



ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ
«ΓΕΦΥΡΩΝΟΝΤΑΣ ΤΟ ΧΑΣΜΑ ΔΕΞΙΟΤΗΤΩΝ ΣΤΟ
ΕΠΙΣΤΗΜΟΝΙΚΟ ΛΟΓΙΣΜΙΚΟ ΜΕ ΤΗ ΧΡΗΣΗ ΑΙ
ASSISTANTS»



Του φοιτητή
Μπασιά Παναγιώτη
Αρ. Μητρώου: 2020101

Επιβλέπων
Ονοματεπώνυμο Ελβίρα-Μαρία
Αρβανίτου
Βαθμίδα Επίκουρη

Ημερομηνία 01-03-2026

Τίτλος Δ.Ε.ΓΕΦΥΡΩΝΟΝΤΑΣ ΤΟ ΧΑΣΜΑ ΔΕΞΙΟΤΗΤΩΝ ΣΤΟ ΕΠΙΣΤΗΜΟΝΙΚΟ ΛΟΓΙΣΜΙΚΟ
ΜΕ ΤΗ ΧΡΗΣΗ AI ASSISTANTS

Κωδικός Δ.Ε.30126

Ονοματεπώνυμο φοιτητή/τών Μπασιά Παναγιώτη

Ονοματεπώνυμο εισηγητή Ελβίρα-Μαρία Αρβανίτου

Ημερομηνία ανάληψης Δ.Ε.20-02-2026

Ημερομηνία περάτωσης Δ.Ε. 06-09-2026

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

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του φοιτητή Μπασιά Παναγιώτη που την εκπόνησε/αν. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης, ο συγγραφέας/δημιουργός εκχωρεί στο Διεθνές Πανεπιστήμιο της Ελλάδος άδεια χρήσης του δικαιώματος αναπαραγωγής, δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσης της εργασίας διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος. Η ανοικτή πρόσβαση στο πλήρες κείμενο της εργασίας, δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού, ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, πώληση, εμπορική χρήση, διανομή, έκδοση, μεταφόρτωση (downloading), ανάρτηση (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραίτητως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Αφιέρωση»

Περίληψη

Η σύγχρονη επιστημονική έρευνα βασίζεται σε σημαντικό βαθμό σε λογισμικό το οποίο αναπτύσσεται από ερευνητές που συχνά δεν διαθέτουν τυπική εκπαίδευση στη μηχανική λογισμικού. Η κατάσταση αυτή έχει οδηγήσει σε ένα τεκμηριωμένο χάσμα δεξιοτήτων, το οποίο μπορεί να επηρεάσει με μετρήσιμο τρόπο την ποιότητα και την αναπαραγωγικότητα του ερευνητικού λογισμικού. Παράλληλα, η ταχεία εξάπλωση των βοηθών προγραμματισμού με τεχνητή νοημοσύνη δημιουργεί ένα σημαντικό ερώτημα: κατά πόσο τα εργαλεία αυτά συμβάλλουν πραγματικά στη μείωση του συγκεκριμένου χάσματος ή απλώς μεταβάλλουν τον τρόπο με τον οποίο αυτό εμφανίζεται.

Η παρούσα εργασία διερευνά το ερώτημα αυτό μέσω μιας μελέτης εξόρυξης δεδομένων από 119 peer-reviewed αποθετήρια ερευνητικού λογισμικού του Journal of Open Source Software. Τα αποθετήρια χωρίστηκαν σε δύο χρονικές περιόδους, πριν και μετά την ευρεία εμφάνιση των AI coding assistants, και αξιολογήθηκαν με ένα πλαίσιο μέτρησης τριών επιπέδων. Το πλαίσιο εξετάζει το τεχνικό χρέος σε επίπεδο κώδικα, την τεκμηρίωση και την αναπαραγωγικότητα. Η βασική μεθοδολογική θέση της εργασίας είναι ότι η εξέταση μόνο των χαρακτηριστικών του κώδικα δεν επαρκεί για να διαπιστωθεί αν το χάσμα δεξιοτήτων έχει πραγματικά μειωθεί. Για τον διαχωρισμό της γεφύρωσης από την απλή απόκρυψη του προβλήματος απαιτείται και η αξιολόγηση ανώτερων ιδιοτήτων, όπως η τεκμηρίωση και η αναπαραγωγικότητα. Λόγω του observational σχεδιασμού της μελέτης, οι διαφορές μεταξύ των δύο χρονικών περιόδων δεν μπορούν να ερμηνευθούν ως αιτιώδης επίδραση των AI coding assistants.

Τα αποτελέσματα δείχνουν ότι μετά την εμφάνιση των AI coding assistants παρατηρήθηκε περιορισμένη αλλά στατιστικά ανιχνεύσιμη βελτίωση σε συγκεκριμένες μετρούμενες ιδιότητες του λογισμικού. Η βελτίωση αυτή εμφανίζεται κυρίως σε χαρακτηριστικά που μπορούν να επηρεαστούν άμεσα από αυτοματοποιημένα εργαλεία. Συγκεκριμένα, ο κώδικας της εποχής AI παρουσιάζει καλύτερη συμμόρφωση με τους τυπικούς κανόνες ύφους και ορθότητας, ενώ η δομική πολυπλοκότητα, η τεκμηρίωση και η αναπαραγωγικότητα δεν παρουσιάζουν ουσιαστική μεταβολή. Επιπλέον, παρατηρήθηκαν δύο τάσεις χωρίς στατιστική σημαντικότητα: αύξηση της επανάληψης κώδικα και μείωση του αυτο-δηλωμένου τεχνικού χρέους. Και οι δύο τάσεις είναι συμβατές με την υπόθεση ότι ορισμένα προβλήματα εξακολουθούν να υπάρχουν, αλλά αναγνωρίζονται ή καταγράφονται λιγότερο συχνά.

Συνολικά, το παρατηρούμενο μοτίβο δεν υποστηρίζει την ευρεία υπόθεση γεφύρωσης του χάσματος δεξιοτήτων και είναι περισσότερο συμβατό με μια περιορισμένη βελτίωση ορισμένων επιφανειακών, περισσότερο αυτοματοποιήσιμων ιδιοτήτων. Αντίθετα, το συνολικό μοτίβο είναι περισσότερο συμβατό με μια μερική απόκρυψή του. Η βελτίωση εντοπίζεται κυρίως σε πτυχές της ποιότητας που μπορούν να ελεγχθούν ή να βελτιωθούν αυτόματα από εργαλεία, ενώ οι ιδιότητες που απαιτούν βαθύτερη κατανόηση και κρίση από τον προγραμματιστή παραμένουν σε μεγάλο βαθμό αμετάβλητες.

«Bridging the Skills Gap in Scientific Software Using AI Assistants»

«Panagiotis Basias»

Abstract

Modern scientific research depends heavily on software developed by researchers who often lack formal training in software engineering. This creates a documented skills gap that can affect the quality, maintainability, and reproducibility of scientific software. At the same time, the rapid adoption of AI coding assistants raises an important question: whether these tools genuinely help bridge this gap or mainly improve the surface characteristics of the produced code.

This thesis investigates this question through a mining study of 119 peer-reviewed scientific software repositories published in the Journal of Open Source Software (JOSS). The repositories are divided into two cohorts representing periods before and after the widespread emergence of AI coding assistants. They are compared using a multi-dimensional measurement framework covering code-level technical debt, documentation, and reproducibility. In addition, repositories within the Post-AI cohort are examined for explicit disclosure of AI-assisted development.

The results show a limited but measurable improvement in rule conformance during the Post-AI period. Pylint violation density and the related debt ratio decreased significantly, while structural characteristics such as cyclomatic complexity, nesting, and maintainability remained largely unchanged. Documentation and reproducibility also showed no meaningful improvement. Two additional non-significant trends were observed: increased code duplication and reduced self-admitted technical debt.

Overall, the observed pattern is more consistent with partial concealment than with broad bridging of the skills gap. However, because the study is observational and explicit AI disclosure was rare, the results cannot establish a causal effect of AI coding assistants.

Keywords: scientific software, skills gap, technical debt, AI coding assistants, reproducibility, repository mining

Ευχαριστίες

Με την ολοκλήρωση της παρούσας διπλωματικής εργασίας, θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες σε όλους όσοι συνέβαλαν με τον δικό τους τρόπο στην πραγματοποίησή της.

Αρχικά, θα ήθελα να ευχαριστήσω την επιβλέπουσα καθηγήτριά μου για την καθοδήγηση, τις συμβουλές και την υποστήριξή της καθ' όλη τη διάρκεια εκπόνησης της εργασίας. Η συμβολή της ήταν ιδιαίτερα σημαντική τόσο για τη διαμόρφωση της ερευνητικής προσέγγισης όσο και για την ολοκλήρωση της τελικής μορφής της διπλωματικής.

Επίσης, θα ήθελα να ευχαριστήσω την οικογένειά μου και τους ανθρώπους που βρίσκονταν δίπλα μου για την υπομονή, την κατανόηση και τη συνεχή υποστήριξή τους κατά τη διάρκεια των σπουδών μου και ιδιαίτερα κατά την περίοδο εκπόνησης της παρούσας εργασίας.

Τέλος, ευχαριστώ όλους όσοι με βοήθησαν, άμεσα ή έμμεσα, στην ολοκλήρωση αυτής της προσπάθειας.

Περιεχόμενα

Περίληψη.....	vi
Abstract	vii
Ευχαριστίες	viii
Περιεχόμενα	ix
Κατάλογος Σχημάτων	xii
Κατάλογος Πινάκων.....	xii
Συντομογραφίες.....	xiii
Κεφάλαιο 1ο: Το Χάσμα Δεξιοτήτων στο Επιστημονικό Λογισμικό και οι AI Coding Assistants .1	
1.1 Εισαγωγή.....	1
1.2 Το Χάσμα Δεξιοτήτων	2
1.3 Το Τεχνικό Χρέος ως Αναλυτικό Πλαίσιο	3
1.4 Βοηθοί Προγραμματισμού με AI: Γέφυρα ή Επίφαση;	4
1.5 Διατύπωση του Προβλήματος και Ερευνητικά Ερωτήματα	5
1.6 Συνεισφορές	6
1.7 Δομή της Διπλωματικής Εργασίας.....	7
Κεφάλαιο 2ο: Σχετική Εργασία	8
2.1 Εισαγωγή.....	8
2.2 Μηχανική Επιστημονικού Λογισμικού και το Χάσμα Δεξιοτήτων	8
2.3 Τεχνικό Χρέος και Ποιότητα Επιστημονικού Λογισμικού	9
2.4 Βοηθοί Προγραμματισμού με Τεχνητή Νοημοσύνη.....	10
2.5 AI Assistance, Τεχνικό Χρέος και το Ερευνητικό Κενό	12
Κεφάλαιο 3ο: Μεθοδολογία.....	14
3.1 Επισκόπηση της Μελέτης.....	14
3.2 Ερευνητικός Στόχος και Ερωτήματα.....	14
3.3 Σχεδιασμός της Μελέτης.....	16
3.3.1 Τύπος Σχεδιασμού	16
3.3.2 Γιατί Mining αντί για Experiment	16
3.3.3 Blinding και Ordering.....	16
3.4 Κατασκευή του Corpus	17
3.4.1 Sampling Frame.....	17
3.4.2 Ορισμοί των Εποχών	17
3.4.3 Κριτήρια Ένταξης και Αποκλεισμού.....	18

3.4.4	Snapshot Resolution	18
3.4.5	Αναπαραγωγιμότητα της Κατασκευής του Corpus	19
3.4.6	Το Τελικό Σύνολο Δεδομένων	21
3.4.7	Ανάλυση της Ροής Επιλογής του Corpus	22
3.5	Attribution της AI Assistance.....	25
3.5.1	Disclosure Signals	25
3.5.2	Manual Review of Borderline Matches.....	25
3.5.3	Η Ασυμμετρία του Non-Disclosure.....	25
3.6	Υποδομή Μέτρησης	26
3.6.1	Reused, Validated Analysers	27
3.6.2	Εντοπισμός Self-Admitted Technical Debt	27
3.6.3	History Mining	28
3.6.4	Covariates	29
3.6.5	Επιχειρησιακοποίηση, Παραδοχές και Περιορισμοί των Μετρικών.....	29
3.7	Πλαίσιο Μέτρησης.....	35
3.8	Πλάνο Ανάλυσης.....	37
3.8.1	Βασικές Συγκρίσεις	37
3.8.2	Προκαθορισμένες Προβλέψεις Pattern.....	38
3.8.3	Εξέταση Confounding	38
3.8.4	Descriptive Reporting.....	38
3.9	Επικύρωση των Εργαλείων	38
3.10	Ηθικά Ζητήματα.....	39
Κεφάλαιο 4ο:	Αποτελέσματα.....	40
4.1	Πώς Διαβάζονται οι Πίνακες Αποτελεσμάτων	40
4.2	RQ1: Άλλαξε το Τεχνικό Χρέος σε Επίπεδο Κώδικα;	40
4.2.1	Τι Βελτιώθηκε	42
4.2.2	Τι Δεν Άλλαξε	43
4.2.3	Δύο Τάσεις που Δείχνουν προς την Άλλη Κατεύθυνση	43
4.2.4	Απάντηση στο RQ1	43
4.3	RQ2: Άλλαξε η Ποιότητα του Ερευνητικού Λογισμικού;	44
4.3.1	Απάντηση στο RQ2	44
4.4	RQ3: Δηλωμένη έναντι Μη Δηλωμένης Χρήσης AI	44
4.5	Έλεγχοι Συγχυτικών Παραγόντων και Robustness	45
4.6	RQ4: Συνδυάζοντας το Συνολικό Μοτίβο.....	45
4.6.1	Το Συμπέρασμα που Μπορεί να Υποστηριχθεί.....	46

4.7	Σύνοψη των Ευρημάτων	46
Κεφάλαιο 5ο:	Συμπεράσματα	48
5.1	Επισκόπηση.....	48
5.2	Κύρια Ευρήματα	48
5.3	Συζήτηση των Ευρημάτων	49
5.3.1	Βελτίωση της συμμόρφωσης χωρίς αντίστοιχη δομική βελτίωση	50
5.3.2	Duplication και Self-Admitted Technical Debt ως ενδεικτικές τάσεις	51
5.3.3	Τεκμηρίωση και αναπαραγωγιμότητα: γιατί το null αποτέλεσμα έχει σημασία	52
5.3.4	Bridging, Null και Concealment.....	53
5.3.5	Disclosure, attribution και provenance debt	54
5.3.6	Εναλλακτικές εξηγήσεις και σημασία του observational σχεδιασμού	55
5.3.7	Επιπτώσεις για την ανάπτυξη επιστημονικού λογισμικού	56
5.3.8	Συνολική ερμηνεία	56
5.4	Περιορισμοί και Απειλές για την Εγκυρότητα.....	57
5.4.1	Internal Validity.....	57
5.4.2	Construct Validity.....	58
5.4.3	External Validity.....	58
5.4.4	Conclusion Validity	58
5.5	Μελλοντική Εργασία.....	59
5.6	Καταληκτική Παρατήρηση	59
	BIBΛΙΟΓΡΑΦΙΑ.....	60

Κατάλογος Σχημάτων

Σχήμα 3.1: Ροή της διαδικασίας συλλογής, επεξεργασίας και ανάλυσης των repositories	26
Σχήμα 4.1: Σύγκριση των κύριων μετρικών μεταξύ των Pre και Post cohorts	42

Κατάλογος Πινάκων

Πίνακας 3.1: Παράθυρα των Cohorts.....	18
Πίνακας 3.2: Από όλες τις εργασίες του JOSS στο τελικό σύνολο ανάλυσης.....	21
Πίνακας 3.3: Πλαίσιο μέτρησης:έντεκα κατηγορίες μετρικών σε τρία tiers.....	36
Πίνακας 4.1: Σύγκριση μεταξύ των εποχών για όλες τις μετρικές.....	41

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΙΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
AI	Artificial Intelligence
API	Application Programming Interface
CI	Continuous Integration
IQR	Interquartile Range
JOSS	Journal of Open Source Software
KLOC	Thousand Lines of Code
LLM	Large Language Model
RQ	Research Question
SATD	Self-Admitted Technical Debt
TD	Technical Debt

Κεφάλαιο 1ο: Το Χάσμα Δεξιοτήτων στο Επιστημονικό Λογισμικό και οι AI Coding Assistants

1.1 Εισαγωγή

Η σύγχρονη επιστήμη βασίζεται στο λογισμικό. Οι κλιματικές προβλέψεις, οι προβλέψεις της δομής πρωτεϊνών, οι ανιχνεύσεις βαρυτικών κυμάτων και τα επιδημιολογικά μοντέλα, τα αποτελέσματα που καθορίζουν τη σύγχρονη έρευνα είναι, σχεδόν χωρίς εξαίρεση, αποτελέσματα προγραμμάτων. Έρευνες σχετικά με τις πρακτικές της έρευνας έχουν καταγράψει την έκταση αυτής της εξάρτησης για περισσότερο από μία δεκαετία. Σε μια έρευνα σχεδόν δύο χιλιάδων επιστημόνων [1], διαπιστώθηκε ότι οι ερευνητές αφιερώνουν περίπου το 30% του χρόνου εργασίας τους στην ανάπτυξη λογισμικού και σημαντικά περισσότερο χρόνο στη χρήση του, ενώ αναφέρθηκε ότι η υπολογιστική εργασία κυριαρχεί στην καθημερινή πρακτική των ερευνητών σε διάφορους κλάδους σε ένα μεγάλο ερευνητικό πανεπιστήμιο [2]. Στις Ηνωμένες Πολιτείες, το 95% των ερωτηθέντων μεταδιδακτορικών ερευνητών δήλωσε ότι χρησιμοποιεί ερευνητικό λογισμικό και, στις περισσότερες περιπτώσεις, λογισμικό που είχαν γράψει οι ίδιοι [3].

Οι άνθρωποι που γράφουν αυτό το λογισμικό δεν είναι, στη συντριπτική τους πλειονότητα, μηχανικοί λογισμικού. Είναι φυσικοί, βιολόγοι, χημικοί και μαθηματικοί, οι οποίοι απέκτησαν γνώσεις προγραμματισμού με τον ίδιο τρόπο που αποκτά κανείς μια εργαστηριακή τεχνική: από συναδέλφους, μέσω μίμησης και υπό την πίεση προθεσμιών. Διαπιστώθηκε ότι η γνώση που απαιτείται για την ανάπτυξη επιστημονικού λογισμικού αποκτάται κυρίως μέσω αυτοδιδασκαλίας και άτυπης ανταλλαγής γνώσεων και όχι μέσω τυπικής εκπαίδευσης [1]. Η παρατήρηση του Wilson από το έργο Software Carpentry παραμένει η πιο συνοπτική διατύπωση της συνέπειας: οι επιστήμονες «αφιερώνουν υπερβολικά πολύ χρόνο παλεύοντας με το λογισμικό, αντί να κάνουν έρευνα, αλλά δεν έχουν ιδέα πόσο αξιόπιστα ή αποδοτικά είναι τα προγράμματά τους» [4].

Αυτό θα είχε μικρότερη σημασία εάν το λογισμικό ήταν περιφερειακό για την επιστήμη. Δεν είναι. Οι Hatton και Roberts [5], σε μια κλασική μελέτη λογισμικού επεξεργασίας σεισμικών δεδομένων, υπέβαλαν εννέα ανεξάρτητες υλοποιήσεις των ίδιων δημοσιευμένων αλγορίθμων σε πανομοιότυπες εισόδους και διαπίστωσαν ότι τα αποτελέσματά τους διέφεραν μεταξύ τους πολύ περισσότερο από την αναμενόμενη αριθμητική ανοχή, η διαφορά μεταξύ υποτίθεται ισοδύναμων επιστημονικών προγραμμάτων ήταν αρκετά μεγάλη ώστε να επηρεάσει τα γεωλογικά συμπεράσματα που προέκυπταν από αυτά. Δύο δεκαετίες αργότερα, ο Merali [6], γράφοντας στο Nature, κατέγραψε περιπτώσεις στις οποίες σφάλματα λογισμικού οδήγησαν στην ανάκληση δημοσιευμένων αποτελεσμάτων, συμπεριλαμβανομένων σημαντικών εργασιών στη δομική βιολογία, των οποίων τα βασικά ευρήματα αντιστράφηκαν εξαιτίας ενός και μόνο σφάλματος στον κώδικα. Οι Jorppa et al. [7], στο Science, κατέγραψαν ότι οι επιστήμονες υιοθετούν συχνά λογισμικό μοντελοποίησης με βάση τη διαθεσιμότητα και τη συνήθεια και όχι την επαληθευμένη ορθότητα, χαρακτηρίζοντας αυτή την πρακτική ως «ανησυχητική» για την αξιοπιστία των αποτελεσμάτων που βασίζονται σε αυτό.

Οι επιπτώσεις ξεπερνούν τις μεμονωμένες ανακλήσεις δημοσιεύσεων. Η αξιοπιστία της υπολογιστικής επιστήμης βασίζεται στην αναπαραγωγικότητα των αποτελεσμάτων της και τα διαθέσιμα στοιχεία σχετικά με αυτό δεν είναι ενθαρρυντικά. Η ευρέως αναφερόμενη έρευνα του Baker σε 1.576 ερευνητές διαπίστωσε ότι περισσότεροι από τους μισούς θεωρούσαν πως η επιστήμη βρίσκεται σε σημαντική κρίση αναπαραγωγικότητας και ότι πάνω από το 70% είχε προσπαθήσει και αποτύχει να αναπαράγει πειράματα άλλου επιστήμονα [8]. Ειδικά για την υπολογιστική εργασία, οι Stodden et al.

[9] προσπάθησαν να αποκτήσουν και να εκτελέσουν τα τεχνουργήματα που βρίσκονταν πίσω από 204 άρθρα που είχαν δημοσιευθεί στο Science, ένα περιοδικό με σαφή πολιτική κοινοποίησης κώδικα και δεδομένων, και κατάφεραν να αναπαράγουν τα ευρήματα μόνο του 26%. Η απόσταση μεταξύ όσων υποστηρίζει η δημοσιευμένη υπολογιστική επιστήμη και όσων μπορούν πραγματικά να αποδείξουν τα αντίστοιχα τεχνουργήματά της αποτελεί, σε μεγάλο βαθμό, ένα χάσμα ποιότητας λογισμικού.

Η μηχανική λογισμικού, ως επιστημονικός κλάδος, υπάρχει ακριβώς επειδή η ανάπτυξη αξιόπιστου λογισμικού είναι δύσκολη και επειδή οι συστηματικές πρακτικές αποδεδειγμένα βοηθούν. Ο έλεγχος εκδόσεων, οι δοκιμές, η αναθεώρηση κώδικα, ο αρθρωτός σχεδιασμός και η διαχείριση εξαρτήσεων δεν είναι ακαδημαϊκές πολυτέλειες· αποτελούν τις συσσωρευμένες πρακτικές άμυνας μιας βιομηχανίας που έμαθε, με μεγάλο κόστος, τι συμβαίνει όταν αυτές απουσιάζουν. Το ερώτημα από το οποίο ξεκινά η παρούσα διπλωματική εργασία είναι γιατί αυτές οι πρακτικές έχουν υιοθετηθεί σε τόσο περιορισμένο βαθμό στην ανάπτυξη επιστημονικού λογισμικού, και αν, και κατά πόσο, μια νέα γενιά εργαλείων ανάπτυξης βασισμένων στην ΑΙ αλλάζει την απάντηση.

1.2 Το Χάσμα Δεξιότητων

Η ελλιπής υιοθέτηση πρακτικών μηχανικής λογισμικού στην επιστήμη δεν αποτελεί αντικείμενο διαφωνίας· είναι ένα από τα καλύτερα τεκμηριωμένα γεγονότα σχετικά με το ερευνητικό λογισμικό. Οι Arvanitou et al. [10], στη συστηματική μελέτη χαρτογράφησης 359 πρωτογενών μελετών που αποτελεί βασικό σημείο αναφοράς για την παρούσα εργασία, συνθέτουν τα διαθέσιμα στοιχεία: παρόλο που οι επιστήμονες επενδύουν περισσότερο από το 40% του χρόνου τους στην ανάπτυξη λογισμικού, σε μεγάλο βαθμό δεν αξιοποιούν τις εξελίξεις της μηχανικής λογισμικού [11, 12].

Μόνο περίπου οι μισοί από τους μεταδιδακτορικούς ερευνητές που συμμετείχαν στην έρευνα των Nangia και Katz [3] είχαν λάβει οποιαδήποτε εκπαίδευση στην ανάπτυξη λογισμικού, ενώ το 75% των ερευνητών που χρηματοδοτούνταν από το National Science Foundation των ΗΠΑ δήλωσε ότι δεν είχε χρόνο για τέτοιου είδους εκπαίδευση [10]. Η εκτίμηση του Wilson ότι μόνο περίπου οι μισοί επιστήμονες γνωρίζουν τις βασικές αρχές των δοκιμών [4] συμφωνεί με το εύρημα των Hannay et al., σύμφωνα με το οποίο, ενώ πολλοί επιστήμονες θεωρούν τις δοκιμές σημαντικές, πολύ λιγότεροι πιστεύουν ότι τις κατανοούν αρκετά καλά ώστε να τις εφαρμόσουν [1].

Ο Segal [13], σε μια μελέτη περίπτωσης μηχανικών λογισμικού που εργάζονταν μαζί με ερευνητές, εντόπισε την πολιτισμική ρίζα του χάσματος: το επιστημονικό λογισμικό γράφεται από τους ίδιους τους τελικούς χρήστες του, εξελίσσεται μαζί με το ερευνητικό ερώτημα και αξιολογείται με βάση το αν παράγει αποτελέσματα που μπορούν να δημοσιευθούν και όχι με βάση κάποια ιδιότητα του ίδιου του λογισμικού. Υπό αυτά τα κίνητρα, πρακτικές των οποίων το όφελος εμφανίζεται αργότερα, όπως οι δοκιμές, η τεκμηρίωση και η καθαρή δομή, αναβάλλονται συστηματικά.

Οι Heaton και Carver [12], εξετάζοντας ισχυρισμούς σχετικά με τις πρακτικές μηχανικής λογισμικού στην υπολογιστική επιστήμη, διαπίστωσαν ότι οι πρακτικές που οι ίδιοι οι επιστήμονες αξιολογούν ως σημαντικότερες, δηλαδή η επαλήθευση και οι δοκιμές, είναι ακριβώς αυτές που υιοθετούνται λιγότερο.

Οι συνέπειες αυτής της αναβολής έχουν ένα όνομα στη βιβλιογραφία της μηχανικής λογισμικού, το οποίο εισάγεται στην επόμενη ενότητα. Αυτό που έχει σημασία εδώ είναι η μορφή του προβλήματος: ένας μεγάλος και αυξανόμενος αριθμός προγραμματιστών παράγει κρίσιμη επιστημονική υποδομή χωρίς να διαθέτει την εκπαίδευση για να αξιολογήσει την ποιότητά της, υπό κίνητρα που επιβραβεύουν την ταχύτητα περισσότερο από την ορθότητα, σε ένα περιβάλλον, όπως

τεκμηριώνεται από τους Schmidberger και Brüggel [14], όπου πάνω από το 70% των εφαρμογών υπολογιστικής υψηλής απόδοσης εξυπηρετούν τομείς εκτός της επιστήμης των υπολογιστών.

Αυτό είναι το χάσμα δεξιοτήτων που αναφέρεται στον τίτλο της παρούσας εργασίας. Δεν αποτελεί προσβολή προς τους επιστήμονες· είναι ένα δομικό χαρακτηριστικό του τρόπου με τον οποίο δημιουργείται το ερευνητικό λογισμικό.

1.3 Το Τεχνικό Χρέος ως Αναλυτικό Πλαίσιο

Για την κατανόηση των συνεπειών του χάσματος δεξιοτήτων, η παρούσα διπλωματική εργασία υιοθετεί την έννοια του τεχνικού χρέους (Technical Debt, TD). Η μεταφορά προέρχεται από τον Cunningham [15], ο οποίος περιέγραψε την κυκλοφορία μη ώριμου κώδικα ως «είσοδο σε χρέος»: ένα μικρό χρέος επιταχύνει την ανάπτυξη, υπό την προϋπόθεση ότι αποπληρώνεται άμεσα, αλλά κάθε λεπτό που δαπανάται για την αντιμετώπιση κώδικα που δεν είναι πλήρως σωστό υπολογίζεται ως τόκος, ενώ ολόκληροι οργανισμοί μηχανικής μπορούν να οδηγηθούν σε αδιέξοδο υπό το βάρος μιας μη ενοποιημένης υλοποίησης. Ο McConnell [16] βελτίωσε την έννοια διακρίνοντας το χρέος που αναλαμβάνεται σκόπιμα, ως στρατηγική επιλογή, από το χρέος που δημιουργείται εν αγνοία μας, μια διάκριση που έχει προφανή σημασία για προγραμματιστές οι οποίοι δεν διαθέτουν την εκπαίδευση ώστε να αναγνωρίζουν την επιλογή που κάνουν. Οι Kruchten, Nord και Ozkaya [17] περιγράφουν τον τρόπο με τον οποίο η μεταφορά εξελίχθηκε από ένα ρητορικό εργαλείο σε ένα αναλυτικό πλαίσιο, ενώ οι Brown et al. [18] επέκτειναν το πεδίο εφαρμογής της πέρα από τον κώδικα, ώστε να περιλαμβάνει την αρχιτεκτονική, την τεκμηρίωση, τις δοκιμές και τα υπόλοιπα τεχνουργήματα της ανάπτυξης.

Το πλαίσιο έφτασε σε συστηματική ωριμότητα με τη μελέτη χαρτογράφησης των Li, Avgeriou και Liang [19], η οποία αποτελεί το δεύτερο βασικό σημείο αναφοράς της παρούσας εργασίας. Εξετάζοντας 94 πρωτογενείς μελέτες, ταξινομούν το TD σε δέκα τύπους, μεταξύ άλλων, χρέος κώδικα, σχεδιασμού, αρχιτεκτονικής, δοκιμών, τεκμηρίωσης, build και υποδομής, και εντοπίζουν οκτώ δραστηριότητες διαχείρισης, από τις οποίες η αναγνώριση, η μέτρηση και η αποπληρωμή κυριαρχούν στη βιβλιογραφία. Σημαντικό για την παρούσα εργασία είναι ότι καταγράφουν την υποστήριξη εργαλείων που είναι διαθέσιμη για κάθε τύπο: το χρέος σε επίπεδο κώδικα μπορεί να μετρηθεί με ώριμες υποδομές στατικής ανάλυσης, γεγονός που καθιστά το TD ένα πλαίσιο που μπορεί να εφαρμοστεί πρακτικά και όχι απλώς μια περιγραφική μεταφορά. Εμπειρικές εργασίες έκτοτε έχουν συνδέσει τη μεταφορά με μεγέθη που ενδιαφέρουν τους επιστήμονες: οι Besker et al. [20] συνθέτουν στοιχεία που δείχνουν ότι το αρχιτεκτονικό χρέος καταναλώνει μετρήσιμο μέρος της προσπάθειας ανάπτυξης, ενώ οι Tom et al. [21] χαρτογραφούν τις αιτίες και τις συνέπειές του σε ολόκληρο τον κύκλο ζωής.

Δύο χαρακτηριστικά καθιστούν το TD κατάλληλο πλαίσιο ειδικά για το επιστημονικό λογισμικό. Πρώτον, η μεταφορά του χρέους περιγράφει την αναβολή υπό την πίεση χρόνου, την οποία ο Segal [13] εντόπισε ως καθοριστική συνθήκη του ερευνητικού προγραμματισμού: ο επιστήμονας που παραλείπει τις δοκιμές για να προλάβει μια προθεσμία συνεδρίου λαμβάνει ακριβώς το δάνειο που περιέγραψε ο Cunningham, συνήθως χωρίς να γνωρίζει το επιτόκιο. Δεύτερον, το πλαίσιο του TD μπορεί ρητά να επεκταθεί σε νέους τύπους χρέους και πρόσφατες εργασίες το έχουν επεκτείνει στο πλαίσιο που εξετάζει η παρούσα εργασία. Οι Bogner et al. [22] εντόπισαν τέσσερις νέους τύπους χρέους σε συστήματα βασισμένα στην AI, χρέος δεδομένων, μοντέλου, διαμόρφωσης και ηθικής, ενώ οι Alves et al. [23] παρέχουν τη μεθοδολογία ταξινόμησης μέσω της οποίας οργανώνονται τέτοιες επεκτάσεις. Πιο άμεσα, οι Moreschini et al. [24], το τρίτο βασικό σημείο αναφοράς της παρούσας εργασίας, καταγράφουν την εξέλιξη του TD από την εποχή του DevOps στην εποχή της παραγωγικής AI και εντοπίζουν τους τύπους χρέους που εισάγει η ίδια η ανάπτυξη με τη βοήθεια AI: χρέος διακυβέρνησης,

χρέος επεξηγησιμότητας και χρέος που σχετίζεται με τα prompts, κώδικας του οποίου η προέλευση και η λογική δεν καταγράφονται πουθενά, επειδή δημιουργήθηκε μέσα σε μια συνομιλία με ένα μοντέλο. Η τελευταία αυτή παρατήρηση επαναδιατυπώνει το ερώτημα που θέτει η παρούσα εργασία και η επόμενη ενότητα το διατυπώνει.

1.4 Βοηθοί Προγραμματισμού με AI: Γέφυρα ή Επίφαση;

Ενώ η κοινότητα της μηχανικής λογισμικού συστηματοποιούσε το τεχνικό χρέος, μια διαφορετική γραμμή έρευνας άλλαζε το ποιος, και τι, γράφει κώδικα. Οι Chen et al. [25] έδειξαν ότι μεγάλα γλωσσικά μοντέλα εκπαιδευμένα σε πηγαίο κώδικα μπορούν να συνθέσουν λειτουργικά προγράμματα από περιγραφές σε φυσική γλώσσα, εισάγοντας το benchmark HumanEval και το μοντέλο Codex, το οποίο θα αποτελούσε τη βάση για τον πρώτο βοηθό προγραμματισμού AI μαζικής αγοράς. Μέσα σε δύο χρόνια, τέτοιοι βοηθοί μετακινήθηκαν από ερευνητικές επιδείξεις σε προεπιλεγμένα στοιχεία του περιβάλλοντος ανάπτυξης και οι πρώτες αναφορές πεδίου από τους δημιουργούς τους περιέγραψαν σημαντικές επιδράσεις στην εμπειρία των προγραμματιστών [26].

Τα εμπειρικά στοιχεία σχετικά με τους AI coding assistants παρουσιάζουν μια σύνθετη εικόνα. Από τη μία πλευρά, αρκετές μελέτες έχουν δείξει ότι τα εργαλεία αυτά μπορούν να προσφέρουν σημαντικά οφέλη παραγωγικότητας. Οι Peng et al. [27], σε ελεγχόμενο πείραμα με το GitHub Copilot, διαπίστωσαν ότι οι συμμετέχοντες που είχαν πρόσβαση στο εργαλείο ολοκλήρωσαν μια προγραμματιστική εργασία σημαντικά ταχύτερα από την ομάδα ελέγχου, με μεγαλύτερα οφέλη για λιγότερο έμπειρους developers. Αντίστοιχα, οι Ziegler et al. [28] έδειξαν ότι η αποδοχή προτάσεων από neural code-completion εργαλεία συνδέεται με υψηλότερη αντιλαμβανόμενη παραγωγικότητα. Ωστόσο, οι Vaithilingam et al. [29] διαπίστωσαν ότι η προτίμηση των χρηστών για τέτοια εργαλεία δεν συνοδεύεται πάντα από αντίστοιχη βελτίωση στον αντικειμενικά μετρημένο χρόνο ολοκλήρωσης ή στην επιτυχία των εργασιών.

Από την πλευρά της ποιότητας, η εικόνα είναι περισσότερο σύνθετη. Οι Pearce et al. [30] εντόπισαν σημαντικό ποσοστό ευάλωτων λύσεων σε κώδικα που παρήχθη από το GitHub Copilot, ενώ οι Perry et al. [31] έδειξαν ότι η χρήση AI assistance μπορεί να συνοδεύεται τόσο από προβλήματα ασφάλειας όσο και από αυξημένη αυτοπεποίθηση των developers ως προς την ορθότητα του παραγόμενου κώδικα. Τα αποτελέσματα δεν είναι ομοιόμορφα, καθώς οι Sandoval et al. [32] κατέγραψαν διαφορετική εικόνα, ενώ οι Dakhel et al. [33] ανέδειξαν τον ρόλο της εμπειρίας του developer στην αξιολόγηση των προτάσεων του assistant.

Επιπλέον, η ποιότητα του παραγόμενου κώδικα φαίνεται να εξαρτάται από τον τρόπο αλληλεπίδρασης με το εργαλείο. Οι Mastropaolo et al. [34] έδειξαν ότι ακόμη και σημασιολογικά ισοδύναμες αναδιατυπώσεις ενός prompt μπορούν να οδηγήσουν σε διαφορετικές λύσεις, οι Yetistiren et al. [35] κατέγραψαν διαφορές μεταξύ διαφορετικών AI coding assistants, ενώ οι Barke et al. [36] έδειξαν ότι οι developers μπορούν να χρησιμοποιούν τα εργαλεία είτε διερευνητικά είτε με στόχο τη γρήγορη επιτάχυνση της ανάπτυξης, όπου οι προτάσεις ενδέχεται να γίνονται αποδεκτές με περιορισμένο έλεγχο.

Συνολικά, τα ευρήματα αυτά περιγράφουν μια ουσιαστική ένταση: οι AI coding assistants μπορούν να μειώσουν το κόστος παραγωγής κώδικα και να υποστηρίξουν ιδιαίτερα τους λιγότερο έμπειρους developers, χωρίς όμως να εξασφαλίζουν αντίστοιχη αυτοματοποίηση της αξιολόγησης του παραγόμενου κώδικα. Η αξιολόγηση εξακολουθεί να απαιτεί γνώσεις και κρίση που ενδέχεται να μην έχουν αποκτηθεί μέσω συστηματικής εκπαίδευσης στη μηχανική λογισμικού. Αυτό είναι ιδιαίτερα σημαντικό για το επιστημονικό λογισμικό και οδηγεί στις δύο εναλλακτικές ερμηνείες που εξετάζει η παρούσα εργασία.

Η υπόθεση της γεφύρωσης. Ο βοηθός υποκαθιστά την εκπαίδευση που λείπει. Παρέχει τις συνήθεις πρακτικές προγραμματισμού, τον βασικό κώδικα, τη δομή των δοκιμών και τη γνώση των APIs που ο επιστήμονας δεν έμαθε ποτέ και, με αυτόν τον τρόπο, αυξάνει το ελάχιστο επίπεδο ποιότητας του επιστημονικού λογισμικού. Τα στοιχεία σχετικά με την παραγωγικότητα [27, 28] και η παρατήρηση ότι οι αρχάριοι αποκομίζουν τα μεγαλύτερα οφέλη [27] είναι συμβατά με αυτή την ερμηνεία.

Η υπόθεση της απόκρυψης. Ο βοηθός καλύπτει το χάσμα χωρίς να το κλείνει. Παράγει τεχνουργήματα των οποίων η επιφανειακή ποιότητα ξεπερνά την ικανότητα του δημιουργού τους να τα αξιολογήσει, μετατοπίζοντας το χρέος από το ορατό επίπεδο του κώδικα, όπου τα εργαλεία των Li et al. [19] μπορούν τουλάχιστον να το εντοπίσουν, σε αόρατες μορφές: λανθασμένους αριθμητικούς υπολογισμούς πίσω από εύλογα αποτελέσματα [6], μη τεκμηριωμένη προέλευση [24] και αδικαιολόγητη αυτοπεποίθηση [31]. Τα στοιχεία σχετικά με την ασφάλεια [30, 31] και την ασυμμετρία μεταξύ ειδικών και αρχάριων [33] είναι συμβατά με αυτή την ερμηνεία. Οι δύο υποθέσεις δεν είναι αμοιβαία αποκλειόμενες, το ίδιο εργαλείο μπορεί να γεφυρώνει το χάσμα σε μία διάσταση ενώ το αποκρύπτει σε μια άλλη, και ακριβώς γι' αυτό το ερώτημα απαιτεί μέτρηση σε πολλαπλές διαστάσεις και όχι μια απλή απόφαση υπέρ της μίας υπόθεσης.

1.5 Διατύπωση του Προβλήματος και Ερευνητικά Ερωτήματα

Η υπάρχουσα βιβλιογραφία έχει τεκμηριώσει ότι το επιστημονικό λογισμικό αναπτύσσεται συχνά από ερευνητές χωρίς τυπική εκπαίδευση στη μηχανική λογισμικού, γεγονός που μπορεί να επηρεάσει την ποιότητα, τη συντηρησιμότητα και την αναπαραγωγικότητα του παραγόμενου λογισμικού [10], [12]. Παράλληλα, η ταχεία εξάπλωση των AI coding assistants έχει δημιουργήσει νέες δυνατότητες αυτοματοποίησης της διαδικασίας ανάπτυξης, ιδιαίτερα για λιγότερο έμπειρους προγραμματιστές [27], [28].

Ωστόσο, οι περισσότερες εμπειρικές μελέτες σχετικά με τους AI coding assistants βασίζονται σε controlled experiments, user studies ή περιορισμένες προγραμματιστικές εργασίες και εξετάζουν κυρίως την παραγωγικότητα, τη λειτουργικότητα ή την ασφάλεια του παραγόμενου κώδικα [27]–[35]. Παραμένει επομένως ασαφές αν η περίοδος ευρείας διάδοσης αυτών των εργαλείων συνοδεύεται από αντίστοιχες μεταβολές στις ιδιότητες πραγματικού, δημοσιευμένου επιστημονικού λογισμικού.

Η παρούσα διπλωματική εργασία εξετάζει το ζήτημα αυτό μέσω μιας μελέτης εξόρυξης δεδομένων από πραγματικά, peer-reviewed repositories του Journal of Open Source Software. Τα repositories χωρίζονται σε δύο χρονικά cohorts, ένα πριν και ένα μετά την ευρεία εμφάνιση των AI coding assistants, και συγκρίνονται ως προς χαρακτηριστικά τεχνικού χρέους σε επίπεδο κώδικα, τεκμηρίωσης και αναπαραγωγικότητας. Επιπλέον, μέσα στο Post cohort εξετάζονται ξεχωριστά τα repositories στα οποία υπάρχει ρητή δήλωση χρήσης AI assistance.

Ο σχεδιασμός είναι observational και επομένως οι διαφορές μεταξύ των δύο χρονικών περιόδων δεν μπορούν από μόνες τους να αποδοθούν αιτιωδώς στη χρήση AI coding assistants. Οι δύο εποχές διαφέρουν και ως προς άλλους παράγοντες, όπως η εξέλιξη του Python ecosystem, η διάδοση εργαλείων linting και continuous integration και οι μεταβολές στις πρακτικές ανάπτυξης. Για τον λόγο αυτό, ο στόχος της μελέτης δεν είναι να αποδείξει ότι οι AI assistants προκάλεσαν κάποια συγκεκριμένη μεταβολή, αλλά να εξετάσει αν το συνολικό μοτίβο που παρατηρείται στο δημοσιευμένο επιστημονικό λογισμικό είναι περισσότερο συμβατό με τη γεφύρωση ή με την απόκρυψη του χάσματος δεξιοτήτων.

Με βάση το παραπάνω πρόβλημα, η εργασία διατυπώνει τα ακόλουθα τέσσερα ερευνητικά ερωτήματα:

RQ1 – Τεχνικό χρέος σε επίπεδο κώδικα μεταξύ των εποχών.

Πώς διαφέρουν τα χαρακτηριστικά του τεχνικού χρέους σε επίπεδο κώδικα του ερευνητικού λογισμικού, όπως το debt ratio, οι παραβιάσεις κώδικα, η πολυπλοκότητα, η επανάληψη κώδικα και το Self-Admitted Technical Debt, μεταξύ repositories που έγιναν αποδεκτά πριν και μετά την εμφάνιση των AI coding assistants;

RQ2 – Ποιότητα ερευνητικού λογισμικού μεταξύ των εποχών.

Πώς διαφέρουν τα χαρακτηριστικά της τεκμηρίωσης και της αναπαραγωγιμότητας του ερευνητικού λογισμικού μεταξύ των δύο χρονικών περιόδων;

RQ3 – Attribution μέσα στην Post εποχή.

Μέσα στο Post cohort, πώς διαφέρουν στις ίδιες μετρικές τα repositories που δηλώνουν ρητά τη χρήση AI assistance από τα σύγχρονα repositories στα οποία δεν υπάρχει αντίστοιχη δήλωση;

RQ4 – Γεφύρωση ή απόκρυψη του χάσματος δεξιοτήτων.

Συνολικά, σε ποιο βαθμό το μοτίβο των διαφορών που παρατηρείται στα RQ1–RQ3 είναι συμβατό με την υπόθεση ότι η AI assistance συμβάλλει στη γεφύρωση του χάσματος δεξιοτήτων ή, αντίθετα, με την υπόθεση ότι βελτιώνει κυρίως επιφανειακές ιδιότητες του λογισμικού χωρίς αντίστοιχη βελτίωση σε ιδιότητες που απαιτούν βαθύτερη ανθρώπινη κρίση;

Το RQ4 αποτελεί τη συνθετική διάσταση της εργασίας. Δεν απαντάται από μία μεμονωμένη μετρική ή ένα στατιστικά σημαντικό αποτέλεσμα, αλλά από το συνολικό μοτίβο των διαφορών μεταξύ των δύο εποχών και, όπου αυτό είναι δυνατό, από τη σύγκριση μεταξύ repositories με και χωρίς ρητή δήλωση χρήσης AI assistance. Η πλήρης επιχειρησιακοποίηση των ερευνητικών ερωτημάτων, οι χρησιμοποιούμενες μετρικές και το σχέδιο ανάλυσης παρουσιάζονται στο Κεφάλαιο 3.

1.6 Συνεισφορές

Η παρούσα διπλωματική εργασία έχει τέσσερις βασικές συνεισφορές.

C1. Εμπειρική μελέτη πραγματικού επιστημονικού λογισμικού.

Η εργασία πραγματοποιεί μια μελέτη εξόρυξης δεδομένων σε 119 peer-reviewed repositories ερευνητικού λογισμικού από το Journal of Open Source Software (JOSS). Η επιλογή πραγματικών repositories συνδέεται με τη μεθοδολογική παράδοση των empirical και mining studies στη μηχανική λογισμικού και επιτρέπει τη μελέτη λογισμικού που έχει αναπτυχθεί και δημοσιευθεί σε πραγματικές ερευνητικές συνθήκες [10], [19]. Τα repositories χωρίζονται σε δύο χρονικές περιόδους, πριν και μετά την ευρεία εμφάνιση των AI coding assistants, ώστε να εξεταστούν πιθανές διαφοροποιήσεις στις ιδιότητες του δημοσιευμένου επιστημονικού λογισμικού.

C2. Αναπαραγώγιμη διαδικασία κατασκευής και μέτρησης του corpus.

Αναπτύσσεται μια συστηματική διαδικασία για τη συλλογή των JOSS submissions, την εφαρμογή προκαθορισμένων κριτηρίων ένταξης και αποκλεισμού, την επιλογή των repositories και την επίλυση του snapshot που αντιστοιχεί στον χρόνο αποδοχής τους. Η χρήση σταθερών κανόνων και seeded random sampling αποσκοπεί στη δυνατότητα επανάληψης της διαδικασίας και στη μείωση των περιθωρίων αυθαίρετων επιλογών κατά την κατασκευή του corpus, ακολουθώντας γενικότερες αρχές εμπειρικής αξιολόγησης και διαχείρισης απειλών εγκυρότητας στη μηχανική λογισμικού [38].

C3. Πολυδιάστατο πλαίσιο μέτρησης της ποιότητας του ερευνητικού λογισμικού.

Η αξιολόγηση δεν βασίζεται σε μία μόνο μετρική, αλλά συνδυάζει χαρακτηριστικά τεχνικού χρέους σε επίπεδο κώδικα, τεκμηρίωσης και αναπαραγωγιμότητας, μαζί με χαρακτηριστικά της δραστηριότητας ανάπτυξης που χρησιμοποιούνται ως περιγραφικό πλαίσιο και πιθανοί συγχυτικοί παράγοντες. Η επιλογή των μετρικών τεχνικού χρέους βασίζεται στην καθιερωμένη ταξινόμηση και στη δυνατότητα αυτοματοποιημένης μέτρησης που περιγράφεται από τους Li et al. [19], ενώ η έμφαση στην αναπαραγωγιμότητα και στις πρακτικές ανάπτυξης συνδέεται με τις ιδιαίτερες απαιτήσεις του επιστημονικού λογισμικού που έχουν καταγραφεί στη σχετική βιβλιογραφία [9], [10], [12].

C4. Εμπειρικά στοιχεία σχετικά με τη γεφύρωση και την απόκρυψη του χάσματος δεξιοτήτων.

Η εργασία εξετάζει αν το συνολικό μοτίβο των διαφορών μεταξύ της Pre και της Post περιόδου είναι περισσότερο συμβατό με μια γενικευμένη βελτίωση της ποιότητας του επιστημονικού λογισμικού ή με μια βελτίωση που περιορίζεται κυρίως σε χαρακτηριστικά τα οποία μπορούν να υποστηριχθούν άμεσα από αυτοματοποιημένα εργαλεία. Η διάκριση αυτή συνδέεται με τα ευρήματα προηγούμενων μελετών, οι οποίες έχουν δείξει τόσο οφέλη παραγωγικότητας από τη χρήση AI coding assistants [27], [28] όσο και προβλήματα σχετικά με την ποιότητα, την ασφάλεια και την ικανότητα αξιολόγησης του παραγόμενου κώδικα [30], [31], [33]. Παράλληλα, εξετάζεται η δυνατότητα σύνδεσης των παρατηρούμενων διαφορών με ρητές δηλώσεις χρήσης AI assistance μέσα στο Post cohort.

1.7 Δομή της Διπλωματικής Εργασίας

Το υπόλοιπο της διπλωματικής εργασίας οργανώνεται σε τέσσερα κύρια κεφάλαια.

Το Κεφάλαιο **2** παρουσιάζει τη σχετική βιβλιογραφία στις τρεις βασικές περιοχές στις οποίες τοποθετείται η παρούσα εργασία: τη μηχανική επιστημονικού λογισμικού και το χάσμα δεξιοτήτων, το τεχνικό χρέος και την ποιότητα λογισμικού, καθώς και τους βοηθούς προγραμματισμού με τεχνητή νοημοσύνη. Στο τέλος του κεφαλαίου συντίθενται οι παραπάνω ερευνητικές κατευθύνσεις και διατυπώνεται το ερευνητικό κενό που επιχειρεί να καλύψει η παρούσα μελέτη.

Το Κεφάλαιο **3** περιγράφει τη μεθοδολογία της εμπειρικής μελέτης εξόρυξης δεδομένων. Παρουσιάζονται ο ερευνητικός στόχος και τα ερευνητικά ερωτήματα, ο σχεδιασμός της μελέτης, η διαδικασία κατασκευής του corpus από repositories του Journal of Open Source Software, η διαδικασία εντοπισμού δηλώσεων χρήσης AI assistance, η υποδομή μέτρησης, οι χρησιμοποιούμενες μετρικές, το πλάνο στατιστικής ανάλυσης, η επικύρωση των εργαλείων και οι απειλές για την εγκυρότητα.

Το Κεφάλαιο **4** παρουσιάζει τα αποτελέσματα της μελέτης. Αρχικά περιγράφεται το τελικό σύνολο δεδομένων και στη συνέχεια παρουσιάζονται οι συγκρίσεις μεταξύ των Pre και Post cohorts για τις μετρικές τεχνικού χρέους, τεκμηρίωσης και αναπαραγωγιμότητας. Επιπλέον, εξετάζεται το υποσύνολο των Post repositories με ρητή δήλωση χρήσης AI assistance, πραγματοποιούνται έλεγχοι συγχυτικών παραγόντων και robustness και, τέλος, συντίθεται το συνολικό μοτίβο των ευρημάτων σε σχέση με τα ερευνητικά ερωτήματα.

Το Κεφάλαιο **5** συνοψίζει τα κύρια ευρήματα της εργασίας, παρουσιάζει τους βασικούς περιορισμούς της μελέτης και διατυπώνει προτάσεις για μελλοντική έρευνα. Τέλος, παρουσιάζεται η συνολική ερμηνεία των αποτελεσμάτων ως προς το κατά πόσο το παρατηρούμενο μοτίβο είναι περισσότερο συμβατό με τη γεφύρωση ή με την απόκρυψη του χάσματος δεξιοτήτων.

Κεφάλαιο 2ο: Σχετική Εργασία

2.1 Εισαγωγή

Η παρούσα διπλωματική εργασία βρίσκεται στη διασταύρωση τριών ερευνητικών περιοχών: της μηχανικής επιστημονικού λογισμικού, της μελέτης και διαχείρισης του τεχνικού χρέους και της πρόσφατης έρευνας σχετικά με τους βοηθούς προγραμματισμού που βασίζονται σε τεχνητή νοημοσύνη. Οι τρεις περιοχές έχουν αναπτυχθεί σε μεγάλο βαθμό ανεξάρτητα. Η πρώτη έχει τεκμηριώσει τις ιδιαίτερες δυσκολίες που αντιμετωπίζουν οι επιστήμονες κατά την ανάπτυξη λογισμικού, η δεύτερη έχει αναπτύξει τρόπους περιγραφής και μέτρησης των συνεπειών κακών πρακτικών ανάπτυξης και η τρίτη εξετάζει τον τρόπο με τον οποίο τα μεγάλα γλωσσικά μοντέλα αλλάζουν τη διαδικασία παραγωγής κώδικα. Αυτό που δεν έχει μελετηθεί επαρκώς είναι η τομή αυτών των τριών περιοχών: εάν, δηλαδή, η χρήση βοηθών ΑΙ από τους προγραμματιστές επιστημονικού λογισμικού μειώνει πραγματικά το χάσμα στις πρακτικές μηχανικής λογισμικού ή απλώς μεταβάλλει τον τρόπο με τον οποίο το χάσμα αυτό εκδηλώνεται.

2.2 Μηχανική Επιστημονικού Λογισμικού και το Χάσμα Δεξιοτήτων

Η εξάρτηση της σύγχρονης επιστήμης από το λογισμικό έχει οδηγήσει στην ανάπτυξη μιας ξεχωριστής ερευνητικής περιοχής γύρω από τη μηχανική επιστημονικού λογισμικού. Ωστόσο, η ανάπτυξη αυτού του λογισμικού διαφέρει σημαντικά από την ανάπτυξη εμπορικών συστημάτων λογισμικού. Οι επιστήμονες είναι συχνά ταυτόχρονα οι τελικοί χρήστες και οι δημιουργοί των προγραμμάτων τους και μαθαίνουν τις απαραίτητες δεξιότητες προγραμματισμού κυρίως μέσω αυτοδιδασκαλίας, συνεργασίας και άτυπης μεταφοράς γνώσης. Οι Hannay et al. [1], σε έρευνα σχετικά με τον τρόπο με τον οποίο οι επιστήμονες αναπτύσσουν και χρησιμοποιούν λογισμικό, διαπίστωσαν ότι η ανάπτυξη λογισμικού αποτελεί σημαντικό μέρος της ερευνητικής τους εργασίας, ενώ η σχετική εκπαίδευση αποκτάται σε μεγάλο βαθμό εκτός τυπικών προγραμμάτων εκπαίδευσης μηχανικής λογισμικού.

Το πρόβλημα δεν περιορίζεται στην έλλειψη τυπικής εκπαίδευσης. Οι Prabhu et al. [2] έδειξαν ότι η υπολογιστική εργασία αποτελεί σημαντικό μέρος της καθημερινής δραστηριότητας των ερευνητών, ενώ οι Nangia και Katz [3] διαπίστωσαν ότι σημαντικό ποσοστό μεταδιδασκτορικών ερευνητών χρησιμοποιεί λογισμικό που έχει αναπτύξει ο ίδιος. Παράλληλα, η εκπαίδευση σε πρακτικές ανάπτυξης λογισμικού παραμένει περιορισμένη. Το Software Carpentry έχει αναδειχθεί ως χαρακτηριστική προσπάθεια αντιμετώπισης αυτού του προβλήματος, με τον Wilson [4] να επισημαίνει ότι οι επιστήμονες συχνά αφιερώνουν σημαντικό χρόνο στην ανάπτυξη λογισμικού χωρίς να έχουν αντίστοιχη γνώση για την αξιοπιστία και την ποιότητά του.

Η συστηματική χαρτογράφηση των Arvanitou et al. [10] παρέχει την πιο ολοκληρωμένη εικόνα αυτής της κατάστασης. Εξετάζοντας 359 πρωτογενείς μελέτες, οι συγγραφείς καταγράφουν την υιοθέτηση πρακτικών μηχανικής λογισμικού στο επιστημονικό λογισμικό και δείχνουν ότι, παρόλο που οι επιστήμονες αφιερώνουν σημαντικό μέρος του χρόνου τους στην ανάπτυξη λογισμικού, η υιοθέτηση πρακτικών μηχανικής λογισμικού παραμένει περιορισμένη. Αντίστοιχα, οι Heaton και Carver [12], μέσα από συστηματική ανασκόπηση της βιβλιογραφίας, έδειξαν ότι οι πρακτικές που οι επιστήμονες θεωρούν σημαντικές, όπως η επαλήθευση και οι δοκιμές, δεν είναι απαραίτητα εκείνες που εφαρμόζουν συστηματικά.

Μια σημαντική εξήγηση για αυτή την απόσταση μεταξύ σημασίας και πρακτικής δίνεται από τον Segal [13]. Η ανάπτυξη επιστημονικού λογισμικού καθοδηγείται συχνά από το επιστημονικό πρόβλημα και όχι από τις απαιτήσεις του ίδιου του λογισμικού. Ο κώδικας εξελίσσεται μαζί με την έρευνα και αξιολογείται κυρίως με βάση το αν μπορεί να παράγει χρήσιμα επιστημονικά αποτελέσματα. Ως αποτέλεσμα, πρακτικές των οποίων το όφελος εμφανίζεται μακροπρόθεσμα, όπως η συστηματική τεκμηρίωση, οι δοκιμές και ο καθαρός σχεδιασμός, μπορούν να αναβληθούν προκειμένου να επιτευχθεί άμεσα ο ερευνητικός στόχος.

Η βιβλιογραφία δείχνει επίσης ότι το πρόβλημα αυτό μπορεί να έχει άμεσες επιπτώσεις στην αξιοπιστία της επιστήμης. Οι Hatton και Roberts [5] έδειξαν ότι διαφορετικές υλοποιήσεις των ίδιων επιστημονικών αλγορίθμων μπορούσαν να παράγουν αποτελέσματα που διέφεραν περισσότερο από την αναμενόμενη αριθμητική ανοχή. Ο Merali [6] κατέγραψε περιπτώσεις στις οποίες σφάλματα σε επιστημονικό λογισμικό επηρέασαν δημοσιευμένα ερευνητικά αποτελέσματα, ενώ οι Jorpa et al. [7] ανέδειξαν τις ανησυχίες σχετικά με την επιλογή και χρήση λογισμικού μοντελοποίησης στην επιστήμη.

Το ζήτημα συνδέεται άμεσα και με την αναπαραγωγικότητα. Η έρευνα του Baker [8] έδειξε ότι η κρίση αναπαραγωγικότητας θεωρείται σημαντικό πρόβλημα από μεγάλο μέρος της επιστημονικής κοινότητας. Ειδικά για την υπολογιστική έρευνα, οι Stodden et al. [9] εξέτασαν την πρακτική αναπαραγωγή αποτελεσμάτων από δημοσιευμένες εργασίες και διαπίστωσαν σημαντική απόσταση μεταξύ της θεωρητικής δυνατότητας αναπαραγωγής και της πραγματικής δυνατότητας εκτέλεσης των αντίστοιχων υπολογιστικών τεχνουργμάτων.

Συνεπώς, η υπάρχουσα βιβλιογραφία τεκμηριώνει με συνέπεια ένα βασικό πρόβλημα: το επιστημονικό λογισμικό είναι κρίσιμο για την παραγωγή επιστημονικής γνώσης, αλλά αναπτύσσεται συχνά από άτομα που δεν διαθέτουν την εκπαίδευση και τις πρακτικές που συνδέονται με τη συστηματική μηχανική λογισμικού [10], [12]. Η βιβλιογραφία, ωστόσο, δεν απαντά στο ερώτημα εάν νέα εργαλεία που αυτοματοποιούν μέρος της ανάπτυξης μπορούν να λειτουργήσουν ως υποκατάστατο αυτής της γνώσης.

2.3 Τεχνικό Χρέος και Ποιότητα Επιστημονικού Λογισμικού

Μία από τις σημαντικότερες έννοιες που επιτρέπουν τη μελέτη των συνεπειών ανεπαρκών πρακτικών ανάπτυξης είναι το τεχνικό χρέος (Technical Debt, TD). Η έννοια εισήχθη από τον Cunningham [15], ο οποίος χρησιμοποίησε τη μεταφορά του οικονομικού χρέους για να περιγράψει την προσωρινή αποδοχή μιας μη ιδανικής τεχνικής λύσης με αντάλλαγμα ταχύτερη ανάπτυξη. Όπως και ένα οικονομικό χρέος, η τεχνική επιλογή μπορεί να είναι χρήσιμη βραχυπρόθεσμα, αλλά δημιουργεί μελλοντικό κόστος εάν δεν αντιμετωπιστεί.

Η έννοια επεκτάθηκε από τον McConnell [16], ο οποίος διέκρινε μεταξύ χρέους που αναλαμβάνεται σκόπιμα και χρέους που δημιουργείται χωρίς πλήρη επίγνωση των συνεπειών του. Οι Kruchten et al. [17] περιέγραψαν την εξέλιξη της έννοιας από μια μεταφορά σε ένα πιο συστηματικό θεωρητικό πλαίσιο, ενώ οι Brown et al. [18] ανέδειξαν ότι το τεχνικό χρέος δεν αφορά αποκλειστικά τον πηγαίο κώδικα, αλλά μπορεί να εμφανιστεί σε διαφορετικά τεχνικά τεχνουργήματα ενός συστήματος.

Η συστηματική χαρτογράφηση των Li et al. [19] αποτελεί σημαντικό σημείο αναφοράς για τη μέτρηση του τεχνικού χρέους. Εξετάζοντας 94 πρωτογενείς μελέτες, οι συγγραφείς ταξινομούν το τεχνικό χρέος σε διαφορετικές κατηγορίες, όπως χρέος κώδικα, σχεδιασμού, αρχιτεκτονικής, δοκιμών, τεκμηρίωσης, build και υποδομής. Παράλληλα, καταγράφουν τις δραστηριότητες που

χρησιμοποιούνται για τη διαχείρισή του, όπως η αναγνώριση, η μέτρηση, η παρακολούθηση και η αποπληρωμή.

Η σημασία της ταξινόμησης αυτής για την παρούσα εργασία είναι διπλή. Πρώτον, παρέχει ένα καθιερωμένο λεξιλόγιο για την περιγραφή διαφορετικών μορφών υποβάθμισης της ποιότητας του λογισμικού. Δεύτερον, ορισμένες μορφές χρέους μπορούν να μετρηθούν αυτοματοποιημένα σε μεγάλα σύνολα repositories. Αυτό καθιστά δυνατή τη μετάβαση από τη μελέτη μεμονωμένων περιπτώσεων σε mining studies μεγάλης κλίμακας.

Το τεχνικό χρέος έχει επίσης μελετηθεί σε σχέση με τον σχεδιασμό και την αρχιτεκτονική. Οι Tom et al. [21] εξετάζουν τις αιτίες και τις συνέπειες του τεχνικού χρέους, ενώ οι Besker et al. [20] εστιάζουν ειδικότερα στο αρχιτεκτονικό τεχνικό χρέος και στη διαχείρισή του. Οι εργασίες αυτές δείχνουν ότι το χρέος δεν αποτελεί απλώς μια ιδιότητα μεμονωμένων γραμμών κώδικα, αλλά μπορεί να επηρεάζει την προσπάθεια ανάπτυξης και τη δυνατότητα μελλοντικής εξέλιξης ενός συστήματος.

Ιδιαίτερο ενδιαφέρον για την παρούσα εργασία παρουσιάζει το Self-Admitted Technical Debt (SATD), δηλαδή το χρέος που αναγνωρίζεται ρητά από τους ίδιους τους developers, συνήθως μέσω σχολίων όπως TODO ή FIXME. Σε αντίθεση με τις αυτοματοποιημένες μετρικές που προσπαθούν να εντοπίσουν προβλήματα στον κώδικα, το SATD παρέχει πληροφορία σχετικά με το εάν οι ίδιοι οι developers αναγνωρίζουν την ύπαρξη ενός προβλήματος. Η διάκριση αυτή είναι σημαντική για τη μελέτη της σχέσης μεταξύ AI assistance και skill gap, καθώς μια μείωση του μετρούμενου χρέους δεν είναι απαραίτητα ισοδύναμη με βελτίωση της ικανότητας του developer να αναγνωρίζει και να διαχειρίζεται το χρέος.

Η σύνδεση του τεχνικού χρέους με την τεχνητή νοημοσύνη έχει αρχίσει να εξετάζεται τα τελευταία χρόνια. Οι Bogner et al. [22] εντόπισαν νέες κατηγορίες χρέους σε συστήματα που βασίζονται στην AI, όπως χρέος δεδομένων, μοντέλου, διαμόρφωσης και ηθικής. Οι Alves et al. [23] παρέχουν ένα ευρύτερο πλαίσιο για την αναγνώριση και διαχείριση διαφορετικών μορφών τεχνικού χρέους.

Πιο πρόσφατα, οι Moreschini et al. [24] μελέτησαν την εξέλιξη του τεχνικού χρέους από την εποχή του DevOps έως την εποχή της παραγωγικής AI. Η εργασία αυτή είναι ιδιαίτερα σημαντική για την παρούσα μελέτη, καθώς δείχνει ότι η χρήση παραγωγικής AI δεν δημιουργεί απλώς νέες δυνατότητες ανάπτυξης, αλλά μπορεί επίσης να δημιουργήσει νέες μορφές τεχνικού χρέους. Μεταξύ άλλων, αναδεικνύονται ζητήματα διακυβέρνησης, επεξηγησιμότητας και prompts, τα οποία μπορούν να επηρεάσουν την προέλευση, την κατανόηση και τη διαχείριση του παραγόμενου κώδικα.

Ωστόσο, η υπάρχουσα βιβλιογραφία γύρω από το τεχνικό χρέος και την AI εστιάζει κυρίως στο πώς χαρακτηρίζεται το χρέος ή ποιοι νέοι τύποι χρέους μπορούν να εμφανιστούν. Δεν εξετάζει επαρκώς εάν η χρήση AI coding assistants από μη εξειδικευμένους developers οδηγεί σε συνολική μείωση του χρέους ή εάν απλώς μετακινεί το χρέος σε μορφές που είναι δυσκολότερο να παρατηρηθούν.

2.4 Βοηθοί Προγραμματισμού με Τεχνητή Νοημοσύνη

Η ανάπτυξη μεγάλων γλωσσικών μοντέλων εκπαιδευμένων σε πηγαίο κώδικα δημιούργησε μια νέα κατηγορία εργαλείων που μπορούν να παράγουν ή να συμπληρώνουν κώδικα με βάση περιγραφές σε φυσική γλώσσα. Οι Chen et al. [25] αξιολόγησαν μεγάλα γλωσσικά μοντέλα εκπαιδευμένα σε κώδικα και παρουσίασαν το HumanEval ως benchmark για την αξιολόγηση της ικανότητάς τους να παράγουν λειτουργικά προγράμματα. Η εργασία αυτή αποτέλεσε ένα από τα βασικά σημεία εκκίνησης για τη σύγχρονη έρευνα γύρω από την παραγωγή κώδικα με χρήση LLMs.

Η μετάβαση από τα ερευνητικά μοντέλα σε εργαλεία που χρησιμοποιούνται μέσα στο περιβάλλον ανάπτυξης περιγράφηκε από τους Bird et al. [26], οι οποίοι μελέτησαν τις πρώτες εμπειρίες από τη χρήση εργαλείων AI-powered pair programming. Έκτοτε, η έρευνα έχει στραφεί ιδιαίτερα στη μέτρηση της παραγωγικότητας, της ποιότητας και της ασφάλειας του κώδικα που παράγεται με βοήθεια AI.

Στο επίπεδο της παραγωγικότητας, οι Peng et al. [27] πραγματοποίησαν ελεγχόμενο πείραμα σχετικά με τη χρήση του GitHub Copilot και ανέφεραν σημαντική μείωση του χρόνου ολοκλήρωσης μιας τυποποιημένης εργασίας προγραμματισμού για τους συμμετέχοντες που είχαν πρόσβαση στον βοηθό. Το αποτέλεσμα είναι ιδιαίτερα ενδιαφέρον για το πρόβλημα που εξετάζεται στην παρούσα εργασία, επειδή τα μεγαλύτερα οφέλη παρατηρήθηκαν στους λιγότερο έμπειρους προγραμματιστές.

Παρόμοια, οι Ziegler et al. [28] μελέτησαν τη χρήση neural code completion και διαπίστωσαν ότι η αποδοχή των προτάσεων του εργαλείου συνδέεται με την αντιλαμβανόμενη παραγωγικότητα των developers. Ωστόσο, η σχέση μεταξύ αντικειμενικής και αντιλαμβανόμενης παραγωγικότητας δεν είναι πάντα ίδια. Οι Vaithilingam et al. [29] διαπίστωσαν ότι οι χρήστες μπορούσαν να προτιμούν εργαλεία παραγωγής κώδικα ακόμη και όταν δεν παρατηρούνταν αντίστοιχη βελτίωση στον αντικειμενικά μετρημένο χρόνο ολοκλήρωσης ή στην επιτυχία των εργασιών.

Η βιβλιογραφία σχετικά με την ποιότητα του παραγόμενου κώδικα παρουσιάζει περισσότερο σύνθετη εικόνα. Οι Pearce et al. [30] εξέτασαν κώδικα που παρήχθη από το GitHub Copilot σε σενάρια που σχετίζονται με την ασφάλεια και διαπίστωσαν σημαντικό ποσοστό ευάλωτων λύσεων. Οι Perry et al. [31], σε user study, βρήκαν ότι οι συμμετέχοντες που είχαν πρόσβαση σε AI assistant μπορούσαν να παράγουν λιγότερο ασφαλή κώδικα και, ταυτόχρονα, να έχουν μεγαλύτερη βεβαιότητα ότι ο κώδικάς τους ήταν ασφαλής. Το δεύτερο εύρημα είναι ιδιαίτερα σημαντικό για την παρούσα εργασία, καθώς υποδεικνύει ότι το πρόβλημα δεν αφορά μόνο την ποιότητα του παραγόμενου artefact αλλά και την ικανότητα του developer να αξιολογεί την ποιότητά του.

Τα αποτελέσματα δεν είναι, ωστόσο, ομοιόμορφα. Οι Sandoval et al. [32] βρήκαν διαφορετική εικόνα σχετικά με τις επιπτώσεις στην ασφάλεια, γεγονός που υποδεικνύει ότι η επίδραση των AI assistants εξαρτάται από το πλαίσιο, την εργασία και τον πληθυσμό των χρηστών. Οι Dakhel et al. [33] υποστηρίζουν αντίστοιχα ότι το Copilot μπορεί να λειτουργήσει ως πλεονέκτημα για έμπειρους προγραμματιστές αλλά ως μειονέκτημα για αρχάριους, καθώς η αξιοποίηση μιας παραγόμενης λύσης προϋποθέτει την ικανότητα αξιολόγησής της.

Άλλες μελέτες έχουν εξετάσει τη σταθερότητα και την ποιότητα των παραγόμενων λύσεων. Οι Mastropaolo et al. [34] έδειξαν ότι διαφορετικές διατυπώσεις ουσιαστικά ισοδύναμων prompts μπορούν να οδηγήσουν σε διαφορετικό κώδικα, ενώ οι Yetistiren et al. [35] εντόπισαν διαφορές στην ποιότητα μεταξύ διαφορετικών εργαλείων παραγωγής κώδικα. Οι Barke et al. [36] μελέτησαν τον τρόπο με τον οποίο οι developers αλληλεπιδρούν με τα code-generating models και διέκριναν μεταξύ διαφορετικών τρόπων χρήσης, συμπεριλαμβανομένης της εξερεύνησης και της γρήγορης επιτάχυνσης της ανάπτυξης. Στη δεύτερη περίπτωση, οι προτάσεις μπορούν να ενσωματώνονται με περιορισμένο έλεγχο από τον developer.

Τα ευρήματα αυτά δημιουργούν μια σημαντική αντίφαση. Από τη μία πλευρά, οι AI assistants μειώνουν το κόστος παραγωγής κώδικα και μπορούν να προσφέρουν σημαντικά οφέλη παραγωγικότητας, ιδιαίτερα σε λιγότερο έμπειρους developers [27], [28]. Από την άλλη, η ικανότητα αξιολόγησης του παραγόμενου κώδικα δεν αυτοματοποιείται στον ίδιο βαθμό και η βιβλιογραφία έχει καταγράψει προβλήματα ασφάλειας, ποιότητας και υπερβολικής εμπιστοσύνης [30], [31], [33].

Αυτό είναι ιδιαίτερα σημαντικό στην περίπτωση του επιστημονικού λογισμικού. Ένας επιστήμονας που δεν έχει αναπτύξει ισχυρές δεξιότητες μηχανικής λογισμικού μπορεί πλέον να παράγει κώδικα που μοιάζει περισσότερο με κώδικα έμπειρου developer, χωρίς όμως να έχει αποκτήσει απαραίτητα την ικανότητα να αξιολογεί τις επιστημονικές και τεχνικές παραδοχές του. Επομένως, η επιφανειακή βελτίωση του κώδικα δεν συνεπάγεται κατ' ανάγκη αντίστοιχη βελτίωση της ποιότητας της έρευνας.

2.5 AI Assistance, Τεχνικό Χρέος και το Ερευνητικό Κενό

Οι τρεις προηγούμενες ερευνητικές περιοχές συγκλίνουν σε ένα κοινό πρόβλημα, το οποίο μέχρι σήμερα έχει μελετηθεί μόνο αποσπασματικά. Η βιβλιογραφία για το scientific software engineering έχει τεκμηριώσει την ύπαρξη ενός χάσματος μεταξύ της σημασίας του λογισμικού για την επιστήμη και του επιπέδου υιοθέτησης πρακτικών μηχανικής λογισμικού [10], [12]. Η βιβλιογραφία για το technical debt παρέχει ένα ώριμο πλαίσιο για την περιγραφή και μέτρηση ορισμένων από τις συνέπειες αυτού του χάσματος [19]. Παράλληλα, η βιβλιογραφία για τους AI coding assistants έχει δείξει ότι η παραγωγή κώδικα μπορεί να επιταχυνθεί σημαντικά, αλλά έχει επίσης αναδείξει προβλήματα στην ασφάλεια, στην ποιότητα και στην αξιολόγηση του παραγόμενου κώδικα [27], [30], [31], [33].

Ωστόσο, οι τρεις γραμμές έρευνας δεν έχουν συνδεθεί επαρκώς σε εμπειρικό επίπεδο. Οι μελέτες σχετικά με την παραγωγικότητα των AI assistants εξετάζουν κυρίως γενικές εργασίες προγραμματισμού και μετρούν τον χρόνο ολοκλήρωσης, την αποδοχή προτάσεων ή την ποσότητα κώδικα [27]–[29]. Οι μελέτες ποιότητας εστιάζουν κυρίως σε χαρακτηριστικά όπως η ασφάλεια και η λειτουργικότητα [30]–[35]. Αυτές οι μετρικές είναι χρήσιμες, αλλά δεν απαντούν άμεσα στο ερώτημα που αφορά το επιστημονικό λογισμικό: εάν το παραγόμενο λογισμικό είναι περισσότερο αναπαραγώγιμο, καλύτερα τεκμηριωμένο και λιγότερο επιβαρυνόμενο με τεχνικό χρέος.

Αντίστοιχα, η βιβλιογραφία του technical debt έχει αρχίσει να εξετάζει την επίδραση της AI και να αναγνωρίζει νέες μορφές χρέους [22], [24], αλλά δεν παρέχει επαρκή εμπειρικά στοιχεία σχετικά με το εάν η χρήση AI assistants από developers με περιορισμένη εκπαίδευση στη μηχανική λογισμικού οδηγεί σε πραγματική μείωση του χρέους. Ιδιαίτερα απουσιάζει μια μεγάλης κλίμακας μελέτη σε πραγματικά, δημοσιευμένα scientific software repositories.

Ένα δεύτερο κενό αφορά τη διάκριση μεταξύ της ποιότητας του artefact και της ικανότητας του δημιουργού να κατανοεί και να αξιολογεί το artefact. Η μελέτη των Perry et al. [31] παρέχει ένα σημαντικό παράδειγμα αυτού του προβλήματος στην περιοχή της ασφάλειας: η πρόσβαση σε AI assistance μπορεί να αλλάξει όχι μόνο τον κώδικα που παράγεται αλλά και την αυτοπεποίθηση του developer σχετικά με την ορθότητά του. Στο επιστημονικό λογισμικό, η αντίστοιχη διάκριση είναι ακόμη πιο κρίσιμη, επειδή ένα πρόγραμμα μπορεί να είναι συντακτικά σωστό, καλοδομημένο και αναπαραγώγιμο, αλλά να υλοποιεί λανθασμένα έναν επιστημονικό υπολογισμό.

Ένα τρίτο κενό αφορά το επίπεδο στο οποίο έχει μελετηθεί η χρήση των assistants. Η πλειονότητα των σχετικών μελετών βασίζεται σε controlled experiments, user studies ή benchmarks [27]–[36]. Αυτές οι προσεγγίσεις προσφέρουν υψηλότερο πειραματικό έλεγχο, αλλά συχνά εξετάζουν μικρές, χρονικά περιορισμένες εργασίες. Δεν είναι επομένως αυτονόητο ότι τα αποτελέσματα μεταφέρονται στο πραγματικό scientific software, το οποίο μπορεί να αναπτύσσεται για μήνες ή χρόνια, να περνά από peer review και τελικά να δημοσιεύεται ως μέρος μιας επιστημονικής εργασίας.

Η παρούσα διπλωματική εργασία επιχειρεί να καλύψει αυτό το κενό με mining πραγματικών repositories ερευνητικού λογισμικού. Αντί να εξετάζει μόνο το εάν ένας developer ολοκληρώνει

γρηγορότερα μια συγκεκριμένη εργασία με τη βοήθεια ενός assistant, εξετάζει εάν οι ιδιότητες του λογισμικού που καταλήγει να δημοσιεύεται μεταβάλλονται μεταξύ της περιόδου πριν και μετά την ευρεία εμφάνιση των AI coding assistants. Η προσέγγιση αυτή επιτρέπει τη μελέτη του τεχνικού χρέους, της τεκμηρίωσης, της αναπαραγωγιμότητας και των χαρακτηριστικών ανάπτυξης σε πραγματικά software artefacts.

Η επιλογή του technical debt ως βασικού πλαισίου βασίζεται στη δυνατότητα σύνδεσης της νέας μελέτης με την καθιερωμένη βιβλιογραφία των Li et al. [19]. Παράλληλα, η μελέτη επεκτείνει το πλαίσιο με μετρικές που σχετίζονται περισσότερο άμεσα με το επιστημονικό λογισμικό, όπως η αναπαραγωγιμότητα και η τεκμηρίωση. Ιδιαίτερη θέση έχει το Self-Admitted Technical Debt, καθώς επιτρέπει μια διαφορετική μορφή παρατήρησης: όχι μόνο πόσο χρέος ανιχνεύεται στον κώδικα, αλλά και κατά πόσο οι developers αναγνωρίζουν οι ίδιοι την ύπαρξή του.

Η βιβλιογραφία επομένως επιτρέπει τη διατύπωση δύο ανταγωνιστικών ερμηνειών. Σύμφωνα με την πρώτη, οι AI assistants μπορούν να λειτουργήσουν ως μηχανισμός γεφύρωσης του skill gap. Η αυτοματοποίηση συνηθισμένων πρακτικών, η παραγωγή boilerplate κώδικα, η δημιουργία τεκμηρίωσης και η παροχή προτεινόμενων λύσεων μπορούν να μεταφέρουν πρακτικές μηχανικής λογισμικού σε developers που δεν τις έχουν διδαχθεί τυπικά. Η υπόθεση αυτή είναι συμβατή με τα ευρήματα σχετικά με την παραγωγικότητα και τα μεγαλύτερα οφέλη για λιγότερο έμπειρους developers [27].

Σύμφωνα με τη δεύτερη ερμηνεία, οι assistants μπορεί να αποκρύπτουν αντί να εξαλείφουν το skill gap. Σε αυτή την περίπτωση, ο developer μπορεί να παράγει καλύτερα εμφανισιακά ή δομικά τεχνουργήματα χωρίς να αποκτά αντίστοιχη ικανότητα αξιολόγησής τους. Το τεχνικό χρέος μπορεί να μετατοπίζεται σε μορφές που είναι δυσκολότερο να ανιχνευθούν, ενώ προβλήματα που απαιτούν επιστημονική ή τεχνική κρίση μπορεί να παραμένουν. Τα ευρήματα σχετικά με την ασφάλεια και την υπερβολική αυτοπεποίθηση των χρηστών [30], [31], καθώς και η εξάρτηση των αποτελεσμάτων από την εμπειρία του developer [33], είναι συμβατά με αυτή την ερμηνεία.

Οι δύο ερμηνείες δεν είναι απαραίτητα αμοιβαία αποκλειόμενες. Ένας AI assistant μπορεί να βελτιώνει ορισμένες πρακτικές ανάπτυξης και ταυτόχρονα να αφήνει ανεπηρέαστες άλλες δεξιότητες που απαιτούν ανθρώπινη κρίση. Για τον λόγο αυτό, μια αξιολόγηση που βασίζεται αποκλειστικά στην παραγωγικότητα ή σε μία μόνο μετρική ποιότητας δεν επαρκεί.

Το ερευνητικό κενό της παρούσας εργασίας βρίσκεται επομένως στην εμπειρική μελέτη αυτής της σχέσης σε πραγματικό scientific software. Ειδικότερα, απουσιάζει μια μελέτη που να συνδυάζει: (α) το πλαίσιο του scientific software engineering και του skill gap [10], (β) την ταξινόμηση και μέτρηση του technical debt [19], (γ) τις πρόσφατες εξελίξεις γύρω από την AI-assisted development [24]–[36] και (δ) mining πραγματικών, peer-reviewed scientific software repositories.

Η παρούσα εργασία τοποθετείται ακριβώς σε αυτό το κενό. Με τη σύγκριση repositories πριν και μετά την εμφάνιση των AI coding assistants και με ένα υποσύνολο repositories στα οποία υπάρχει ρητό disclosure χρήσης AI assistance, εξετάζει εάν η αλλαγή που παρατηρείται στα software artefacts είναι συμβατή περισσότερο με τη γεφύρωση ή με την απόκρυψη του skill gap. Η μεθοδολογία και το πλαίσιο μέτρησης που ακολουθούν σχεδιάζονται έτσι ώστε η διάκριση αυτή να βασίζεται όχι σε μία μόνο μετρική, αλλά σε ένα προκαθορισμένο μοτίβο αλλαγών στο τεχνικό χρέος, την τεκμηρίωση, την αναπαραγωγιμότητα και τα χαρακτηριστικά ανάπτυξης.

Κεφάλαιο 3ο: Μεθοδολογία

3.1 Επισκόπηση της Μελέτης

Η βιβλιογραφία που εξετάστηκε στην Ενότητα 2.0 θέτει τα δύο βασικά σημεία από τα οποία ξεκινά η παρούσα διπλωματική εργασία.

Πρώτον, το επιστημονικό λογισμικό γράφεται σε μεγάλο βαθμό από άτομα χωρίς εκπαίδευση στη μηχανική λογισμικού, και οι συνέπειες αυτού του κενού έχουν καταγραφεί και μπορούν να μετρηθούν [10, 1, 12]. Δεύτερον, η κοινότητα της μηχανικής λογισμικού διαθέτει ένα ώριμο πλαίσιο, το τεχνικό χρέος και τη διαχείρισή του [19], για την περιγραφή και την ποσοτικοποίηση αυτών των συνεπειών. Το πλαίσιο αυτό έχει πρόσφατα επεκταθεί και στους τύπους χρέους που εισάγει η ίδια η ανάπτυξη με τη βοήθεια AI [24].

Μεταξύ του τέλους του 2022 και των μέσων του 2023, οι συνθήκες κάτω από τις οποίες αναπτύσσεται το επιστημονικό λογισμικό άλλαξαν. Οι βοηθοί προγραμματισμού με AI μετακινήθηκαν από ερευνητικά πρωτότυπα σε βασικά στοιχεία του περιβάλλοντος ανάπτυξης [25, 26], και η υιοθέτησή τους ήταν ταχύτερη ακριβώς στον πληθυσμό που αφορά αυτή η διπλωματική εργασία: developers για τους οποίους ο κώδικας αποτελεί μέσο και όχι αντικείμενο εξειδίκευσης [27, 33]. Αν αυτά τα εργαλεία γεφυρώνουν το κενό δεξιοτήτων, τότε το ερευνητικό λογισμικό που παράχθηκε μετά την εμφάνισή τους θα πρέπει να παρουσιάζει μετρήσιμα λιγότερο τεχνικό χρέος, καλύτερη τεκμηρίωση και ισχυρότερη αναπαραγωγικότητα σε σχέση με το λογισμικό που παράχθηκε πριν από αυτά. Αν, αντίθετα, απλώς αποκρύπτουν το κενό, τότε τα παρατηρήσιμα δεδομένα θα πρέπει να δείχνουν κάτι πιο σύνθετο: οι επιφανειακές μετρικές να βελτιώνονται, ενώ οι ιδιότητες που απαιτούν κρίση, η πειθαρχία στην αναπαραγωγικότητα, η ειλικρινής τεκμηρίωση και το χρέος που αναγνωρίζεται αντί να συσσωρεύεται, παραμένουν στάσιμες ή υποβαθμίζονται.

Η παρούσα ενότητα περιγράφει μια εμπειρική μελέτη σχεδιασμένη ώστε να εξετάσει άμεσα αυτό το ιστορικό. Ο σχεδιασμός είναι μια μελέτη εξόρυξης repositories δύο cohorts: ένα corpus από repositories ερευνητικού λογισμικού που έχουν αξιολογηθεί από κριτές και έγιναν αποδεκτά για δημοσίευση πριν από την εποχή των AI assistants συγκρίνεται, με τα ίδια εργαλεία μέτρησης, με ένα corpus που έγινε αποδεκτό μετά από αυτήν. Ένα υποσύνολο attribution, repositories που δηλώνουν ρητά τη χρήση AI assistance σε σύγκριση με σύγχρονα repositories που δεν το δηλώνουν, παρέχει τη σύνδεση μεταξύ της παρατηρούμενης διαφοράς μεταξύ των εποχών και του ίδιου του εργαλείου. Η μελέτη ακολουθεί έτσι τη μεθοδολογική παράδοση των mining και mapping studies που αποτελούν τη βάση της [19, 10, 24], αντί για την παράδοση των controlled experiments: θυσιάζει τον πειραματικό έλεγχο για ρεαλισμό και κλίμακα, μετρώντας το λογισμικό που πραγματικά δημοσίευσαν οι επιστήμονες και όχι το λογισμικό που παράγουν υπό παρατήρηση.

3.2 Ερευνητικός Στόχος και Ερωτήματα

Ακολουθώντας τη διατύπωση Goal-Question-Metric [37], που υιοθετείται και από τις δύο βασικές mapping studies [19, 10], ο στόχος της παρούσας μελέτης είναι:

Ανάλυση repositories ερευνητικού λογισμικού που έχουν αξιολογηθεί από κριτές, με σκοπό τη σύγκριση μεταξύ των εποχών πριν και μετά την εμφάνιση των AI assistants ως προς το τεχνικό χρέος, την τεκμηρίωση, την αναπαραγωγικότητα και τα χαρακτηριστικά της δραστηριότητας ανάπτυξης, από την οπτική της κοινότητας του research software, στο πλαίσιο open-source επιστημονικού λογισμικού που έχει γίνει αποδεκτό από ένα peer-reviewed software journal.

Τέσσερα ερευνητικά ερωτήματα συγκεκριμενοποιούν τον παραπάνω στόχο.

3.2.1.1 RQ1, Τεχνικό χρέος σε επίπεδο κώδικα μεταξύ των εποχών

Πώς διαφέρουν τα χαρακτηριστικά του τεχνικού χρέους σε επίπεδο κώδικα του ερευνητικού λογισμικού, debt ratio, code smells, complexity, duplication και self-admitted technical debt, μεταξύ repositories που έγιναν αποδεκτά πριν και μετά την εμφάνιση των AI coding assistants;

Αιτιολόγηση. Αυτοί είναι οι τύποι χρέους για τους οποίους οι Li et al. [19] τεκμηριώνουν ώριμη υποστήριξη μέτρησης, γεγονός που τους καθιστά συγκρίσιμους σε εκατοντάδες repositories χωρίς ανθρώπινη κρίση. Το Self-Admitted Technical Debt (SATD), δηλαδή το χρέος που αναγνωρίζεται από τους developers μέσα σε σχόλια, περιλαμβάνεται επειδή διαχωρίζει τις δύο υποθέσεις με τρόπο που οι αυτοματοποιημένες μετρικές δεν μπορούν: ένας developer που γράφει TODO γνωρίζει ότι υπάρχει χρέος, επομένως μια μείωση του SATD μαζί με σταθερό ή αυξανόμενο μετρούμενο χρέος αποτελεί ένδειξη απόκρυψης και όχι βελτίωσης.

3.2.1.2 RQ2, Ποιότητα έρευνας μεταξύ των εποχών

Πώς διαφέρουν τα χαρακτηριστικά της αναπαραγωγιμότητας και της τεκμηρίωσης του ερευνητικού λογισμικού μεταξύ των δύο εποχών;

Αιτιολόγηση. Η αναπαραγωγιμότητα είναι η ιδιότητα που κάνει το ερευνητικό λογισμικό κατάλληλο για έρευνα [9, 8], ενώ η τεκμηρίωση είναι το μέσο μέσω του οποίου ο επιστημονικός κώδικας μεταφέρει τις παραδοχές του. Καμία από τις δύο δεν μετριέται από τα συμβατικά εργαλεία αξιολόγησης κώδικα, γεγονός που εξηγεί γιατί η βιβλιογραφία σχετικά με την AI assistance, η οποία επικεντρώνεται στην παραγωγικότητα [27] και την ασφάλεια [30, 31], τις έχει σε μεγάλο βαθμό αφήσει ανεξέταστες.

3.2.1.3 RQ3, Attribution μέσα στην post εποχή

Μέσα στο post-AI cohort, πώς διαφέρουν, στις ίδιες μετρικές, τα repositories που δηλώνουν ρητά τη χρήση AI assistance από τα σύγχρονα repositories που δεν τη δηλώνουν;

Αιτιολόγηση. Τα RQ1 και RQ2 συγκρίνουν εποχές, και οι εποχές περιλαμβάνουν όλα όσα άλλαξαν με την πάροδο του χρόνου. Το RQ3 διατηρεί τον χρόνο περίπου σταθερό: τα repositories που δηλώνουν χρήση και αυτά που δεν δηλώνουν αντιμετώπισαν το ίδιο Python ecosystem, το ίδιο επίπεδο αξιολόγησης από το journal και τις ίδιες community norms. Επομένως, μια διαφορά μεταξύ τους είναι πολύ πιο δύσκολο να αποδοθεί σε background drift. Είναι το πλησιέστερο σημείο στο οποίο μπορεί να φτάσει αυτός ο observational σχεδιασμός στην απομόνωση της επίδρασης του εργαλείου.

3.2.1.4 RQ4, Γεφύρωση ή απόκρυψη

Συνολικά, δείχνει το μοτίβο των διαφορών στα RQ1–RQ3 ότι η AI assistance κλείνει το κενό δεξιοτήτων που εντοπίστηκε από τους Arvanitou et al. [10] ή ότι το αποκρύπτει;

Αιτιολόγηση. Οι δύο υποθέσεις που παρουσιάστηκαν στην Ενότητα 1.3 προβλέπουν ορισμένες κοινές βελτιώσεις στις επιφανειακές μετρικές, αλλά διαφέρουν ως προς το μοτίβο: η γεφύρωση προβλέπει βελτίωση σε όλα τα tiers με κοινή κατεύθυνση, ενώ η απόκρυψη προβλέπει βελτίωση κυρίως στις μετρικές που βελτιστοποιούν άμεσα οι assistants (συμμόρφωση στο style, boilerplate documentation), με στασιμότητα ή υποβάθμιση στις ιδιότητες που απαιτούν βαθύτερη ανθρώπινη κρίση και γνώσεις μηχανικής λογισμικού (πειθαρχία στην αναπαραγωγιμότητα, ειλικρινής αναγνώριση χρέους). Το RQ4 απαντάται από το συνολικό μοτίβο και όχι από ένα μόνο test· η Ενότητα 3.8.2 καθορίζει εκ των προτέρων την υπογραφή που προβλέπει κάθε υπόθεση.

3.3 Σχεδιασμός της Μελέτης

3.3.1 Τύπος Σχεδιασμού

Η μελέτη είναι μια cross-sectional, observational σύγκριση δύο cohorts με ένα nested attribution subsample. Η μονάδα ανάλυσης είναι ένα repository σε ένα σταθερό ιστορικό snapshot, την κατάσταση του τη στιγμή της αποδοχής από το journal, και κάθε repository μετريέται ακριβώς μία φορά, με τα ίδια αυτοματοποιημένα εργαλεία, χωρίς να είναι γνωστό το cohort κατά τη μέτρηση.

Τυπικά, δύο cohorts, Pre και Post, προέρχονται από το ίδιο sampling frame με τα ίδια κριτήρια ένταξης και διαφέρουν μόνο ως προς την ημερομηνία αποδοχής. Μέσα στο Post, η διαδικασία attribution της Ενότητας 3.5 χωρίζει τα repositories σε Post-Disclosed και Post-Undisclosed. Τα RQ1 και RQ2 συγκρίνουν Pre με Post, ενώ το RQ3 συγκρίνει τα δύο υποσύνολα μέσα στο Post.

3.3.2 Γιατί Mining αντί για Experiment

Ένα controlled experiment, οι ίδιοι developers να ολοκληρώνουν tasks με και χωρίς assistant, θα εντόπιζε πιο καθαρά την αιτιώδη επίδραση της assistance, και μια προηγούμενη εκδοχή της εργασίας είχε σχεδιαστεί ακριβώς ως τέτοια μελέτη. Τρεις παράγοντες δικαιολογούν τον observational σχεδιασμό που υιοθετείται εδώ.

Ecological validity. Ένα πείραμα μετρά τι παράγουν οι developers σε sessions διάρκειας εξήντα λεπτών, σε συνθετικά tasks και υπό παρατήρηση. Η παρούσα μελέτη μετρά τι πραγματικά δημοσίευσαν οι επιστήμονες: λογισμικό που αναπτύχθηκε για μήνες, υποβλήθηκε σε peer review και αρχειοθετήθηκε ως το artefact πίσω από ένα paper. Για μια διπλωματική εργασία της οποίας το ερώτημα είναι αν η έρευνα βελτιώθηκε, το δημοσιευμένο record είναι ο πληθυσμός ενδιαφέροντος και όχι ένα proxy για αυτόν.

Scale and feasibility. Ο πληθυσμός ενδιαφέροντος, scientific developers χωρίς εκπαίδευση στη μηχανική λογισμικού, είναι μικρός, κατανεμημένος και απαιτεί σημαντικό κόστος για στρατολόγηση σε multi-hour sessions που θα χρειαζόταν ένα adequately powered experiment. Ο mining σχεδιασμός αντικαθιστά μια ντουζίνα συμμετέχοντες με εκατοντάδες repositories, και ο βασικός περιορισμός ισχύος είναι ο υπολογισμός και όχι η προθυμία συμμετοχής.

Methodological continuity. Οι τρεις μελέτες που αποτελούν τη βάση της παρούσας εργασίας [19, 10, 24] είναι όλες secondary ή mining studies. Τα εργαλεία, οι taxonomies και τα validity frameworks τους [38] μεταφέρονται άμεσα, ενώ τα αποτελέσματα που εκφράζονται με τη δική τους ορολογία είναι συγκρίσιμα με τη βιβλιογραφία που οργανώνουν.

Το τίμημα είναι η αιτιώδης συμπερασματολογία, και ο σχεδιασμός το αναγνωρίζει ανοιχτά: μια διαφορά μεταξύ εποχών είναι συμβατή με την εξήγηση της AI assistance, αλλά από μόνη της δεν μπορεί να την αποδείξει. Τρία χαρακτηριστικά περιορίζουν, αλλά δεν εξαλείφουν, το κενό. Το sampling frame διατηρεί περίπου σταθερό το επίπεδο αξιολόγησης μεταξύ των εποχών. Οι measured covariates επιτρέπουν την εξέταση του background drift αντί να θεωρείται ότι απουσιάζει. Και το attribution subsample (RQ3) παρέχει μια σύγκριση μέσα στην ίδια εποχή, την οποία το background drift δεν μπορεί να εξηγήσει. Η Ενότητα 4.6 επιστρέφει σε ό,τι παραμένει.

3.3.3 Blinding και Ordering

Όλα τα εργαλεία είναι πλήρως αυτοματοποιημένα και δεν λαμβάνουν πληροφορίες για το cohort: ένα repository μετريέται μόνο από τα αρχεία και το ιστορικό του. Η σειρά μέτρησης τυχαιοποιείται μεταξύ των cohorts, ώστε οποιοδήποτε drift στο περιβάλλον μέτρησης (tool versions,

machine load) να μην μπορεί να ευθυγραμμιστεί με το όριο μεταξύ των cohorts. Το ένα manual βήμα της μελέτης, η αξιολόγηση οριακών disclosure matches, πραγματοποιείται πριν εξεταστεί οποιαδήποτε metric για το συγκεκριμένο repository.

3.4 Κατασκευή του Corpus

3.4.1 Sampling Frame

Το sampling frame είναι το σύνολο των software submissions που έγιναν αποδεκτές από το Journal of Open Source Software (JOSS). Το JOSS επιλέγεται αντί για μια γενική αναζήτηση repositories για τέσσερις ιδιότητες που μια γενική mining διαδικασία δεν μπορεί να προσφέρει.

Population match. Το JOSS δημοσιεύει research software: κάθε submission είναι ένα software artefact που αναπτύχθηκε για να υποστηρίξει επιστημονική εργασία, κυρίως από ακαδημαϊκούς συγγραφείς. Το frame επομένως δειγματοληπτεί τον πληθυσμό της mapping study των Arvanitou et al. [10], scientific software developers, και όχι τον γενικό open-source πληθυσμό.

Quality floor και σταθερότητα του. Κάθε accepted submission έχει περάσει από την ίδια δημόσια, checklist-driven αξιολόγηση, installability, documentation, automated tests και community guidelines, η οποία εφαρμόζεται συνεχώς και στις δύο εποχές. Αυτό είναι σημαντικό για δύο λόγους: απομακρύνει abandoned και toy repositories, ενώ επειδή το checklist άλλαξε ελάχιστα κατά τη διάρκεια του study window, διατηρεί περίπου σταθερό το επίπεδο αποδοχής. Αυτή είναι η ιδιότητα που κάνει μια cross-era σύγκριση ουσιαστικά εφικτή..

Timestamping. Η αποδοχή είναι ένα χρονολογημένο editorial event, παρέχοντας σε κάθε repository έναν σαφή ορισμό εποχής και ένα τεκμηριωμένο snapshot moment, την κατάσταση του λογισμικού που πραγματικά αποδέχθηκαν οι reviewers.

Archival. Το JOSS απαιτεί μια archived έκδοση του λογισμικού κατά την αποδοχή (συνήθως ένα Zenodo deposit ή ένα release tag), γεγονός που καθιστά το snapshot ανακτήσιμο ακόμη και όταν το ιστορικό του repository έχει τροποποιηθεί αργότερα.

3.4.2 Ορισμοί των Εποχών

Δύο αποφάσεις σχετικά με τα όρια χρειάζονται αιτιολόγηση. Το transition window εξαιρείται επειδή ένα repository που έγινε αποδεκτό στις αρχές του 2023 αναπτύχθηκε σε σημαντικό βαθμό το 2022, υπό συνθήκες που δεν ανήκουν καθαρά σε καμία από τις δύο εποχές. Η συμπερίληψή του θα θόλωνε ακριβώς την αντίθεση που η μελέτη προσπαθεί να μετρήσει. Επίσης, το Pre window ξεκινά το 2019 αντί για νωρίτερα, ώστε τα cohorts να έχουν συγκρίσιμο χρονικό εύρος και το Pre cohort να μην κυριαρχείται από ένα παλαιότερο Python ecosystem (pre-3.6 idioms, pre-pyproject packaging), του οποίου οι διαφορές από το παρόν δεν σχετίζονται με την AI.

Μια σημαντική λεπτομέρεια της ανάθεσης πρέπει να αναφερθεί: η αποδοχή καθυστερεί σε σχέση με την ανάπτυξη. Ένα repository που έγινε αποδεκτό στα μέσα του 2023 μπορεί να περιέχει ιστορικό ανάπτυξης πολλών ετών πριν από την AI εποχή. Η ανάθεση εποχής με βάση την ημερομηνία αποδοχής είναι επομένως συντηρητική, ωθεί το Post cohort προς την κατάσταση του Pre και μειώνει οποιαδήποτε πραγματική διαφορά. Η covariate του repository age επιτρέπει την εξέταση αυτής της μείωσης, ενώ το attribution subsample δεν επηρεάζεται από αυτό, καθώς το disclosure αποτελεί ένδειξη για το πώς αναπτύχθηκε πραγματικά το λογισμικό.

Πίνακας 3.1: Παράθυρα των Cohorts

Cohort	Acceptance window	Αιτιολόγηση
Pre	2019-01-01 – 2022-10-31	Τελειώνει πριν από τη δημόσια κυκλοφορία του ChatGPT
(excluded)	2022-11-01 – 2023-06-30	Transition window: η υιοθέτηση ήταν σε εξέλιξη αλλά μερική· το λογισμικό που έγινε αποδεκτό εδώ αναπτύχθηκε υπό μικτές συνθήκες
Post	2023-07-01 – 2025-12-31	Οι assistants έχουν πλέον καθιερωθεί ως mainstream εργαλεία ανάπτυξης

3.4.3 Κριτήρια Ένταξης και Αποκλεισμού

Ένα candidate repository εισέρχεται στο corpus εάν ικανοποιεί όλα τα κριτήρια ένταξης και κανένα από τα κριτήρια αποκλεισμού.

I1. Έχει γίνει αποδεκτό από το JOSS μέσα σε ένα από τα δύο cohort windows.

I2. Η Python είναι η κύρια γλώσσα υλοποίησης ($\geq 70\%$ των source lines), καθώς τα static instruments του pipeline είναι Python analysers.

I3. Περιέχει μεταξύ 500 και 50.000 logical lines Python: το κατώτερο όριο αποκλείει trivial wrappers, ενώ το ανώτερο όριο αποκλείει monorepos των οποίων η μέτρηση θα κυριαρχούνταν από vendored code.

I4. Το repository είναι δημόσια clonable και το acceptance-time snapshot μπορεί να επιλυθεί.

E1. Forks και mirrors άλλων μελών του corpus.

E2. Repositories στα οποία το measured content είναι κατά πλειοψηφία auto-generated ή vendored, όπως εντοπίζεται από συμβατικά markers ([vendor/](#), [third_party/](#), generated-file headers).

E3. Repositories που δεν ολοκληρώνουν το measurement pipeline μετά τη documented retry procedure. Αυτά καταγράφονται μαζί με τους λόγους και αναφέρονται, καθώς η σιωπηλή απώλεια repositories αποτελεί από μόνη της απειλή για την εγκυρότητα.

Και τα δύο cohorts περιορίζονται σε 60 repositories. Όταν περισσότερα candidates πληρούν τα κριτήρια, το cohort επιλέγεται μέσω seeded random sampling, ώστε το corpus να μπορεί να αναπαραχθεί από το δημοσιευμένο seed. Κάθε στάδιο filtering καταγράφεται και το κεφάλαιο αποτελεσμάτων παρουσιάζει την πλήρη ροή attrition από τα candidates έως τα measured repositories, ακολουθώντας τη συμβατική αναφορά των βασικών mapping studies [19, 10].

3.4.4 Snapshot Resolution

Κάθε repository μετρίεται στο acceptance commit του, το οποίο επιλύεται με μια σταθερή σειρά προτεραιότητας:

1. Το archived version tag που καταγράφεται στα JOSS metadata.
2. Εάν αυτό αποτύχει, το release tag που βρίσκεται πλησιέστερα στην ημερομηνία αποδοχής.

3. Εάν και αυτό αποτύχει, το τελευταίο commit του default branch πριν από την ημερομηνία αποδοχής.

Ο κανόνας που ενεργοποιήθηκε καταγράφεται για κάθε repository και η κατανομή του αναφέρεται, καθώς μια συστηματική διαφορά στη δυνατότητα επίλυσης των snapshots μεταξύ των εποχών, για παράδειγμα, καλύτερη συμμόρφωση με archival στις πιο πρόσφατες χρονιές, θα αποτελούσε από μόνη της ένα measurement artefact που αξίζει να αναφερθεί.

3.4.5 Αναπαραγωγικότητα της Κατασκευής του Corpus

Η κατασκευή του corpus σχεδιάστηκε ώστε να είναι όσο το δυνατόν περισσότερο αναπαραγώγιμη και να περιορίζει τις υποκειμενικές επιλογές κατά τη διαδικασία δειγματοληψίας. Για τον λόγο αυτό, η επιλογή των repositories δεν πραγματοποιήθηκε χειροκίνητα με βάση την κρίση του ερευνητή, αλλά μέσω μιας προκαθορισμένης και deterministic διαδικασίας, η οποία συνδυάζει σταθερά κριτήρια ένταξης και αποκλεισμού, seeded random sampling και πλήρη καταγραφή των ενδιάμεσων σταδίων. Η προσέγγιση αυτή είναι ιδιαίτερα σημαντική σε μία observational mining study, καθώς ακόμη και μικρές μη τεκμηριωμένες αποφάσεις κατά την επιλογή των repositories μπορούν να μεταβάλουν τη σύνθεση των cohorts και, κατά συνέπεια, τα αποτελέσματα των συγκρίσεων.

Η διαδικασία ξεκινά από το σύνολο των υποψηφίων JOSS submissions που εμπίπτουν στα δύο προκαθορισμένα χρονικά παράθυρα. Κάθε candidate repository πρέπει να ικανοποιεί το ίδιο σύνολο κριτηρίων ανεξάρτητα από το cohort στο οποίο ανήκει. Συγκεκριμένα, η εργασία πρέπει να έχει γίνει αποδεκτή από το JOSS μέσα σε ένα από τα δύο χρονικά παράθυρα, η Python πρέπει να αποτελεί την κύρια γλώσσα υλοποίησης με τουλάχιστον 70% των source lines, το repository πρέπει να περιέχει μεταξύ 500 και 50.000 logical lines Python και να είναι δημόσια προσβάσιμο, ενώ πρέπει επίσης να είναι δυνατό να επιλυθεί το snapshot που αντιστοιχεί στον χρόνο αποδοχής. Παράλληλα, αποκλείονται forks ή mirrors άλλων repositories του corpus, repositories των οποίων το μετρούμενο περιεχόμενο αποτελείται κυρίως από auto-generated ή vendored code και περιπτώσεις που δεν ολοκληρώνουν επιτυχώς το measurement pipeline μετά την προκαθορισμένη retry procedure.

Η χρήση αυτών των ίδιων κανόνων και στα δύο cohorts έχει δύο στόχους. Πρώτον, περιορίζει τον κίνδυνο να χρησιμοποιηθούν διαφορετικά standards επιλεξιμότητας για το Pre και το Post cohort. Δεύτερον, επιτρέπει σε έναν ανεξάρτητο ερευνητή να εφαρμόσει τα ίδια criteria στην ίδια αρχική λίστα υποψηφίων και να ανακατασκευάσει τη διαδικασία επιλογής χωρίς να χρειάζεται να γνωρίζει μη καταγεγραμμένες αποφάσεις του αρχικού analyst. Η δυνατότητα αυτή είναι ουσιώδης για τη reproducibility της ίδιας της mining study και όχι μόνο για τη reproducibility του λογισμικού που μελετάται.

Όταν ο αριθμός των κατάλληλων repositories υπερβαίνει τον προκαθορισμένο στόχο, η τελική επιλογή δεν βασίζεται στη σειρά εμφάνισης των candidates ή σε χειροκίνητη προτίμηση. Αντίθετα, χρησιμοποιείται **seeded random sampling**. Η σειρά των candidates τυχαioποιείται με σταθερό seed και το pipeline εξετάζει διαδοχικά τα repositories μέχρι να συγκεντρωθούν 60 κατάλληλες περιπτώσεις σε κάθε cohort. Η χρήση σταθερού seed σημαίνει ότι η ίδια αρχική λίστα, οι ίδιοι κανόνες φιλτραρίσματος και η ίδια configuration παράγουν την ίδια σειρά ελέγχου και, συνεπώς, το ίδιο corpus. Η επιλογή αυτή περιορίζει τα analyst degrees of freedom και εμποδίζει την εκ των υστέρων αντικατάσταση repositories με βάση τα αποτελέσματα των μετρικών τους. Η αναπαραγώγιμη αυτή διαδικασία αποτελεί ήδη μία από τις δηλωμένες μεθοδολογικές συνεισφορές της εργασίας.

Στην πραγματική εκτέλεση της διαδικασίας χρειάστηκε να ελεγχθούν **96 Pre** και **88 Post candidate repositories** μέχρι να επιλεγούν **60 repositories** ανά cohort. Το γεγονός ότι οι δύο αριθμοί είναι σχετικά

κοντά έχει σημασία από πλευράς selection bias. Περίπου το 65% των repositories που εξετάστηκαν πληρούσαν τελικά τα κριτήρια καταλληλότητας και η αναλογία ήταν παρόμοια στις δύο περιόδους. Επομένως, δεν παρατηρείται έντονη ασυμμετρία σύμφωνα με την οποία ένα cohort θα απαιτούσε πολύ πιο επιθετικό filtering από το άλλο. Μια τέτοια ασυμμετρία θα μπορούσε να υποδηλώνει ότι τα κριτήρια ένταξης αλληλεπιδρούν με την εποχή και θα δυσχέραινε την ερμηνεία των συγκρίσεων.

Η διαδικασία δεν σταματά στην αρχική επιλογή των 120 repositories. Κάθε επιλεγμένο repository πρέπει στη συνέχεια να ολοκληρώσει το measurement pipeline. Από τα 120 επιλεγμένα repositories, **119** μετρήθηκαν επιτυχώς. Η μοναδική αποτυχία προήλθε από το Post cohort, με αποτέλεσμα το τελικό analytical dataset να περιλαμβάνει **60 Pre** και **59 Post repositories**. Η αποτυχία αυτή δεν αντικαταστάθηκε με νέο repository μετά την ολοκλήρωση της μέτρησης, επειδή μια τέτοια εκ των υστέρων αντικατάσταση θα μπορούσε να εισαγάγει πρόσθετη ερευνητική ευχέρεια και να μεταβάλει το προκαθορισμένο sample. Αντίθετα, η απώλεια καταγράφηκε ρητά και το τελικό sample size αναφέρεται με διαφάνεια.

Η καταγραφή της συγκεκριμένης αποτυχίας είναι σημαντική και από την άποψη του attrition bias. Σε μία mining study, η αποτυχία μέτρησης δεν είναι απαραίτητα τυχαία. Για παράδειγμα, παλαιότερα repositories μπορεί να παρουσιάζουν μεγαλύτερη δυσκολία στην επαναφορά dependencies ή στη resolution ιστορικών snapshots. Για τον λόγο αυτό, η μεθοδολογία ορίζει ότι περιπτώσεις που αποτυγχάνουν μετά τη documented retry procedure δεν εξαφανίζονται σιωπηρά από το dataset, αλλά καταγράφονται μαζί με τον λόγο αποκλεισμού τους. Στην παρούσα μελέτη, η απώλεια μόνο ενός repository δεν δημιουργεί από μόνη της μεγάλη αριθμητική ανισορροπία μεταξύ των cohorts, αλλά εξακολουθεί να αναφέρεται ώστε ο αναγνώστης να γνωρίζει ακριβώς πώς προέκυψε το τελικό $n=119$.

Επιπλέον, για κάθε repository διατηρείται το ιστορικό του τρόπου με τον οποίο επιλύθηκε το acceptance-time snapshot. Η διαδικασία ακολουθεί προκαθορισμένη σειρά προτεραιότητας: αρχικά χρησιμοποιείται το archived version tag των JOSS metadata, στη συνέχεια το πλησιέστερο release tag και, ως τελευταία επιλογή, το τελευταίο commit του default branch πριν από την ημερομηνία αποδοχής. Ο ενεργοποιημένος κανόνας καταγράφεται ανά repository, επειδή ενδεχόμενη διαφορά στην ποιότητα της snapshot resolution μεταξύ των cohorts θα μπορούσε να δημιουργήσει measurement artefact. Στην τελική εφαρμογή, 109 από τα 120 επιλεγμένα repositories επιλύθηκαν μέσω της ακριβούς αρχειοθετημένης έκδοσης, ενώ μόνο 11 χρειάστηκαν fallback rule.

Συνολικά, η reproducibility της κατασκευής του corpus στηρίζεται σε τέσσερα στοιχεία: προκαθορισμένα **eligibility criteria**, σταθερό **random seeded**, πλήρη καταγραφή της **attrition flow** και **deterministic** κανόνες **snapshot resolution**. Η αξία αυτής της διαδικασίας δεν είναι μόνο πρακτική. Σε μία observational σύγκριση δύο χρονικών περιόδων, όπου η αιτιώδης απόδοση είναι ήδη περιορισμένη, είναι ιδιαίτερα σημαντικό τουλάχιστον η διαδικασία επιλογής των μονάδων ανάλυσης να μην εισάγει επιπλέον μη ελεγχόμενη αβεβαιότητα. Με αυτόν τον τρόπο, η μελέτη επιδιώκει να διασφαλίσει ότι οι διαφορές που παρατηρούνται αργότερα στις μετρικές δεν αποτελούν προϊόν αδιαφανούς ή μη επαναλήψιμης επιλογής των repositories, αλλά προκύπτουν από ένα corpus που μπορεί να ανακατασκευαστεί από τους δημοσιευμένους κανόνες, τα configuration artefacts και το σταθερό seed.

3.4.6 Το Τελικό Σύνολο Δεδομένων

Το σύνολο δεδομένων δημιουργήθηκε από την πλήρη λίστα των εργασιών που έγιναν δεκτές από το Journal of Open Source Software (JOSS). Ο Πίνακας 3.2 παρουσιάζει κάθε στάδιο φιλτραρίσματος, από την πλήρη λίστα μέχρι τα αποθετήρια που τελικά μετρήθηκαν.

Πίνακας 3.2: Από όλες τις εργασίες του JOSS στο τελικό σύνολο ανάλυσης

Στάδιο	Πλήθος
Όλες οι αποδεκτές εργασίες του JOSS (από το JOSS API)	3.638
Αφαιρέθηκαν: εκτός της περιόδου μελέτης	740
Αφαιρέθηκαν: εντός της μεταβατικής περιόδου (Νοέμβριος 2022 – Ιούνιος 2023)	281
Υπόλοιπα εντός των δύο περιόδων των cohorts	2.617
Υπόλοιπα μετά το φίλτρο γλώσσας Python	1.693
Υπόλοιπα μετά τον καθαρισμό των URL των αποθετηρίων	1.583
Ελέγχθηκαν ένα προς ένα από τη διαδικασία δειγματοληψίας (Pre / Post)	96 / 88
Επιλέχθηκαν για το corpus (Pre / Post)	60 / 60
Μετρήθηκαν με επιτυχία (Pre / Post)	(60 / 59)=119

Η δειγματοληψία πραγματοποιήθηκε όπως είχε σχεδιαστεί. Η διαδικασία δειγματοληψίας πέρασε από τη λίστα των υποψηφίων αποθετηρίων με τυχαία σειρά, η οποία καθοριζόταν από ένα σταθερό seed ώστε η διαδικασία να μπορεί να επαναληφθεί ακριβώς, και αποδεχόταν τα αποθετήρια που περνούσαν τους ελέγχους καταλληλότητας μέχρι να συγκεντρωθούν 60 αποθετήρια σε κάθε cohort. Περίπου το 65% των υποψηφίων που ελέγχθηκαν ήταν κατάλληλοι και το ποσοστό αυτό ήταν σχεδόν ίδιο και στα δύο cohorts. Αυτό έχει σημασία, καθώς δείχνει ότι το φιλτράρισμα δεν ευνόησε τη μία εποχή έναντι της άλλης. Ένα Post αποθετήριο απέτυχε κατά τη διαδικασία μέτρησης, επομένως το τελικό σύνολο ανάλυσης είναι $n = 60$ για το Pre και $n = 59$ για το Post.

Κάθε αποθετήριο μετρήθηκε στο snapshot της αποδοχής του, δηλαδή στην κατάσταση του κώδικα τη στιγμή που το JOSS το αποδέχθηκε. Για 109 από τα 120 αποθετήρια (91,7%), το snapshot προήλθε από την ακριβή αρχειοθετημένη έκδοση (version tag), η οποία αποτελεί τον πιο αξιόπιστο κανόνα. Μόνο 11 αποθετήρια χρειάστηκαν έναν από τους δύο εναλλακτικούς κανόνες. Η Ενότητα 4.6 ελέγχει αν αυτά τα 11 αποθετήρια επηρεάζουν τα αποτελέσματα.

Κατά την ανάγνωση των συγκρίσεων πρέπει να ληφθούν υπόψη δύο χαρακτηριστικά του corpus:

- **Τα Post αποθετήρια είναι μεγαλύτερα.** Το διάμεσο Post αποθετήριο έχει 7.232 λογικές γραμμές Python, ενώ το διάμεσο Pre αποθετήριο έχει 4.172. Το μέγεθος του αποθετηρίου καταγράφεται ως συγχυτικός παράγοντας (covariate) και δύο μετρικές φαίνεται να είναι ευαίσθητες σε αυτό, όπως παρουσιάζεται στην Ενότητα 4.6.

- **Το Post είναι περισσότερο συγκεντρωμένο στα τελευταία έτη.** 36 από τα 60 Post αποθετήρια έγιναν αποδεκτά το 2025, απλώς επειδή το JOSS δημοσιεύει περισσότερες εργασίες κάθε χρόνο.

3.4.7 Ανάλυση της Ροής Επιλογής του Corpus

Η μετάβαση από το αρχικό σύνολο των 3.638 εργασιών του Journal of Open Source Software (JOSS) στο τελικό αναλυτικό corpus των 119 repositories δεν αποτελεί μία ενιαία διαδικασία αποκλεισμού, αλλά μια διαδοχή σταδίων με διαφορετικό μεθοδολογικό σκοπό. Για τον λόγο αυτό, η attrition flow δεν πρέπει να ερμηνεύεται απλώς ως αριθμητική μείωση του διαθέσιμου δείγματος. Κάθε στάδιο περιορίζει διαφορετική πηγή ετερογένειας και συμβάλλει στον καθορισμό του τελικού πληθυσμού στον οποίο αναφέρονται τα αποτελέσματα της μελέτης. **Ο Πίνακας 3.2** καταγράφει αριθμητικά αυτή τη διαδικασία, από τις 3.638 αρχικές εργασίες έως τα 119 repositories που ολοκλήρωσαν επιτυχώς το measurement pipeline.

Στο πρώτο στάδιο εφαρμόστηκαν τα χρονικά όρια του ερευνητικού σχεδιασμού. Από τις 3.638 αρχικές εργασίες, 740 βρίσκονταν εκτός της συνολικής περιόδου μελέτης και συνεπώς δεν μπορούσαν να ενταχθούν σε κανένα από τα δύο cohorts. Ο αριθμός αυτός αντιστοιχεί περίπου στο 20,3% του αρχικού συνόλου. Επιπλέον, 281 εργασίες, δηλαδή περίπου 7,7% του αρχικού συνόλου, ανήκαν στη μεταβατική περίοδο μεταξύ Νοεμβρίου 2022 και Ιουνίου 2023 και αποκλείστηκαν σκόπιμα. Μετά την εφαρμογή των δύο αυτών χρονικών κανόνων παρέμειναν 2.617 εργασίες, δηλαδή περίπου το 71,9% της αρχικής λίστας.

Ο αποκλεισμός της μεταβατικής περιόδου διαφέρει ουσιαστικά από έναν αποκλεισμό λόγω τεχνικής ακαταλληλότητας ή ανεπαρκούς ποιότητας. Οι 281 αυτές εργασίες δεν θεωρούνται λιγότερο κατάλληλες ως software projects. Αποκλείονται επειδή η χρονική τους θέση βρίσκεται στο διάστημα ανάμεσα στις δύο καθαρά ορισμένες εποχές του σχεδιασμού. Η συμπερίληψή τους είτε στο Pre είτε στο Post cohort θα αύξανε την αβεβαιότητα ως προς την ταξινόμηση της εποχής και θα μπορούσε να μειώσει τη διακριτική ικανότητα της σύγκρισης. Επομένως, η συγκεκριμένη απώλεια παρατηρήσεων αποτελεί συνειδητή σχεδιαστική επιλογή και όχι αποτέλεσμα αποτυχίας του corpus-building pipeline.

Το επόμενο στάδιο αφορούσε το φίλτρο γλώσσας προγραμματισμού. Από τις 2.617 εργασίες που βρίσκονταν εντός των χρονικών παραθύρων, 1.693 παρέμειναν μετά την εφαρμογή του κριτηρίου της Python. Αυτό αντιστοιχεί περίπου στο 64,7% των χρονικά επιλέξιμων εργασιών, ενώ περίπου το 35,3% αποκλείστηκε στο συγκεκριμένο στάδιο.

Η σχετικά μεγάλη αυτή μείωση είναι αναμενόμενη, καθώς το JOSS δημοσιεύει scientific software που έχει υλοποιηθεί σε πολλές διαφορετικές γλώσσες προγραμματισμού. Η παρούσα εργασία περιορίζεται σε repositories στα οποία η Python αποτελεί την κύρια γλώσσα υλοποίησης, επειδή τα βασικά static-analysis instruments του measurement pipeline είναι Python-specific. Η επιλογή αυτή περιορίζει την εξωτερική εγκυρότητα της μελέτης, αλλά ταυτόχρονα αυξάνει τη συγκρισιμότητα μεταξύ των μονάδων ανάλυσης. Η χρήση διαφορετικών language-specific analysers για διαφορετικά repositories θα εισήγαγε πρόσθετη μεταβλητότητα στις ίδιες τις μετρήσεις, καθιστώντας δυσκολότερο τον διαχωρισμό μεταξύ πραγματικής διαφοράς των cohorts και διαφοράς που προκύπτει από τα εργαλεία μέτρησης.

Ακολούθησε το στάδιο καθαρισμού και επαλήθευσης των repository URLs. Μετά το συγκεκριμένο φίλτρο παρέμειναν 1.583 repositories, σε σύγκριση με τα 1.693 του προηγούμενου σταδίου. Επομένως, περίπου το 93,5% των περιπτώσεων που είχαν ήδη περάσει το φίλτρο γλώσσας διατηρήθηκαν, ενώ περίπου το 6,5% αποκλείστηκε κατά τον καθαρισμό των URLs. Η σχετικά μικρή μείωση δείχνει ότι,

αφού μία εργασία ικανοποιούσε τα βασικά χρονικά και γλωσσικά criteria, η μεγάλη πλειονότητα των repository references μπορούσε να διατηρηθεί στη δεξαμενή των υποψήφιων περιπτώσεων.

Ιδιαίτερη προσοχή απαιτείται στην ερμηνεία του επόμενου σταδίου. Τα 1.583 repositories δεν ελέγχθηκαν όλα εξαντλητικά. Ο σχεδιασμός είχε προκαθορίσει στόχο 60 repositories ανά cohort. Η sampling procedure χρησιμοποίησε σταθερό random seed, δημιούργησε αναπαραγώγιμη τυχαία σειρά των candidates και εξέταζε διαδοχικά τις περιπτώσεις μέχρι να συγκεντρωθούν 60 κατάλληλα repositories για κάθε χρονική περίοδο. Για την επίτευξη αυτού του στόχου χρειάστηκε να εξεταστούν 96 Pre candidates και 88 Post candidates.

Από τα 96 Pre repositories που εξετάστηκαν, 60 πέρασαν τα κριτήρια καταλληλότητας, ποσοστό 62,5%. Στο Post cohort, 60 από τα 88 repositories που εξετάστηκαν κρίθηκαν κατάλληλα, δηλαδή περίπου 68,2%. Συνολικά, 120 από τα 184 repositories που αξιολογήθηκαν σε αυτό το στάδιο εισήχθησαν στο corpus, ποσοστό περίπου 65,2%. Η διαφορά μεταξύ των δύο cohorts είναι περίπου 5,7 ποσοστιαίες μονάδες, γεγονός που δεν υποδεικνύει από μόνο του έντονη ασυμμετρία στη διαδικασία επιλογής.

Το στοιχείο αυτό έχει σημασία για την αξιολόγηση πιθανού selection bias. Αν η διαδικασία απαιτούσε, για παράδειγμα, τον έλεγχο πολύ μεγαλύτερου αριθμού Pre repositories για να εντοπιστούν 60 κατάλληλες περιπτώσεις, ενώ το Post cohort συμπληρωνόταν σχεδόν αμέσως, αυτό θα μπορούσε να υποδηλώνει ότι τα κριτήρια καταλληλότητας αλληλεπιδρούν συστηματικά με την εποχή. Στην παρούσα περίπτωση, οι δύο αναλογίες είναι αρκετά κοντά ώστε να μην παρέχουν ένδειξη μεγάλης διαφοροποίησης στη δυσκολία συγκρότησης των δύο cohorts. Η ίδια η καταγραφή της διαδικασίας αναφέρει ότι το ποσοστό καταλληλότητας ήταν παρόμοιο μεταξύ των δύο περιόδων.

Παράλληλα, οι συγκεκριμένες αναλογίες δεν πρέπει να επεκταθούν σε όλα τα 1.583 repositories της candidate pool. Η δειγματοληψία σταμάτησε μόλις συγκεντρώθηκαν οι 60 κατάλληλες περιπτώσεις ανά cohort. Επομένως, το 62,5% και το 68,2% περιγράφουν αποκλειστικά τους candidates που πράγματι εξετάστηκαν και δεν αποτελούν εκτίμηση του συνολικού ποσοστού επιλεξιμότητας όλων των repositories που παρέμεναν μετά τον καθαρισμό των URLs.

Μετά την εφαρμογή της sampling procedure, το corpus περιλάμβανε ισορροπημένα 60 Pre και 60 Post repositories. Η ισορροπημένη αυτή αρχική σύνθεση είναι χρήσιμη για τη σύγκριση των δύο χρονικών ομάδων, καθώς αποφεύγει μια μεγάλη διαφορά sample size που θα μπορούσε να επηρεάσει την ακρίβεια των εκτιμήσεων μεταξύ των cohorts.

Από τα 120 επιλεγμένα repositories, 119 ολοκλήρωσαν επιτυχώς τη διαδικασία μέτρησης. Ένα repository του Post cohort απέτυχε κατά το measurement stage, με αποτέλεσμα το τελικό analytical dataset να αποτελείται από 60 Pre και 59 Post repositories. Αυτό αντιστοιχεί σε ποσοστό επιτυχούς μέτρησης περίπου 99,2% μεταξύ των repositories που είχαν ήδη επιλεγεί για το corpus.

Η πολύ μικρή αυτή τελική attrition είναι σημαντική από την άποψη της εσωτερικής εγκυρότητας. Σε μία repository-mining study, οι αποτυχίες μέτρησης δεν είναι κατ' ανάγκη τυχαίες. Παλαιότερα repositories, για παράδειγμα, μπορεί να παρουσιάζουν μεγαλύτερες δυσκολίες ως προς την ανάκτηση ιστορικών artefacts ή τη συμβατότητα με σύγχρονα εργαλεία. Εάν μεγάλος αριθμός repositories αποτύγχανε και οι αποτυχίες συγκεντρώνονταν σε μία μόνο χρονική περίοδο, το τελικό corpus θα μπορούσε να αποτελεί συστηματικά επιλεγμένο υποσύνολο του αρχικού population.

Στην παρούσα περίπτωση, η απώλεια μόνο μίας από τις 120 επιλεγμένες μονάδες δεν δημιουργεί σημαντική αριθμητική διαφοροποίηση. Παρ' όλα αυτά, η περίπτωση καταγράφεται αντί να αντικατασταθεί εκ των υστέρων από διαφορετικό repository. Με αυτόν τον τρόπο αποφεύγεται η

εισαγωγή πρόσθετων analyst degrees of freedom μετά την ολοκλήρωση της προκαθορισμένης sampling procedure και διατηρείται η αναπαραγωγικότητα της κατασκευής του corpus.

Ένα ακόμη σημαντικό χαρακτηριστικό της τελικής σύνθεσης του dataset αφορά το μέγεθος των repositories. Το διάμεσο Post repository περιλαμβάνει 7.232 logical lines Python, ενώ το αντίστοιχο Pre repository περιλαμβάνει 4.172. Η διαφορά αυτή δείχνει ότι τα δύο cohorts δεν είναι πλήρως ισοδύναμα ως προς το software size. Αυτό δεν αποτελεί λόγο απόρριψης της σύγκρισης, αλλά δημιουργεί πιθανό confounding για metrics που εξαρτώνται από τον όγκο του source code. Για τον λόγο αυτό το repository size καταγράφεται ως covariate και εξετάζεται ειδικά στην Ενότητα 4.6.

Η παρατήρηση αυτή είναι ιδιαίτερα σημαντική επειδή η μελέτη δεν συγκρίνει τεχνητά projects ίδιου μεγέθους, αλλά πραγματικό δημοσιευμένο scientific software. Η αύξηση του median repository size στη νεότερη περίοδο μπορεί να αντανakλά ευρύτερες αλλαγές στο scope, στην ωριμότητα ή στις απαιτήσεις των projects. Παράλληλα, όμως, μπορεί να επηρεάζει metrics όπως η duplication και ο minimum maintainability index. Επομένως, η διαφορά μεγέθους δεν πρέπει να αγνοείται κατά την ερμηνεία των αποτελεσμάτων.

Το Post cohort παρουσιάζει επίσης διαφορετική χρονική συγκέντρωση. Από τα 60 αρχικά επιλεγμένα Post repositories, 36 έγιναν αποδεκτά το 2025. Η συγκέντρωση αυτή αντανakλά την αύξηση του αριθμού δημοσιεύσεων του JOSS στις πιο πρόσφατες χρονιές και όχι διαφορετικό sampling rule. Παρόλα αυτά, έχει ερμηνευτική σημασία, επειδή το Post cohort αντιπροσωπεύει σε μεγαλύτερο βαθμό την ώριμη φάση της περιόδου 2023–2025 παρά ένα απολύτως ομοιόμορφο distribution σε όλα τα Post έτη.

Αν συγκριθεί μόνο το αρχικό πλήθος με το τελικό αναλυτικό sample, τα 119 repositories αντιστοιχούν περίπου στο 3,3% των 3.638 αρχικών JOSS papers. Το ποσοστό αυτό, ωστόσο, δεν πρέπει να περιγραφεί ως ποσοστό «επιβίωσης» μετά από quality filtering. Μεγάλο μέρος της μείωσης προκύπτει από τον προκαθορισμένο ερευνητικό scope: χρονικά όρια, αποκλεισμός της transition period, περιορισμός σε Python και, κυρίως, προκαθορισμένο sample size 60 repositories ανά cohort. Το $(n=119)$ αποτελεί συνεπώς προϊόν του sampling design και όχι απλώς το μικρό υπόλοιπο μιας σειράς αποτυχημένων repositories.

Η παρατήρηση αυτή είναι σημαντική για την external validity. Το τελικό dataset δεν επιχειρεί να αναπαραστήσει το σύνολο των 3.638 JOSS papers με αναλογική δειγματοληψία. Αντιπροσωπεύει ένα πολύ πιο συγκεκριμένο population: peer-reviewed, open-source, Python-dominant scientific software, αποδεκτό εντός των δύο προκαθορισμένων χρονικών παραθύρων και ικανό να αναλυθεί στο acceptance-time snapshot.

Συνολικά, η attrition flow προσφέρει τρεις βασικές ενδείξεις για την ποιότητα της κατασκευής του corpus. Πρώτον, η μεγάλη μείωση από τις 3.638 αρχικές εργασίες οφείλεται κυρίως στον προκαθορισμένο scope και όχι σε εκ των υστέρων επιλογή με βάση τα αποτελέσματα. Δεύτερον, η παρόμοια αναλογία καταλληλότητας των Pre και Post candidates που πράγματι εξετάστηκαν δεν δείχνει έντονη διαφοροποίηση της sampling procedure μεταξύ των εποχών. Τρίτον, η επιτυχής μέτρηση 119 από τα 120 επιλεγμένα repositories δείχνει ότι η τελική απώλεια λόγω του measurement pipeline ήταν πολύ περιορισμένη.

Επομένως, παρότι η επιλογή του corpus δεν εξαλείφει τους γενικότερους περιορισμούς ενός observational σχεδιασμού, η πλήρης καταγραφή της attrition flow επιτρέπει στον αναγνώστη να γνωρίζει με ακρίβεια πώς προέκυψε το τελικό dataset και ποιο population αντιπροσωπεύουν οι συγκρίσεις που ακολουθούν. Αυτό είναι κρίσιμο για την ερμηνεία των αποτελεσμάτων των RQ1–RQ4,

καθώς διαχωρίζει τις πραγματικές διαφορές που παρατηρούνται μεταξύ των cohorts από πιθανές διαφοροποιήσεις που θα μπορούσαν να έχουν εισαχθεί κατά τη διαδικασία επιλογής του δείγματος.

3.5 Attribution της AI Assistance

Η σύγκριση μεταξύ των εποχών δείχνει αν άλλαξε το record· το attribution είναι αυτό που συνδέει οποιαδήποτε αλλαγή με το εργαλείο. Εφόσον δεν υπάρχει registry έργων που χρησιμοποιούν AI assistance, το attribution βασίζεται στο disclosure: δηλώσεις των ίδιων των developers ότι χρησιμοποιήθηκε AI assistance.

3.5.1 Disclosure Signals

Ένα repository στο Post cohort ταξινομείται ως Post-Disclosed εάν οποιοδήποτε από τα ακόλουθα artefacts περιέχει disclosure σχετικά με AI-assisted development: το repository README και η τεκμηρίωση των contributors· το JOSS paper source ([paper.md](#)), του οποίου οι conventions για acknowledgements ακολουθούν αυτές της συνηθισμένης επιστημονικής δημοσίευσης· ή τα τελευταία 200 commit messages πριν από το acceptance snapshot, όπου μπορούν να εμφανιστούν tool-generated trailers (για παράδειγμα co-authorship trailers που προστίθενται από assistant integrations) και ρητές αναφορές.

Το matching είναι rule-based, case-insensitive και word-boundary aware, χρησιμοποιώντας μια σταθερή λίστα patterns που καλύπτει τους βασικούς assistants και γενικές διατυπώσεις disclosure («AI-assisted», «generated with», «LLM-assisted» και τα product names των σημαντικότερων εργαλείων). Η πλήρης λίστα patterns αποτελεί configuration artefact και αναπαράγεται αυτούσια. Η συμπεριφορά του classifier πρέπει να είναι auditable: ο αναγνώστης θα πρέπει να μπορεί να προσδιορίσει, για οποιοδήποτε κείμενο, ακριβώς τι θα αποφάσιζε ο classifier.

Η επιλογή rule-based matching αντί για learned classifier γίνεται σκόπιμα. Ένας learned classifier θα μπορούσε να προσφέρει υψηλότερο recall, με κόστος αποφάσεων που δεν μπορούν να εξηγηθούν case by case. Σε μια μελέτη της οποίας τα αιτιώδη claims βασίζονται σε αυτό το partition, ένας auditable αλλά under-inclusive classifier είναι προτιμότερος από έναν opaque και comprehensive classifier.

3.5.2 Manual Review of Borderline Matches

Το pattern matching σε φυσική γλώσσα παράγει προβλέψιμα false positives: μια dependency που έχει όνομα σχετικό με κάποιο model, μια πρόταση στο [paper.md](#) που αναφέρεται στα language models ως ερευνητικό θέμα ή ένα commit message που αφαιρεί μια AI integration. Κάθε raw match επομένως περνά από manual review, κατά την οποία εξετάζεται το matched context (έως 120 χαρακτήρες γύρω από το hit) και το match επιβεβαιώνεται ή απορρίπτεται με βάση ένα γραπτό rubric: ένα match επιβεβαιώνεται μόνο εάν το κείμενο δηλώνει ότι χρησιμοποιήθηκε AI assistance στην ανάπτυξη του συγκεκριμένου λογισμικού. Οι αποφάσεις καταγράφονται σε ένα decisions file, το οποίο το pipeline διαβάζει ξανά, ώστε το τελικό partition να είναι αναπαραγώγιμο από τα δημοσιευμένα artefacts. Το review πραγματοποιείται πριν εξεταστεί οποιαδήποτε metric του συγκεκριμένου repository και αναφέρεται το confirmation rate.

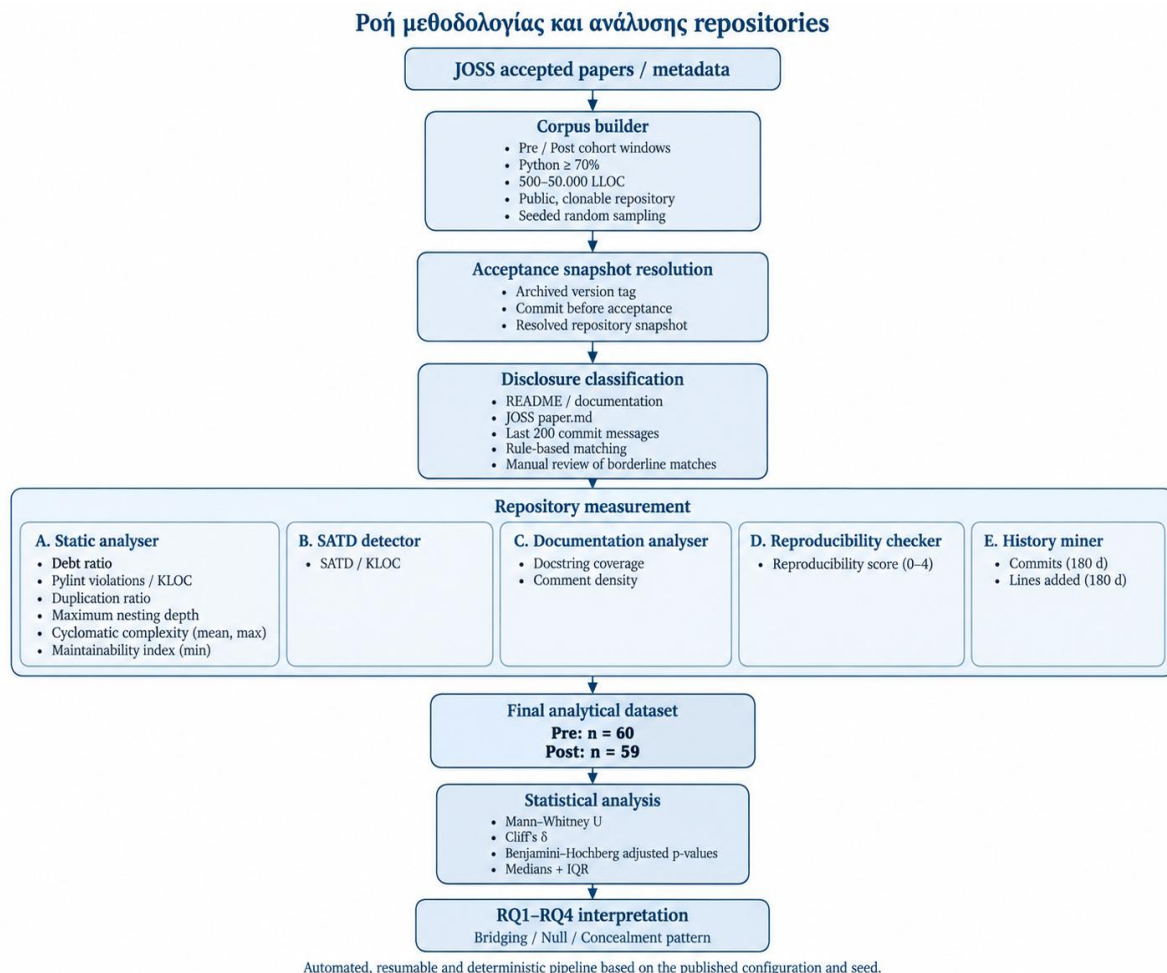
3.5.3 Η Ασυμμετρία του Non-Disclosure

Ο βασικός περιορισμός του disclosure-based attribution πρέπει να δηλωθεί κατά τον σχεδιασμό και όχι να ανακαλυφθεί κατά την ερμηνεία: non-disclosure δεν σημαίνει non-use. Οι developers σπάνια τεκμηριώνουν τη διαμόρφωση του editor τους και το πραγματικό ποσοστό χρήσης assistants μεταξύ

των Post-Undisclosed repositories είναι άγνωστο και πιθανότατα σημαντικό. Το partition επομένως συγκρίνει γνωστά assisted repositories με μια ομάδα mixed-and-unknown, και η contamination της ομάδας σύγκρισης ωθεί τη μετρούμενη διαφορά προς το μηδέν: οποιαδήποτε διαφορά RQ3 που παραμένει είναι υποεκτίμηση της πραγματικής διαφοράς, ενώ ένα null αποτέλεσμα στο RQ3 δεν παρέχει πληροφορία και δεν μπορεί να θεωρηθεί αθωωτικό. Όλη η ερμηνεία του RQ3 δεσμεύεται από αυτή την ασυμμετρία, και αυτή αναφέρεται πρώτη μεταξύ των threats to construct validity.

3.6 Υποδομή Μέτρησης

Η υποδομή αποτελείται από το προηγούμενος επικυρωμένο measurement pipeline και τρία components ειδικά για τον mining σχεδιασμό: τον corpus builder, τον disclosure classifier και τον history miner. Όλα τα components είναι automated, resumable και deterministic με δεδομένα τα δημοσιευμένα configuration και seed. Ο πλήρης source code συνοδεύει τη διπλωματική εργασία ως replication package. Το **Σχήμα 3.1** παρουσιάζει τη ροή της διαδικασίας:



Σχήμα 3.1: Ροή της διαδικασίας συλλογής, επεξεργασίας και ανάλυσης των repositories. Η διαδικασία ξεκινά από την κατασκευή του corpus και την επίλυση του acceptance snapshot, συνεχίζει με τον εντοπισμό δηλώσεων χρήσης AI assistance και την εξαγωγή των μετρικών τεχνικού χρέους,

τεκμηρίωσης, αναπαραγωγιμότητας και δραστηριότητας ανάπτυξης και καταλήγει στη στατιστική σύγκριση των Pre και Post cohorts και στη σύνθεση των αποτελεσμάτων για τα RQ1–RQ4.

3.6.1 Reused, Validated Analysers

Τρεις βασικοί analysers επαναχρησιμοποιούνται από το validated measurement pipeline. Ο static analyser υπολογίζει radon-based complexity και maintainability, Pylint violations με προκαθορισμένη configuration κανόνων, clone detection και approximate debt ratio σύμφωνα με την παράδοση SQA [19]. Ο documentation analyser υπολογίζει docstring coverage σε documentable objects και comment density. Ο reproducibility checker είχε αρχικά σχεδιαστεί γύρω από ένα τετραμερές πλαίσιο τεχνικών προϋποθέσεων αναπαραγωγιμότητας: dependency declaration, unmodified execution, determinism και external configuration. Τα detection characteristics των analysers, δηλαδή τι μπορεί και τι δεν μπορεί να εντοπίσει κάθε metric, καθώς και τα false-positive rates σε clean code, καθορίστηκαν πάνω σε seeded-defect fixtures πριν μετρηθεί οποιοδήποτε repository.

Στην εφαρμογή του reproducibility checker σε πραγματικά mined repositories απαιτείται ιδιαίτερη προσοχή. Σε αντίθεση με τα synthetic fixtures, τα πραγματικά repositories δεν διαθέτουν προκαθορισμένο entry point και συχνά εξαρτώνται από historical software environments. Για τον λόγο αυτό χρησιμοποιείται μια documented entry-point ladder: αρχικά εξετάζεται αν υπάρχει installable package με console entry point, στη συνέχεια το quick-start command που μπορεί να εξαχθεί από το README και, ως τελευταία επιλογή, το test suite. Ο ενεργοποιημένος κανόνας καταγράφεται ώστε να είναι δυνατός ο έλεγχος πιθανών διαφορών μεταξύ των cohorts.

Ωστόσο, στην τελική mining run οι installation/execution-related έλεγχοι δεν αντιμετωπίζονται ως ισότιμη βάση διαχρονικής σύγκρισης. Η δυνατότητα εγκατάστασης ή εκτέλεσης ενός παλαιότερου repository σε σύγχρονο Python environment μπορεί να επηρεάζεται από dependency ageing, αλλαγές στο packaging ecosystem και software bit rot, ανεξάρτητα από την αρχική ποιότητα ή αναπαραγωγιμότητα του λογισμικού. Κατά συνέπεια, το αρχικό τετραμερές πλαίσιο διατηρείται ως θεωρητική και validation βάση του checker, ενώ η μεταβλητή που χρησιμοποιείται στην κύρια cross-era ανάλυση ερμηνεύεται ως περιορισμένος repository-level proxy και όχι ως πλήρης μέτρηση end-to-end reproducibility.

Το correctness oracle του validation harness δεν εφαρμόζεται στα mined repositories, καθώς απαιτεί analytical reference solution, η οποία υπάρχει για τα synthetic fixtures αλλά όχι για αυθαίρετο επιστημονικό κώδικα. Οι συνέπειες αυτού του περιορισμού για την κάλυψη του construct της μελέτης συζητούνται στην Ενότητα 5.4.2.

3.6.2 Εντοπισμός Self-Admitted Technical Debt

Το SATD μετριέται με matching γραμμών σχολίων απέναντι στο standard marker vocabulary της mining literature (TODO, FIXME, HACK, XXX και τις φράσεις workaround και temporary), κανονικοποιημένο ανά χίλιες logical lines. Το Listing 1 παρουσιάζει τον πυρήνα του detector. Η σκόπιμη απλότητά του είναι το βασικό σημείο, καθώς η αναλυτική αξία του SATD σε αυτή τη μελέτη βρίσκεται στο ότι αποτελεί μέτρο αναγνώρισης, και η αναγνώριση είναι ακριβώς αυτό που αντιπροσωπεύει ένα comment marker.

```

from pathlib import Path
import re

SATD_MARKERS = [
    r"\bTODO\b",
    r"\bFIXME\b",
    r"\bHACK\b",
    r"\bXXX\b",
    r"\bworkaround\b",
    r"\btemporary\b",
]

_PATTERN = re.compile("|".join(SATD_MARKERS), re.IGNORECASE)

def satd_per_kloc(repo: Path, lloc_total: int) -> dict:
    """Count self-admitted technical debt markers in Python comments.

    Only comment text is searched: a string literal containing 'TODO'
    is user-facing text, not an admission of debt.
    """

    hits = []

    for pyfile in repo.rglob("*.py"):
        try:
            lines = pyfile.read_text(errors="ignore").splitlines()
        except OSError:
            continue

        for lineno, line in enumerate(lines, start=1):
            comment = line.partition("#")[2]

            if comment and _PATTERN.search(comment):
                hits.append((str(pyfile.relative_to(repo)), lineno))

    density = 1000.0 * len(hits) / lloc_total if lloc_total else None

    return {
        "satd_count": len(hits),
        "satd_per_kloc": density,
        "satd_locations": hits[:200], # capped audit trail
    }

```

Listing 1. Εντοπισμός SATD σε ένα repository snapshot

3.6.3 History Mining

Οι μετρικές development activity εξάγονται από το commit history των 180 ημερών πριν από το acceptance snapshot: αριθμός commits, γραμμές που προστέθηκαν και διαγράφηκαν, αριθμός διαφορετικών authors (οι authors αποθηκεύονται μόνο ως salted hashes) και churn, γραμμές που προστέθηκαν συν γραμμές που διαγράφηκαν, διαιρεμένες με τις τελικές γραμμές, ως mining analogue του write-versus-retain ratio. Το Listing 2 παρουσιάζει την εξαγωγή. Ένα παράθυρο σταθερό σε σχέση με την ημερομηνία αποδοχής κάθε repository, αντί για ένα ημερολογιακά ευθυγραμμισμένο παράθυρο, διατηρεί τη μετρική συγκρίσιμη μεταξύ των cohorts.

```

import hashlib
from datetime import timedelta
from git import Repo

WINDOW_DAYS = 180

def history_metrics(repo_path, acceptance_commit, salt: str) -> dict:
    """Commit-activity measures in the window before acceptance.

    All quantities are computed from commits reachable from the
    acceptance commit only, so post-acceptance history cannot leak in.
    """

    repo = Repo(repo_path)
    head = repo.commit(acceptance_commit)

    cutoff = head.committed_datetime - timedelta(days=WINDOW_DAYS)

    commits, added, deleted, authors = 0, 0, 0, set()

    for c in repo.iter_commits(acceptance_commit):
        if c.committed_datetime < cutoff:
            break

        commits += 1

        stats = c.stats.total
        added += stats["insertions"]
        deleted += stats["deletions"]

        authors.add(
            hashlib.sha256(
                (
                    salt
                    + (c.author.email or c.author.name or "")
                ).encode()
            ).hexdigest()[:16]
        )

    return {
        "commits_180d": commits,
        "lines_added_180d": added,
        "lines_deleted_180d": deleted,
        "n_authors_180d": len(authors),
    }

```

Listing 2. Εξαγωγή history-window σε σχέση με το acceptance snapshot

3.6.4 Covariates

Για κάθε repository το pipeline καταγράφει covariates που χρησιμοποιούνται από την ανάλυση για την εξέταση του confounding αντί για την υπόθεση ότι απουσιάζει: repository size (logical lines), αριθμός αρχείων, repository age κατά την αποδοχή, συνολικός αριθμός commits, αριθμός authors, JOSS subject tag ως domain proxy και η παρουσία test directory και continuous-integration configuration. Τα δύο τελευταία λειτουργούν επίσης ως coarse practice-adoption indicators από μόνα τους, συνδέοντας τη μελέτη με τον κατάλογο πρακτικών των Arvanitou et al. [10].

3.6.5 Επιχειρησιακοποίηση, Παραδοχές και Περιορισμοί των Μετρικών

Η επιλογή των μετρικών αποτελεί κρίσιμο μέρος του ερευνητικού σχεδιασμού, επειδή καμία από τις έννοιες που εξετάζονται στην παρούσα εργασία —τεχνικό χρέος, συντηρησιμότητα, τεκμηρίωση ή αναπαραγωγιμότητα— δεν είναι άμεσα παρατηρήσιμη ως μία ενιαία αριθμητική ποσότητα. Αντίθετα, καθεμία επιχειρησιακοποιείται μέσω συγκεκριμένων, μετρήσιμων ιδιοτήτων του repository. Για τον λόγο αυτό, οι μετρικές δεν αντιμετωπίζονται ως απόλυτες μετρήσεις της συνολικής «ποιότητας» του λογισμικού, αλλά ως επιμέρους proxies που παρατηρούν διαφορετικές όψεις της.

Η διάκριση αυτή είναι ιδιαίτερα σημαντική στο πλαίσιο της παρούσας μελέτης. Ένα repository μπορεί, για παράδειγμα, να παρουσιάζει χαμηλή πυκνότητα παραβιάσεων στατικής ανάλυσης χωρίς να διαθέτει καλή αρχιτεκτονική, ενώ ένα έργο με υψηλή docstring coverage μπορεί να περιέχει docstrings χαμηλής πληροφοριακής αξίας. Αντίστοιχα, η ύπαρξη σωστά δηλωμένων dependencies αποτελεί προϋπόθεση αναπαραγωγιμότητας αλλά δεν αποδεικνύει ότι τα επιστημονικά αποτελέσματα μπορούν πράγματι να αναπαραχθούν. Επομένως, η ερμηνεία βασίζεται στο συνολικό μοτίβο πολλών μετρικών και όχι σε μία μεμονωμένη τιμή.

Οι δεκατρείς επιμέρους μεταβλητές της κύριας ανάλυσης οργανώνονται σε τρία επίπεδα. Το Tier 1 αποτυπώνει τη δραστηριότητα ανάπτυξης και χρησιμοποιείται κυρίως ως context. Το Tier 2 περιλαμβάνει proxies τεχνικού χρέους και δομικών ιδιοτήτων του κώδικα. Το Tier 3 περιλαμβάνει χαρακτηριστικά τεκμηρίωσης και repository-level αναπαραγωγιμότητας, τα οποία συνδέονται ειδικότερα με τις απαιτήσεις του επιστημονικού λογισμικού [9], [10], [12].

3.6.5.1 Development Activity: Commits και Lines Added

Οι δύο μεταβλητές του Tier 1 **Commits (180 d)** και **Lines added (180 d)**, εξάγονται από το ιστορικό των 180 ημερών πριν από το acceptance snapshot. Η χρήση σταθερού χρονικού παραθύρου ως προς την ημερομηνία αποδοχής επιτρέπει τη σύγκριση repositories που έγιναν δεκτά σε διαφορετικά ημερολογιακά έτη, χωρίς η μέτρηση να εξαρτάται από διαφορετική διάρκεια παρατήρησης.

Ο αριθμός των commits χρησιμοποιείται ως ένδειξη της έντασης της development activity. Δεν αντιμετωπίζεται, όμως, ως άμεση μέτρηση παραγωγικότητας. Ο αριθμός των commits επηρεάζεται έντονα από τις πρακτικές κάθε ομάδας. Ένας developer μπορεί να πραγματοποιεί πολλά μικρά commits, ενώ μία άλλη ομάδα μπορεί να χρησιμοποιεί squash merging και να καταγράφει το ίδιο development effort σε ελάχιστα commits. Κατά συνέπεια, δύο repositories με διαφορετικό αριθμό commits δεν μπορούν να ταξινομηθούν αξιόπιστα ως περισσότερο ή λιγότερο παραγωγικά αποκλειστικά με βάση αυτή τη μεταβλητή.

Η ίδια επιφύλαξη ισχύει για τις γραμμές που προστέθηκαν. Μεγαλύτερος αριθμός added lines δεν σημαίνει κατ' ανάγκη μεγαλύτερη παραγωγικότητα ή καλύτερο λογισμικό. Η αύξηση μπορεί να αντανakλά νέα functionality, αλλά μπορεί επίσης να προκύπτει από duplication, generated code, refactoring ή απλώς από μεγαλύτερο score του project. Επιπλέον, ο μικρότερος όγκος κώδικα δεν αποτελεί απαραίτητα αρνητικό αποτέλεσμα: μία πιο συμπαγής υλοποίηση μπορεί να επιλύει το ίδιο πρόβλημα με λιγότερο code.

Για τον λόγο αυτό, οι μεταβλητές development activity χρησιμοποιούνται ως περιγραφικό πλαίσιο και όχι ως evidence ότι οι AI coding assistants αύξησαν ή μείωσαν την παραγωγικότητα. Η διάκριση αυτή είναι σημαντική, επειδή η παραγωγικότητα αποτελεί latent construct που δεν μπορεί να εξαχθεί αξιόπιστα από το repository history χωρίς πληροφορίες για τον χρόνο εργασίας, τον αριθμό των developers, τις διαδικασίες review και το score των αλλαγών.

3.6.5.2 Pylint Violations ανά KLOC

Η πυκνότητα **Pylint violations/KLOC** χρησιμοποιείται ως proxy της συμμόρφωσης του κώδικα με ένα σταθερό σύνολο κανόνων στατικής ανάλυσης. Η κανονικοποίηση ανά χίλιες logical lines of code είναι απαραίτητη, επειδή ένας απόλυτος αριθμός violations θα ήταν σχεδόν αναπόφευκτα μεγαλύτερος σε μεγαλύτερα repositories. Με την κανονικοποίηση επιχειρείται να συγκριθεί η πυκνότητα προβλημάτων και όχι απλώς ο συνολικός όγκος τους.

Η συγκεκριμένη metric είναι ιδιαίτερα χρήσιμη για την παρούσα εργασία επειδή οι παραβιάσεις που ανιχνεύονται είναι σε σημαντικό βαθμό **tool-addressable**. Δηλαδή, ένα linter, IDE ή AI assistant μπορεί να εντοπίσει ή να διορθώσει πολλά από αυτά τα προβλήματα χωρίς να απαιτείται συνολική κατανόηση της αρχιτεκτονικής του συστήματος. Επομένως, η metric βρίσκεται κοντά στο επίπεδο στο οποίο θα μπορούσε θεωρητικά να εμφανιστεί άμεσα το αποτέλεσμα μεγαλύτερης αυτοματοποίησης της ανάπτυξης.

Ταυτόχρονα, η χαμηλή violation density δεν ισοδυναμεί με υψηλή συνολική ποιότητα. Ένα πρόγραμμα μπορεί να συμμορφώνεται πλήρως με τους κανόνες του Pylint και παρ' όλα αυτά να διαθέτει ακατάλληλη αρχιτεκτονική, ανεπαρκείς δοκιμές ή ακόμη και λανθασμένη επιστημονική λογική. Συνεπώς, η metric ερμηνεύεται ως **rule conformance** και όχι ως άμεση μέτρηση correctness ή software quality στο σύνολό της.

3.6.5.3 Debt Ratio

Το **debt ratio** χρησιμοποιείται ως συνοπτικό proxy του code-level technical debt, ακολουθώντας την παράδοση της προσέγγισης SQALE που χρησιμοποιείται ευρέως στη βιβλιογραφία τεχνικού χρέους [19]. Η λογική του είναι ότι οι παραβιάσεις του κώδικα μπορούν να μεταφραστούν σε ένα κατά προσέγγιση remediation cost και να συσχετιστούν με το συνολικό μέγεθος της βάσης κώδικα.

Στο συγκεκριμένο pipeline, η metric αποτελεί προσέγγιση της λογικής SQALE και όχι πλήρη αναπαραγωγή εμπορικής υλοποίησης. Για τον λόγο αυτό δεν ερμηνεύεται ως αντικειμενικός χρηματικός ή χρονικός υπολογισμός του πραγματικού κόστους αποπληρωμής του technical debt.

Επιπλέον, το debt ratio και η πυκνότητα Pylint violations δεν είναι πλήρως ανεξάρτητες μετρικές, επειδή μέρος του υπολογισμού του debt ratio προέρχεται από violations που ανιχνεύονται από τον static-analysis layer. Συνεπώς, εάν και οι δύο μεταβλητές παρουσιάζουν αλλαγή προς την ίδια κατεύθυνση, το αποτέλεσμα δεν πρέπει να ερμηνεύεται ως δύο πλήρως ανεξάρτητες επιβεβαιώσεις του ίδιου φαινομένου. Αντίθετα, αποτελούν δύο σχετικές οπτικές της ίδιας ευρύτερης διάστασης: της συμμόρφωσης του κώδικα με ανιχνεύσιμους κανόνες ποιότητας.

3.6.5.4 Cyclomatic Complexity: Mean και Maximum

Η κυκλοματική πολυπλοκότητα χρησιμοποιείται για την αποτύπωση της πολυπλοκότητας της control-flow structure. Με απλοποιημένους όρους, αυξάνεται καθώς αυξάνονται οι ανεξάρτητες διαδρομές εκτέλεσης που δημιουργούνται από branches και άλλες δομές ελέγχου.

Στην παρούσα μελέτη χρησιμοποιούνται δύο διαφορετικές εκδοχές της metric: η μέση και η μέγιστη κυκλοματική πολυπλοκότητα. Η μέση τιμή αποτυπώνει τη γενικότερη δομική κατάσταση του repository. Αντίθετα, η μέγιστη τιμή λειτουργεί ως worst-case indicator και επιτρέπει τον εντοπισμό έργων που περιέχουν τουλάχιστον μία ιδιαίτερα σύνθετη μονάδα κώδικα.

Η διάκριση είναι σημαντική επειδή δύο repositories μπορούν να έχουν παρόμοιο μέσο όρο αλλά διαφορετική κατανομή. Ένα έργο μπορεί να περιλαμβάνει πολλές μέτρια σύνθετες functions, ενώ ένα δεύτερο μπορεί να είναι γενικά απλό αλλά να περιέχει μία εξαιρετικά σύνθετη function. Η χρήση και των δύο μεταβλητών περιορίζει τον κίνδυνο απώλειας αυτής της πληροφορίας.

Παρόλα αυτά, η cyclomatic complexity αποτελεί structural και όχι semantic metric. Δεν μπορεί να προσδιορίσει αν η πολυπλοκότητα είναι δικαιολογημένη από το επιστημονικό πρόβλημα ούτε αν το πρόγραμμα παράγει σωστά αποτελέσματα. Μία αριθμητικά λανθασμένη συνάρτηση μπορεί να έχει

εξαιρετικά χαμηλή πολυπλοκότητα, ενώ ένας σωστός επιστημονικός αλγόριθμος μπορεί να απαιτεί εγγενώς πιο περίπλοκη control flow.

3.6.5.5 Maximum Nesting Depth

Το **maximum nesting depth** καταγράφει το μεγαλύτερο επίπεδο εμφώλευσης δομών ελέγχου στο repository. Βαθιά εμφωλευμένες δομές μπορούν να δυσκολέψουν την κατανόηση του κώδικα, επειδή ο αναγνώστης πρέπει να διατηρεί ταυτόχρονα πολλαπλές συνθήκες και επίπεδα control flow.

Η χρήση της μέγιστης και όχι της μέσης τιμής είναι σκόπιμη. Ο στόχος είναι να καταγραφεί το worst structural case, δηλαδή το αν υπάρχει κάποιο τμήμα του repository στο οποίο η control-flow structure γίνεται ιδιαίτερα δύσκολη στην παρακολούθηση.

Η metric, ωστόσο, είναι ευαίσθητη στο τοπικό design. Μία μόνο function μπορεί να καθορίσει την τιμή ολόκληρου του repository. Δεν πρέπει επομένως να ερμηνεύεται ως περιγραφή της συνολικής αρχιτεκτονικής του project. Αποτελεί συγκεκριμένο indicator structural complexity, ο οποίος συμπληρώνει και δεν αντικαθιστά την cyclomatic complexity.

3.6.5.6 Maintainability Index

Ο **Maintainability Index** συνδυάζει χαρακτηριστικά του κώδικα ώστε να παρέχει έναν συνοπτικό indicator συντηρησιμότητας. Στην παρούσα εργασία χρησιμοποιείται η ελάχιστη τιμή που παρατηρείται στο repository, με στόχο να αποτυπωθεί το χειρότερο σημείο της βάσης κώδικα και όχι ένας μέσος όρος που θα μπορούσε να αποκρύψει ένα ιδιαίτερα προβληματικό αρχείο.

Αυτή η επιλογή έχει όμως συγκεκριμένη συνέπεια. Όσο μεγαλύτερο είναι ένα repository και όσο περισσότερα αρχεία περιέχει, τόσο περισσότερες είναι οι ευκαιρίες να υπάρχει τουλάχιστον ένα αρχείο με ιδιαίτερα χαμηλό maintainability index. Επομένως, η minimum metric είναι εκ φύσεως πιθανό να σχετίζεται με repository size. Αυτός είναι ένας από τους λόγους για τους οποίους το μέγεθος καταγράφεται ως covariate και εξετάζεται ρητά στην ανάλυση confounding.

Επιπλέον, ο Maintainability Index αποτελεί composite metric και συνεπώς δεν πρέπει να αντιμετωπίζεται ως απευθείας observation της ανθρώπινης προσπάθειας συντήρησης. Μία χαμηλή τιμή σημαίνει ότι συγκεκριμένα structural characteristics του κώδικα είναι δυσμενή σύμφωνα με το instrument· δεν σημαίνει ότι έχει μετρηθεί ο πραγματικός χρόνος ή το κόστος που θα απαιτούνταν για μελλοντική maintenance.

3.6.5.7 Duplication Ratio

Το **duplication ratio** επιχειρησιακοποιεί την επανάληψη κώδικα μέσω clone-window detection. Η επανάληψη έχει ενδιαφέρον για το technical debt επειδή η ίδια ή πολύ παρόμοια λογική σε πολλαπλά σημεία αυξάνει την πιθανότητα ασυνεπών μελλοντικών αλλαγών και μπορεί να υποδεικνύει απουσία κατάλληλης abstraction.

Ωστόσο, η duplication metric αποτελεί proxy και όχι semantic clone detector. Η ομοιότητα συγκεκριμένων παραθύρων κώδικα δεν σημαίνει κατ' ανάγκη ότι τα δύο τμήματα έπρεπε να έχουν ενοποιηθεί. Σε scientific software, επαναλαμβανόμενες δομές μπορεί να είναι σκόπιμες, για παράδειγμα όταν παρόμοια numerical procedures εφαρμόζονται σε διαφορετικά experimental configurations.

Επιπλέον, όσο μεγαλύτερος είναι ο συνολικός όγκος κώδικα, τόσο αυξάνεται ο αριθμός των πιθανών παραθύρων που μπορούν να συγκριθούν και, συνεπώς, η πιθανότητα εντοπισμού duplication. Για τον

λόγο αυτό η metric πρέπει να εξετάζεται σε συνδυασμό με repository size και όχι ως πλήρως ανεξάρτητη ένδειξη deterioration ή improvement.

3.6.5.8 Self-Admitted Technical Debt

Η **SATD density** υπολογίζεται από την παρουσία συγκεκριμένων markers σε comments, όπως TODO, FIXME, HACK, XXX, workaround και temporary, και κανονικοποιείται ανά KLOC. Η metric διαφέρει θεμελιωδώς από τις υπόλοιπες Tier 2 μεταβλητές.

Το SATD δεν επιχειρεί να μετρήσει όλο το technical debt. Μετρά το χρέος που ο **developer** αναγνωρίζει και καταγράφει ρητά. Ένα repository μπορεί να διαθέτει σημαντικό technical debt χωρίς κανένα SATD marker, είτε επειδή οι developers δεν το έχουν αναγνωρίσει είτε επειδή χρησιμοποιούν διαφορετικές πρακτικές τεκμηρίωσης.

Αυτή η ιδιότητα είναι ιδιαίτερα σημαντική για το RQ4. Μείωση του SATD μπορεί να σημαίνει πραγματική μείωση προβλημάτων. Μπορεί όμως επίσης να σημαίνει μείωση της αναγνώρισης ή της ρητής καταγραφής τους. Για τον λόγο αυτό, το SATD δεν ερμηνεύεται ανεξάρτητα. Εξετάζεται σε συνδυασμό με τις υπόλοιπες μετρικές technical debt. Μείωση SATD μαζί με γενικευμένη μείωση structural debt είναι συμβατή με πραγματική improvement, ενώ μείωση SATD χωρίς αντίστοιχη βελτίωση των άλλων structural indicators δημιουργεί διαφορετική ερμηνευτική εικόνα.

Επιπλέον, το fixed marker vocabulary αποτελεί περιορισμό construct validity. Οι developers μπορεί να χρησιμοποιούν διαφορετικές λέξεις ή conventions για την καταγραφή προβλημάτων, και η σχετική γλωσσική πρακτική μπορεί να μεταβάλλεται μεταξύ χρονικών περιόδων. Επομένως, η metric αποτυπώνει ένα συγκεκριμένο vocabulary αναγνώρισης και όχι κάθε πιθανή μορφή self-admitted debt.

3.6.5.9 Docstring Coverage

Η **docstring coverage** μετρά το ποσοστό των documentable objects που διαθέτουν docstring. Χρησιμοποιείται ως proxy της συστηματικότητας της API-level τεκμηρίωσης.

Το πλεονέκτημα της metric είναι ότι μπορεί να εφαρμοστεί αντικειμενικά και με τον ίδιο τρόπο σε μεγάλο αριθμό repositories. Το βασικό της μειονέκτημα είναι ότι μετρά παρουσία και όχι ποιότητα. Ένα docstring μπορεί να υπάρχει αλλά να είναι ελλιπές, ανακριβές ή απλώς να επαναλαμβάνει το όνομα της function χωρίς να εξηγεί τη συμπεριφορά της.

Επομένως, αύξηση του coverage δεν θα αρκούσε από μόνη της για να αποδειχθεί καλύτερη επιστημονική τεκμηρίωση. Αντίστοιχα, σταθερό coverage δεν σημαίνει ότι το περιεχόμενο των docstrings παρέμεινε ποιοτικά ίδιο. Η metric περιορίζεται σε μία μετρήσιμη structural διάσταση της documentation.

3.6.5.10 Comment Density

Η **comment density** αποτυπώνει την αναλογία σχολίων σε σχέση με το μέγεθος του πηγαίου κώδικα. Χρησιμοποιείται συμπληρωματικά προς τη docstring coverage, επειδή η τεκμηρίωση δεν εμφανίζεται αποκλειστικά σε formal docstrings. Σε scientific software, σημαντικές πληροφορίες σχετικά με numerical assumptions, approximations ή implementation decisions μπορεί να εμφανίζονται ως inline comments.

Ωστόσο, μεγαλύτερη comment density δεν σημαίνει αυτόματα καλύτερο documentation. Υπερβολικά σχόλια μπορεί να είναι περιττά, stale ή να περιγράφουν αυτονόητες λειτουργίες. Αντίστροφα, καθαρός και καλά δομημένος κώδικας μπορεί να απαιτεί λιγότερα comments. Για τον λόγο αυτό, η metric

χρησιμοποιείται ως quantitative indicator και όχι ως direct evaluation της πληροφοριακής ποιότητας του documentation.

Η χρήση δύο διαφορετικών documentation metrics —docstring coverage και comment density— μειώνει τον κίνδυνο να ταυτιστεί η τεκμηρίωση αποκλειστικά με μία μορφή γραπτής πληροφορίας, αλλά ούτε ο συνδυασμός τους καλύπτει πλήρως το construct της documentation quality.

3.6.5.11 Repository-Level Reproducibility Score

Η αναπαραγωγικότητα αποτελεί ιδιαίτερα δύσκολο construct για automated repository mining. Ένας πλήρης έλεγχος θα απαιτούσε την πραγματική εγκατάσταση κάθε project, την ανακατασκευή του κατάλληλου historical environment, την εκτέλεση της επιστημονικής διαδικασίας και τη σύγκριση του παραγόμενου αποτελέσματος με δημοσιευμένο ground truth. Μια τέτοια διαδικασία δεν είναι εφικτή με ενιαίο τρόπο για αυθαίρετα repositories διαφορετικών επιστημονικών domains.

Για τον λόγο αυτό, η μελέτη χρησιμοποιεί ένα **repository-level reproducibility score** που αποτυπώνει συγκεκριμένες τεχνικές προϋποθέσεις της αναπαραγωγικότητας και όχι την πλήρη επιστημονική reproduction. Ο checker έχει σχεδιαστεί γύρω από χαρακτηριστικά όπως η declaration των dependencies, η δυνατότητα εκτέλεσης και η αποφυγή configuration που εξαρτάται από συγκεκριμένο τοπικό environment.

Στην τελική mining run οι installation/execution-related έλεγχοι που θα ήταν ιδιαίτερα ευαίσθητοι στη χρονική ηλικία των dependencies δεν χρησιμοποιούνται ως ισότιμη βάση σύγκρισης των cohorts. Η εκτέλεση παλαιότερου software σε σύγχρονο Python environment θα μπορούσε διαφορετικά να μετατρέψει το software ageing ή dependency bit rot σε φαινομενική διαφορά reproducibility μεταξύ Pre και Post.

Κατά συνέπεια, η συγκεκριμένη μεταβλητή πρέπει να ερμηνεύεται ως περιορισμένος **repository-level proxy**. Δεν αποδεικνύει ότι τα δημοσιευμένα επιστημονικά αποτελέσματα μπορούν να αναπαραχθούν end-to-end και δεν αξιολογεί την numerical correctness του λογισμικού.

3.6.5.12 Τι Δεν Μετρούν οι Δεκατρείς Μεταβλητές

Ο σημαντικότερος κοινός περιορισμός του measurement framework είναι ότι καμία από τις δεκατρείς μεταβλητές δεν αποτελεί **correctness oracle**. Ο static analyser μπορεί να εξετάσει complexity, violations, duplication και άλλες structural ιδιότητες, αλλά δεν γνωρίζει ποιο επιστημονικό αποτέλεσμα θα έπρεπε να παράγει το πρόγραμμα. Αντίστοιχα, η documentation και η reproducibility μπορούν να αξιολογηθούν μόνο μέσω συγκεκριμένων observable proxies.

Η validation study έχει σχεδιαστεί ακριβώς ώστε αυτό το όριο να μην παραμένει θεωρητική υπόθεση. Στα synthetic fixtures, numerical correctness defects μπορούν να παραμένουν αόρατα στις structural metrics, ενώ εντοπίζονται μόνο όταν υπάρχει explicit analytical oracle. Στα mined repositories δεν υπάρχει αντίστοιχο ground truth, επειδή κάθε repository επιλύει διαφορετικό επιστημονικό πρόβλημα.

Επομένως, όταν η παρούσα εργασία αναφέρεται σε «βελτίωση της ποιότητας», η διατύπωση περιορίζεται αυστηρά στις μετρούμενες ιδιότητες: rule conformance, structural complexity, duplication, acknowledged technical debt, documentation και repository-level reproducibility. Δεν υποστηρίζεται ότι ένα repository με καλύτερες τιμές στις συγκεκριμένες μετρικές είναι κατ' ανάγκη επιστημονικά ορθότερο.

3.6.5.13 Πολυδιάστατη Ερμηνεία αντί Μίας Συνολικής Βαθμολογίας

Για τον ίδιο λόγο, οι δεκατρείς μεταβλητές δεν συνδυάζονται σε έναν ενιαίο «quality score». Μία τέτοια σύνθεση θα απαιτούσε αυθαίρετες επιλογές σχετικά με το βάρος κάθε διάστασης. Δεν υπάρχει εμπειρική βάση για να υποστηριχθεί, για παράδειγμα, ότι μία μονάδα βελτίωσης στην cyclomatic complexity έχει την ίδια σημασία με μία αντίστοιχη μεταβολή στη docstring coverage ή στην reproducibility.

Η παρούσα μελέτη προτιμά επομένως μία πολυδιάστατη ερμηνεία. Κάθε metric αναλύεται ξεχωριστά και στη συνέχεια εξετάζεται αν οι κατευθύνσεις των effects σχηματίζουν ένα συνεκτικό συνολικό pattern. Η προσέγγιση αυτή συνδέεται άμεσα με το RQ4: η υπόθεση της γεφύρωσης δεν απαιτεί απλώς ένα χαμηλότερο aggregate score, αλλά βελτίωση σε πολλαπλές διαφορετικές διαστάσεις. Αντίστοιχα, η υπόθεση της απόκρυψης αφορά την πιθανότητα να βελτιώνονται οι πιο εύκολα αυτοματοποιήσιμες ιδιότητες χωρίς αντίστοιχη αλλαγή στις ιδιότητες που απαιτούν βαθύτερη ανθρώπινη κρίση.

Με αυτόν τον τρόπο, οι περιορισμοί των επιμέρους proxies δεν αγνοούνται αλλά ενσωματώνονται στον ίδιο τον σχεδιασμό της ανάλυσης. Καμία metric δεν αντιμετωπίζεται ως πλήρης απεικόνιση της ποιότητας και κανένα μεμονωμένο αποτέλεσμα δεν αρκεί για την απάντηση του κεντρικού ερευνητικού ερωτήματος.

3.7 Πλαίσιο Μέτρησης

Οι μετρικές της μελέτης οργανώνονται σε τρία επίπεδα (tiers), τα οποία αντιπροσωπεύουν διαφορετικές διαστάσεις της ανάλυσης. Το πρώτο επίπεδο περιλαμβάνει χαρακτηριστικά της δραστηριότητας ανάπτυξης που χρησιμοποιούνται κυρίως ως περιγραφικό πλαίσιο και ως πιθανοί συγχυτικοί παράγοντες. Το δεύτερο επίπεδο περιλαμβάνει μετρικές τεχνικού χρέους σε επίπεδο κώδικα, στηριζόμενο στην καθιερωμένη βιβλιογραφία για την αναγνώριση και μέτρηση του technical debt [19]. Το τρίτο επίπεδο αφορά ιδιότητες που συνδέονται ειδικότερα με την ποιότητα του ερευνητικού λογισμικού, όπως η τεκμηρίωση και η αναπαραγωγικότητα [9], [10], [12]. **Ο Πίνακας 3.3** παρουσιάζει τις βασικές κατηγορίες μετρικών και τον τρόπο επιχειρησιακοποίησής τους. Ορισμένες κατηγορίες αποδίδονται μέσω περισσότερων από μίας επιμέρους μεταβλητών και, για τον λόγο αυτό, ο αριθμός των κατηγοριών δεν ταυτίζεται απαραίτητα με τον αριθμό των επιμέρους μεταβλητών που παρουσιάζονται στα αποτελέσματα.

Πίνακας 3.3: Πλαίσιο μέτρησης: έντεκα κατηγορίες μετρικών σε τρία tiers.

ID	Metric	Definition	Source
Tier 1 – Development activity (context for RQ1–RQ3)			
M1	Commit activity	Commits during the 180 days before acceptance	History miner
M2	Churn	(Lines added + deleted) / final LLOC, same window	History miner
M3	Team size	Distinct (hashed) authors in the same window	History miner
Tier 2 – Code-level technical debt (RQ1)			
M4	Debt ratio	Approximate SQALE-tradition-based remediation ratio	Static analyser
M5	Violation density	Pylint violations per KLOC, fixed rule set	Static analyser
M6	Structural complexity and maintainability	Mean and maximum cyclomatic complexity, maximum nesting depth, and maintainability index	Static analyser
M7	Duplication	Clone-window duplication ratio	Static analyser
M8	SATD density	Self-admitted debt markers per KLOC	SATD detector
Tier 3 – Research quality (RQ2)			
M9	Reproducibility	Repository-level reproducibility proxy based on four era-neutral components (0–4 score)	Repro checker
M10	Documentation	Docstring coverage; comment density	Docs analyser
M11	Practice adoption	Tests present; CI configured	Covariate extractor

Το Tier 1 αποτελεί context και όχι outcome. Στον εγκαταλειμμένο experimental σχεδιασμό, η productivity ήταν primary outcome. Σε έναν mining σχεδιασμό, η commit activity συγγχεί την productivity με τη συνήθεια, τη διαδικασία της ομάδας και την πολιτική squash-merge. Επομένως, τα M1–M3 αντιμετωπίζονται ως descriptive context και covariates και όχι ως evidence ότι η assistance έκανε κάποιον γρηγορότερο.

Το M8 είναι η βασική metric. Η SATD density είναι η μοναδική Tier 2 metric που μετρά τι γνωρίζουν οι developers για το χρέος τους και όχι το ίδιο το χρέος. Η αλληλεπίδρασή της με τα M4–M7 μεταφέρει το RQ4: η μείωση του SATD μαζί με μείωση του μετρούμενου χρέους είναι συμβατή με bridging· η μείωση του SATD μαζί με σταθερό ή αυξανόμενο measured debt αποτελεί signature concealment, λιγότερο αναγνωρισμένο και όχι λιγότερο χρέος.

Τι χάθηκε με το correctness oracle. Ο experimental σχεδιασμός περιλάμβανε numerical correctness σε σχέση με ground truth. Δεν υπάρχει αντίστοιχο oracle για αυθαίρετο mined code, επομένως το Tier 3 δεν παρατηρεί πλέον άμεσα την correctness. Η validation study παρουσιάζει με ακρίβεια αυτό το blind spot, αριθμητικά λανθασμένος κώδικας μπορεί να είναι αόρατος σε κάθε metric αυτού του πίνακα, και το discussion chapter δεσμεύεται να το σεβαστεί: η μελέτη μπορεί να δείξει πού μετακινήθηκε το χρέος, όχι αν η επιστήμη υπολόγισε σωστά.

3.8 Πλάνο Ανάλυσης

Το πλάνο ανάλυσης καθορίστηκε πριν από την εξέταση των αποτελεσμάτων, ώστε να περιοριστεί ο κίνδυνος επιλεκτικής έμφασης σε μετρικές που παρουσιάζουν εκ των υστέρων ευνοϊκές ή στατιστικά σημαντικές διαφορές. Η μελέτη περιλαμβάνει πολλαπλές μετρικές που καλύπτουν διαφορετικές διαστάσεις του τεχνικού χρέους, της τεκμηρίωσης, της αναπαραγωγιμότητας και της δραστηριότητας ανάπτυξης. Για τον λόγο αυτό, όλες οι προκαθορισμένες συγκρίσεις αναφέρονται ανεξάρτητα από την κατεύθυνση ή τη στατιστική σημαντικότητά τους. Η προσέγγιση αυτή είναι ιδιαίτερα σημαντική στο πλαίσιο της μηχανικής επιστημονικού λογισμικού, καθώς οι Arvanitou et al. [10] έχουν επισημάνει την περιορισμένη αναφορά αρνητικών αποτελεσμάτων στη σχετική βιβλιογραφία.

3.8.1 Βασικές Συγκρίσεις

Για κάθε metric, το Pre έναντι Post (RQ1, RQ2) και το Post-Disclosed έναντι Post-Undisclosed (RQ3) συγκρίνονται με το Mann–Whitney U test, καθώς τα cohorts αποτελούν ανεξάρτητα δείγματα και οι metrics είναι bounded, skewed ή ordinal, χωρίς να θεωρείται δεδομένη η κανονικότητα. Τα effect sizes αναφέρονται ως Cliff's δ , μαζί με τη διάμεσο και το interquartile range (IQR) κάθε ομάδας. Τα effect sizes παρουσιάζονται για κάθε προκαθορισμένη σύγκριση ανεξάρτητα από τη στατιστική σημαντικότητα, ώστε η ερμηνεία να λαμβάνει υπόψη τόσο το μέγεθος όσο και την κατεύθυνση της παρατηρούμενης διαφοράς και να μην βασίζεται αποκλειστικά στις τιμές p .

Επειδή πραγματοποιούνται πολλαπλές συγκρίσεις μεταξύ των cohorts, οι αρχικές τιμές p προσαρμόζονται με τη μέθοδο Benjamini–Hochberg, με στόχο τον περιορισμό του false discovery rate. Ως στατιστικά σημαντικά θεωρούνται τα αποτελέσματα για τα οποία η προσαρμοσμένη τιμή p (p_{adj}) είναι μικρότερη από 0,05. Παράλληλα, η ερμηνεία των αποτελεσμάτων δεν βασίζεται αποκλειστικά στη στατιστική σημαντικότητα. Για κάθε σύγκριση αναφέρονται επίσης το effect size μέσω του Cliff's δ , καθώς και οι διάμεσοι και τα interquartile ranges των δύο ομάδων. Με τον τρόπο αυτό, τόσο τα στατιστικά σημαντικά όσο και τα μη σημαντικά αποτελέσματα αξιολογούνται ως προς το μέγεθος και την κατεύθυνση της παρατηρούμενης διαφοράς.

3.8.2 Προκαθορισμένες Προβλέψεις Pattern

Το RQ4 απαντάται με βάση predictions που έχουν καθοριστεί εδώ, πριν από τη μέτρηση:

P1. Bridging signature. Το Post βελτιώνεται στο Tier 2 (M4–M7 μειώνονται) και στο Tier 3 (M9, M10 αυξάνονται), ενώ το SATD (M8) μπορεί να κινηθεί προς οποιαδήποτε κατεύθυνση· οι βελτιώσεις ενισχύονται και δεν εξασθενούν στο Post-Disclosed.

P2. Concealment signature. Το Post βελτιώνεται σε style-adjacent metrics (M5, το coverage component του M10), ενώ το M9 παραμένει στάσιμο ή μειώνεται και το M8 μειώνεται σε σχέση με σταθερά ή αυξανόμενα M4/M6/M7. Η απόκλιση ενισχύεται στο Post-Disclosed.

P3. Null signature. Οι διαφορές μεταξύ των εποχών είναι μικρές ($|\delta| < 0.15$) σε όλα τα tiers και η αντίθεση disclosed/undisclosed είναι ακόμη μικρότερη, μια ερμηνεία σύμφωνα με την οποία είτε η υιοθέτηση είναι πολύ περιορισμένη σε αυτόν τον πληθυσμό είτε τα instruments είναι υπερβολικά coarse, και η διπλωματική εργασία το αναφέρει αντίστοιχα.

Οι προβλέψεις είναι σκόπιμα falsifiable και προς τις δύο κατευθύνσεις, και το discussion chapter δεσμεύεται να αναφέρει με ποιο signature μοιάζουν περισσότερο τα δεδομένα, συμπεριλαμβανομένου και του τρίτου.

3.8.3 Εξέταση Confounding

Για την εξέταση πιθανών συγχυτικών παραγόντων υπολογίζονται Spearman correlations μεταξύ των κύριων metrics και των repository-level covariates που περιγράφονται στην Ενότητα 3.6.4. Συσχετίσεις με απόλυτη τιμή $|\rho| > 0,4$ επισημαίνονται ως δυνητικά σημαντικές για την ερμηνεία των αποτελεσμάτων. Ιδιαίτερη προσοχή δίνεται σε περιπτώσεις όπου μία metric σχετίζεται με covariate που διαφέρει μεταξύ των Pre και Post cohorts, καθώς μέρος της παρατηρούμενης διαφοράς μεταξύ των εποχών μπορεί να συνδέεται με τον συγκεκριμένο παράγοντα και όχι με την εποχή αυτή καθαυτή. Οι συσχετίσεις αυτές χρησιμοποιούνται επομένως ως έλεγχος ευαισθησίας και ως πλαίσιο για την ερμηνεία των κύριων συγκρίσεων και όχι ως βάση αιτιώδους προσαρμογής.

3.8.4 Descriptive Reporting

Πέρα από τις κύριες στατιστικές συγκρίσεις, το κεφάλαιο αποτελεσμάτων παρουσιάζει περιγραφικά στοιχεία που είναι απαραίτητα για την ερμηνεία του corpus και των αναλύσεων. Ειδικότερα, παρουσιάζεται η πλήρης attrition flow από το αρχικό σύνολο των JOSS submissions έως το τελικό analytical dataset, ο τρόπος επίλυσης των acceptance-time snapshots και η συχνότητα των fallback rules, καθώς και το πλήθος και το ποσοστό των Post repositories στα οποία εντοπίστηκε επιβεβαιωμένη δήλωση χρήσης AI assistance. Τα στοιχεία αυτά χρησιμοποιούνται ως περιγραφικό πλαίσιο για την αξιολόγηση της κατασκευής του corpus, της robustness των snapshots και των περιορισμών του disclosure-based attribution. Αναλύσεις πέρα από το προκαθορισμένο πλάνο χαρακτηρίζονται ως exploratory.

3.9 Επικύρωση των Εργαλείων

Πριν μετρηθεί οποιοδήποτε repository, το measurement pipeline επικυρώθηκε σε synthetic fixtures με seeded και καταγεγραμμένα σε manifest defects, ένα instrument-validation βήμα σύμφωνα με την παράδοση της mutation analysis, σχεδιασμένο ώστε τα detection characteristics του pipeline να αποτελούν καθορισμένα δεδομένα και όχι υποθέσεις κατά την ερμηνεία των mining results.

Το validation harness αποτελείται από reference implementations δύο scientific programs (έναν Runge–Kutta integrator με closed-form analytical solution και ένα least-squares calibration script), από

τα οποία ένας generator παράγει mutant variants εισάγοντας defects από έναν κατάλογο δεκαπέντε τύπων που καλύπτει code quality (long methods, duplication, magic numbers, deep nesting, global state), numerical correctness (sign errors, coefficient perturbations, integrator-order degradation), reproducibility (hardcoded paths, missing dependency declarations, non-determinism) και documentation (stripped docstrings, stale comments).

Κάθε injection καταγράφεται σε machine-readable manifest· το manifest και όχι η μνήμη αποτελεί το ground truth. Τα structural mutants κατασκευάζονται ώστε να είναι αριθμητικά bit-identical με τα references τους, ενώ τα correctness mutants είναι structurally clean, ώστε το cross-tier leakage, ένα correctness defect που επισημαίνεται από ένα style metric, να μπορεί να εντοπιστεί ως fixture error και όχι να απορροφηθεί ως noise.

Το harness εκτελεί ολόκληρο το pipeline σε όλα τα fixtures και αναφέρει, για κάθε defect type: detection rate, false-positive rate σε clean references, το detecting metric και το effect size του metric delta. Δύο ευρήματα της validation έχουν άμεση σημασία για την ερμηνεία των mining results. Πρώτον, η calibration του static analyser, ένα documented σύνολο pylint rules που απενεργοποιήθηκαν επειδή επισημαίνουν legitimate scientific idioms, μειώνει τα clean-reference violations στο μηδέν, και κάθε αναφερόμενο false-positive rate εξαρτάται από αυτή τη configuration. Δεύτερον, τα correctness defects είναι αόρατα στο static tier και μπορούν να εντοπιστούν μόνο από το correctness oracle, η εμπειρική απόδειξη ότι οι metrics της mining study, οι οποίες αναγκαστικά αποκλείουν το oracle, δεν παρατηρούν την correctness.

3.10 Ηθικά Ζητήματα

Η μελέτη επεξεργάζεται μόνο δημόσια artefacts: open-source repositories, το δημόσιο ιστορικό τους και τα metadata ενός open-access journal. Δεν συμμετέχουν άτομα και δεν προσπελούνται private data. Παρ' όλα αυτά, εφαρμόζονται τρεις πρακτικές.

Author privacy. Η authorship στα commits αποτελεί προσωπικό δεδομένο ακόμη και όταν είναι δημόσια. Τα author identifiers γίνονται salted-hashed κατά την εξαγωγή (Listing 2)· κανένα όνομα ή email δεν εμφανίζεται στο dataset, στο replication package ή στη διπλωματική εργασία και δεν πραγματοποιείται κανενός είδους per-person analysis. Η μονάδα ανάλυσης παραμένει σε όλη τη μελέτη το repository.

No repository-level shaming. Τα αποτελέσματα αναφέρονται σε aggregate μορφή. Όταν χρειάζεται να αναφερθεί ένα συγκεκριμένο repository, για παράδειγμα ως disclosure example, παρατίθεται μόνο για πρακτικές που οι ίδιοι οι authors έχουν δημοσιοποιήσει και κανένα repository δεν αναφέρεται σε σχέση με unfavourable metric.

Infrastructure courtesy. Το cloning και το API access σέβονται τα rate limits και χρησιμοποιούν caching επιθετικά· κάθε repository γίνεται fetch μία φορά. Το replication package περιλαμβάνει το derived dataset και το corpus specification (identifiers, snapshots, seed) αντί για redistributed copies των ίδιων των repositories, σεβόμενο τις upstream licences.

Κεφάλαιο 4ο: Αποτελέσματα

Το κεφάλαιο αυτό παρουσιάζει τα αποτελέσματα της μελέτης εξόρυξης δεδομένων. Η δομή ακολουθεί τα ερευνητικά ερωτήματα. Η Ενότητα 4.1 εξηγεί πώς πρέπει να διαβάζονται οι αριθμοί στους πίνακες αποτελεσμάτων. Οι Ενότητες 4.2 έως 4.4 απαντούν στα RQ1, RQ2 και RQ3 αντίστοιχα. Η Ενότητα 4.5 παρουσιάζει τους ελέγχους για συγχυτικούς παράγοντες, ενώ η Ενότητα 4.6 συνδυάζει όλα τα ευρήματα για να απαντήσει στο RQ4.

Μία βασική αρχή από το σχέδιο ανάλυσης που εφαρμόζεται σε όλο το κεφάλαιο είναι ότι όλες οι προγραμματισμένες συγκρίσεις παρουσιάζονται, συμπεριλαμβανομένων εκείνων που δεν δείχνουν κάποια διαφορά. Ένα αποτέλεσμα που δείχνει «καμία αλλαγή» εξακολουθεί να αποτελεί αποτέλεσμα.

4.1 Πώς Διαβάζονται οι Πίνακες Αποτελεσμάτων

Κάθε σύγκριση σε αυτό το κεφάλαιο χρησιμοποιεί τα ίδια τρία στατιστικά μεγέθη, επομένως αυτά εξηγούνται μία φορά εδώ.

Διάμεσος και IQR. Για κάθε μετρική, ο Πίνακας 4.1 παρουσιάζει τη διάμεσο και το interquartile range (IQR) κάθε cohort. Η διάμεσος χρησιμοποιείται αντί του μέσου όρου επειδή οι περισσότερες από αυτές τις μετρικές έχουν ασύμμετρη κατανομή και ορισμένα πολύ μεγάλα αποθετήρια θα μπορούσαν να επηρεάσουν σημαντικά τον μέσο όρο. Το IQR περιγράφει τη διασπορά του κεντρικού 50% των παρατηρήσεων και παρουσιάζεται σε παρένθεση μετά τη διάμεσο.

Έλεγχος Mann–Whitney U. Έλεγχος Mann–Whitney U. Ο έλεγχος αυτός εξετάζει αν οι κατανομές των δύο ανεξάρτητων cohorts διαφέρουν συστηματικά. Επειδή στις κύριες συγκρίσεις εξετάζονται δεκατρείς επιμέρους μεταβλητές, οι αρχικές τιμές p προσαρμόζονται με τη μέθοδο Benjamini–Hochberg (p_{adj}), ώστε να περιοριστεί το false discovery rate. Ένα αποτέλεσμα θεωρείται στατιστικά σημαντικό μόνο όταν $p_{adj} < 0,05$.

Cliff's delta (δ). Η τιμή p δείχνει αν μια διαφορά είναι στατιστικά αξιόπιστη, ενώ το Cliff's δ περιγράφει το μέγεθος και την κατεύθυνσή της. Η τιμή του κυμαίνεται από -1 έως $+1$. Στην παρούσα ανάλυση, θετικό δ σημαίνει ότι η τιμή στο Pre cohort ήταν μεγαλύτερη από την αντίστοιχη τιμή στο Post cohort. Ως γενικός οδηγός, $|\delta|$ περίπου 0,15 θεωρείται μικρό effect, περίπου 0,33 μέτριο και πάνω από 0,47 μεγάλο.

4.2 RQ1: Άλλαξε το Τεχνικό Χρέος σε Επίπεδο Κώδικα;

Το RQ1 εξετάζει αν το τεχνικό χρέος σε επίπεδο κώδικα του ερευνητικού λογισμικού διαφέρει στην εποχή της ΑΙ. Ο Πίνακας 4.1 παρουσιάζει την πλήρη σύγκριση για τις δεκατρείς επιμέρους μεταβλητές που περιλαμβάνονται στις κύριες αναλύσεις. Οι μετρικές του Tier 2 (τεχνικό χρέος κώδικα) παρουσιάζονται στο πρώτο τμήμα.

Πίνακας 4.1: Σύγκριση μεταξύ των εποχών για όλες τις μετρικές

Θετικό δ = η τιμή του Pre ήταν μεγαλύτερη. Οι αστερίσκοι δείχνουν προσαρμοσμένη στατιστική σημαντικότητα ($p_{adj} < 0,05$).

Μετρική	Διάμ. Pre (IQR)	Διάμ. Post (IQR)	δ	p	p_{adj}
Tier 2 – Τεχνικό χρέος σε επίπεδο κώδικα					
Debt ratio	0,052 (0,035)	0,042 (0,026)	0,30	0,005	0,034*
Pylint violations / KLOC	209 (191,64)	163 (113,280)	0,34	0,002	0,021*
Duplication ratio	0,35 (0,29)	0,45 (0,33)	-0,26	0,014	0,062
Max. nesting depth	8 (4)	8 (2)	0,19	0,077	0,201
Cyclomatic complexity (mean)	3,36 (1,99)	3,35 (2,17)	-0,03	0,792	0,792
Cyclomatic complexity (max)	28,5 (20)	27 (30)	-0,03	0,786	0,792
Maintainability index (min)	27,3 (24,3)	22,8 (29,3)	0,05	0,637	0,792
SATD / KLOC	0,56 (1,27)	0,23 (0,99)	0,17	0,098	0,213
Tier 3 – Ποιότητα ερευνητικού λογισμικού					
Docstring coverage	0,54 (0,32)	0,53 (0,36)	0,03	0,764	0,792
Comment density	0,17 (0,14)	0,16 (0,16)	0,04	0,692	0,792
Reproducibility score (0–4)	2 (0,25)	2 (0)	-0,13	0,063	0,201
Tier 1 – Δραστηριότητα ανάπτυξης (πλαίσιο)					
Commits (180 d)	97,5 (137,5)	90 (132,5)	0,07	0,536	0,792
Lines added (180 d)	5.640 (18.422)	6.000 (15.463)	-0,07	0,525	0,792

Σημείωση: n = 60 για το Pre cohort και n = 59 για το Post cohort. Το IQR παρουσιάζεται ως πλάτος του ενδοτεταρτημοριακού εύρους (Q3 – Q1).

Σύγκριση κύριων μετρικών μεταξύ Pre και Post cohort

Διάμεσες τιμές ανά cohort (Pre n=60, Post n=59)



Σημείωση: Τα μόνα στατιστικά σημαντικά αποτελέσματα μετά από διόρθωση Benjamini–Hochberg είναι οι μειώσεις στις Pylint violations / KLOC και στο debt ratio. Χαμηλότερες τιμές είναι καλύτερες για όλες τις μετρικές εκτός από το docstring coverage και το reproducibility score, όπου υψηλότερες τιμές είναι καλύτερες.

Σχήμα 4.1: Σύγκριση των κύριων μετρικών μεταξύ των Pre και Post cohorts. Οι τιμές αντιστοιχούν στις διαμέσους των δύο ομάδων. Ο αστερίσκος δηλώνει στατιστική σημαντικότητα μετά από διόρθωση Benjamini–Hochberg ($\text{padj} < 0,05$).

4.2.1 Τι Βελτιώθηκε

Δύο μετρικές παρουσίασαν βελτίωση και είναι τα μόνα στατιστικά σημαντικά αποτελέσματα σε ολόκληρη τη μελέτη.

Οι παραβιάσεις Pylint μειώθηκαν από διάμεση τιμή 209 ανά χίλιες γραμμές στην Pre σε 163 στην Post. Πρόκειται για μέτριο effect ($\delta = 0,34$) και είναι σαφώς στατιστικά σημαντικό ($\text{padj} = 0,021$). Οι παραβιάσεις Pylint μετρούν πόσο συχνά ο κώδικας παραβιάζει τυπικούς κανόνες ύφους και ορθότητας της Python, όπως αχρησιμοποίητες μεταβλητές, κακή ονομασία, έλλειψη συμβάσεων και παρόμοια προβλήματα.

Το debt ratio μειώθηκε από 0,052 σε 0,042, επίσης με μέτριο effect ($\delta = 0,30$, $\text{padj} = 0,034$). Το debt ratio εκτιμά πόση «εργασία επιδιόρθωσης» χρειάζεται ο κώδικας σε σχέση με το μέγεθός του.

Υπάρχει όμως ένα σημαντικό σημείο που πρέπει να αναφερθεί πριν θεωρηθούν αυτά τα δύο αποτελέσματα ως δύο ανεξάρτητα στοιχεία: δεν είναι ανεξάρτητα. Το debt ratio στο συγκεκριμένο

pipeline υπολογίζεται εν μέρει από τις παραβιάσεις Pylint. Επομένως, η πιο σωστή σύνοψη δεν είναι «βελτιώθηκαν δύο μετρικές», αλλά «ένα πράγμα βελτιώθηκε και μετρήθηκε δύο φορές»: ο κώδικας της εποχής ΑΙ ακολουθεί καλύτερα τους τυπικούς κανόνες. Αυτό το είδος ποιότητας θα ονομάζεται **συμμόρφωση με τους κανόνες (rule conformance)** στο υπόλοιπο κεφάλαιο.

4.2.2 Τι Δεν Άλλαξε

Οι δομικές ιδιότητες του κώδικα δεν παρουσίασαν ουσιαστική μεταβολή. Η μέση κυκλοματική πολυπλοκότητα, περίπου ο αριθμός των διαφορετικών διαδρομών αποφάσεων που μπορεί να έχει μια τυπική συνάρτηση, είναι 3,36 στο Pre και 3,35 στο Post. Η μέγιστη πολυπλοκότητα, ο δείκτης maintainability του χειρότερου αρχείου και το μέγιστο βάθος εμφώλευσης είναι όλα στατιστικά μη διακριτά μεταξύ των δύο εποχών (δ μεταξύ $-0,03$ και $0,19$, με όλες τις προσαρμοσμένες τιμές p αρκετά πάνω από $0,05$).

Με απλά λόγια: η δομή του κώδικα, δηλαδή το πόσο περίπλοκη ή «μπερδεμένη» είναι η λογική του, παραμένει ίδια πριν και μετά.

4.2.3 Δύο Τάσεις που Δείχνουν προς την Άλλη Κατεύθυνση

Δύο μετρικές παρουσιάζουν διαφορές που δεν περνούν το όριο στατιστικής σημαντικότητας, αλλά αναφέρονται επειδή αυτό απαιτεί το σχέδιο ανάλυσης και επειδή και οι δύο δείχνουν προς μια κατεύθυνση που αξίζει προσοχής.

Η επανάληψη κώδικα αυξήθηκε. Το duplication ratio αυξήθηκε από $0,35$ σε $0,45$ ($\delta = -0,26$, $\text{padj} = 0,062$). Περισσότερος επαναλαμβανόμενος κώδικας τύπου copy-paste εμφανίζεται στα Post αποθετήρια. Πρόκειται για το τρίτο μεγαλύτερο effect της μελέτης και βρίσκεται λίγο κάτω από το όριο στατιστικής σημαντικότητας. Ωστόσο, η Ενότητα 4.6 δείχνει ότι αυτή η μετρική συσχετίζεται με το μέγεθος του αποθετηρίου, ενώ τα Post αποθετήρια είναι μεγαλύτερα. Επομένως, μέρος αυτής της αύξησης μπορεί απλώς να οφείλεται στο μέγεθος και όχι στην εποχή. Η τάση επομένως παρουσιάζεται ως ενδεικτική και όχι ως τεκμηριωμένη.

Το αυτο-δηλωμένο τεχνικό χρέος μειώθηκε. Το SATD, η πυκνότητα σχολίων όπως TODO, FIXME και παρόμοιων σημείων όπου οι developers αναγνωρίζουν ένα γνωστό πρόβλημα, μειώθηκε περισσότερο από το μισό στη διάμεσο τιμή, από $0,56$ σε $0,23$ δείκτες ανά χίλιες γραμμές ($\delta = 0,17$, $\text{padj} = 0,213$). Από μόνο του, ένας μικρότερος αριθμός TODO θα μπορούσε να σημαίνει ότι υπάρχουν λιγότερα προβλήματα. Σε συνδυασμό όμως με τις σταθερές δομικές μετρικές και την αύξηση της επανάληψης κώδικα, μπορεί να σημαίνει κάτι άλλο: τα προβλήματα εξακολουθούν να υπάρχουν, αλλά οι developers τα αναγνωρίζουν λιγότερο. Η μεθοδολογία είχε προσδιορίσει ακριβώς αυτόν τον συνδυασμό ως πιθανή ένδειξη απόκρυψης (concealment). Επειδή το effect είναι μικρό και μη στατιστικά σημαντικό, το κεφάλαιο αυτό το αντιμετωπίζει αυστηρά ως τάση.

4.2.4 Απάντηση στο RQ1

Το τεχνικό χρέος σε επίπεδο κώδικα άλλαξε, αλλά μόνο με περιορισμένο τρόπο. Ο κώδικας της εποχής ΑΙ ακολουθεί καλύτερα τους τυπικούς κανόνες ύφους και ορθότητας, με μέτριο και στατιστικά σημαντικό effect, ενώ η δομική πολυπλοκότητα παραμένει ίδια και η επανάληψη κώδικα παρουσιάζει χειρότερη τάση. Η βελτίωση βρίσκεται ακριβώς στο επίπεδο του κώδικα που επηρεάζεται άμεσα από αυτοματοποιημένα εργαλεία, τόσο από AI assistants όσο και από σύγχρονα linters και formatters.

4.3 RQ2: Άλλαξε η Ποιότητα του Ερευνητικού Λογισμικού;

Το RQ2 αφορά τις ιδιότητες που κάνουν το ερευνητικό λογισμικό κατάλληλο για ερευνητική χρήση: την τεκμηρίωση και την αναπαραγωγικότητα. Η απάντηση είναι σύντομη: **δεν παρατηρήθηκε ανιχνεύσιμη μεταβολή στις μετρούμενες ιδιότητες τεκμηρίωσης και αναπαραγωγικότητας.**

Η τεκμηρίωση παραμένει σταθερή. Η κάλυψη docstrings, το ποσοστό των συναρτήσεων και κλάσεων που διαθέτουν τεκμηρίωση, είναι 0,54 στο Pre και 0,53 στο Post ($\delta = 0,03$). Η πυκνότητα σχολίων είναι 0,17 έναντι 0,16 ($\delta = 0,04$). Πρόκειται για διαφορές σχεδόν μηδενικές.

Η αναπαραγωγικότητα παραμένει σταθερή, ίσως ελαφρώς χειρότερη. Το score αναπαραγωγικότητας (0 έως 4: δηλωμένες εξαρτήσεις, εκτέλεση χωρίς τροποποιήσεις, ντετερμινιστικότητα και απουσία hardcoded paths) έχει διάμεση τιμή 2 και στα δύο cohorts, με μια μικρή αρνητική τάση ($\delta = -0,13$). Αυτό σημαίνει ότι, αν υπάρχει κάποια κατεύθυνση, τα Post αποθετήρια έχουν ελαφρώς χαμηλότερη επίδοση.

Η αναπαραγωγικότητα επίσης δεν παρουσίασε στατιστικά ανιχνεύσιμη μεταβολή. Η repository-level μεταβλητή αναπαραγωγικότητας έχει διάμεση τιμή 2 και στα δύο cohorts, με μικρό effect ($\delta = -0,13$, $\text{adj} = 0,201$). Το αποτέλεσμα αυτό πρέπει να ερμηνεύεται με βάση τους περιορισμούς που περιγράφηκαν στην Ενότητα 3.6.5.11. Η μεταβλητή αποτυπώνει συγκεκριμένες τεχνικές προϋποθέσεις αναπαραγωγικότητας και όχι πλήρη end-to-end reproduction των επιστημονικών αποτελεσμάτων. Επιπλέον, οι installation/execution-related έλεγχοι που είναι ιδιαίτερα ευαίσθητοι στο dependency ageing δεν αντιμετωπίζονται ως ισοδύναμη βάση σύγκρισης μεταξύ των δύο εποχών. Συνεπώς, το αποτέλεσμα υποστηρίζει μόνο ότι δεν παρατηρήθηκε ανιχνεύσιμη μεταβολή στις μετρούμενες repository-level ιδιότητες αναπαραγωγικότητας.

4.3.1 Απάντηση στο RQ2

Στα 119 peer-reviewed αποθετήρια ερευνητικού λογισμικού, η εποχή που έφερε τους AI coding assistants δεν έφερε κάποια ανιχνεύσιμη βελτίωση στην τεκμηρίωση ή στην αναπαραγωγικότητα. Σύμφωνα με το σχέδιο ανάλυσης, αυτό το μηδενικό αποτέλεσμα αναφέρεται ως εύρημα από μόνο του: οι ιδιότητες που κάνουν το δημοσιευμένο επιστημονικό λογισμικό κατάλληλο για έρευνα δεν παρουσίασαν αλλαγή.

4.4 RQ3: Δηλωμένη έναντι Μη Δηλωμένης Χρήσης AI

Το RQ3 είχε σχεδιαστεί για να συγκρίνει, μέσα στο Post cohort, τα αποθετήρια που δηλώνουν ανοιχτά τη χρήση AI assistance με εκείνα που δεν την αναφέρουν. Η σύγκριση όμως δεν μπορούσε να πραγματοποιηθεί με ουσιαστικό τρόπο, για έναν απλό λόγο: σχεδόν κανείς δεν έκανε σχετική δήλωση.

Από τα 60 αρχικά επιλεγμένα Post repositories, 3 περιείχαν επιβεβαιωμένη δήλωση χρήσης AI-assisted development. Μετά την αποτυχία μέτρησης ενός μη δηλωμένου repository, το τελικό analytical partition αποτελούνταν από 3 Post-Disclosed και 56 Post-Undisclosed repositories. Μία σύγκριση μεταξύ 3 αποθετηρίων και 56 δεν μπορεί να υποστηρίξει στατιστικά συμπεράσματα. Ο πλήρης πίνακας σύγκρισης υπάρχει στο replication package για λόγους πληρότητας. Η μικρότερη προσαρμοσμένη τιμή p σε αυτόν είναι 0,40, για τη δραστηριότητα commits, όπου τα τρία δηλωμένα αποθετήρια είναι ασυνήθιστα ενεργά ($\delta = 0,75$). Με $n = 3$, αυτό αποτελεί απλώς μια παρατήρηση και όχι απόδειξη. Η προειδοποίηση της μεθοδολογίας ισχύει πλήρως: η μη δήλωση δεν σημαίνει μη χρήση. Επομένως, είναι πιθανό τα 56 Post-Undisclosed repositories να περιλαμβάνουν και περιπτώσεις στις οποίες

χρησιμοποιήθηκε AI assistance χωρίς σχετική δήλωση. Για τον λόγο αυτό, η απουσία disclosure δεν μπορεί να ερμηνευθεί ως απουσία χρήσης AI.

Το ίδιο το ποσοστό δήλωσης αποτελεί το πραγματικό εύρημα του RQ3. Σε ένα venue όπου ολόκληρη η υποβολή αποτελεί δημόσιο αποθετήριο και αξιολογείται μέσω ανοιχτής διαδικασίας peer review, κατά τη διάρκεια ετών όπου έρευνες developers και δεδομένα της βιομηχανίας αναφέρουν ευρεία χρήση assistants, μόνο ένας στους είκοσι συγγραφείς το κατέγραψε. Η πρακτική συνέπεια είναι σημαντική: το γραπτό αρχείο του ερευνητικού λογισμικού σήμερα δεν περιέχει τις απαραίτητες πληροφορίες για να συνδεθεί η ποιότητά του με τα εργαλεία που χρησιμοποιήθηκαν για την παραγωγή του.

Με την ορολογία των Moreschini et al. [24], το εύρημα αυτό είναι συμβατό με την έννοια του **prompt-provenance debt** και αποτελεί ένδειξη ελλιπούς καταγραφής της προέλευσης της AI-assisted development στο εξεταζόμενο δείγμα του JOSS.

4.5 Έλεγχοι Συγγυτικών Παραγόντων και Robustness

Πριν ερμηνευτεί το συνολικό μοτίβο, πραγματοποιήθηκαν δύο προγραμματισμένοι έλεγχοι.

Έλεγχος covariates. Κάθε μετρική συσχετίστηκε με τη μέθοδο Spearman με το μέγεθος του αποθετηρίου, το μέγεθος της ομάδας και την ηλικία του αποθετηρίου. Δύο ζεύγη ξεπέρασαν το προκαθορισμένο όριο προειδοποίησης $|r| > 0,4$ και τα δύο αφορούσαν το μέγεθος: ο ελάχιστος δείκτης maintainability ($\rho = -0,67$) και το duplication ratio ($\rho = 0,55$). Και τα δύο έχουν λογική εξήγηση, μια μεγαλύτερη βάση κώδικα περιέχει συνήθως ένα χειρότερο «χειρότερο αρχείο», ενώ μια μέτρηση duplication που βασίζεται σε παράθυρα αυξάνεται με τον συνολικό όγκο του κώδικα.

Αυτό έχει σημασία επειδή το Post cohort είναι μεγαλύτερο. Οι συνέπειες είναι οι εξής: το σταθερό αποτέλεσμα στον δείκτη maintainability δεν είναι αρκετά πληροφοριακό, καθώς το μέγεθος μπορεί να κρύβει μια πραγματική διαφορά προς οποιαδήποτε κατεύθυνση, ενώ η αύξηση του duplication που παρουσιάστηκε στην Ενότητα 4.3 δεν μπορεί να διαχωριστεί με βεβαιότητα από την επίδραση του μεγέθους με το συγκεκριμένο μέγεθος δείγματος.

Καμία άλλη μετρική, συμπεριλαμβανομένων των δύο στατιστικά σημαντικών αποτελεσμάτων, της πυκνότητας παραβιάσεων Pylint και του debt ratio, δεν επισημάνθηκε ως προβληματική σε σχέση με κάποιο covariate.

Έλεγχος robustness του snapshot. Το 91,7% των αποθετηρίων μετρήθηκε στην ακριβή αρχειοθετημένη έκδοση αποδοχής. Όταν οι βασικές συγκρίσεις επαναλήφθηκαν μόνο σε αυτό το ακριβές υποσύνολο, η κατεύθυνση και το κατά προσέγγιση μέγεθος κάθε effect παρέμειναν ίδια. Τα 11 αποθετήρια που επιλύθηκαν με fallback κανόνες δεν καθορίζουν κάποιο από τα αποτελέσματα.

4.6 RQ4: Συνδυάζοντας το Συνολικό Μοτίβο

Το σχέδιο ανάλυσης είχε ορίσει, πριν από οποιαδήποτε μέτρηση, τρία πιθανά μοτίβα για τη συνολική εικόνα: **γεφύρωση (bridging)**, όπου όλα βελτιώνονται μαζί, **απόκρυψη (concealment)**, όπου οι επιφανειακές μετρικές βελτιώνονται αλλά εκείνες που εξαρτώνται από την κρίση του developer παραμένουν σταθερές, και **μηδενικό αποτέλεσμα (null)**, όπου τίποτα δεν αλλάζει. Τα δεδομένα συγκρίνονται με κάθε ένα από αυτά με τη σειρά.

Bridging: δεν υποστηρίζεται από το παρατηρούμενο μοτίβο. Το bridging προέβλεπε βελτίωση τόσο στο Tier 2 όσο και στο Tier 3. Το Tier 3 παραμένει σταθερό σε όλες τις μετρικές. Επομένως,

οποιαδήποτε βελτίωση παρουσιάστηκε μεταξύ των δύο εποχών δεν έφτασε στην τεκμηρίωση ή στην αναπαραγωγικότητα, η κοινή βελτίωση που απαιτούσε η υπόθεση του bridging δεν εμφανίστηκε.

Null: δεν υποστηρίζεται από το παρατηρούμενο μοτίβο. Το null signature απαιτούσε όλα τα effects να είναι μικρότερα από $|\delta| = 0,15$. Τα δύο effects που σχετίζονται με τη συμμόρφωση στους κανόνες (0,30 και 0,34) είναι σαφώς μεγαλύτερα. Επομένως, κάτι πραγματικό άλλαξε.

Concealment: παρουσιάζει μερική αντιστοίχιση με το παρατηρούμενο μοτίβο. Το concealment προέβλεπε ακριβώς το μοτίβο που εμφανίζουν τα δεδομένα: βελτίωση συγκεντρωμένη στις μετρικές που βελτιστοποιούν άμεσα τα αυτοματοποιημένα εργαλεία, όπως οι παραβιάσεις κανόνων ύφους, καμία αλλαγή στις ιδιότητες που απαιτούν την κρίση του developer, όπως η τεκμηρίωση, η αναπαραγωγικότητα και η δομή, αύξηση της επανάληψης κώδικα και μείωση του αναγνωρισμένου χρέους (SATD) ενώ το δομικό χρέος παραμένει σταθερό.

Δύο σημεία αυτού του μοτίβου αποτελούν μόνο τάσεις, η μείωση του SATD και η αύξηση της επανάληψης κώδικα δεν περνούν το όριο στατιστικής σημαντικότητας. Επιπλέον, υπάρχει μία λεπτομέρεια που ξεπερνά την αυστηρότερη πρόβλεψη του concealment: το μετρούμενο debt ratio μειώθηκε. Επομένως, η αντιστοίχιση είναι μερική και όχι τέλεια.

4.6.1 Το Συμπέρασμα που Μπορεί να Υποστηριχθεί

Με προσεκτική διατύπωση, το εύρημα είναι το εξής:

Μεταξύ της προ-AI εποχής και της εποχής AI, το δημοσιευμένο ερευνητικό λογισμικό παρουσίασε παρατηρούμενη αλλά περιορισμένη βελτίωση σε συγκεκριμένες μετρούμενες ιδιότητες: συμμορφώνεται περισσότερο με τους τυπικούς κανόνες προγραμματισμού, δηλαδή με το επίπεδο ποιότητας που επηρεάζεται άμεσα από αυτοματοποιημένα εργαλεία. Η δομική πολυπλοκότητα, η τεκμηρίωση και η αναπαραγωγικότητα δεν παρουσίασαν ανιχνεύσιμη αλλαγή, ενώ δύο μη στατιστικά σημαντικές τάσεις, η αύξηση της επανάληψης κώδικα και η μείωση του αυτο-δηλωμένου χρέους, δείχνουν προς την κατεύθυνση που προβλέπει η υπόθεση της απόκρυψης.

Ένα ερώτημα που ο συγκεκριμένος σχεδιασμός δεν μπορεί να απαντήσει είναι αν η βελτίωση στην επιφάνεια του κώδικα προκλήθηκε πράγματι από τους AI assistants ή από τα εργαλεία linting και formatting που διαδόθηκαν κατά τα ίδια χρόνια. Το υποσύνολο attribution, το οποίο είχε σχεδιαστεί για να διαχωρίσει αυτές τις δύο εξηγήσεις, κατέρρευσε λόγω του ποσοστού δήλωσης μόλις 5%.

Επομένως, η σύγκριση μεταξύ των εποχών μετρά την ίδια την εποχή και η ερμηνεία της υπόθεσης της απόκρυψης πρέπει να παραμείνει εντός του περιορισμού που είχε τεθεί εξ αρχής από τη μεθοδολογία: τα αποτελέσματα είναι **συμβατά με** την ερμηνεία της απόκρυψης, αλλά δεν την αποδεικνύουν.

4.7 Σύνοψη των Ευρημάτων

1. **Η συμμόρφωση με τους κανόνες βελτιώθηκε.** Η πυκνότητα παραβιάσεων Pylint μειώθηκε από 209 σε 163 ανά KLOC ($\delta = 0,34$) και το debt ratio που προκύπτει από τις παραβιάσεις μειώθηκε από 0,052 σε 0,042 ($\delta = 0,30$). Αυτά είναι τα μόνα στατιστικά σημαντικά effects και μετρούν σε σημαντικό βαθμό το ίδιο πράγμα.
2. **Η δομή του κώδικα δεν άλλαξε.** Η πολυπλοκότητα, το nesting και το maintainability παραμένουν σταθερά μεταξύ των εποχών.

3. **Οι μετρούμενες ιδιότητες τεκμηρίωσης και αναπαραγωγιμότητας δεν παρουσίασαν ανιχνεύσιμη μεταβολή μεταξύ των εποχών.** Η τάση της αναπαραγωγιμότητας είναι οριακά αρνητική, παρά την προκατάληψη που ευνοεί το Post.
4. **Παρουσιάστηκαν δύο δυσμενείς τάσεις.** Η επανάληψη κώδικα αυξήθηκε ($\delta = -0,26$), ενώ το αυτο-δηλωμένο τεχνικό χρέος μειώθηκε ($\delta = 0,17$). Καμία από τις δύο τάσεις δεν είναι στατιστικά σημαντική, αλλά και οι δύο αναφέρονται.
5. **Η δήλωση χρήσης ΑΙ είναι μόλις 5%.** Μόνο 3 από τα 60 Post αποθετήρια δηλώνουν χρήση ΑΙ. Η σύγκριση attribution δεν είναι πληροφοριακή, ενώ το ίδιο το ποσοστό καταγράφει ένα κενό προέλευσης στο δημοσιευμένο ερευνητικό λογισμικό.
6. **Τελική αξιολόγηση του μοτίβου.** Το παρατηρούμενο μοτίβο δεν υποστηρίζει το προκαθορισμένο bridging signature ούτε το null signature, ενώ παρουσιάζει μερική αντιστοίχιση με το concealment signature: η βελτίωση συγκεντρώνεται σε χαρακτηριστικά που μπορούν να επηρεαστούν άμεσα από αυτοματοποιημένα εργαλεία, ενώ παρατηρείται στασιμότητα σε ιδιότητες που απαιτούν βαθύτερη ανθρώπινη κρίση.

Κεφάλαιο 5ο: Συμπεράσματα

5.1 Επισκόπηση

Η παρούσα διπλωματική εργασία ξεκίνησε από ένα καλά τεκμηριωμένο πρόβλημα, το χάσμα δεξιοτήτων στην ανάπτυξη επιστημονικού λογισμικού [10], και έθεσε ένα νέο ερώτημα σχετικά με αυτό: αν οι βοηθοί προγραμματισμού με AI, που έγιναν mainstream εργαλεία ανάπτυξης μέσα σε λίγους μήνες, μειώνουν πραγματικά αυτό το χάσμα ή απλώς αλλάζουν τον τρόπο με τον οποίο εκδηλώνεται. Για να απαντηθεί το ερώτημα αυτό, σχεδιάστηκε ένα πλαίσιο μέτρησης τριών επιπέδων και εφαρμόστηκε σε 119 peer-reviewed αποθετήρια ερευνητικού λογισμικού από το Journal of Open Source Software, χωρισμένα σε μια εποχή πριν και μια εποχή μετά την εμφάνιση των AI coding assistants.

Το κεντρικό επιχείρημα της εργασίας ήταν μεθοδολογικό: ότι μια αξιολόγηση η οποία μετρά μόνο την παραγωγικότητα και το τεχνικό χρέος σε επίπεδο κώδικα δεν μπορεί δομικά να διακρίνει αν το χάσμα γεφυρώνεται ή αποκρύπτεται, επειδή και οι δύο υποθέσεις προβλέπουν τις ίδιες επιφανειακές βελτιώσεις. Η εξέταση ενός πρόσθετου επιπέδου μέτρησης, που περιλαμβάνει την αναπαραγωγικότητα, την τεκμηρίωση και το αναγνωρισμένο χρέος, επιτρέπει την πληρέστερη εξέταση αυτής της διάκρισης.

5.2 Κύρια Ευρήματα

Τα αποτελέσματα σχηματίζουν ένα συνεκτικό αλλά σύνθετο μοτίβο. Το δημοσιευμένο ερευνητικό λογισμικό της Post περιόδου παρουσίασε περιορισμένη αλλά μετρήσιμη βελτίωση σε χαρακτηριστικά που μπορούν να επηρεαστούν άμεσα από αυτοματοποιημένα εργαλεία. Συγκεκριμένα, η πυκνότητα παραβιάσεων Pylint και το debt ratio μειώθηκαν με μέτριο και στατιστικά σημαντικό effect, γεγονός που δείχνει ότι ο κώδικας της Post περιόδου παρουσιάζει καλύτερη συμμόρφωση με τυπικούς κανόνες ύφους και ορθότητας. Αυτά ήταν τα μόνα στατιστικά σημαντικά ευρήματα της κύριας σύγκρισης και, επειδή το debt ratio στο χρησιμοποιούμενο pipeline επηρεάζεται εν μέρει από τις παραβιάσεις Pylint, δεν πρέπει να αντιμετωπίζονται ως δύο πλήρως ανεξάρτητες ενδείξεις βελτίωσης.

Πέρα από αυτό το επίπεδο, δεν παρατηρήθηκε αντίστοιχη βελτίωση στις υπόλοιπες διαστάσεις που εξετάστηκαν. Η δομική πολυπλοκότητα, το nesting και η maintainability παρέμειναν σε μεγάλο βαθμό σταθερά. Αντίστοιχα, οι μετρούμενες ιδιότητες τεκμηρίωσης και αναπαραγωγικότητας δεν παρουσίασαν ανιχνεύσιμη βελτίωση μεταξύ των δύο περιόδων. Παράλληλα, εμφανίστηκαν δύο μη στατιστικά σημαντικές τάσεις: αύξηση της επανάληψης κώδικα και μείωση του Self-Admitted Technical Debt. Οι τάσεις αυτές είναι συμβατές με την ερμηνεία της απόκρυψης του χάσματος δεξιοτήτων, χωρίς όμως να παρέχουν από μόνες τους επαρκή στατιστική τεκμηρίωση για την υποστήριξή της.

Σε σχέση με τα τρία προκαθορισμένα patterns της μεθοδολογίας, το παρατηρούμενο μοτίβο δεν υποστηρίζει **bridging signature**, καθώς δεν εμφανίστηκε η αναμενόμενη κοινή βελτίωση τόσο στις μετρικές τεχνικού χρέους όσο και στις μετρικές τεκμηρίωσης και αναπαραγωγικότητας. Παράλληλα, το μοτίβο δεν αντιστοιχεί πλήρως στο **null signature**, καθώς οι μεταβολές στις παραβιάσεις Pylint και στο debt ratio είχαν μέτριο effect και παρέμειναν στατιστικά σημαντικές μετά την προσαρμογή για πολλαπλές συγκρίσεις. Από τις τρεις προκαθορισμένες ερμηνείες, το συνολικό αποτέλεσμα παρουσιάζει μεγαλύτερη, αλλά μόνο μερική, αντιστοίχιση με το **concealment signature**: η βελτίωση συγκεντρώνεται κυρίως σε χαρακτηριστικά που μπορούν να υποστηριχθούν άμεσα από αυτοματοποιημένα εργαλεία, ενώ οι υπόλοιπες ιδιότητες παραμένουν σε μεγάλο βαθμό σταθερές. Η

αντιστοίχιση δεν είναι πλήρης, μεταξύ άλλων επειδή το μετρούμενο debt ratio παρουσίασε επίσης βελτίωση.

Ένα δεύτερο σημαντικό εύρημα προέκυψε από το RQ3. Από τα 60 αρχικά επιλεγμένα Post repositories, μόνο 3 περιείχαν επιβεβαιωμένη δήλωση χρήσης AI assistance. Μετά την αποτυχία μέτρησης ενός μη δηλωμένου repository, η αναλυτική σύγκριση περιορίστηκε σε 3 Post-Disclosed έναντι 56 Post-Undisclosed repositories. Το πολύ μικρό μέγεθος της Post-Disclosed ομάδας δεν επέτρεψε την εξαγωγή αξιόπιστων στατιστικών συμπερασμάτων σχετικά με τη διαφορά μεταξύ repositories με και χωρίς ρητή δήλωση χρήσης AI. Επιπλέον, η απουσία disclosure δεν μπορεί να θεωρηθεί ένδειξη απουσίας χρήσης AI assistance. Το ίδιο το χαμηλό ποσοστό disclosure αποτελεί, ωστόσο, εύρημα σχετικά με την περιορισμένη καταγραφή της χρήσης τέτοιων εργαλείων. Με την ορολογία των Moreschini et al. [24], το αποτέλεσμα αυτό είναι συμβατό με την έννοια του prompt-related provenance debt και αποτελεί ένδειξη ελλιπούς καταγραφής της προέλευσης της AI-assisted development στο εξεταζόμενο δείγμα του JOSS.

Συνολικά, τα αποτελέσματα δείχνουν ότι οι διαφορές μεταξύ της Pre και της Post περιόδου συγκεντρώνονται κυρίως στη συμμόρφωση του κώδικα με αυτοματοποιήσιμους κανόνες, χωρίς αντίστοιχη μεταβολή στις μετρούμενες ιδιότητες τεκμηρίωσης, αναπαραγωγιμότητας και δομικής πολυπλοκότητας. Το μοτίβο αυτό είναι περισσότερο συμβατό με μια μερική απόκρυψη παρά με μια γενικευμένη γεφύρωση του χάσματος δεξιοτήτων. Ωστόσο, επειδή η μελέτη είναι observational και η δυνατότητα άμεσου attribution της χρήσης AI ήταν εξαιρετικά περιορισμένη, τα αποτελέσματα δεν επιτρέπουν την αιτιώδη απόδοση των παρατηρούμενων διαφορών στους AI coding assistants.

5.3 Συζήτηση των Ευρημάτων

Τα αποτελέσματα της παρούσας μελέτης δεν υποδεικνύουν μια απλή μετάβαση από «χαμηλότερη» σε «υψηλότερη» ποιότητα λογισμικού μεταξύ της Pre και της Post περιόδου. Αντίθετα, παρουσιάζουν ένα διαφοροποιημένο μοτίβο, στο οποίο ορισμένες ιδιότητες του κώδικα εμφανίζουν σαφή βελτίωση, ενώ άλλες παραμένουν ουσιαστικά αμετάβλητες. Η διάκριση αυτή είναι κεντρική για την ερμηνεία των αποτελεσμάτων, επειδή το βασικό ερευνητικό ερώτημα της εργασίας δεν αφορά απλώς το αν το λογισμικό της Post περιόδου είναι «καλύτερο», αλλά το αν οι αλλαγές που παρατηρούνται είναι συμβατές με πραγματική γεφύρωση του χάσματος δεξιοτήτων ή περιορίζονται κυρίως σε χαρακτηριστικά που μπορούν να υποστηριχθούν άμεσα από εργαλεία αυτοματοποίησης.

Το ισχυρότερο εμπειρικό αποτέλεσμα της μελέτης βρίσκεται στη συμμόρφωση με τυπικούς κανόνες προγραμματισμού. Η πυκνότητα παραβιάσεων Pylint μειώθηκε από διάμεση τιμή 209 σε 163 παραβιάσεις ανά KLOC, με μέτριο effect size ($\delta = 0,34$) και στατιστική σημαντικότητα μετά τη διόρθωση για πολλαπλές συγκρίσεις. Παράλληλα, το debt ratio μειώθηκε από 0,052 σε 0,042, επίσης με μέτριο effect ($\delta = 0,30$). Αντίθετα, η μέση και μέγιστη κυκλοματική πολυπλοκότητα, το μέγιστο nesting και ο δείκτης maintainability δεν εμφάνισαν αντίστοιχη μεταβολή.

Αυτό σημαίνει ότι το βασικό εύρημα δεν είναι μια οριζόντια βελτίωση της ποιότητας του λογισμικού. Η αλλαγή συγκεντρώνεται σε ένα συγκεκριμένο επίπεδο: στην ικανότητα του κώδικα να συμμορφώνεται με κανόνες που μπορούν να αναγνωριστούν και να διορθωθούν μηχανικά. Η παρατήρηση αυτή είναι ιδιαίτερα σημαντική επειδή το συγκεκριμένο είδος προβλήματος είναι ακριβώς εκείνο στο οποίο τόσο οι σύγχρονοι static analysers όσο και οι AI coding assistants μπορούν να παρέχουν άμεση υποστήριξη.

5.3.1 Βελτίωση της συμμόρφωσης χωρίς αντίστοιχη δομική βελτίωση

Η σημαντική μείωση των Pylint violations αποτελεί το σαφέστερο σημείο διαφοροποίησης μεταξύ των δύο περιόδων. Οι παραβιάσεις αυτές περιλαμβάνουν ένα ευρύ φάσμα προβλημάτων, όπως μη χρησιμοποιούμενες μεταβλητές, μη συνεπή naming conventions, συγκεκριμένες μορφές λανθασμένης χρήσης της Python και άλλες αποκλίσεις από προκαθορισμένους κανόνες. Πρόκειται, δηλαδή, για προβλήματα των οποίων η ανίχνευση μπορεί σε μεγάλο βαθμό να αυτοματοποιηθεί.

Το γεγονός ότι η σημαντικότερη βελτίωση της Post περιόδου εμφανίζεται ακριβώς σε αυτή τη διάσταση είναι συμβατό με τη γενικότερη εξέλιξη των σύγχρονων development environments. Κατά την περίοδο ευρείας διάδοσης των AI coding assistants, οι developers δεν απέκτησαν μόνο πρόσβαση σε μοντέλα παραγωγής κώδικα. Παράλληλα, το οικοσύστημα ανάπτυξης Python έχει ενσωματώσει πιο συστηματικά linters, formatters, IDE diagnostics, pre-commit hooks και continuous integration. Επομένως, η observed temporal difference δεν μπορεί να ερμηνευθεί ως αποκλειστική επίδραση της AI assistance. Η ίδια η μεθοδολογία της μελέτης αναγνωρίζει ότι η σύγκριση Pre–Post μετρά ένα σύνολο αλλαγών της εποχής και όχι μία απομονωμένη τεχνολογική παρέμβαση.

Παρόλα αυτά, η μορφή της βελτίωσης έχει θεωρητική σημασία. Η προηγούμενη βιβλιογραφία έχει δείξει ότι οι AI coding assistants μπορούν να μειώσουν το κόστος παραγωγής κώδικα και να επιταχύνουν την ολοκλήρωση προγραμματιστικών εργασιών. Οι Peng et al. [27] ανέφεραν ταχύτερη ολοκλήρωση μιας ελεγχόμενης προγραμματιστικής εργασίας με χρήση GitHub Copilot, με ιδιαίτερα έντονο όφελος για λιγότερο έμπειρους developers, ενώ οι Ziegler et al. [28] συνέδεσαν τη χρήση neural code completion με αυξημένη αντιλαμβανόμενη παραγωγικότητα.

Η παρούσα μελέτη δεν μετρά άμεσα παραγωγικότητα και επομένως δεν μπορεί να επιβεβαιώσει ή να απορρίψει αυτά τα αποτελέσματα. Ωστόσο, προσθέτει μια διαφορετική διάσταση: ακόμη και αν η παραγωγή κώδικα γίνεται γρηγορότερη, αυτό δεν συνεπάγεται αναγκαστικά αντίστοιχη βελτίωση στη βαθύτερη δομή του λογισμικού. Η σταθερότητα της κυκλοματικής πολυπλοκότητας, του nesting και της maintainability υποδεικνύει ότι η βελτίωση στους κανόνες δεν συνοδεύεται από ανιχνεύσιμη αλλαγή στον τρόπο με τον οποίο οργανώνεται η λογική του κώδικα.

Η διάκριση αυτή έχει ιδιαίτερη σημασία για το χάσμα δεξιοτήτων στο επιστημονικό λογισμικό. Οι Arvanitou et al. [10] και οι Heaton και Carver [12] έχουν δείξει ότι οι αδυναμίες στην ανάπτυξη ερευνητικού λογισμικού δεν περιορίζονται στη σύνταξη ή στο coding style, αλλά συνδέονται με βαθύτερες πρακτικές μηχανικής λογισμικού, όπως οι δοκιμές, η επαλήθευση, η τεκμηρίωση και η συστηματική οργάνωση της ανάπτυξης.

Με αυτή την έννοια, ένας assistant μπορεί να λειτουργεί αποτελεσματικά ως μηχανισμός τοπικής διόρθωσης χωρίς να αντικαθιστά τη συνολική γνώση software engineering. Μπορεί, για παράδειγμα, να προτείνει σωστή σύνταξη, καλύτερα ονόματα, idiomatic Python ή μικρότερες διορθώσεις που αποτρέπουν συγκεκριμένα warnings. Είναι όμως διαφορετικό πρόβλημα να αποφασίσει κανείς πώς πρέπει να αποσυντεθεί μια επιστημονική εφαρμογή, πού απαιτούνται abstractions, ποιο επίπεδο coupling είναι αποδεκτό, πώς πρέπει να οργανωθεί μια test strategy ή ποιο μέρος της επιστημονικής διαδικασίας πρέπει να τεκμηριωθεί.

Η απουσία σημαντικής μεταβολής στις structural metrics είναι επομένως συμβατή με την ιδέα ότι η αυτοματοποίηση μπορεί να αυξάνει το «minimum level» της επιφανειακής ποιότητας χωρίς να μεταβάλλει εξίσου τις ιδιότητες που απαιτούν σχεδιαστική κρίση.

Ταυτόχρονα, χρειάζεται προσοχή στην ερμηνεία του maintainability. Η ανάλυση των covariates έδειξε ισχυρή σχέση ανάμεσα στο repository size και στον ελάχιστο maintainability index ($\rho = -0,67$). Το Post

cohort είναι επίσης μεγαλύτερο σε μέγεθος. Συνεπώς, το null αποτέλεσμα για τη συγκεκριμένη metric δεν μπορεί να ερμηνευθεί με την ίδια εμπιστοσύνη όπως τα αποτελέσματα των Pylint violations ή του debt ratio.

Αυτό ενισχύει ένα γενικότερο συμπέρασμα: η παρούσα εργασία δεν δείχνει ότι «η δομή δεν μπορεί να βελτιωθεί με AI», αλλά ότι στο συγκεκριμένο corpus και με τα συγκεκριμένα instruments δεν παρατηρείται η συστηματική structural improvement που θα αναμενόταν από ένα ισχυρό bridging effect.

5.3.2 Duplication και Self-Admitted Technical Debt ως ενδεικτικές τάσεις

Δύο από τα αποτελέσματα που δεν πέρασαν το όριο της στατιστικής σημαντικότητας παραμένουν ιδιαίτερα ενδιαφέροντα ως προς την κατεύθυνσή τους: η αύξηση του duplication ratio και η μείωση του Self-Admitted Technical Debt.

Το duplication ratio αυξήθηκε από 0,35 στην Pre περίοδο σε 0,45 στην Post, με effect size $\delta = -0,26$. Πρόκειται για μη στατιστικά σημαντικό αποτέλεσμα μετά τη διόρθωση για πολλαπλές συγκρίσεις, επομένως δεν μπορεί να παρουσιαστεί ως τεκμηριωμένη επιδείνωση. Επιπλέον, η ανάλυση covariates έδειξε ότι το duplication σχετίζεται με το μέγεθος του repository ($\rho = 0,55$), ενώ τα Post repositories είναι μεγαλύτερα. Κατά συνέπεια, τουλάχιστον μέρος της διαφοράς μπορεί να εξηγείται από το repository size και όχι από την εποχή ανάπτυξης.

Παρόλα αυτά, η κατεύθυνση της τάσης έχει ενδιαφέρον στο πλαίσιο της βιβλιογραφίας των AI assistants. Ένα εργαλείο που μειώνει σημαντικά το κόστος παραγωγής νέου κώδικα μπορεί θεωρητικά να αλλάξει και το trade-off μεταξύ επαναχρησιμοποίησης υπάρχουσας λογικής και παραγωγής μιας νέας τοπικής λύσης. Όταν η παραγωγή ενός νέου block κώδικα είναι πολύ φθηνή, η άμεση δημιουργία μιας νέας λύσης μπορεί να γίνεται πιο ελκυστική από την κατανόηση, abstraction και επαναχρησιμοποίηση μιας ήδη υπάρχουσας υλοποίησης. Η παρούσα μελέτη δεν μπορεί να αποδείξει ότι αυτός ο μηχανισμός βρίσκεται πίσω από την observed duplication trend, αλλά το αποτέλεσμα δημιουργεί μια εύλογη υπόθεση για μελλοντική έρευνα.

Παρόμοια προσοχή απαιτείται και για το SATD. Η διάμεση πυκνότητα Self-Admitted Technical Debt μειώθηκε από 0,56 σε 0,23 markers ανά KLOC, αλλά με μικρό effect ($\delta = 0,17$) και χωρίς στατιστική σημαντικότητα μετά την προσαρμογή. Το αποτέλεσμα επομένως δεν αποτελεί απόδειξη ότι οι developers της Post περιόδου αναγνωρίζουν λιγότερο χρέος.

Ωστόσο, το SATD έχει διαφορετική ερμηνευτική αξία από τις υπόλοιπες static metrics. Ένα TODO ή FIXME δεν αποτελεί μόνο ένδειξη ύπαρξης προβλήματος. Αποτελεί επίσης ένδειξη ότι ο developer έχει αναγνωρίσει το πρόβλημα και το έχει καταγράψει. Συνεπώς, η μείωση του SATD μπορεί να έχει δύο αντίθετες ερμηνείες. Μπορεί πράγματι να σημαίνει ότι υπάρχουν λιγότερα προβλήματα που απαιτούν μελλοντική εργασία. Μπορεί όμως να σημαίνει ότι τα προβλήματα υπάρχουν αλλά δηλώνονται λιγότερο.

Η δεύτερη ερμηνεία αποκτά ενδιαφέρον επειδή οι structural metrics δεν παρουσιάζουν ταυτόχρονα σαφή βελτίωση. Εάν το SATD μειωνόταν μαζί με την πολυπλοκότητα, το duplication και άλλες μορφές technical debt, το αποτέλεσμα θα μπορούσε να διαβαστεί πιο φυσικά ως αποπληρωμή του χρέους. Όταν όμως το αναγνωρισμένο χρέος μειώνεται χωρίς αντίστοιχη συνολική μεταβολή των structural metrics, η πιθανότητα μεταβολής της αναγνώρισης και όχι αποκλειστικά της πραγματικής ποσότητας χρέους γίνεται ερευνητικά ενδιαφέροντα.

Η βιβλιογραφία γύρω από τους AI assistants προσφέρει έναν σχετικό μηχανισμό. Οι Perry et al. [31] κατέγραψαν περιπτώσεις όπου η χρήση AI assistance συνδέεται όχι μόνο με ζητήματα ασφάλειας αλλά και με αυξημένη αυτοπεποίθηση των developers ως προς την ορθότητα των λύσεων, ενώ οι Dakhel et al. [33] ανέδειξαν τη σημασία της εμπειρίας του developer στην αποτελεσματική αξιολόγηση των προτάσεων.

Η σύνδεση αυτή δεν αποτελεί απόδειξη ότι τα μειωμένα TODO/FIXME του Post cohort οφείλονται σε overconfidence. Δείχνει όμως γιατί η metric είναι θεωρητικά σημαντική. Εάν εργαλεία παραγωγής κώδικα κάνουν τις λύσεις να εμφανίζονται περισσότερο ολοκληρωμένες ή πειστικές, ενώ η ικανότητα του χρήστη να αξιολογήσει βαθύτερα προβλήματα παραμένει αμετάβλητη, τότε η πραγματική μείωση της ορατής αναγνώρισης του χρέους θα ήταν μία πιθανή manifestation του concealment.

Για τον λόγο αυτό, τόσο το duplication όσο και το SATD πρέπει να παραμείνουν στη Discussion ως ενδεικτικές τάσεις και όχι ως επιβεβαιωμένα **effects**. Η αξία τους προκύπτει από τη θέση τους μέσα στο συνολικό pattern και όχι από τη στατιστική τους σημαντικότητα ως μεμονωμένα αποτελέσματα.

5.3.3 Τεκμηρίωση και αναπαραγωγιμότητα: γιατί το null αποτέλεσμα έχει σημασία

Το RQ2 εξετάζει δύο ιδιότητες που έχουν ιδιαίτερη σημασία ειδικά στο επιστημονικό λογισμικό: την τεκμηρίωση και την αναπαραγωγιμότητα. Σε αντίθεση με το coding style, οι ιδιότητες αυτές δεν αφορούν μόνο το αν ο κώδικας είναι καθαρός ή αν περνά έναν static analyser. Αφορούν το αν ένα τρίτο πρόσωπο μπορεί να κατανοήσει, να εγκαταστήσει, να επαναχρησιμοποιήσει και, σε κάποιο βαθμό, να αναπαράγει την υπολογιστική διαδικασία.

Στη μελέτη, το docstring coverage παρέμεινε σχεδόν αμετάβλητο, από 0,54 στο Pre σε 0,53 στο Post, ενώ το comment density μεταβλήθηκε από 0,17 σε 0,16. Τα effect sizes ήταν σχεδόν μηδενικά. Αντίστοιχα, το repository-level reproducibility score παρουσίασε την ίδια διάμεση τιμή στα δύο cohorts, χωρίς στατιστικά σημαντική διαφορά.

Ένα null αποτέλεσμα σε αυτές τις διαστάσεις δεν σημαίνει ότι «δεν συνέβη τίποτα ενδιαφέρον». Αντίθετα, έχει ιδιαίτερη σημασία επειδή η βιβλιογραφία του scientific software έχει επανειλημμένα εντοπίσει την τεκμηρίωση, τις δοκιμές, τη διαχείριση dependencies και την αναπαραγωγιμότητα ως κεντρικά σημεία αδυναμίας. Οι Arvanitou et al. [10] έδειξαν περιορισμένη υιοθέτηση πρακτικών software engineering από επιστημονικούς developers, ενώ οι Heaton και Carver [12] κατέγραψαν την απόσταση μεταξύ της αντιλαμβανόμενης σημασίας πρακτικών όπως η verification/testing και της πραγματικής εφαρμογής τους.

Ακριβώς αυτές είναι οι περιοχές στις οποίες θα έπρεπε να εμφανιστεί βελτίωση εάν οι AI assistants λειτουργούσαν ως ευρύ υποκατάστατο της ελλιπούς εκπαίδευσης στη μηχανική λογισμικού. Ένας πραγματικά strong bridging mechanism δεν θα βελτιώνει μόνο το syntax ή τη συμμόρφωση με κανόνες. Θα βοηθούσε επίσης τον developer να παράγει καλύτερες επεξηγήσεις, πληρέστερα docstrings, πιο ξεκάθαρες οδηγίες εκτέλεσης και πιο αναπαραγώγιμα artefacts.

Τα αποτελέσματα δεν παρέχουν τέτοια ένδειξη. Η τεκμηρίωση παραμένει πρακτικά σταθερή. Αυτό έχει ενδιαφέρον επειδή η παραγωγή docstrings και σχολίων αποτελεί στην πραγματικότητα εργασία την οποία ένα language model μπορεί τεχνικά να εκτελέσει πολύ εύκολα. Επομένως, η απουσία αύξησης δεν μπορεί απλώς να εξηγηθεί από τεχνική αδυναμία των assistants να παράγουν κείμενο τεκμηρίωσης.

Μία πιθανή ερμηνεία είναι ότι το bottleneck δεν βρίσκεται στην παραγωγή κειμένου αλλά στην απόφαση τι πρέπει να τεκμηριωθεί και γιατί. Ένα εργαλείο μπορεί να δημιουργήσει ένα συντακτικά πλήρες docstring όταν του ζητηθεί, αλλά η συστηματική τεκμηρίωση ενός επιστημονικού project

απαιτεί από τον developer να αναγνωρίσει ποια assumptions, units, input constraints, methodological choices και execution steps είναι κρίσιμα για έναν μελλοντικό χρήστη. Αυτή η απόφαση απαιτεί domain knowledge και engineering judgement.

Παρόμοια προσοχή χρειάζεται στην ερμηνεία της αναπαραγωγιμότητας. Όπως αναλύθηκε στη Μεθοδολογία και στα Αποτελέσματα, η τελική mining run δεν αποτελεί πλήρη reproduction experiment των 119 repositories. Συνεπώς, το αποτέλεσμα πρέπει να ερμηνεύεται ως στασιμότητα στις μετρούμενες repository-level ενδείξεις reproducibility και όχι ως απόδειξη ότι η πραγματική δυνατότητα αναπαραγωγής επιστημονικών αποτελεσμάτων είναι ίδια μεταξύ των δύο περιόδων.

Παρόλα αυτά, το γεγονός ότι ακόμη και αυτές οι βασικές και περισσότερο era-neutral διαστάσεις δεν παρουσιάζουν ανιχνεύσιμη βελτίωση είναι σημαντικό. Δεν εμφανίζεται, δηλαδή, το broad improvement που θα υποστήριζε ισχυρά την υπόθεση της γεφύρωσης.

Η ερμηνεία αυτή συνδέεται και με το θεμελιώδες πρόβλημα της επιστημονικής αναπαραγωγιμότητας που παρουσιάστηκε στη σχετική εργασία. Οι Stodden et al. [9] έδειξαν πόσο δύσκολη μπορεί να είναι η πραγματική αναπαραγωγή computational research ακόμη και όταν υπάρχει πολιτική κοινοποίησης κώδικα και δεδομένων. Η απλή παραγωγή περισσότερου ή καθαρότερου κώδικα δεν λύνει αυτόματα το πρόβλημα, επειδή η reproducibility απαιτεί ένα σύνολο πρακτικών που ξεπερνούν τον τοπικό πηγαίο κώδικα.

5.3.4 Bridging, Null και Concealment

Η ερμηνεία των αποτελεσμάτων πρέπει τελικά να επιστρέψει στις τρεις προκαθορισμένες προβλέψεις του πλάνου ανάλυσης: bridging, null και concealment. Το σημαντικό εδώ είναι ότι οι κατηγορίες αυτές δεν κατασκευάστηκαν εκ των υστέρων για να ταιριάζουν στα δεδομένα. Ορίστηκαν πριν από τη μέτρηση ως διαφορετικές προβλέψεις για το συνολικό pattern.

Το **bridging signature** προέβλεπε ευρύτερη βελτίωση. Εάν οι assistants λειτουργούσαν ως αποτελεσματικό υποκατάστατο μέρους της ελλιπούς software-engineering εκπαίδευσης, θα αναμενόταν η αλλαγή να εμφανίζεται τόσο στο code-level technical debt όσο και σε υψηλότερες ιδιότητες όπως documentation και reproducibility. Αυτή η πρόβλεψη δεν επιβεβαιώνεται. Η βελτίωση περιορίζεται κυρίως στη rule conformance, ενώ το Tier 3 παραμένει ουσιαστικά σταθερό.

Το **null signature** επίσης δεν περιγράφει ικανοποιητικά τα αποτελέσματα. Αν η μετάβαση στην Post περίοδο δεν συνοδευόταν από καμία ουσιαστική μεταβολή, τα effect sizes θα αναμενόταν μικρά σε όλες τις μετρικές. Αντίθετα, οι Pylint violations και το debt ratio εμφανίζουν effects της τάξης του 0,30–0,34 και παραμένουν statistically significant μετά την Benjamini–Hochberg correction. Επομένως, δεν είναι ακριβές να περιγραφούν οι δύο εποχές ως πλήρως ισοδύναμες.

Το **concealment signature** εμφανίζει τη μεγαλύτερη αντιστοίχιση, αλλά η αντιστοίχιση είναι μόνο μερική. Το pattern που παρατηρείται πράγματι περιλαμβάνει βελτίωση σε style-adjacent και tool-addressable characteristics, στασιμότητα σε documentation/reproducibility/structural characteristics και δύο τάσεις που κινούνται προς την προβλεπόμενη κατεύθυνση: περισσότερο duplication και λιγότερο SATD. Ωστόσο, το debt ratio βελτιώνεται επίσης και οι δύο τελευταίες τάσεις δεν είναι statistically significant. Για τον λόγο αυτό, η λέξη μερική είναι ουσιαστική και πρέπει να διατηρείται σε όλη τη συζήτηση.

Η συνολική εικόνα επομένως δεν υποστηρίζει τη διατύπωση ότι οι AI coding assistants «αποτυγχάνουν» ή ότι «χειροτερεύουν» το επιστημονικό λογισμικό. Τα αποτελέσματα δείχνουν πραγματική βελτίωση σε μία διάσταση. Το ζήτημα είναι ότι η βελτίωση αυτή δεν επεκτείνεται στις

υπόλοιπες διαστάσεις με τρόπο που να δικαιολογεί τον ισχυρότερο ισχυρισμό ότι το underlying skills gap έχει γεφυρωθεί.

Η διάκριση αυτή συμφωνεί με την ένταση που έχει ήδη εντοπιστεί στη σχετική βιβλιογραφία. Από τη μία πλευρά, μελέτες παραγωγικότητας έχουν καταγράψει ουσιαστικά οφέλη [27], [28]. Από την άλλη, οι Pearce et al. [30] και Perry et al. [31] έχουν αναδείξει ζητήματα ασφάλειας και αξιολόγησης του παραγόμενου κώδικα, ενώ οι Dakhel et al. [33] δείχνουν ότι η εμπειρία του developer εξακολουθεί να παίζει σημαντικό ρόλο.

Η παρούσα μελέτη επεκτείνει αυτή την ένταση από controlled programming tasks σε πραγματικό, peer-reviewed scientific software. Το ενδιαφέρον εύρημα δεν είναι ότι οι assistants είναι «καλοί» ή «κακοί», αλλά ότι η πιθανή επίδραση της εποχής τους φαίνεται ανομοιόμορφη μεταξύ διαφορετικών layers της ποιότητας.

5.3.5 Disclosure, attribution και provenance debt

Το RQ3 είχε σχεδιαστεί ώστε να παρέχει την πιο άμεση σύνδεση μεταξύ των observed differences και της πραγματικής χρήσης AI assistance. Αν υπήρχε επαρκής αριθμός Post-Disclosed repositories, η σύγκριση με το Post-Undisclosed subset θα μπορούσε να περιορίσει σημαντικά το πρόβλημα του temporal confounding, επειδή και οι δύο ομάδες θα ανήκαν στην ίδια γενική εποχή.

Στην πράξη, όμως, μόνο 3 από τα 60 αρχικά Post repositories περιείχαν επιβεβαιωμένη δήλωση AI-assisted development. Μετά την αποτυχία μέτρησης ενός μη δηλωμένου repository, η comparison group ήταν 3 έναντι 56. Η ανισορροπία αυτή καθιστά οποιαδήποτε inferential σύγκριση μη αξιόπιστη.

Η κατάρρευση του attribution analysis δεν πρέπει να αντιμετωπιστεί μόνο ως αποτυχία της μεθοδολογίας. Το ίδιο το χαμηλό disclosure rate αποτελεί σημαντικό εύρημα.

Σε ένα περιβάλλον όπως το JOSS, το software artefact βρίσκεται στο κέντρο της δημοσίευσης. Το repository είναι δημόσιο, εξετάζεται από reviewers και συνοδεύει μια ανοιχτή επιστημονική εργασία. Παρ' όλα αυτά, μόνο περίπου το 5% του εξεταζόμενου Post corpus περιέχει explicit disclosure της χρήσης AI assistance.

Αυτό δημιουργεί πρόβλημα provenance. Για σημαντικό μέρος του σύγχρονου software-development process, το δημοσιευμένο artefact δεν καταγράφει ποια εργαλεία συνέβαλαν στην παραγωγή του. Η κατάσταση αυτή διαφέρει από πολλές άλλες επιστημονικές διαδικασίες, όπου θεωρείται αναμενόμενο να καταγράφονται κρίσιμα εργαλεία, datasets, όργανα ή μέθοδοι.

Η έννοια αυτή συνδέεται άμεσα με τη συζήτηση των Moreschini et al. [24] γύρω από νέες μορφές technical debt στην εποχή της generative AI, μεταξύ των οποίων debt που σχετίζεται με την προέλευση και το prompt context της generated solution.

Στην περίπτωση του επιστημονικού λογισμικού, η έλλειψη provenance έχει τουλάχιστον δύο συνέπειες. Πρώτον, δυσκολεύει την εμπειρική έρευνα: χωρίς αξιόπιστη πληροφορία χρήσης δεν μπορεί να πραγματοποιηθεί ισχυρή attribution analysis. Δεύτερον, μπορεί να δυσκολεύει και την επιστημονική αξιολόγηση, επειδή οι reviewers και οι μελλοντικοί maintainers δεν γνωρίζουν ποια τμήματα της development process υποστηρίχθηκαν ή παρήχθησαν από ένα probabilistic generative system.

Ταυτόχρονα, πρέπει να αποφευχθεί μια λανθασμένη αντίστροφη υπόθεση. Η απουσία disclosure δεν σημαίνει απουσία χρήσης. Το Post-Undisclosed subset είναι επομένως μια ομάδα «unknown usage» και όχι μια πραγματική ομάδα ελέγχου χωρίς AI. Η πιθανή contamination αυτής της ομάδας αναμένεται να

μειώνει οποιαδήποτε πραγματική difference μεταξύ disclosed και undisclosed repositories, όχι να την ενισχύει.

Το RQ3 επομένως δεν μπορεί να απαντήσει αξιόπιστα αν η χρήση AI assistance σχετίζεται με καλύτερες ή χειρότερες μετρικές στο επίπεδο repository. Μπορεί όμως να απαντήσει κάτι διαφορετικό: με τις σημερινές πρακτικές καταγραφής, η open scientific software ecosystem δεν παρέχει ακόμη επαρκή provenance information ώστε τέτοιο ερώτημα να εξεταστεί συστηματικά.

5.3.6 Εναλλακτικές εξηγήσεις και σημασία του observational σχεδιασμού

Ο σημαντικότερος περιορισμός στην ερμηνεία των αποτελεσμάτων είναι ότι το Pre/Post boundary δεν απομονώνει την AI assistance. Μεταξύ των δύο περιόδων άλλαξαν ταυτόχρονα πολλοί παράγοντες: το Python ecosystem, η ωριμότητα package-management practices, η διάδοση των linters και formatters, η continuous integration και πιθανώς οι expectations των reviewers και της κοινότητας.

Αυτό σημαίνει ότι η σημαντική μείωση των Pylint violations μπορεί να προκύπτει από πολλαπλούς μηχανισμούς. Ένας AI assistant μπορεί να παράγει περισσότερο idiomatic code. Ένας σύγχρονος IDE μπορεί να διορθώνει warnings πριν από το commit. Ένα formatter ή pre-commit configuration μπορεί να επιβάλλει conventions αυτόματα. Οι reviewers μπορεί επίσης να είναι περισσότερο ευαισθητοποιημένοι σε συγκεκριμένες πρακτικές.

Η μελέτη επομένως δεν μπορεί να αποφανθεί ποιο ποσοστό της αλλαγής οφείλεται σε κάθε μηχανισμό. Το claim πρέπει να παραμένει χρονικό και pattern-based: η **Post** περίοδος χαρακτηρίζεται από καλύτερη **rule conformance**, και το συνολικό pattern είναι περισσότερο συμβατό με partial concealment παρά με broad bridging. Δεν μπορεί να μετατραπεί σε αιτιώδη διατύπωση του τύπου «οι AI assistants προκάλεσαν τη βελτίωση». Η ίδια η ενότητα απειλών εγκυρότητας θέτει αυτό το όριο ρητά.

Υπάρχει επιπλέον ένας παράγοντας που μπορεί να μειώνει τις διαφορές: το ίδιο το JOSS. Τα repositories του corpus δεν αποτελούν τυχαία open-source projects. Έχουν περάσει από δημόσια, checklist-driven διαδικασία review, με απαιτήσεις σχετικές με documentation, tests, installability και community guidelines. Αυτό δημιουργεί ένα κοινό quality floor τόσο στο Pre όσο και στο Post cohort.

Ένα τέτοιο quality floor μπορεί να λειτουργεί ως εξισωτικός παράγοντας. Αν το JOSS review αναγκάζει ακόμη και σχετικά αδύναμα repositories να βελτιώσουν documentation ή packaging πριν από την αποδοχή, τότε μια ενδεχόμενη προγενέστερη διαφορά μεταξύ Pre και Post projects μπορεί να είναι μικρότερη τη στιγμή του acceptance snapshot. Με αυτή την έννοια, το null αποτέλεσμα σε documentation και reproducibility μπορεί εν μέρει να αντανάκλα τη σταθεροποιητική επίδραση του peer review.

Η ερμηνεία αυτή δεν μειώνει την αξία του corpus. Αντίθετα, εξηγεί γιατί η επιλογή του JOSS είναι ταυτόχρονα δύναμη και περιορισμός. Προσφέρει ένα σχετικά ομοιογενές και πραγματικό population of published research software, αλλά εξετάζει software που έχει ήδη περάσει από quality control. Συνεπώς, τα αποτελέσματα δεν πρέπει να γενικεύονται αυτόματα σε μη δημοσιευμένα research scripts, ad-hoc notebooks ή internal laboratory code, όπου το skills gap μπορεί να είναι πολύ πιο έντονο.

5.3.7 Επιπτώσεις για την ανάπτυξη επιστημονικού λογισμικού

Το συνολικό αποτέλεσμα έχει πρακτικές συνέπειες για το πώς πρέπει να αντιμετωπίζεται η χρήση AI coding assistants στην επιστημονική ανάπτυξη λογισμικού.

Πρώτον, τα δεδομένα δεν υποστηρίζουν μια θέση σύμφωνα με την οποία οι assistants δεν προσφέρουν καμία αξία. Η βελτίωση της rule conformance είναι πραγματική στο εξεταζόμενο corpus και μπορεί να είναι πρακτικά χρήσιμη. Για developers χωρίς μεγάλη εμπειρία στην Python, η αυτόματη παραγωγή περισσότερο idiomatic code και η αποφυγή συνηθισμένων violations μπορεί να μειώσει ένα μέρος της καθημερινής τριβής και να βελτιώσει τη βασική ποιότητα του artifact.

Δεύτερον, όμως, η χρήση τέτοιων εργαλείων δεν φαίνεται επαρκής ως υποκατάστατο της software-engineering εκπαίδευσης. Οι αδυναμίες που έχουν καταγραφεί επί χρόνια στο scientific software αφορούν ακριβώς τις πρακτικές που απαιτούν κατανόηση του κύκλου ζωής του λογισμικού: testing, verification, documentation, reproducibility, maintainability και long-term design. Το γεγονός ότι αυτές οι διαστάσεις δεν παρουσιάζουν αντίστοιχη βελτίωση σημαίνει ότι η διάδοση AI assistance δεν καταργεί την ανάγκη εκπαίδευσης σε αυτές.

Τρίτον, η αξιολόγηση της ποιότητας δεν πρέπει να περιορίζεται στην εμφανή καθαρότητα του κώδικα. Ένας κώδικας με λιγότερα linter warnings μπορεί να είναι ευκολότερος στην ανάγνωση και συντήρηση, αλλά αυτό δεν παρέχει από μόνο του evidence για scientific correctness. Ο σημαντικότερος blind spot της παρούσας μελέτης είναι ακριβώς ότι η αριθμητική και επιστημονική ορθότητα δεν μπορεί να μετρηθεί με general-purpose repository mining. Όπως έχει ήδη δείξει η βιβλιογραφία του scientific software, ο φαινομενικά έγκυρος κώδικας μπορεί να παράγει επιστημονικά λανθασμένα αποτελέσματα.

Τέταρτον, χρειάζεται σαφέστερη πολιτική disclosure. Εάν η χρήση AI assistance αποτελεί πλέον ουσιαστικό μέρος της development process, η καταγραφή της μπορεί να αντιμετωπιστεί με τρόπο ανάλογο με άλλα methodological artefacts. Αυτό δεν σημαίνει ότι κάθε completion πρέπει να καταγράφεται. Σημαίνει ότι τα peer-reviewed software venues μπορούν να ορίσουν ένα ελάχιστο disclosure convention: ποιο εργαλείο χρησιμοποιήθηκε, σε ποια κατηγορία εργασιών και αν ο generated code υποβλήθηκε σε ανεξάρτητο verification ή review.

Μια τέτοια πρακτική θα είχε αξία όχι μόνο για λόγους transparency αλλά και για την ίδια την επιστημονική έρευνα. Χωρίς provenance, οποιαδήποτε μελλοντική προσπάθεια να μετρηθεί εμπειρικά η επίδραση των assistants θα αντιμετωπίζει το ίδιο attribution problem με το RQ3 της παρούσας μελέτης.

5.3.8 Συνολική ερμηνεία

Η συνολική εικόνα που προκύπτει είναι περισσότερο σύνθετη από μια απλή θέση υπέρ ή κατά των AI coding assistants. Τα αποτελέσματα δείχνουν ότι η Post περίοδος συνοδεύεται από βελτίωση σε συγκεκριμένες διαστάσεις της ποιότητας του κώδικα, αλλά η βελτίωση αυτή δεν επεκτείνεται με την ίδια ένταση σε χαρακτηριστικά που απαιτούν βαθύτερη engineering judgement.

Αυτό αποτελεί ακριβώς τη διάκριση ανάμεσα στην επιφανειακή υποστήριξη και στη γεφύρωση ενός skills gap. Εάν ένα εργαλείο βοηθά έναν developer να παράγει κώδικα που περνά ευκολότερα από έναν linter, αυτό αποτελεί πραγματικό όφελος. Δεν σημαίνει όμως αναγκαστικά ότι ο developer απέκτησε τις δεξιότητες για να αξιολογήσει τη μακροπρόθεσμη maintainability, να σχεδιάσει κατάλληλη test strategy, να τεκμηριώσει τις επιστημονικές assumptions ή να εγγυηθεί την reproducibility και correctness της υπολογιστικής διαδικασίας.

Η βιβλιογραφία γύρω από τους AI coding assistants έχει ήδη δείξει αυτή τη δυαδικότητα. Τα εργαλεία μπορούν να προσφέρουν productivity gains [27], [28], ενώ ταυτόχρονα η ποιότητα και η αξιολόγηση των generated solutions εξακολουθούν να εξαρτώνται από το context, το prompt, το εργαλείο και την εμπειρία του χρήστη [30]–[36]. Η παρούσα μελέτη δείχνει ότι μια παρόμοια δυαδικότητα είναι ορατή και σε πραγματικό peer-reviewed scientific software: καλύτερη rule conformance χωρίς αντίστοιχη broad improvement στις υπόλοιπες διαστάσεις.

Για τον λόγο αυτό, το παρατηρούμενο pattern δεν υποστηρίζει την ισχυρή εκδοχή της υπόθεσης της γεφύρωσης. Δεν υποστηρίζει όμως ούτε μια πλήρως αρνητική ερμηνεία. Η πιο ακριβής διατύπωση είναι ότι τα αποτελέσματα είναι περισσότερο συμβατά με μερική απόκρυψη του χάσματος δεξιοτήτων, επειδή η improvement συγκεντρώνεται κυρίως σε χαρακτηριστικά που μπορούν να υποστηριχθούν άμεσα από αυτοματοποιημένα εργαλεία, ενώ οι μετρούμενες ιδιότητες που απαιτούν ευρύτερη ανθρώπινη κρίση παραμένουν κατά βάση σταθερές.

Η λέξη «συμβατά» είναι κρίσιμη. Η observational φύση της μελέτης, το temporal confounding και η αποτυχία της attribution comparison λόγω του πολύ χαμηλού disclosure rate δεν επιτρέπουν αιτιώδη συμπεράσματα. Η εργασία επομένως δεν αποδεικνύει ότι οι AI coding assistants αποκρύπτουν το skills gap. Δείχνει ότι το pattern που εμφανίζεται στο συγκεκριμένο population of published scientific software βρίσκεται πιο κοντά σε αυτή την προκαθορισμένη πρόβλεψη από ό,τι στο broad bridging ή στο null pattern.

Αυτό είναι και το βασικότερο επιστημονικό συμπέρασμα της μελέτης: η αξιολόγηση των AI coding assistants δεν πρέπει να περιορίζεται στο πόσο κώδικα παράγουν ή στο αν ο παραγόμενος κώδικας φαίνεται καθαρότερος. Για να αξιολογηθεί αν τα εργαλεία μειώνουν πραγματικά ένα engineering skills gap, απαιτείται μέτρηση σε διαφορετικά επίπεδα, από την τοπική συμμόρφωση του κώδικα έως την τεκμηρίωση, τη reproducibility, την αναγνώριση technical debt και, όπου αυτό είναι δυνατό, την επιστημονική correctness. Η βελτίωση σε ένα μόνο επίπεδο δεν αποτελεί από μόνη της απόδειξη ότι το υποκείμενο χάσμα έχει εξαφανιστεί.

5.4 Περιορισμοί και Απειλές για την Εγκυρότητά

Τα συμπεράσματα της παρούσας μελέτης πρέπει να ερμηνευθούν εντός των ορίων του ερευνητικού σχεδιασμού.

5.4.1 Internal Validity

Temporal confounding. Η βασική απειλή, χωρίς να μετριάζεται: τα πάντα συνμεταβάλλονται με το όριο μεταξύ των εποχών, η ωριμότητα του Python ecosystem, οι packaging norms, η διαθεσιμότητα CI, οι community expectations και το JOSS reviewer pool. Επομένως, μια διαφορά μεταξύ των εποχών δεν μπορεί ποτέ να αποδοθεί από μόνη της στην AI assistance. Τα mitigations του σχεδιασμού είναι το stable review checklist, η εξέταση των covariates, οι έλεγχοι συσχέτισης Spearman μεταξύ χαρακτηριστικών των repositories και μετρικών αξιολόγησης και, όπου εφαρμόζεται, η within-era attribution contrast (RQ3), την οποία το background drift δεν μπορεί να εξηγήσει. Ό,τι παραμένει μετά και από τα τρία αποτελεί ένα σύνολο evidence συμβατό με tool effect, το discussion chapter, βάσει αυτής της παραγράφου, δεν επιτρέπεται να ισχυριστεί κάτι περισσότερο.

Acceptance-lag dilution. Η ανάθεση εποχής με βάση την ημερομηνία αποδοχής αναμειγνύει pre-era development στο Post cohort, ωθώντας τις διαφορές μεταξύ των εποχών προς το μηδέν. Ο περιορισμός του dilution αντιμετωπίζεται μέσω της προσεκτικής επιλογής των χρονικών ορίων των cohorts και της αξιολόγησης της σταθερότητας των αποτελεσμάτων μέσω διαφορετικών snapshots των repositories..

Attrition asymmetry. Τα repositories που αποτυγχάνουν στη μέτρηση αποκλείονται (E3), και η αποτυχία μπορεί να συσχετίζεται με την εποχή, παλαιότερα snapshots μπορεί να είναι λιγότερο αξιόπιστα στην επαναφορά τους. Η attrition flow αναφέρεται ανά cohort, και differential attrition rate άνω των δέκα ποσοστιαίων μονάδων ενεργοποιεί documented re-examination των λόγων αποτυχίας.

5.4.2 Construct Validity

Non-disclosure is not non-use. Το attribution partition συγκρίνει known-assisted με mixed-and-unknown οι διαφορές του RQ3 είναι υποεκτιμήσεις και τα nulls του RQ3 δεν είναι πληροφοριακά. Αυτός είναι ο σημαντικότερος construct limitation της μελέτης και επαναλαμβάνεται κάθε φορά που ερμηνεύονται αποτελέσματα του RQ3.

Disclosure correlates with conscientiousness. Οι developers που δηλώνουν τη χρήση εργαλείων μπορεί να είναι συστηματικά πιο προσεκτικοί στην τεκμηρίωση γενικότερα, αυξάνοντας τα documentation metrics του Post-Disclosed για λόγους που δεν σχετίζονται με τον assistant. Η qualitative coding των disclosure statements παρέχει μερική εικόνα πάνω σε αυτό.

Correctness is unobserved. Αυτό αποδεικνύεται και δεν θεωρείται δεδομένο, μέσω της validation study: αριθμητικά λανθασμένος κώδικας μπορεί να λάβει τέλεια βαθμολογία σε κάθε metric του Πίνακα II. Οι claims σχετικά με «research quality» αφορούν την reproducibility, την documentation και το debt, και ποτέ το αν η επιστήμη υπολόγισε τους σωστούς αριθμούς.

Proxy metrics. Το debt ratio προσεγγίζει το SQALE model αντί να αναπαράγει μια commercial implementation· το reproducibility score δίνει ίσο βάρος σε τέσσερα binary components· τα SATD markers μετρούν vocabulary αναγνώρισης και το vocabulary μπορεί να αλλάζει με τις τάσεις. Κάθε metric τεκμηριώνεται στο validation report μαζί με τα γνωστά blind spots της.

5.4.3 External Validity

One journal, one language. Το frame είναι το JOSS και η γλώσσα είναι η Python. Οι authors του JOSS επιλέγουν οι ίδιοι να συμμετέχουν σε open και review-willing development, επομένως το corpus πιθανότατα αντιπροσωπεύει το upper tail της πρακτικής στο research software· οι επιδράσεις στον γενικό πληθυσμό unpublished scientific scripts μπορεί να είναι μεγαλύτερες. Τα ευρήματα γενικεύονται σε peer-reviewed, open-source Python research software και πέρα από αυτό μόνο μέσω επιχειρηματολογίας.

Era, not tool. Ακόμη και όταν το attribution είναι επιτυχές, η μελέτη παρατηρεί τους mainstream assistants της περιόδου 2023–2025 όπως πραγματικά διαμορφώθηκαν από τους χρήστες τους και όχι κάποιο συγκεκριμένο product ή capability level.

5.4.4 Conclusion Validity

Subsample power: το μέγεθος της Post-Disclosed ομάδας δεν ελέγχεται από τη μελέτη· εάν είναι μικρό, η ανάλυση RQ3 αναφέρει interval estimates και περιορίζει τη χρήση hypothesis tests.

Analyst degrees of freedom: το πλάνο καθορίζεται εδώ, πριν από τη μέτρηση· οτιδήποτε πέρα από αυτό χαρακτηρίζεται ως exploratory.

5.5 Μελλοντική Εργασία

Τρεις κατευθύνσεις προκύπτουν άμεσα. Πρώτον, η κατάρρευση του RQ3 δείχνει ότι η μελέτη της επίδρασης των AI assistants στο δημοσιευμένο λογισμικό απαιτεί έναν αξιόπιστο μηχανισμό attribution· η καθιέρωση συμβάσεων δήλωσης χρήσης AI στα peer-reviewed venues, αντίστοιχων με τα acknowledgements της συνηθισμένης επιστημονικής δημοσίευσης, θα καθιστούσε εφικτή τη σύγκριση που εδώ δεν μπόρεσε να πραγματοποιηθεί. Δεύτερον, το τυφλό σημείο της αριθμητικής ορθότητας θα μπορούσε να αντιμετωπιστεί σε μια στοχευμένη μελέτη σε τομείς όπου υπάρχουν αναλυτικές λύσεις αναφοράς ή καθιερωμένα benchmark, ώστε να εξεταστεί αν η επιφανειακή βελτίωση συνοδεύεται ή όχι από αντίστοιχη ορθότητα. Τρίτον, ένα controlled experiment με τον πληθυσμό των επιστημονικών προγραμματιστών, παρά τις δυσκολίες στρατολόγησης που οδήγησαν στην παρούσα επιλογή σχεδιασμού, θα μπορούσε να παρατηρήσει άμεσα τη διάσταση της κατανόησης, η οποία σε μια mining μελέτη παραμένει εξ ορισμού απρόσιτη.

5.6 Καταληκτική Παρατήρηση

Το παρατηρούμενο μοτίβο μεταξύ της Pre και της Post περιόδου είναι περισσότερο συμβατό με μια μερική απόκρυψη του skill gap παρά με μια γενικευμένη γεφύρωσή του: η βελτίωση συγκεντρώνεται σε χαρακτηριστικά που μπορούν να υποστηριχθούν άμεσα από αυτοματοποιημένα εργαλεία, ενώ η τεκμηρίωση και η αναπαραγωγικότητα παραμένουν ουσιαστικά σταθερές. Ωστόσο, λόγω του observational σχεδιασμού και της περιορισμένης δυνατότητας attribution, τα αποτελέσματα δεν επιτρέπουν την αιτιώδη απόδοση αυτού του μοτίβου στους AI coding assistants. Η μελέτη αναδεικνύει επομένως την ανάγκη για ισχυρότερες πρακτικές disclosure και για μελλοντικές μελέτες που θα μπορούν να απομονώσουν άμεσα την επίδραση των εργαλείων.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- [1] J. E. Hannay, C. MacLeod, J. Singer, H. P. Langtangen, D. Pfahl, and G. Wilson, "How do scientists develop and use scientific software?" in *Proc. ICSE Workshop Softw. Eng. Comput. Sci. Eng.*, Vancouver, BC, Canada, 2009, pp. 1–8.
- [2] P. Prabhu *et al.*, "A survey of the practice of computational science," in *Proc. 24th ACM/IEEE Conf. High Perform. Comput., Netw., Storage Anal.*, Seattle, WA, USA, 2011, pp. 19:1–19:12.
- [3] U. Nangia and D. S. Katz, "Surveying the U.S. national postdoctoral association regarding software use and training in research," in *Proc. Workshop Sustain. Softw. Sci.: Pract. Exp. (WSSSPE 5.1)*, Manchester, U.K., 2017.
- [4] G. Wilson, "Software carpentry: Getting scientists to write better code by making them more productive," *Comput. Sci. Eng.*, vol. 8, no. 6, pp. 66–69, 2006.
- [5] L. Hatton and A. Roberts, "How accurate is scientific software?" *IEEE Trans. Softw. Eng.*, vol. 20, no. 10, pp. 785–797, 1994.
- [6] Z. Merali, "Computational science: ...Error," *Nature*, vol. 467, no. 7317, pp. 775–777, 2010.
- [7] L. N. Joppa *et al.*, "Troubling trends in scientific software use," *Science*, vol. 340, no. 6134, pp. 814–815, 2013.
- [8] M. Baker, "1,500 scientists lift the lid on reproducibility," *Nature*, vol. 533, no. 7604, pp. 452–454, 2016.
- [9] V. Stodden, J. Seiler, and Z. Ma, "An empirical analysis of journal policy effectiveness for computational reproducibility," *Proc. Nat. Acad. Sci.*, vol. 115, no. 11, pp. 2584–2589, 2018.
- [10] E.-M. Arvanitou, A. Ampatzoglou, A. Chatzigeorgiou, and J. C. Carver, "Software engineering practices for scientific software development: A systematic mapping study," *J. Syst. Softw.*, vol. 172, 2021, Art. no. 110848.
- [11] S. Faulk, E. Loh, M. L. Van De Vanter, S. Squires, and L. G. Votta, "Scientific computing's productivity gridlock: How software engineering can help," *Comput. Sci. Eng.*, vol. 11, no. 6, pp. 30–39, 2009.
- [12] D. Heaton and J. C. Carver, "Claims about the use of software engineering practices in science: A systematic literature review," *Inf. Softw. Technol.*, vol. 67, pp. 207–219, 2015.
- [13] J. Segal, "When software engineers met research scientists: A case study," *Empir. Softw. Eng.*, vol. 10, no. 4, pp. 517–536, 2005.
- [14] M. Schmidberger and B. Brügge, "Need of software engineering methods for high performance computing applications," in *Proc. 11th Int. Symp. Parallel Distrib. Comput.*, Munich, Germany, 2012, pp. 40–46.
- [15] W. Cunningham, "The WyCash portfolio management system," in *Addendum Proc. Object-Oriented Program. Syst., Lang., Appl. (OOPSLA '92)*, Vancouver, BC, Canada, 1992, pp. 29–30.
- [16] S. McConnell, "Managing technical debt," *Construx Softw.*, Bellevue, WA, USA, Tech. Rep., 2008.

- [17] P. Kruchten, R. L. Nord, and I. Ozkaya, "Technical debt: From metaphor to theory and practice," *IEEE Softw.*, vol. 29, no. 6, pp. 18–21, 2012.
- [18] N. Brown *et al.*, "Managing technical debt in software-reliant systems," in *Proc. FSE/SDP Workshop Future Softw. Eng. Res. (FoSER '10)*, Santa Fe, NM, USA, 2010, pp. 47–52.
- [19] Z. Li, P. Avgeriou, and P. Liang, "A systematic mapping study on technical debt and its management," *J. Syst. Softw.*, vol. 101, pp. 193–220, 2015.
- [20] T. Besker, A. Martini, and J. Bosch, "Managing architectural technical debt: A unified model and systematic literature review," *J. Syst. Softw.*, vol. 135, pp. 1–16, 2018.
- [21] E. Tom, A. Aurum, and R. Vidgen, "An exploration of technical debt," *J. Syst. Softw.*, vol. 86, no. 6, pp. 1498–1516, 2013.
- [22] J. Bogner, R. Verdecchia, and I. Gerostathopoulos, "Characterizing technical debt and antipatterns in AI-based systems: A systematic mapping study," in *Proc. IEEE/ACM Int. Conf. Tech. Debt (TechDebt)*, Madrid, Spain, 2021, pp. 64–73.
- [23] N. S. R. Alves, T. S. Mendes, M. G. de Mendonça, R. O. Spínola, F. Shull, and C. Seaman, "Identification and management of technical debt: A systematic mapping study," *Inf. Softw. Technol.*, vol. 70, pp. 100–121, 2016.
- [24] S. Moreschini *et al.*, "The evolution of technical debt from DevOps to Generative AI: A multivocal literature review," *J. Syst. Softw.*, vol. 231, 2026, Art. no. 112599.
- [25] M. Chen *et al.*, "Evaluating large language models trained on code," 2021, *arXiv:2107.03374*.
- [26] C. Bird *et al.*, "Taking flight with Copilot: Early insights and opportunities of AI-powered pair-programming tools," *Queue*, vol. 20, no. 6, pp. 35–57, 2022.
- [27] S. Peng, E. Kalliamvakou, P. Cihon, and M. Demirer, "The impact of AI on developer productivity: Evidence from GitHub Copilot," 2023, *arXiv:2302.06590*.
- [28] A. Ziegler *et al.*, "Productivity assessment of neural code completion," in *Proc. 6th ACM SIGPLAN Int. Symp. Mach. Program. (MAPS 2022)*, San Diego, CA, USA, 2022, pp. 21–29.
- [29] P. Vaithilingam, T. Zhang, and E. L. Glassman, "Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models," in *Ext. Abstr. CHI Conf. Human Factors Comput. Syst. (CHI EA '22)*, New Orleans, LA, USA, 2022.
- [30] H. Pearce, B. Ahmad, B. Tan, B. Dolan-Gavitt, and R. Karri, "Asleep at the keyboard? Assessing the security of GitHub Copilot's code contributions," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Francisco, CA, USA, 2022, pp. 754–768.
- [31] N. Perry, M. Srivastava, D. Kumar, and D. Boneh, "Do users write more insecure code with AI assistants?" in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS '23)*, Copenhagen, Denmark, 2023, pp. 2785–2799.
- [32] G. Sandoval, H. Pearce, T. Nys, R. Karri, S. Garg, and B. Dolan-Gavitt, "Lost at C: A user study on the security implications of large language model code assistants," in *Proc. 32nd USENIX Secur. Symp. (USENIX Security 23)*, Anaheim, CA, USA, 2023.
- [33] A. Moradi Dakhel, V. Majdinasab, A. Nikanjam, F. Khomh, M. C. Desmarais, and Z. M. Jiang, "GitHub Copilot AI pair programmer: Asset or liability?" *J. Syst. Softw.*, vol. 203, 2023, Art. no. 111734.

- [34] A. Mastropaolo, L. Pascarella, E. Guglielmi, M. Ciniselli, S. Scalabrino, R. Oliveto, and G. Bavota, "On the robustness of code generation techniques: An empirical study on GitHub Copilot," in *Proc. 45th Int. Conf. Softw. Eng. (ICSE '23)*, Melbourne, Australia, 2023, pp. 2149–2160.
- [35] B. Yetiştiren, I. Özsoy, M. Ayerdem, and E. Tüzün, "Evaluating the code quality of AI-assisted code generation tools: An empirical study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT," 2023, *arXiv:2304.10778*.
- [36] S. Barke, M. B. James, and N. Polikarpova, "Grounded Copilot: How programmers interact with code-generating models," *Proc. ACM Program. Lang.*, vol. 7, no. OOPSLA1, pp. 85–111, 2023.
- [37] V. R. Basili, G. Caldiera, and H. D. Rombach, "Goal question metric paradigm," in *Encyclopedia of Software Engineering*. New York, NY, USA: Wiley, 1994, pp. 528–532.
- [38] A. Ampatzoglou, S. Bibi, P. Avgeriou, and A. Chatzigeorgiou, "Guidelines for managing threats to validity of secondary studies in software engineering," in *Contemporary Empirical Methods in Software Engineering*. Cham, Switzerland: Springer, 2020, pp. 415–441.
- [39] D. Spencer, *Card Sorting: Designing Usable Categories*, 1st ed. Brooklyn, NY, USA: Rosenfeld Media, 2009.