ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΤΕΧΝΙΚΟ ΧΡΕΟΣ ΣΤΟ ΑΝΟΙΧΤΟ ΛΟΓΙΣΜΙΚΟ

Διπλωματική Εργασία

της

Θωμαής Αδαμίδου

Θεσσαλονίκη, Φεβρουάριος 2024

ΤΕΧΝΙΚΟ ΧΡΕΟΣ ΣΤΟ ΑΝΟΙΧΤΟ ΛΟΓΙΣΜΙΚΟ

Θωμαής Αδαμίδου

Πτυχίο Φυσικής, Πανεπιστήμιο Ιωαννίνων 2022

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων Καθηγητής
Απόστολος Αμπατζόγλου

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 26/02/2024

Ονοματεπώνυμο 1	Ονοματεπώνυμο 2	Ονοματεπώνυμο 3
Απόστολος Αμπατζόγλου	Αλέξανδρος Χατζηγεωργίου	Στυλιανός Ξυνόγαλος

Θωμαή Αδαμίδου

.....

Περίληψη

Η παρούσα έρευνα στοχεύει να εμβαθύνει την κατανόησή μας για το τεχνικό χρέος σε έργα λογισμικού ανοιχτού κώδικα και τις επιπτώσεις του στη βιωσιμότητα του έργου, την ασφάλεια και την ικανοποίηση των χρηστών. Ειδικότερα αναλύεται η συσσώρευση και η διαχείριση του τεχνικού χρέους μέσω μιας ολοκληρωμένης αξιολόγησης διαφόρων εφαρμογών Java ανοιχτού κώδικα πάνω από μια δεκαετία ιστορίας ανάπτυξης χρησιμοποιώντας εργαλεία όπως το SonarQube. Η μελέτη διερευνά τις τάσεις στην εισαγωγή και την επίλυση τεχνικού χρέους, εξετάζοντας τον αντίκτυπό του στην αξιοπιστία, την ποιότητα και την πιθανότητα τεχνολογικών διαταραχών του λογισμικού. Τα ευρήματα υπογραμμίζουν τη σημασία της υιοθέτησης στρατηγικών για την αντιμετώπιση του τεχνικού χρέους, όπως η ευθυγράμμιση με έργα ανοιχτού κώδικα και η εφαρμογή βέλτιστων πρακτικών κωδικοποίησης και διαχείρισης έργων. Αυτή η προσέγγιση όχι μόνο ενισχύει τη συνεχή βελτίωση του οικοσυστήματος ανοιχτού κώδικα, αλλά μειώνει επίσης το μακροπρόθεσμο κόστος και τους κινδύνους που σχετίζονται με την τεχνική χρέος, διασφαλίζοντας έτσι την ακεραιότητα και την αποτελεσματικότητα της ανάπτυξης λογισμικού ανοιχτού κώδικα.

Λέξεις Κλειδιά: τεχνικό χρέος, ανοιχτό λογισμικό, SonarQube

Abstract

This research aims to deepen our understanding of technical debt in open source software projects and its effects on project sustainability, security and user satisfaction. In particular the accumulation and management of technical debt is analyzed through a comprehensive evaluation of various open source Java applications over a decade of development history using tools such as SonarQube. The study explores trends in the introduction and resolution of technical debt, examining its impact on reliability, quality and the potential for software disruption. The findings highlight the importance of adopting strategies to address technical debt, such as aligning with open source projects and implementing coding and project management best practices. This approach not only fosters the continuous improvement of the open source ecosystem, but also reduces the long-term costs and risks associated with technical debt, thus ensuring the integrity and efficiency of open source software development.

Keywords: technical debt, open source, SonarQube

Ευχαριστίες

Με την ολοκλήρωση της μεταπτυχιακής διπλωματικής μου εργασίας, θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες σε όσους συνέβαλλαν στην ολοκλήρωση της.

Ευχαριστώ θερμά τον επιβλέπων καθηγητή μου κύριο Απόστολο Αμπατζόγλου για την εμπιστοσύνη που μου έδειξε στην ανάθεση του θέματος της διπλωματικής εργασίας. Η επιστημονική του καθοδήγηση και το ενδιαφέρον του, καθώς και η πολύτιμη βοήθεια και στήριξη τόσο του ίδιου, όσο και της ομάδας του, ήταν καταλυτικός παράγοντας, ώστε να έρθει εις πέρας η ολοκλήρωση της παρούσης διπλωματικής.

Τέλος, ιδιαίτερες ευχαριστίες θα ήθελα να εκφράσω στην οικογένεια μου για την ανεκτίμητη υποστήριξη τους και την υπομονή που επέδειξαν σε όλη τη διάρκεια των σπουδών μου, όσο και στην παρούσα εργασία.

Πίνακας περιεχομένων

Λίστα εικόνων

Λίστα πινάκων

Ακρωνύμια

API	Application Programming Interface
OSS	Open Source Software
SRP	Single Responsibility Principle
TD	Technical Debt
TDM	Technical Debt Management
XP	Extreme Programming
XSS	Cross-Site Scripting

1 Εισαγωγή

1.1 Τεχνικό χρέος

1.1.1 Η έννοια του τεχνικού χρέους

Στο σημερινό τοπίο ανάπτυξης λογισμικού, τα έργα ανοιχτού κώδικα διαδραματίζουν κρίσιμο ρόλο στις τεχνολογικές πλατφόρμες. Χρησιμοποιούνται ευρέως σε διάφορα στάδια του κύκλου ζωής ανάπτυξης λογισμικού, που χρησιμοποιούνται από βιβλιοθήκες και πλαίσια έως εργαλεία ανάπτυξης, δοκιμών και κατασκευής. Ωστόσο, η εκτεταμένη εξάρτηση από λογισμικό ανοιχτού κώδικα εισάγει μια σημαντική πρόκληση όσον αφορά το τεχνικό χρέος. Το τεχνικό χρέος είναι μια έννοια στην ανάπτυξη λογισμικού που αναφέρεται στο σιωπηρό κόστος ή στη συνέπεια της επιλογής μιας εύκολης ή γρήγορης λύσης αντί μιας καλύτερης, ή πιο ολοκληρωμένης λύσης που θα χρειαζόταν περισσότερο χρόνο για να εφαρμοστεί (Alves et al., 2016). Είναι ανάλογο με το χρηματοοικονομικό χρέος. Ακριβώς όπως το χρηματοοικονομικό χρέος συσσωρεύει τόκους με την πάροδο του χρόνου, το τεχνικό χρέος συγκεντρώνει «τόκους» με τη μορφή πρόσθετης εργασίας, αναποτελεσματικότητας και πιθανών προβλημάτων στη γραμμή εργασιών.

Το τεχνικό χρέος, το οποίο αναφέρεται στις ενδεχόμενες αρνητικές συνέπειες που προκύπτουν από συμβιβασμούς που πραγματοποιούνται κατά την ανάπτυξη λογισμικού, ενέχει κίνδυνο σε έργα ανοιχτού κώδικα. Οι Guo et al. (2016) εντόπισαν διάφορους παράγοντες που συμβάλλουν στο τεχνικό χρέος σε έργα ανοιχτού κώδικα, συμπεριλαμβανομένης της ανεπαρκούς τεκμηρίωσης, της έλλειψης αυτοματοποιημένων δοκιμών και της πολυπλοκότητας του κώδικα. Αυτοί οι παράγοντες μπορεί να οδηγήσουν σε αυξημένο κόστος συντήρησης, μειωμένη ποιότητα λογισμικού και παρεμπόδιση της ανάπτυξης μακροπρόθεσμα. Υπάρχουν διάφορες αιτίες τεχνικού χρέους, όπως:

1. *Χρονικοί περιορισμοί:* Οι προγραμματιστές μπορούν να επιλέξουν να εφαρμόσουν γρήγορες επιδιορθώσεις ή προσωρινές λύσεις για την τήρηση των προθεσμιών, αντί να αφιερώσουν χρόνο για να σχεδιάσουν μια πιο ισχυρή και επεκτάσιμη λύση.

2. *Έλλειψη κατανόησης:* Μερικές φορές οι προγραμματιστές μπορεί να μην κατανοούν πλήρως τις απαιτήσεις ή την αρχιτεκτονική του συστήματος, οδηγώντας σε μη βέλτιστες λύσεις.
3. *Πίεση στις δυνατότητες αποστολής:* Ενδέχεται να υπάρξει πίεση από ενδιαφερόμενους φορείς για γρήγορη παράδοση νέων λειτουργιών, οδηγώντας σε συντομεύσεις που μπορεί να οδηγήσουν σε τεχνικό χρέος.
4. *Κωδικός παλαιού τύπου:* Οι κληρονομιές από παλαιότερα συστήματα ή απαρχαιωμένες τεχνολογίες μπορεί να δημιουργήσουν τεχνικό χρέος, καθώς αυτά τα συστήματα ενδέχεται να απαιτούν ενημερώσεις ή ενοποίηση με σύγχρονες τεχνολογίες.
5. *Κακή επικοινωνία:* Η αναποτελεσματική επικοινωνία μεταξύ των μελών της ομάδας ή μεταξύ διαφορετικών ομάδων μπορεί να οδηγήσει σε παρεξηγήσεις ή παρερμηνείες, με αποτέλεσμα μη βέλτιστες λύσεις.
6. *Scope Creep:* Οι αλλαγές στις απαιτήσεις του έργου ή οι προσθήκες χαρακτηριστικών μπορεί να οδηγήσουν σε βιαστικές υλοποιήσεις, οι οποίες μπορεί να συμβάλουν σε τεχνικό χρέος.

Το τεχνικό χρέος μπορεί να εκδηλωθεί με διάφορες μορφές, όπως ακατάστατο ή περιττό κώδικα, έλλειψη τεκμηρίωσης, αναποτελεσματικοί αλγόριθμοι ή αρχιτεκτονικές και εξάρτηση από ξεπερασμένες τεχνολογίες. Ενώ η ανάληψη τεχνικού χρέους μπορεί μερικές φορές να είναι μια στρατηγική απόφαση για την επίτευξη βραχυπρόθεσμων στόχων, είναι απαραίτητο να το διαχειριστείτε και να το αντιμετωπίσετε με την πάροδο του χρόνου για να αποτρέψετε το να γίνει συντριπτικό και να εμποδίσει τη μακροπρόθεσμη επιτυχία του έργου.

1.1.2 Τύποι τεχνικού χρέους

Το τεχνικό χρέος αναφέρεται στην έννοια των συσσωρευμένων υποχρεώσεων συντήρησης που αντιμετωπίζουν οι ομάδες λογισμικού ως αποτέλεσμα της λήψης συντομεύσεων ή της λήψης πρόσφορων αποφάσεων κατά τη διαδικασία ανάπτυξης (Alves et al., 2016). Αυτές οι υποχρεώσεις διατροφής μπορούν να εκδηλωθούν με διάφορες μορφές ή τύπους τεχνικού χρέους (Klinger et al., 2011). Μερικοί συνήθεις τύποι τεχνικού χρέους περιλαμβάνουν:

1. *Χρέος κώδικα:* Αυτός ο τύπος τεχνικού χρέους προκύπτει όταν οι προγραμματιστές κάνουν συντομεύσεις ή κάνουν συμβιβασμούς στη βάση κωδικών. Παραδείγματα χρέους κώδικα μπορεί να περιλαμβάνουν κακώς γραμμένο ή χωρίς σχολιασμό κώδικα, έλλειψη σωστής διαχείρισης σφαλμάτων και υπερβολική σύζευξη μεταξύ μονάδων ή κλάσεων.
2. *Χρέος σχεδιασμού:* Το χρέος σχεδιασμού αναφέρεται στους συμβιβασμούς ή τις συντομεύσεις που γίνονται στη συνολική αρχιτεκτονική ή σχεδιασμό ενός συστήματος λογισμικού. Παραδείγματα χρέους σχεδιασμού μπορεί να περιλαμβάνουν ανεπαρκή διαχωρισμό των ανησυχιών, στενή σύζευξη μεταξύ εξαρτημάτων και έλλειψη επεκτασιμότητας ή επεκτασιμότητας.
3. *Έλεγχος χρέους:* Ο έλεγχος του χρέους πραγματοποιείται όταν υπάρχει έλλειψη επαρκών δοκιμών ή ανεπαρκής κάλυψη δοκιμής. Παραδείγματα ελέγχου του χρέους μπορεί να περιλαμβάνουν ελλιπείς ή αναποτελεσματικές δοκιμαστικές περιπτώσεις, έλλειψη αυτοματοποιημένων δοκιμών και περιορισμένο έλεγχο παλινδρόμησης.
4. *Χρέος υποδομής*: Το χρέος υποδομής αναφέρεται στη συσσώρευση τεχνικού χρέους στην υποκείμενη υποδομή ή περιβάλλον στο οποίο βασίζεται ένα σύστημα λογισμικού. Παραδείγματα χρέους υποδομής μπορεί να περιλαμβάνουν ξεπερασμένες ή μη υποστηριζόμενες τεχνολογίες, ανεπαρκείς πόρους υλικού και κακώς βελτιστοποιημένες διαμορφώσεις βάσεων δεδομένων.
5. *Χρέος τεκμηρίωσης:* Το χρέος τεκμηρίωσης προκύπτει όταν υπάρχει ανεπαρκής ή ξεπερασμένη τεκμηρίωση για το σύστημα λογισμικού. Παραδείγματα χρέους τεκμηρίωσης μπορεί να περιλαμβάνουν έλλειψη εγχειριδίων ή οδηγών χρήστη, παλιά τεκμηρίωση API και ελλιπή τεκμηρίωση αρχιτεκτονικής συστήματος.
6. *Χρέος ανάπτυξης:* Το χρέος ανάπτυξης αναφέρεται στο τεχνικό χρέος που προκύπτει κατά τη διαδικασία ανάπτυξης ή κυκλοφορίας ενός συστήματος λογισμικού. Παραδείγματα χρέους ανάπτυξης μπορεί να περιλαμβάνουν μη αυτόματες και επιρρεπείς σε σφάλματα διαδικασίες ανάπτυξης, έλλειψη αυτοματοποιημένων αγωγών κατασκευής και ανάπτυξης και ασυνεπείς διαμορφώσεις περιβάλλοντος.
7. *Χρέος συντήρησης:* Το χρέος διατροφής αναφέρεται στο τεχνικό χρέος που συσσωρεύεται ως αποτέλεσμα αναβληθέντων ή παραμελημένων εργασιών συντήρησης. Παραδείγματα χρέους συντήρησης μπορεί να περιλαμβάνουν

καθυστερημένες διορθώσεις σφαλμάτων, παραμελημένες ενημερώσεις ασφαλείας και αναβληθείσες βελτιστοποιήσεις απόδοσης (Tan et al., 2021). Συνολικά, το τεχνικό χρέος μπορεί να λάβει διάφορες μορφές και να συσσωρευτεί σε διαφορετικούς τομείς ανάπτυξης λογισμικού, όπως κώδικα, σχεδιασμό, δοκιμή, υποδομή, τεκμηρίωση, ανάπτυξη και συντήρηση.

1.1.3 Διαχείριση τεχνικού χρέους

Η διαχείριση τεχνικού χρέους αναφέρεται στη διαδικασία εντοπισμού, αξιολόγησης, ιεράρχησης και μετριασμού του τεχνικού χρέους στην ανάπτυξη λογισμικού (Seaman & Guo, 2011). Περιλαμβάνει την ενεργή διαχείριση και αντιμετώπιση των συσσωρευμένων συμβιβασμών, συντομεύσεων και ελλείψεων στην αρχιτεκτονική, το σχεδιασμό, τον κώδικα, την τεκμηρίωση, τη δοκιμή, την υποδομή και την ανάπτυξη ενός συστήματος λογισμικού. Η διαχείριση του τεχνικού χρέους είναι μια κρίσιμη διαδικασία που επιτρέπει στις ομάδες ανάπτυξης να διατηρήσουν έναν ισορροπημένο ρυθμό μεταξύ της ταχείας καινοτομίας και της μακροπρόθεσμης υγείας ενός συστήματος λογισμικού. Ακολουθούν τα βασικά βήματα που εμπλέκονται στην τεχνική διαχείριση του χρέους (Brown et al., 2010):

1. Ταυτοποίηση

Το πρώτο βήμα για τη διαχείριση του τεχνικού χρέους είναι ο ακριβής προσδιορισμός του εντός του συστήματος λογισμικού. Αυτό περιλαμβάνει τη διεξαγωγή ολοκληρωμένων ελέγχων κώδικα, αρχιτεκτονικές αξιολογήσεις και τη χρήση εργαλείων που μπορούν να βοηθήσουν στον εντοπισμό περιοχών υψηλής πολυπλοκότητας, κακής απόδοσης ή μη τήρησης προτύπων κωδικοποίησης.

2. Αξιολόγηση

Μόλις εντοπιστεί, κάθε περίπτωση τεχνικού χρέους πρέπει να αξιολογηθεί ως προς τον αντίκτυπό του στο σύστημα, συμπεριλαμβανομένων των πιθανών κινδύνων, της προσπάθειας που απαιτείται για την αποκατάσταση και των επιπτώσεών του σε μελλοντικές αναπτυξιακές δραστηριότητες. Αυτό το βήμα βοηθά στην ιεράρχηση του τεχνικού χρέους με βάση τη σοβαρότητα και τους τεχνικούς ή επιχειρηματικούς κινδύνους που εγκυμονεί.

3. Προτεραιότητα

Δεν είναι όλα τα τεχνικά χρέη ίσα. Ορισμένα ενδέχεται να επηρεάσουν κρίσιμα την απόδοση ή την ασφάλεια του συστήματος, ενώ άλλα μπορεί να έχουν μόνο μικρό αντίκτυπο. Η ιεράρχηση του τεχνικού χρέους περιλαμβάνει την αξιολόγηση των αντισταθμίσεων μεταξύ άμεσων διορθώσεων έναντι μακροπρόθεσμων λύσεων και ευθυγράμμιση των προσπαθειών αποκατάστασης με τους επιχειρηματικούς στόχους και τους οδικούς χάρτες προϊόντων.

4. Μετριασμός

Ο μετριασμός του τεχνικού χρέους περιλαμβάνει τη λήψη μέτρων για την αντιμετώπισή του. Αυτό μπορεί να περιλαμβάνει ανακατασκευή κώδικα, βελτίωση τεκμηρίωσης, βελτίωση πρακτικών δοκιμών, ενημέρωση ή αντικατάσταση υποδομής και βελτίωση των διαδικασιών ανάπτυξης. Οι στρατηγικές μετριασμού θα πρέπει να εφαρμόζονται με βάση την ιεράρχηση προτεραιοτήτων για να διασφαλιστεί ότι οι πόροι βελτιστοποιούνται και ο αντίκτυπος της μείωσης του χρέους μεγιστοποιείται.

5. Πρόληψη

Ενώ η διαχείριση του υφιστάμενου τεχνικού χρέους είναι ζωτικής σημασίας, η αποτροπή της συσσώρευσης νέου τεχνικού χρέους είναι εξίσου σημαντική. Η θέσπιση ισχυρών προτύπων κωδικοποίησης, η διασφάλιση επαρκούς αρχιτεκτονικού σχεδιασμού, η ενσωμάτωση συνεχών δοκιμών και η καλλιέργεια μιας κουλτούρας ποιότητας μπορεί να βοηθήσει στην ελαχιστοποίηση της δημιουργίας νέου τεχνικού χρέους.

6. Παρακολούθηση

Η συνεχής παρακολούθηση του συστήματος λογισμικού για την εμφάνιση νέων τεχνικών χρεών είναι απαραίτητη. Οι τακτικές προγραμματισμένες αναθεωρήσεις, η αυτοματοποιημένη επεξεργασία εργαλείων και ο καθορισμός βασικών δεικτών απόδοσης που σχετίζονται με την ποιότητα του κώδικα και την αρχιτεκτονική του συστήματος μπορούν να βοηθήσουν στη διατήρηση της επαγρύπνησης έναντι του νέου χρέους.

Ειδικότερα, οι βέλτιστες πρακτικές στην τεχνική διαχείριση χρέους περιλαμβάνουν:

1. *Τακτικά προγραμματισμένες αναθεωρήσεις:* Οι περιοδικές αναθεωρήσεις του κώδικα και της αρχιτεκτονικής του συστήματος μπορούν να βοηθήσουν στον έγκαιρο εντοπισμό και αξιολόγηση του τεχνικού χρέους.
2. *Αυτοματοποιημένη εργαλεία:* Η χρήση εργαλείων που μπορούν να αυτοματοποιήσουν τον εντοπισμό πιθανών τεχνικών χρεών (π.χ. εργαλεία

ανάλυσης στατικού κώδικα) μπορεί να βοηθήσει σημαντικά στη διαδικασία αναγνώρισης.

3. *Ισορροπία μεταξύ βραχυπρόθεσμων και μακροπρόθεσμων στόχων:* Η διασφάλιση ότι οι προσπάθειες διαχείρισης τεχνικού χρέους δεν καταπνίγουν εντελώς την καινοτομία ή η παροχή νέων χαρακτηριστικών είναι ζωτικής σημασίας. Η εύρεση της σωστής ισορροπίας είναι το κλειδί.
4. *Επικοινωνία με τα ενδιαφερόμενα μέρη*: Η ενημέρωση όλων των ενδιαφερομένων, από τις ομάδες ανάπτυξης έως τους ηγέτες επιχειρήσεων, σχετικά με την κατάσταση του τεχνικού χρέους και τις στρατηγικές για τη διαχείρισή του είναι απαραίτητη για την ευθυγράμμιση των προτεραιοτήτων του οργανισμού.
5. *Συνεχής βελτίωση:* Η υιοθέτηση μιας νοοτροπίας συνεχούς βελτίωσης και μάθησης από προηγούμενες περιπτώσεις τεχνικού χρέους μπορεί να βοηθήσει στη βελτίωση των διαδικασιών και στην πρόληψη του μελλοντικού χρέους.

Η αποτελεσματική τεχνική διαχείριση του χρέους είναι μια ολοκληρωμένη στρατηγική που περιλαμβάνει προσδιορισμό, αξιολόγηση, ιεράρχηση προτεραιοτήτων και αποκατάσταση, με παράλληλες προσπάθειες για την πρόληψη της συσσώρευσης νέου χρέους. Με την ενσωμάτωση αυτών των διαδικασιών στον κύκλο ζωής ανάπτυξης λογισμικού, οι ομάδες μπορούν να διασφαλίσουν ότι τα προϊόντα τους παραμένουν καινοτόμα, αποτελεσματικά και διατηρούμενα με την πάροδο του χρόνου.

1.2 Ανοιχτό λογισμικό

Το Λογισμικό Ανοιχτού Κώδικα (OSS-Open Source Software) αναφέρεται σε λογισμικό υπολογιστή με τον πηγαίο κώδικα του οποίου είναι διαθέσιμος και αδειοδοτημένος με τρόπο που επιτρέπει στους χρήστες να μελετούν, να τροποποιούν και να διανέμουν ελεύθερα το λογισμικό. Αυτό σημαίνει ότι ο πηγαίος κώδικας είναι προσβάσιμος σε οποιονδήποτε επιθυμεί να τον δει ή να τον τροποποιήσει, αντί να παραμείνει αποκλειστικός και αποκλεισμένος από τους αρχικούς προγραμματιστές. Το Λογισμικό Ανοιχτού Κώδικα αναφέρεται σε λογισμικό που παρέχει στους χρήστες την ελευθερία να χρησιμοποιούν, να τροποποιούν, να διανέμουν και να μοιράζονται ανοιχτά τον πηγαίο κώδικα του λογισμικού (Seaman & Guo, 2011). Αυτό σημαίνει ότι ο καθένας μπορεί να έχει πρόσβαση και να τροποποιήσει τον υποκείμενο κώδικα του λογισμικού,

επιτρέποντας μεγαλύτερη διαφάνεια, συνεργασία και προσαρμογή. Συνήθως αναπτύσσεται από μια κοινότητα εθελοντών και διατίθεται στο κοινό χωρίς κόστος. Μερικά παραδείγματα λογισμικού ανοιχτού κώδικα περιλαμβάνουν το λειτουργικό σύστημα Linux, τον διακομιστή ιστού Apache και το πρόγραμμα περιήγησης ιστού Firefox. Τα βασικά χαρακτηριστικά του λογισμικού ανοιχτού κώδικα περιλαμβάνουν (Seaman & Guo, 2011):

1. *Πρόσβαση στον πηγαίο κώδικα*: Οι χρήστες έχουν πρόσβαση στον υποκείμενο πηγαίο κώδικα του λογισμικού, επιτρέποντάς τους να κατανοήσουν πώς λειτουργεί, να κάνουν τροποποιήσεις και να τον προσαρμόσουν σύμφωνα με τις ανάγκες τους.
2. *Διανομή:* Το λογισμικό ανοιχτού κώδικα μπορεί να διανεμηθεί ελεύθερα, επιτρέποντας σε οποιονδήποτε να το μοιραστεί με άλλους χωρίς περιορισμούς.
3. *Δυνατότητα τροποποίησης:* Οι χρήστες επιτρέπεται να τροποποιήσουν το λογισμικό για να προσθέσουν νέες δυνατότητες, να διορθώσουν σφάλματα ή να βελτιώσουν την απόδοση, συμβάλλοντας συχνά στην κοινότητα.
4. *Συνεργασία κοινότητας:* Τα έργα ανοιχτού κώδικα έχουν συχνά μια κοινότητα προγραμματιστών και χρηστών που συνεργάζονται για την ανάπτυξη, τη δοκιμή, την τεκμηρίωση και την υποστήριξη του λογισμικού.
5. *Διαφάνεια:* Η διαδικασία ανάπτυξης και η λήψη αποφάσεων σε έργα ανοιχτού κώδικα είναι συχνά διαφανείς, με συζητήσεις, αλλαγές κώδικα και ζητήματα που παρακολουθούνται δημόσια.
6. *Άδεια χρήσης:* Το λογισμικό ανοιχτού κώδικα διανέμεται συνήθως με άδειες που έχουν εγκριθεί από την Πρωτοβουλία Ανοικτού Κώδικα (OSI), όπως η Γενική Άδεια Δημόσιας Χρήσης GNU (GPL), Άδεια MIT, Άδεια χρήσης Apache κ.λπ. Αυτές οι άδειες διασφαλίζουν ότι το λογισμικό παραμένει ανοιχτό και δωρεάν για χρήση και τροποποίηση από άλλους, ενώ παρέχει επίσης νομική προστασία και οδηγίες για τη διανομή και τη χρήση του.

Το λογισμικό ανοιχτού κώδικα γίνεται όλο και πιο δημοφιλές λόγω της συνεργατικής φύσης, της ευελιξίας και της ικανότητάς του να ενθαρρύνει την καινοτομία. Πολλά γνωστά έργα λογισμικού, συμπεριλαμβανομένου του λειτουργικού συστήματος Linux, του διακομιστή ιστού Apache, του προγράμματος περιήγησης ιστού Firefox και του συστήματος διαχείρισης περιεχομένου WordPress, είναι ανοιχτού κώδικα. Επιπλέον,

οι αρχές ανοιχτού κώδικα έχουν υιοθετηθεί σε διάφορους κλάδους πέρα από την ανάπτυξη λογισμικού, συμπεριλαμβανομένου του υλικού, των δεδομένων, ακόμη και της διακυβέρνησης.

1.3 Στόχος εργασίας

Σκοπός της παρούσας έρευνας είναι η διερεύνηση του τεχνικού χρέους σε ανοιχτό λογισμικό Ο στόχος της μελέτης είναι να αναλύσει και να κατανοήσει τους παράγοντες που συμβάλλουν στο τεχνικό χρέος σε έργα ανοιχτού κώδικα και τον αντίκτυπό τους στη βιωσιμότητα του έργου. Με αυτό τον τρόπο θα παρέχει πληροφορίες και συστάσεις για προγραμματιστές και συντηρητές έργων σχετικά με τη διαχείριση του τεχνικού χρέους σε έργα ανοιχτού κώδικα, τον μετριασμό των κινδύνων που σχετίζονται με αυτό και τη διασφάλιση της μακροπρόθεσμης βιωσιμότητας του λογισμικού.

Ειδικότερα, η έρευνα εστιάζει στην ανάλυση του τεχνικού χρέους σε έργα ανοιχτού λογισμικού, προερχόμενα από πέντε διακεκριμένα Foundations: Apache, Eclipse, Oracle, Open Cluster, και Code Libs. Καταγράφτηκαν 50 project από το κάθε Foundation και αναλύθηκαν σε σύνολο από όλα τα Foundation 250 έργα. Για κάθε ένα από τα εξεταζόμενα έργα, καταγράφηκαν διάφορες μετρήσεις και δημιουργήθηκε ένα Data set. Αναλυτικότερα καταγράφτηκαν τα εξής: Quality Gate, Lines of Code, Reliability, Maintainability, Security, Security Review, Coverage, Duplications, Technical Debt, καθώς και τέσσερις διαστάσεις του Clean Code: Consistency, Intentionality, Adaptability, και Responsibility. Επίσης, αξιολογήθηκε η σοβαρότητα (Severity) του τεχνικού χρέους σε τρεις βαθμίδες: High, Medium, και Low.

Ειδικότερα, η έρευνα επιχειρεί να κατανοήσει τη φύση, τον αντίκτυπο και τη διαχείριση του τεχνικού χρέους σε λογισμικό ανοιχτού κώδικα. Οι ερευνητικοί στόχοι της μελέτης είναι οι ακόλουθοι:

- Ο έλεγχος της ενδεχόμενης ύπαρξης σημαντικής σχέσης ανάμεσα στο τεχνικό χρέος και τη συνέπεια, την πρόθεση, την προσαρμοστικότητα και την ευθύνη.
- Η ύπαρξη σημαντικής συσχέτισης μεταξύ τεχνικού χρέους και της αξιοπιστίας.
- Η ύπαρξη σημαντικής συσχέτισης μεταξύ τεχνικού χρέους και της συντήρησης και ασφάλειας.

2 Σχετικές εργασίες & γενικές πληροφορίες

2.1 Σχετικές εργασίες

Στο παρόν κεφάλαιο εξετάζονται έρευνες που αναλύουν διάφορες μετρήσεις και παράγοντες για να ποσοτικοποιήσουν την έκταση του τεχνικού χρέους σε έργα λογισμικού ανοιχτού κώδικα. Λαμβάνουν υπόψη πτυχές όπως η πολυπλοκότητα του κώδικα, η αντιγραφή κώδικα, οι μυρωδιές κώδικα και ο αριθμός των ανοιχτών ζητημάτων και σφαλμάτων. Αξιοποιώντας αυτές τις μετρήσεις, οι ερευνητές μπορούν να εκτιμήσουν το ποσό του τεχνικού χρέους που έχει συσσωρευτεί σε ένα έργο και να εντοπίσουν τομείς που απαιτούν άμεση προσοχή (Seaman & Guo, 2011). Επίσης, η κατανόηση του αντίκτυπου του τεχνικού χρέους σε έργα λογισμικού ανοιχτού κώδικα επιτρέπει στους προγραμματιστές και τους συντηρητές έργων να δώσουν προτεραιότητα στις προσπάθειές τους και να κατανείμουν τους πόρους αποτελεσματικά. Μπορούν να επικεντρωθούν στην αντιμετώπιση των περιοχών με το υψηλότερο τεχνικό χρέος, όπως ο κώδικας που είναι υπερβολικά περίπλοκος ή επιρρεπής σε σφάλματα, προκειμένου να βελτιώσει τη συνολική ποιότητα και τη δυνατότητα συντήρησης του λογισμικού. Συνολικά, η διαχείριση του τεχνικού χρέους σε λογισμικό ανοιχτού κώδικα απαιτεί ολοκληρωμένη κατανόηση των επιπτώσεών του και ανάπτυξη στρατηγικών για τον μετριασμό των επιπτώσεών του. Ειδικότερα, παρουσιάζονται μελέτες που έχουν διεξαχθεί για την ανάλυση και την ποσοτικοποίηση του τεχνικού χρέους στο πλαίσιο του λογισμικού ανοιχτού κώδικα και συνέβαλλαν στην κατανόηση στο πλαίσιο αυτού.

Η έρευνα των Ampatzoglou et al. (2019) εστίασε στη σημασία της προσοχής κατά την αναδιαμόρφωση του πηγαίου κώδικα με βάση τις προτάσεις εργαλείων, τονίζοντας ότι οι ανακατασκευές σε επίπεδο αρχιτεκτονικής είναι ασφαλέστερες από αυτές σε επίπεδο πηγαίου κώδικα. Ειδικότερα, διερεύνησαν την εφαρμογή της Αρχής Ενιαίας Ευθύνης (SRP-Single Responsibility Principle) στην ανάπτυξη λογισμικού για την ενίσχυση της αρθρωτής δομής, με μελέτες που πραγματοποιήθηκαν σε συστήματα λογισμικού μεγάλης κλίμακας. Τα αποτελέσματα υποδεικνύουν ότι οι ανακατασκευές που βασίζονται σε SRP επηρεάζουν θετικά τη σπονδυλωτή, με παρατηρούμενες αντισταθμίσεις μεταξύ σύζευξης και συνοχής σε επίπεδο κατηγορίας. Τα βασικά ερευνητικά τους ερωτήματα επικεντρώθηκαν στα εξής: «*Πώς υποστηρίζουν τα αποτελέσματα της μελέτης την ιδέα ότι οι ανακατασκευές σε επίπεδο αρχιτεκτονικής είναι ασφαλέστερες από τις ανακατασκευές σε*

επίπεδο πηγαίου κώδικα;», «*Ποιες συγκεκριμένες γνώσεις παρέχουν οι μελέτες στους επαγγελματίες και τους ερευνητές σχετικά με την εφαρμογή της αρχής της ενιαίας ευθύνης (SRP) στην ανάπτυξη λογισμικού;*», «*Με ποιους τρόπους οι παρατηρούμενες αντισταθμίσεις μεταξύ σύζευξης και συνοχής σε επίπεδο τάξης επηρεάζουν τη διαδικασία λήψης αποφάσεων για τους προγραμματιστές λογισμικού κατά την εφαρμογή ανακατασκευών που βασίζονται σε SRP;*». Τα αποτελέσματα της μελέτης υποδηλώνουν ότι οι ανακατασκευές σε επίπεδο αρχιτεκτονικής φαίνεται να είναι πιο ασφαλείς από αυτές σε επίπεδο πηγαίου κώδικα. Η μελέτη διαπίστωσε ότι σε επίπεδο αρχιτεκτονικής, υπάρχει μεγαλύτερη πιθανότητα να αυξηθεί ένα ακίνητο ποιότητας χωρίς να επηρεαστεί το άλλο, υποδεικνύοντας μια πιο θετική επίδραση συνολικά. Επιπλέον, η μελέτη παρατήρησε ότι σε επίπεδο αρχιτεκτονικής, υπήρχαν μόνο θετικά και περιορισμένα ουδέτερα αποτελέσματα στην αρθρωτή δομή, υποστηρίζοντας περαιτέρω την ιδέα ότι οι ανακατασκευές σε επίπεδο αρχιτεκτονικής είναι λιγότερο πιθανό να έχουν αρνητικές συνέπειες σε σύγκριση με τις ανακατασκευές σε επίπεδο πηγαίου κώδικα.

Ακόμη σύμφωνα με τα αποτελέσματα των Ampatzoglou et al. (2019), οι μελέτες παρέχουν πολύτιμες γνώσεις για επαγγελματίες και ερευνητές σχετικά με την εφαρμογή της αρχής της ενιαίας ευθύνης (SRP) στην ανάπτυξη λογισμικού. Υπογραμμίζουν τα πλεονεκτήματα της εφαρμογής του SRP για τη βελτίωση της σπονδυλωτικότητας του λογισμικού και τη μείωση των μυρωδιών όπως η Long Method, η God Class και το Large Package. Τα ευρήματα υποδηλώνουν ότι η συμμόρφωση με το SRP μπορεί να οδηγήσει σε αυξημένη συνοχή και μειωμένη σύζευξη, ενισχύοντας τελικά τη δομική ποιότητα των συστημάτων λογισμικού. Επιπλέον, οι μελέτες υπογραμμίζουν τη σημασία της εξέτασης του επιπέδου ευκρίνειας στο οποίο εφαρμόζονται οι ανακατασκευές, με τις ανακατασκευές σε επίπεδο αρχιτεκτονικής να δείχνουν πιο θετικά αποτελέσματα στην αρθρωτή δομή σε σύγκριση με τις ανακατασκευές σε επίπεδο μεθόδου και τάξης. Αυτές οι πληροφορίες μπορούν να βοηθήσουν τους επαγγελματίες να λαμβάνουν τεκμηριωμένες αποφάσεις όταν εφαρμόζουν ανακατασκευές για τη βελτίωση της ποιότητας και της δυνατότητας συντήρησης του λογισμικού. Κατανοώντας τον αντίκτυπο των ανακατασκευών που βασίζονται σε SRP στην αρθρωτή βαθμίδα σε διαφορετικά επίπεδα ευαισθησίας, οι επαγγελματίες μπορούν να δώσουν προτεραιότητα και να σχεδιάσουν αποτελεσματικά τις ανακατασκευές για να επιτύχουν τα επιθυμητά αποτελέσματα σε έργα ανάπτυξης λογισμικού. Οι ερευνητές μπορούν επίσης να επωφεληθούν από αυτά τα ευρήματα αποκτώντας μια βαθύτερη κατανόηση της σχέσης μεταξύ της συμμόρφωσης με

το SRP και της αρθρωτής δομής λογισμικού, συμβάλλοντας στην πρόοδο των αρχών και πρακτικών μηχανικής λογισμικού. Ως εκ τούτου, οι μελέτες παρέχουν πρακτική καθοδήγηση και θεωρητικές γνώσεις για επαγγελματίες και ερευνητές σχετικά με την εφαρμογή της αρχής της ενιαίας ευθύνης στην ανάπτυξη λογισμικού.

Μια έρευνα που διεξήχθη από τους Ampatzoglou et al. (2015) ανέλυσε την οικονομική πτυχή της διαχείρισης τεχνικού χρέους στην ανάπτυξη λογισμικού. Αναλυτικότερα, εισήγαγε ένα γλωσσάρι χρηματοοικονομικών όρων που σχετίζονται με το τεχνικό χρέος και ταξινομεί τις χρηματοοικονομικές προσεγγίσεις που χρησιμοποιούνται στην τεχνική διαχείριση του χρέους. Η μελέτη διεξήγαγε μια συστηματική βιβλιογραφική ανασκόπηση για να αναλύσει την υπάρχουσα έρευνα σχετικά με τις οικονομικές πτυχές του τεχνικού χρέους και προσδιόρισε τη σημασία της διεπιστημονικής συνεργασίας, των οικονομικών προσεγγίσεων και των νέων τεχνικών σε αυτόν τον τομέα. Τα κύρια ερευνητικά ερωτήματα κινήθηκαν στο εξής πλαίσιο: «*Ποιοι είναι ορισμένοι κοινοί χρηματοοικονομικοί όροι που χρησιμοποιούνται σε συζητήσεις που σχετίζονται με την τεχνική διαχείριση του χρέους;*», «*Πώς συμβάλλουν οικονομικές προσεγγίσεις όπως η ανάλυση κόστους/οφέλους και η διαχείριση χαρτοφυλακίου στην αποτελεσματική διαχείριση του τεχνικού χρέους στην ανάπτυξη λογισμικού;*», «*Ποια είναι τα βασικά ευρήματα σχετικά με την έμφαση στη μηχανική λογισμικού έναντι των χρηματοοικονομικών προσεγγίσεων στην έρευνα που σχετίζεται με την τεχνική διαχείριση του χρέους*;». Σύμφωνα με τα αποτελέσματα μερικοί κοινοί χρηματοοικονομικοί όροι που χρησιμοποιούνται σε συζητήσεις σχετικά με τη διαχείριση τεχνικής χρέους περιλαμβάνουν κεφάλαιο, τόκος, κόστος, απόδοση επένδυσης, περιουσιακό στοιχείο, επένδυση, κίνδυνος, παρούσα αξία, παραγωγικότητα, υποχρέωση, έσοδα και καθαρή παρούσα αξία. Ακόμη, τα βασικά ευρήματα σχετικά με την έμφαση στη μηχανική λογισμικού έναντι των χρηματοοικονομικών προσεγγίσεων στην έρευνα που σχετίζεται με την τεχνική διαχείριση του χρέους είναι τα εξής. Περίπου το 56% των μελετών που εξέτασαν το τεχνικό χρέος τονίζουν την πτυχή της μηχανικής λογισμικού χωρίς συγκεκριμένη εστίαση σε καμία οικονομική πτυχή. Περίπου το 22% των μελετών εξισορροπεί τόσο τη μηχανική λογισμικού όσο και τις οικονομικές πτυχές χρησιμοποιώντας οικονομικές προσεγγίσεις για τη διαχείριση τεχνικού χρέους ενώ χρησιμοποιούν τεχνολογίες μηχανικής λογισμικού. Ένας περιορισμένος αριθμός μελετών, περίπου 10%, τονίζουν κυρίως τη χρήση οικονομικών προσεγγίσεων στην τεχνική διαχείριση του χρέους. Αυτά τα ευρήματα

υπογραμμίζουν τη σημασία της εξισορρόπησης τόσο της μηχανικής λογισμικού όσο και των οικονομικών πτυχών για την αποτελεσματική διαχείριση του τεχνικού χρέους.

Υπό αυτό το πλαίσιο, η διαχείριση του τεχνικού χρέους σε λογισμικό ανοιχτού κώδικα είναι ζωτικής σημασίας λόγω των πιθανών οικονομικών του επιπτώσεων, του αυξημένου κόστους ανάπτυξης και συντήρησης, της μειωμένης αξιοπιστίας και ποιότητας του λογισμικού και της υψηλότερης πιθανότητας τεχνολογικών διαταραχών. Συμπερασματικά, η διαχείριση του τεχνικού χρέους σε λογισμικό ανοιχτού κώδικα απαιτεί ολοκληρωμένη κατανόηση του αντικτύπου του και ανάπτυξη στρατηγικών για τον μετριασμό των επιπτώσεών του. Η διαχείριση του τεχνικού χρέους σε λογισμικό ανοιχτού κώδικα απαιτεί ολοκληρωμένη κατανόηση του αντικτύπου του και ανάπτυξη στρατηγικών για τον μετριασμό των επιπτώσεών του. Η ανάπτυξη στρατηγικών για τον μετριασμό των επιπτώσεων του τεχνικού χρέους σε λογισμικό ανοιχτού κώδικα είναι ζωτικής σημασίας λόγω των πιθανών οικονομικών του επιπτώσεων, του αυξημένου κόστους ανάπτυξης και συντήρησης, της μειωμένης αξιοπιστίας και ποιότητας του λογισμικού και της υψηλότερης πιθανότητας τεχνολογικών διαταραχών.

Η εργασία των Li et al. (2015), παρουσίασε μια μελέτη συστηματικής χαρτογράφησης για το τεχνικό χρέος (TD-Technical Debt) και τη διαχείρισή του στην ανάπτυξη λογισμικού. Η μελέτη αξιολόγησε την ποιότητα 94 πρωτογενών μελετών με βάση συγκεκριμένα κριτήρια, εξήγαγε δεδομένα από κάθε μελέτη και συνέθεσε τις πληροφορίες για να απαντήσει σε ερευνητικά ερωτήματα. Τα αποτελέσματα έδειξαν μέση βαθμολογία ποιότητας 3,17 από 5,00 για τις επιλεγμένες μελέτες, με τις περισσότερες να εμπίπτουν σε ένα μέτριο εύρος ποιότητας. Η μελέτη επικεντρώθηκε στο τεχνικό χρέος στη μηχανική λογισμικού, συμπεριλαμβανομένων διαφόρων πτυχών, όπως οι τύποι TD, τα διακυβευμένα χαρακτηριστικά ποιότητας, οι δραστηριότητες διαχείρισης TD, οι προσεγγίσεις και τα εργαλεία που χρησιμοποιούνται στο TDM. Τα βασικά ερωτήματα εστίασαν στο «*Ποιες ήταν οι βασικές δραστηριότητες και προσεγγίσεις που επισημάνθηκαν στις επιλεγμένες μελέτες για τη διαχείριση του τεχνικού χρέους;*», «*Πώς κατανεμήθηκαν οι επιλεγμένες μελέτες σε διαφορετικούς τύπους συγγραφέων, τόπους δημοσίευσης και έτη δημοσίευσης;*», «*Ποιοι ήταν οι περιορισμοί της τεχνικής μεταφοράς του χρέους που συζητήθηκε στην εργασία και πόσες μελέτες αντιμετώπισαν αυτούς τους περιορισμούς;*». Συμπερασματικά, οι επιλεγμένες μελέτες υπογράμμισαν αρκετές βασικές δραστηριότητες και προσεγγίσεις για τη διαχείριση του τεχνικού χρέους. Αυτές περιλαμβάνουν τις εξής:

Προσδιορισμό και ταξινόμηση των τύπων τεχνικού χρέους: Οι μελέτες ταξινόμησαν το τεχνικό χρέος σε 10 διαφορετικούς τύπους για να παρέχουν μια δομημένη κατανόηση των διαφόρων μορφών τεχνικού χρέους. *Δραστηριότητες διαχείρισης τεχνικού χρέους (TDM-Technical Debt Management):* Προσδιορίστηκαν οκτώ δραστηριότητες TDM, υπογραμμίζοντας τη σημασία της ενεργής διαχείρισης τεχνικού χρέους για την πρόληψη μακροπρόθεσμων αρνητικών επιπτώσεων στα συστήματα λογισμικού. *Εργαλεία για τη διαχείριση τεχνικού χρέους:* Οι μελέτες συγκέντρωσαν 29 εργαλεία ειδικά σχεδιασμένα για τη διαχείριση τεχνικού χρέους, υπογραμμίζοντας την ανάγκη για ειδικά εργαλεία για την αντιμετώπιση διαφορετικών τύπων τεχνικού χρέους σε όλη τη διαδικασία διαχείρισης. *Εμπειρικές μελέτες και αποδεικτικά στοιχεία υψηλής ποιότητας:* Υπάρχει έκκληση για περισσότερες εμπειρικές μελέτες με αποδεικτικά στοιχεία υψηλής ποιότητας σχετικά με ολόκληρη τη διαδικασία τεχνικής διαχείρισης χρέους και την εφαρμογή ειδικών προσεγγίσεων σε βιομηχανικά περιβάλλοντα. *Εξελιγμένα και αποκλειστικά εργαλεία TDM:* Οι μελέτες τόνισαν την ανάγκη για πιο εξελιγμένα και εξειδικευμένα εργαλεία για την αποτελεσματική διαχείριση των διαφόρων τύπων τεχνικών χρεών που μπορεί να προκύψουν κατά τη διαδικασία ανάπτυξης λογισμικού.

Παράλληλα, οι περιορισμοί της τεχνικής μεταφοράς του χρέους που εξετάστηκαν περιλάμβαναν τα ακόλουθα σημεία, όπως ανεπάρκεια στην περιγραφή των σύγχρονων αναπτυξιακών προσεγγίσεων. Μια μελέτη υποστήριξε ότι η τεχνική αλληγορία του χρέους δεν επαρκεί για να περιγράψει συστήματα που ενσωματώνονται σε σύγχρονες προσεγγίσεις ανάπτυξης όπως ο Extreme Programming (XP), επειδή το χρέος που προκύπτει σε αυτές τις προσεγγίσεις συνήθως αποπληρώνεται σε στο εγγύς μέλλον, με αποτέλεσμα σχετικά μικρό συσσωρευμένο χρέος. Αδυναμίες της μεταφοράς του τεχνικού χρέους. Μια άλλη μελέτη ανέδειξε αρκετές σημαντικές ελλείψεις της μεταφοράς του τεχνικού χρέους, συμπεριλαμβανομένης της έλλειψης μιας τυπικής μονάδας μέτρησης και της εξάρτησης του ποσού των τόκων που πρέπει να επιστραφεί από τη μελλοντική ανάπτυξη που επηρεάζεται από το χρέος. Μόνο τρεις μελέτες αντιμετώπισαν τα όρια της τεχνικής μεταφοράς του χρέους, παρέχοντας πληροφορίες σχετικά με την ανεπάρκειά του σε ορισμένα πλαίσια και τις ελλείψεις του όσον αφορά τη μέτρηση και την εκτίμηση επιπτώσεων.

Η συστηματική μελέτη χαρτογράφησης των Alves et al. (2016) παρείχε πολύτιμες γνώσεις για την τρέχουσα κατάσταση της έρευνας για το τεχνικό χρέος και προσέφερε καθοδήγηση τόσο για ερευνητές όσο και για τους επαγγελματίες. Τα βασικά ερευνητικά

ερωτήματα ήταν «*Ποιοι είναι ορισμένοι κοινοί μηχανισμοί που προτείνονται για τον εντοπισμό τεχνικού χρέους σε έργα λογισμικού;*», «*Πώς επηρεάζει η τεχνική μεταφορά του χρέους τη συντήρηση του λογισμικού όσον αφορά την παραγωγικότητα και το κόστος;*», «*Ποιοι είναι οι βασικοί στόχοι αυτής της μελέτης και πώς μπορούν τα ευρήματα να εφαρμοστούν στην πράξη;*». Σύμφωνα με τα συμπεράσματα προτάθηκαν αρκετοί κοινοί μηχανισμοί για τον εντοπισμό τεχνικού χρέους σε έργα λογισμικού. Μερικοί από αυτούς τους μηχανισμούς περιλαμβάνουν το χρέος σχεδίασης. Αυτός ο τύπος χρέους προσδιορίζεται αναλύοντας τον πηγαίο κώδικα και ανιχνεύοντας παραβιάσεις καλών αρχών σχεδιασμού, όπως υπερβολικά μεγάλες ή στενά συζευγμένες κλάσεις. Το αρχιτεκτονικό χρέος που επικεντρώνεται σε ζητήματα εντός της αρχιτεκτονικής του προϊόντος, όπως παραβιάσεις της αρθρωτής δομής, που μπορεί να επηρεάσουν αρχιτεκτονικές απαιτήσεις όπως η απόδοση και η ευρωστία. Η αντιμετώπιση του χρέους της αρχιτεκτονικής απαιτεί συχνά πιο εκτεταμένες προσπάθειες ανάπτυξης πέρα από απλές παρεμβάσεις κώδικα. Το χρέος τεκμηρίωσης περιλαμβάνει προβλήματα που σχετίζονται με ανεπαρκή ή ελλιπή τεκμηρίωση έργου λογισμικού. Αυτός ο τύπος χρέους μπορεί να αναγνωριστεί από έλλειψη ή ανεπαρκή τεκμηρίωση σε διάφορα αντικείμενα έργων. Το δοκιμαστικό χρέος αναφέρεται σε ζητήματα σε δραστηριότητες δοκιμών που μπορούν να θέσουν σε κίνδυνο την ποιότητα των διαδικασιών δοκιμών. Τα παραδείγματα περιλαμβάνουν προγραμματισμένες δοκιμές που δεν εκτελέστηκαν ή ελλείψεις στη σουίτα δοκιμών, όπως χαμηλή κάλυψη κώδικα. Το χρέος κώδικα επικεντρώνεται σε προβλήματα στον πηγαίο κώδικα που εμποδίζουν την αναγνωσιμότητα και τη συντήρηση του κώδικα. Το χρέος κώδικα συνήθως προσδιορίζεται με την εξέταση του κώδικα για ζητήματα που προκύπτουν από κακές πρακτικές κωδικοποίησης. Το χρέος ελαττώματος σχετίζεται με γνωστά ελαττώματα στο λογισμικό, τα οποία συχνά ανακαλύπτονται κατά τη διάρκεια δοκιμών ή αναφέρονται από χρήστες. Η αντιμετώπιση του χρέους ελαττωμάτων περιλαμβάνει την επίλυση εντοπισμένων ζητημάτων για τη βελτίωση της ποιότητας του λογισμικού. Αυτοί οι μηχανισμοί βοηθούν τις ομάδες λογισμικού να αναγνωρίζουν διαφορετικούς τύπους τεχνικών χρεών στα έργα τους, δίνοντάς τους τη δυνατότητα να ιεραρχήσουν και να αντιμετωπίσουν αποτελεσματικά αυτά τα ζητήματα για να διατηρήσουν την ποιότητα του λογισμικού και να μειώσουν τις μελλοντικές προκλήσεις συντήρησης.

Η έρευνα των Guo et al. (2016), εμβάθυνε στον αντίκτυπο της διαχείρισης του τεχνικού χρέους σε έργα λογισμικού, παρέχοντας πολύτιμες πληροφορίες στους

διαχειριστές λογισμικού ώστε να λαμβάνουν τεκμηριωμένες αποφάσεις. Η μελέτη εστίασε στο να εξετάσει τα βραχυπρόθεσμα οφέλη από τη δημιουργία τεχνικού χρέους σε ένα έργο λογισμικού, και τις διαφορά της προτεινόμενης προσέγγισης για τη διαχείριση του τεχνικού χρέους από τις τρέχουσες σιωπηρές μεθόδους. Ενώ με βάση τα ευρήματα σύγκρινε το κόστος παρακολούθησης και παρακολούθησης τεχνικού χρέους στο τρέχον έργο λογισμικού. Σύμφωνα με τα αποτελέσματα, η προτεινόμενη προσέγγιση για τη διαχείριση του τεχνικού χρέους διαφέρει από τις τρέχουσες σιωπηρές μεθόδους με τη ρητή αναγνώριση. Η προτεινόμενη προσέγγιση προωθεί τον ρητό προσδιορισμό τεχνικού χρέους εντός του έργου λογισμικού, διασφαλίζοντας ότι όλες οι περιπτώσεις τεχνικού χρέους αναγνωρίζονται και αναγνωρίζονται. Με τη μέτρηση και παρακολούθηση, η προσέγγιση δίνει έμφαση στη μέτρηση και την παρακολούθηση του τεχνικού χρέους, παρέχοντας στους διαχειριστές λογισμικού μετρήσιμα δεδομένα για τη λήψη τεκμηριωμένων αποφάσεων. Με την ενημερωμένη λήψη αποφάσεων παρέχοντας σκληρά δεδομένα που συλλέγονται μέσω της σωστής μέτρησης, η προσέγγιση επιτρέπει στους διαχειριστές λογισμικού να λαμβάνουν αποφάσεις με βάση στοιχεία αντί να βασίζονται αποκλειστικά στην εμπειρία ή το ένστικτο. Μέσω της εστίασης στο κόστος και τα οφέλη. Η προσέγγιση στοχεύει να διερευνήσει τόσο το κόστος όσο και τα οφέλη της διαχείρισης τεχνικού χρέους, βοηθώντας τους διαχειριστές λογισμικού να κατανοήσουν τις συνέπειες των αποφάσεών τους. Τέλος, με την μακροπρόθεσμη προοπτική, αναγνωρίζοντας τον μακροπρόθεσμο αντίκτυπο του τεχνικού χρέους, η προσέγγιση ενθαρρύνει την προληπτική διαχείριση για την πρόληψη σοβαρών ολισθήσεων χρονοδιαγράμματος, υπερβάσεις προϋπολογισμού και αποτυχίες έργων. Συνολικά, η προτεινόμενη προσέγγιση υποστηρίζει μια πιο δομημένη και βασισμένη σε δεδομένα μέθοδο διαχείρισης του τεχνικού χρέους, με στόχο τον μετριασμό των κινδύνων και τη βελτίωση της συνολικής επιτυχίας των έργων λογισμικού.

Τα βασικά ευρήματα σχετικά με το κόστος παρακολούθησης και παρακολούθησης τεχνικού χρέους στο τρέχον έργο λογισμικού περιλαμβάνουν το κόστος διαχείρισης τεχνικού χρέους στο έργο λογισμικού προήλθε από δραστηριότητες όπως η αναγνώριση, η ανάλυση, η αξιολόγηση, η επικοινωνία και η τεκμηρίωση. Η ανάλυση και αξιολόγηση στοιχείων τεχνικού χρέους αναδείχθηκε ως η πιο χρονοβόρα και εντατική δραστηριότητα πόρων, συμβάλλοντας σημαντικά στο συνολικό κόστος διαχείρισης τεχνικού χρέους. Οι προγραμματιστές εντόπισαν τεχνουργήματα λογισμικού χαμηλής ποιότητας που θεωρήθηκαν ως περιπτώσεις τεχνικού χρέους από τις ενότητες για τις οποίες ήταν

υπεύθυνοι, υπογραμμίζοντας τη σημασία των προληπτικών πρακτικών αναγνώρισης. Δραστηριότητες όπως η ενημέρωση της λίστας τεχνικών χρεών, η τεκμηρίωση αποφάσεων σχετικά με στοιχεία τεχνικού χρέους και η επικοινωνία με τα μέλη της ομάδας ήταν βασικά στοιχεία για την παρακολούθηση και την παρακολούθηση του τεχνικού χρέους. Παράλληλα, η μελέτη αποκάλυψε ένα σημαντικό κόστος εκκίνησης που σχετίζεται με την παρακολούθηση και την παρακολούθηση του τεχνικού χρέους, υποδεικνύοντας μια αρχική επένδυση που απαιτείται για τη δημιουργία αποτελεσματικών πρακτικών τεχνικής διαχείρισης χρέους. Ενώ υπήρχε ένα αρχικό κόστος εκκίνησης, το κόστος της συνεχιζόμενης διαχείρισης του τεχνικού χρέους παρατηρήθηκε να μειώνεται σε πολύ λογικά επίπεδα με την πάροδο του χρόνου, υποδηλώνοντας κέρδη αποτελεσματικότητας καθώς ωρίμαζε η διαδικασία. Αυτά τα ευρήματα ρίχνουν φως στην κατανομή πόρων, τη δέσμευση χρόνου και τις δραστηριότητες που εμπλέκονται στην παρακολούθηση και παρακολούθηση του τεχνικού χρέους σε έργα λογισμικού, τονίζοντας τη σημασία των συστηματικών προσεγγίσεων διαχείρισης για την αποτελεσματική αντιμετώπιση του τεχνικού χρέους.

Η μελέτη των Jesus & Melo, (2017), ερεύνησε τους παράγοντες που συμβάλλουν στη συσσώρευση τεχνικού χρέους λογισμικού χρησιμοποιώντας στατιστική ανάλυση δεδομένων που συλλέγονται από πενήντα έργα σε δημόσιο χώρο αποθήκευσης λογισμικού. Οι συγγραφείς εξέτασαν τη σχέση μεταξύ του μεγέθους του έργου, του αριθμού των συνεργατών, της ταχύτητας ανάπτυξης και του αριθμού των δεσμεύσεων για τη διόρθωση ελαττωμάτων με το ποσό του τεχνικού χρέους στα έργα που αναλύθηκαν. Χρησιμοποίησαν το SonarQube για τη μέτρηση του τεχνικού χρέους και διαπίστωσαν ότι το μέγεθος του έργου και ο αριθμός των συνεργατών σχετίζονται σημαντικά με το ποσό του τεχνικού χρέους. Ωστόσο, ο αριθμός των δεσμεύσεων που έγιναν για τη διόρθωση ελαττωμάτων δεν έδειξε ισχυρή συσχέτιση με το ύψος του τεχνικού χρέους. Η μελέτη εντόπισε ευκαιρίες για περαιτέρω έρευνα για την κατανόηση του αντίκτυπου των πτυχών του έργου στη συσσώρευση τεχνικού χρέους και πρότεινε τη συμπερίληψη πρόσθετων μεταβλητών όπως η κάλυψη δοκιμής, η εμπειρία των συνεργατών, το πλαίσιο και η μεθοδολογία που χρησιμοποιείται για την ανάπτυξη λογισμικού. Οι συγγραφείς τόνισαν επίσης την ανάγκη για διερεύνηση ποιοτικών μεθόδων, όπως μελέτες περιπτώσεων, για την επιβεβαίωση των ποσοτικών ευρημάτων και συνέστησαν να διερευνηθεί πώς πτυχές του σχεδιασμού λογισμικού συμβάλλουν στη συσσώρευση τόκων στο τεχνικό χρέος. Η μελέτη παρέχει πολύτιμες γνώσεις σχετικά με τους παράγοντες που επηρεάζουν τη

συσσώρευση τεχνικού χρέους και προσφέρει ευκαιρίες για μελλοντική έρευνα σε αυτόν τον τομέα. Συμπερασματικά, εξήχθη ότι μεγαλύτερα έργα με περισσότερους συνεργάτες και ταχύτερους κύκλους ανάπτυξης τείνουν να συσσωρεύουν περισσότερο τεχνικό χρέος, τονίζοντας τη σημασία της διαχείρισης τεχνικού χρέους σε έργα λογισμικού ανοιχτού κώδικα, ανεξάρτητα από το μέγεθος και την ταχύτητα ανάπτυξής τους.

2.2 Γενικές πληροφορίες

Κατά την εξέταση της συσχέτισης μεταξύ τεχνικού χρέους και χρηστικότητας δεδομένων, γίνεται προφανές ότι ο αντίκτυπος των ελαττωμάτων στην ποιότητα του λογισμικού εκτείνεται πέρα από τις άμεσες τεχνικές πτυχές. Τα τρέχοντα εμπειρικά μοντέλα έχουν επικεντρωθεί κυρίως στην παροχή αποδεικτικών στοιχείων για τις αρνητικές επιπτώσεις του τεχνικού χρέους στην απόδοση, αλλά δεν λαμβάνουν πλήρως υπόψη το υποκείμενο πρόβλημα βελτιστοποίησης - τους συμβιβασμούς μεταξύ του κόστους του χρέους και των ωφελειών που προκύπτουν από την ανάληψη τεχνικού χρέους (Madeyski, 2020).

Παράλληλα, οι έμμεσες οικονομικές επιπτώσεις του τεχνικού χρέους, όπως το χαμηλό ηθικό και ο κύκλος εργασιών των προγραμματιστών, δεν είναι πάντα άμεσα εμφανείς, αλλά ενδεχομένως να παρουσιάζουν μακροχρόνιες επιπτώσεις. Για παράδειγμα, το τεχνικό χρέος που οδηγεί σε ζητήματα ποιότητας δεδομένων και ασυνέπειες μπορεί να έχει αρνητικές επιπτώσεις στις διαδικασίες λήψης αποφάσεων, εμποδίζοντας την ακριβή και έγκαιρη ανάλυση. Επιπλέον, το τεχνικό χρέος μπορεί επίσης να εμποδίσει την προσβασιμότητα και την κατανόηση των δεδομένων (Rantala & Mäntylä, 2020). Συνεπώς, οι οργανισμοί πρέπει να εξισορροπήσουν την αυστηρότητα της αντιμετώπισης του τεχνικού χρέους με τη χρηστικότητα των δεδομένων τους.

Με τον εντοπισμό και τη διαχείριση του τεχνικού χρέους σε όλο τον κύκλο ζωής ανάπτυξης του λογισμικού, οι οργανισμοί μπορούν να αποτρέψουν τη συσσώρευση αυξανόμενου χρέους και να μετριάσουν τις αρνητικές συνέπειες στη χρηστικότητα των δεδομένων (Zhang & Wu, 2016).

Αυτή η προσέγγιση θα περιλαμβάνει όχι μόνο την ιεράρχηση της επίλυσης του τεχνικού χρέους, αλλά και την εξέταση των επιπτώσεών του στη συνέπεια, την ποιότητα, την προσβασιμότητα και την κατανόηση των δεδομένων. Αντιμετωπίζοντας το τεχνικό χρέος και διασφαλίζοντας την ποιότητα λογισμικού, οι οργανισμοί μπορούν να

βελτιώσουν τη χρηστικότητα των δεδομένων, ελαχιστοποιώντας την εμφάνιση ελαττωμάτων λογισμικού που μπορούν να επηρεάσουν την ποιότητα, τη συνέπεια, την προσβασιμότητα και την ερμηνεία των δεδομένων. Σύμφωνα με τους Rantala & Mäntylä (2020), οι διαφορετικοί τύποι τεχνικού χρέους, όπως χρέος κωδικού, χρέος σχεδιασμού και αρχιτεκτονικής, περιβαλλοντικό χρέος, χρέος διανομής γνώσεων και τεκμηρίωσης και έλεγχος του χρέους, μπορούν να συμβάλουν στην υποβάθμιση της συντηρησιμότητας. Επομένως, οι οργανισμοί πρέπει να υιοθετήσουν αποτελεσματικές τεχνικές και μεθόδους για τη διαχείριση του τεχνικού χρέους από την άποψη της αρχιτεκτονικής λογισμικού (Madeyski, 2020). Αυτό περιλαμβάνει τον εντοπισμό και την ιεράρχηση του τεχνικού χρέους με βάση μετρήσεις όπως η τάση για σφάλματα, η τάση αλλαγής, οι μυρωδιές κώδικα (code smells)[1] και οι παραβιάσεις κώδικα (Lenarduzzi et al., 2021). Ακόμη, η βελτίωση του κόστους ανάπτυξης με την πάροδο του χρόνου χωρίς διακοπή της διαδικασίας, μπορεί να αποτελέσει μια λύση για την αποτελεσματική διαχείριση του τεχνικού χρέους. Εξετάζοντας τους συμβιβασμούς μεταξύ του κόστους και των οφελών του τεχνικού χρέους, οι οργανισμοί δύναται να λάβουν τεκμηριωμένες αποφάσεις σχετικά με το πότε και πώς να το αναλάβουν.

Αυτή η προσέγγιση θα βελτιώσει τελικά την ποιότητα του λογισμικού, θα βελτιώσει τις διαδικασίες λήψης αποφάσεων και θα αυξήσει τη συνολική χρηστικότητα των δεδομένων σε οργανισμούς (Rantala & Mäntylä, 2020).

Αν και είναι σημαντικό να δοθεί προτεραιότητα στην επίλυση του τεχνικού χρέους και να ληφθεί υπόψη ο αντίκτυπός του στη συνέπεια, την ποιότητα, την προσβασιμότητα και την κατανόηση των δεδομένων, είναι σημαντικό να ληφθεί υπόψη και το αντίθετο επιχείρημα. Ένα επιχείρημα κατά της προτεραιότητας της τεχνικής επίλυσης χρεών είναι ότι μπορεί να εκτρέψει τους πόρους και την προσοχή από την προσθήκη νέων λειτουργιών ή λειτουργιών που μπορούν να ωφελήσουν άμεσα τους χρήστες (Borocki et al., 2016). Σε μια ανταγωνιστική αγορά, οι οργανισμοί πρέπει να καινοτομούν συνεχώς και να παρέχουν νέα χαρακτηριστικά για να παραμείνουν μπροστά. Η προτεραιότητα στην τεχνική επίλυση χρεών μπορεί να επιβραδύνει την κυκλοφορία νέων χαρακτηριστικών, επηρεάζοντας δυνητικά την ικανότητα του οργανισμού να ανταποκρίνεται στις απαιτήσεις της αγοράς

[1] Στον προγραμματισμό υπολογιστών, η μυρωδιά κώδικα (code smells) είναι οποιοδήποτε χαρακτηριστικό στον πηγαίο κώδικα ενός προγράμματος που πιθανώς υποδεικνύει ένα βαθύτερο πρόβλημα.

και τις προσδοκίες των πελατών. Ωστόσο, αυτό το επιχείρημα παραβλέπει τις μακροπρόθεσμες επιπτώσεις του τεχνικού χρέους στη χρηστικότητα των δεδομένων.

Με την προληπτική διαχείριση του τεχνικού χρέους, οι οργανισμοί μπορούν να εξασφαλίσουν τη μακροπρόθεσμη χρηστικότητα και τη δυνατότητα συντήρησης των δεδομένων και των συστημάτων λογισμικού τους (Lenarduzzi et al., 2021).

Αυτή η προληπτική προσέγγιση μπορεί να ελαχιστοποιήσει τη συσσώρευση τεχνικού χρέους, μειώνοντας έτσι τον αντίκτυπο στη χρηστικότητα των δεδομένων και στην ποιότητα του λογισμικού μακροπρόθεσμα. Επιπλέον, λαμβάνοντας υπόψη τις αρνητικές επιπτώσεις των ευέλικτων διαδικασιών, εάν δεν εφαρμοστούν σωστά σε όλη τη ροή αξίας της ανάπτυξης προϊόντων, οι οργανισμοί μπορούν να αποφύγουν τη δημιουργία τεχνικού χρέους που έχει τη δυνατότητα να εμποδίσει τη χρηστικότητα των δεδομένων (Malakuti & Heuschkel, 2021). Για να ξεπεραστούν αυτές οι προκλήσεις και να βελτιστοποιηθεί η διαχείριση του τεχνικού χρέους, είναι σημαντικό για τους οργανισμούς να υιοθετήσουν μια πολυδιάστατη προσέγγιση. Αυτή η προσέγγιση θα πρέπει να λαμβάνει υπόψη τους συμβιβασμούς μεταξύ του κόστους του τεχνικού χρέους και των οφελών που προκύπτουν από την ανάληψη αυτού. Με αυτόν τον τρόπο, οι οργανισμοί μπορούν να λαμβάνουν τεκμηριωμένες αποφάσεις σχετικά με την ιεράρχηση της τεχνικής επίλυσης χρεών με βάση τον αντίκτυπό της στη χρηστικότητα των δεδομένων και την ποιότητα του λογισμικού. Τέλος, οι οργανισμοί θα πρέπει επίσης να λαμβάνουν υπόψη το επίπεδο ωριμότητας της υποδομής, των μεθόδων και των ικανοτήτων τους κατά την εφαρμογή τεχνικών διαχείρισης χρέους (Karamitsos et al., 2020). Αυτό θα βοηθήσει να διασφαλιστεί ότι οι επιλεγμένες τεχνικές είναι εφικτές και αποτελεσματικές στην πράξη. Συμπερασματικά, τα ακριβή και χρησιμοποιήσιμα δεδομένα είναι απαραίτητα για τη λήψη τεκμηριωμένων αποφάσεων στον σύγχρονο κόσμο.

Το τεχνικό χρέος μπορεί να έχει σημαντικές συνέπειες στην ευχρηστία των δεδομένων. Πρώτον, η συσσώρευση τεχνικού χρέους μπορεί να οδηγήσει σε μείωση της ποιότητας των δεδομένων. Καθώς συσσωρεύεται τεχνικό χρέος, καθίσταται πιο δύσκολη η διατήρηση και η ενημέρωση συστημάτων δεδομένων, με αποτέλεσμα πιθανά σφάλματα, ασυνέπειες και ελλιπή δεδομένα. Αυτά τα ζητήματα ποιότητας δεδομένων μπορούν να υπονομεύσουν την αξιοπιστία των δεδομένων, επηρεάζοντας τελικά τη χρηστικότητά τους για σκοπούς λήψης αποφάσεων και ανάλυσης (Nilsson et al., 2013).

Συγχρόνως, το τεχνικό χρέος μπορεί να εμποδίσει την ενοποίηση και τη διαλειτουργικότητα των δεδομένων. Καθώς συσσωρεύεται το τεχνικό χρέος, αυξάνεται η

πολυπλοκότητα και η ευθραυστότητα των συστημάτων δεδομένων. Αυτό μπορεί να καταστήσει δύσκολη την ενοποίηση των δεδομένων από διαφορετικές πηγές ή συστήματα, οδηγώντας σε σιλό δεδομένων[2] και περιορισμένη διαλειτουργικότητα (Strandberg et al., 2020). Ως αποτέλεσμα, η ευχρηστία των δεδομένων μπορεί να τεθεί σε κίνδυνο, καθώς οι οργανισμοί μπορεί να δυσκολεύονται να αποκτήσουν πρόσβαση και να συνδυάσουν σχετικά δεδομένα για ανάλυση και λήψη αποφάσεων.

Επιπρόσθετα, το τεχνικό χρέος μπορεί να επηρεάσει την επεκτασιμότητα και την απόδοση των συστημάτων δεδομένων. Καθώς το τεχνικό χρέος συσσωρεύεται, η υποκείμενη αρχιτεκτονική και υποδομή λογισμικού μπορεί να γίνονται όλο και πιο αναποτελεσματική και ξεπερασμένη. Αυτό συνεπάγεται με πιο αργούς χρόνους επεξεργασίας και ανάκτησης δεδομένων, περιορισμένη χωρητικότητα αποθήκευσης και δυσκολίες στο χειρισμό μεγάλου όγκου δεδομένων. Αυτά τα ζητήματα απόδοσης μπορούν να εμποδίσουν τη χρηστικότητα των δεδομένων αυξάνοντας τον χρόνο που απαιτείται για την πρόσβαση και την ανάλυση δεδομένων, περιορίζοντας την ικανότητα διαχείρισης αναλυτικών ή μεγάλων δεδομένων, επηρεάζοντας τη συνολική εμπειρία του χρήστη (Stephenson et al., 2010).

Από την άλλη πλευρά, είναι σημαντικό να εξεταστεί το αντίθετο επιχείρημα ότι το τεχνικό χρέος μπορεί να μην έχει πάντα άμεσες και έμμεσες αρνητικές επιπτώσεις στην ευχρηστία των δεδομένων. Αν και είναι αλήθεια ότι το τεχνικό χρέος μπορεί να οδηγήσει σε αυξημένη πολυπλοκότητα και ευθραυστότητα στα συστήματα δεδομένων, είναι σημαντικό να αναγνωριστεί ότι σε ορισμένες περιπτώσεις, τα βραχυπρόθεσμα οφέλη από την ανάληψη τεχνικού χρέους μπορεί να υπερβαίνουν τις μακροπρόθεσμες συνέπειες (Biase et al., 2019).

Για παράδειγμα, σε ταχύρρυθμους και ανταγωνιστικούς κλάδους, ο χρόνος για την αγορά νέων προϊόντων και υπηρεσιών μπορεί να είναι κρίσιμος παράγοντας για την επιτυχία ενός οργανισμού.

Σε αυτές τις περιπτώσεις, η κυκλοφορία λογισμικού με τεχνικό χρέος μπορεί να είναι μια στρατηγική απόφαση για την τήρηση των προθεσμιών και την παραμονή μπροστά από τον ανταγωνισμό (Stanko et al., 2014). Ωστόσο, είναι σημαντικό να αναγνωρίσουμε ότι αυτή η προσέγγιση μπορεί να έχει επιπτώσεις στην ευχρηστία των

[2] Ένα σιλό δεδομένων είναι μια συλλογή πληροφοριών που απομονώνονται από έναν οργανισμό και είναι απρόσιτα σε όλα τα μέρη μιας εταιρικής ιεραρχίας.

δεδομένων μακροπρόθεσμα. Το απλήρωτο τεχνικό χρέος μπορεί να οδηγήσει σε αυξανόμενο χρέος και επιβράδυνση της ανάπτυξης, καθώς συσσωρεύεται σε όλο τον κύκλο ζωής ανάπτυξης λογισμικού (Zhang & Wu, 2016). Αυτό μπορεί να έχει ως αποτέλεσμα καθυστερημένες εργασίες συντήρησης και παρεμπόδιση της προόδου, επηρεάζοντας τελικά τη χρηστικότητα των συστημάτων δεδομένων. Για την αποτελεσματική διαχείριση του τεχνικού χρέους και τη διασφάλιση της χρηστικότητας των δεδομένων, είναι απαραίτητο να δημιουργηθούν στρατηγικές και διαδικασίες που δίνουν προτεραιότητα στον εντοπισμό, την αξιολόγηση και τον μετριασμό του τεχνικού χρέους κατά τη διαδικασία ανάπτυξης (Lenarduzzi et al., 2021).

Στρατηγικές για τη διαχείριση τεχνικού χρέους, όπως η επισημοποίηση της σχέσης μεταξύ κόστους και οφέλους και η ιεράρχηση με βάση μετρήσεις όπως μυρωδιές κώδικα (code smells) και παραβιάσεις, μπορούν να συμβάλουν στη βελτίωση της ποιότητας του λογισμικού και της λήψης αποφάσεων κατά τη διάρκεια των δραστηριοτήτων συντήρησης. Επιπλέον, τα εμπειρικά μοντέλα τεχνικού χρέους τονίζουν τις αρνητικές επιπτώσεις που μπορεί να έχει στην απόδοση και παρέχουν πλαίσια αποφάσεων για την παρακολούθηση και την αποφυγή υπερβολικής συσσώρευσης τεχνικού χρέους (Ramasubbu & Kemerer, 2013).

Παρόλα αυτά, τα μοντέλα ενδέχεται να μην λαμβάνουν πλήρως υπόψη τους συμβιβασμούς μεταξύ του κόστους και των οφελών από τη δημιουργία τεχνικού χρέους. Παράλληλα, είναι σημαντικό να εξισορροπηθεί η αυστηρότητα με τη χρηστικότητα των μεθόδων εκτίμησης κατά την αξιολόγηση των οικονομικών επιπτώσεων του τεχνικού χρέους (Lenarduzzi et al., 2021). Αυτό οφείλεται στο γεγονός ότι οι υπερβολικά περίπλοκες ή χρονοβόρες μέθοδοι εκτίμησης μπορεί να αποτρέψουν τους οργανισμούς από την ακριβή αξιολόγηση του αντίκτυπου του τεχνικού χρέους στη χρηστικότητα των δεδομένων.

Στο πλαίσιο της διαχείρισης λογισμικού, η κατανόηση των οικονομικών επιπτώσεων του τεχνικού χρέους είναι ζωτικής σημασίας για τη λήψη τεκμηριωμένων αποφάσεων. Σύμφωνα με τους Fernández-Sánchez et al. (2017), το τεχνικό χρέος συχνά παραβλέπει την κοινωνική πτυχή αυτού του φαινομένου, η οποία είναι απαραίτητη για μια συνολική κατανόηση. Συνεπώς, είναι επιτακτική ανάγκη να ληφθεί υπόψη η κοινωνικο-τεχνική φύση του τεχνικού χρέους προκειμένου να αντιμετωπιστεί αποτελεσματικά η πολυπλοκότητά του. Ενσωματώνοντας τις προοπτικές των προγραμματιστών, των διαχειριστών και άλλων ενδιαφερομένων, οι οργανισμοί μπορούν να αναπτύξουν

στρατηγικές για τη μείωση του τεχνικού χρέους που λαμβάνουν υπόψη όχι μόνο το οικονομικό κόστος, αλλά και τον αντίκτυπο στο ηθικό της ομάδας και τις συνολικές πρακτικές ανάπτυξης λογισμικού (Karamitsos et al., 2020).

Η μέτρηση των οικονομικών οφελών των κοινοτικών εφαρμογών και των επενδύσεων σε πληροφοριακά συστήματα περιλαμβάνει επίσημη ανάλυση κόστους-οφέλους, παρόμοια με την παρακολούθηση του παραγωγικού χρόνου και την άμεση συσχέτισή του με θετικές αλλαγές στις οικονομικές μετρήσεις του οργανισμού.

Για να μειώσουν το τεχνικό χρέος και να βελτιώσουν τη χρηστικότητα των δεδομένων, οι οργανισμοί μπορούν να εφαρμόσουν διάφορες μεθόδους, όπως:

1. Υιοθέτηση μιας συστηματικής τεχνικής προσέγγισης διαχείρισης χρέους που επικεντρώνεται στον εντοπισμό και την ιεράρχηση του αρχιτεκτονικού χρέους (Malakuti & Heuschkel, 2021).
2. Εφαρμογή αυστηρών διαδικασιών για την αξιολόγηση και την αντιμετώπιση του τεχνικού χρέους, συμπεριλαμβανομένης της χρήσης μετρήσεων όπως μυρωδιές κώδικα (smell codes) και παραβιάσεις, ώστε να δοθεί προτεραιότητα σε τομείς προς βελτίωση.
3. Η διαχείριση του τεχνικού χρέους μέσω στρατηγικών όπως η επισημοποίηση της σχέσης μεταξύ κόστους και οφέλους, καθώς και η ιεράρχηση με βάση μετρήσεις μπορεί να συμβάλει στη βελτίωση της ποιότητας του λογισμικού και στη λήψη αποφάσεων κατά τη διάρκεια των δραστηριοτήτων συντήρησης (Lenarduzzi et al., 2021).
4. Η βελτιστοποίηση του κόστους ανάπτυξης με την πάροδο του χρόνου χωρίς να σταματήσει η διαδικασία ανάπτυξης, μπορεί να αποτελέσει μια βιώσιμη λύση για τη διαχείριση του τεχνικού χρέους.

Για να διασφαλιστεί η χρηστικότητα των δεδομένων και να αποφευχθεί η υπερβολική συσσώρευση τεχνικού χρέους, οι οργανισμοί θα πρέπει να υιοθετήσουν μια συστηματική προσέγγιση για τη διαχείριση του τεχνικού χρέους. Αυτό περιλαμβάνει τη διεξαγωγή μιας ανάλυσης κατάστασης για την κατανόηση της τρέχουσας κατάστασης του τεχνικού χρέους, τη διαμόρφωση μιας στρατηγικής για την αντιμετώπισή του, την εφαρμογή της στρατηγικής μέσω επαναληπτικών βελτιώσεων στο λογισμικό και την αξιολόγηση και τον έλεγχο της αποτελεσματικότητας της εφαρμοσμένης στρατηγικής (Dyda et al., 2021).

Αυτή η ολοκληρωμένη προσέγγιση στην τεχνική διαχείριση του χρέους όχι μόνο επιτρέπει στους οργανισμούς να ιεραρχούν και να αντιμετωπίζουν ζητήματα που μπορεί να επηρεάσουν τη χρηστικότητα των δεδομένων, αλλά βοηθά επίσης στη λήψη τεκμηριωμένων αποφάσεων και στη βελτίωση της συνολικής ποιότητας του λογισμικού (Lenarduzzi et al., 2021). Συνοψίζοντας, η διαχείριση του τεχνικού χρέους και η βελτίωση της χρηστικότητας των δεδομένων απαιτεί μια ολοκληρωμένη προσέγγιση που λαμβάνει υπόψη τις κοινωνικές και τεχνικές πτυχές του τεχνικού χρέους.

Από την άλλη πλευρά, αν και είναι σημαντικό να αναγνωριστούν οι τεχνικές πτυχές του τεχνικού χρέους, δεν πρέπει να παραλειφθούν οι σημαντικές κοινωνικές επιπτώσεις που παρουσιάζει. Η εστίαση σε μετρήσεις όπως οι μυρωδιές κώδικα και οι παραβιάσεις μπορεί να αντιμετωπιστούν τεχνικά, αλλά δεν λαμβάνουν υπόψη το ανθρώπινο στοιχείο κατά την ανάπτυξη του λογισμικού. Ο αρνητικός αντίκτυπος του τεχνικού χρέους στο ηθικό της ομάδας ανάπτυξης και η πιθανότητα τζίρου των προγραμματιστών λόγω συστημικών προβλημάτων δεν πρέπει να υποτιμάται (Li et al., 2022).

Παράλληλα, η έμφαση στην παρακολούθηση του κόστους και του οφέλους μπορεί να υπεραπλουστεύσει την πολυπλοκότητα της διαχείρισης του τεχνικού χρέους. Είναι σημαντικό να αναγνωριστεί πως οι οικονομικές επιπτώσεις του τεχνικού χρέους δεν είναι πάντα άμεσες. Για παράδειγμα, το χαμηλό ηθικό που προκαλείται από το τεχνικό χρέος θα μπορούσε να οδηγήσει σε μειωμένη παραγωγικότητα και ποιότητα, καταλήγοντας τελικά σε υψηλότερο κόστος μακροπρόθεσμα (Fernández-Sánchez et al., 2017).

Για να αντιμετωπιστεί η πολύπλευρη φύση του τεχνικού χρέους, είναι κεντρικής σημασίας να υιοθετηθεί μια ολιστική προσέγγιση που να λαμβάνει υπόψη τόσο τις κοινωνικές όσο και τις τεχνικές πτυχές. Ενώ οι μετρήσεις και οι οικονομικές επιπτώσεις παρέχουν πολύτιμες γνώσεις, προσφέρουν μόνο μερική άποψη του συνολικού αντίκτυπου του τεχνικού χρέους (Spínola et al., 2013).

Ακόμη, οι κοινωνικές επιπτώσεις του τεχνικού χρέους, όπως το χαμηλό ηθικό και ο κύκλος εργασιών προγραμματιστών, υπογραμμίζουν τη σημασία του να λαμβάνεται υπόψη το ανθρώπινο στοιχείο στην ανάπτυξη λογισμικού. Είναι σημαντικό να αναγνωριστεί ότι το τεχνικό χρέος δεν είναι μόνο ένα τεχνικό ζήτημα, αλλά ένα κοινωνικο-τεχνικό ζήτημα που απαιτεί μια συνολική προσέγγιση για την αντιμετώπιση της πολυπλοκότητάς του. Αυτή η ολοκληρωμένη προσέγγιση περιλαμβάνει την ανάπτυξη καλύτερων τρόπων εντοπισμού, παρακολούθησης, κατηγοριοποίησης, μέτρησης,

ιεράρχησης και εξόφλησης του τεχνικού χρέους. Επίσης, περιλαμβάνει την προώθηση της συνεργασίας και της επικοινωνίας μεταξύ του ακαδημαϊκού κόσμου, της βιομηχανίας και των προμηθευτών εργαλείων για την συλλογική αντιμετώπιση των προκλήσεων που σχετίζονται με την τεχνική διαχείριση του χρέους (Aniche et al., 2019).

Συμπερασματικά, η διαχείριση του τεχνικού χρέους θα πρέπει να λαμβάνει υπόψη τη διττή φύση αυτού του προβλήματος, που να περιλαμβάνει τόσο κοινωνικές όσο και τεχνικές επιπτώσεις. Ενώ οι μετρήσεις και οι οικονομικοί παράγοντες παίζουν σημαντικό ρόλο, δεν παρέχουν πλήρη κατανόηση του συνολικού αντίκτυπου του τεχνικού χρέους. Είναι σημαντικό να αναγνωριστείς η σημαντική επίδραση του τεχνικού χρέους στη συμπεριφορά της ομάδας, καθώς μπορεί να οδηγήσει σε χαμηλό ηθικό και ακόμη και σε κύκλο εργασιών προγραμματιστών (Salentin & Hacks, 2020).

Τέλος, ολιστική προσέγγιση για τη διαχείριση του τεχνικού χρέους απαιτεί ολοκληρωμένες στρατηγικές για τον εντοπισμό, την παρακολούθηση, την κατηγοριοποίηση, τη μέτρηση, την ιεράρχηση προτεραιοτήτων και την επίλυση. Αυτές οι στρατηγικές δεν πρέπει να επικεντρώνονται μόνο στις τεχνικές πτυχές αλλά και να ενσωματώνουν εκτιμήσεις για την κοινωνική δυναμική εντός των ομάδων ανάπτυξης λογισμικού (Karamitsos et al., 2020). Αυτό περιλαμβάνει την προώθηση της συνεργασίας και της επικοινωνίας μεταξύ του ακαδημαϊκού κόσμου, της βιομηχανίας και των προμηθευτών εργαλείων για την συλλογική αντιμετώπιση των προκλήσεων που σχετίζονται με την τεχνική διαχείριση του χρέους.

2.2.1 SonarQube πλατφόρμα

Η SonarQube είναι μια πλατφόρμα ανοιχτού κώδικα για συνεχή έλεγχο της ποιότητας του κώδικα. Αρχικά αναπτύχθηκε από τη SonarSource, παρέχει εργαλεία για την ανάλυση και διαχείριση τεχνικού χρέους, την ποιότητα του κώδικα, τις ευπάθειες ασφαλείας και την κάλυψη κώδικα σε πολλές γλώσσες προγραμματισμού. Το SonarQube ενσωματώνεται απρόσκοπτα στη διαδικασία ανάπτυξης λογισμικού, επιτρέποντας στους προγραμματιστές να εντοπίζουν και να αντιμετωπίζουν ζητήματα νωρίς στον κύκλο ζωής της ανάπτυξης (Molnar & Motogna, 2020).

Η πλατφόρμα SonarQube είναι ένα ισχυρό εργαλείο που βοηθά τους προγραμματιστές και τις ομάδες ανάπτυξης να διασφαλίσουν την ποιότητα και την αξιοπιστία του κώδικά τους. Παρέχει ένα ολοκληρωμένο σύνολο λειτουργιών, όπως

ανάλυση στατικού κώδικα, κάλυψη κώδικα, ανίχνευση αντιγραφής κώδικα και ανίχνευση οσμής κώδικα. Χρησιμοποιώντας την πλατφόρμα SonarQube, οι προγραμματιστές μπορούν να εντοπίσουν και να διορθώσουν πιθανά προβλήματα στον κώδικά τους νωρίς στη διαδικασία ανάπτυξης, οδηγώντας σε βελτιωμένη ποιότητα κώδικα και μειωμένο τεχνικό χρέος. Παράλληλα, το SonarQube επιτρέπει στους προγραμματιστές να παρακολουθούν την εξέλιξη του τεχνικού χρέους με την πάροδο του χρόνου, επιτρέποντάς τους να κατανοήσουν πώς αλλάζει και συσσωρεύεται στο λογισμικό τους (Molnar & Motogna, 2020). Επιπλέον, το SonarQube ενσωματώνεται απρόσκοπτα με δημοφιλή εργαλεία ανάπτυξης και ροές εργασίας, όπως το Jenkins και το Git, καθιστώντας εύκολη την ενσωμάτωση της ανάλυσης ποιότητας κώδικα στον κύκλο ζωής ανάπτυξης λογισμικού. Συνολικά, η πλατφόρμα SonarQube είναι ένας πολύτιμος πόρος για προγραμματιστές και ομάδες ανάπτυξης που θέλουν να βελτιώσουν ποιότητα κώδικα, μείωση του τεχνικού χρέους και βελτίωση της συνολικής αξιοπιστίας του λογισμικού. Τα βασικά χαρακτηριστικά του SonarQube περιλαμβάνουν:

- *Ανάλυση στατικού κώδικα:* Το SonarQube εκτελεί ανάλυση στατικού κώδικα για να εντοπίσει σφάλματα, μυρωδιές κώδικα και ευπάθειες ασφαλείας στον πηγαίο κώδικα. Αναλύει διάφορες πτυχές της ποιότητας του κώδικα, συμπεριλαμβανομένης της πολυπλοκότητας του κώδικα, της δυνατότητας συντήρησης, των αντιγραφών και της τήρησης των προτύπων κωδικοποίησης.
- *Παρακολούθηση ζητημάτων:* Το SonarQube παρέχει λεπτομερείς αναφορές για ζητήματα κώδικα, κατηγοριοποιώντας τα με βάση τη σοβαρότητα και τον αντίκτυπο. Οι προγραμματιστές μπορούν να δώσουν προτεραιότητα και να παρακολουθούν αυτά τα ζητήματα, να τα αναθέσουν σε μέλη της ομάδας και να παρακολουθούν την επίλυσή τους με την πάροδο του χρόνου.
- *Τεχνική διαχείριση χρέους:* Το SonarQube υπολογίζει και οπτικοποιεί τεχνικές μετρήσεις χρέους, όπως ο εκτιμώμενος χρόνος που απαιτείται για τη διόρθωση προβλημάτων κώδικα και το κόστος διατήρησης της βάσης κωδικών με την πάροδο του χρόνου. Αυτό βοηθά τις ομάδες να κατανοήσουν τις μακροπρόθεσμες επιπτώσεις των ζητημάτων ποιότητας κώδικα και να δώσουν προτεραιότητα στις προσπάθειες αναδιαμόρφωσης ανάλογα.
- *Ανίχνευση ευπάθειας ασφαλείας:* Το SonarQube περιλαμβάνει ενσωματωμένους κανόνες ασφαλείας και δυνατότητες ανίχνευσης ευπάθειας για τον εντοπισμό πιθανών κινδύνων ασφαλείας στη βάση κωδικών. Επισημαίνει κοινές ευπάθειες

ασφαλείας, όπως επιθέσεις injection, δέσμες ενεργειών μεταξύ τοποθεσιών (cross-site scripting-XSS) και μη ασφαλείς μηχανισμούς ελέγχου ταυτότητας.

- *Ανάλυση κάλυψης κώδικα:* Το SonarQube ενσωματώνεται με εργαλεία κάλυψης κώδικα για τη μέτρηση του ποσοστού του κώδικα που καλύπτεται από αυτοματοποιημένες δοκιμές. Παρέχει πληροφορίες για τομείς της βάσης κωδικών που δεν διαθέτουν επαρκή κάλυψη δοκιμών, βοηθώντας τις ομάδες να βελτιώσουν τη συνολική ποιότητα και αξιοπιστία του λογισμικού τους.
- *Ενσωμάτωση με αγωγούς CI/CD:* Το SonarQube ενσωματώνεται απρόσκοπτα με αγωγούς συνεχούς ενοποίησης και συνεχούς ανάπτυξης (CI/CD), επιτρέποντας στους προγραμματιστές να αυτοματοποιούν τους ελέγχους ποιότητας κώδικα ως μέρος της ροής εργασιών ανάπτυξής τους. Υποστηρίζει δημοφιλή εργαλεία CI/CD όπως τα Jenkins, Azure DevOps και GitLab CI.

Συνοψίζοντας, το SonarQube είναι μια ισχυρή πλατφόρμα για τη βελτίωση της ποιότητας του κώδικα, τη μείωση του τεχνικού χρέους και την ενίσχυση της ασφάλειας των έργων λογισμικού. Παρέχοντας χρήσιμες πληροφορίες και αυτοματοποιημένα εργαλεία ανάλυσης, δίνει τη δυνατότητα στις ομάδες ανάπτυξης να παρέχουν λογισμικό υψηλότερης ποιότητας πιο αποτελεσματικά.

3 Μεθοδολογία έρευνας

3.1 Επιλογή περιπτώσεων & μονάδες ανάλυσης

Λαμβάνοντας υπόψη τους στόχους της παρούσας έρευνας και τη διαθεσιμότητα των δεδομένων από το SonarCloud επιλέχθηκαν πέντε Foundation τα οποία είναι το Apache, Eclipse, Oracle, Open Cluster και Code Libs project τα οποία είναι διαθέσιμα με τα δεδομένα τους στο SonarCloud και υπάρχει άμεση πρόσβαση σε αυτά, ενώ μας επιτρέπουν να προχωρήσουμε σε αξιόπιστη και συνολική ανάλυση. Ειδικότερα, τα συγκεκριμένα project επιλέχθηκαν διότι έχουν υψηλή δημοτικότητα ή ευρεία χρήση στη βιομηχανία του ανοιχτού λογισμικού, ενώ συγχρόνως έχουν κριθεί σημαντικά για την ανάπτυξη του ανοιχτού λογισμικού κοινότητας ή την επιχειρηματική κοινότητα. Αξίζει να σημειωθεί ότι διαφορετικές κατηγορίες έργων για ευρύτερη κάλυψη και αντιπροσώπευση. Παρακάτω δίνονται περιληπτικά στοιχεία για τα προαναφερθέντα Foundation:

- *Apache:* Το Apache SonarQube είναι μια πλατφόρμα ανοιχτού κώδικα για συνεχή επιθεώρηση της ποιότητας του κώδικα. Παρέχει έναν κεντρικό πίνακα εργαλείων που εμφανίζει μετρήσεις ποιότητας κώδικα και αποτελέσματα ανάλυσης, επιτρέποντας στους προγραμματιστές να παρακολουθούν και να παρακολουθούν εύκολα την υγεία των έργων τους. Η πλατφόρμα υποστηρίζει ανάλυση για διάφορες γλώσσες προγραμματισμού, συμπεριλαμβανομένων των Java, C++ και Python, και προσφέρει ένα ευρύ φάσμα συνόλων κανόνων για τον εντοπισμό κοινών ζητημάτων κώδικα και παραβιάσεων βέλτιστων πρακτικών (García-Muñoz et al., 2016).
- *Eclipse:* Το Eclipse SonarQube είναι μια ενοποίηση της πλατφόρμας ποιότητας κώδικα SonarQube με το Eclipse IDE. Αυτή η ενοποίηση επιτρέπει στους προγραμματιστές να πραγματοποιούν ανάλυση κώδικα και να προβάλλουν μετρήσεις ποιότητας απευθείας στο περιβάλλον του Eclipse, καθιστώντας βολικό και αποτελεσματικό την αντιμετώπιση προβλημάτων κώδικα σε πραγματικό χρόνο κατά τη διάρκεια της διαδικασίας ανάπτυξης .
- *Open Cluster:* Το Open Cluster είναι ένα έργο ανοιχτού κώδικα που στοχεύει να επεκτείνει τις δυνατότητες της πλατφόρμας SonarQube παρέχοντας πρόσθετα και ενσωματώσεις για την ανάλυση της ποιότητας του κώδικα σε διάφορους τομείς και γλώσσες προγραμματισμού. Αυτά τα πρόσθετα και οι ενσωματώσεις επιτρέπουν

στους προγραμματιστές να προσαρμόσουν τη διαδικασία ανάλυσης κώδικα σύμφωνα με τις ειδικές τεχνικές απαιτήσεις τους. Η πλατφόρμα SonarQube προσφέρει μια σειρά από δυνατότητες για ανάλυση κώδικα, όπως ανάλυση στατικού κώδικα, κάλυψη κώδικα, ανίχνευση αντιγραφής κώδικα και ανίχνευση οσμής κώδικα (García-Muñoz et al., 2016).

- *Code Libs:* Το Code Libs είναι μια ολοκληρωμένη βιβλιοθήκη αποσπασμάτων κώδικα και βέλτιστων πρακτικών που μπορούν να ενσωματωθούν με την πλατφόρμα SonarQube. Αυτά τα αποσπάσματα κώδικα και οι βέλτιστες πρακτικές μπορούν να χρησιμοποιηθούν για τη βελτίωση της ποιότητας του κώδικα, τη μείωση του τεχνικού χρέους και την επιβολή προτύπων κωδικοποίησης σε έργα λογισμικού.
- *Oracle:* Το Oracle SonarQube είναι μια έκδοση της πλατφόρμας SonarQube ειδικά προσαρμοσμένη για να λειτουργεί απρόσκοπτα με την ανάπτυξη λογισμικού Oracle. Παρέχει πρόσθετες δυνατότητες και ενσωματώσεις που έχουν βελτιστοποιηθεί για βάσεις δεδομένων, εφαρμογές και γλώσσες προγραμματισμού Oracle. Η πλατφόρμα SonarQube προσφέρει μια σειρά από δυνατότητες και ενσωματώσεις για την ανάλυση της ποιότητας του κώδικα σε διάφορους τομείς και γλώσσες προγραμματισμού.

3.2 Συλλογή δεδομένων

Από αυτά τα πέντε Foundation καταγράφθηκαν 50 project για το κάθε ένα, από το μεγαλύτερο μέγεθος προς το μικρότερο. Οι μετρήσεις που επιλέχτηκαν από όλα τα project σημειώθηκαν σε ένα φύλλο Excel. Αναλυτικότερα στο Excel καταγράφηκαν για το κάθε ένα project του κάθε Foundation, οι εξής μετρήσεις: name of project, link, Quality Gate, Lines of code, Reliability, Maintainability, Security, Security Review, Coverage, Duplications, Technical Debt, Clean Code Attribute (1. Consistency, 2. Intentionality, 3. Adaptability, 4. Responsibility) και Severity (1. High, 2. Medium, 3. Low). Αυτές είναι οι κύριες καταγραφές που συλλέχτηκαν, ώστε να προχωρήσει η έρευνα στην ανάλυση των δεδομένων.

3.3 Ανάλυση δεδομένων

Στη συνέχεια για την ανάλυση δεδομένων χρησιμοποιήθηκε το SPSS για να απαντηθούν τα ερευνητικά ερωτήματα και να πραγματοποιηθούν στατιστικές αναλύσεις . Τα δεδομένα εισήχθησαν από το Excel στο SPSS, ώστε να αναλυθούν τα δεδομένα. Εφαρμόστηκε ανάλυση διακύμανσης (ANOVA) για να εξεταστούν τυχόν στατιστικές διαφορές μεταξύ των διαφορετικών Foundation projects, χρησιμοποιώντας ως ανεξάρτητη μεταβλητή τα διάφορα Foundation projects και ως εξαρτημένη μεταβλητή τις μετρήσεις που πραγματοποιήθηκαν.

Βασικός σκοπός της παρούσας διπλωματικής εργασίας είναι η διερεύνηση του τεχνικού χρέους σε ανοιχτό λογισμικό. Για το σκοπό αυτό, τέθηκαν οι κάτωθι ερευνητικοί στόχοι:

- Ο έλεγχος της ενδεχόμενης ύπαρξης σημαντικής σχέσης ανάμεσα στο τεχνικό χρέος και τη συνέπεια, την πρόθεση, την προσαρμοστικότητα και την ευθύνη.
- Η ύπαρξη σημαντικής συσχέτισης μεταξύ τεχνικού χρέους και της αξιοπιστίας.
- Η ύπαρξη σημαντικής συσχέτισης μεταξύ τεχνικού χρέους και της συντήρησης και ασφάλειας.

4 Αποτελέσματα

4.1 Περιγραφική στατιστική

Για τη στατιστική ανάλυση των δεδομένων, αξιοποιήθηκε το πρόγραμμα SPSS v.26. Για όλες τις μεταβλητές, αλλά και τις συνολικές βαθμολογίες, υπολογίστηκαν διάφορα μέτρα περιγραφικής στατιστικής ανάλυσης (μέση τιμή, τυπική απόκλιση, συχνότητα, ποσοστό, ελάχιστη/μέγιστη τιμή, μέση τιμή, εύρος, κύρτωση, σκεδαστικότητα).

Στο παρακάτω Πίνακας 1 παρουσιάζονται η μέγιστη, η ελάχιστη τιμή, ο μέσος όρος, η τυπική απόκλιση, η σκεδαστικότητα και η κύρτωση. Ειδικότερα, το εύρος του τεχνικού χρέους κυμαίνεται από 0 έως 62400, με μέση τιμή 4259,882 και τυπική απόκλιση 13432,78. Η ασσυμετρία είναι 3,252 και η κύρτωση 9,246, δείχνοντας μια σχετικά συγκεντρωμένη κατανομή με μια τάση προς τις χαμηλότερες τιμές αλλά με την παρουσία κάποιων ακραίων τιμών. Στη μεταβλητή της συνέπειας η μέγιστη τιμή φτάνει τις 61000, με μέση τιμή 706,334 και τυπική απόκλιση 6237,68. Η ασσυμετρία και η κύρτωση είναι αντίστοιχα 9,087 και 82,368, υποδηλώνοντας μια κατανομή με σημαντικά περισσότερες χαμηλές τιμές αλλά και την ύπαρξη ακραίων τιμών. Στην πρόθεση η μέγιστη τιμή είνα108.000 και μέση τιμή 1120,314, η πρόθεση δείχνει επίσης μεγάλη διαφοροποίηση στην ποιότητα κώδικα ανάμεσα στα έργα, με τυπική απόκλιση 10204,79. Η υψηλή ασσυμετρία (9,955) και κύρτωση (100,851) ενισχύουν την άποψη για μια κατανομή με ακραίες τιμές. Στην προσαρμοστικότητα η μέγιστη τιμή είναι 45354, με μέση τιμή 397,619 και τυπική απόκλιση 3627,52. Η ασσυμετρία και η κύρτωση υποδεικνύουν μια ακόμη μεγαλύτερη κλίση προς τις υψηλότερες τιμές, δείχνοντας έντονη μη συμμετρική κατανομή. Στην μεταβλητή της αξιοπιστίας η μέγιστη τιμή είναι 596, η μέση τιμή 11,14463 και η τυπική απόκλιση 69,237, εμφανίζει μικρότερη διακύμανση σε σχέση με τις άλλες μετρήσεις, αλλά εξακολουθεί να έχει σημαντική στρέψη και κύρτωση. Η συντηρησιμότητα απεικονίζει μέγιστη τιμή 1.300, μέση τιμή 52,3479 και τυπική απόκλιση 152,627, με σχετικά μικρότερη ασσυμετρία και κύρτωση σε σύγκριση με τις άλλες μετρήσεις, δείχνοντας πιο ισορροπημένη κατανομή. Τέλος, η ασφάλεια έχει μέγιστη τιμή 16, μέση τιμή 0,1612 και τυπική απόκλιση 1,2436, ενώ η αναθεώρηση ασφαλείας 163,00 και τυπική απόκλιση 2,942, συγχρόνως οι τιμές ασσυμετρίας και κύρτωσης για την

ασφάλεια υποδηλώνουν μια κατανομή με τάση προς τις χαμηλότερες τιμές και την παρουσία ελάχιστων ακραίων τιμών.

Πίνακας 1: Περιγραφικά δεδομένα μεταβλητών

	Minimum	Maximum	Mean	Std. Deviation	Skewness		Kurtosis	
	Statistic	Statistic	Statistic	Statistic	Statistic	Std. Error	Statistic	Std. Error
Τεχνικό χρέος	,00	62400,00	4259,882	13432,78	3,252	,226	9,246	,449
Συνέπεια	,00	61000,00	706,334	6237,68	9,087	,156	82,368	,312
Πρόθεση	,00	108000,00	1120,314	10204,79	9,955	,156	100,851	,312
Προσαρμοστικότητα	,00	45354,00	397,6190	3627,52	10,268	,156	112,560	,312
Αξιοπιστία	,00	596,000	11,14463	69,237	7,609	,156	58,187	,312
Συντηρησιμότητα	,00	1300,00	52,3479	152,627	4,674	,156	26,797	,312
Ασφάλεια	,00	16,00	,1612	1,24360	10,459	,156	120,869	,312
Αναθεώρηση ασφαλείας	,00	163,00	2,9421	16,47420	8,238	,156	74,667	,312

Η ασσυμετρία (Skewness) στις μετρήσεις όπως το τεχνικό χρέος, τη συνέπεια, και την πρόθεση δείχνει ξεκάθαρα μια σημαντική ματαβολή προς τις υψηλότερες τιμές, με τις τιμές να κυμαίνονται από 3,252 έως 10,459 για διάφορες μετρήσεις. Αυτό υποδηλώνει ότι, ενώ η μέση τιμή μπορεί να δίνει μια γενική ένδειξη για την τάση των δεδομένων, η παρουσία ακραίων τιμών προς το υψηλότερο τμήμα της κατανομής μπορεί να παραμορφώνει την πραγματική εικόνα της καταστάσεως.

Η κύρτωση (Kurtosis) ενισχύει την παραπάνω εικόνα, με τιμές από 9,246 έως 120,869 για τις ίδιες μετρήσεις, υποδηλώνοντας μια εξαιρετικά "κορυφαία" κατανομή που επικεντρώνεται γύρω από τον μέσο όρο με σημαντικές ακραίες τιμές. Αυτό ερμηνεύεται πως σε σύγκριση με μια κανονική κατανομή, υπάρχει μια σημαντικά μεγαλύτερη πιθανότητα να συναντήσουμε ακραίες τιμές στα δεδομένα, πράγμα που θα πρέπει να ληφθεί υπόψη κατά την αξιολόγηση και την ερμηνεία τους.

Οι αξιολογήσεις αυτές καταδεικνύουν ότι τα δεδομένα για τις μετρήσεις παρουσιάζουν μια σημαντική ασυμμετρία και έντονες ακραίες τιμές, πράγμα που υποδηλώνει την ανάγκη για προσεκτική ανάλυση και διαχείριση των ακραίων περιπτώσεων κατά την εφαρμογή μέτρων βελτίωσης στην ανάπτυξη και συντήρηση λογισμικού.

Στον παρακάτω Πίνακας **2** παρουσιάζονται το άθροισμα, η μέση τιμή, η μέγιστη, και η ελάχιστη τιμή των οργανισμών.

Πίνακας 2: Περιγραφικά δεδομένα οργανισμών

	Lines of Code	Reliability	Maintainability	Security	Security Review	Coverage	Duplications	Technical Debt	Consistency	Intentionality	Adaptability	Responsibility	High	Medium	Low
APACHE	*10950*	*764*	*51191*	*8*	*65*	*136388*	*908195*	*122236*	*46690*	*49725*	*46354*	*0*	*46654*	*47397*	*49059*
SUM	219	15	1024	0	1	7178	23287	4075	934	995	927	0	933	948	981
AVERAGE	790	524	45292	5	45	45535	45510	45476	45323	45450	45354	0	45536	45386	45540
MAX	23	0	0	0	0	0	0	6	0	0	0	0	0	0	0
MIN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ECLIPSE	*235490*	*1794*	*183506*	*3*	*444*	*136910*	*862783*	*130234*	*123526*	*219354*	*95623*	*0*	*107431*	*83252*	*206456*
SUM	5887	41	4171	0	10	5476	22705	8682	2807	5101	1017	0	2442	1892	4692
AVERAGE	45543	596	45447	3	163	45558	45536	62400	61000	108000	45354	0	53000	40000	101000
MAX	5	0	0	0	0	0	0	2	0	0	0	0	0	0	0
MIN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ORACLE	*779154*	*29*	*47127*	*9*	*114*	*0*	*181495*	*47361*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
SUM	17708	1	982	0	2	0	4654	7894	0	0	0	0	0	0	0
AVERAGE	45541	18	45078	9	38	0	45450	45508	0	0	0	0	0	0	0
MAX	1	0	0	0	0	0	0	29	0	0	0	0	0	0	0
MIN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OPEN CL.	*118149*	*37*	*190*	*16*	*73*	*45846*	*272341*	*48064*	*15*	*0*	*6*	*0*	*5*	*16*	*0*
SUM	23630	1	4	0	1	6549	6190	12016	0	0	0	0	0	0	0
AVERAGE	45538	30	143	16	64	45446	45508	45352	9	0	5	0	4	10	0
MAX	7	0	0	0	0	0	0	48	0	0	0	0	0	0	0
MIN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CODE LIB	*12793*	*73*	*47719*	*3*	*16*	*45456*	*544759*	*137651*	*702*	*2034*	*595*	*0*	*457*	*1622*	*1252*
SUM	457	1	954	0	0	2841	17024	9832	14	41	12	0	9	32	25
AVERAGE	962	37	45323	2	7	45456	45524	45506	319	724	118	0	107	593	456
MAX	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MIN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Βάσει των μετρήσεων αυτών του πίνακα :

- Apache: Παρατηρείτε πως η συντηρησιμότητα σε αυτό το foundation είναι σχετικά υψηλή το οποίο σημαίνει πως ο κώδικας μπορεί να ενημερωθεί και να διορθωθεί σχετικά εύκολα. Μεγάλη έμφαση δίνεται στις διπλοτυπίες ,πιθανώς αυτό να εξηγείτε λόγω μεγάλων αριθμών κώδικα που περιέχει . Ωστόσο το Τεχνικό χρέος φαίνεται σχετικά υψηλό το οποίο μπορεί να υποδηλώνει ότι ίσως χρειάζεται εργασία για την αντιμετώπιση αδυναμιών.
- Eclipse: Το Eclipse εμφανίζει από τις υψηλότερες μέσες τιμές στις μετρήσεις σε σχέση με τα υπόλοιπα foundations , ιδιαίτερα αυτό μπορεί να παρατηρηθεί στη συντηρησιμότητα καθώς και στο Τεχνικό χρέος. Αυτές οι παρατηρήσεις είναι συνηθισμένες για μεγάλα και πολύπλοκα έργα.
- Oracle: Στο foundation αυτό παρατηρείτε υψηλότερη αξιοπιστία συγκριτικά με τα υπόλοιπα , όμως με πολύ χαμηλή συντηρισιμότητα , αυτό μπορεί να σημαίνει πως υπάρχουν πιθανές προκλήσεις στην ανάπτυξη καθώς και στη διαχείριση του

κώδικα. Επιπλέον το Τεχνικό Χρέος εμφανίζει σχετικά μέτρια τιμή το οποίο υποδηλώνει πως μπορούν να υπάρξουν βελτιώσεις στην εργασία .

- Open Cluster: Το Open Cluster φαίνεται να έχει σταθερή αξιοπιστία με χαμηλή συντηρησιμότητα, υποδηλώνοντας έτσι πιθανές προκλήσεις στην συντήρηση και διαχείριση του κώδικα .Όσο αφορά την ασφάλεια η εξαιρετικά χαμηλή τιμή της, οδηγεί στο συμπέρασμα ανάγκης ενίσχυσης της ασφάλειας .Το μετρίως υψηλό Τεχνικό χρέος είναι ένδειξη πως υπάρχει περιθώριο για βελτιώσεις.
- Code Libs: Το foundation αυτό εμφανίζει αξιοπρεπή αξιοπιστία και ασφάλεια καθώς και συνέπεια, πράγμα το οποίο πιθανόν να αντικατοπτρίζει την προσπάθεια για υψηλή ποιότητα τόσο του κώδικα όσο και της ασφάλειάς του .Αξιοσημείωτο είναι το γεγονός πως το Τεχνικό χρέος έχει από της υψηλές τιμές του πίνακα ,ένδειξη η οποία οδηγεί στο συμπέρασμα πως υπάρχει περιθώριο βελτιστοποίησης.

Συμπερασματικά , κάθε foundation δείχνει διαφορετικά επίπεδα επιδόσεων ως προς τις μετρήσεις με διαφορετικές προκλήσεις και προτεραιότητες το κάθε ένα. Κοινός στόχος φαίνεται να είναι η ασφάλεια και η αξιοπιστία ως προς όλα ,ενώ το Τεχνικό χρέος και κάποιες άλλες μετρήσεις διαφέρουν , αυτό υποδηλώνει πως υπάρχουν διαφορετικές προσεγγίσεις καθώς και στάδια ανάπτυξης των έργων σε κάθε foundation χωριστά.

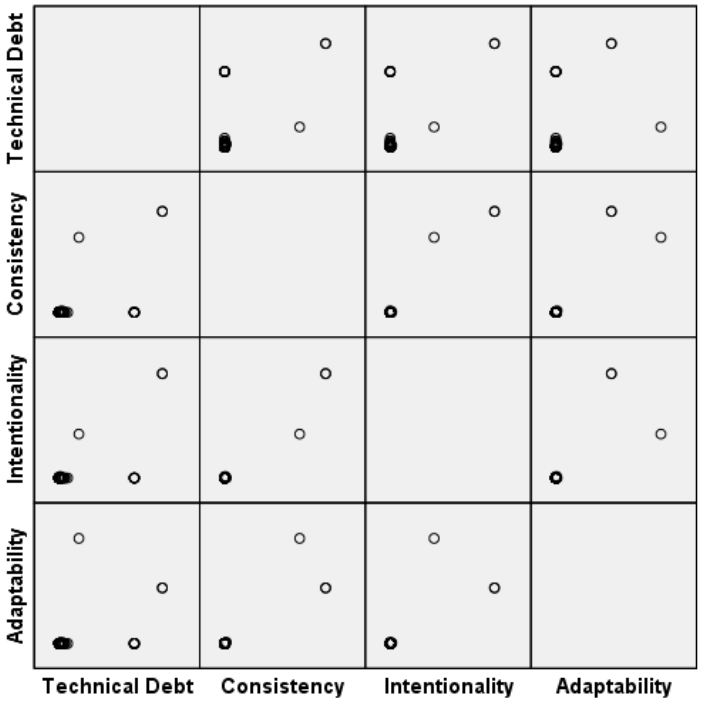
4.2 Συσχετίσεις μεταξύ μεταβλητών

*4.2.1 Τεχνικό χρέος*συνέπεια, πρόθεση, προσαρμοστικότητα*

Αρχικά, χρησιμοποιήθηκε ανάλυση διακύμανσης Anova για να εξεταστεί αν υπήρχαν στατιστικά σημαντικές διαφορές μεταξύ του τεχνικού χρέους και των μεταβλητών της συνέπειας, της πρόθεσης και της προσαρμοστικότητας Από τα αποτελέσματα προκύπτει το συμπέρασμα ότι υπήρξαν στατιστικά σημαντικές διαφορές (Εικόνα 1). Ειδικότερα για το τεχνικό χρέος συσχετίζεται με τη συνέπεια ($F(93, 20)= 0.000$ $p<0.05$, Πίνακας 3), την πρόθεση ($F(93, 20)= 0.000$ $p<0.05$, Πίνακας 3), και την προσαρμοστικότητα ($F(93, 20)= 0.000$ $p<0.05$, Πίνακας 3), όπου βρέθηκαν ως σημαντικοί παράγοντες.

Πίνακας 3: ANOVA αποτελέσματα τεχνικού χρέους*συνέπειας, πρόθεσης, προσαρμοστικότητας

		Sum of Squares	df	Mean Square	F	Sig.
Συνέπεια (Consistency)	Between Groups	9241433760,623	93	99370255,491	161933,114	,000
	Within Groups	12273,000	20	613,650		
	Total	9241446033,623	113			
Πρόθεση (Intentionality)	Between Groups	24756105818,281	93	266194686,218	63636,760	,000
	Within Groups	83660,667	20	4183,033		
	Total	24756189478,947	113			
Προσαρμοστικότητα (Adaptability)	Between Groups	3128342560,237	93	33638092,046	93227,656	,000
	Within Groups	7216,333	20	360,817		
	Total	3128349776,570	113			



Εικόνα 1: Matrix scatter τεχνικού χρέους*συνέπεια, πρόθεση, προσαρμοστικότητα

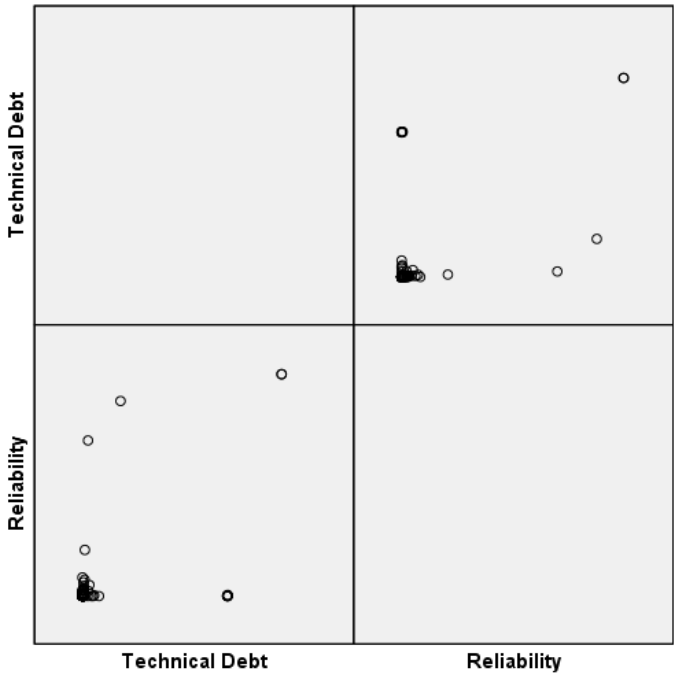
*4.2.2 Τεχνικό χρέος*αξιοπιστία*

Στη συνέχεια, χρησιμοποιήθηκε one way ανάλυση διακύμανσης Anova για να εξεταστεί αν υπήρχαν στατιστικά σημαντικές διαφορές μεταξύ του τεχνικού χρέους και της μεταβλητής της αξιοπιστίας. Από τα αποτελέσματα προκύπτει το συμπέρασμα ότι υπήρξε στατιστικά σημαντική διαφορά (Εικόνα 2). Ειδικότερα το τεχνικό χρέος

συσχετίζεται με την αξιοπιστία (F(93, 20)= 0.000 p<0.05, Πίνακας 4), όπου βρέθηκε ως σημαντικός παράγοντας.

Πίνακας 4: ANOVA αποτελέσματα τεχνικού χρέους*αξιοπιστίας

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	1119949,421	93	12042,467	139,961	,000
Within Groups	1720,833	20	86,042		
Total	1121670,254	113			



Εικόνα 2: Matrix scatter τεχνικού χρέους*αξιοπιστία

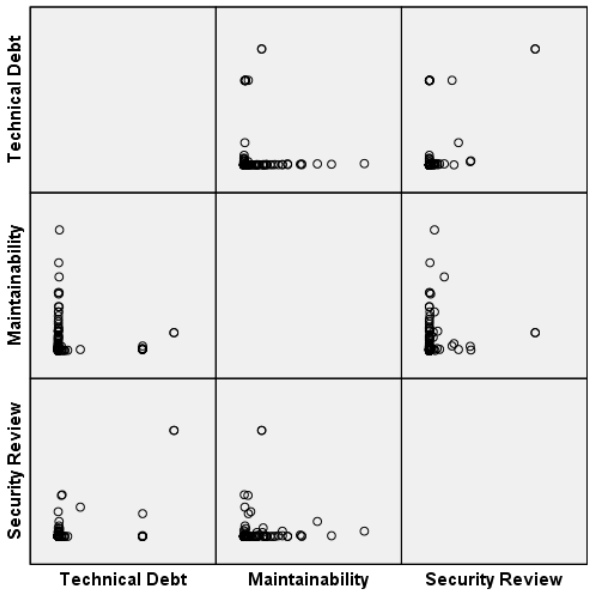
4.2.3 Τεχνικό χρέος συντηρησιμότητας, αναθεώρησης ασφαλείας*

Στη συνέχεια, χρησιμοποιήθηκε ανάλυση διακύμανσης Anova για να εξεταστεί αν υπήρχαν στατιστικά σημαντικές διαφορές μεταξύ του τεχνικού χρέους και των μεταβλητών της συντηρησιμότητας και της αναθεώρησης ασφαλείας. Από τα αποτελέσματα προκύπτει το συμπέρασμα ότι υπήρξαν στατιστικά σημαντικά διαφορές (

Εικόνα **3**). Ειδικότερα το τεχνικό χρέος συσχετίζεται με την συντηρησιμότητα (F(93, 20)= 0.000 p<0.05, Πίνακας 5), και αναθεώρηση ασφαλείας (F(93, 20)= 0.000 p<0.05, Πίνακας 5), όπου βρέθηκαν ως σημαντικοί παράγοντες.

Πίνακας 5: ANOVA αποτελέσματα τεχνικού χρέους*συντηρησιμότητας, αναθεώρησης ασφαλείας

		Sum of Squares	df	Mean Square	F	Sig.
Συντηρησιμότητα (Maintainability)	Between Groups	4628222,901	93	49765,838	4,117	,000
	Within Groups	241781,833	20	12089,092		
	Total	4870004,734	113			
Αναθεώρηση Ασφαλείας (Security Review)	Between Groups	62056,360	93	667,273	13,153	,000
	Within Groups	1014,667	20	50,733		
	Total	63071,026	113			



Εικόνα 3: Matrix scatter τεχνικού χρέους*συντηρησιμότητας, αναθεώρησης ασφαλείας

5 Συμπεράσματα

Ο βασικός σκοπός της παρούσας διπλωματικής εργασίας ήταν η διερεύνηση του τεχνικού χρέους σε ανοιχτό λογισμικό. Σύμφωνα με τη διαθεσιμότητα των δεδομένων από το SonarCloud επιλέχθηκαν πέντε Foundation τα οποία είναι το Apache, Eclipse, Oracle, Open Cluster και Code Libs project τα οποία είναι διαθέσιμα με τα δεδομένα τους στο SonarCloud καθώς υπάρχει άμεση πρόσβαση σε αυτά, ενώ μας επιτρέπουν να προχωρήσουμε σε αξιόπιστη και συνολική ανάλυση. Ειδικότερα, υπήρξαν στατιστικά σημαντικές διαφορές μεταξύ διαφορές μεταξύ του τεχνικού χρέους και των μεταβλητών της συνέπειας, της πρόθεσης και της προσαρμοστικότητας. Το τεχνικό χρέος συσχετίστηκε με τη συνέπεια ($F(93, 20)= 0.000$ $p<0.05$), την πρόθεση ($F(93, 20)= 0.000$ $p<0.05$), και την προσαρμοστικότητα ($F(93, 20)= 0.000$ $p<0.05$), όπου βρέθηκαν ως σημαντικοί παράγοντες. Από τα αποτελέσματα προκύπτει το συμπέρασμα ότι υπήρξε στατιστικά σημαντική διαφορά μεταξύ του τεχνικού χρέους και της μεταβλητής της αξιοπιστίας. Παράλληλα, το τεχνικό χρέος συσχετίστηκε με την αξιοπιστία ($F(93, 20)= 0.000$ $p<0.05$), όπου βρέθηκε ως σημαντικός παράγοντας. Από τα αποτελέσματα προέκυψε το συμπέρασμα ότι υπήρξαν στατιστικά σημαντικά διαφορές μεταξύ του τεχνικού χρέους και των μεταβλητών της συντηρησιμότητας και της αναθεώρησης ασφαλείας. Αναλυτικότερα, το τεχνικό χρέος συσχετίστηκε με την συντηρησιμότητα ($F(93, 20)= 0.000$ $p<0.05$), και την αναθεώρηση ασφαλείας ($F(93, 20)= 0.000$ $p<0.05$), όπου βρέθηκαν ως σημαντικοί παράγοντες.

Μέσα από την παρούσα έρευνα καταδείχτηκε ότι η διαχείριση του τεχνικού χρέους σε έργα λογισμικού ανοιχτού κώδικα είναι ζωτικής σημασίας για την ελαχιστοποίηση των οικονομικών του επιπτώσεων, τη μείωση του κόστους ανάπτυξης και συντήρησης, τη βελτίωση της αξιοπιστίας και ποιότητας του λογισμικού και τη διασφάλιση της μακροπρόθεσμης επιτυχίας και βιωσιμότητας έργων ανοιχτού κώδικα. Για την αποτελεσματική διαχείριση του τεχνικού χρέους σε λογισμικό ανοιχτού κώδικα, είναι απαραίτητο να έχουμε πλήρη κατανόηση των επιπτώσεών του και να αναπτύξουμε στρατηγικές για τον μετριασμό των επιπτώσεών του. Συμπερασματικά, η έρευνα και οι μελέτες που έγιναν σε διάφορους τομείς έδειξαν ότι το τεχνικό χρέος σε λογισμικό ανοιχτού κώδικα μπορεί να έχει σημαντικές οικονομικές επιπτώσεις και αυξημένο κόστος (Krogh & Spaeth, 2007). Επιπλέον, η διαφάνεια και η κοινή αναστοχαστικότητα των κοινοτήτων λογισμικού ανοιχτού κώδικα παρέχουν μια άνευ προηγουμένου ευκαιρία στους ερευνητές να έχουν πρόσβαση σε δεδομένα και να συμμετάσχουν σε διάλογο

σχετικά με τη λειτουργία του. Ως εκ τούτου, η μελέτη του τεχνικού χρέους σε λογισμικό ανοιχτού κώδικα παρέχει ένα μοναδικό πλαίσιο για διεπιστημονική έρευνα και προωθεί μεγαλύτερο διάλογο μεταξύ κλάδων και τομέων. Με βάση την έρευνα που διεξάγεται σε διάφορους τομείς, είναι σαφές ότι το τεχνικό χρέος σε έργα λογισμικού ανοιχτού κώδικα έχει σημαντικά οικονομικά επιπτώσεις. Επιπροσθέτως, η συνεργατική φύση της ανάπτυξης ανοιχτού κώδικα και η εξάρτηση από μια κοινότητα εθελοντών υπογραμμίζουν τη σημασία της αποτελεσματικής διαχείρισης του τεχνικού χρέους για τη διασφάλιση της βιωσιμότητας και της ανάπτυξης έργων ανοιχτού κώδικα. Επομένως, πριν και μετά την εφαρμογή στρατηγικών για τον μετριασμό του τεχνικού χρέους σε έργα λογισμικού ανοιχτού κώδικα, είναι απαραίτητο να διεξάγονται εμπειρικές μελέτες για την αξιολόγηση της κατάστασης της πρακτικής και την κατανόηση των επιπτώσεων αυτών των στρατηγικών.

Αυτές οι μελέτες θα παράσχουν πληροφορίες για τον αντίκτυπο του τεχνικού χρέους στην ποιότητα του λογισμικού, θα εντοπίσουν προκλήσεις και ευκαιρίες για τους συνεισφέροντες ανοιχτού κώδικα, θα προτείνουν βέλτιστες πρακτικές για τη μείωση του τεχνικού χρέους σε λογισμικό ανοιχτού κώδικα και θα διερευνήσουν μελλοντικές τάσεις στην εξέλιξη της ανάπτυξης ανοιχτού κώδικα σε σχέση με τεχνικό χρέος.

Η διαφάνεια και η κοινοτική αντανακλαστικότητα των κοινοτήτων λογισμικού ανοιχτού κώδικα παρέχουν μια άνευ προηγουμένου ευκαιρία στους ερευνητές να έχουν πρόσβαση σε δεδομένα και να εμπλακούν σε διάλογο σχετικά με τη λειτουργία του, το οποίο με τη σειρά του παρέχει ένα μοναδικό πλαίσιο για διεπιστημονική έρευνα και προωθεί τον μεγαλύτερο διάλογο μεταξύ κλάδων και πεδίων.

Μελλοντικές τάσεις και επιπτώσεις για την ανάπτυξη ανοιχτού κώδικα όσον αφορά το τεχνικό χρέος περιλαμβάνουν την ανάγκη για βελτιωμένα εργαλεία και πρακτικές για τον εντοπισμό και τη διαχείριση του τεχνικού χρέους, την ενσωμάτωση της τεχνικής διαχείρισης του χρέους στη διαδικασία ανάπτυξης και την υιοθέτηση προσεγγίσεων που δίνουν προτεραιότητα στη μακροπρόθεσμη βιωσιμότητα και ποιότητα έναντι των βραχυπρόθεσμων κερδών (Bavitz et al., 2018).

Τέλος, η ερευνητική κοινότητα της μηχανικής λογισμικού θα πρέπει να συνεχίσει να εστιάζει στο τεχνικό χρέος για να ενισχύσει την αποτελεσματικότητα και την αξιοπιστία των έργων ανοιχτού κώδικα. Λόγω του μοναδικού περιβάλλοντος συνεργασίας σε κοινότητες ανοιχτού κώδικα, υπάρχει μια άνευ προηγουμένου ευκαιρία για διεπιστημονική έρευνα και προώθηση του διαλόγου σε διάφορους τομείς, συμβάλλοντας

στη βελτίωση στρατηγικών και εργαλείων που μπορούν προληπτικά να ελαχιστοποιήσουν τη συσσώρευση τεχνικού χρέους και να διατηρήσουν την ευρωστία των συστημάτων λογισμικού ανοιχτού κώδικα.

Βιβλιογραφία

Alves, N. S. R., Mendes, T. S., De Mendonça, M. G., Spinola, R. O., Shull, F., & Seaman, C. (2016). Identification and management of technical debt: A systematic mapping study. Information and Software Technology, 70, 100–121.

Ampatzoglou, A., Ampatzoglou, A., Chatzigeorgiou, A., & Avgeriou, P. (2015). The financial aspect of managing technical debt: A systematic literature review. Information and Software Technology, 64, 52-73.

Ampatzoglou, A., Tsintzira, A. A., Arvanitou, E. M., Chatzigeorgiou, A., Stamelos, I., Moga, A., … Kehagias, D. (2019). Applying the single responsibility principle in industry: Modularity benefits and trade-offs. In ACM International Conference Proceeding Series (pp. 347–352). Association for Computing Machinery.

Aniche, M., Yoder, J., & Kon, F. (2019). Current challenges in practical object-oriented software design. In Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results, ICSE-NIER 2019 (pp. 113–116). Institute of Electrical and Electronics Engineers Inc.

Bavitz, C., Hessekiel, K., & Ritvo, D T. (2018). Organization & Structure of Open Source Software Development Initiatives. Retrieved from: https://cyber.harvard.edu/publications/2017/03/OrganizationStructure

Besker, T., Martini, A., & Bosch, J. (2018). Technical debt cripples software developer productivity: A longitudinal study on developers' daily software development work. In Proceedings - International Conference on Software Engineering (pp. 105–114). IEEE Computer Society.

Biase, D. M., Rastogi, A., Bruntink, M., & Van Deursen, A. (2019). The delta maintainability model: Measuring maintainability of fine-grained code changes. In Proceedings - 2019 IEEE/ACM International Conference on Technical Debt, TechDebt 2019 (pp. 113–122). Institute of Electrical and Electronics Engineers Inc.

Borocki, J., Djakovic, V., Bunčić, S., & Buncic, S. (2016). Analysis of innovation factors at joint stock companies on emerging markets applying strategic planning model. DAAAM International Scientific Book 2016, 361-374.

Brown, N., Cai, Y., Guo, Y., Kazman, R., Kim, M., Kruchten, P., … Zazworka, N. (2010). Managing technical debt in software-reliant systems. In Proceedings of the FSE/SDP Workshop on the Future of Software Engineering Research, FoSER 2010 (pp. 47–51).

Dyda, A., Fahim, M., Fraser, J., Kirrane, M., Wong, I., McNeil, K., & Sullivan, C. (2021). Managing the digital disruption associated with covid-19-driven rapid digital transformation in brisbane, australia. Applied Clinical Informatics, 12(05), 1135-1143.

Fernández-Sánchez, C., Garbajosa, J., Yagüe, A., & Pérez, J. (2017). Identification and analysis of the elements required to manage technical debt by means of a systematic mapping study. Journal of Systems and Software, 124, 22-38.

García-Munoz, J., García-Valls, M., & Escribano-Barreno, J. (2016). Improved metrics handling in SonarQube for software quality monitoring. In *Advances in Intelligent Systems and Computing* (Vol. 474, pp. 463–470). Springer Verlag.

Giraldo, F. D., España, S., Pastor, Ó., & Giraldo, W. J. (2016). Considerations about quality in model-driven engineering. Software Quality Journal, 26(2), 685-750.

Guo, Y., Spínola, R. O., & Seaman, C. (2016). Exploring the costs of technical debt management – a case study. Empirical Software Engineering, 21(1), 159–182.

Jesus, J. S., & Melo, A. C. V. (2017). Technical debt and the software project characteristics. A repository-based exploratory analysis. In Proceedings - 2017 IEEE 19th Conference on Business Informatics, CBI 2017 (Vol. 1, pp. 444–453). Institute of Electrical and Electronics Engineers Inc.

Karamitsos, I., Albarhami, S., & Apostolopoulos, C. (2020). Applying devops practices of continuous automation for machine learning. Information, 11(7), 363.

Klinger, T., Tarr, P., Wagstrom, P., & Williams, C. (2011). An enterprise perspective on technical debt. In Proceedings - International Conference on Software Engineering (pp. 35–38).

Krogh, G., & Spaeth, S. (2007). The open source software phenomenon: Characteristics that promote research. Journal of Strategic Information Systems, 16(3), 236–253.

Lenarduzzi, V., Martini, A., Saarimaki, N., & Tamburri, D. A. (2021). Technical Debt Impacting Lead-Times: An Exploratory Study. In Proceedings - 2021 47th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2021 (pp. 188–195). Institute of Electrical and Electronics Engineers Inc.

Li, H., Qu, Y., Liu, Y., Zhang, T., Ai, J., & Guo, S. (2022). Self-admitted technical debt detection by learning its comprehensive semantics via graph neural networks. Software: Practice and Experience, 52(10), 2152-2176.

Li, Z., Avgeriou, P., & Liang, P. (2015). A systematic mapping study on technical debt and its management. Journal of Systems and Software, 101, 193–220.

Madeyski, L. (2020). Technical debt aware estimations in software engineering: a systematic mapping study. E-Informatica Software Engineering Journal, 14(1).

Malakuti, S., & Heuschkel, J. (2021). The Need for Holistic Technical Debt Management across the Value Stream: Lessons Learnt and Open Challenges. In Proceedings - 2021 IEEE/ACM International Conference on Technical Debt, TechDebt 2021 (pp. 109–113). Institute of Electrical and Electronics Engineers Inc.

Molnar, A. J., & Motogna, S. (2020). Longitudinal evaluation of open-source software maintainability. In ENASE 2020 - Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering (pp. 120–131). SciTePress.

Molnar, A. J., & Motogna, S. (2020). Long-term evaluation of technical debt in open-source software. In International Symposium on Empirical Software Engineering and Measurement. IEEE Computer Society.

Nilsson, M., Eriksson, L. O., & Wästerlund, D. S. (2013). Strategy pattern creation in forest planning in Swedish forest-owning companies. Forests, 4(3), 553-574.

Ramasubbu, N., & Kemerer, C. F. (2013). Towards a model for optimizing technical debt in software products. In 2013 4th International Workshop on Managing Technical Debt, MTD 2013 - Proceedings (pp. 51–54). IEEE Computer Society.

Rantala, L. and Mäntylä, M. (2020). Predicting technical debt from commit contents: reproduction and extension with automated feature selection. Software Quality Journal, 28(4), 1551-1579.

Rule, A., Tabard, A., & Hollan, J. D. (2018). Exploration and explanation in Computational notebooks. In Conference on Human Factors in Computing Systems - Proceedings (Vol. 2018-April). Association for Computing Machinery.

Salentin, J., & Hacks, S. (2020). Towards a Catalog of Enterprise Architecture Smells. In WI2020 Community Tracks (pp. 276–290). GITO Verlag.

Schobel, J., Pryss, R., Probst, T., Schlee, W., Schickler, M., & Reichert, M. (2018). Learnability of a configurator empowering end users to create mobile data collection instruments: usability study (preprint).

Seaman, C., & Guo, Y. (2011). Measuring and Monitoring Technical Debt. Advances in Computers (Vol. 82, pp. 25–46).

Sengik, A. R., Lunardi, G. L., Bianchi, I. S., & Wiedenhöft, G. C. (2022). Using design science research to propose an it governance model for higher education institutions. Education and Information Technologies, 27(8), 11285-11305.

Spínola, R. O., Vetrò, A., Zazworka, N., Seaman, C., & Shull, F. (2013). Investigating technical debt folklore: Shedding some light on technical debt opinion. In 2013 4th International Workshop on Managing Technical Debt, MTD 2013 - Proceedings (pp. 1–7). IEEE Computer Society.

Stanko, M. A., Molina-Castillo, F., & Harmancioğlu, N. (2014). It won't fit! for innovative products, sometimes that's for the best. Journal of Product Innovation Management, 32(1), 122-137.

Stephenson, G., Molotkov, R., Guzman, N. d., & Lafferty, L. (2010). Real-time diagnostics of gas lift systems using intelligent agents: a case study. SPE Production &Amp; Operations, 25(01), 111-123.

Strandberg, P. E., Ostrand, T. J., Weyuker, E. J., Afzal, W., & Sundmark, D. (2020). Intermittently failing tests in the embedded systems domain. In ISSTA 2020 - Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis (pp. 337–348). Association for Computing Machinery, Inc.

Tan, J., Feitosa, D., Avgeriou, P., & Lungu, M. (2021). Evolution of technical debt remediation in Python: A case study on the Apache Software Ecosystem. Journal of Software: Evolution and Process, 33(4).

Wilkinson, S. R., Eisenhauer, G., Kapadia, A. J., Knight, K., Logan, J., Widener, P., & Wolf, M. (2022). F***workflows: when parts of FAIR are missing. In Proceedings - 2022 IEEE 18th International Conference on e-Science, eScience 2022 (pp. 507–512). Institute of Electrical and Electronics Engineers Inc.

Zhang, C. and Wu, Y. (2016). A flowchart for rapid technical debt management decision making. Journal of Software, 11(2), 212-219.