentiment in the Eclipse ecosystem, Empirical Software Engineering, 26(6), 2021

[PS375] Yutaro Kashiwa, Ryoma Nishikawa, Yasutaka Kamei, Masanari Kondo, Emad Shihab, Ryosuke Sato, and Naoyasu Ubayashi, An empirical study on self-admitted technical debt in modern code review, Information and Software Technology,146,2022

[PS376] Yutian Tang, Haoyu Wang, Xian Zhan, Xiapu Luo, Yajin Zhou, Hao Zhou, Qiben Yan, Yulei Sui, Jacky Keung ,”A Systematical Study on Application Performance Management Libraries for Apps. “, IEEE Transactions on Software Engineering, 48 (08) pp, 3044-3065,2022

[PS377] Yuxiang Gao, Yi Zhu, and Yu Zhao, Dealing with imbalanced data for interpretable defect prediction,

[PS378] Yuyang Liu, Ehsan Noei, and Kelly Lyons, How ReadMe files are structured in open source Java projects, Information and Software Technology, 148,2022

[PS379] Zehao Lin, Guodun Li, Jingfeng Zhang, Yue Deng, Xiangji Zeng, Yin Zhang, and Yao Wan. 2022. XCode: Towards Cross-Language Code Representation with Large-Scale Pre-Training. ACM Trans. Softw. Eng. Methodol.

[PS380] Zeinab Azadeh Kermansaravi, Md Saidur Rahman, Foutse Khomh, Fehmi Jaafar, and Yann-Gaël Guéhéneuc, Investigating design anti-pattern and design pattern mutations and their change- and fault-proneness,Empirical Software Engineering,26 (1), 2021

[PS381] Zhaoqiang Guo, Shiran Liu, Jinping Liu, Yanhui Li, Lin Chen, Hongmin Lu, and Yuming Zhou. 2021. How Far Have We Progressed in Identifying Self-admitted Technical Debts? A Comprehensive Empirical Study. ACM Trans. Softw. Eng. Methodol.

[PS382] Zhengliang Li, Zhiwei Jiang, Xiang Chen, Kaibo Cao, and Qing Gu, "Laprob: A Label propagation-Based software bug localization method", Information and Software Technology, Volume 130, 2021

[PS383] Zhenhao Li, Tse-Hsun Chen, Jinqiu Yang, Weiyi Shang ,"Studying Duplicate Logging Statements and Their Relationships With Code Clones " , IEEE Transactions on Software Engineering, 48 (07) pp,2476-2494,2022

[PS384] Zhiding Li, Chenqi Shang, Jianjie Wu, and Yuan Li, Microservice extraction based on knowledge graph from monolithic applications, Information and Software Technology,150,2022

[PS385] Zhifei Chen, Wanwangying Ma, Lin Chen, and Wei Song, Collaboration in software ecosystems: A study of work groups in open environment, Information and Software Technology, 145,2022

[PS386] Zhipeng Gao, Xin Xia, David Lo, and John Grundy. 2021. Technical Q8A Site Answer Recommendation via Question Boosting. ACM Trans. Softw. Eng. Methodol.

[PS387] Zhongxin Liu, Xin Xia, David Lo, Zhenchang Xing, Ahmed E. Hassan, Shanping Li , "Which Variables Should I Log?" , IEEE Transactions on Software Engineering, 47 (09) pp. 2012-2031 ,2021

[PS388] Zhongxing Yu, Chenggang Bai, Lionel Seinturier, Martin Monperrus , "Characterizing the Usage, Evolution and Impact of Java Annotations in Practice", IEEE Transactions on Software Engineering, 47 (05) pp, 969-986 ,2021

[PS389] Zhou Xu, Li Li, Meng Yan, Jin Liu, Xiapu Luo, John Grundy, Yifeng Zhang, Xiaohong Zhang , "A comprehensive comparative study of clustering-based unsupervised defect prediction models." , Journal of Systems and Software, 172 (02) pp. 110862 ,2021

[PS390] Zhou Xu, Tao Zhang, Jacky Keung, Meng Yan, Xiapu Luo, Xiaohong Zhang, Ling Xu,and Yutian Tang, Feature selection and embedding based cross project framework for identifying crashing fault residence, Information and Software Technology, Volume 131, 2021

[PS391] Zhuo Zhang, Yan Lei, Xiaoguang Mao, Meng Yan, Ling Xu,and Xiaohong Zhang, A study of effectiveness of deep learning in locating real faults, Information and Software Technology, Volume 131, 2021

[PS392] Zi Peng, Tse-Hsun Chen, Jinqiu Yang, "Revisiting Test Impact Analysis in Continuous Testing From the Perspective of Code Dependencies ", IEEE Transactions on Software Engineering, 48 (06) pp.1979-1993 ,2022

[PS393] Zijian Jiang, Hao Zhong, Na Meng, "Investigating and recommending co-changed entities for JavaScript programs" , Journal of Systems and Software, 180 (10) pp. 111027 ,2021

[PS394] Ziyi Zhou, Huiqun Yu, Guisheng Fan, Zijie Huang, and Xingguang Yang, Summarizing source code with hierarchical code representation, Information and Software Technology, 143,2022

[PS395] Zubair Khaliq, Sheikh Umar Farooq, and Dawood Ashraf Khan, A deep learning-based automated framework for functional User Interface testing, Information and Software Technology, 150,2022