



ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΠΛΗΡΟΦΟΡΙΑΚΑ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

Interplay between AI and software testing

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ταταργιώτη Δημητρίου

Επιβλέπων: Κρητικός Κυριάκος

Μέλη εξεταστικής επιτροπής: Γιάννης Χαραλαμπίδης, Παναγιώτης Συμεωνίδης

Σάμος, Φεβρουάριος 2026

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή Κρητικό Κυριάκο για την ποιοτική καθοδήγηση και την άριστη συνεργασία, καθώς και την οικογένειά μου για την αμέριστη αγάπη και στήριξή τους.

© 2026

του

ΤΑΤΑΡΓΙΩΤΗ ΔΗΜΗΤΡΙΟΥ

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

Περιεχόμενα

ΠΕΡΙΛΗΨΗ.....	13
ABSTRACT.....	14
1. ΕΙΣΑΓΩΓΗ.....	15
1.1 ΠΛΑΙΣΙΟ ΕΡΓΑΣΙΑΣ	15
1.2 ΑΝΤΙΚΕΙΜΕΝΟ ΕΡΓΑΣΙΑΣ.....	16
1.3 ΔΟΜΗ ΑΝΑΦΟΡΑΣ.....	17
2. ΕΝΝΟΙΟΛΟΓΙΚΗ ΘΕΜΕΛΙΩΣΗ	18
2.1 ΚΥΚΛΟΣ ΖΩΗΣ ΑΝΑΠΤΥΞΗΣ ΛΟΓΙΣΜΙΚΟΥ	18
2.1.1 ΟΡΙΣΜΟΣ	18
2.1.2 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ ΚΥΚΛΟΥ ΖΩΗΣ ΑΝΑΠΤΥΞΗΣ ΛΟΓΙΣΜΙΚΟΥ	18
2.1.3 ΜΟΝΤΕΛΑ ΑΝΑΠΤΥΞΗΣ ΛΟΓΙΣΜΙΚΟΥ	21
2.2 ΔΟΚΙΜΗ ΛΟΓΙΣΜΙΚΟΥ	23
2.2.1 ΟΡΙΣΜΟΣ & ΣΚΟΠΟΣ	23
2.2.2 ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ	24
2.2.3 ΕΙΔΗ ΔΟΚΙΜΗΣ ΛΟΓΙΣΜΙΚΟΥ	26
2.2.4. ΚΥΡΙΟΙ ΤΥΠΟΙ ΔΟΚΙΜΗΣ ΛΟΓΙΣΜΙΚΟΥ	28
2.3 ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ.....	31
2.3.1 ΟΡΙΣΜΟΣ	31
2.3.2 ΕΙΔΗ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ	31
2.3.3. ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΣΥΣΤΗΜΑΤΩΝ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ.....	31
3. ΜΕΘΟΔΟΛΟΓΙΑ ΑΝΑΣΚΟΠΗΣΗΣ	32
3.1 ΣΤΟΧΟΣ ΚΑΙ ΕΡΕΥΝΗΤΙΚΑ ΕΡΩΤΗΜΑΤΑ.....	32
3.2 ΣΤΡΑΤΗΓΙΚΗ ΑΝΑΖΗΤΗΣΗΣ	32
3.3 ΦΙΛΤΡΑΡΙΣΜΑ ΕΠΙΛΕΓΜΕΝΗΣ ΒΙΒΛΙΟΓΡΑΦΙΑΣ.....	36
3.3.1 ΚΡΙΤΗΡΙΑ ΕΝΤΑΞΗΣ/ΑΠΟΚΛΕΙΣΜΟΥ	36
3.3.2 ΚΡΙΤΗΡΙΑ ΠΟΙΟΤΗΤΑΣ	37
3.3.3 ΑΠΟΤΕΛΕΣΜΑΤΑ ΦΙΛΤΡΑΡΙΣΜΑΤΟΣ ΒΙΒΛΙΟΓΡΑΦΙΑΣ	37
4.ΑΝΑΛΥΣΗ ΑΡΘΡΩΝ ΑΝΑ ΚΑΤΕΥΘΥΝΣΗ ΣΥΝΕΙΣΦΟΡΑΣ	41
4.1 ΣΥΝΕΙΣΦΟΡΑ ΤΗΣ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ ΣΤΗ ΔΟΚΙΜΗ ΛΟΓΙΣΜΙΚΟΥ	41
4.1.2 ΕΝΝΟΙΟΛΟΓΙΚΟΣ ΧΑΡΤΗΣ & ΤΑΞΙΝΟΜΗΣΗ ΕΠΙΛΕΓΜΕΝΩΝ ΑΡΘΡΩΝ.....	42

4.1.3 ΑΝΑΛΥΣΗ ΣΥΝΕΙΣΦΟΡΩΝ.....	45
4.2. ΣΥΝΕΙΣΦΟΡΑ ΤΗΣ ΔΟΚΙΜΗΣ ΛΟΓΙΣΜΙΚΟΥ ΣΤΗΝ ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ.....	66
4.2.1 ΔΥΝΑΤΑ ΣΗΜΕΙΑ & ΑΔΥΝΑΜΙΕΣ	66
4.2.2 ΕΝΝΟΙΟΛΟΓΙΚΟΣ ΧΑΡΤΗΣ & ΤΑΞΙΝΟΜΗΣΗ ΕΠΙΛΕΓΜΕΝΩΝ ΑΡΘΡΩΝ.....	68
4.2.3 ΑΝΑΛΥΣΗ.....	72
4.3 ΠΕΡΙΛΗΨΗ.....	94
5. ΕΡΕΥΝΗΤΙΚΑ ΚΕΝΑ ΚΑΙ ΠΡΟΤΕΙΝΟΜΕΝΕΣ ΕΡΕΥΝΗΤΙΚΕΣ ΚΑΤΕΥΘΥΝΣΕΙΣ.....	96
5.1 ΕΓΓΕΝΕΙΣ ΑΔΥΝΑΜΙΕΣ ΤΩΝ LLM.....	97
5.2 ΠΛΗΡΟΤΗΤΑ ΣΥΝΟΛΩΝ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΜΕΤΡΙΚΩΝ	98
5.3 ΑΔΥΝΑΜΙΑ ΕΠΑΝΑΣΧΕΔΙΑΣΜΟΥ ΡΟΗΣ	98
5.4 ΜΕΤΡΙΚΕΣ ΚΑΛΥΨΗΣ ΩΣ ΑΠΟΔΕΙΞΗ ΛΕΙΤΟΥΡΓΙΚΗΣ ΟΡΘΟΤΗΤΑΣ ΜΟΝΤΕΛΩΝ	99
5.5 ΑΣΑΦΗΣ ΔΥΝΑΤΟΤΗΤΑ ΕΠΕΚΤΑΣΗΣ ΧΡΗΣΗΣ DNN ΩΣ TEST ORACLES ΣΕ ΕΥΡΥΤΕΡΑ ΠΛΑΙΣΙΑ	99
5.6 ΚΑΘΟΡΙΣΜΟΣ ΕΝΔΕΔΕΙΓΜΕΝΟΥ ΑΡΙΘΜΟΥ ΜΕΤΑΜΟΡΦΙΚΗΣ ΣΧΕΣΕΩΣ.....	100
5.7 ΕΣΤΙΑΣΗ ΣΤΗ ΔΙΑΓΝΩΣΗ ΚΑΙ ΟΧΙ ΣΤΗ ΔΙΟΡΘΩΣΗ	101
6. ΣΥΜΠΕΡΑΣΜΑΤΑ.....	102
6.1 ΣΥΝΕΙΣΦΟΡΑ	102
6.2 ΕΜΠΕΙΡΙΑ	102
6.3 ΑΝΑΛΥΣΗ ΟΡΡΟΤΗΤΕΣ - ΘΡΕΑΤΣ.....	102
7. ΒΙΒΛΙΟΓΡΑΦΙΑ.....	107

Λίστα Σχημάτων

Σχήμα 1. Ποσοστιαία κατανομή επιλεγμένων άρθρων με κριτήριο τη δημοσίευση σε περιοδικό ή συνέδριο	38
Σχήμα 2. Ποσοστιαία κατανομή επιλεγμένων άρθρων ανά εκδότη	39
Σχήμα 3. Κατανομή άρθρων ανά έτος δημοσίευσης	40
Σχήμα 4. Εννοιολογικός χάρτης AI in software testing	43
Σχήμα 5. Κατανομή άρθρων AI στη δοκιμή λογισμικού ανά κυρία κατεύθυνση συνεισφοράς.....	45
Σχήμα 6. Εννοιολογικός χάρτης software testing in AI	69
Σχήμα 7. Κατανομή άρθρων software testing in AI ανά κυρία κατεύθυνση συνεισφοράς	72

Λίστα Πινάκων

Πίνακας 1. Λέξεις-κλειδιά και αιτιολόγηση της χρήσης τους στη μεθοδολογία ανασκόπησης	33
Πίνακας 2. Ανάλυση δυνατών σημείων-αδυναμιών στο πεδίο εφαρμογής της ΑΙ στη δοκιμή λογισμικού	41
Πίνακας 3. Πίνακας ταξινόμησης των προσεγγίσεων ΑΙ με βάση τον αντίστοιχο εννοιολογικό χάρτη	43
Πίνακας 4. Ανάλυση δυνατών σημείων & αδυναμιών στο πεδίο εφαρμογής της δοκιμής λογισμικού στην ΑΙ	68
Πίνακας 5. Πίνακας ταξινόμησης προσεγγίσεων software testing in ΑΙ με βάση τον αντίστοιχο εννοιολογικό χάρτη.....	69
Πίνακας 6. Ερευνητικά κενά και αντίστοιχες προτεινόμενες ερευνητικές κατευθύνσεις.....	96
Πίνακας 7. Ανάλυση ευκαιριών & απειλών της τεχνητής νοημοσύνης στη δοκιμή λογισμικού	103
Πίνακας 8. Ανάλυση ευκαιριών & απειλών δοκιμής λογισμικού στην τεχνητή νοημοσύνη.....	103

Ακρωνύμια

AI	Artificial Intelligence
ANN	Artificial Neural Network, Approximate Nearest Neighbor
APFD	Average Percentage of Faults Detected
API	Application Programming Interface
CC	Coincidental Correct
CGF	Coverage-Guided Fuzzing
CI/CD	Continuous Integration/ Continuous Delivery
CNN	Convolutated Neural Network
DL	Deep Learning
DNN	Deep Neural Network
FCCI	Fuzzy Coincidental Correct Identification
GK-GRU	Gini Kernel–Gated Recurrent Unit
GUI	Graphical User Interface
IREP	Incremental Reduced Error Pruning
LLM	Large Language Model
MC/DC	Modified Condition/ Decision Coverage
ML	Machine Learning
MR	Metamorphic Relations
MT	Metamorphic Testing
NLP	Natural Language Processing
PBT	Property Based Testing
QA	Question Answering
RE	Relation Extraction

SAS	State Activity Score
SDLC	Software Development Life Cycle
SLR	Systematic Literature Review
SMOTE	Synthetic Minority Over-sampling Technique
STD	State Transition Diagram
SUT	System Under Test
TCP	Test Case Prioritization
TQA	Text Question Answering
UML	Unified Modeling Language
V & V	Verification & Validation
VQA	Video Question Answering
WASPAS	Weighted Arithmetic Sum Product Assessment
XP	Extreme Programming
Z-BES	ZScore–Bald EagleSearch

ΠΕΡΙΛΗΨΗ

Η έλευση της τεχνητής νοημοσύνης έχει επιφέρει νέες ευκαιρίες και προκλήσεις στον τομέα της δοκιμής λογισμικού. Η αξιοποίηση συστημάτων και προσεγγίσεων τεχνητής νοημοσύνης φαντάζει πολλά υποσχόμενη για την επίλυση παραδοσιακών προβλημάτων της δοκιμής λογισμικού, όπως είναι η εξάρτηση από την ανθρώπινη εμπειρία, οι συχνά μεταβαλλόμενες απαιτήσεις των ενδιαφερόμενων μερών και ο χρόνος, η προσπάθεια και το κόστος που απαιτείται για τη διενέργεια των δοκιμών. Από την άλλη μεριά, η υπερβολική εμπιστοσύνη σε συστήματα τεχνητής νοημοσύνης για τη διεξαγωγή δοκιμών λογισμικού χωρίς να υπάρχουν αξιόπιστοι μηχανισμοί ελέγχου και κατά συνέπεια διασφάλισης της ποιότητάς τους φαντάζει απερίσκεπτη.

Η συγκεκριμένη μελέτη φιλοδοξεί, μέσω συστηματικής βιβλιογραφικής ανασκόπησης της σύγχρονης βιβλιογραφίας, να αναδείξει τη συνεισφορά της τεχνητής νοημοσύνης στη δοκιμή λογισμικού καθώς και τη συνεισφορά εφαρμογής μεθόδων δοκιμής λογισμικού για την αξιολόγηση και βελτίωση συστημάτων τεχνητής νοημοσύνης, ενώ παρέχει δυνατά σημεία και περιορισμούς των σύγχρονων σχετικών προσεγγίσεων, ερευνητικά κενά και προτεινόμενες ερευνητικές κατευθύνσεις για την αντιμετώπιση αυτών.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ

Τεχνητή Νοημοσύνη – Δοκιμή Λογισμικού – Βιβλιογραφική Ανασκόπηση – Προκλήσεις – Ερευνητικά Κενά – Ερευνητικές Κατευθύνσεις – Ανάλυση SWOT

ABSTRACT

The arrival of artificial intelligence (AI) has brought new opportunities and challenges in the field of software testing. AI-based systems and approaches seems promising to address some of the key, traditional problems of software testing, such as dependency from human experience, the variable requirements of the stakeholders and the time, effort and cost required for testing. On the other hand, trusting these AI-based systems to conduct software testing without utilizing reliable mechanisms of testing and quality assurance of such systems seems reckless.

This study aspires, through a systematic literature review of the modern relevant literature, to highlight the contribution of AI in software testing and the contribution of the application of software testing techniques in evaluating and improving AI-based systems, as well as to supply the main strengths and limitations of modern relevant approaches, core research gaps and corresponding research directions to address them.

KEYWORDS

Artificial Intelligence – Software Testing – Literature Review–Challenges – Research Gaps – Research Directions – SWOT Analysis

1. ΕΙΣΑΓΩΓΗ

1.1 ΠΛΑΙΣΙΟ ΕΡΓΑΣΙΑΣ

Η δοκιμή λογισμικού αποτελεί βασική δραστηριότητα στη μηχανική λογισμικού και στοχεύει στη διασφάλιση της αξιοπιστίας, της ακρίβειας καθώς και της ποιότητας των συστημάτων λογισμικού. Παρ' όλα αυτά, τα συστήματα λογισμικού έχουν γίνει πιο περίπλοκα τα τελευταία χρόνια λόγω των περισσότερων αναγκών των πελατών και των συνεχών μεταβολών στις απαιτήσεις τους, τον σκληρό ανταγωνισμό στην αγορά αλλά και λόγω της έλευσης νέων τεχνολογιών. Αυτό έχει οδηγήσει σε μια σειρά από διαρκείς προκλήσεις για τις παραδοσιακές δοκιμές λογισμικού, όπως το υψηλό κόστος δοκιμών, η κακή κάλυψη των δοκιμών, η καθυστερημένη ανίχνευση σφαλμάτων σε προχωρημένο στάδιο ανάπτυξης και η υπερβολική εξάρτηση από την ανθρώπινη εμπειρία. Όλες αυτές οι δυσκολίες οδηγούν σε μειωμένη αποτελεσματικότητα των δοκιμών και σε μεγαλύτερη πιθανότητα εισαγωγής ελαττωμάτων σε περιβάλλοντα παραγωγής. Επιπλέον, οι παραδοσιακές μέθοδοι αυτοματισμού δοκιμών, αν και χρήσιμες, βασίζονται κυρίως σε κανόνες και απαιτούν πολλή χειροκίνητη εργασία στο σχεδιασμό δοκιμών, τη συντήρηση και την επαναπροσαρμογή τους στις αλλαγές του συστήματος.

Η τεχνητή νοημοσύνη, σε αυτήν την περίπτωση, αποτελεί μια πιθανή ευκαιρία για την υπέρβαση ορισμένων περιορισμών των συμβατικών δοκιμών λογισμικού. Με τη βοήθεια της μηχανικής μάθησης, της βαθιάς μάθησης και άλλων μεθόδων που βασίζονται στην Τεχνητή Νοημοσύνη, η διαδικασία δοκιμών λογισμικού μπορεί να βασίζεται περισσότερο σε δεδομένα, να είναι προγνωστική και περισσότερο «αυτόνομη». Οι λύσεις που βασίζονται στην Τεχνητή Νοημοσύνη μπορούν να βελτιώσουν την κάλυψη των δοκιμών, να ελαχιστοποιήσουν την ανθρώπινη παρέμβαση και να βελτιστοποιήσουν την εκτέλεση των δοκιμών. Αυτές οι στρατηγικές παρέχουν νέες προσεγγίσεις σχετικά με το πώς μπορούν να αντιμετωπιστούν μακροχρόνια προβλήματα στον τομέα των δοκιμών λογισμικού.

Αντίστοιχα, ελλοχεύουν κίνδυνοι λόγω της υπερβολικής εμπιστοσύνης στην τεχνητή νοημοσύνη, καθώς πρόκειται μεν για μία ώριμη τεχνολογία όμως αυτό δεν ισχύει για όλα τα πεδία της, ειδικότερα τα αναδυόμενα, ενώ και η λειτουργική ωριμότητα των συστημάτων τεχνητής νοημοσύνης ποικίλει. Συνεπώς, θα πρέπει να εφαρμοστούν οι αρχές και οι τεχνικές της

δοκιμής λογισμικού για τη διασφάλιση της ποιότητας και την ανίχνευση ελαττωμάτων των συστημάτων τεχνητής νοημοσύνης. Η ανίχνευση ελαττωμάτων και η διόρθωση αυτών είναι θεμελιώδους σημασίας, καθώς η τεχνητή νοημοσύνη ενσωματώνεται σταδιακά σε όλες τις εκφάνσεις της σύγχρονης ζωής και δεν περιορίζεται στη μηχανική και τη δοκιμή λογισμικού.

1.2 ΑΝΤΙΚΕΙΜΕΝΟ ΕΡΓΑΣΙΑΣ

Η μελέτη που επιτελέστηκε στο πλαίσιο της τρέχουσας διπλωματικής εργασίας έχει σημαντική αξία, καθώς παρέχει μια συστηματική ανάλυση της διάδρασης μεταξύ τεχνητής νοημοσύνης και δοκιμών λογισμικού. Με βάση μια διεξοδική ανασκόπηση της διαθέσιμης βιβλιογραφίας, η εργασία αξιολογεί τη χρήση μεθόδων Τεχνητής Νοημοσύνης σε διάφορες λειτουργίες δοκιμών, απαντά στο ερώτημα κατά πόσο η Τεχνητή Νοημοσύνη μπορεί να λύσει τα τρέχοντα προβλήματα δοκιμών, ενώ παρέχει τις κύριες σχετικές ερευνητικές τάσεις που αναδύονται. Επιπλέον, καθορίζει ποιες μέθοδοι δοκιμών είναι οι ενδεδειγμένες για την αξιολόγηση νέων εργαλείων τεχνητής νοημοσύνης. Τέλος, εντοπίζει ανεπίλυτα ζητήματα, ερευνητικά κενά και περιορισμούς που παρατηρούνται στις υπάρχουσες προσεγγίσεις εφαρμογής τεχνικών τεχνητής νοημοσύνης στη δοκιμή λογισμικού και εφαρμογής τεχνικών δοκιμής λογισμικού σε συστήματα τεχνητής νοημοσύνης.

Σκοπός, λοιπόν, της παρούσας εργασίας είναι η μελέτη της αλληλεπίδρασης μεταξύ της τεχνητής νοημοσύνης και της δοκιμής λογισμικού, η συνεισφορά του ενός κλάδου στον άλλο αλλά και ο προσδιορισμός πιθανών ερευνητικών κενών και μελλοντικών προκλήσεων, αναφορικά με την εφαρμογή της τεχνητής νοημοσύνης στη δοκιμή λογισμικού, αλλά και με την εφαρμογή αποδοτικών μεθόδων δοκιμής λογισμικού σε συστήματα τεχνητής νοημοσύνης, καθώς επίσης και η πρόταση αντίστοιχων ερευνητικών κατευθύνσεων.

Αυτή η μελέτη διαθέτει δύο κύρια δυνατά σημεία. Πρώτον, αποτελεί μια καλά οργανωμένη ανασκόπηση των μεθόδων δοκιμών λογισμικού που βασίζονται στην Τεχνητή Νοημοσύνη αλλά και των μεθόδων και τεχνικών που ενδείκνυνται για την αποτελεσματική δοκιμή και βελτίωση της επίδοσης συστημάτων τεχνητής νοημοσύνης. Δεύτερον, εξετάζει κριτικά την τρέχουσα βιβλιογραφία, πέρα από τις τεχνικές και λειτουργικές δυνατότητες των εργαλείων, ανακαλύπτει ποια είναι τα κενά στη βιβλιογραφία και προσδιορίζει πού πρέπει να κατευθυνθεί η έρευνα σε αυτόν τον ταχέως αναπτυσσόμενο τομέα. Συνεπώς, η μελέτη μας συνιστά μια ενοποιημένη πηγή

πληροφοριών για ερευνητές και επαγγελματίες ενώ τους κατευθύνει ερευνητικά με σκοπό να βελτιώσουν τα συστήματα και πρωτότυπά τους.

1.3 ΔΟΜΗ ΑΝΑΦΟΡΑΣ

Η αναφορά χωρίζεται στα εξής επόμενα κεφάλαια. Το κεφάλαιο με τίτλο «Εννοιολογική Θεμελίωση» ασχολείται με την επεξήγηση των βασικών εννοιών που απαντώνται στη βιβλιογραφία που μελετήθηκε. Το κεφάλαιο με τίτλο «Μεθοδολογία Ανασκόπησης» περιγράφει αναλυτικά τη μεθοδολογία βιβλιογραφικής ανασκόπησης που ακολουθήθηκε. Το κεφάλαιο με τίτλο «Ανάλυση Άρθρων ανά Κατεύθυνση Συνεισφοράς -» έχει σκοπό με αφετηρία τη μελέτη των άρθρων που συλλέχθηκαν, να χαρτογραφήσει το πεδίο της συνεισφοράς της τεχνητής νοημοσύνης στη δοκιμή λογισμικού και της συνεισφοράς της δοκιμής λογισμικού στην τεχνητή νοημοσύνη, να ομαδοποιήσει τα άρθρα ανά κύρια κατεύθυνση συνεισφοράς καθώς και να αναδείξει τα δυνατά σημεία και τις αδυναμίες των δύο αυτών κατευθύνσεων συνεισφοράς /πεδίων. Το κεφάλαιο με τίτλο «Ερευνητικές Προκλήσεις και Προτεινόμενες Ερευνητικές Κατευθύνσεις» έχει σκοπό να αναδείξει τα ερευνητικά κενά που προκύπτουν από τη μελέτη της βιβλιογραφίας και την πρόταση αντίστοιχων ερευνητικών κατευθύνσεων για την αντιμετώπιση αυτών. Τέλος, το κεφάλαιο με τίτλο «Συμπεράσματα» αποσκοπεί στην ανάδειξη της συνεισφοράς της παρούσας μελέτης και της εμπειρίας που αποκομίσθηκε στο πλαίσιο εκπόνησης αυτής, καθώς επίσης και των σημαντικότερων συμπερασμάτων που προέκυψαν.

2. ΕΝΝΟΙΟΛΟΓΙΚΗ ΘΕΜΕΛΙΩΣΗ

Το παρόν κεφάλαιο στοχεύει στην οικοδόμηση του εννοιολογικού υποβάθρου που είναι απαραίτητο για τη κατανόηση της μελέτης και των συνεισφορών της. Παρατίθενται ορισμοί και αναλύσεις για θεμελιώδεις έννοιες όπως του κύκλου ζωής λογισμικού, της δοκιμής λογισμικού και της τεχνητής νοημοσύνης που είναι άκρως σχετικές με το αντικείμενο της μελέτης. Η δομή του κεφαλαίου περιλαμβάνει ενότητες όπου η κάθε μία εστιάζει σε μια βασική έννοια από αυτές που προαναφέρθηκαν.

2.1 ΚΥΚΛΟΣ ΖΩΗΣ ΑΝΑΠΤΥΞΗΣ ΛΟΓΙΣΜΙΚΟΥ

2.1.1 ΟΡΙΣΜΟΣ

Ο Κύκλος Ζωής Ανάπτυξης Λογισμικού (Software Development Life Cycle, SDLC) είναι μια διαδικασία στη μηχανική λογισμικού, η οποία είναι κυκλική και οργανώνει κατάλληλα τις δραστηριότητες ανάπτυξης, οι οποίες περιλαμβάνουν την ανάλυση απαιτήσεων, τον σχεδιασμό, την υλοποίηση, την εγκατάσταση και τη συντήρηση συστημάτων λογισμικού. Μπορεί να οριστεί ως μια συλλογή συνδεδεμένων δραστηριοτήτων που αποσκοπούν στην παραγωγή ποιοτικών προϊόντων λογισμικού και ο χαρακτηρισμός ως «ποιοτικών» προκύπτει υπό το πρίσμα ότι αυτά ανταποκρίνονται σε συγκεκριμένες απαιτήσεις και ορθές πρακτικές. Επειδή ο SDLC δεν είναι ένα εφάπαξ γεγονός, τα συστήματα λογισμικού υφίστανται πολλούς κύκλους ανάπτυξης με τη ροή του χρόνου λόγω αλλαγών στις απαιτήσεις καθώς και την ανάγκη για διορθώσεις αστοχιών ή λειτουργικής βελτίωσης. Στη συνέχεια θα αναλύσουμε τις κύριες φάσεις/δραστηριότητες που απαρτίζουν τον SDLC και έπειτα θα προχωρήσουμε σε διάφορα γενικά μοντέλα ανάπτυξης λογισμικού, τα οποία επιτελούν τις δραστηριότητες αυτές με διαφορετικούς τρόπους ή ροές.

2.1.2 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ ΚΥΚΛΟΥ ΖΩΗΣ ΑΝΑΠΤΥΞΗΣ ΛΟΓΙΣΜΙΚΟΥ

Ανάλυση απαιτήσεων (Requirements Analysis): Η φάση της ανάλυσης απαιτήσεων θεωρείται ένα από τα πιο κρίσιμα στάδια του κύκλου ζωής ανάπτυξης λογισμικού, καθώς παρέχει τη βάση πάνω στην οποία χτίζονται όλες οι επόμενες δραστηριότητες ανάπτυξης. ατελείς ή ασαφείς απαιτήσεις μπορούν να οδηγήσουν σε σημαντικές προσπάθειες επανασχεδιασμού και εκ νέου υλοποίησης του λογισμικού σε επόμενο κύκλο ανάπτυξης, σε μεταγενέστερο στάδιο της διαδικασίας, υπογραμμίζοντας την ανάγκη για σαφή και πλήρη ανάλυση από την αρχή. Αυτή η

φάση περιλαμβάνει τη συστηματική ανίχνευση, τεκμηρίωση και επικύρωση των αναγκών/απαιτήσεων των ενδιαφερόμενων μερών, προκειμένου να διασφαλιστεί ότι το τελικό σύστημα ευθυγραμμίζεται με τους επιχειρησιακούς στόχους και τις προσδοκίες των χρηστών (Coronato, 2018). Επιπρόσθετα, η ανάλυση απαιτήσεων όχι μόνο προσδιορίζει τις λειτουργικές και μη λειτουργικές απαιτήσεις, αλλά προσδιορίζει και τους περιορισμούς του συστήματος, τις διεπαφές καθώς επίσης και σαφή κριτήρια αποδοχής (TutorialsPoint, n.d).

Σχεδιασμός (Design): Οι απαιτήσεις μετατρέπονται σε επίσημα μοντέλα συστήματος που περιγράφουν τη δομή και τη συμπεριφορά του. Υπάρχουν δομικά διαγράμματα που καλύπτουν τη δομή του συστήματος (component diagram - αρχιτεκτονική, class diagram - βασικές οντότητες) καθώς και συμπεριφορικά διαγράμματα που καλύπτουν την συμπεριφορά του συστήματος (use-case diagrams, sequence diagrams, state diagrams, activity diagrams) σε διαφορετικά επίπεδα. Ο συνδυασμός των μοντέλων χρησιμοποιείται για να περιγράψει την αλληλεπίδραση όλων των μερών του συστήματος, τη ροή ελέγχου και δεδομένων στο σύστημα, καθώς και την απόκριση του συστήματος σε συμβάντα. Εξασφαλίζεται έτσι περιορισμός περιττής ή λανθασμένης εργασίας, καθώς υπάρχει σαφής μοντελοποίηση που βοηθά στην ανίχνευση τυχόν ασυνεπειών και κινδύνων σε πολύ πρώιμα στάδια πριν από οποιαδήποτε υλοποίηση. Τίθεται επίσης ζήτημα κατάλληλης ενοποίησης των συστατικών του συστήματος, καθώς αν δεν σχεδιαστούν σωστά οι διεπαφές των συστατικών, αυτά δεν θα μπορούν να αλληλεπιδρούν. Επιπλέον, αν δεν προδιαγραφεί η σωστή συμπεριφορά των συστατικών, δεν θα ικανοποιούνται σωστά οι απαιτήσεις.

Υλοποίηση (Implementation): Επικεντρώνεται στη μετατροπή του συστήματος σε λειτουργικά στοιχεία λογισμικού μέσω δομημένων και συστηματικών πρακτικών ανάπτυξης κώδικα. Περιλαμβάνει την επιλογή κατάλληλων γλωσσών προγραμματισμού, την εφαρμογή συνεπών προτύπων κωδικοποίησης και την ανάπτυξη κάθε επί μέρους συστατικών και των μονάδων τους σύμφωνα με τις εγκεκριμένες προδιαγραφές σχεδιασμού (TutorialsPoint, n.d.).

Επαλήθευση και Επικύρωση (Verification and Validation, V&V): Αποτελούν θεμελιώδεις δραστηριότητες διασφάλισης ποιότητας στον κύκλο ζωής ανάπτυξης λογισμικού. Η επαλήθευση εστιάζει στον έλεγχο του κατά πόσο το σύστημα υλοποιείται σωστά σύμφωνα με τις καθορισμένες προδιαγραφές, ενώ η επικύρωση εξετάζει αν το τελικό προϊόν ανταποκρίνεται στις πραγματικές ανάγκες των χρηστών. Οι δραστηριότητες δοκιμών αποτελούν βασικό μηχανισμό

υλοποίησης τόσο της επαλήθευσης όσο και της επικύρωσης. Μέσω συστηματικών ελέγχων και εκτελέσεων σεναρίων χρήσης, οι δοκιμές παρέχουν εμπειρικά τεκμήρια σχετικά με τη συμμόρφωση του συστήματος στις προδιαγραφές και με το βαθμό στον οποίο ικανοποιούνται οι ανάγκες των τελικών χρηστών. Συμπληρωματικά στην δραστηριότητα αυτή μπορούν να εφαρμόζονται και επιθεωρήσεις, οι οποίες εστιάζουν στον έλεγχο της ορθότητας όλων των τεχνουργημάτων που παράγονται στις προηγούμενες δραστηριότητες του κύκλου ζωής και του βαθμού συνέπειας μεταξύ τους, όπως τεκμηρίωση συστήματος, σχεδιαστικά διαγράμματα, καθώς και ο κώδικας, ακόμη και σε πρόωπη μορφή(όταν δηλαδή δεν έχει υλοποιηθεί κάποια ολοκληρωμένη έκδοση του συστήματος).

Παράδοση και Εγκατάσταση (Deployment): Περιλαμβάνει την παράδοση του πλήρως δοκιμασμένου συστήματος λογισμικού στους τελικούς χρήστες ή την διάταξή του στο περιβάλλον παραγωγής, διασφαλίζοντας ότι έχει διαμορφωθεί, εγκατασταθεί και ενεργοποιηθεί σύμφωνα με τις οργανωτικές και τεχνικές απαιτήσεις. Μπορεί να ακολουθεί διάφορες στρατηγικές μετάβασης/διάταξης (deployment strategies) (συμπεριλαμβανομένης της άμεσης μετάβασης (direct cut over ή “big bang”), της σταδιακής υλοποίησης (phased implementation), της πιλοτικής ανάπτυξης (pilot implementation) ή της παράλληλης λειτουργίας (parallel run)), καθεμία από τις οποίες επιλέγεται με βάση την πολυπλοκότητα του συστήματος, το απαιτούμενο επίπεδο ανοχής κινδύνου και τους λειτουργικούς περιορισμούς (Coronato, 2018).

Οι δραστηριότητες της φάσης συχνά περιλαμβάνουν τη δημιουργία και διαμόρφωση του περιβάλλοντος, τη μετεγκατάσταση δεδομένων, την εκπαίδευση των χρηστών και τη δημιουργία λειτουργικής τεκμηρίωσης, ώστε οι τελικοί χρήστες και οι διαχειριστές να μπορούν να αλληλεπιδρούν αποτελεσματικά με το σύστημα. Ο σωστός συντονισμός μεταξύ προγραμματιστών, διαχειριστών συστημάτων και ενδιαφερόμενων μερών είναι απαραίτητος σε αυτό το στάδιο για την ελαχιστοποίηση του χρόνου διακοπής λειτουργίας και τη διασφάλιση ομαλής μετάβασης. Η επιτυχία όχι μόνο καθιστά το σύστημα διαθέσιμο στους χρήστες, αλλά και θέτει τα θεμέλια για τη συνεχή παρακολούθηση, υποστήριξη και συντήρησή του (TutorialsPoint, n.d.,).

Συντήρηση (Maintenance): Η φάση συντήρησης ξεκινά μετά την επιτυχή εγκατάσταση του συστήματος και επικεντρώνεται στη διασφάλιση της σωστής, αποτελεσματικής και ασφαλούς λειτουργίας του λογισμικού καθ' όλη τη διάρκεια του κύκλου ζωής του. Οι δραστηριότητες

συντήρησης περιλαμβάνουν τη διορθωτική συντήρηση (επιδιόρθωση ελαττωμάτων που εντοπίζονται κατά τη λειτουργία), την προσαρμοστική συντήρηση (τροποποίηση του συστήματος για την προσαρμογή σε αλλαγές στο περιβάλλον ή στις επιχειρηματικές διαδικασίες) και την τελειοποιητική συντήρηση (βελτίωση της απόδοσης ή προσθήκη νέων λειτουργιών) (Coronato, 2018). Επιπροσθέτως, είναι αξιοσημείωτη η σημασία της προληπτικής συντήρησης, η οποία περιλαμβάνει την προληπτική αναγνώριση πιθανών βλαβών, την ενημέρωση της τεκμηρίωσης και την εφαρμογή ενημερώσεων ασφαλείας για τη μείωση μελλοντικών κινδύνων. Επειδή το λογισμικό συχνά εξελίσσεται πολύ μετά την αρχική του κυκλοφορία, η συντήρηση συνήθως καταναλώνει σημαντικό μέρος του συνολικού κόστους και της προσπάθειας ανάπτυξης, ειδικότερα για συστήματα λογισμικού μακράς διάρκειας. Η αποτελεσματική συντήρηση εξασφαλίζει τη μακροπρόθεσμη σταθερότητα του λογισμικού, διατηρεί ή αυξάνει την ικανοποίηση των χρηστών και υποστηρίζει την ικανότητα του συστήματος να προσαρμόζεται στις μεταβαλλόμενες οργανωτικές/επιχειρησιακές ανάγκες (TutorialsPoint, n.d., όπως αναφέρεται στους Yas et al., 2023).

2.1.3 ΜΟΝΤΕΛΑ ΑΝΑΠΤΥΞΗΣ ΛΟΓΙΣΜΙΚΟΥ

Μοντέλο ανάπτυξης λογισμικού: είναι ένα μοντέλο της διαδικασίας ανάπτυξης (software process model) που καθορίζει τη διάταξη των σταδίων ανάπτυξης και εξέλιξης του λογισμικού και τα κριτήρια μετάβασης από ένα στάδιο στο επόμενο (π.χ. κριτήρια ολοκλήρωσης του τρέχοντος σταδίου και κριτήρια εισόδου/επιλογής για το επόμενο) (Boehm, 1988). Στην ανάπτυξη λογισμικού, έχουν σχηματισθεί και χρησιμοποιηθεί διάφορα μοντέλα ανάπτυξης για την εξισορρόπηση της ευελιξίας και των πιθανών κινδύνων κατά τη διαδικασία ανάπτυξης, το καθένα με τα δικά του πλεονεκτήματα και μειονεκτήματα. Συνήθως, κάθε εταιρεία λογισμικού εφαρμόζει τα δικά της σχετικά μοντέλα. Αυτά συνήθως έχουν προκύψει από θεωρητικά/γενικά μοντέλα που έχουν προταθεί στην βιβλιογραφία, τα οποία ονομάζουμε ως παραδοσιακά.

Τα παραδοσιακά μοντέλα που βασίζονται σε πλάνο (plan-based models), όπως το μοντέλο του Καταρράκτη (Waterfall), ακολουθούν μια καλά καθορισμένη διαδοχική ροή εκτέλεσης των διαδικασιών ανάπτυξης σε κάθε κύκλο ανάπτυξης του λογισμικού, η οποία παρέχει προβλεψιμότητα, εκτενή τεκμηρίωση και ισχυρή ιχνηλασιμότητα, αλλά συχνά υποφέρει από χαμηλή ανταπόκριση στις αλλαγές (Yas et al., 2023; Mishra & Alzoubi, 2023). Το μοντέλο V (Sommerville, 2016), μια εξειδικευμένη παραλλαγή του Waterfall, αντιστοιχίζει κάθε στάδιο

ανάπτυξης με αντίστοιχες δραστηριότητες επαλήθευσης και επικύρωσης. Οι δραστηριότητες αυτές διαφοροποιούνται ανάλογα με τη φύση των παραδοτέων κάθε φάσης. Στα πρώιμα στάδια περιλαμβάνουν κυρίως ανασκοπήσεις και επιθεωρήσεις απαιτήσεων και σχεδιαστικών τεκμηρίων (όπως διαγράμματα UML), ενώ στα μεταγενέστερα επικεντρώνονται σε συστηματικές δοκιμές του υλοποιημένου συστήματος. Το μοντέλο θεωρείται ιδιαίτερα κατάλληλο για συστήματα κρίσιμης ασφαλείας, διατηρώντας ωστόσο την χαρακτηριστική «ακαμψία» του Waterfall (Mishra & Alzoubi, 2023).

Από την άλλη πλευρά, τα επαναληπτικά και σταδιακά μοντέλα (incremental t models) χωρίζουν την ανάπτυξη σε επαναλαμβανόμενους κύκλους (όπου σε κάθε κύκλο παράγεται μια επαύξηση του συστήματος), επιτρέποντας στις ομάδες να παραδίδουν λειτουργικά υποσύνολα (δηλ. μια σύνθεση όλων των επαυξήσεων που έχουν παραχθεί) νωρίς και να κατασκευάζουν/βελτιώνουν επαναληπτικά το σύστημα. Το μοντέλο σπειροειδούς ανάπτυξης (Spiral) βασίζεται περαιτέρω σε αυτό, ενσωματώνοντας ρητή ανάλυση κινδύνου σε κάθε επανάληψη, γεγονός που το καθιστά ιδανικό για μεγάλα έργα υψηλού κινδύνου, αν και μπορεί να αυξήσει τα διοικητικά έξοδα και το κόστος (Mishra & Alzoubi, 2023). Αντίστοιχα, η εξελικτική προτυποποίηση (evolutionary prototyping) βασίζεται στη σταδιακή ωρίμανση ενός αρχικού λειτουργικού πρωτότυπου, το οποίο εξελίσσεται μέσω συνεχούς ανατροφοδότησης έως ότου μετασχηματιστεί στο τελικό σύστημα. Αυτές οι μέθοδοι λειτουργούν ιδιαίτερα καλά όταν οι απαιτήσεις είναι ασταθείς, η ανατροφοδότηση από τα ενδιαφερόμενα μέρη είναι συχνή και η ταχύτητα αποτελεί προτεραιότητα. Ωστόσο, μπορούν να δημιουργήσουν περιορισμούς ως προς τη διατήρηση επαρκούς τεκμηρίωσης και τον μακροπρόθεσμο αρχιτεκτονικό σχεδιασμό (Mishra & Alzoubi, 2023).

Οι περιορισμοί αυτοί μπορούν να μετριαστούν μέσω της ενσωμάτωσης πρακτικών αρχιτεκτονικής καθοδήγησης, μηχανισμών κλιμάκωσης και ελεγχόμενης τεκμηρίωσης, επιτρέποντας την εφαρμογή ευέλικτων προσεγγίσεων ακόμη και σε μεγάλης κλίμακας ή μακράς διάρκειας συστήματα. Πιο πρόσφατα, οι πρακτικές DevOps¹ και οι αγωγοί CI/CD² έχουν

¹ Το DevOps είναι ένα σύνολο τεχνικών (σχεδιασμός, υλοποίηση, εγκατάσταση, δοκιμή και επίβλεψη) για την ταχεία και αξιόπιστη ανάπτυξη και δοκιμή ενημερώσεων λογισμικού, επιτρέποντας στους προγραμματιστές, τους χειριστές και τους δοκιμαστές να επικοινωνούν πιο αποτελεσματικά (Awad et al., 2023).

² Οι αρχές του DevOps υλοποιούνται επιχειρησιακά μέσω μηχανισμών συνεχούς ενσωμάτωσης, δοκιμών και παράδοσης, γνωστών ως αγωγοί CI/CD (Continuous Integration / Continuous Delivery), που επιτρέπουν τη συστηματική και επαναληπτική μετάβαση από την ανάπτυξη στη λειτουργία (Awad et al., 2023).

αποκτήσει ιδιαίτερη σημασία, μεταφέροντας την ευέλικτη φιλοσοφία πέρα από τα όρια της ανάπτυξης και καλύπτοντας ολόκληρη τη διαδρομή «*from code to production*». Αυτές οι πρακτικές μειώνουν σημαντικά το χρόνο που απαιτείται για την κυκλοφορία ενός προϊόντος στην αγορά, διευκολύνουν τις συχνές ενημερώσεις και ενισχύουν τη σταθερότητα του συστήματος, ιδιαίτερα σε περιβάλλοντα εγγενή ως προς υπολογιστικό νέφος (cloud-native) (Yas et al., 2023).

Τέλος, τα υβριδικά μοντέλα συνδυάζουν τα καλύτερα στοιχεία και των δύο προσεγγίσεων: σχεδιασμό και δομημένη ανάπτυξη στις αρχικές φάσεις (ανάλυση απαιτήσεων, σχεδιασμός) με Agile (ευέλικτη) εκτέλεση για υλοποίηση, επαλήθευση & επικύρωση και διάταξη. Οι υβριδικές μεθοδολογίες είναι πλεονεκτικές σε περιβάλλοντα όπου η κανονιστική συμμόρφωση, η συμμετοχή των ενδιαφερομένων μερών και ο κίνδυνος ποικίλλουν μεταξύ των φάσεων του έργου, και τα πλαίσια λήψης αποφάσεων (όπως τα δέντρα αποφάσεων) μπορούν να καθοδηγήσουν τις ομάδες στην επιλογή του καταλληλότερου συνδυασμού προσεγγίσεών τους (Mishra & Alzoubi, 2023).

2.2 ΔΟΚΙΜΗ ΛΟΓΙΣΜΙΚΟΥ

2.2.1 ΟΡΙΣΜΟΣ & ΣΚΟΠΟΣ

Η δοκιμή λογισμικού είναι η διαδικασία επαλήθευσης και επικύρωσης της σωστής συμπεριφοράς του συστήματος υπό δοκιμή (System Under Test, “SUT”) σύμφωνα με τις απαιτήσεις (Barr et al., 2015). Είναι συστηματική δραστηριότητα που διεξάγεται για να παράσχει σε όλα τα ενδιαφερόμενα μέρη πληροφορίες σχετικά με την συμπεριφορά και ποιότητα του συστήματος υπό δοκιμή. Συνήθως, ένας οργανισμός ανάπτυξης λογισμικού δαπανά το 30 - 40% των προσπαθειών για δοκιμές (Pressman, 1987) και πάνω από το 50% του συνολικού κόστους του έργου, καθώς η δοκιμή του συστήματος μπορεί να απαιτεί εξειδικευμένη εμπειρογνωμοσύνη, ειδικό εξοπλισμό, βαριά υποδομή αλλά και να επιτελείται περισσότερες από μία φορές κατά τη διάρκεια μιας διαδοχής κύκλων ανάπτυξης (Ramlar & Wolfmaier, 2006).

Ο κύριος στόχος των δοκιμών λογισμικού είναι η επικύρωση και επαλήθευση ενός προϊόντος λογισμικού ώστε να εξασφαλιστεί η βελτίωση της αξιοπιστίας και ποιότητάς του. Οι δοκιμές λογισμικού δεν γίνονται μόνο για τον εντοπισμό σφαλμάτων σε ένα προϊόν λογισμικού, αλλά λειτουργούν και ως μέτρο για να διασφαλιστεί πως το προϊόν λειτουργεί όπως αναμένεται. Η επαλήθευση (verification) συνεπάγεται τη διασφάλιση ότι ένα προϊόν λογισμικού πετυχαίνει το

στόχο του χωρίς σφάλματα ή ελαττώματα, έχει δηλαδή την αναμενόμενη δομή, συμπεριφορά και ποιότητα με βάση τις προδιαγραφές του. Η επικύρωση μπορεί να οριστεί ως η διαδικασία διασφάλισης πως ένα προϊόν λογισμικού πληροί τις πραγματικές απαιτήσεις, αλλά και προσδοκίες, των ενδιαφερόμενων μερών του. (Borah et al., 2022).

2.2.2 ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ

Οι έννοιες test case, test data και test oracle είναι θεμελιώδεις για την κατανόηση της φάσης δοκιμής λογισμικού. Συνεπώς, τις αναλύουμε περαιτέρω παρακάτω.

Περίπτωση δοκιμής (Test case): Καθορίζει τις συνθήκες και τα βήματα που απαιτούνται για τη δοκιμή και τον τρόπο επικύρωσης της αναμενόμενης συμπεριφοράς του συστήματος. Μπορεί να θεωρηθεί ως μία συνταγή με βάση την οποία κάποιος μπορεί να ελέγξει ένα στοιχείο ενός λογισμικού. Όταν αποκαλύπτει μια αστοχία, θεωρείται επιτυχής (successful). (Durelli et al., 2019).

Οι περιπτώσεις δοκιμής (test cases) συνήθως προκύπτουν ως παράγωγα των απαιτήσεων χρηστών (user requirements) και συστήματος (system requirements) καθώς και των προδιαγραφών σχεδίασης. Οι απαιτήσεις χρηστών (user requirements) περιγράφουν τους περιορισμούς που οφείλει να ικανοποιεί το σύστημα, εκφράζοντας τις ανάγκες των χρηστών και των ενδιαφερόμενων μερών. Οι απαιτήσεις συστήματος αποτυπώνουν τη συμπεριφορά του συστήματος σε κατάλληλο επίπεδο αφάιρεσης, συνήθως μη τεχνικό, προσδιορίζοντας τις λειτουργικές δυνατότητες και τις αλληλεπιδράσεις με το περιβάλλον. Οι προδιαγραφές σχεδίασης μεταβαίνουν στο τεχνικό επίπεδο, περιγράφοντας την αρχιτεκτονική, τα επιμέρους συστατικά και τον τρόπο υλοποίησης του συστήματος. Οι προδιαγραφές των περιπτώσεων δοκιμής περιλαμβάνουν:

- Τις προϋποθέσεις, οι οποίες περιγράφουν το περιβάλλον και την κατάσταση του συστήματος υπό δοκιμή πριν την εκτέλεση της περίπτωσης δοκιμής.
- Τα βήματα δοκιμής, τα οποία περιγράφουν τις ενέργειες που απαιτούνται για την εκτέλεση της περίπτωσης δοκιμής, όπως η προετοιμασία του περιβάλλοντος, η εισαγωγή των δεδομένων δοκιμής, η εκτέλεση του συστήματος ή της μονάδας και η σύγκριση των πραγματικών αποτελεσμάτων με τα αναμενόμενα. Τα βήματα αυτά παραμένουν

εννοιολογικά ίδια τόσο στις χειρωνακτικές όσο και στις αυτοματοποιημένες δοκιμές, διαφοροποιούμενα κυρίως ως προς τον βαθμό αυτοματοποίησης της εκτέλεσής τους.

- Τα αναμενόμενα αποτελέσματα, περιγράφουν τα προσδοκώμενα αποτελέσματα της εκτέλεσης της περίπτωσης δοκιμής. (Zubair et al., 2022)

Αφού εκτελεστεί η περίπτωση δοκιμής, προκύπτουν τα πραγματικά αποτελέσματα, που περιγράφουν τα αποτελέσματα της εκτέλεσης της περίπτωσης δοκιμής, τα οποία παράγονται από το υλοποιημένο σύστημα ή μέρη του ανάλογα με το είδος εκτελούμενης δοκιμής.

Δεδομένα δοκιμών (Test Data): Τα δεδομένα δοκιμών αναφέρονται στις πραγματικές τιμές και «διαμορφώσεις» (configurations)³ που χρησιμοποιούνται για την εκτέλεση αυτών των περιπτώσεων δοκιμών και, ως εκ τούτου, επηρεάζουν άμεσα την ακρίβεια, την πληρότητα και την αξιοπιστία των αποτελεσμάτων των δοκιμών. Ο συστηματικός σχεδιασμός των περιπτώσεων δοκιμών και η διαθεσιμότητα ποικίλων και αντιπροσωπευτικών δεδομένων δοκιμών είναι απαραίτητα για την επίτευξη επαρκούς κάλυψης σε διαφορετικά επίπεδα δοκιμών, συμπεριλαμβανομένων των δοκιμών μονάδας, ολοκλήρωσης, συστήματος και παλινδρόμησης (unit, integration, system and regression tests). Τα ανεπαρκή, μη αντιπροσωπευτικά ή κακής ποιότητας δεδομένα δοκιμών περιορίζουν την αποτελεσματικότητα της ανίχνευσης σφαλμάτων και ενδέχεται να αποκρύψουν υποκείμενα ελαττώματα που εκδηλώνονται μόνο υπό συγκεκριμένες περιβαλλοντικές συνθήκες ή ακραίες περιπτώσεις (Zubair et al., 2022). Η κατασκευή και η επιμέλεια των δεδομένων δοκιμών απαιτούν προσοχή ως προς την εγκυρότητα, τη μεταβλητότητα και την ιχνηλασιμότητα, καθώς τα σύγχρονα συστήματα βασίζονται όλο και περισσότερο σε εκτεταμένα σύνολα δεδομένων και στρατηγικές δοκιμών με τεχνητή νοημοσύνη που εξαρτώνται από εισροές υψηλής ποιότητας για ουσιαστική ανάλυση (Borah et al., 2022). Η εγκυρότητα αφορά τη διασφάλιση πως τα δεδομένα δοκιμής αναπαριστούν πιστά ρεαλιστικές και επιτρεπτές συνθήκες λειτουργίας του υπό δοκιμή συστήματος, η μεταβλητότητα τη διασφάλιση πως καλύπτεται ευρύ φάσμα σεναρίων και οριακών περιπτώσεων και όχι μόνο τυπικές περιπτώσεις λειτουργίας και η ιχνηλασιμότητα τη δυνατότητα συσχέτισης

³ Ο όρος «διαμορφώσεις» (configurations) αφορά τις ρυθμίσεις και συνθήκες λειτουργίας του συστήματος που επηρεάζουν την εκτέλεση και τα αποτελέσματα των δοκιμών (π.χ. εκδόσεις λογισμικού, ρυθμίσεις εφαρμογών, δικαιώματα χρηστών).

συγκεκριμένων δεδομένων δοκιμής με τις αντίστοιχες απαιτήσεις, περιπτώσεις δοκιμής και αναφερόμενα σφάλματα που τυχόν προκύψουν.

Test oracle: «Όταν ένα πρόγραμμα εκτελείται με μια συγκεκριμένη είσοδο, απαιτείται ένας μηχανισμός για τη διάκριση μεταξύ της σωστής και της λάθους συμπεριφοράς (αναλόγως των απαιτήσεων) του συστήματος υπό δοκιμή. Αν και σε απλές περιπτώσεις η αξιολόγηση της ορθότητας μπορεί να πραγματοποιηθεί μέσω άμεσης σύγκρισης μεταξύ αναμενόμενων και πραγματικών αποτελεσμάτων, σε πολλά σύγχρονα συστήματα ο καθορισμός του αναμενόμενου αποτελέσματος δεν είναι πάντα εφικτός ή είναι ιδιαίτερα σύνθετος, είτε λόγω του μεγάλου όγκου ή της φύσης των δεδομένων είτε λόγω της ύπαρξης πρακτικά απεριόριστων εισόδων. Χαρακτηριστικά παραδείγματα αποτελούν εφαρμογές επιστημονικών υπολογισμών ή μηχανικής μάθησης, όπου το ακριβές αναμενόμενο αποτέλεσμα συχνά δεν είναι γνωστό εκ των προτέρων και η ορθότητα εκτιμάται βάσει προσεγγιστικών ή στατιστικών κριτηρίων. Στις περιπτώσεις αυτές απαιτείται ένας μηχανισμός ή ένα σύνολο κριτηρίων που επιτρέπουν την εκτίμηση της αποδεκτής συμπεριφοράς του συστήματος, γνωστός ως *test oracle*. Υπάρχουν τρεις κύριες κατηγορίες *test oracle*: *specified*, *derived* και *implicit*. Τα *specified oracles* βασίζονται σε ρητές και συνήθως τυπικές προδιαγραφές της αναμενόμενης συμπεριφοράς του συστήματος, έναντι των οποίων αξιολογούνται τα αποτελέσματα. Τα *derived oracles* προκύπτουν από πληροφορία που εξάγεται από υπάρχοντα τεχνουργήματα, προηγούμενες εκδόσεις ή παρατηρήσεις εκτέλεσης. Τέλος, τα *implicit oracles* αξιοποιούν γενική γνώση σχετικά με το τι συνιστά αποδεκτή ή μη αποδεκτή συμπεριφορά, όπως η αποφυγή καταρρεύσεων ή άλλων προφανών ανωμαλιών (Barr et al., 2015).

2.2.3 ΕΙΔΗ ΔΟΚΙΜΗΣ ΛΟΓΙΣΜΙΚΟΥ

Υπάρχουν δύο θεμελιώδεις προσεγγίσεις στη δοκιμή λογισμικού, το *black-box* και το *white-box testing*, οι οποίες διακρίνονται με βάση τον βαθμό γνώσης που διαθέτει ο εξεταστής για την εσωτερική λειτουργία του συστήματος. Στη βιβλιογραφία αναφέρεται επίσης και η ενδιάμεση μορφή *grey-box testing*, η οποία συνδυάζει στοιχεία και από τις δύο προσεγγίσεις, χωρίς ωστόσο να αποτελεί αυτοτελή θεμελιώδη κατηγορία. Στο *black-box testing*, ο έλεγχος εστιάζει αποκλειστικά στη συμπεριφορά του συστήματος βάσει εισόδων και εξόδων, χωρίς πρόσβαση

στη (εσωτερική) δομή του κώδικα. Τεχνικές όπως η ισοδύναμη κατάτμηση⁴ (*equivalence partitioning*) και η ανάλυση οριακών τιμών (*boundary value analysis*)⁵ στο πλαίσιο της προσέγγισης/στρατηγικής δοκιμής συμπεριφοράς (*behavioral testing*)⁶ είναι ιδιαίτερα χρήσιμες για την αξιολόγηση της λειτουργικότητας και των απαιτήσεων του συστήματος (Borah et al., 2022). Αντίθετα, το *white-box testing* βασίζεται σε πλήρη ορατότητα της εσωτερικής λογικής και δομής του κώδικα. Περιλαμβάνει τεχνικές όπως η κάλυψη εντολών (*statement coverage*)⁷ και κάλυψη κλαδιών (*branch coverage*)⁸. Οι δύο προσεγγίσεις είναι αλληλοσυμπληρούμενες: η πρώτη αξιολογεί τη συμπεριφορά του συστήματος από εξωτερική οπτική, επαληθεύοντας ότι ικανοποιούνται οι καθορισμένες απαιτήσεις και οι προσδοκίες των χρηστών μέσω της παρατήρησης των εισόδων και εξόδων. Αντίθετα, η δεύτερη επικεντρώνεται στην εσωτερική δομή και λογική της υλοποίησης, εξετάζοντας τα μονοπάτια εκτέλεσης, τις συνθήκες ελέγχου και τον βαθμό κάλυψης του κώδικα, με στόχο τον εντοπισμό λογικών σφαλμάτων, αδυναμιών και πιθανών σημείων αστοχίας.

Αυτοματοποιημένες Δοκιμές: Οι αυτοματοποιημένες δοκιμές μπορούν να περιγραφούν ως η χρήση προγραμμάτων και δομών λογισμικού για την αυτόματη διεξαγωγή των περιπτώσεων δοκιμών, τη σύγκριση των πραγματικών και των αναμενόμενων αποτελεσμάτων και την παραγωγή εκθέσεων/αναφορών δοκιμών με ελάχιστη (ιδανικά μηδενική) ανθρώπινη παρέμβαση. Η αυτοματοποίηση των δοκιμών είναι συνήθως ένα έργο ανάπτυξης αυτόματα εκτελέσιμων σεναρίων δοκιμών (*automated test scripts*) που μπορούν να αλληλεπιδράσουν με το σύστημα που δοκιμάζεται μέσω διεπαφών, οι οποίες μπορεί να είναι διεπαφές προγραμματισμού εφαρμογών (API), γραφικές διεπαφές χρήστη (GUI) ή διεπαφές γραμμής εντολών.

⁴ Η ισοδύναμη κατάτμηση αποτελεί τεχνική σχεδίασης δοκιμών κατά την οποία το σύνολο πιθανών εισόδων διαχωρίζεται σε κλάσεις που αναμένεται να επιδεικνύουν παρόμοια συμπεριφορά, επιτρέποντας την επιλογή αντιπροσωπευτικών τιμών αντί της εξαντλητικής δοκιμής.

⁵ Η ανάλυση οριακών τιμών επικεντρώνεται στον έλεγχο τιμών στα άκρα ή πλησίον των ορίων των περιοχών εισόδου, όπου η πιθανότητα εμφάνισης σφαλμάτων είναι αυξημένη, καθώς συχνά περιοχές τιμών πλησίον των ορίων αντιστοιχούν σε μεταβάσεις λογικής ή ελέγχου ροής, όπου μικρές αστοχίες στις συνθήκες σύγκρισης μπορούν να οδηγήσουν σε διαφορετική συμπεριφορά του συστήματος.

⁶ Αξιολογεί τη συμπεριφορά μιας εφαρμογής λογισμικού σε απόκριση σε διάφορες εισόδους και σεναρία. Στόχος της είναι να διασφαλίσει πως το λογισμικό λειτουργεί όπως προβλέπεται, ανταποκρίνεται στις απαιτήσεις των χρηστών και συμπεριφέρεται όπως αναμένεται σε διαφορετικές περιπτώσεις χρήσης.

⁷ Αξιολογεί κατά πόσο κάθε εκτελέσιμη εντολή του υπό δοκιμή προγράμματος έχει εκτελεστεί τουλάχιστον μία φορά κατά την εκτέλεση των δοκιμών.

⁸ Εξετάζει αν ο κάθε πιθανός κλάδος απόφασης έχει εκτελεστεί τουλάχιστον μια φορά κατά την εκτέλεση των δοκιμών.

Η αυτοματοποίηση διαδραματίζει καθοριστικό ρόλο στη διαχείριση αυτών των διαφορετικών τύπων δοκιμών, ιδίως σε περιβάλλοντα που απαιτούν επαναλαμβανόμενη, μεγάλης κλίμακας ή ταχεία εκτέλεση σειρών δοκιμών. Τα αυτοματοποιημένα εργαλεία δοκιμών βελτιώνουν τη συνέπεια και την αποτελεσματικότητα σε τύπους δοκιμών όπου η συχνότητα εκτέλεσης και ο όγκος των δεδομένων είναι σημαντικοί. Ωστόσο, οι παραδοσιακές προσεγγίσεις αυτοματοποίησης αντιμετωπίζουν περιορισμούς όταν έχουν να κάνουν με δυναμικές διεπαφές χρήστη, εξελισσόμενες βάσεις κώδικα και πολύπλοκα πρότυπα συμπεριφοράς, γεγονός που έχει οδηγήσει στην αυξανόμενη υιοθέτηση της αυτοματοποίησης με βάση την τεχνητή νοημοσύνη για την έξυπνη επιλογή δοκιμών, την πρόβλεψη σφαλμάτων και την προσαρμοστική εκτέλεση δοκιμών. Συνολικά, η ποικιλία των τύπων δοκιμών και η αυξανόμενη ενσωμάτωση της αυτοματοποίησης αντικατοπτρίζουν την ανάγκη για ολοκληρωμένη, πολυδιάστατη αξιολόγηση της ποιότητας του λογισμικού στη σύγχρονη πρακτική της μηχανικής λογισμικού (Zubair et al., 2022).

2.2.4. ΚΥΡΙΟΙ ΤΥΠΟΙ ΔΟΚΙΜΗΣ ΛΟΓΙΣΜΙΚΟΥ

Παρατίθενται παρακάτω οι κύριοι τύποι δοκιμής λογισμικού (Borah et al. , 2022):

- *Δοκιμές ανάπτυξης (Development testing)*: Περιλαμβάνουν τους ακόλουθους τύπους δοκιμών:
 - *Δοκιμές μονάδων (unit testing)*: Πρόκειται για τη δοκιμή μεμονωμένων μονάδων ή ενοτήτων ενός προϊόντος λογισμικού. Η μονάδα υπό δοκιμή μπορεί να είναι μια μέθοδος ή κλάση (Zubair et al., 2022). Γίνεται έλεγχος κυρίως λειτουργικής συμπεριφοράς και μόνο σε ορισμένες περιπτώσεις μπορεί να ελέγχεται και η απόδοση των μονάδων (σε επίπεδο μονάδων, οι δοκιμές απόδοσης ελέγχουν για παράδειγμα πόσο γρήγορα εκτελείται κάποια μέθοδος ή συνάρτηση).
 - *Δοκιμές Συστατικών (Component testing)*: Αφορούν την αξιολόγηση ενός συστατικού σε απομόνωση από τα υπόλοιπα μέρη του συστήματος, με στόχο την επαλήθευση της εσωτερικής του λογικής και της αναμενόμενης συμπεριφοράς του. Οι εξαρτήσεις από άλλα συστατικά αντικαθίστανται συνήθως από προσομοιωμένα αντικείμενα (*mocks* ή *stubs*) που παρουσιάζουν ιδανική συμπεριφορά στα σενάρια υπό δοκιμή, ώστε ο έλεγχος να επικεντρώνεται αποκλειστικά στο υπό δοκιμή στοιχείο.

- *Δοκιμές Ενοποίησης (Integration testing)*: Μετά την επιτυχή επαλήθευση σε απομόνωση, πολλαπλές μονάδες ή συστατικά συνδυάζονται και ελέγχονται από κοινού, προκειμένου να διαπιστωθεί ότι οι διεπαφές και οι αλληλεπιδράσεις τους λειτουργούν σωστά και ότι δεν έχουν εισαχθεί σφάλματα κατά τον συνδυασμό τους (Zubair et al., 2022).

Στη σύγχρονη βιβλιογραφία, οι δοκιμές σε απομόνωση τοποθετούνται στο χαμηλότερο επίπεδο, ενώ η δοκιμή ενοποίησης επικεντρώνεται στη συνεργασία μεταξύ δύο ή περισσότερων τέτοιων στοιχείων.

Δοκιμές συστήματος (System Testing): Σε αυτό τον τύπο δοκιμής, το σύστημα ελέγχεται πλήρως ως ένα ενιαίο σύνολο, με στόχο την επιβεβαίωση ότι ικανοποιεί τις καθορισμένες λειτουργικές και μη λειτουργικές απαιτήσεις. Αν και η έμφαση δίνεται συνήθως στο συνολικό προϊόν, είναι δυνατόν να εφαρμοστεί αντίστοιχη προσέγγιση και σε επιμέρους υποσυστήματα, όταν αυτά αντιμετωπίζονται ως αυτόνομες οντότητες προς αξιολόγηση.

- *Δοκιμές έκδοσης (release testing)*: Πρόκειται για το σύνολο των δραστηριοτήτων επαλήθευσης που πραγματοποιούνται στο πλήρως ενοποιημένο σύστημα πριν από τη διάθεσή του στους τελικούς χρήστες ή την εγκατάστασή του στο περιβάλλον παραγωγής. Στόχος των δοκιμών αυτών είναι να επιβεβαιωθεί ότι το προϊόν ικανοποιεί τις καθορισμένες λειτουργικές και μη λειτουργικές απαιτήσεις και είναι κατάλληλο για λειτουργία υπό ρεαλιστικές συνθήκες. Στο πλαίσιο του release testing μπορούν να ενταχθούν διαφορετικοί τύποι ελέγχων, όπως δοκιμές που βασίζονται στις απαιτήσεις (requirements testing), δοκιμές σεναρίων από άκρο σε άκρο (end-to-end testing), καθώς και δραστηριότητες ελέγχου απόδοσης, ασφάλειας ή χρηστικότητας (performance, security or reliability testing), ανάλογα με τις ανάγκες και το επίπεδο κινδύνου του έργου. Αν και αρκετοί από τους προαναφερόμενους τύπους δοκιμών εφαρμόζονται και στις δοκιμές συστήματος, στο πλαίσιο του release testing αξιοποιούνται κυρίως ως τελικός μηχανισμός επαλήθευσης της ετοιμότητας του συστήματος προς διάθεση. Ορισμένοι από αυτούς τους τύπους αναλύονται στη συνέχεια:

- *Δοκιμές απαιτήσεων (requirements testing)*: Αποτελούν τη διαδικασία σχεδίασης και εκτέλεσης περιπτώσεων δοκιμής που απορρέουν άμεσα από τις προδιαγεγραμμένες απαιτήσεις του συστήματος. Πέρα από την επαλήθευση της αναμενόμενης λειτουργικότητας, οι δοκιμές αυτές περιλαμβάνουν τη διασφάλιση της ιχνηλασιμότητας μεταξύ απαιτήσεων και αποτελεσμάτων, τεκμηριώνοντας τον βαθμό κάλυψης και πληρότητας.
- *Δοκιμές σεναρίων (scenario testing)*: Βασίζονται στην εκτέλεση ρεαλιστικών και ολοκληρωμένων ροών χρήσης (*end-to-end scenarios*), οι οποίες προσομοιώνουν τον τρόπο με τον οποίο οι χρήστες αλληλεπιδρούν με το σύστημα στην πράξη, επιτρέποντας την αξιολόγηση της συνολικής συμπεριφοράς του.
- *Δοκιμή χρηστών (user testing)*: Αναφέρεται στην αξιολόγηση του συστήματος από πραγματικούς ή αντιπροσωπευτικούς χρήστες, με στόχο την επιβεβαίωση ότι το προϊόν ανταποκρίνεται στις επιχειρησιακές ανάγκες και στις προσδοκίες χρήσης. Μπορούν να λάβουν διάφορες μορφές, όπως οι δοκιμές άλφα, οι δοκιμές βήτα και οι δοκιμές αποδοχής.
 - *Δοκιμή άλφα (alpha testing)*: Πραγματοποιείται σε ελεγχόμενο περιβάλλον, συνήθως από την ομάδα ανάπτυξης ή από περιορισμένο αριθμό χρηστών, με στόχο τον εντοπισμό προβλημάτων πριν από τη διάθεση του προϊόντος σε ευρύτερο κοινό.
 - *Δοκιμή βήτα (beta testing)*: Διεξάγεται από επιλεγμένους τελικούς χρήστες σε πραγματικές συνθήκες λειτουργίας (δηλ. σε πραγματικά περιβάλλοντα παραγωγής), επιτρέποντας τη συλλογή ανατροφοδότησης σχετικά με τη χρηστικότητα, την αξιοπιστία και τη συνολική εμπειρία χρήσης πριν από την ευρεία διάθεση του συστήματος. Σκοπός είναι η ανάδειξη και διόρθωση προβλημάτων που σχετίζονται με σφάλματα ειδικά σε ένα λειτουργικό σύστημα, μη αναμενόμενες συμπεριφορές χρηστών, ζητήματα απόδοσης υπό πραγματική χρήση και ζητήματα ενοποίησης με πραγματικά οικοσυστήματα.
 - *Δοκιμή αποδοχής (acceptance testing)*: Αποτελεί τη διαδικασία μέσω της οποίας οι πελάτες ή οι εξουσιοδοτημένοι εκπρόσωποί τους επιβεβαιώνουν ότι το σύστημα πληροί τις συμφωνημένες απαιτήσεις και μπορεί να γίνει αποδεκτό για επιχειρησιακή χρήση με βάση σαφώς προδιαγεγραμμένα κριτήρια αποδοχής.

2.3 ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

2.3.1 ΟΡΙΣΜΟΣ

Τεχνητή νοημοσύνη (Artificial Intelligence-AI): «Η τεχνητή νοημοσύνη είναι ένας γενικός όρος που αναφέρεται σε οποιαδήποτε μηχανή ή αλγόριθμο που είναι ικανός να παρατηρεί το περιβάλλον του, να μαθαίνει και, με βάση τις γνώσεις και την εμπειρία που έχει αποκτήσει, να αναλαμβάνει ευφυή δράση ή να προτείνει αποφάσεις.» (Craglia et al., 2018).

2.3.2 ΕΙΔΗ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ

Μεταξύ των σημαντικότερων προσεγγίσεων / ειδών της ΑΙ συγκαταλέγονται (Russell & Norvig, 2020):

- *Μηχανική μάθηση (machine learning):* επιτρέπει στα συστήματα να βελτιώνουν την απόδοσή τους μέσω εμπειρίας.
- *Βαθιά μάθηση (deep learning):* αξιοποιεί νευρωνικά δίκτυα (neural networks, NN) πολλαπλών επιπέδων για την εκμάθηση πολύπλοκων προτύπων από δεδομένα.
- *Εμπειρα συστήματα (expert systems):* βασίζονται σε κανόνες και κωδικοποιημένη γνώση ειδικών, με σκοπό την υποστήριξη της λήψης αποφάσεων ή την επίλυση προβλημάτων σε συγκεκριμένους τομείς.
- *Επεξεργασία φυσικής γλώσσας (natural language processing, NLP):* επιτρέπει την κατανόηση και παραγωγή ανθρώπινης γλώσσας.
- *Ασαφής λογική (fuzzy logic):* επιτρέπει συλλογισμούς υπό συνθήκες αβεβαιότητας μέσω βαθμών αλήθειας (Zadeh, 1965).

2.3.3. ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΣΥΣΤΗΜΑΤΩΝ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ

Βασικός σκοπός της ΑΙ είναι η αυτοματοποίηση γνωστικών διεργασιών, όπως η αναγνώριση προτύπων, η πρόβλεψη και η υποστήριξη λήψης αποφάσεων. Τα συστήματα ΑΙ χαρακτηρίζονται από δυνατότητα μάθησης, προσαρμοστικότητας και γενίκευσης γνώσης από παραδείγματα. Η ανάπτυξη εφαρμογών ΑΙ ακολουθεί συνήθως επαναληπτική διαδικασία που περιλαμβάνει συλλογή και προετοιμασία δεδομένων, εκπαίδευση μοντέλων, αξιολόγηση της απόδοσης και βελτίωση βάσει ανατροφοδότησης (Russell & Norvig, 2020).

3. ΜΕΘΟΔΟΛΟΓΙΑ ΑΝΑΣΚΟΠΗΣΗΣ

Το παρόν κεφάλαιο περιγράφει τη μεθοδολογία που ακολουθήθηκε για τη διεξαγωγή της βιβλιογραφικής ανασκόπησης σχετικά με την εφαρμογή της τεχνητής νοημοσύνης στις δοκιμές λογισμικού και των μεθόδων δοκιμής λογισμικού σε συστήματα τεχνητής νοημοσύνης. Η έρευνα σχεδιάστηκε ως συστηματική βιβλιογραφική ανασκόπηση (*systematic literature review*, *SLR*), υιοθετώντας τις κατευθυντήριες γραμμές που προτείνονται από τους Kitchenham και Charters (2007) στο πλαίσιο της μηχανικής λογισμικού που βασίζεται σε τεκμήρια. Σύμφωνα με τη συγκεκριμένη μεθοδολογία, το κεφάλαιο οργανώνεται γύρω από τα βασικά στάδια της διαδικασίας, τα οποία περιλαμβάνουν τον σχεδιασμό της ανασκόπησης, τη στρατηγική αναζήτησης της βιβλιογραφίας και τα κριτήρια επιλογής και αποκλεισμού.

3.1 ΣΤΟΧΟΣ ΚΑΙ ΕΡΕΥΝΗΤΙΚΑ ΕΡΩΤΗΜΑΤΑ

Ο κύριος στόχος της ανασκόπησης είναι ο προσδιορισμός του βαθμού αλληλεπίδρασης που υπάρχει μεταξύ τεχνητής νοημοσύνης και δοκιμής λογισμικού μέσω επιμέρους αναλύσεων της σύγχρονης βιβλιογραφίας για την εφαρμογή της τεχνητής νοημοσύνης στη δοκιμή λογισμικού, αλλά και στην εφαρμογή κατάλληλων μεθόδων δοκιμής λογισμικού για τον έλεγχο συστημάτων τεχνητής νοημοσύνης. Τα βασικά ερευνητικά ερωτήματα προς απάντηση είναι τα εξής:

1. Ποιες είναι οι βασικές κατευθύνσεις συνεισφοράς της Τεχνητής Νοημοσύνης στη Δοκιμή Λογισμικού; (αναλυτική απάντηση στην Ενότητα 4.1)
2. Ποιες είναι οι βασικές κατευθύνσεις συνεισφοράς της Δοκιμής Λογισμικού στην Τεχνητή Νοημοσύνη; (αναλυτική απάντηση στην Ενότητα 4.2)
3. Εντοπίζονται σχετικά ερευνητικά κενά στη σύγχρονη βιβλιογραφία; (αναλυτική απάντηση στην Ενότητα 5)
4. Ποιες ερευνητικές κατευθύνσεις προτείνονται για την κάλυψη των αντίστοιχων ερευνητικών κενών; (αναλυτική απάντηση Ενότητα 5)

3.2 ΣΤΡΑΤΗΓΙΚΗ ΑΝΑΖΗΤΗΣΗΣ

Η βιβλιογραφική αναζήτηση πραγματοποιήθηκε τόσο στις πλατφόρμες GoogleScholar και Web of Science όσο και απευθείας στις ψηφιακές βιβλιοθήκες των εκδοτών ACM, Elsevier, IEEE, MDPI και Springer, για τη διασφάλιση συγκέντρωσης βιβλιογραφικών πηγών

ακαδημαϊκής εγκυρότητας και υψηλού κύρους, οι παραπάνω πλατφόρμες και ψηφιακές βιβλιοθήκες είναι οι πλέον γνωστές και ευρέως χρησιμοποιούμενες.

Κατά την αναζήτηση, χρησιμοποιήθηκαν οι λέξεις-κλειδιά: Artificial Intelligence, Software Testing, Machine Learning, Deep Learning, Neural Networks, Challenges, Software Testing Activities, AI in Software Testing, Test Case Generation, Test Case Validation, Machine Learning for Testing, Natural Language Processing for Testing, Test Automation, Test case optimization, AI – driven testing Tools, Test Oracle, Automated Test Oracle Generation, IoT, Language Models, NLP Techniques, Code Smell, Code Smell Detection, Deep Reinforcement Learning, Android Testing, Bug Reports, Bug Detection, Automated Test Generation Software Testing Automation, GUI Testing, Test Data Generation Regression Testing, Mutation Testing, Genetic Algorithm, Test Case Prioritization, Clustering, Unit Test Generation, Metamorphic Relations, Quality Assessment, Deep Neural Networks, Deep Learning Testing, Large Language Models, LLM – Driven Testing, Relation Extraction, Black Box Testing, White Box Testing, Gray Box Testing, Fuzzy Logic

Στον Πίνακα 1 που ακολουθεί, παρατίθενται οι λέξεις-κλειδιά και η αντίστοιχη αιτιολόγηση της χρήσης τους στη μεθοδολογία ανασκόπησης.

Πίνακας 1. Λέξεις-κλειδιά και αιτιολόγηση της χρήσης τους στη μεθοδολογία ανασκόπησης

Λέξη-κλειδί	Αιτιολόγηση χρήσης
Artificial Intelligence / AI	Γενικός όρος για τεχνικές ευφυών συστημάτων / τεχνητής νοημοσύνης
Machine Learning	Υποπεδίο της AI με ευρεία χρήση στη δοκιμή
Deep Learning / Neural Networks	Σύγχρονες τεχνικές μάθησης με αυξανόμενη εφαρμογή
Large Language Models / LLMs	Νεότερα μοντέλα μηχανικής μάθησης για ισχυρή παραγωγή & ανάλυση κειμένου (μεταξύ άλλων)

Natural Language Processing/ NLP	Επεξεργασία απαιτήσεων, bug reports κ.λπ.
Test Case Generation	Παραγωγή(περιπτώσεων) δοκιμών
Test Case Validation	Έλεγχος ορθότητας δοκιμών
Test Automation	Αυτοματοποίηση διαδικασιών ελέγχου
Test Oracle	Μηχανισμοί αξιολόγησης αποτελεσμάτων δοκιμών
Metamorphic Relations	Χρησιμοποιούνται για την αντιμετώπιση του oracle problem
Test Data Generation	Παραγωγή εισόδων/δεδομένων δοκιμών
Regression Testing	Επαναληπτικές δοκιμές μετά από αλλαγές
Mutation Testing	Αξιολόγηση ποιότητας test suite (σουίτας δοκιμής, δηλ. συνόλου δοκιμών)
Genetic Algorithm / Clustering	Τέτοιοι αλγόριθμοι χρησιμοποιούνται για την παραγωγή και βελτιστοποίηση δεδομένων δοκιμών και την προτεραιοποίηση των περιπτώσεων δοκιμών
Unit Test Generation	Παραγωγή δοκιμών μονάδας
GUI / Android Testing	Έλεγχος σε εξειδικευμένα περιβάλλοντα εφαρμογής
Bug Detection / Reports	Ανίχνευση & ανάλυση σφαλμάτων

Black / White / Gray Box Testing	Προσεγγίσεις ελέγχου/δοκιμής
Fuzzy Logic	Τεχνικές ασαφούς λογικής που χρησιμοποιούνται στον χειρισμό μη αναμενόμενων σωστών αποτελεσμάτων δοκιμής, στην προτεραιοποίηση περιπτώσεων δοκιμών, κα.

Οι συνδυασμοί των λέξεων-κλειδιών που εφαρμόστηκαν στις επερωτήσεις προς τις βιβλιογραφικές πλατφόρμες και βιβλιοθήκες ακολούθησαν το ακόλουθο μοτίβο, το οποίο περιλαμβάνει 2 βασικά μέρη που αντιστοιχούν στους δύο ερευνητικούς τομείς στους οποίους εστιάζει η ανασκόπηση όσον αφορά την αλληλεπίδρασή τους:

(“AI” OR “Machine Learning” OR “Deep Learning” OR “Deep Neural Networks” OR “Large Language Models” OR “LLMs” OR “Natural Language Processing” OR “NLP” OR “Fuzzy Logic”) AND (“Test Case Generation” OR “Test Case Validation” OR “Test Automation” OR “Test Oracle” OR “Metamorphic Relations” OR “Test Data Generation” OR “Regression Testing” OR “Mutation Testing” OR “Genetic Algorithm/ Clustering” OR “Unit Test Generation” OR “Bug Detection” OR “Bug Reports” OR “Black Box Testing” OR “White Box Testing” OR “Gray Box Testing”)

Η διαδικασία αναζήτησης ακολούθησε επαναληπτική προσέγγιση. Αρχικά εντοπίστηκαν μελέτες εννοιολογικής θεμελίωσης και σχετικές βιβλιογραφικές ανασκοπήσεις, από τις οποίες προέκυψαν πρόσθετοι όροι και επιμέρους θεματικές περιοχές. Οι νέες αυτές έννοιες αξιοποιήθηκαν ως εμπλουτισμένες λέξεις-κλειδιά σε επόμενους κύκλους αναζήτησης στις επιλεγμένες βάσεις δεδομένων, με στόχο τη διεύρυνση και βελτίωση της κάλυψης της βιβλιογραφίας. Η προσέγγιση αυτή διαφοροποιείται από την τεχνική snowballing, καθώς ο εντοπισμός επιπλέον μελετών βασίστηκε κυρίως στην εξέλιξη των λέξεων-κλειδιών και όχι στη διερεύνηση των βιβλιογραφικών αναφορών.

3.3 ΦΙΛΤΡΑΡΙΣΜΑ ΕΠΙΛΕΓΜΕΝΗΣ ΒΙΒΛΙΟΓΡΑΦΙΑΣ

Μετά την ολοκλήρωση της διαδικασίας αναζήτησης, εφαρμόστηκαν διάφορα είδη κριτηρίων ώστε να επιλεγούν μόνο εκείνες οι μελέτες που σχετίζονται άμεσα με το αντικείμενο της έρευνας και έχουν το κατάλληλο επίπεδο ποιότητας.

3.3.1 ΚΡΙΤΗΡΙΑ ΕΝΤΑΞΗΣ/ΑΠΟΚΛΕΙΣΜΟΥ

Τα κριτήρια ένταξης και αποκλεισμού εφαρμόστηκαν προκειμένου να επιλεγούν οι πλέον κατάλληλες και έγκυρες μελέτες. Τα κριτήρια αυτά βασίστηκαν σε μεταδεδομένα των δημοσιεύσεων και στόχευαν στη διασφάλιση της συνάφειας και της ποιότητας του περιεχομένου τους. Στη συνέχεια παρουσιάζονται τα κριτήρια ένταξης και αποκλεισμού που εφαρμόστηκαν για το φιλτράρισμα της βιβλιογραφίας.

Κριτήρια ένταξης/αποκλεισμού:

- *Τύπος Δημοσίευσης*: επιλογή δημοσιεύσεων μόνο σε επιστημονικά περιοδικά ή πρακτικά συνεδρίων, οπότε αποκλείονται πηγές που εντάσσονται στην γκρίζα και την κίτρινη βιβλιογραφία. Η γκρίζα βιβλιογραφία, παρότι ενδέχεται να περιέχει χρήσιμες πληροφορίες, δεν αξιολογείται συστηματικά με τα ίδια κριτήρια (για παράδειγμα μέσω διαδικασίας peer review), ενώ η λεγόμενη κίτρινη βιβλιογραφία περιλαμβάνει πηγές περιορισμένης τεκμηρίωσης και επιστημονικής αυστηρότητας.
- *Έτος δημοσίευσης*: μεταξύ 2018 και 2025, ώστε να αποτυπώνεται η πρόσφατη ερευνητική δραστηριότητα στον ταχέως εξελισσόμενο τομέα της τεχνητής νοημοσύνης.
- *Γλώσσα συγγραφής*: επιλέχθηκε η αγγλική, καθώς αποτελεί την κυρίαρχη γλώσσα δημοσίευσης στην επιστημονική βιβλιογραφία, εξασφαλίζοντας πρόσβαση σε ευρύ και αντιπροσωπευτικό σύνολο peer – reviewed εργασιών.
- *Συνάφεια με το αντικείμενο της έρευνας*. Ο βαθμός συνάφειας με το αντικείμενο της έρευνας (συνεισφορά ΑΙ στην δοκιμή λογισμικού και το αντίστροφο) εφαρμόστηκε αρχικά ως κριτήριο ένταξης/ αποκλεισμού με βάση τα μεταδεδομένα των δημοσιεύσεων. Στις περιπτώσεις όπου δεν ήταν ξεκάθαρο από αυτά αν η μελέτη θα πρέπει να συμπεριληφθεί, εξεταζόταν και το περιεχόμενο

αυτής, οπότε ο βαθμός συνάφειας λειτούργησε και ως κριτήριο ποιότητας, όπως αναφέρεται και παρακάτω.

- *Έκταση της μελέτης*: αποκλεισμός δημοσιεύσεων μικρότερων των 8 μονόστηλων σελίδων, ώστε να διασφαλιστεί η ωριμότητα και το επιστημονικό βάθος της μελέτης/προσέγγισης.

3.3.2 ΚΡΙΤΗΡΙΑ ΠΟΙΟΤΗΤΑΣ

Εκτός από την εφαρμογή των παραπάνω κριτηρίων, πραγματοποιήθηκε ποιοτική αξιολόγηση των μελετών με βάση το περιεχόμενό τους. Στόχος ήταν να διασφαλιστεί ότι οι επιλεγμένες εργασίες παρουσιάζουν επαρκή επιστημονική τεκμηρίωση και ουσιαστική συμβολή στο αντικείμενο.

Τα κριτήρια ποιότητας που εφαρμόστηκαν περιελάμβαναν:

- Τη σαφή περιγραφή της ερευνητικής μεθοδολογίας.
- Την επάρκεια της τεχνικής ανάλυσης και των παρεχόμενων αποτελεσμάτων.
- Τον βαθμό συνάφειας με την εφαρμογή τεχνικών τεχνητής νοημοσύνης στη δοκιμή λογισμικού και τεχνικών δοκιμής λογισμικού σε συστήματα τεχνητής νοημοσύνης. Ο βαθμός συνάφειας εφαρμόστηκε εις διπλούν διότι είναι ένα κρίσιμο κριτήριο. Αρχικά, ο έλεγχος της συνάφειας εφαρμόστηκε με βάση τα μεταδεδομένα μιας δημοσίευσης (οπότε χρησιμοποιήθηκε ως κριτήριο ένταξης/αποκλεισμού). Αν με βάση αυτά, δεν ήταν ξεκάθαρη η συνάφεια, τότε εξεταζόταν το περιεχόμενο της δημοσίευσης για την τελική διαλογή (οπότε η συνάφεια γινόταν πλέον κριτήριο ποιότητας).

3.3.3 ΑΠΟΤΕΛΕΣΜΑΤΑ ΦΙΛΤΡΑΡΙΣΜΑΤΟΣ ΒΙΒΛΙΟΓΡΑΦΙΑΣ

Η εφαρμογή της στρατηγικής αναζήτησης στις επιλεγμένες ψηφιακές βάσεις δεδομένων επέστρεψε σημαντικά μεγαλύτερο αριθμό αποτελεσμάτων από εκείνον που τελικά εξετάστηκε σε βάθος. Σε πρώτο στάδιο πραγματοποιήθηκε προκαταρκτικός έλεγχος με βάση τους τίτλους και τις περιλήψεις, με στόχο την απομάκρυνση δημοσιεύσεων που ήταν προφανώς εκτός αντικειμένου. Από τον προκαταρκτικό αυτόν έλεγχο εντάχθηκαν τελικά 200 μελέτες προς περαιτέρω φιλτράρισμα. Ο αρχικός αυτός αριθμός περιορίστηκε κατόπιν εφαρμογής των κριτηρίων ένταξης σε 44 δυνητικά σχετικές μελέτες. Η αναλυτική εξέταση των μελετών αυτών

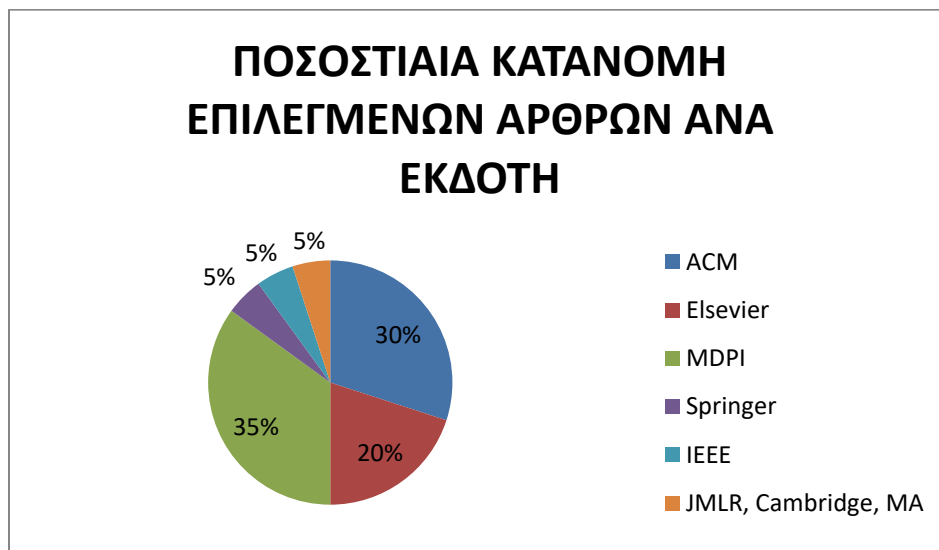
με την εφαρμογή των κριτηρίων ποιότητας οδήγησε στην τελική επιλογή 20 μελετών πρωτογενούς ερευνητικής συνεισφοράς. Στα παρακάτω Σχήματα 1 και 2 εμφανίζονται δύο αντίστοιχα που απεικονίζουν την ποσοστιαία κατανομή δημοσιευμένων άρθρων ανά τύπο, καθώς επίσης και την ποσοστιαία κατανομή άρθρων ανά εκδότη. Σχήμα3 απεικονίζεται η κατανομή των άρθρων ανά έτος δημοσίευσης.

Όπως φαίνεται από το Σχήμα 1, έχουμε αυξημένη παρουσία δημοσιεύσεων σε επιστημονικά περιοδικά, η οποία ενδέχεται να υποδηλώνει ότι σημαντικό μέρος της έρευνας στον συγκεκριμένο τομέα έχει ωριμάσει επαρκώς, ώστε να παρουσιάζεται σε μορφή εκτενέστερων και πληρέστερα αξιολογημένων εργασιών. Διότι τα περιοδικά συχνά απαιτούν πιο αναλυτική τεκμηρίωση, εκτενή πειραματική αξιολόγηση και σαφέστερη μεθοδολογική περιγραφή, στοιχεία που ενισχύουν την αξιοπιστία των αποτελεσμάτων. Παράλληλα, η παρουσία μελετών από πρακτικά συνεδρίων παραμένει ιδιαίτερα σημαντική, καθώς τα συνέδρια αποτελούν βασικό χώρο δημοσίευσης νέων ιδεών και αναδυόμενων προσεγγίσεων, ιδιαίτερα στον ταχέως εξελισσόμενο τομέα της τεχνητής νοημοσύνης και της δοκιμής λογισμικού.



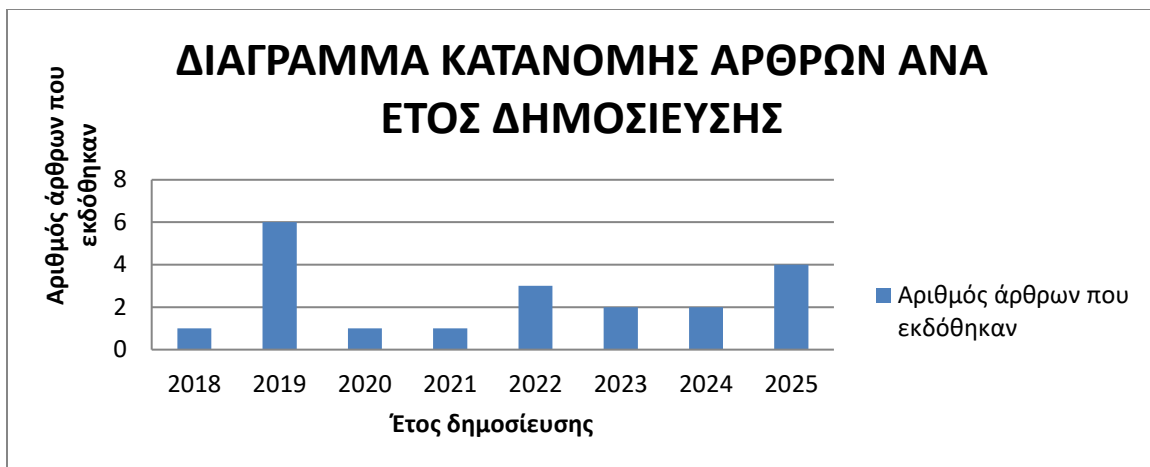
Σχήμα 1. Ποσοστιαία κατανομή επιλεγμένων άρθρων με κριτήριο τη δημοσίευση σε περιοδικό ή συνέδριο

Όσον αφορά την κατανομή ανά εκδότη, όπως φαίνεται από το Σχήμα 2, υπάρχει ξεκάθαρη συγκέντρωση άρθρων σε αναγνωρισμένους επιστημονικούς οργανισμούς, όπως οι ACM, Springer, MDPI και IEEE (95%). Αυτή υποδηλώνει ότι το μεγαλύτερο μέρος της επιλεγμένης βιβλιογραφίας προέρχεται από καθιερωμένα κανάλια επιστημονικής διάχυσης, γεγονός που ενισχύει περαιτέρω την εγκυρότητα της ανασκόπησης.



Σχήμα 2. Ποσοστιαία κατανομή επιλεγμένων άρθρων ανά εκδότη

Ελέγχοντας την κατανομή των άρθρων ανά έτος δημοσίευσης (Σχήμα 3), παρατηρούμε πως το ενδιαφέρον για την αλληλεπίδραση μεταξύ ΑΙ και δοκιμής λογισμικού είναι αμείωτο τα τελευταία έτη, καθώς παρατηρούμε αυξητική τάση το 2025 σε σχέση με το διάστημα 2022-2024 όπου ο αριθμός δημοσιεύσεων ανά έτος έδειχνε να έχει σταθεροποιηθεί, με βάση πάντα το υπό μελέτη δείγμα της βιβλιογραφίας που έχει δημοσιευθεί. Μπορούμε να ερμηνεύσουμε την αυξητική αυτή τάση αφενός ως αύξηση του ερευνητικού ενδιαφέροντος, αφετέρου ως ένδειξη ωρίμανσης όλων των παραμέτρων για την εντατικοποίηση της έρευνας (εξέλιξη εργαλείων και προσεγγίσεων τεχνητής νοημοσύνης).



Σχήμα 3. Κατανομή άρθρων ανά έτος δημοσίευσης

Συνολικά, η εφαρμογή δομημένης στρατηγικής αναζήτησης, η χρήση επαναληπτικών κύκλων εμπλουτισμού των λέξεων-κλειδιών και η συστηματική αξιολόγηση των μελετών βάσει προκαθορισμένων κριτηρίων συνέβαλαν στη διαμόρφωση ενός συνεκτικού και τεκμηριωμένου συνόλου άρθρων. Η προσέγγιση αυτή ενισχύει τη διαφάνεια της διαδικασίας, προσδίδοντας αξιοπιστία στα ευρήματα της ανασκόπησης.

4.ΑΝΑΛΥΣΗ ΑΡΘΡΩΝ ΑΝΑ ΚΑΤΕΥΘΥΝΣΗ ΣΥΝΕΙΣΦΟΡΑΣ

Στο παρόν κεφάλαιο, εξετάζονται ξεχωριστά τα πεδία συνεισφοράς της τεχνητής νοημοσύνης στη δοκιμή λογισμικού και της δοκιμής λογισμικού στην τεχνητή νοημοσύνη. Για το εκάστοτε πεδίο, παρατίθεται πρώτα ένας πίνακας ανάλυσης strengths-weaknesses (δυνατών σημείων-αδυναμιών) ώστε να αναδειχθούν τα βασικά ισχυρά σημεία και αδυναμίες των προτεινόμενων προσεγγίσεων στο πεδίο αυτό. Ο πίνακας αυτός είναι επιστέγασμα της ανάλυσης που πραγματοποιήθηκε και μπορεί να θεωρηθεί ένα είδος επισκόπησης των αποτελεσμάτων της. Έπειτα, παρέχεται ένας εννοιολογικός χάρτης του πεδίου για την ιεραρχική ταξινόμηση των συνεισφορών στο πεδίο αυτό. Τέλος, τα άρθρα ανά πεδίο ταξινομούνται σύμφωνα με την κύρια κατεύθυνση συνεισφοράς τους και αναλύεται το περιεχόμενό τους.

4.1 ΣΥΝΕΙΣΦΟΡΑ ΤΗΣ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ ΣΤΗ ΔΟΚΙΜΗ ΛΟΓΙΣΜΙΚΟΥ
Πίνακας 2. Ανάλυση δυνατών σημείων-αδυναμιών στο πεδίο εφαρμογής της ΑΙ στη δοκιμή λογισμικού

ΔΥΝΑΤΑ ΣΗΜΕΙΑ	ΑΔΥΝΑΜΙΕΣ
Υψηλά ποσοστά ακρίβειας στην ανίχνευση σφαλμάτων (Dewangan et al., 2022, , Braga et al., 2018)	Αδυναμία επανασχεδιασμού ροής (Yang et al., 2025)
Βελτίωση προτεραιοποίησης test cases προς εκτέλεση (Novel Fuzzy Clustering Methods for Test Case Prioritization in Software Projects, Shrivathsan et al., 2019)	Εγγενείς αδυναμίες των LLMs στην παρούσα μορφή τους (Yang et al., 2025)
Αυτόματη δημιουργία αποτελεσματικών περιπτώσεων και δεδομένων δοκιμών (Wang et al., 2021, Kumar et al., 2025, Ouédraogo et al., 2025)	Επαναληπτικές διαδικασίες αυξάνουν το υπολογιστικό κόστος (Shrivathsan et al., 2019)
	Εξάρτηση των προσεγγίσεων τεχνητής νοημοσύνης που αξιοποιούν διαδικασίες μάθησης (ML, DL) από την πρόσβαση σε ποιοτικά δεδομένα εκπαίδευσης (Braga et al., 2018)

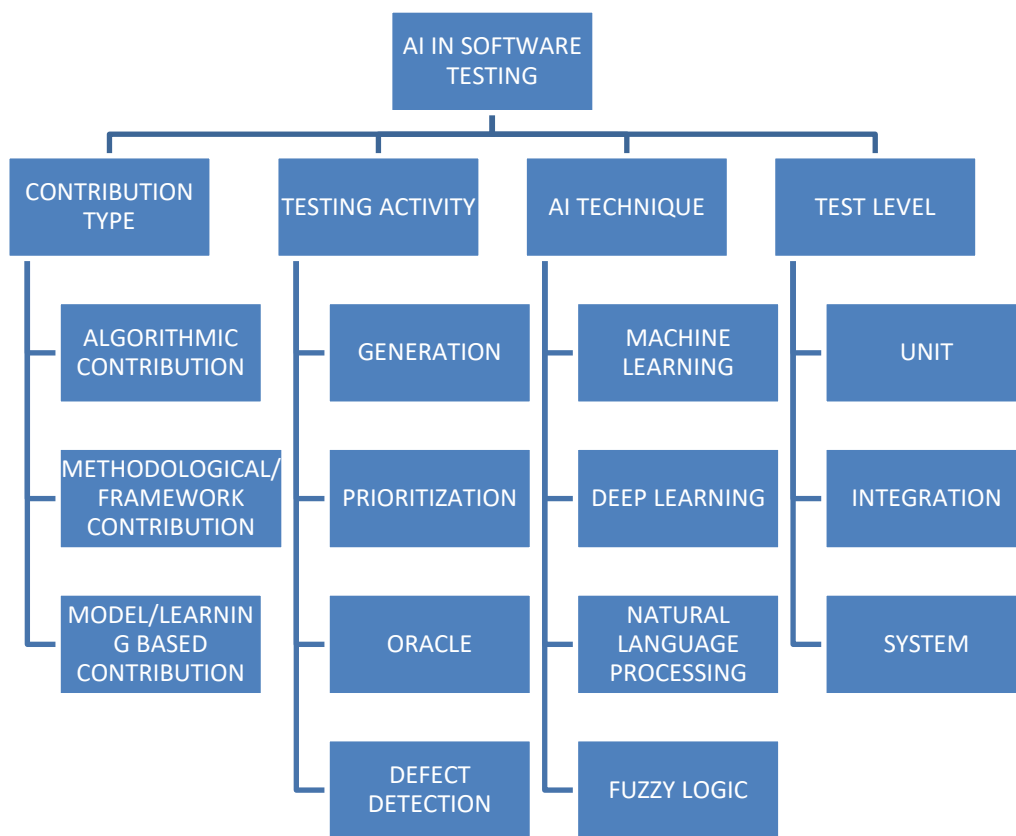
Ο πίνακας συνοψίζει τα βασικά δυνατά σημεία και αδυναμίες της συνεισφοράς της τεχνητής νοημοσύνης στη δοκιμή λογισμικού, όπως προκύπτουν από τη μελέτη που πραγματοποιήσαμε όσον αφορά τη σχετική βιβλιογραφία. Σημαντικά πλεονεκτήματα αποτελούν

τα υψηλά ποσοστά ακρίβειας στην ανίχνευση σφαλμάτων, η δυνατότητα αυτόματης παραγωγής αποτελεσματικών δεδομένων και περιπτώσεων δοκιμών και η αποδοτική προτεραιοποίηση των περιπτώσεων δοκιμών προς εκτέλεση. Προκύπτουν ωστόσο και βασικοί περιορισμοί, όπως η αδυναμία επανασχεδιασμού ροής, οι εγγενείς περιορισμοί των LLMs στην παρούσα μορφή τους καθώς και η ενδεχόμενη αύξηση του υπολογιστικού κόστους κατά την εκτέλεση επαναληπτικών διαδικασιών. Μια επίσης σημαντική πρόκληση που μπορεί να θεωρηθεί αδυναμία των μεθόδων στην παρούσα φάση αλλά και απειλή για τη μελλοντική τους εξέλιξη, είναι η εξάρτηση των προσεγγίσεων τεχνητής νοημοσύνης που αξιοποιούν διαδικασίες μάθησης (ML, DL) από την πρόσβαση σε ποιοτικά δεδομένα εκπαίδευσης, η οποία μπορεί για διάφορους λόγους να είναι περιορισμένη. Στην περίπτωση της δοκιμής λογισμικού που μας ενδιαφέρει στο πλαίσιο της μελέτης, αν η προτεινόμενη προσέγγιση βασίζεται σε δεδομένα εκπαίδευσης που προκύπτουν από την αλληλεπίδραση χρηστών με το σύστημα, προκύπτει ζήτημα εφαρμογής αυτής σε συστήματα που έχουν αναπτυχθεί πρόσφατα (Braga et al., 2018).

4.1.2 ΕΝΝΟΙΟΛΟΓΙΚΟΣ ΧΑΡΤΗΣ & ΤΑΞΙΝΟΜΗΣΗ ΕΠΙΛΕΓΜΕΝΩΝ ΑΡΘΡΩΝ

Ο εννοιολογικός χάρτης του πεδίου της συνεισφοράς της τεχνητής νοημοσύνης στη δοκιμή λογισμικού φαίνεται στο Σχήμα 4. Η βιβλιογραφία που μελετήθηκε δύναται να οργανωθεί βάσει πολλαπλών διαστάσεων κατηγοριοποίησης, κάθε μια από τις οποίες αναδεικνύει διαφορετικές πτυχές της ερευνητικής δραστηριότητας. Οι κατηγορίες των διαστάσεων επεξηγούνται στην Ενότητα 4.1.3. Μια πρώτη βασική διάσταση αφορά τον τύπο συνεισφοράς των μελετών, όπου διακρίνονται προσεγγίσεις που προτείνουν νέους αλγορίθμους, νέα μοντέλα και πλαίσια υποστήριξης που βρίσκουν εφαρμογή στο πεδίο της δοκιμής λογισμικού. Μια δεύτερη διάσταση αφορά το είδος υποστήριξης που παρέχεται στη διαδικασία δοκιμών, όπως για παράδειγμα την παραγωγή περιπτώσεων ή δεδομένων δοκιμής, την ιεράρχηση δοκιμών ή ακόμη και την κατασκευή μηχανισμών αξιολόγησης αποτελεσμάτων (test oracles). Οι μελέτες μπορούν επίσης να ταξινομηθούν με βάση το είδος τεχνητής νοημοσύνης που αξιοποιούν, συμπεριλαμβανομένων τεχνικών μηχανικής μάθησης, βαθιάς μάθησης, επεξεργασίας φυσικής γλώσσας και ασαφούς λογικής. Μια τελευταία δυνατή διάσταση κατηγοριοποίησης αφορά το επίπεδο των συστατικών που δοκιμάζονται, στο οποίο εφαρμόζεται η τεχνητή νοημοσύνη, όπως στο επίπεδο μεμονωμένων μονάδων, ενοποιήσεων συστατικών ή όλου του συστήματος. Στον Πίνακα 3 παρουσιάζεται η ταξινόμηση των προσεγγίσεων με βάση

τον εννοιολογικό χάρτη σε μία ή περισσότερες κατηγορίες ανά διάσταση. Ενώ η κατανομή των άρθρων ανά κατεύθυνση συνεισφοράς παρουσιάζεται στο Σχήμα 5.



Σχήμα 4. Εννοιολογικός χάρτης AI in software testing

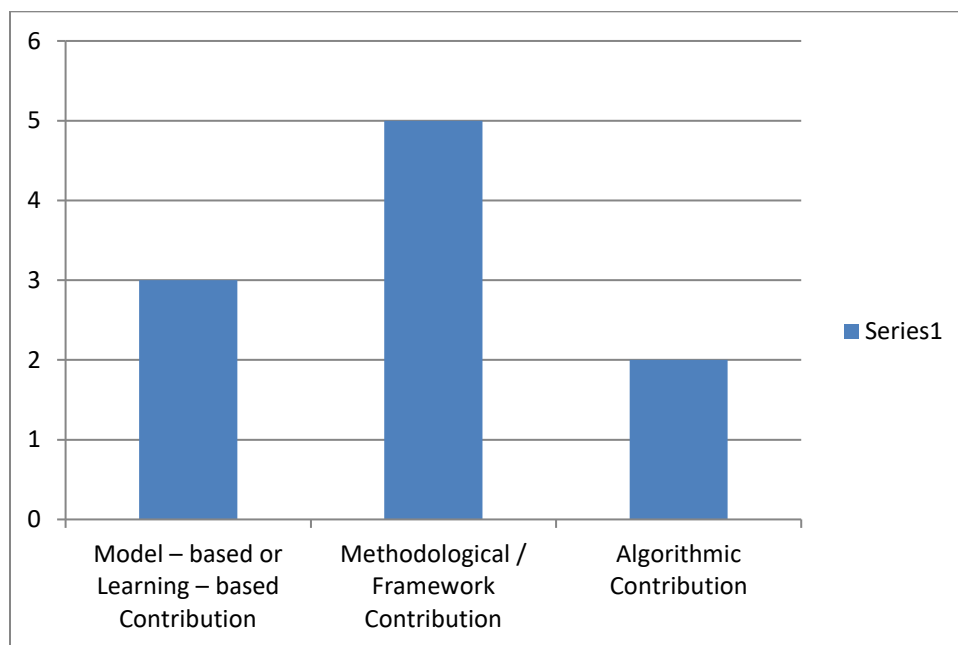
Πίνακας 3. Πίνακας ταξινόμησης των προσεγγίσεων AI με βάση τον αντίστοιχο εννοιολογικό χάρτη

ΑΡΘΡΟ	CONTRIBUTION TYPE	TESTING ACTIVITY	AI TECHNIQUE	TEST LEVEL
<i>A Machine Learning Approach to Generate Test Oracles(Braga et al., 2018)</i>	MODEL-BASED OR LEARNING-BASED CONTRIBUTION	ORACLE	MACHINE LEARNING	SYSTEM
<i>A method for IoT devices test case generation using language models (Kumar et al.,2025)</i>	METHODOLOGICAL / FRAMEWORK CONTRIBUTION	GENERATION	NATURAL LANGUAGE PROCESSING	SYSTEM,INTEGRATION

<i>Code Smell Detection Using Ensemble Machine Learning Algorithms (Dewangan et al., 2022)</i>	MODEL-BASED OR LEARNING-BASED CONTRIBUTION	DEFECT DETECTION	MACHINE LEARNING, DEEP LEARNING	UNIT
<i>Deep Reinforcement Learning for Black-box Testing of Android Apps (Romdhana et al., 2022)</i>	MODEL-BASED OR LEARNING-BASED CONTRIBUTION	GENERATION	DEEP LEARNING	SYSTEM
<i>Enriching automatic test case generation by extracting relevant test inputs from bug reports (Ouédraogo et al., 2025)</i>	METHODOLOGICAL / FRAMEWORK CONTRIBUTION	GENERATION	NATURAL LANGUAGE PROCESSING	UNIT
<i>FCCI: A fuzzy expert system for identifying coincidental correct test cases (Sabaghetal., 2020)</i>	METHODOLOGICAL / FRAMEWORK CONTRIBUTION	DEFECT DETECTION	FUZZY LOGIC	
<i>Finite State GUI Testing with Test Case Prioritization Using Z-BES and GK-GRU (Kumar et al., 2023)</i>	METHODOLOGICAL / FRAMEWORK CONTRIBUTION	PRIORITIZATION	DEEP LEARNING, REINFORCEMENT LEARNING, FUZZY LOGIC	SYSTEM
<i>Mutated Specification-Based Test Data Generation with a Genetic Algorithm (Wangetal., 2021)</i>	ALGORITHMIC CONTRIBUTION	GENERATION	GENETIC ALGORITHM	UNIT ⁹
<i>Novel Fuzzy Clustering Methods</i>	ALGORITHMIC CONTRIBUTION	PRIORITIZATION	FUZZY LOGIC	

⁹ Κατηγοριοποιήθηκε ως unit με βάση την πειραματική αξιολόγηση σε μεμονωμένες συναρτήσεις, αν και η μέθοδος δεν περιορίζεται εγγενώς σε αυτό το επίπεδο.

<i>for Test Case Prioritization in Software Projects(Shrivathsa n et al.,2019)</i>				
<i>TestLoter: A logic-driven framework for automated unit test generation and error repair using large language models (Yang et al.,2025)</i>	METHODOLOGICAL / FRAMEWORK CONTRIBUTION	GENERATION	NATURAL LANGUAGE PROCESSING	UNIT



Σχήμα 5. Κατανομή άρθρων ΑΙ στη δοκιμή λογισμικού ανά κυρία κατεύθυνση συνεισφοράς

4.1.3 ΑΝΑΛΥΣΗ ΣΥΝΕΙΣΦΟΡΩΝ

Η ανάλυση της κάθε επιλεγμένης μελέτης, η οποία προτείνει μια συγκεκριμένη συνεισφορά, πραγματοποιήθηκε με βάση τον τύπο συνεισφοράς, καθώς η διάσταση αυτή επιτρέπει την ανάδειξη της μεθοδολογικής εξέλιξης του πεδίου. Σε αυτό το πλαίσιο, η ανάλυση διαχωρίζεται σε ξεχωριστές υποενότητες όπου κάθε μία εστιάζει σε διαφορετικό τύπο συνεισφοράς.

4.1.3.1 MODEL-BASED OR LEARNING-BASED CONTRIBUTION

Αφορά εργασίες που εισάγουν μοντέλα μάθησης ή μοντέλα αναπαράστασης γνώσης (όπως τεχνικές μηχανικής και βαθιάς μάθησης, fuzzy συστήματα, γνωστικά μοντέλα ή μηχανισμούς συλλογισμού βασισμένους σε μεγάλα γλωσσικά μοντέλα) ως τον κύριο μηχανισμό υποστήριξης ή εκτέλεσης διαδικασιών ελέγχου λογισμικού. Η συνεισφορά των εργασιών αυτών εντοπίζεται στον τρόπο με τον οποίο το μοντέλο μαθαίνει, αναπαριστά πληροφορία ή λαμβάνει αποφάσεις. Συχνά, οι προτεινόμενες προσεγγίσεις συγκρίνονται με εναλλακτικά μοντέλα ή στρατηγικές μάθησης ως προς την αποτελεσματικότητά τους.

Παρατίθενται τώρα και αναλύονται τα αντίστοιχα άρθρα που εντάσσονται στη συγκεκριμένη κατεύθυνση συνεισφοράς.

4.1.3.1.1 A Machine Learning Approach to Generate Test Oracles

Στο άρθρο *A Machine Learning Approach to Generate Test Oracles* (Braga et al., 2018) παρουσιάζεται μια προσέγγιση βασισμένη στη μηχανική μάθηση με στόχο την αυτοματοποίηση δημιουργίας test oracles. Μέσω των test oracles ερμηνεύονται τα αποτελέσματα των δοκιμών και διακρίνεται κατά πόσο το υπό δοκιμή σύστημα συμπεριφέρεται κατά αναμενόμενο ή μη τρόπο. Η «χειροκίνητη» κατασκευή των test oracles αποτελεί σημαντικό εμπόδιο στην επίτευξη πλήρως αυτοματοποιημένων δοκιμών. Η προσέγγιση που παρουσιάζεται εστιάζει στη χρήση αρχείων καταγραφής δεδομένων χρήσης (logs), τα οποία φιλτράρονται και ταξινομούνται κατάλληλα, προκειμένου να εκπαιδευτεί ένα σύστημα μηχανικής μάθησης ώστε να συμπεριφερθεί ως test oracle. Τα δεδομένα χρήσης συλλέγονται και θεωρούνται ως έγκυρη συμπεριφορά, οπότε ένα υποσύνολο αυτών τροποποιείται κατάλληλα ώστε να επισημανθεί ως ελαττωματική συμπεριφορά, και έτσι δημιουργείται το τελικό σύνολο επισημασμένων δεδομένων για την εκπαίδευση του αλγορίθμου μηχανικής μάθησης (εποπτευόμενη μάθηση), ο οποίος αντίστοιχα λειτουργεί ως test oracle.

Οι συγγραφείς παραθέτουν το σύνολο των δεδομένων που προέκυψαν από την αλληλεπίδραση των χρηστών με το σύστημα που συλλέχθηκαν σε πίνακα, ωστόσο μεταξύ αυτών ξεχωρίζουν τις κατηγορίες “NumberOfVisibleElements” και “PageToBeLoaded”. Ο λόγος είναι γιατί αυτές αποτελούν γενικές πληροφορίες που μπορούν να συλλεχθούν ανεξάρτητα από την εφαρμογή υπό δοκιμή, αλλά και επειδή η αποτύπωση και ερμηνεία των οπτικών στοιχείων

σε συνδυασμό με την επόμενη σελίδα/«οθόνη» επαρκεί για να κατανοήσει κανείς αν η εφαρμογή επιδεικνύει ή όχι την αναμενόμενη συμπεριφορά. Οι συγγραφείς αιτιολογούν περαιτέρω αυτήν την επιλογή αναφέροντας πως η συντριπτική πλειοψηφία των εφαρμογών αποτυπώνουν το αποτέλεσμα των ενεργειών των χρηστών με οπτικό τρόπο. Μέσω αυτής της προσέγγισης, το σύστημα μπορεί να ταξινομεί αυτόματα τη συμπεριφορά της εφαρμογής ως έγκυρη (valid), ελαττωματική (faulty), ή πιθανόν ελαττωματική (potentially faulty), μειώνοντας σημαντικά την ανθρώπινη συμμετοχή στη διαδικασία δοκιμών.

Η προτεινόμενη μέθοδος αποτελείται από τέσσερα στάδια: καταγραφή ενεργειών χρήστη, προετοιμασία δεδομένων, εκπαίδευση μοντέλου μηχανικής μάθησης και αξιολόγηση μάθησης. Για την καταγραφή ενεργειών χρήστη, εισάγεται ένα στοιχείο καταγραφής ενεργειών και παρεμβάλλεται ένας διακομιστής καταγραφής (capture server) μεταξύ της εφαρμογής και του διακομιστή της εφαρμογής (application server). Για την προστασία της ιδιωτικότητας (privacy) των χρηστών δεν συλλέχθηκαν στοιχεία που θα μπορούσαν να ταυτοποιήσουν το χρήστη ή τα δεδομένα που χρησιμοποίησε κατά την αλληλεπίδρασή του με την εφαρμογή. Αντί αυτών, συλλέχθηκαν στοιχεία, όπως ο αριθμός των ορατών στοιχείων και η επόμενη σελίδα που φορτώθηκε. Στη συνέχεια, στο στάδιο προετοιμασίας των δεδομένων, χρησιμοποιείται ανάλυση κέρδους πληροφοριών (information gain) για να διατηρηθούν μόνο τα χαρακτηριστικά που έχουν σημασία για την ταξινόμηση. Επειδή τα συλλεχθέντα αρχεία καταγραφής αντιπροσωπεύουν μόνο σωστή συμπεριφορά, οι συγγραφείς εισάγουν τεχνητά σφάλματα τροποποιώντας χαρακτηριστικά σε τυχαία επιλεγμένα υποσύνολα των δεδομένων. Για παράδειγμα, ένα αρχείο καταγραφής μπορεί να περιγράψει πως μετά από μία ενέργεια χρήστη (π.χ. επιλογή του πλήκτρου «είσοδος») εμφανίζονται 240 στοιχεία στην διεπαφή και ο χρήστης μεταφέρεται στην οθόνη login. Για τη δημιουργία τεχνητών σφαλμάτων, οι συγγραφείς τροποποιούν αυτές τις τιμές, π.χ. αλλάζοντας τον αριθμό των ορατών στοιχείων σε 120 αντί 240 και την οθόνη που εμφανίζεται σε άλλη αντί αυτή του login. Η τροποποίηση αυτή δημιουργεί ένα παράδειγμα λανθασμένης συμπεριφοράς, που χρησιμοποιείται κατά την εκπαίδευση του μοντέλου. Αυτή η διαδικασία δημιουργεί επισημασμένα παραδείγματα έγκυρης, ελαττωματικής και ενδεχομένως ελαττωματικής συμπεριφοράς¹⁰ που απαιτούνται για την εποπτευόμενη

¹⁰ Ως ενδεχομένως ελαττωματική συμπεριφορά, οι συγγραφείς ορίζουν εκείνες τις περιπτώσεις δοκιμών των οποίων τα αναμενόμενα αποτελέσματα αποκλίνουν από τα αποτελέσματα που παράγονται από τη χρήση της εφαρμογής.

μάθηση. Στη συνέχεια, το βελτιωμένο σύνολο δεδομένων χρησιμοποιείται για την εκπαίδευση των σχετικών αλγορίθμων, συμπεριλαμβανομένου του IREP (JRIP)¹¹, οι οποίοι συνδυάζονται με τον αλγόριθμο AdaBoostM1¹² ώστε να παραχθεί το τελικό μοντέλο μάθησης για την ταξινόμηση των εκτελέσεων λογισμικού. Τέλος, η προσέγγιση επικυρώνεται εμπειρικά μέσω πειραμάτων που χρησιμοποιούν τυπικές μετρικές απόδοσης στη μηχανική μάθηση (accuracy, precision, recall, specificity, f – measure, ROC curve, AUC).

Θετικό στοιχείο της προσέγγισης είναι τα υψηλά ποσοστά αποτελεσματικότητας στην ανίχνευση αστοχιών, με ποσοστό accuracy 94% και 72% αντίστοιχα, στα 2 πρώτα πειράματα όπου έγινε εισαγωγή τυχαίων λαθών (στο δεύτερο πείραμα από εξειδικευμένο άτομο), 98% στο τρίτο πείραμα όπου χρησιμοποιήθηκε mutation testing¹⁴ και 79% σε πραγματική εφαρμογή μεγάλης κλίμακας. Επιπλέον, η χρήση πραγματικών δεδομένων αλληλεπιδράσεων χρηστών με το υπό δοκιμή σύστημα μπορεί να ευνοήσει τη διαρκή διεξαγωγή δοκιμών παλινδρόμησης (regression testing) σε πραγματικές συνθήκες παραγωγής λογισμικού. Οι δοκιμές παλινδρόμησης μπορούν να επιτευχθούν με την επαναληπτική συλλογή δεδομένων από τις αλληλεπιδράσεις χρηστών με το σύστημα υπό δοκιμή, ώστε να εμπλουτίζονται τα δεδομένα εκπαίδευσης και να βελτιώνεται η επίδοση του αλγορίθμου που λειτουργεί ως test oracle. Η προσέγγιση επιτυγχάνει τη σημαντική μείωση του κόστους των δοκιμών και του παράγοντα ανθρώπινου λάθους, ενώ παράλληλα επιτρέπει ταχύτερη ανατροφοδότηση για την ποιότητα του λογισμικού και τους πιθανούς επιχειρηματικούς κινδύνους, χάρη στην εξάλειψη της ανάγκης για χειροκίνητα καθορισμένα αναμενόμενα αποτελέσματα.

Παρά τα πολλά υποσχόμενα αποτελέσματα, οι συγγραφείς αναγνωρίζουν διάφορους περιορισμούς. Τα αρχικά πειράματα βασίστηκαν σε μεγάλο βαθμό σε μια φανταστική εφαρμογή μικρής κλίμακας (“ShoppingCart”), οπότε ενδεχομένως η προσέγγιση να μην είναι εξίσου

ωστόσο η διαφορά είναι ανεπαρκής για τον αλγόριθμο (λειτουργεί ως classifier) να υποδείξει αποτυχημένη λειτουργία της εφαρμογής.

¹¹ Ο αλγόριθμος IREP (Incremental Reduced Error Pruning) ανήκει στην ομάδα των "symbolic techniques". Χρησιμοποιείται για την επαναληπτική δόμηση συνόλου κανόνων ταξινόμησης σύμφωνα με τα παραδείγματα (των δεδομένων εκπαίδευσης).

¹² Ο αλγόριθμος AdaBoostM1 ανήκει στην κατηγορία αλγορίθμων “Boosting Algorithm family” και μάλιστα είναι από τους πιο γνωστούς. Οι αλγόριθμοι αυτοί χρησιμοποιούνται συνδυαστικά με αλγορίθμους μάθησης (learning algorithms) για να ενισχύσουν την απόδοσή τους.

¹⁴ Είναι μία τεχνική αξιολόγησης της αποτελεσματικότητας των δοκιμών λογισμικού κατά την οποία δημιουργούνται ελαφρώς τροποποιημένες εκδόσεις ενός προγράμματος (mutants) που προσομοιώνουν πιθανά σφάλματα προγραμματισμού. Η αποτελεσματικότητα της δοκιμής εξαρτάται από την ικανότητά της να ανιχνεύσει τις διαφοροποιήσεις στη συμπεριφορά των μεταλλαγμένων εκδόσεων (Jia & Harman, 2011).

αποτελεσματική (και κατά συνέπεια επεκτάσιμη) σε πραγματικές εφαρμογές μεγαλύτερης κλίμακας. Για να κριθεί κατά πόσο η προσέγγιση μπορεί να είναι αποδοτική σε ρεαλιστικά σενάρια, θα πρέπει να προηγηθεί περαιτέρω πειραματική επικύρωση σε πραγματικές εφαρμογές. Επειδή το μοντέλο μάθησης κατασκευάζεται αποκλειστικά από ιστορικά αρχεία καταγραφής αλληλεπιδράσεων χρηστών, η μέθοδος δεν μπορεί να λειτουργήσει σε συστήματα που έχουν αναπτυχθεί πρόσφατα και δεν διαθέτουν ιστορικό λειτουργίας, καθιστώντας την ακατάλληλη για επαλήθευση σε πρώιμο στάδιο ή επικύρωση πριν από την κυκλοφορία. Επιπλέον, η διαδικασία εκπαίδευσης εξαρτάται από τεχνητά εισαγόμενα σφάλματα για τη δημιουργία παραδειγμάτων λανθασμένης συμπεριφοράς («οι ενέργειες που συνθέτουν τη βάση δεδομένων καταγράφηκαν από μια λειτουργική έκδοση της εφαρμογής, δηλαδή οι καταγεγραμμένες πληροφορίες θεωρήθηκαν σωστές»), τα οποία ενδέχεται να μην αντικατοπτρίζουν πλήρως την πολυπλοκότητα και την ποικιλομορφία των πραγματικών ελαττωμάτων λογισμικού και να εισάγουν μεροληψία στη διαδικασία μάθησης. Κατά συνέπεια, ενώ η προσέγγιση προσφέρει σημαντικά οφέλη για την παρακολούθηση μετά την ανάπτυξη και την ανίχνευση σφαλμάτων σε ζωντανά συστήματα, η εξάρτησή της από τα δεδομένα των χρηστών και τα συνθετικά σφάλματα περιορίζει τη δυνατότητα εφαρμογής της σε ολόκληρο τον κύκλο ζωής ανάπτυξης λογισμικού καθώς και περιορίζει την αποτελεσματικότητά της ως καθολική λύση δοκιμών.

4.1.3.1.2 Deep Reinforcement Learning for Black-box Testing of Android Apps

Στο άρθρο *Deep Reinforcement Learning for Black-box Testing of Android Apps* (Romdhana et al., 2022) διερευνάται η χρήση τεχνικών Deep Reinforcement Learning για τη δοκιμή μαύρου κουτιού (black-box testing) σε εφαρμογές (apps) που εκτελούνται σε Android συσκευές. Οι συγγραφείς εστιάζουν στην αυτόματη παραγωγή περιπτώσεων δοκιμών χωρίς γνώση της εσωτερικής δομής του συστήματος. Η προσέγγιση αυτή αντιμετωπίζει το testing ως πρόβλημα λήψης αποφάσεων. Ο πράκτορας μάθησης (learning agent) μαθαίνει να αλληλεπιδρά με το σύστημα και να προσαρμόζει τη στρατηγική ελέγχου. Στόχος είναι η αποτελεσματικότερη ανίχνευση σφαλμάτων σε σύγκριση με παραδοσιακές τεχνικές δοκιμής μαύρου κουτιού.

Παρότι η προτεινόμενη προσέγγιση δεν απαιτεί πρόσβαση στον πηγαίο κώδικα, ο πράκτορας διαθέτει μηχανισμό αλληλεπίδρασης με την εφαρμογή μέσω της γραφικής διεπαφής χρήστη. Σε κάθε βήμα εξάγεται η ιεραρχία των διαθέσιμων στοιχείων της οθόνης, προσδιορίζονται οι εφικτές ενέργειες και επιλέγεται μία προς εκτέλεση, ενώ στη συνέχεια

παρατηρείται η νέα κατάσταση του συστήματος. Η ανταμοιβή βασίζεται σε εξωτερικά παρατηρήσιμες μεταβολές: όταν η νέα κατάσταση είναι παρόμοια με την προηγούμενη επιβάλλεται αρνητική ανταμοιβή, ενώ η μετάβαση σε νέα κατάσταση οδηγεί σε θετική ανταμοιβή. Με τον τρόπο αυτό, η εξερεύνηση νέων δραστηριοτήτων (activities) λειτουργεί ως υποκατάστατο (proxy) της πιθανής ενεργοποίησης διαφορετικών τμημάτων κώδικα. Η στρατηγική βασίζεται στην παρακολούθηση μεταβάσεων μεταξύ activities αντί στον συνεχή υπολογισμό κάλυψης. Η πραγματική κάλυψη μετριέται εκ των υστέρων για σκοπούς αξιολόγησης, μέσω καθιερωμένων εργαλείων instrumentation (στις συγκρίσεις με Sapienz (Mao et al., 2016) και TimeMachine χρησιμοποιείται το Emma¹⁵ ενώ στη σύγκριση με το Q-Testing γίνεται χρήση του JaCoCo¹⁶).

Η προτεινόμενη χρήση του συστήματος παρουσιάζει μια σειρά από σημαντικά πρακτικά πλεονεκτήματα. Πρώτον, υλοποιεί πλήρως αυτοματοποιημένο black-box testing, καθώς βασίζεται αποκλειστικά στις πληροφορίες που παρέχει η γραφική διεπαφή χρήστη της εφαρμογής υπό δοκιμή. Με τον τρόπο αυτό καθίσταται δυνατός ο έλεγχος εφαρμογών παραγωγής χωρίς καμία τροποποίηση ή instrumentation του κώδικα, σε αντίθεση με άλλα εργαλεία της βιβλιογραφίας που εξαρτώνται από δεδομένα κάλυψης για να κατευθύνουν την εξερεύνηση. Επιπλέον, προσφέρει ευρεία συμβατότητα, υποστηρίζοντας πολλαπλά λειτουργικά συστήματα ανάπτυξης και επιτρέποντας την εκτέλεση δοκιμών τόσο σε εξομοιωτές όσο και σε πραγματικές συσκευές Android για διαφορετικές εκδόσεις του λειτουργικού. Ένα ακόμη κρίσιμο χαρακτηριστικό είναι η δυνατότητα επαναχρησιμοποίησης της μαθημένης στρατηγικής, καθώς μετά την ολοκλήρωση της εκπαίδευσης το νευρωνικό μοντέλο αποθηκεύεται και μπορεί να εφαρμοστεί σε μεταγενέστερες εκδόσεις της ίδιας εφαρμογής, μειώνοντας σημαντικά τον χρόνο επανεκκίνησης της διαδικασίας. Τέλος, η αρχιτεκτονική του συστήματος είναι αρθρωτή, με σαφή διαχωρισμό μεταξύ του περιβάλλοντος της εφαρμογής και του αλγορίθμου ενισχυτικής μάθησης, γεγονός που ευνοεί την αντικατάσταση ή ενσωμάτωση νέων τεχνικών.

Παρά τα σημαντικά πλεονεκτήματα, η προσέγγιση συνοδεύεται και από ορισμένους περιορισμούς. Καταρχάς, τα οφέλη της ενισχυτικής μάθησης δεν είναι πάντοτε εμφανή σε απλούστερες εφαρμογές, όπου η απόδοση μπορεί να είναι συγκρίσιμη με αυτήν απλών

¹⁵ <https://emma.sourceforge.net/>

¹⁶ <https://www.jacoco.org/jacoco/>

στρατηγικών, όπως η τυχαία εξερεύνηση, γεγονός που σε ορισμένες περιπτώσεις δεν δικαιολογεί τη χρήση της μεθόδου αντί απλούστερων υφιστάμενων προσεγγίσεων. Επιπλέον, η υποστήριξη γεγονότων σε επίπεδο συστήματος παραμένει περιορισμένη. Αν και παρέχονται βασικές λειτουργίες, όπως η ενεργοποίηση ή απενεργοποίηση της σύνδεσης στο διαδίκτυο, πιο σύνθετες παρεμβάσεις απαιτούν rooted συσκευές¹⁷. Η απαίτηση αυτή ενδέχεται να δημιουργήσει πρακτικά προβλήματα, καθώς οι συγγραφείς επισημαίνουν πως υπάρχουν περιπτώσεις, όπου η ανίχνευση της κατάστασης root συνοδεύεται από αδυναμία εκκίνησης ή παύση εκτέλεσης της εφαρμογής που το προτεινόμενο σύστημα επιχειρεί να ελέγξει, με τη λογική πως έτσι αντιμετωπίζεται προληπτικά η ύπαρξη οποιουδήποτε πιθανού κενού ασφαλείας που προκύπτει από την άρση των περιορισμών του κατασκευαστή και μπορεί να εκθέσει τους χρήστες.

4.1.3.1.3 Code Smell Detection Using Ensemble Machine Learning Algorithms

Στο άρθρο *Code Smell Detection Using Ensemble Machine Learning Algorithms* (Dewangan et al., 2022) παρουσιάζεται ένα καινοτόμο συνδυαστικό πλαίσιο αλγορίθμων ensemble machine learning και deep learning για την ανίχνευση «οσμών κώδικα» (“code smells”). Οι οσμές κώδικα αποτελούν ενδείξεις κακού σχεδιασμού και υλοποίησης καθώς και αποκλίσεων από βασικές αρχές και “καλές πρακτικές” που έχουν θεσπιστεί ενώ έχουν αρνητικό αντίκτυπο τόσο στη δυνατότητα συντήρησης όσο και στην ποιότητα του λογισμικού. Το άρθρο έχει επιλεγεί προς ανάλυση, παρόλο που η ανίχνευση code smells δεν μπορεί να θεωρηθεί με αυστηρά κριτήρια είδος δοκιμής λογισμικού. Ο λόγος είναι διττός. Ένας από τους βασικούς στόχους της μελέτης μου είναι οι τρόποι με τους οποίους η χρήση τεχνητής νοημοσύνης μπορεί να συνεισφέρει στον τομέα της δοκιμής λογισμικού, όπου βασική πρόκληση αποτελεί η έγκαιρη ανίχνευση ελαττωμάτων και διόρθωση αυτών. Με την έγκαιρη αναγνώριση code smells και των αντίστοιχων ελαττωμάτων που αυτές υποδεικνύουν χάρη στη χρήση τεχνητής νοημοσύνης, μπορεί να μειωθεί ο χρόνος, η προσπάθεια και το κόστος διεξαγωγής δοκιμών. Επιπλέον, πως ανταποτελέσματα της συγκεκριμένης μελέτης είναι ενθαρρυντικά, τότε μια προτεινόμενη ερευνητική κατεύθυνση θα ήταν η επέκταση της χρήσης των αλγορίθμων ensemble και deep learning για την ανίχνευση ελαττωμάτων και όχι ενδείξεων αυτών.

Η προτεινόμενη προσέγγιση λειτουργεί αξιοποιώντας τέσσερα σύνολα δεδομένων αναφοράς για τον εντοπισμό code smells. Κάθε σύνολο δεδομένων αντιστοιχεί σε μία κατηγορία

¹⁷ Πρόκειται για συσκευές στις οποίες έχουν αρθεί οι περιορισμοί και ασφαλιστικές δικλίδες του κατασκευαστή, με αποτέλεσμα οι χρήστες να έχουν πλήρη πρόσβαση στο λειτουργικό σύστημα.

code smell και το πρόβλημα διατυπώνεται ως δυαδική ταξινόμηση (binary classification). Πριν από την εκπαίδευση των αλγορίθμων, οι συγγραφείς εφαρμόζουν την επιλογή χαρακτηριστικών Chi-square, μια στατιστική μέθοδο που αξιολογεί τον βαθμό εξάρτησης μεταξύ ενός χαρακτηριστικού και της κατηγορίας στόχου (Forman, 2003), για να εξαλείψουν τις ασθενώς συσχετισμένες μετρικές με την ανίχνευση code smells και να επιλέξουν τις μετρικές κώδικα με το μεγαλύτερο αντίκτυπο στην ανίχνευση code smells. Για να αντιμετωπίσει τη σοβαρή ανισορροπία κλάσεων που είναι εγγενής στα σύνολα δεδομένων για την ανίχνευση code smells (υπερισχύει η “καθαρή” κλάση αντί αυτής με code smell), το πλαίσιο χρησιμοποιεί στη συνέχεια την τεχνική SMOTE (Synthetic Minority Over-sampling Technique) class balancing για να δημιουργήσει συνθετικά δείγματα για τις μειοψηφικές κλάσεις ώστε να παραχθούν ισορροπημένα δεδομένα εκπαίδευσης. Τα βελτιωμένα σύνολα δεδομένων χρησιμοποιούνται για την εκπαίδευση πέντε αλγορίθμων μηχανικής μάθησης (AdaBoost, Bagging, Max Voting, Gradient Boosting και XGBoost) και δύο μοντέλων βαθιάς μάθησης (ANN¹⁸ και CNN¹⁹), επιτρέποντας την αξιόπιστη και ακριβή ταξινόμηση των code smells. Οι παραπάνω 5 αλγόριθμοι που χρησιμοποιήθηκαν αποτελούν ensemble μεθόδους, οι οποίες βασίζονται στον συνδυασμό πολλαπλών base learners, ωστόσο το άρθρο δεν προσδιορίζει ρητά την ακριβή μορφή τους, δίνοντας έμφαση στην απόδοση του συνολικού σχήματος. Η αποτελεσματικότητα του πλαισίου, και ειδικότερα των αλγορίθμων μηχανικής μάθησης (machine learning) και βαθιάς μάθησης (deep learning) που ενσωματώνει, αξιολογείται συστηματικά χρησιμοποιώντας τυπικές μετρικές απόδοσης (accuracy, precision, recall, F - measure), αξιολογώντας έτσι την απόδοση της ταξινόμησης. Οι συγγραφείς διαπιστώνουν πως κάθε αλγόριθμος ensemble και deep learning που χρησιμοποιήθηκε αποδίδει περισσότερο ή λιγότερο από τους υπόλοιπους ανάλογα με το σύνολο δεδομένων που χρησιμοποιείται για την ανίχνευση code smells. Δεν προκύπτει ξεκάθαρο συμπέρασμα λοιπόν για το αν κάποιος μεμονωμένος αλγόριθμος υπερσχύει έναντι των άλλων, καθώς επιτυγχάνονται διαφορετικά ποσοστά ακρίβειας (accuracy) ανά διαφορετικό συνδυασμό χαρακτηριστικών και συνόλων δεδομένων. Στο πλαίσιο της συγκεκριμένης μελέτης

¹⁸ (Artificial Neural Network) Είναι ένα μαθηματικό μοντέλο που βασίζεται σε χαρακτηριστικά βιολογικών νευρωνικών δικτύων, συμπεριλαμβανομένης της δομής και της λειτουργικότητάς τους. Χρησιμοποιεί μια τεχνητή νευρωνική μέθοδο υπολογισμού για την επεξεργασία δεδομένων και αποτελείται από ένα δίκτυο τεχνητών νευρώνων που συνδέονται μεταξύ τους (Grossi & Buscema, 2007).

¹⁹ (Convolutional Neural Network) Μια από τις πιο γνωστές και χρησιμοποιούμενες μεθόδους deep learning. Το βασικό του πλεονέκτημα σε σχέση με μεθόδους που έχουν προηγηθεί και το καθιστούν δημοφιλέστερο αντί αυτών, είναι πως αναγνωρίζει σημαντικά χωρικά στοιχεία με ελάχιστο αριθμό παραμέτρων σε σχέση με πλήρως συνδεδεμένα δίκτυα. (Alzubaidi et al., 2021)

φαίνεται πως εν τέλει είναι καλύτεροι οι ensemble αλγόριθμοι, καθώς επιτυγχάνουν ποσοστό accuracy 100% επανειλημμένα και σε πολλαπλές ρυθμίσεις, σε αντίθεση με τους deep learning, όπου δεν επιτυγχάνεται 100% accuracy σε κανένα από τα σύνολα δεδομένων που εξετάστηκαν. Ως ομάδα/τύπος αλγορίθμων δηλαδή υπερισχύουν οι ensemble των deep learning στη συγκεκριμένη μελέτη, αλλά δεν υπάρχει κάποιος συγκεκριμένος αλγόριθμος είτε ensemble είτε deep learning που να υπερισχύει καθολικά έναντι όλων των υπολοίπων.

Βασικό θετικό στοιχείο του πλαισίου είναι η πολύ υψηλή ακρίβεια ανίχνευσης, επιτυγχάνοντας ποσοστό ακόμη και 100% για διάφορους τύπους code smells, κυρίως FeatureEnvy²⁰ και LongMethod²¹. Επιβεβαιώνεται έτσι η υψηλή αποτελεσματικότητα των τεχνικών ensemble learning και η υπεροχή της προσέγγισης αντί άλλων αντίστοιχων που είχαν προηγηθεί σε άλλες μελέτες. Επιπλέον, η ενσωμάτωση της επιλογής χαρακτηριστικών Chi-square και της εξισορρόπησης κλάσεων SMOTE επιτρέπει την καλύτερη διαχείριση μη ισορροπημένων συνόλων δεδομένων. Η προσέγγιση έχει επικυρωθεί σε τέσσερις κατηγορίες code smells (DataClass, GodClass, FeatureEnvy, and LongMethod) και προσφέρει σημαντικά πρακτικά οφέλη, περιορίζοντας τη λήψη κακών αποφάσεων σχεδιασμού και υλοποίησης που καθιστούν το λογισμικό πιο δύσκολο και πιο ακριβό να συντηρηθεί με την πάροδο του χρόνου.

Ως προς τα αρνητικά στοιχεία, η αξιολόγηση περιορίζεται ρητά σε δύο επίπεδα code smells (class – level και method level) και τέσσερις κατηγορίες αυτών. Εγείρονται επομένως προβληματισμοί για τη δυνατότητα εφαρμογής του πλαισίου με εξίσου θεαματική ακρίβεια σε ένα ευρύτερο φάσμα γνωστών code smells. Ενδεικτικά, πέρα από τα smells επιπέδου κλάσης και μεθόδου, η βιβλιογραφία αναγνωρίζει και code smells αρχιτεκτονικής κλίμακας, όπως κυκλικές εξαρτήσεις μεταξύ πακέτων, υπερσυγκέντρωση ευθυνών σε κεντρικά συστατικά και παραβιάσεις επιπέδων αρχιτεκτονικής (Mo et al., 2016), τα οποία δεν εξετάζονται στην παρούσα μελέτη. Ακόμη, όταν η αναμενόμενη τιμή σε ένα χαρακτηριστικό είναι μικρότερη από πέντε, το Chi-square μπορεί να οδηγήσει σε σφάλματα στα συμπεράσματα. Οι συντάκτες, αναλύοντας τα σύνολα των δεδομένων διαπίστωσαν πως αυτή η συνθήκη δεν προκύπτει στο σύνολο δεδομένων

²⁰ Περίπτωση κατά την οποία μια μέθοδος φαίνεται να ενδιαφέρεται περισσότερο για τα δεδομένα ή τις λειτουργίες άλλης κλάσης παρά για εκείνα της κλάσης στην οποία ανήκει, με αποτέλεσμα αυξημένη σύζευξη και μειωμένη συνοχή.

²¹ Μέθοδος με υπερβολικό μήκος ή πολυπλοκότητα, η οποία καθίσταται δυσκολότερη στην κατανόηση, τη συντήρηση και τον έλεγχο, συχνά υποδεικνύοντας την ανάγκη διάσπασης σε μικρότερες, πιο εστιασμένες λειτουργίες.

που χρησιμοποίησαν, ωστόσο θα μπορούσε κάλλιστα να προκύψει υπό άλλες συνθήκες (σε άλλο σύνολο δεδομένων). Επιπλέον, για την παραγωγή του συνόλου δεδομένων χρησιμοποιήθηκαν μονό εφαρμογές γραμμένες σε Java, οπότε η τεχνική που προτείνεται ενδέχεται να είναι ακατάλληλη για εφαρμογές σε άλλες γλώσσες προγραμματισμού, όπως C και C++. Οι συγγραφείς αναφέρουν πως ίσως οι μετρικές (απόδοσης) που χρησιμοποίησαν για την αξιολόγηση του πλαισίου να ήταν ανεπαρκείς, οπότε χρησιμοποίησαν επιπλέον μετρικές και τεχνικές μέτρησης απόδοσης (PPV, Sensitivity, F-measure, AUC_ROC_score, Accuracy, MCC, Cohen_Kappa_score) ως προσπάθεια θωράκισης των πειραματικών τους ευρημάτων. Ωστόσο, η σχεδόν τέλεια ακρίβεια (accuracy) που αναφέρθηκε, με βάση την οποία μάλιστα έγινε η τελική αξιολόγηση των αλγορίθμων, σε συνδυασμό με την εξάρτηση από σύνολα δεδομένων αναφοράς πιθανόν να αποτελεί ένδειξη overfitting και εγείρει ζητήματα αβεβαιότητας ως προς την επίτευξη αντίστοιχου ποσοστού σε άλλα σύνολα δεδομένων για το εκάστοτε υπό δοκιμή σύστημα λογισμικού. Τελικά, οι ίδιες μεθοδολογικές επιλογές που επιτρέπουν υψηλή απόδοση σε ελεγχόμενα σύνολα δεδομένων υποδεικνύουν αδυναμίες ως προς τη δυνατότητα επέκτασης και την ανθεκτικότητά τους σε μη ελεγχόμενα σύνολα.

4.1.3.2 ALGORITHMIC CONTRIBUTION

Αυτός ο τύπος συνεισφοράς αφορά εργασίες που προτείνουν είτε νέο αλγόριθμο ή ουσιαστική τροποποίηση υπάρχοντος αλγορίθμου που σχετίζεται με τη δοκιμή λογισμικού, με σαφή μαθηματικό ή υπολογιστικό μηχανισμό. Η συνεισφορά εντοπίζεται στον τρόπο επίλυσης του εκάστοτε προβλήματος (π.χ. clustering, search, optimization), ανεξάρτητα από το αν ο αλγόριθμος ενσωματώνεται σε ευρύτερο μεθοδολογικό πλαίσιο. Οι εργασίες αυτού του τύπου συνοδεύονται συνήθως από ψευδοκώδικα, σαφή αλγοριθμικά βήματα ή και θεωρητική ανάλυση της υπολογιστικής συμπεριφοράς των αλγορίθμων. Αντίστοιχα, η αξιολόγηση της συμπεριφοράς μπορεί να είναι πειραματική, ή να χρησιμοποιηθεί κάποιο είδος προσομοίωσης. Στην συνέχεια, αναλύουμε συγκεκριμένες μελέτες που αντιστοιχούν σε αυτό τον τύπο συνεισφοράς. Στο πλαίσιο της επιλογής μελετών συνεισφοράς της τεχνητής νοημοσύνης στη δοκιμή λογισμικού, εντάσσονται στην κατηγορία είτε μελέτες όπου ο αλγόριθμος εντάσσεται σε κατηγορία αλγορίθμων τεχνητής νοημοσύνης (γενετικός αλγόριθμος) είτε μελέτες όπου χρησιμοποιείται κάποια τεχνική τεχνητής νοημοσύνης (fuzzy clustering) σε κάποια από τα βήματα εκτέλεσης του αλγορίθμου.

4.1.3.2.1 Novel Fuzzy Clustering Methods for Test Case Prioritization in Software Projects

Το άρθρο *Novel Fuzzy Clustering Methods for Test Case Prioritization in Software Projects* (Shrivathsan et al., 2019) εστιάζει στη μείωση του κόστους της παλινδρομικής δοκιμής (regression testing) μέσω της προτεραιοποίησης περιπτώσεων δοκιμών (Test Case Prioritization-TCP) με τη χρήση fuzzy clustering (ασαφούς ομαδοποίησης).

Η προτεινόμενη προσέγγιση (fuzzy clustering) υλοποιείται μέσω της συστηματικής ομαδοποίησης και κατάταξης των test cases με βάση κοινά χαρακτηριστικά με στόχο την αποτελεσματικότερη προτεραιοποίησή τους. Γίνεται, δηλαδή, υπολογισμός του βαθμού ομοιότητας μεταξύ test cases μέσω ειδικού συντελεστή, ενσωμάτωση του μέτρου κυριαρχίας (dominancy) για την αξιολόγηση της σχετικής σημασίας τους, διπλή ομαδοποίηση test cases και faulty functions ώστε να λαμβάνεται υπόψη η μεταξύ τους σχέση ενώ εφαρμόζεται η μέθοδος WASPAS (Weighted Arithmetic Sum Product Assessment) (Zavadskas et al., 2012) για την τελική κατάταξη των test cases. Στόχος της διαδικασίας είναι η εκτέλεση πρώτα των test cases με τη μεγαλύτερη πιθανότητα εντοπισμού σφαλμάτων.

Η προτεινόμενη προσέγγιση παρουσιάζει πλεονεκτήματα αλλά και ορισμένους περιορισμούς στην πράξη. Καταρχάς, επιτυγχάνεται συστηματική προτεραιοποίηση test cases στο regression testing. Επιπροσθέτως, γίνεται καλύτερος συνδυασμός ομαδοποίησης και κατάταξης (clustering + ranking), καθώς εκτελούνται πρώτα τα test cases με μεγαλύτερη πιθανότητα εντοπισμού σφαλμάτων. Συνεπώς, βελτιώνεται η αποτελεσματικότητα της διαδικασίας testing και περιορίζεται ο φόρτος της παλινδρομικής δοκιμής.

Στα αρνητικά, η απόδοση της μεθόδου εξαρτάται σημαντικά από την επιλογή παραμέτρων, όπως ο αριθμός των συστάδων (clusters) και τα κατώφλια ομαδοποίησης (grouping thresholds), γεγονός που καθιστά κρίσιμη τη διαδικασία ρύθμισης/διαμόρφωσης. Η εφαρμογή της προτεινόμενης μεθόδου προϋποθέτει μια σειρά διαδοχικών σταδίων επεξεργασίας, όπως η μετατροπή γλωσσικών αξιολογήσεων σε fuzzy βαθμούς συσχέτισης, η αναγωγή σε δύο καταστάσεις βάσει κατωφλίων και η επίλυση προβλημάτων 0–1 ακέραιου προγραμματισμού με τη χρήση κατάλληλων εργαλείων βελτιστοποίησης, γεγονός που υποδηλώνει ότι η προσέγγιση συνεπάγεται αυξημένες απαιτήσεις υλοποίησης, ιδίως σε περιβάλλοντα μεγάλης κλίμακας.

4.1.3.2.2 Mutated Specification-Based Test Data Generation with a Genetic Algorithm

Στο άρθρο (Wang et al., 2021), η προτεινόμενη μέθοδος που παρουσιάζεται βασίζεται στη δημιουργία μεταλλαγμένων εκδοχών των αρχικών προδιαγραφών (mutated specifications), στις οποίες εισάγονται επιπλέον μεταβλητές (dummy variables). Στη συνέχεια, ένας γενετικός αλγόριθμος χρησιμοποιείται για την αναζήτηση κατάλληλων τιμών εισόδου που ικανοποιούν τις μεταλλαγμένες προδιαγραφές και μπορούν να χρησιμοποιηθούν για την παραγωγή δεδομένων δοκιμών. Τα υποψήφια δεδομένα δοκιμών αξιολογούνται μέσω συνάρτησης καταλληλότητας (fitness function) με βάση την ικανότητά τους να εντοπίζουν σφάλματα. Μέσω επαναληπτικών κύκλων εξέλιξης γίνεται σταδιακή επιλογή δεδομένων που είναι πιο αποδοτικά στην ανίχνευση σφαλμάτων και συντελούν στη βελτίωση της δοκιμής λογισμικού.

Αυτή η προσέγγιση παρουσιάζει πλεονεκτήματα αλλά και ορισμένους περιορισμούς. Συμβάλλει στη συστηματική παραγωγή δεδομένων δοκιμής βάσει προδιαγραφών, συνδυάζοντας mutation testing και γενετικούς αλγορίθμους, γεγονός που οδηγεί σε βελτίωση της κάλυψης των προδιαγραφών από τον έλεγχο του λογισμικού και ενίσχυση της ανίχνευσης σφαλμάτων. Αυτό αποδεικνύεται καθώς στο πλαίσιο αξιολόγησής της, η μελέτη συγκρίνεται με δυο κλασικές εφαρμογές αντίστοιχης συμβατικής μεθόδου SBT (Stocks & Carrington, 1996) που παράγει απευθείας δεδομένα εισόδου από τις προδιαγραφές με τη χρήση του μηχανισμού Z3 (De Moura & Bjørner, 2008), ο οποίος αναλαμβάνει να προσδιορίσει τιμές μεταβλητών που ικανοποιούν ένα σύνολο λογικών και μαθηματικών περιορισμών με σκοπό τη δημιουργία δεδομένων δοκιμής. Τα αποτελέσματα της σύγκρισης δείχνουν πως η προτεινόμενη προσέγγιση παράγει δεδομένα δοκιμών ικανά να εξαλείψουν μεταλλαγμένα προγράμματα, ξεπερνώντας σε απόδοση τη συμβατική μέθοδο.

Παρότι οι γενετικοί αλγόριθμοι χρησιμοποιούνται συχνά ως αποδοτικές ευρετικές τεχνικές αναζήτησης σε σύγκριση με εξαντλητικές προσεγγίσεις, στην παρούσα εργασία το υπολογιστικό κόστος αυξάνεται σημαντικά λόγω της διαδικασίας αξιολόγησης της καταλληλότητας των λύσεων, η οποία απαιτεί την επαναληπτική εκτέλεση όλων των μεταλλαγμένων προγραμμάτων (mutant programs) σε κάθε αντίστοιχη επανάληψη εκτέλεσης του γενετικού αλγορίθμου. Επιπλέον, υπάρχουν περιορισμοί ως προς τη γενίκευση σε διαφορετικά είδη προδιαγραφών, καθώς οι ίδιοι οι συγγραφείς επισημαίνουν πως ο αλγόριθμος δύναται να λειτουργήσει μόνο στην περίπτωση που υπάρχουν αποκλειστικά αριθμητικές

συσχετίσεις μεταξύ εισόδων και εξόδων και απαιτείται η προσθήκη επεκτάσεων (extensions) στην προσέγγισή τους για να επιτευχθεί η διεύρυνση του πεδίου εφαρμογής. Τέλος, η αξιολόγηση της μελέτης είναι περιορισμένη, καθώς συγκρίνεται απλώς με 2 εφαρμογές της συμβατικής μεθόδου.

4.1.3.3 METHODOLOGICAL / FRAMEWORKCONTRIBUTION

Αυτός ο τύπος συνεισφοράς περιλαμβάνει εργασίες που προτείνουν διαδικασίες, μεθοδολογίες ή μεθοδολογικά πλαίσια, τα οποία οργανώνουν τη διαδικασία του ελέγχου λογισμικού σε διακριτά στάδια ή ροές εργασίας. Η έμφαση δεν δίνεται στον επιμέρους αλγόριθμο, αλλά πολλών αλγορίθμων που υλοποιούν τα αντίστοιχα βήματα της διαδικασίας(π.χ. παραγωγή test cases, προτεραιοποίηση, εκτέλεση και επιδιόρθωση), καθώς και στον τρόπο με τον οποίο δομείται συνολικά η διαδικασία της δοκιμής. Οι εργασίες αυτού του τύπου περιγράφονται συχνά μέσω διαγραμμάτων ροής διαδικασιών, διαγραμμάτων δραστηριοτήτων ή αρχιτεκτονικών απεικονίσεων που αποτυπώνουν τα επιμέρους στάδια, τις αλληλεπιδράσεις και τη συνολική οργάνωση της προτεινόμενης λύσης. Οι εργασίες αυτές αναλύονται στη συνέχεια, αποκαλύπτοντας τις συνεισφορές τους.

4.1.3.3.1 A method for IoT devices test case generation using language models

Το άρθρο *A method for IoT devices test case generation using language models* (Kumar et al.2025)παρουσιάζει τη μέθοδο TCG-IoT, ένα πλαίσιο αυτόματης παραγωγής περιπτώσεων δοκιμών για συσκευές IoT. Στοχεύει στην αυτοματοποίηση της δοκιμής τόσο σε επίπεδο λογισμικού όσο και υλικού. Η προσέγγιση αξιοποιεί μεγάλα γλωσσικά μοντέλα (LLMs) για την παραγωγή δομημένων περιπτώσεων δοκιμών από τεχνικές προδιαγραφές. Το άρθρο εστιάζει στη βελτίωση της κάλυψης, αξιοπιστίας και αποδοτικότητας του IoT testing (δοκιμής IoT συσκευών) σε ρεαλιστικά σενάρια.

Αυτό επιτυγχάνεται χρησιμοποιώντας μια μέθοδο που αξιοποιεί LLMs και Retrieval-Augmented Generation (RAG) για τη μετατροπή προδιαγραφών σε test cases ειδικά για συσκευές IoT. Η αξιοποίηση RAG επιτρέπει την ενσωμάτωση εξωτερικής γνώσης, αντλώντας σχετικές πληροφορίες από κατάλληλη διανυσματική βάση δεδομένων (vector database) κατασκευασμένη από επιστημονικές και τεχνικές πηγές μειώνοντας τις ασάφειες. Το πλαίσιο, μάλιστα, λαμβάνει υπόψη ιδιαιτερότητες του IoT μέσω της ενσωμάτωσης εξειδικευμένης

γνώσης που ανακτάται από τεχνική τεκμηρίωση, πρότυπα και περιγραφές συσκευών. Οι πληροφορίες αυτές αξιοποιούνται για τον εμπλουτισμό των prompts του γλωσσικού μοντέλου.

Η προτεινόμενη μεθοδολογία παρουσιάζει σημαντικά πλεονεκτήματα ως προς τη δυνατότητα επέκτασης, τη συστηματικότητα και την αξιοπιστία δοκιμών σε περιβάλλοντα IoT. Η αρχιτεκτονική της επιτυγχάνει ευρεία κάλυψη των επιμέρους υποσυστημάτων, καθώς υποστηρίζει την παράλληλη επεξεργασία πολλαπλών συνόλων απαιτήσεων και την παραγωγή προσαρμοσμένων περιπτώσεων δοκιμής για επιμέρους υποσυστήματα. Καθίσταται έτσι δυνατή η εφαρμογή της τόσο σε μικρής κλίμακας εγκαταστάσεις όσο και σε σύνθετες υποδομές. Η αξιοποίηση της τεκμηριωμένης γνώσης μέσω του μηχανισμού RAG διασφαλίζει την ευθυγράμμιση τόσο με προκαθορισμένες απαιτήσεις και αρχές επαλήθευσης και επικύρωσης αλλά και με τις καθιερωμένες βέλτιστες πρακτικές της βιομηχανίας.

Παρά τα θετικά της, η προτεινόμενη μέθοδος παρουσιάζει και ορισμένους περιορισμούς. Καταρχάς, η αποτελεσματικότητα της μεθόδου εξαρτάται από την ποιότητα και πληρότητα των τεχνικών προδιαγραφών που δίνονται ως είσοδος. Ενδεικτικά, οι συγγραφείς αναφέρουν πως στην περίπτωση που δεν αναφέρονται ρητά λεπτομέρειες αναφορικά με την αυθεντικοποίηση και κρυπτογράφηση στο documentation που χρησιμοποιεί το προτεινόμενο πλαίσιο, υπάρχει κίνδυνος να μην ελεγχθούν επαρκώς τα χαρακτηριστικά ασφαλείας των υπό δοκιμή συστημάτων IoT. Το πλαίσιο περιέχει σημεία ελέγχου, όπου οι παραγόμενες περιπτώσεις δοκιμής επαληθεύονται από ειδικούς. Επομένως, παρά την προσπάθεια για αυτοματοποίηση, δεν εξαλείφεται η ανάγκη ανθρώπινης επίβλεψης, ειδικά όταν ελέγχεται η ορθότητα των περιπτώσεων δοκιμής. Αξίζει τέλος να σημειωθεί πως η αξιολόγηση της μεθόδου βασίζεται σε μελέτη περίπτωσης και περιλαμβάνει περισσότερο ποιοτικά και ενδεικτικά ποσοτικά στοιχεία, όπως relevance scores για τα αποτελέσματα των RAG queries. Δεν παρουσιάζονται όμως εκτενείς ποσοτικές μετρικές αξιολόγησης της ποιότητας των παραγόμενων test cases.

4.1.3.3.2 Enriching automatic test case generation by extracting relevant test inputs from bug reports

Στο άρθρο *Enriching automatic test case generation by extracting relevant test inputs from bug reports* (Ouédraogo et al., 2025) προτείνεται το εργαλείο BRMiner για τον εμπλουτισμό της αυτόματης παραγωγής περιπτώσεων δοκιμών μέσω bug reports (αναφορών σφαλμάτων)

Το άρθρο εισάγει μια νέα προσέγγιση που αξιοποιεί τα bug reports ως πηγή γνώσης για την ενίσχυση της αυτόματης παραγωγής περιπτώσεων δοκιμών τον εμπλουτισμό των δεδομένων εισόδου που χρησιμοποιούνται από εργαλεία αυτόματης παραγωγής δοκιμών. Αρχικά πραγματοποιείται διαχωρισμός μεταξύ φυσικού κειμένου και αποσπασμάτων πηγαίου κώδικα. Στα τμήματα κώδικα εφαρμόζεται στατική ανάλυση μέσω parser, δηλαδή μηχανισμού που αναγνωρίζει τη συντακτική δομή του προγράμματος και επιτρέπει την αξιόπιστη εξαγωγή στοιχείων, όπως κυριολεκτικά (literals), μεταβλητές και κλήσεις μεθόδων. Παράλληλα, από το ελεύθερο κείμενο αντλούνται πιθανές τιμές εισόδου με τη χρήση κανονικών εκφράσεων. Η διαδικασία αυτή οδηγεί σε ένα εκτεταμένο σύνολο υποψηφίων δεδομένων, το οποίο ενδέχεται να περιλαμβάνει θόρυβο ή άσχετες πληροφορίες. Για τον λόγο αυτό ακολουθεί στάδιο σημασιολογικού φιλτραρίσματος με τη βοήθεια μεγάλου γλωσσικού μοντέλου, το οποίο αξιολογεί τη συνάφεια κάθε υποψήφιας τιμής σε σχέση με το αναφερόμενο σφάλμα και διατηρεί εκείνες που είναι πιθανότερο να συμβάλουν στην αναπαραγωγή του. Τα τελικά επιλεγμένα δεδομένα εισάγονται ως πρόσθετα seeds στο εργαλείο EvoSuite (Fraser & Arcuri, 2011), τα οποία αναλαμβάνουν την παραγωγή, εκτέλεση και αξιολόγηση των περιπτώσεων δοκιμών. Συνεπώς, η μέθοδος δεν αντικαθιστά τους υπάρχοντες μηχανισμούς δοκιμής, αλλά τους καθοδηγεί με πιο ρεαλιστικές και στοχευμένες τιμές, αυξάνοντας την πιθανότητα αποκάλυψης σφαλμάτων.

Η εφαρμογή του BRMiner συνοδεύεται από πλεονεκτήματα αλλά και προκλήσεις. Με την εφαρμογή της προτεινόμενης προσέγγισης αντιμετωπίζεται το πρόβλημα των μη ρεαλιστικών ή τυχαίων test inputs, τα οποία παράγονται από τα υπάρχοντα εργαλεία. Το άρθρο δείχνει ότι ο εμπλουτισμός των περιπτώσεων δοκιμών με δεδομένα από bug reports μπορεί να ενισχύσει την αποτελεσματικότητα της δοκιμής, χωρίς να απαιτείται να αντικατασταθούν τα υπάρχοντα εργαλεία. Οι συγγραφείς επιβεβαίωσαν πειραματικά πως η χρήση του BRMiner σε σύγκριση με baseline προσεγγίσεις, επιτυγχάνει σημαντική αύξηση της δυνατότητας ανίχνευσης bugs και βελτίωση της κάλυψης κώδικα (code coverage). Ως προς την αποτελεσματικότητα, αναφέρουν πως συγκριτικά με τις baseline προσεγγίσεις, με τη χρήση του BRMiner επιτυγχάνεται ανίχνευση περισσότερων bugs με μικρότερο αριθμό δοκιμών (tests).

Στα αρνητικά συγκαταλέγεται η εξάρτηση της απόδοσης από την ποιότητα και πληρότητα των bug reports. Η αξιολόγηση βασίζεται κυρίως σε μετρικές όπως η ανίχνευση

σφαλμάτων και η κάλυψη κώδικα, οι οποίες, αν και σημαντικές, δεν αποτυπώνουν συνολικά την ποιότητα ενός συνόλου δοκιμών. Επιπλέον, η διαδικασία tokenization²² ενδέχεται να μην ανακτά όλα τα σχετικά δεδομένα εισόδου, ιδίως σε περιπτώσεις σύνθετων εκφράσεων ή συνενώσεων τιμών, γεγονός που μπορεί να οδηγεί σε υποεκτίμηση της αποτελεσματικότητας της μεθόδου. Για παράδειγμα, μια σύνθετη έκφραση όπως `temperature > 20 AND humidity < 30` μπορεί μετά τη διαδικασία να διασπαστεί σε μεμονωμένα tokens (`temperature`, `20`, `humidity`, `30`). Αν η μέθοδος επεξεργάζεται τα tokens ανεξάρτητα, ενδέχεται να χαθεί η λογική και δομική σχέση μεταξύ των επιμέρους συνθηκών, γεγονός που μπορεί να οδηγήσει σε ελλιπή ανάκτηση των σχετικών δεδομένων εισόδου. Η πειραματική διαδικασία περιορίστηκε επίσης από συγκεκριμένο χρονικό προϋπολογισμό και μικρό αριθμό επαναλήψεων, στοιχείο που ενδέχεται να επηρέασε την πληρότητα της διερεύνησης. Η αξιολόγηση πραγματοποιήθηκε αποκλειστικά σε έργα του συνόλου Defects4J²³ (Just et al., 2014), γεγονός που αφήνει ανοικτό το ζήτημα της δυνατότητας εφαρμογής σε διαφορετικά περιβάλλοντα και γλώσσες. Τέλος, η μελέτη εστιάζει στην ενσωμάτωση με το EvoSuite, χωρίς να διερευνά τη συμπεριφορά της προσέγγισης σε άλλα εργαλεία, ούτε αντιμετωπίζει σε βάθος το πρόβλημα της διατήρησης της συνάφειας των δεδομένων καθώς οι εκδόσεις του λογισμικού εξελίσσονται.

4.1.3.3 TestLoter: A logic-driven framework for automated unit test generation and error repair using large language models

Στο άρθρο *TestLoter: A logic-driven framework for automated unit test generation and error repair using large language models* (Yangetal., 2025) παρουσιάζεται η ανάπτυξη και εφαρμογή του TestLoter, ενός πλαισίου που χρησιμοποιεί μεγάλα γλωσσικά μοντέλα (LLM) τόσο για την αυτοματοποίηση της παραγωγής περιπτώσεων δοκιμής μονάδων, όσο και για τη διόρθωση λαθών στην παραγωγή τους. Η συνεισφορά του στο πεδίο προκύπτει από τα δύο (2) βασικά του χαρακτηριστικά. Χάρη στην προσέγγιση “Logic Driven Chain Thought”, η οποία καθοδηγεί συστηματικά τα LLMs να δημιουργούν περιπτώσεις δοκιμών συνδυάζοντας τις δυνατότητες δομημένης ανάλυσης των «δοκιμών λευκού κουτιού (white-box testing)» με τα χαρακτηριστικά λειτουργικής επικύρωσης των δοκιμών «μαύρου κουτιού», επιτυγχάνεται βαθιά σημασιολογική ανάλυση του κώδικα και μεγαλύτερα ποσοστά κάλυψης σε σύγκριση με πιο

²² Διαδικασία κατά την οποία το κείμενο διασπάται σε μικρότερες μονάδες πληροφορίας όπως λέξεις, αριθμούς ή σύμβολα, ώστε να καταστεί δυνατή η υπολογιστική τους επεξεργασία.

²³ Ευρέως χρησιμοποιούμενο αποθετήριο πραγματικών σφαλμάτων λογισμικού για τη γλώσσα Java, το οποίο παρέχει ελεγμένα bugs και αντίστοιχες διορθωμένες εκδόσεις

διαδεδομένα αντίστοιχα εργαλεία. Αντίστοιχα, η εφαρμογή ενός ιεραρχικού μηχανισμού επιδιόρθωσης που εφαρμόζει διαφοροποιημένες στρατηγικές επιδιόρθωσης προσαρμοσμένες στις διακριτές κατηγορίες σφαλμάτων που προκύπτουν σε περιπτώσεις δοκιμών από τη χρήση LLMs, εξασφαλίζει την παραγωγή σωστών εκτελέσιμων περιπτώσεων δοκιμής με αυτοματοποιημένο τρόπο.

Τα βασικά θετικά στοιχεία του TestLoter είναι πως, μετά από δοκιμές και συγκρίσεις με αντίστοιχα πλαίσια/εργαλεία παραγωγής περιπτώσεων δοκιμής, εξασφαλίζει μεγαλύτερη επάρκεια/ κάλυψη. Για την αξιολόγηση χρησιμοποιήθηκε κοινό σύνολο δεδομένων. Συγκεκριμένα, το πειραματικό υλικό προήλθε αφενός από το corpus αξιολόγησης του ChatUniTest (Chen et al., 2024) και αφετέρου από ευρέως χρησιμοποιούμενα έργα ανοιχτού κώδικα που φιλοξενοούνται στο MavenCentralRepository²⁴. Στο άρθρο, η κάλυψη διακρίνεται σε line coverage (μετρική ελέγχου που εκφράζει το ποσοστό των εκτελέσιμων γραμμών πηγαίου κώδικα που εκτελέστηκαν τουλάχιστον μία φορά κατά τη διάρκεια των δοκιμών) και branch coverage (μετρική που αποτιμά το ποσοστό των δυνατών εκβάσεων των σημείων απόφασης που εκτελέστηκαν κατά τις δοκιμές). Συγκεκριμένα, η χρήση του TestLoter εξασφαλίζει μεγαλύτερη ποσοστιαία κάλυψη line coverage και branch coverage 8.5% και 10.9% αντίστοιχα από το εργαλείο ChatUniTest²⁵ (Chen et al., 2024), καθώς και 10% και 15% αντίστοιχα από το εργαλείο EvoSuite²⁶ (Fraser & Arcuri, 2011). Συγκρίνοντας το ποσοστό αδυναμίας εκτέλεσης των τελικών περιπτώσεων δοκιμών που παράγονται από το TestLoter και το ChatUniTest, αναδεικνύεται μιας διαφορά της τάξης 15% (2.3% για το TestLoter έναντι 17.3% για το ChatUniTest). Τέλος, κατόπιν δοκιμών τόσο σε λογισμικό με καλά δομημένο κώδικα και τυποποιημένη μορφή διεπαφής όσο και σε λογισμικό με λιγότερο τυποποιημένο κώδικα, προκύπτει πως οι στρατηγικές επιδιόρθωσης του TestLoter μειώνουν αποτελεσματικά τον αριθμό των failed assertions, επιτυγχάνοντας μέση μείωση 21,6%.

Ως προς τα αρνητικά του στοιχεία, το πλαίσιο εξακολουθεί να έχει περιορισμούς όταν τα σφάλματα αφορούν τη διαχείριση εξαιρέσεων ή όταν η βασική αιτία (σφάλματος) βρίσκεται εκτός της ίδιας της διατύπωσης. Αυτό υποδηλώνει ότι, ενώ το TestLoter έχει καλή απόδοση στα πειράματά που περιλαμβάνει το άρθρο, απαιτούνται περαιτέρω βελτιώσεις για τη διαχείριση πιο

²⁴ Διαθέσιμο στο: <https://search.maven.org/>

²⁵ Διαθέσιμο στο <https://github.com/ZJU-ACES-ISE/ChatUniTest>

²⁶ Διαθέσιμο στο <https://github.com/EvoSuite/evosuite>

σύνθετων σεναρίων σφαλμάτων. Οι συντάκτες του άρθρου επισημαίνουν επίσης, πως λόγω περιορισμένου προϋπολογισμού (budget), χρησιμοποίησαν ως κύριο μοντέλο το DeepSeek – V3, καθώς και μικρότερα μοντέλα (Llama 3, DeepseekR-1). Τα μοντέλα αυτά συχνά δεν κατάφεραν να ερμηνεύσουν με ακρίβεια τις προτροπές, παράγοντας απαντήσεις που δεν πληρούσαν τα απαιτούμενα κριτήρια. Ως αποτέλεσμα, η γενίκευσή του TestLoter σε διαφορετικά LLMs απαιτεί περαιτέρω επικύρωση. Επιπρόσθετα, για την δοκιμή της απόδοσης του TestLoter χρησιμοποιήθηκε περιορισμένος αριθμός συνόλων δεδομένων, τα οποία ναι μεν περιλαμβάνουν διάφορα σενάρια δοκιμών, ωστόσο ενδέχεται τα αποτελέσματα της μελέτης να μην είναι ενδεικτικά για την απόδοση του TestLoter σε μεγαλύτερο εύρος εφαρμογών. Τέλος, οι συγγραφείς του άρθρου επισημαίνουν πως υπάρχουν περιθώρια βελτίωσης της ευελιξίας και της προσαρμοστικότητας του πλαισίου, καθώς η εσωτερική εγκυρότητα του συστήματος υποστηρίζεται από μια σταθερή αλλά περιορισμένη ροή εργασίας. Όταν το στάδιο δημιουργίας περιπτώσεων δοκιμών δεν καταφέρνει να παραγάγει βιώσιμες περιπτώσεις δοκιμών ή η φάση επιδιόρθωσης δεν είναι σε θέση να αντιμετωπίσει σφάλματα σύνταξης και εκτέλεσης, ολόκληρο το πλαίσιο καθίσταται μη λειτουργικό. αυτή η πρόκληση, στοχεύουν στη μετάβαση από το υπάρχον μοντέλο ροής εργασίας σε ένα μοντέλο βασισμένο σε πράκτορες.

4.1.3.3.4 FCCI: A fuzzy expert system for identifying coincidental correct test cases

Στο άρθρο *FCCI: A fuzzy expert system for identifying coincidental correct test cases* (Sabagh et al., 2020) παρουσιάζεται μια καινοτόμος προσέγγιση για την αντιμετώπιση του προβλήματος της «τυχαίας ορθότητας» (“coincidental correct”, “CC”) στις δοκιμές λογισμικού, όπου προβληματικά μέρη του υπό δοκιμή λογισμικού παράγουν σωστά αποτελέσματα κατά την εκτέλεσή του και έτσι δεν ανιχνεύονται. Στο άρθρο παρουσιάζεται ένα πλαίσιο, το οποίο συνδυάζει τον στατιστικό εντοπισμό σφαλμάτων με την «ασαφή λογική» (“fuzzy reasoning”), προκειμένου να εκτιμά με μεγαλύτερη ακρίβεια αν μια περίπτωση δοκιμής παράγει αποτέλεσμα που είναι τυχαία σωστό. Επιστρατεύοντας τεχνικές ασαφούς λογικής, το πλαίσιο επιδιώκει να βελτιώσει την ακρίβεια στον εντοπισμό σφαλμάτων και να ενισχύσει την αποδοτικότητα των διαδικασιών δοκιμής λογισμικού, μέσω της μοντελοποίησης της ασάφειας στη συμπεριφορά του προγράμματος υπό δοκιμή.

Το πλαίσιο FCCI (Fuzzy Coincidental Case Identification) λειτουργεί συλλέγοντας πρώτα προφίλ εκτέλεσης από επιτυχημένες και αποτυχημένες δοκιμές και χρησιμοποιώντας

παραδοσιακές τεχνικές εντοπισμού σφαλμάτων υπολογίζει «βαθμολογίες ύποπτης συμπεριφοράς». Με τον όρο «προφίλ εκτέλεσης», εννοούμε μια δομημένη καταγραφή των τμημάτων ενός προγράμματος που εκτελέστηκαν κατά τη διάρκεια μιας περίπτωσης δοκιμής, η οποία παρέχει τα δεδομένα συμπεριφοράς που απαιτούνται για τον εντοπισμό σφαλμάτων και την ανίχνευση CC. Αυτές οι βαθμολογίες, μαζί με στατικές μετρήσεις πολυπλοκότητας κώδικα και χαρακτηριστικά εκτέλεσης, χρησιμοποιούνται στη συνέχεια ως εισροές σε ένα ασαφές σύστημα εμπειρογνομόνων (fuzzy expert system). Οι στατικές μετρήσεις πολυπλοκότητας κώδικα και τα χαρακτηριστικά εκτέλεσης χρησιμεύουν ως συμπληρωματικές πηγές στοιχείων που καθοδηγούν τη διαδικασία λήψης αποφάσεων με τη χρήση ασαφούς λογικής (fuzzy logic). Αυτά τα ποσοτικά χαρακτηριστικά μετατρέπονται με τη χρήση συναρτήσεων συμμετοχής (membership functions) σε “βαθμούς εννοιών” όπως χαμηλό, μέτριο και υψηλό. Οι όροι χαμηλό, μέτριο και υψηλό δεν αναφέρονται στον βαθμό ασάφειας της συμπεριφοράς του τμήματος κώδικα, αλλά στο μέγεθος των αριθμητικών τιμών των παραγόντων που χρησιμοποιούνται για την εκτίμηση της coincidental correctness. Στη συνέχεια, εφαρμόζεται ένα σύνολο ασαφών κανόνων που έχουν οριστεί από ειδικούς προκειμένου να εκτιμηθεί η πιθανότητα ένα επιτυχημένο αποτέλεσμα περίπτωσης δοκιμής να είναι τυχαία σωστό. Η αρχιτεκτονική του πλαισίου επιτρέπει τη συστηματική ταξινόμηση των δοκιμαστικών σεναρίων CC και μη CC, συνοδευόμενη από εκτενή εμπειρική αξιολόγηση σε πολλαπλά προγράμματα αναφοράς.

Το βασικό πλεονέκτημα του FCCI είναι η ικανότητά του να εντοπίζει με σαφήνεια τυχαία σωστές περιπτώσεις δοκιμών, διορθώνοντας έτσι μια σημαντική αδυναμία των παραδοσιακών μεθόδων εντοπισμού σφαλμάτων. Η χρήση της ασαφούς λογικής επιτρέπει στο πλαίσιο να λειτουργεί αποτελεσματικά σε συνθήκες αβεβαιότητας, επιτρέποντας τη λήψη αποφάσεων ακόμη και όταν τα στοιχεία από τη συμπεριφορά εκτέλεσης είναι αντιφατικά ή ελλιπή. Τα πειραματικά αποτελέσματα αποκαλύπτουν σημαντικές βελτιώσεις στην ακρίβεια εντοπισμού σφαλμάτων, συμπεριλαμβανομένης υψηλότερης ακρίβειας και καλύτερων βαθμολογιών EXAM (η μετρική που χρησιμοποιείται για την αξιολόγηση των συστημάτων εντοπισμού των περιπτώσεων CC όπου χαμηλότερη βαθμολογία σημαίνει πως πρέπει να ελεγχθεί μικρότερο μέρος του κώδικα για τον εντοπισμό λαθών) σε σύγκριση με τις υπάρχουσες τεχνικές. Για παράδειγμα, στο Siemenssuite, με έλεγχο μικρότερο του 10% του κώδικα, η DStar3 σε συνδυασμό με FCCI εντοπίζει το 86.5% των σφαλμάτων, ενώ η αμέσως καλύτερη ανταγωνιστική προσέγγιση DStar3 σε συνδυασμό με Feyzi εντοπίζει 78.1%. Με την

ενσωμάτωση πολλαπλών πηγών στοιχείων (δεδομένων εκτέλεσης, στατικών μετρήσεων και μετρήσεων ύποπτων στοιχείων) το FCCI παρέχει μια ισχυρή και πρακτική λύση που μειώνει σημαντικά τον κόπο των προγραμματιστών για τον εντοπισμό σφαλμάτων.

Παρά την ισχυρή απόδοσή του, το FCCI παρουσιάζει περιορισμούς που σχετίζονται με την επεκτασιμότητα και την αυτοματοποίηση. Για την κατασκευή fuzzy membership functions και fuzzy rule sets απαιτείται σημαντική εξειδίκευση στον τομέα. Η διαμόρφωσή τους κατά συνέπεια είναι χρονοβόρα και ενδέχεται να είναι υποκειμενική. Λόγω των παραπάνω, δεν μπορούμε να ισχυριστούμε πως το πλαίσιο συνεισφέρει ιδιαίτερος στην αυτοματοποίηση της ανίχνευσης CC, λόγω της προεργασίας που απαιτείται από εξειδικευμένο προσωπικό για να είναι λειτουργικό σε διαφορετικά περιβάλλοντα δοκιμής λογισμικού. Ο πρόσθετος υπολογιστικός φόρτος που εισάγεται από την «ασαφή συμπερασματολογία²⁷» και την εξαγωγή χαρακτηριστικών μπορεί να εμποδίσει την ανάπτυξη σε μεγάλης κλίμακας βιομηχανικά συστήματα. Επιπλέον, η αποτελεσματικότητα του πλαισίου εξαρτάται από τη διαθεσιμότητα ποικίλων και ολοκληρωμένων εκτελέσεων δοκιμών, περιορίζοντας τη δυνατότητα εφαρμογής του σε πρώιμα στάδια ανάπτυξης ή σε περιβάλλοντα με σποραδικά δεδομένα δοκιμών. Συνεπώς, διαφαίνεται η ανάγκη μελλοντικής έρευνας για τη δυνατότητα αυτοματοποίησης της δημιουργίας «fuzzy logic» και των επιμέρους δομικών της στοιχείων («fuzzy rules», “membership functions”, “fuzzy inference”) καθώς επίσης και για τη βελτίωση της αποδοτικότητας εφαρμογής fuzzy logic για να μπορέσει να επιτευχθεί μεγαλύτερη υιοθέτηση του πλαισίου.

4.1.3.3.5 Finite State GUI Testing with Test Case Prioritization Using Z-BES and GK-GRU

Το άρθρο *Finite State GUI Testing with Test Case Prioritization Using Z-BES and GK-GRU* (Kumar et al., 2023) παρουσιάζει ένα ολοκληρωμένο πλαίσιο για τη βελτίωση της ακρίβειας και της αποτελεσματικότητας των δοκιμών γραφικής διεπαφής χρήστη (Graphical User Interface–GUI) μέσω της ενσωμάτωσης μοντελοποίησης πεπερασμένων καταστάσεων, προτεραιοποίησης περιπτώσεων δοκιμών και τεχνικών βαθιάς μάθησης. Οι συγγραφείς προτείνουν μια υβριδική προσέγγιση που συνδυάζει τον αλγόριθμο βελτιστοποίησης Z Score–Bald Eagle Search (Z-BES) με έναν ταξινομητή Gini Kernel – Gated Recurrent Unit (GK-GRU)

²⁷ “fuzzy inference” είναι η διαδικασία λήψης αποφάσεων όπου συνδυάζοντας τους κανόνες «fuzzy rules» και εκτελώντας τις membership functions, υπολογίζεται η πιθανότητα ενός CC case και γίνεται η αντίστοιχη κατηγοριοποίησή του.

για τη βελτίωση των αυτοματοποιημένων δοκιμών GUI. Ο πρωταρχικός στόχος είναι να αντιμετωπιστούν οι περιορισμοί των παραδοσιακών δοκιμών GUI (όπως η ανεπαρκής κάλυψη, τα προβλήματα επεκτασιμότητας και η αδυναμία χειρισμού σύνθετων αλληλεπιδράσεων των χρηστών) μέσω της κατασκευής ενός έξυπνου, βασισμένου σε δεδομένα αγωγού (pipeline) που αξιολογεί συστηματικά τη συμπεριφορά του GUI και εντοπίζει πιθανές αναντιστοιχίες μεταξύ του σχεδιασμού της διεπαφής χρήστη και των απαιτήσεων του λογισμικού.

Η προτεινόμενη μεθοδολογία ξεκινά με τη συλλογή ιστορικών δεδομένων του έργου, συμπεριλαμβανομένων αρχείων GUI, περιπτώσεων δοκιμών και αρχείων καταγραφής εκτέλεσης. Στη συνέχεια, οι περιπτώσεις δοκιμών ιεραρχούνται με βάση τον αλγόριθμο Z-BES, ο οποίος εστιάζει αποτελεσματικά τις προσπάθειες δοκιμών σε περιοχές υψηλού αντίκτυπου, δηλαδή περιοχές που σύμφωνα με τα κριτήρια του αλγορίθμου είναι πιο πιθανό να ανιχνεύσουν αστοχίες. Από τις ιεραρχημένες περιπτώσεις δοκιμών, εξάγονται βασικά χαρακτηριστικά και δημιουργείται ένα διάγραμμα μετάβασης κατάστασης (State Transition Diagram, STD) για τη μοντελοποίηση της συμπεριφοράς του συστήματος. Η σημασία κάθε κατάστασης ποσοτικοποιείται με τη χρήση ενισχυτικής μάθησης, παράγοντας ένα σκορ δραστηριότητας κατάστασης (State Activity Score, SAS). Παράλληλα, εξάγονται τα στοιχεία GUI και το κειμενικό τους περιεχόμενο και συγκρίνονται με τα χαρακτηριστικά των περιπτώσεων δοκιμών χρησιμοποιώντας ένα μέτρο ομοιότητας. Αυτές οι βαθμολογίες ομοιότητας και οι τιμές SAS συνδυάζονται μέσω ενός συστήματος “ασαφούς λογικής” (fuzzy logic) για να χαρακτηριστούν τα αποτελέσματα των δοκιμών. Για να καταστεί ισχυρή η μάθηση, τα κειμενικά δεδομένα μετατρέπονται σε αριθμητικές αναπαραστάσεις χρησιμοποιώντας ενσωμάτωση λέξεων GS-BERT, και τέλος κατόπιν αυτής το μοντέλο GK-GRU εκτελεί την τελική ταξινόμηση της απόδοσης GUI.

Το πλαίσιο επιδεικνύει εξαιρετική απόδοση, επιτυγχάνοντας ποσοστό άνω του 98% στις τυπικές μετρικές αξιολόγησης συστημάτων μηχανικής μάθησης (accuracy, precision, recall, F-measure, sensitivity), διατηρώντας παράλληλα χαμηλά ποσοστά ψευδών θετικών (false positives) και ψευδών αρνητικών (false negatives) αποτελεσμάτων. Οι μετρικές αφορούν την αποτελεσματικότητα του μηχανισμού GK-GRU. Ο αλγόριθμος Z-BES μειώνει σημαντικά τον χρόνο προτεραιοποίησης των δοκιμών (5587 ms), ενώ το μοντέλο GK-GRU βελτιώνει περαιτέρω την ταχύτητα μάθησης και την ποιότητα ταξινόμησης. Συνδυάζοντας

βελτιστοποίηση, μοντελοποίηση πεπερασμένων καταστάσεων, βαθιά μάθηση (deep learning) και ασαφή λογική (fuzzy logic), η προσέγγιση προσφέρει μια εξαιρετικά δομημένη και επεκτάσιμη λύση ικανή να διαχειρίζεται σύνθετα περιβάλλοντα GUI. Η ενσωμάτωση της ανάλυσης ομοιότητας και της ενισχυτικής μάθησης επιτρέπει την ακριβή αναγνώριση της ελαττωματικής συμπεριφοράς του GUI, οδηγώντας σε βελτιωμένη ανίχνευση ελαττωμάτων και καλύτερη ευθυγράμμιση μεταξύ της συμπεριφοράς του συστήματος και των προκαθορισμένων απαιτήσεων.

Παρά την ισχυρή απόδοσή της, η προσέγγιση παρουσιάζει αρκετούς περιορισμούς. Το σύστημα βασίζεται σε μεγάλο βαθμό σε καλά δομημένα ιστορικά δεδομένα, τα οποία ενδέχεται να μην είναι άμεσα διαθέσιμα για εφαρμογές που έχουν αναπτυχθεί πρόσφατα ή εξελίσσονται ραγδαία. Το πλαίσιο παρουσιάζει επί του παρόντος μειωμένη αποτελεσματικότητα κατά τη διαχείριση GUI με μη συμβατικές διατάξεις ή κακώς δομημένα οπτικά χαρακτηριστικά, καθώς εστιάζει κυρίως στην ομοιότητα κειμένου και στη συμπεριφορά κατάστασης και όχι σε λεπτομερή οπτικά χαρακτηριστικά. Επιπλέον, η υπολογιστική πολυπλοκότητα της ενσωμάτωσης πολλαπλών προηγμένων τεχνικών (βελτιστοποίηση, ενισχυτική μάθηση, βαθιά μάθηση και ασαφής συμπερασμός) μπορεί να περιορίσει την πρακτική εφαρμογή σε περιβάλλοντα με περιορισμένους πόρους υπολογιστικής ισχύος. Τέλος, οι συγγραφείς επισημαίνουν πως η επιμονή των σφαλμάτων ταξινόμησης συνδέεται με την ανεπαρκή μοντελοποίηση της εμφάνισης του GUI και των χαρακτηριστικών δραστηριότητας, γεγονός που υποδηλώνει ότι μελλοντικά θα πρέπει να ενσωματωθεί η βελτιωμένη εξαγωγή οπτικών χαρακτηριστικών και η τμηματοποίηση του GUI, προκειμένου να εξαχθούν πιο λεπτομερή οπτικά χαρακτηριστικά ανά τμήμα, ώστε οι δοκιμές να μπορούν να εστιάσουν σε κάθε τμήμα ξεχωριστά.

4.2. ΣΥΝΕΙΣΦΟΡΑ ΤΗΣ ΔΟΚΙΜΗΣ ΛΟΓΙΣΜΙΚΟΥ ΣΤΗΝ ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

4.2.1 ΔΥΝΑΤΑ ΣΗΜΕΙΑ & ΑΔΥΝΑΜΙΕΣ

Στα δυνατά σημεία της συνεισφοράς της δοκιμής λογισμικού στην τεχνητή νοημοσύνη, συγκαταλέγονται η εισαγωγή νέων μετρικών και τεχνικών δοκιμής (neuron-coverage, coverage guided fuzzing) που επιτρέπουν τη συστηματικότερη αξιολόγηση συστημάτων που χρησιμοποιούν DNN, και η επέκταση παραδοσιακών τεχνικών δοκιμής λογισμικού (MC/DC) σε περιβάλλοντα τεχνητής νοημοσύνης. Επίσης, τεχνικές όπως το metamorphic testing παρέχουν τη

δυνατότητα αξιολόγησης συστημάτων τεχνητής νοημοσύνης όταν δεν είναι εφικτή η χρήση παραδοσιακών μηχανισμών test oracle. Αυτό είναι κάτι που απαντάται σε συστήματα τεχνητής νοημοσύνης, καθώς δεν είναι πάντοτε εφικτό να προκαθοριστεί μια αναμενόμενη σωστή τιμή. Παρά τα πλεονεκτήματα αυτά, η εφαρμογή των προτεινόμενων μεθόδων μπορεί να αυξήσει το υπολογιστικό κόστος κυρίως λόγω επαναληπτικών διαδικασιών ή εφαρμογής τους σε μοντέλα μεγαλύτερης κλίμακας (π.χ. DNN μεγαλύτερου βάθους) ή λόγω χρήσης μεγάλους σε πλήθους μεταμορφικών σχέσεων.

Πίνακας 4. Ανάλυση δυνατών σημείων & αδυναμιών στο πεδίο εφαρμογής της δοκιμής λογισμικού στην AI

ΔΥΝΑΤΑ ΣΗΜΕΙΑ	ΑΔΥΝΑΜΙΕΣ
Αντικατάσταση παραδοσιακών μεθόδων και μετρικών δοκιμής λογισμικού με νέες που ανταποκρίνονται επαρκώς σε σύγχρονα συστήματα τεχνητής νοημοσύνης Methodological / Framework Contribution π.χ. αντικατάσταση code coverage από neuron coverage – Pei et al., 2019, Xie et al., 2019) και επίτευξη επέκτασης καθιερωμένων πρακτικών testing σε σύγχρονα περιβάλλοντα/προβλήματα (Sun et al., 2019)	Επαναληπτικές διαδικασίες αυξάνουν το υπολογιστικό κόστος (επαναλαμβανόμενες εκτελέσεις για την αναζήτηση boundary candidates, Dobslaw & Feldt, 2024, προσεγγίσεις fuzzy logic, Xie et al., 2019)
Δυνατότητα συστηματικής δοκιμής μοντέλων AI με τη χρήση DNN ως test oracles (differential testing, coverage-guided fuzzing) (Pei et al., 2019, Xie et al., 2019)	Αυξημένο υπολογιστικό κόστος από την επέκταση των προτεινόμενων μεθοδολογιών σε πιο περίπλοκα συστήματα (π.χ. DNN μεγαλύτερου βάθους) (Sun et al., 2019)
Δυνατότητα «παράκαμψης» ²⁸ μηχανισμών test oracles με τη χρήση metamorphic relations (MRs) (Sun et al., 2023)	Περιορισμένος αριθμός MRs δεν καλύπτει πλήρως όλες τις διαστάσεις δοκιμών, ενώ αυξημένος αριθμός μπορεί να αυξήσει εκθετικά τον αριθμό παραγόμενων περιπτώσεων δοκιμών, άρα και το υπολογιστικό κόστος και το χρόνο διεξαγωγής των δοκιμών (ο

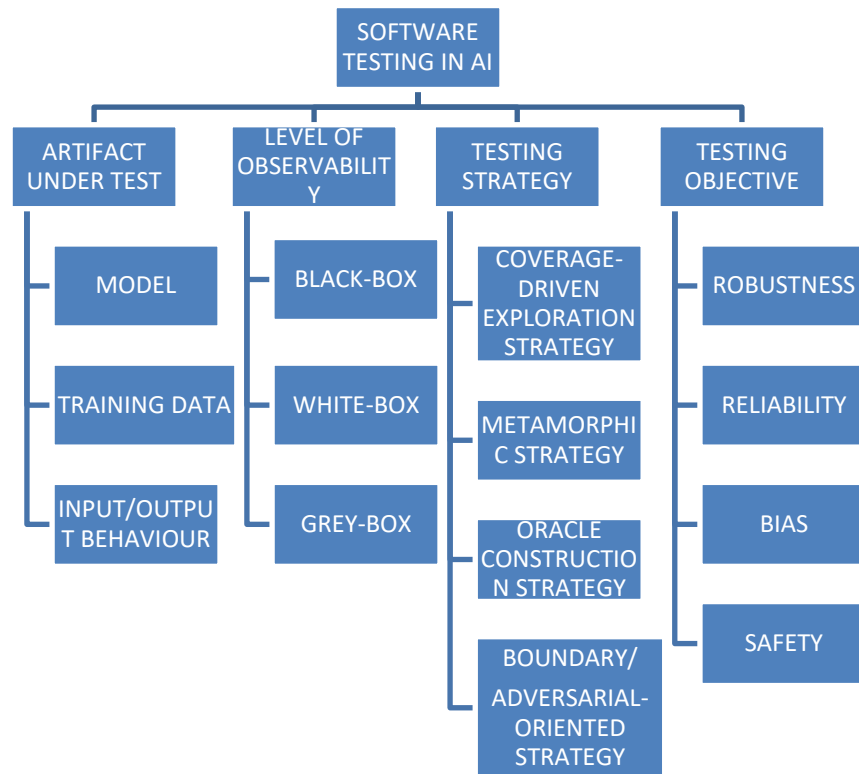
²⁸ Σε συστήματα τεχνητής νοημοσύνης είναι πολλές φορές άγνωστος/μη προφανής ο τρόπος με τον οποίο μια συγκεκριμένη είσοδος παράγει ένα συγκεκριμένο αποτέλεσμα καθώς επίσης και ποιο είναι το αναμενόμενο σωστό αποτέλεσμα. Προκύπτει έτσι ζήτημα με την εφαρμογή παραδοσιακών test oracles που αντιμετωπίζεται με την εφαρμογή metamorphic relations, καθώς η δοκιμή μετατοπίζεται από τη σύγκριση αναμενόμενης και παραγόμενης τιμής στη σχέση μεταξύ εισόδου και εξόδου.

	συσχετισμός με την τεχνητή νοημοσύνη είναι η χρήση MRs όταν δεν είναι εφικτή η χρήση παραδοσιακών test oracles σε AI συστήματα) (Sun et al., 2023, Tu et al., 2021)
--	--

4.2.2 ΕΝΝΟΙΟΛΟΓΙΚΟΣ ΧΑΡΤΗΣ & ΤΑΞΙΝΟΜΗΣΗ ΕΠΙΛΕΓΜΕΝΩΝ ΑΡΘΡΩΝ

Η βιβλιογραφία που αφορά τη συνεισφορά της δοκιμής λογισμικού στην Τεχνητή Νοημοσύνη μπορεί αντίστοιχα να οργανωθεί με βάση διαφορετικές κατευθύνσεις όπου ο αντίστοιχος εννοιολογικός χάρτης εμφανίζεται στο Σχήμα 6. Μια πρώτη δυνατή διάσταση αφορά το αντικείμενο υπό δοκιμή (artifact under test). Ωστόσο, επειδή μελετώνται συστήματα τεχνητής νοημοσύνης, ο όρος “artifact under test” διευρύνεται πέρα από το υπό δοκιμή μοντέλο, σύστημα ή υποσύστημα, στα δεδομένα εκπαίδευσης αυτού, αφού αυτά τελικά αποτελούν λειτουργικό συστατικό του με την ευρύτερη έννοια, καθώς διαμορφώνουν τη συμπεριφορά του. Η συμπεριφορά εισόδου – εξόδου αποτελεί τη λειτουργική εκδήλωση αυτής της μαθησιακής διαδικασίας/εξάρτησης από τα δεδομένα εκπαίδευσης, και είναι ούτως ή άλλως χρήσιμη εννοιολογικά στο πλαίσιο μελέτης συστημάτων όπου το μόνο παρατηρήσιμο στοιχείο είναι η σχέση εισόδου – εξόδου. Μια επιπλέον δυνατή διάσταση ταξινόμησης αφορά τη δυνατότητα παρατήρησης της εσωτερικής δομής του υπό δοκιμή συστήματος (black-box, white-box και gray-box προσεγγίσεις). Επιπλέον, η σχετική βιβλιογραφία δύναται να οργανωθεί με κριτήριο τη στρατηγική δοκιμής που εφαρμόστηκε (έλεγχος δομής - μεγιστοποίηση κάλυψης, metamorphic testing προσεγγίσεις, επίλυση προβλήματος-κατασκευή test oracle, boundary/adversarial analysis³⁰). Τέλος, μια ακόμη δυνατή διάσταση αφορά το χαρακτηριστικό που «στοχεύει» η δοκιμή, όπως για παράδειγμα η ανθεκτικότητα, η αξιοπιστία, η απουσία μεροληψίας (bias) και η ασφάλεια.

³⁰Αναλύεται στην Ενότητα 4.2.3.4.



Σχήμα 6. Εννοιολογικός χάρτης software testing in AI

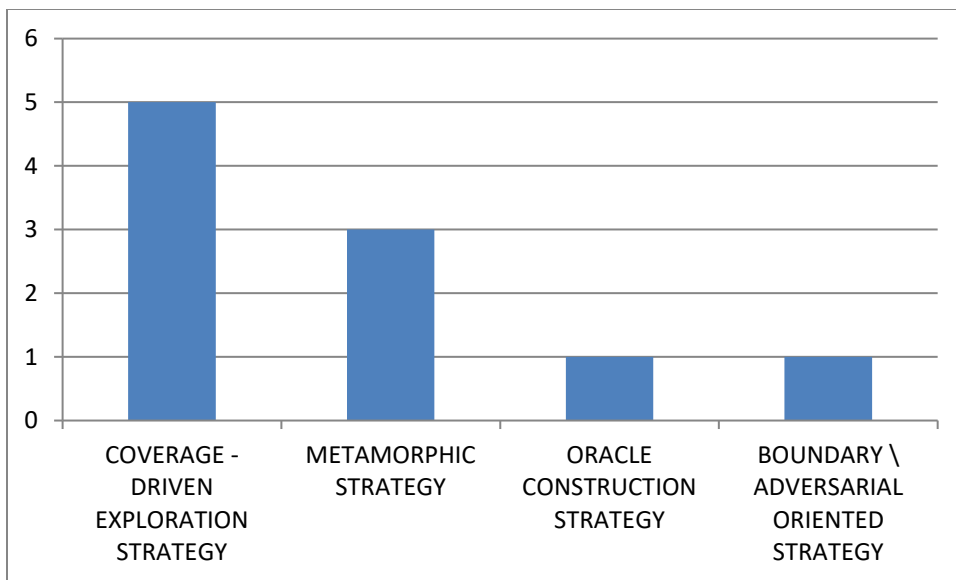
Πίνακας 5. Πίνακας ταξινόμησης προσεγγίσεων software testing in AI με βάση τον αντίστοιχο εννοιολογικό χάρτη

ΑΡΘΡΟ	ARTIFACT UNDER TEST	LEVEL OF OBSERVABILITY	TESTING STRATEGY	TESTING OBJECTIVE
<i>A metamorphic testing approach for assessing Question Answering Systems (Tu et al., 2021)</i>	INPUT/OUTPUT BEHAVIOUR	BLACK - BOX	METAMORPHIC STRATEGY	ROBUSTNESS

<i>A Method and Experiment to evaluate Deep Neural Networks as Test Oracles for Scientific Software (Santiago Junior, 2022)</i>	INPUT/OUTPUT BEHAVIOUR	BLACK - BOX	ORACLE CONSTRUCTION STRATEGY	RELIABILITY
<i>Automated Boundary Identification for Machine Learning Classifiers (Dobslaw & Feldt, 2024)</i>	MODEL	BLACK - BOX	BOUNDARY /ADVERSARIOLO RIENTED STRATEGY	RELIABILITY, ROBUSTNESS
<i>DeepCNP: An Efficient White-Box Testing of Deep Neural Networks by Aligning Critical Neuron Paths(Liu et al.,2025)</i>	MODEL	WHITE-BOX	COVERAGE - DRIVEN EXPLORATION STRATEGY	RELIABILITY
<i>DeepHunter: A Coverage-Guided Fuzz Testing Framework for Deep Neural Networks (Xie et al., 2019)</i>	MODEL	GREY - BOX	COVERAGE - DRIVEN EXPLORATION STRATEGY(υβριδική προσέγγιση: χρησιμοποιεί metamorphic mutations στρατηγική για τη δημιουργία νέων inputs)	RELIABILITY
<i>DeepXplore (Peietal., 2019)</i>	MODEL	WHITE-BOX	COVERAGE - DRIVEN EXPLORATION	RELIABILITY
<i>Improving Early Fault Detection in Machine Learning Systems Using Data Diversity-Driven</i>	INPUT/OUTPUT BEHAVIOUR, TRAINING	BLACK - BOX	METAMORPHIC STRATEGY	RELIABILITY

<i>Metamorphic Relation Prioritization (Srinivasan&Kane wala, 2024)</i>	DATA			
<i>Metamorphic Testing of Relation Extraction Models (Sunetal., 2023)</i>	MODEL	BLACK-BOX	METAMORPHIC STRATEGY	ROBUSTNESS, RELIABILITY
<i>Structural Test Coverage Criteria for Deep Neural Networks (Sunetal., 2019)</i>	MODEL	WHITE-BOX	COVERAGE - DRIVEN EXPLORATION STRATEGY	RELIABILITY, SAFETY
<i>TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing (Odena et al., 2019)</i>	MODEL	GREY - BOX	COVERAGE - DRIVEN EXPLORATION STRATEGY	RELIABILITY, ROBUSTNESS

Παρουσιάζεται στη συνέχεια η κατανομή των άρθρων ανά κατεύθυνση συνεισφοράς (3^η διάσταση - διότι αυτή προσδιορίζει την στρατηγική ή κατεύθυνση δοκιμής ενός ΑΙ συστήματος) στο Σχήμα 7.



Σχήμα 7. Κατανομή άρθρων software testing in AI ανά κύρια κατεύθυνση συνεισφοράς

4.2.3 ΑΝΑΛΥΣΗ

Η ανάλυση της κάθε επιλεγμένης μελέτης πραγματοποιήθηκε με βάση την στρατηγική δοκιμής που ακολουθήθηκε, καθώς η διάσταση αυτή επιτρέπει την ανάδειξη της μεθοδολογικής εξέλιξης του πεδίου.

4.2.3.1 COVERAGE-DRIVEN EXPLORATION STRATEGY

Η στρατηγική αυτή περιλαμβάνει προσεγγίσεις δοκιμής συστημάτων τεχνητής νοημοσύνης που στοχεύουν στη συστηματική εξερεύνηση της εσωτερικής συμπεριφοράς ενός μοντέλου (ML) μέσω της μεγιστοποίησης της κάλυψης χρησιμοποιώντας αντίστοιχες μετρικές. Η βασική ιδέα είναι ότι όσο περισσότερες νευρωνικές μονάδες, διαδρομές ενεργοποίησης ή λειτουργικές καταστάσεις ενεργοποιούνται κατά την εκτέλεση, τόσο αυξάνεται η πιθανότητα αποκάλυψης σφαλμάτων ή απρόβλεπτων συμπεριφορών. Οι προσεγγίσεις αυτές συνήθως παράγουν ή μεταλλάσσουν δεδομένα εισόδου με καθοδηγητικό σήμα την αύξηση της κάλυψης, μετατρέποντας τη διαδικασία δοκιμής σε πρόβλημα αναζήτησης ή βελτιστοποίησης. Στην κατηγορία αυτή εντάσσονται μελέτες στις οποίες η μετρική κάλυψης αποτελεί τον κύριο οδηγό της εξερεύνησης και το βασικό κριτήριο αξιολόγησης της αποτελεσματικότητας της μεθόδου, ανεξάρτητα από το αν χρησιμοποιούνται τεχνικές fuzzing, εξελικτικοί αλγόριθμοι ή άλλες στρατηγικές αναζήτησης.

4.2.3.1.1 DeepXplore: Automated White box Testing of Deep Learning Systems

Το άρθρο *DeepXplore: Automated White box Testing of Deep Learning Systems* (Pei et al., 2019) προβάλλει το DeepXplore, το πρώτο πλαίσιο white box testing για συστήματα βαθιάς μάθησης, το οποίο στοχεύει στον συστηματικό εντοπισμό λανθασμένων συμπεριφορών σε σπάνιες ακραίες περιπτώσεις εισόδου (corner cases). Το DeepXplore εισάγει την έννοια της κάλυψης νευρώνων (neuron coverage) ως νέα μετρική καταλληλότητας για τις δοκιμές βαθιάς μάθησης, χρησιμοποιεί διαφορικές δοκιμές (differential testing)³¹ με χρήση πολλαπλών μοντέλων ως διασταυρούμενες αναφορές για την αποφυγή χειροκίνητου ελέγχου των αποτελεσμάτων και διατυπώνει τη δημιουργία δοκιμών ως ένα κοινό πρόβλημα βελτιστοποίησης που μεγιστοποιεί ταυτόχρονα την κάλυψη νευρώνων και τις διαφορές συμπεριφοράς. Η παραγωγή των test inputs διατυπώνεται ως ενιαίο πρόβλημα βελτιστοποίησης (joint optimization problem), το οποίο στοχεύει ταυτόχρονα στη μεγιστοποίηση της κάλυψης των νευρώνων και του αριθμού διαφοροποιημένων συμπεριφορών.

Χρησιμοποιήθηκαν 5 δημοφιλή datasets (MNIST, ImageNet, Driving, Contagio/VirusTotal και Drebin) και 3 DNN ανά dataset. Τα DNNs που αναφέρονται αποτελούν τα συστήματα υπό δοκιμή (systems under test). Για κάθε σύνολο δεδομένων χρησιμοποιούνται τρία διαφορετικά DNNs, τα οποία έχουν παρόμοια λειτουργικότητα αλλά διαφέρουν ως προς την αρχιτεκτονική και τη διαδικασία εκπαίδευσης. Η χρήση πολλαπλών μοντέλων είναι απαραίτητη στο πλαίσιο της προσέγγισης DeepXplore, καθώς εφαρμόζεται differential testing. Συγκεκριμένα, τα μοντέλα λειτουργούν ως διασταυρούμενα oracles: εάν για το ίδιο input παράγουν διαφορετικές προβλέψεις, τότε εντοπίζεται πιθανή λανθασμένη συμπεριφορά. Οι διαφορές μεταξύ των μοντέλων (π.χ. διαφορετικές αρχιτεκτονικές όπως LeNet, VGG ή ResNet και διαφορετικά εκπαιδευμένα βάρη) είναι σκόπιμες, ώστε να αυξάνεται η πιθανότητα εμφάνισης διαφοροποιημένης συμπεριφοράς και, συνεπώς, ο εντοπισμός σφαλμάτων. Ωστόσο, δεν διευκρινίζεται ρητά ότι τα μοντέλα έχουν εκπαιδευτεί σε διαφορετικά υποσύνολα των δεδομένων, αλλά κυρίως ότι διαφοροποιούνται ως προς την αρχιτεκτονική και την ανεξάρτητη εκπαίδευσή τους.

³¹ Πολλαπλά μοντέλα που υλοποιούν την ίδια λειτουργία εκτελούνται στο ίδιο input και οι προβλέψεις τους συγκρίνονται. Όταν τα μοντέλα παράγουν διαφορετικές εξόδους, προκύπτει ένδειξη πιθανής λανθασμένης συμπεριφοράς, επιτρέποντας έτσι τον εντοπισμό σφαλμάτων χωρίς να απαιτείται εκ των προτέρων γνώση του σωστού αποτελέσματος.

Τα test inputs παράγονται αυτόματα, ώστε να μεγιστοποιούν την κάλυψη νευρώνων και την εμφάνιση διαφορετικής συμπεριφοράς μεταξύ μοντέλων (joint optimization problem). Το σύστημα καθοδηγεί κατάλληλα τις αλλαγές στα test inputs (iterative gradient ascend), αντί να τα τροποποιεί τυχαία, ώστε να επιτυγχάνεται η προαναφερόμενη κοινή βελτιστοποίηση. Με αυτόν τον τρόπο, η δοκιμή λογισμικού καθοδηγεί ενεργά την εξερεύνηση προβληματικών περιοχών του χώρου εισόδων. Ως προς την αξιολόγηση της μεθόδου, οι συγγραφείς αναφέρουν πως το DeepXplore μπορεί να εντοπίζει χιλιάδες λανθασμένες συμπεριφορές σε πραγματικά συστήματα DL, παράγοντας κατά μέσο όρο ένα test input που αποκαλύπτει σφάλμα μέσα σε χρόνο ενός δευτερολέπτου. Επιπλέον, τα παραγόμενα inputs πέτυχαν έως και 34.4% και 33.2% υψηλότερη τιμή neuron coverage σε σύγκριση με τυχαία και adversarial inputs, αντίστοιχα, ενώ η χρήση τους για επανεκπαίδευση των μοντέλων οδήγησε σε βελτίωση της ακρίβειας έως 3%.

Η προσέγγιση του DeepXplore παρουσιάζει σημαντικά πλεονεκτήματα, αλλά και ορισμένες προκλήσεις ως προς την εφαρμογή της. Στα θετικά συγκαταλέγεται η χρήση differential testing για την εξάλειψη manual test oracles και η εισαγωγή της κάλυψης νευρώνων ως νέας μετρικής αξιολόγησης για νευρωνικά δίκτυα. Η μελέτη δείχνει ότι η παραδοσιακή μετρική code coverage δεν αποτελεί κατάλληλο δείκτη για την αξιολόγηση της δοκιμής μοντέλων DNN. Ενδεικτικά, ακόμη και 10 τυχαία επιλεγμένα test inputs οδηγούν σε 100% κάλυψη κώδικα στα εξεταζόμενα μοντέλα, ενώ η αντίστοιχη κάλυψη νευρώνων παραμένει μικρότερη του 34%, γεγονός που υποδεικνύει πως μεγάλο μέρος της εσωτερικής συμπεριφοράς των DNN παραμένει ανεξερεύνητο. Για το λόγο αυτό, η προτεινόμενη προσέγγιση δεν χρησιμοποιεί το neuron coverage μόνο ως μετρική αξιολόγησης αλλά το ενσωματώνει και στη διαδικασία παραγωγής δοκιμών, διατυπώνοντας το πρόβλημα ως διαδικασία βελτιστοποίησης που επιδιώκει ταυτόχρονα τη μεγιστοποίηση της κάλυψης νευρώνων αλλά και την πρόκληση διαφορών στη συμπεριφορά πολλαπλών μοντέλων. Με τον τρόπο αυτόν επιτυγχάνεται συστηματικότερη διερεύνηση του χώρου εισόδων και εντοπισμός λανθασμένων συμπεριφορών σε ακραίες περιπτώσεις (corner cases).

Όμως, υπάρχουν σαφώς και περιορισμοί στο προτεινόμενο πλαίσιο. Το DeepXplore βασίζεται στην παραγωγή test inputs μέσω gradient-based βελτιστοποίησης και στη χρήση προκαθορισμένων domain-specific μετασχηματισμών. Αυτό πρακτικά σημαίνει πως δεν υποστηρίζονται πραγματικά σενάρια, καθώς η εξερεύνηση του χώρου εισόδων περιορίζεται σε

τοπικές τροποποιήσεις γύρω από seed inputs. Οι συγγραφείς αναγνωρίζουν για παράδειγμα πως η μοντελοποίηση σύνθετων και ρεαλιστικών μετασχηματισμών, όπως ο αντίκτυπος της σκιάς από τα γύρω αντικείμενα παραμένει μια ανοικτή πρόκληση. Για πολλούς ρεαλιστικούς μετασχηματισμούς, δεν είναι εύκολο να υπολογιστούν με ακρίβεια οι απαιτούμενες πληροφορίες κλίσης εισόδου (input gradient). Ως αποτέλεσμα, αποτελεί πρόκληση η εύρεση αποτελεσματικών περιπτώσεων δοκιμών που ανιχνεύουν σφάλματα για τέτοιου είδους μετασχηματισμούς, όπως για παράδειγμα η μοντελοποίηση διαφόρων καιρικών συνθηκών. Τονίζεται επίσης πως η τοπική αναζήτηση βάσει κλίσης που εφαρμόζεται δεν εγγυάται την απουσία σφαλμάτων, οπότε κάποιες ελαττωματικές συμπεριφορές μπορεί να μην ανιχνευθούν ακόμη και κατόπιν της σχετικής δοκιμής.

4.2.3.1.2 DeepHunter: A Coverage-Guided Fuzz Testing Framework for Deep Neural Networks

Το άρθρο *DeepHunter: A Coverage-Guided Fuzz Testing Framework for Deep Neural Networks* (Xie et al., 2019) προτείνει το DeepHunter, ένα γενικού σκοπού πλαίσιο coverage-guided fuzz testing για βαθιά νευρωνικά δίκτυα, με στόχο τον συστηματικό εντοπισμό σφαλμάτων και λανθασμένων συμπεριφορών σε DNN-based λογισμικό. Προωθεί μία metamorphic mutations στρατηγική, η οποία δημιουργεί νέα test inputs που διατηρούν τη σημασιολογία τους από ανθρώπινη οπτική (semantics-preserving tests), μειώνοντας την παραγωγή μη έγκυρων εισόδων. Εντάσσεται στην κατηγορία coverage-guided strategy και όχι metamorphic strategy παρόλο που αποτελεί υβριδική προσέγγιση, με τη λογική ότι η συνολική διαδικασία δοκιμής καθοδηγείται κυρίως από ανατροφοδότηση κάλυψης. Επιπλέον, το παρόν άρθρο μπορεί να ομαδοποιηθεί με τα άρθρα DeepXplore (Pei et al., 2019) και Tensorfuzz (Odena et al., 2019) αποτελώντας ορόσημο για τη μετάβαση σε συστηματικές δοκιμές DNN. Το πλαίσιο αυτό αξιοποιεί πολλαπλά testing criteria για DNNs (NeuronCoverage, KMNC, NBC, SNAC, TKNC) (Ma et al., 2018) ως μηχανισμό ανατροφοδότησης για την καθοδήγηση της παραγωγής inputs. Πραγματοποιήθηκαν συγκριτικά πειράματα με άλλες σύγχρονες προσεγγίσεις δοκιμής αλλά και τυχαία παραγωγή δεδομένων δοκιμών, χρησιμοποιώντας ως βασικές μετρικές την ποσοστιαία κάλυψη και το μέσο αριθμό μοναδικών σφαλμάτων που εμφανίζονται. Τα αποτελέσματα δείχνουν πως το DeepHunter επιτυγχάνει υψηλότερα επίπεδα κάλυψης και εντοπίζει περισσότερες λανθασμένες συμπεριφορές. Για παράδειγμα, στο μοντέλο LeNET-5 επιτυγχάνει κάλυψη 69,14% με κριτήριο KMNC και εντοπίζει έως 1207 σφάλματα, υπερβαίνοντας σημαντικά τα αποτελέσματα των ανταγωνιστικών εργαλείων.

Στο DeepHunter, η δοκιμή λογισμικού υλοποιείται ως διαδικασία coverage-guided fuzz testing, η οποία καθοδηγεί συστηματικά την παραγωγή test inputs για deep learning συστήματα. Το πλαίσιο ξεκινά από αρχικά seed inputs, τα οποία χρησιμοποιούνται ως βάση για περαιτέρω δοκιμές. Παράγονται νέα test inputs μέσω metamorphic mutations, που διατηρούν τη σημασιολογία των δεδομένων (π.χ. οπτικά έγκυρες εικόνες). Η επιλογή των επόμενων inputs καθοδηγείται από κριτήρια κάλυψης για DNNs (NeuronCoverage, KMNC, NBC, SNAC, TKNC). Έπειτα, η ανατροφοδότηση από τα coverage metrics χρησιμοποιείται για να αποφασιστεί ποια inputs αξίζει να εξερευνηθούν περαιτέρω. Η διαδικασία fuzzing επαναλαμβάνεται, με στόχο τη μέγιστη κάλυψη της εσωτερικής συμπεριφοράς του μοντέλου και τον εντοπισμό σφαλμάτων. Το DeepHunter λειτουργεί χωρίς χειροκίνητα test oracles, βασιζόμενο στα κριτήρια κάλυψης και στις παρατηρούμενες αποκλίσεις συμπεριφοράς.

Η προσέγγιση του DeepHunter παρουσιάζει σημαντικά πλεονεκτήματα για τον έλεγχο deep learning συστημάτων, αλλά και πρακτικά μειονεκτήματα. Αρχικά, επιτρέπει συστηματικό coverage-guided fuzz testing για DNNs, χρησιμοποιώντας εναλλακτικά, διαφορετικά coverage criteria (NC, KMNC, NBC, SNAC, TKNC) ως μηχανισμό ανατροφοδότησης για την καθοδήγηση της εξερεύνησης του χώρου εισόδων, προσφέροντας, έτσι, πιο πλούσια αξιολόγηση της εσωτερικής συμπεριφοράς. Γενικά, αποδεικνύεται πιο αποτελεσματικό από υπάρχοντα εργαλεία/προσεγγίσεις ως προς κάλυψη και εύρεση σφαλμάτων. Συγκεκριμένα, μια διαμόρφωση του DeepHunter που χρησιμοποιεί πιθανοκρατική στρατηγική επιλογής seeds (DH+Prob) επιτυγχάνει βελτίωση της τάξης του 81.8% για το μοντέλο LeNet-5, 40.3% για το ResNet-20 και 16.6% για το VGG16 σε σύγκριση με τη δεύτερη καλύτερη μέθοδο. Η δεύτερη καλύτερη μέθοδος δεν αντιστοιχεί σε συγκεκριμένη μέθοδο, αλλά στη δεύτερη καλύτερη επίδοση μεταξύ των συγκρινόμενων μεθόδων ανάλογα με το εκάστοτε κριτήριο (π.χ. DeepTest, TensorFuzz ή παραλλαγές του DeepHunter).

Ωστόσο, η αποτελεσματικότητα της προσέγγισης επηρεάζεται από το coverage criterion που χρησιμοποιείται ως μηχανισμός ανατροφοδότησης στο fuzzing, το οποίο επιλέγεται κατά τη διαμόρφωση της εκτέλεσης και όχι μέσω αυτοματοποιημένης διαδικασίας. Ακόμη, η διαδικασία fuzzing με διαφορετικά coverage metrics μπορεί να οδηγήσει σε αυξημένο υπολογιστικό κόστος, αφού απαιτεί την επαναλαμβανόμενη αξιολόγηση των μοντέλων για μεγάλο αριθμό

παραγόμενων εισόδων και τον υπολογισμό της αντίστοιχης μετρικής κάλυψης σε κάθε κύκλο fuzzing.

4.2.3.1.3 DeepCNP: An efficient white-box testing of deep neural networks by aligning critical neuron paths

Το άρθρο *DeepCNP: An Efficient White-Box Testing of Deep Neural Networks by Aligning Critical Neuron Paths* (Liu et al., 2025) εξετάζει την αυξανόμενη ανάγκη για συστηματικές και αξιόπιστες δοκιμές DNN, ιδίως σε κρίσιμους τομείς όπου η λήψη λάθους απόφασης από το μοντέλο μπορεί να οδηγήσει σε σοβαρές συνέπειες. Οι συγγραφείς παρατηρούν ότι οι υπάρχουσες τεχνικές δοκιμών για νευρωνικά δίκτυα, ειδικά οι προσεγγίσεις fuzzing με καθοδήγηση κάλυψης, συχνά δυσκολεύονται να εξερευνήσουν επαρκώς την εσωτερική λογική λήψης αποφάσεων σύνθετων μοντέλων. Για να ξεπεράσουν αυτόν τον περιορισμό, προτείνουν το DeepCNP, ένα πλαίσιο δοκιμών white-box που μοντελοποιεί τη συμπεριφορά λήψης αποφάσεων ενός νευρωνικού δικτύου χρησιμοποιώντας κρίσιμες διαδρομές νευρώνων που προέρχονται από κατανομές εξόδου νευρώνων σε δεδομένα εκπαίδευσης. Ευθυγραμμίζοντας τη δημιουργία δοκιμών με αυτές τις διαδρομές, το πλαίσιο στοχεύει να ελέγξει την εσωτερική λογική του μοντέλου στο σύνολό της και να εκθέσει πιο αποτελεσματικά τις εσφαλμένες συμπεριφορές.

Το πλαίσιο ξεκινά με την ανάλυση των αρχικών επισημασμένων δεδομένων εκπαίδευσης του υπό δοκιμή μοντέλου, προκειμένου να χαρτογραφήσει την εσωτερική λογική λήψης αποφάσεών του. Για κάθε κατηγορία εξόδου, υπολογίζονται οι κατανομές ενεργοποίησης των νευρώνων και χρησιμοποιούνται για τον προσδιορισμό των κρίσιμων νευρώνων σε κάθε στρώμα, οι οποίοι στη συνέχεια συνδέονται για να σχηματίσουν κρίσιμες διαδρομές νευρώνων που αντιπροσωπεύουν κυρίαρχες συμπεριφορές λήψης αποφάσεων. Έπειτα, επιλέγεται ένα σύνολο σωστά επισημασμένων/ταξινομημένων δεδομένων ως σημεία αφετηρίας (“σπόροι”) για τη δοκιμή ασαφούς λογικής. Στη συνέχεια, η δημιουργία δοκιμών προχωρά μέσω μιας διαδικασίας μετάλλαξης καθοδηγούμενης από ευθυγράμμιση, στην οποία κάθε σπόρος διαταράσσεται επαναληπτικά, έτσι ώστε οι εσωτερικές ενεργοποιήσεις του μοντέλου να ευθυγραμμίζονται όλο και περισσότερο με επιλεγμένες κρίσιμες διαδρομές νευρώνων. Μια δυναμική στρατηγική επιλογής σπόρων δίνει περαιτέρω προτεραιότητα σε εισόδους που είναι πιο πιθανό να εξερευνήσουν ανεξερεύνητες ή ασθενώς δοκιμασμένες διαδρομές λήψης

αποφάσεων. Το πλαίσιο παρακολουθεί συνεχώς την κάλυψη της διαδρομής και τις παραγόμενες συμπεριφορές, παράγοντας κατάλληλα νέες περιπτώσεις δοκιμών.

Τα πειραματικά αποτελέσματα δείχνουν ότι το DeepCNP βελτιώνει σημαντικά τόσο την αποδοτικότητα όσο και την αποτελεσματικότητα της ανίχνευσης ελαττωμάτων σε σύγκριση με τεχνικές δοκιμών white-box και fuzzing με βάση την κάλυψη. Η μελέτη αξιολογήθηκε πειραματικά σε 4 σύνολα δεδομένων (MNIST³², Fashion-MNIST³³, CIFAR-10³⁴, Contagio³⁵/VirusTotal³⁶) και διαφορετικά μοντέλα DNN. Συνολικά χρησιμοποιήθηκαν 8 διαφορετικά μοντέλα DNN, με δύο μοντέλα ανά σύνολο δεδομένων. Πιο συγκεκριμένα, για τα MNIST και Fashion-MNIST χρησιμοποιήθηκαν παραλλαγές του LeNet και ένα ResNet-20, για το CIFAR-10 τα ResNet-20 και VGG-16, ενώ για το σύνολο δεδομένων PDF χρησιμοποιήθηκαν δύο εξειδικευμένα μοντέλα (PDF_D1 και PDF_D2). Τα πειραματικά αποτελέσματα αποδεικνύουν ότι το DeepCNP προσφέρει σημαντικές βελτιώσεις σε σχέση με τις υπάρχουσες τεχνικές. Σε όλα τα αξιολογηθέντα benchmarks, το πλαίσιο ανακαλύπτει σταθερά σημαντικά περισσότερα ελαττώματα μοντέλων, εντοπίζοντας μεταξύ 5,46% και 96% περισσότερες ετικέτες σφαλμάτων σε σύγκριση με τις πιο σύγχρονες μεθόδους, ενώ παράλληλα δημιουργεί περιπτώσεις δοκιμών με σημαντικά υψηλότερη απόδοση (την μετρούν με βάση τον αριθμό των cases που παράγονται ανά μονάδα χρόνου). Συγκεκριμένα, το DeepCNP βελτιώνει την απόδοση δημιουργίας περιπτώσεων δοκιμών έως και 331,82%, με συνολική αύξηση της απόδοσης που κυμαίνεται από 2,05 έως 31,34 φορές σε σύγκριση με τις ανταγωνιστικές προσεγγίσεις. Κατάφερε επίσης να εντοπίσει περισσότερα σφάλματα ταξινόμησης κατά ποσοστό έως 96%.

Συνεπώς, με τη μοντελοποίηση και την ευθυγράμμιση κρίσιμων νευρωνικών διαδρομών, το πλαίσιο επιτυγχάνει μια βαθύτερη και πιο συστηματική εξερεύνηση της εσωτερικής λογικής λήψης αποφάσεων του μοντέλου, επιτρέποντας την ολοκληρωμένη δοκιμή διαφορετικών συμπεριφορών λήψης αποφάσεων αντί της απλής αύξησης των επιφανειακών μετρήσεων κάλυψης. Αυτά τα πλεονεκτήματα επικυρώνονται μέσω εκτεταμένων πειραμάτων σε πολλαπλά σύνολα δεδομένων και αρχιτεκτονικές δικτύων, συμπεριλαμβανομένων τόσο εργασιών

³² Διαθέσιμο στο <https://www.tensorflow.org/datasets/catalog/mnist>

³³ Διαθέσιμο στο <https://www.kaggle.com/datasets/zalando-research/fashionmnist>

³⁴ Διαθέσιμο στο <https://www.cs.toronto.edu/~kriz/cifar.html>

³⁵ Πρόκειται για αποθετήριο δειγμάτων κακόβουλου λογισμικού, διαθέσιμο στο <https://contagiodump.blogspot.com/>

³⁶ Διαθέσιμο στο <https://www.virustotal.com/>

ταξινόμησης εικόνων όσο και ανίχνευσης κακόβουλου λογισμικού, υπογραμμίζοντας την ανθεκτικότητα, την πιθανή δυνατότητα επέκτασης και την πρακτική συνάφεια της προτεινόμενης μεθόδου για τη βελτίωση της αξιοπιστίας των συστημάτων βαθιάς μάθησης.

Το DeepCNP παρουσιάζει αρκετούς περιορισμούς που αναγνωρίζονται ρητά στο άρθρο. Πρώτον, οι συγγραφείς σημειώνουν ότι η κατασκευή κρίσιμων νευρωνικών διαδρομών εισάγει υψηλό υπολογιστικό κόστος, ιδιαίτερα για εργασίες ταξινόμησης μεγάλης κλίμακας με πολλές κατηγορίες εξόδου, όπως το ImageNet, καθιστώντας την επεκτασιμότητα μια πρόκληση για πολύ μεγάλα σύνολα δεδομένων. Επιπλέον, αναφέρουν ότι τόσο η στρατηγική δυναμικής επιλογής σπόρων (seeds) όσο και η στρατηγική ευθυγράμμισης κρίσιμων νευρωνικών διαδρομών είναι επί του παρόντος εφαρμόσιμες μόνο σε μοντέλα ταξινόμησης, γεγονός που περιορίζει τη χρήση του πλαισίου για άλλους τύπους εργασιών βαθιάς μάθησης. Επιπλέον, ενώ η πειραματική αξιολόγηση είναι ευρεία σε βάθος (καλύπτοντας πολλαπλά σύνολα δεδομένων, αρχιτεκτονικές μοντέλων και βασικές γραμμές σύγκρισης) το πεδίο εφαρμογής αυτής της αξιολόγησης παραμένει περιορισμένο, καθώς περιορίζεται στην ταξινόμηση εικόνων και την ανίχνευση κακόβουλου λογισμικού PDF. Ως αποτέλεσμα, η δυνατότητα γενίκευσης του DeepCNP σε άλλους τομείς βαθιάς μάθησης, όπως η επεξεργασία φυσικής γλώσσας ή η αναγνώριση ομιλίας, παραμένει ένα ανοιχτό ζήτημα για μελλοντική έρευνα.

4.2.3.1.4 Structural Test Coverage Criteria for Deep Neural Networks

Το άρθρο *Structural Test Coverage Criteria for Deep Neural Networks* (Sun et al., 2019) προτείνει νέα δομικά κριτήρια κάλυψης (structural test coverage criteria), ειδικά σχεδιασμένα για deep neural networks, προκειμένου να καταστεί εφικτός ο συστηματικός έλεγχός τους σε κρίσιμες εφαρμογές. Επισημαίνεται στην αρχή ότι τα παραδοσιακά κριτήρια κάλυψης λογισμικού (π.χ. statement, branch, Modified Condition/Decision Coverage, MC/DC³⁷) δεν εφαρμόζονται άμεσα σε DNNs. Επομένως, η εργασία αντλεί έμπνευση από το MC/DC coverage και το προσαρμόζει στα δομικά χαρακτηριστικά των νευρωνικών δικτύων. Προτείνεται, έτσι, μία οικογένεια τεσσάρων νέων κριτηρίων κάλυψης, τα οποία λαμβάνουν υπόψη τις ενεργοποιήσεις νευρώνων, τις αιτιακές σχέσεις μεταξύ γειτονικών επιπέδων και τις μεταβολές πρόσημου ή τιμών ενεργοποίησης. Στόχος είναι τα κριτήρια αυτά να οδηγούν στη δημιουργία test inputs, να εντοπίζουν ανεπιθύμητες ή επικίνδυνες συμπεριφορές (π.χ. adversarial examples)

³⁷ Αναπτύχθηκε από τη NASA και χρησιμοποιείται ως οδηγός για την ανάπτυξη λογισμικού αεροναυτικής ηλεκτρονικής, με σκοπό να διασφαλιστεί η επαρκής δοκιμή εφαρμογών με την υψηλότερη βαθμίδα κρισιμότητας.

και να επιτυγχάνουν ισορροπία μεταξύ αποτελεσματικότητας ελέγχου και υπολογιστικού κόστους.

Τα κριτήρια αυτά αποτυπώνουν διαφορετικούς τύπους αιτιακής επίδρασης μεταξύ νευρώνων. Βασίζονται στη σύγκριση τιμών ενεργοποίησης μεταξύ ζευγών test inputs και χρησιμοποιούν προκαθορισμένες κρίσιμες τιμές για να διακρίνουν αλλαγές στη συμπεριφορά των νευρώνων. Από αυτά μπορεί να αξιολογηθεί αν ένα σύνολο test inputs εξερευνά επαρκώς τις εσωτερικές λογικές διαδρομές του δικτύου. Επομένως, μπορούν να χρησιμοποιηθούν για την καθοδήγηση της δημιουργίας test inputs, με στόχο την αύξηση της δομικής κάλυψης του DNN. Για να χρησιμοποιήσουν τα κριτήρια στην πράξη, οι συγγραφείς περιγράφουν μια διαδικασία παραγωγής περιπτώσεων δοκιμών με τη χρήση αλγορίθμου που αναζητά ζεύγη εισόδων που ικανοποιούν τις απαιτούμενες συνθήκες ενεργοποίησης, επιτρέποντας συστηματική αξιολόγηση της κάλυψης σύμφωνα με τα προτεινόμενα κριτήρια. Ο αλγόριθμος θα μπορούσε να χαρακτηριστεί ως αλγόριθμος βελτιστοποίησης καθοδηγούμενος από περιορισμούς.

Ως προς την αξιολόγηση της μελέτης, τα πειράματα δείχνουν ότι τα προτεινόμενα κριτήρια επιτυγχάνουν υψηλά ποσοστά κάλυψης (άνω του 86%) ενώ παράλληλα εντοπίζουν σημαντικό ποσοστό adversarial παραδειγμάτων. Για παράδειγμα, στο CIFAR-10 η κάλυψη ξεπερνά το 90% και τα adversarial παραδείγματα το 84.36% κατά μέσο όρο.

Τα προτεινόμενα δομικά κριτήρια κάλυψης παρουσιάζουν σημαντικά πλεονεκτήματα για τον έλεγχο DNN αλλά και ορισμένους περιορισμούς ως προς την εφαρμογή τους. Τα προτεινόμενα δομικά κριτήρια κάλυψης είναι ειδικά σχεδιασμένα για DNNs, καλύπτοντας το κενό των παραδοσιακών μετρικών κάλυψης λογισμικού. Επιπλέον, βασίζονται σε αιτιακές σχέσεις μεταξύ νευρώνων, προσφέροντας βαθύτερη εικόνα της εσωτερικής συμπεριφοράς. Είναι εμπνευσμένα από καθιερωμένες πρακτικές testing (MC/DC), άρα έχουν θεωρητική τεκμηρίωση. Μπορούν, μάλιστα, να χρησιμοποιηθούν για καθοδήγηση της δημιουργίας test inputs και μπορούν πιο εύκολα να εντοπίσουν adversarial ή επικίνδυνες συμπεριφορές.

Στον αντίποδα, η εφαρμογή τους είναι υπολογιστικά απαιτητική, ειδικά σε μεγάλα δίκτυα. Επιπροσθέτως, τα κριτήρια κάλυψης δεν εγγυώνται λειτουργική ορθότητα αλλά κυρίως μετρούν την εξερεύνηση της δομής.

4.2.3.1.5 TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing

Το άρθρο *TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing* (Odena et al., 2019) προτείνει το TensorFuzz ως ένα γενικού σκοπού πλαίσιο coverage-guided fuzzing για τον εντοπισμό σφαλμάτων σε νευρωνικά δίκτυα, καθώς εμφανίζονται περιορισμοί στις παραδοσιακές τεχνικές testing και coverage metrics για DNNs.

Η μεθοδολογία του TensorFuzz βασίζεται στη συνδυαστική χρήση coverage-guided fuzzing και property-based testing³⁸(PBT). Ο χρήστης υποδεικνύει συγκεκριμένη ιδιότητα που η συνάρτηση θα πρέπει να ικανοποιεί και το σύστημα PBT αυτόματα παράγει περίπτωση δοκιμής που προσπαθεί να παραβιάσει τη συγκεκριμένη ιδιότητα (Odena et al., 2019) για τον συστηματικό έλεγχο νευρωνικών δικτύων. Ξεκινά από έγκυρα seed inputs (π.χ. εικόνες, διανύσματα), παράγει νέα inputs μέσω μεταλλάξεων (mutations), χρησιμοποιεί ενεργοποιήσεις του δικτύου ως βάση για τον ορισμό κάλυψης, ελέγχει τη “νέα κάλυψη” μέσω αναζήτησης approximate nearest neighbor (ANN), διατηρεί το παραγόμενο inputμόνο αν οδηγεί σε νέα εσωτερική κατάσταση του μοντέλου, ορίζει τις objective functions (ιδιότητες που δεν πρέπει να παραβιάζονται) και παράγει test cases που αποτελούν παραδείγματα παραβίασης των ιδιοτήτων.

Η αξιολόγηση της προσέγγισης έγινε με πειραματική εφαρμογή του TensorFuzz σε διάφορα σενάρια testing DNN (numerical errors, quantization errors, debugging Tensorflow code). Τα αποτελέσματα δείχνουν πως το TensorFuzz μπορεί να εντοπίσει σφάλματα αποτελεσματικά, βρίσκοντας για παράδειγμα inputs που προκαλούν NaN τιμές σε όλα τα πειράματα, ενώ μια προσέγγιση τυχαίας αναζήτησης με 10 εκατομμύρια μεταλλάξεις δεν εντόπισε κανένα τέτοιο σφάλμα. Εντόπισε επίσης διαφωνίες σε μοντέλα 32-bit και 16-bit σε περίπου 70% των περιπτώσεων.

Η προσέγγιση του TensorFuzz παρουσιάζει ισχυρά πρακτικά θετικά στοιχεία αλλά και συγκεκριμένους περιορισμούς. Στα θετικά συγκαταλέγονται η δυνατότητα εντοπισμού αριθμητικών σφαλμάτων σε νευρωνικά δίκτυα (τύπου not a number value, NaN) τα οποία αναφέρεται από τους συγγραφείς πως είναι πρόκληση να εντοπισθούν και να αντιμετωπισθούν στα νευρωνικά δίκτυα. Η προτεινόμενη προσέγγιση αναδεικνύει διαφορές μεταξύ διαφορετικών

³⁸ Είναι μια πρακτική δοκιμή λογισμικού όπου η ορθότητα ενός συστήματος ορίζεται μέσω ενός συνόλου από ιδιότητες, δηλαδή λογικούς ισχυρισμούς για την αναμενόμενη συμπεριφορά του συστήματος. Η εστίαση είναι ως προς το τι θα πρέπει να ισχύει για το σύστημα αντί για το πως θα ελεγχθεί. Ένα βασικό χαρακτηριστικό της πρακτικής αυτής είναι η αυτόματη παραγωγή των εισόδων (δοκιμής).

εκδοχών των μοντέλων που προκύπτουν λόγω quantization³⁹. Επιπλέον, μπορεί να βοηθήσει στην παραγωγή μετασχηματισμένου κώδικα με παράλληλη διατήρηση της σημασιολογίας. Οι συγγραφείς αναφέρουν πως αυτό επετεύχθη συγκρίνοντας την αρχική και την τροποποιημένη υλοποίηση του μηχανισμού τυχαίας αναστροφής εικόνων. Αναφέρεται επίσης από τους συγγραφείς πως η προσέγγιση μπορεί να ανιχνεύσει πραγματικά σφάλματα σε υλοποιήσεις πραγματικών μοντέλων. Υποστηρίζουν ήταν δυνατό με τη χρήση του Tensorfuzz ένα γνωστό ζήτημα σε ένα ευρέως χρησιμοποιούμενο τμήμα κώδικα Tensorflow.

Παρά τα σημαντικά πλεονεκτήματα της προσέγγισης, η μέθοδος δεν παρέχει εγγυήσεις πληρότητας ως προς την ορθότητα του μοντέλου, αλλά λειτουργεί στο πλαίσιο εξερεύνησης, αυξάνοντας την πιθανότητα αποκάλυψης σφαλμάτων. Επιπλέον, η αποτελεσματικότητά της εξαρτάται άμεσα από τον ορισμό του objective, δηλαδή από το είδος των συμπεριφορών που επιδιώκεται να εντοπιστούν. Για παράδειγμα, αν η objective function ελέγχει την εμφάνιση NaN, το σύστημα θα αναζητήσει inputs που παραβιάζουν αυτήν την ιδιότητα, με αποτέλεσμα η μέθοδος να παρουσιάζει διαφορετική αποτελεσματικότητα από αυτήν που θα παρουσίαζε αν η ιδιότητα ήταν διαφορετική, π.χ. ανίχνευση μεταξύ διαφορών μοντέλου και quantized εκδοχής του. Τέλος, οι ίδιοι οι συγγραφείς αναγνωρίζουν ότι η διαδικασία μπορεί να οδηγήσει σε false negatives, καθώς η μη εύρεση διαφορών δεν συνεπάγεται κατ' ανάγκη ισοδυναμία συμπεριφοράς.

4.2.3.2 METAMORPHIC STRATEGY

Αφορά προσεγγίσεις δοκιμής όπου η ορθότητα της συμπεριφοράς ενός συστήματος δεν αξιολογείται μέσω ενός παραδοσιακού oracle, αλλά μέσω σχέσεων που πρέπει να διατηρούνται μεταξύ πολλαπλών εκτελέσεων του συστήματος. Οι σχέσεις αυτές, γνωστές ως μεταμορφικές σχέσεις (metamorphic relations), ορίζουν πώς θα πρέπει να μεταβάλλεται ή να μένει αμετάβλητη η έξοδος, όταν εφαρμόζονται συγκεκριμένοι μετασχηματισμοί στα δεδομένα εισόδου. Η συγκεκριμένη προσέγγιση είναι ιδιαίτερα χρήσιμη όταν δεν είναι εφικτό, ή είναι ιδιαίτερα δύσκολο, να καθοριστεί εκ των προτέρων το σωστό αποτέλεσμα. Στην κατηγορία αυτή εντάσσονται μελέτες όπου ο βασικός μηχανισμός ανίχνευσης σφαλμάτων βασίζεται στον

³⁹ Δημοφιλής προσέγγιση για τη μείωση του υπολογιστικού κόστους ή του μεγέθους νευρωνικών δικτύων μέσω συμπίεσης των μοντέλων όπου αριθμητικές παράμετροι υψηλής ακρίβειας μετατρέπονται σε αριθμητικές παραμέτρους χαμηλότερης ακρίβειας διατηρώντας παράλληλα τη σωστή λειτουργική συμπεριφορά του μοντέλου.

ορισμό, την επαλήθευση ή την αξιοποίηση μεταμορφικών σχέσεων ως βασικό κριτήριο εγκυρότητας των αποτελεσμάτων.

η

4.2.3.2.1 Improving Early Fault Detection in Machine Learning Systems Using Data Diversity-Driven Metamorphic Relation Prioritization

Το άρθρο *Improving Early Fault Detection in Machine Learning Systems Using Data Diversity-Driven Metamorphic Relation Prioritization* (Srinivasan & Kanewala, 2024) πραγματεύεται τη βελτίωση της έγκαιρης ανίχνευσης σφαλμάτων σε συστήματα μηχανικής μάθησης μέσω προτεραιοποίησης μεταμορφικών σχέσεων (metamorphic relations - MRs). Τα ML/DL συστήματα αντιμετωπίζουν έντονα το test oracle ⁴⁰. ενδεδειγμένη προτεινόμενη λύση σε αυτές τις περιπτώσεις είναι το metamorphic testing (μεταμορφική δοκιμή). MRs παρουσιάζουν διαφορετική αποτελεσματικότητα ανίχνευσης σφαλμάτων, η εκτέλεση όλων των MRs είναι ασύμφορη, καθώς μπορεί να αυξήσει σημαντικά το χρόνο εκτέλεσης και τους υπολογιστικούς πόρους που απαιτούνται για την εκτέλεση των δοκιμών. Με βάση το πλαίσιο αυτό, στόχος του άρθρου είναι ο πρώιμος εντοπισμός σφαλμάτων με όσο το δυνατόν λιγότερες MRs.

Η κύρια συνεισφορά του άρθρου είναι μια αυτοματοποιημένη μέθοδος προτεραιοποίησης metamorphic relations βασισμένη στη διαφοροποίηση των δεδομένων. Αρχικά, εισάγονται πέντε data diversity-based μετρικές (metrics) για MR prioritization. Οι μετρικές καλύπτουν ποικιλία όψεων διαφοροποίησης (rule-based, anomaly-based, clustering-based, data distribution-based, feature-based). προτεραιοποίηση βασίζεται στη διαφορά μεταξύ source και follow-up test cases. Ειδικότερα, ως αρχικό σύνολο δεδομένων εκπαίδευσης χρησιμοποιούνται source test cases, και παράγονται follow-up test cases μέσω μετασχηματισμού του αρχικού συνόλου δεδομένων ανάλογα με τον ορισμό της εκάστοτε MR. Κάθε μετρική από τις παραπάνω εφαρμόζεται ξεχωριστά και υπολογίζει έναν βαθμό διαφοροποίησης (MR diversity value) μεταξύ source και follow-up test case για κάθε MR. Αυτό επιτρέπει την προτεραιοποίηση των MRs, με σειρά εκτέλεσης πρώτα εκείνων με μεγαλύτερο βαθμό διαφοροποίησης μεταξύ source και follow up test case, καθώς κατά τους συγγραφείς αυτό υποδεικνύει μεγαλύτερη δυνατότητα ανίχνευσης ελαττωμάτων.

⁴⁰ Δυσκολία καθορισμού της σωστής εξόδου για κάθε είσοδο σε συστήματα μηχανικής μάθησης

Η αποτελεσματικότητα της προτεινόμενης μεθόδου αξιολογείται πειραματικά σε ML και DL συστήματα (kNN (IBk), LinearRegression, NaiveBayes, CNN (MNIST&Fashion-MNIST)) και συγκρίνεται με καθιερωμένες προσεγγίσεις (random MR ordering, fault-based prioritization, neuron activation coverage-based ordering). Τα αποτελέσματα φανερώνουν σημαντικά υψηλότερη απόδοση στον εντοπισμό σφαλμάτων σε σύγκριση με μεθόδους προτεραιοποίησης που βασίζονται στη μετρική κάλυψης του neuron coverage. Αναφέρουν πως, για CNN που χρησιμοποιούν τα σύνολα δεδομένων MNIST⁴¹ και Fashion⁴², η προτεινόμενη προσέγγιση επιτυγχάνει υψηλότερο Μέσο Ποσοστό Ελαττωμάτων που ανιχνεύτηκαν (Average Percentage of Faults Detected, APFD) μεταξύ 0,35 και 0,88 συγκριτικά με προσεγγίσεις που βασίζονται στη μετρική κάλυψης ενεργοποίησης νευρώνων (neuron activation coverage). Επιπλέον, αναφέρεται εκφρασμένη σε ποσοστό η βελτίωση στην αποτελεσματικότητα ανίχνευσης ελαττωμάτων κατά 5% με 550% σε σύγκριση με προσεγγίσεις προσανατολισμένες στην κάλυψη ενεργοποίησης νευρώνων, και μείωση του χρόνου ανίχνευσης ελαττωμάτων σε ποσοστό μέχρι και 62% σε σχέση με προσεγγίσεις τυχαίας εκτέλεσης Metamorphic Relations (MR).

Αναφορικά με τους αντίστοιχους περιορισμούς, η προτεινόμενη προσέγγιση εφαρμόζεται αποκλειστικά σε ταξινομητές επιβλεπόμενης μηχανικής μάθησης. Μάλιστα, οι συγγραφείς αναγνωρίζουν ρητά πως δεν ενδείκνυται για περιπτώσεις μη επιβλεπόμενης μηχανικής μάθησης, καθώς η μέθοδος βασίζεται σε εποπτευόμενα μοντέλα και μεταμορφικές σχέσεις που προϋποθέτουν την ύπαρξη ετικετών, κάτι που δεν ισχύει στα μη επιβλεπόμενα μοντέλα. Η εφαρμογή της μεθόδου έγινε με τη χρήση αλγορίθμων της βιβλιοθήκης WekaML, οι οποίοι παρά τη δημοφιλία και την ευρεία χρήση τους, είναι αμφίβολο αν μπορούν να χρησιμοποιηθούν σε υλοποιήσεις εμπορικών εφαρμογών μηχανικής μάθησης, όπως για παράδειγμα οχημάτων αυτόματης οδήγησης. Ο λόγος έχει να κάνει με την πολυπλοκότητα υλοποίησης βιομηχανικών ΑΙ συστημάτων σε σχέση με την υλοποίηση μεμονωμένων αλγορίθμων σε ακαδημαϊκό περιβάλλον και όχι με ενδεχόμενη εξειδίκευση της προσέγγισης σε υλοποιήσεις με τη χρήση συγκεκριμένης γλώσσας προγραμματισμού. Παρόλο που χρησιμοποιήθηκαν σύνολα δεδομένων μεγάλης κλίμακας, οι συγγραφείς αναφέρουν πως το μέγεθός τους ενδέχεται να μην επαρκεί για την πλήρη ανάδειξη της αποτελεσματικότητας της μεθόδου. Αντίστοιχα, ο αριθμός των mutants

⁴¹ Διαθέσιμο στο <https://www.tensorflow.org/datasets/catalog/mnist>

⁴² Διαθέσιμο στο <https://www.kaggle.com/datasets/zalando-research/fashionmnist>

που χρησιμοποιήθηκαν θεωρείται σχετικά περιορισμένος⁴³. Οι συγγραφείς τέλος επισημαίνουν, πως η αύξηση του αριθμού MR οδηγεί σε εκθετική αύξηση των παραγόμενων test cases, οπότε η εκτέλεση των τελευταίων μπορεί να αυξήσει σημαντικά το χρόνο και τους πόρους που απαιτούνται για το metamorphic testing.

4.2.3.2.2 Metamorphic Testing of Relation Extraction Models

Το άρθρο *Metamorphic Testing of Relation Extraction Models* (Sun et al., 2023) εξετάζει την εφαρμογή του metamorphic testing για την αξιολόγηση μοντέλων εξαγωγής συσχετίσεων (RelationExtraction, RE), αντιμετωπίζοντας το πρόβλημα της έλλειψης αξιόπιστων test oracles.

Σε μεθοδολογικό επίπεδο, ορίζονται 8 metamorphic relations (MRs) ειδικά σχεδιασμένες για RE. Οι MRs κατηγοριοποιούνται σε τρεις ομάδες: αντικατάσταση (replacement), εναλλαγή (swap) και συνδυασμός προτάσεων (combination). Οι μετασχηματισμοί εφαρμόζονται σε προτάσεις φυσικής γλώσσας, χωρίς να αλλοιώνεται η σημασιολογική σχέση των οντοτήτων. Η αξιολόγηση ελέγχει αν το μοντέλο παραβιάζει τις αναμενόμενες σχέσεις μεταξύ αρχικής και μετασχηματισμένης εισόδου. Διότι η παραβίαση μιας MR θεωρείται ένδειξη λανθασμένης ή ασταθούς συμπεριφοράς του μοντέλου. Χρησιμοποιήθηκαν 3 μοντέλα RE (NCB – Zhou & Chen, 2021, LUKE – Yamada et al., 2020 και BERT_{EM} + MTB⁴⁴). Η αξιολόγησή τους έγινε με βάση τη μετρική violation rate (VR), η οποία μετρά το ποσοστό των περιπτώσεων όπου τα μοντέλα παραβιάζουν τις MRs. Χαμηλότερες τιμές υποδηλώνουν μικρότερο αριθμό παραβιάσεων. Τα πειραματικά αποτελέσματα έδειξαν πως τα μοντέλα NCB, BERT_{EM} + MTB και LUKE παρουσίασαν τιμές VR 21.7%, 22.5% και 25.9% αντίστοιχα. Δεδομένου ότι ο VR αποτελεί δείκτη ασυνέπειας της συμπεριφοράς του μοντέλου, χαμηλότερες τιμές αντιστοιχούν σε καλύτερη απόδοση, καθώς συνεπάγονται λιγότερες παραβιάσεις. Ωστόσο, από την οπτική της δοκιμής λογισμικού, η ύπαρξη μη μηδενικών τιμών VR είναι επιθυμητή, καθώς δείχνει πως η προτεινόμενη μέθοδος δύναται να εντοπίζει σφάλματα ή μη αναμενόμενες συμπεριφορές.

Η εφαρμογή του metamorphic testing στα μοντέλα RE παρουσιάζει σαφή πλεονεκτήματα. Η προσέγγιση επιτρέπει την αξιολόγηση της συμπεριφοράς των μοντέλων χωρίς την ανάγκη ύπαρξης test oracles ή χειροκίνητων labels. Παράλληλα, αναλύοντας τις παραβιάσεις των MRs προσφέρει βαθύτερη κατανόηση της συμπεριφοράς των μοντέλων και

⁴³ 220 για Linear Regression, IBK, Naïve Bayes, 292 για CNN-MNIST Dataset, CNN Fashion Dataset

⁴⁴ <https://github.com/thunlp/OpenNRE>

αναδεικνύει συγκεκριμένες αδυναμίες τους, όπως η υπερβολική εξάρτηση από πληροφορία τύπου οντοτήτων, η περιορισμένη γενίκευση και φαινόμενα overfitting. Τα πειραματικά αποτελέσματα έδειξαν υψηλή ικανότητα αποκάλυψης λανθανόντων σφαλμάτων⁴⁵, ενώ η ανάλυση των παραβιάσεων επέτρεψε τη σύγκριση των μοντέλων ως προς τα ισχυρά και αδύναμα σημεία τους. Πιο συγκεκριμένα, συνέκριναν τα μοντέλα εξετάζοντας τις τιμές violation rate ανά MR και ανά μοντέλο, και στη συνέχεια ερμήνευσαν τις διαφορές αυτές σε συνάρτηση με την ιδιότητα που ελέγχει κάθε relation. Επιπλέον, η προσέγγιση είναι επεκτάσιμη, καθώς επιτρέπει τον σχεδιασμό νέων μεταμορφικών σχέσεων και μπορεί να αξιοποιηθεί όχι μόνο για τον εντοπισμό αλλά και για τη διάγνωση και βελτίωση των μοντέλων RE.

Οι συγγραφείς αναγνωρίζουν και ορισμένους περιορισμούς. Η παραγωγή follow up inputs βασίζεται εν μέρει σε εργαλεία επεξεργασίας φυσικής γλώσσας (στο πλαίσιο υλοποίησης των MR- R_2 και MR- R_4 – δηλαδή 2 από τις 8 σχέσεις που λαμβάνονται υπ' όψιν), συνεπώς η ποιότητα των follow up inputs εξαρτάται από την αξιοπιστία των αντίστοιχων εργαλείων. Για τις MR- R_1 και MR- R_3 αντίστοιχα, η τυχαία αντικατάσταση οντοτήτων μπορεί να οδηγήσει σε γραμματικά ή σημασιολογικά λάθη στις προτάσεις. Παρότι πραγματοποιήθηκε επιθεώρηση της γραμματικής και σημασιολογικής εγκυρότητας των παραγόμενων follow up inputs, οι συγγραφείς αναγνωρίζουν πως είναι αδύνατο να διασφαλιστεί πλήρως πως τα test inputs είναι πλήρως απαλλαγμένα από σφάλματα, τόσο στην προτεινόμενη προσέγγιση όσο και εν γένει στην επικύρωση διεργασιών deep learning με MRs. Η μελέτη δεν περιλαμβάνει άμεση πειραματική σύγκριση της προτεινόμενης προσέγγισης με άλλες μεθόδους δοκιμής, η αξιολόγηση επικεντρώνεται (και ενδεχομένως περιορίζεται) στην εφαρμογή metamorphic testing σε διαφορετικά μοντέλα RE. Τέλος, η χρήση 8 MRs δεν καλύπτει όλες τις διαστάσεις RE, με αποτέλεσμα να υπάρχει πιθανότητα αδυναμίας ανίχνευσης ορισμένων σφαλμάτων ή ατελούς αποτύπωσης της συμπεριφοράς των μοντέλων.

4.2.3.2.3 A metamorphic testing approach for assessing Question Answering Systems

Στο άρθρο *A metamorphic testing approach for assessing Question Answering Systems* (Tu et al., 2021) η μεθοδολογία βασίζεται στη συστηματική αξιολόγηση δύο κατηγοριών συστημάτων Question Answering (QA), τα Textual Question Answering (TQA) και τα Visual Question Answering (VQA), μέσω Metamorphic Testing (MT). Τα συστήματα TQA

⁴⁵ Ο όρος αναφέρεται στη δυνατότητα της προτεινόμενης προσέγγισης να εντοπίζει σφάλματα μέσω των παραβιάσεων των metamorphic relations, που δεν ανιχνεύονται από παραδοσιακές μετρικές.

δέχονται ως είσοδο κείμενο για να απαντήσουν σε ερωτήσεις φυσικής γλώσσας, ενώ τα συστήματα VQA δέχονται ως είσοδο εικόνα. Αρχικά, ορίζεται ένα σύνολο αρχικών εισόδων (source inputs), δηλαδή ζεύγη passage - question για τα συστήματα TQA που εξάγονται από υπάρχον σύνολο δεδομένων (SQuAD 2.0, Rajpurkar et al., 2016) και image - question για τα συστήματα VQA που επίσης εξάγονται από υπάρχον σύνολο δεδομένων (DAQUAR, Malinowski&Fritz, 2014), καθώς και ένα σύνολο 17 μεταμορφικών σχέσεων (MRs), οι οποίες αποτυπώνουν ιδιότητες που θα πρέπει να ικανοποιούνται σύμφωνα με τις προσδοκίες των χρηστών. Έπειτα, με βάση κάθε MR δημιουργούνται αυτόματα τροποποιημένες (follow - up) εισοδοί, οι οποίες αποτελούν ελεγχόμενες παραλλαγές των αρχικών δεδομένων. Κατόπιν, τα συστήματα QA εκτελούνται τόσο με τις αρχικές όσο και με τις follow-up εισόδους και συλλέγονται οι αντίστοιχες απαντήσεις. Επειδή τα συστήματα επιστρέφουν απαντήσεις σε μορφή φράσης ή πρότασης, εφαρμόζεται ανάλυση σημασιολογικής ομοιότητας μεταξύ των αρχικών και των follow – up απαντήσεων, σε συνάρτηση με τη συγκεκριμένη MR, ώστε να διαπιστωθεί εάν παραβιάζεται η αναμενόμενη συμπεριφορά. Στη συνέχεια υπολογίζεται για κάθε MR και για κάθε σύστημα ο δείκτης Violation Rate (VR), ο οποίος εκφράζει το ποσοστό των περιπτώσεων όπου εντοπίστηκε παραβίαση. Τέλος, τα αποτελέσματα χρησιμοποιούνται για συγκριτική ανάλυση των συστημάτων ως προς την ικανότητά τους να κατανοούν και να επεξεργάζονται ερωτήσεις, κείμενα και εικόνες, παρέχοντας στους χρήστες τεκμηριωμένη βάση επιλογής συστήματος ανάλογα με τις ανάγκες τους.

Χρησιμοποιήθηκαν 4 συστήματα QA: AllenNLP-TQA και Transformers-TQA για την κατηγορία textual question answering καθώς και AllenNLP-VQA και CloudCV-VQA για την κατηγορία visual question answering⁴⁶. Για την προετοιμασία πηγών εισόδου TQA χρησιμοποιήθηκε το σύνολο δεδομένων SQuAD 2.0⁴⁷ και για την προετοιμασία πηγών εισόδου VQA το σύνολο δεδομένων DAQUAR⁴⁸. Η συγκριτική ανάλυση των τεσσάρων QA μοντέλων βάσει του δείκτη Violation Rate (VR) έδειξε σημαντικές διαφοροποιήσεις τόσο μεταξύ των επιμέρους συστημάτων όσο και μεταξύ των κατηγοριών TQA και VQA. Σε συνολικό επίπεδο,

⁴⁶ Οι υλοποιήσεις των συστημάτων βασίζονται στις πλατφόρμες AllenNLP, HuggingFaceTransformers και CloudCV, οι οποίες είναι διαθέσιμες στους ακόλουθους συνδέσμους:

AllenNLP: <https://github.com/allenai/allennlp>

Transformers: <https://github.com/huggingface/transformers>

CloudCVVQA: <https://github.com/Cloud-CV/VQA>

⁴⁷ Διαθέσιμο στο <https://rajpurkar.github.io/SQuAD-explorer/>

⁴⁸ Διαθέσιμο στο <https://www.cs.toronto.edu/~mren/imageqa/data/daquar/>

τα μοντέλα TQA παρουσίασαν χαμηλότερα μέσα ποσοστά παραβίασης (44.71% και 32.11%) σε σχέση με τα μοντέλα VQA (56.68% και 48.47%), υποδηλώνοντας καλύτερη συνολική σταθερότητα. Το Transformers-TQA αναδείχθηκε ως το πιο ανθεκτικό μοντέλο, ενώ το AleenNLP-VQA εμφάνισε τη μεγαλύτερη ευαισθησία.

Η προτεινόμενη προσέγγιση τοποθετείται σε σχέση με τη σχετική βιβλιογραφία, καθώς οι συγγραφές επισημαίνουν πως οι περισσότερες υπάρχουσες μελέτες επικεντρώνονται κυρίως στη διερεύνηση της robustness των συστημάτων QA ή σε επιμέρους πτυχές της λειτουργίας τους. Σε αντίθεση με αυτές, η παρούσα εργασία προτείνει μια πιο ολοκληρωμένη και προσανατολισμένη στο χρήστη προσέγγιση αξιολόγησης, βασισμένη σε ένα σύνολο *metamorphic relations* που καλύπτουν διαφορετικές διαστάσεις της συμπεριφοράς των συστημάτων. Παράλληλα, στο πειραματικό μέρος, πραγματοποιείται συγκριτική ανάλυση μεταξύ των QA συστημάτων που μελετώνται, με βάση το violation rate. Ωστόσο, η σύγκριση αυτή περιορίζεται στα υπό εξέταση συστήματα και δεν περιλαμβάνει άμεση εμπειρική αντιπαράβολή με άλλες ερευνητικές προσεγγίσεις.

Η μελέτη παρουσιάζει θετικά στοιχεία. Μέσω του ορισμού 17 μεταμορφικών σχέσεων (MRs), οι οποίες καλύπτουν διακριτές πτυχές της λειτουργικότητας των συστημάτων TQA και VQA (κατανόηση ερωτήσεων, επεξεργασία κειμένου και επεξεργασία εικόνας), η προσέγγιση επιτρέπει μια πολυδιάστατη ανάλυση της συμπεριφοράς των μοντέλων. Επιπλέον, με τη χρήση της μετρικής VR, καθίσταται δυνατή η σύγκριση τεσσάρων πραγματικών μοντέλων QA, γεγονός που υποδεικνύει πως η προσέγγιση μπορεί να έχει πρακτική εφαρμογή για τη σύγκριση συστημάτων QA, προκειμένου να αναδεικνύει συγκριτικά πλεονεκτήματα και αδυναμίες μεταξύ αυτών.

Το κατά πόσο πραγματικά υπάρχει δυνατότητα γενίκευσης φυσικά απαιτεί εμπειρική τεκμηρίωση, καθώς στη συγκεκριμένη υλοποίηση της μεθόδου χρησιμοποιήθηκαν συγκεκριμένα σύνολα δεδομένων και διεπαφές προγραμματισμού εφαρμογών (Application Programming Interfaces, APIs) γεγονός που ενδέχεται να περιορίζει την αποτελεσματικότητά της σε ευρύτερο πλαίσιο από αυτό της μελέτης. Επιπλέον, σε ορισμένα σενάρια εφαρμογής *metamorphic testing*, η χρήση μεγάλου αριθμού MRs μπορεί να αυξήσει εκθετικά το πλήθος των παραγόμενων test cases, αυξάνοντας κατά συνέπεια το χρόνο και τους υπολογιστικούς πόρους που απαιτούνται για τη διεξαγωγή των δοκιμών (Dobslaw and Feldt, 2024). Στην παρούσα

μελέτη ωστόσο, επειδή οι 17 MRs εφαρμόζονται ανεξάρτητα, γεγονός που υποδεικνύει γραμμική και όχι εκθετική αύξηση των παραγόμενων test cases, ο αριθμός τους δεν αποτελεί πρόβλημα αλλά αποδεικνύεται μάλλον πρακτικός για την αξιολόγηση των εξεταζόμενων συστημάτων QA.όποια ανησυχία αφορά επέκταση της μεθόδου σε άλλα σενάρια.

4.2.3.3 ORACLE CONSTRUCTION STRATEGY

Η στρατηγική αυτή περιλαμβάνει προσεγγίσεις που στοχεύουν στη δημιουργία ή στην προσεγγιστική υλοποίηση μηχανισμών ικανών να αποφασίζουν εάν η έξοδος ενός συστήματος τεχνητής νοημοσύνης είναι ορθή ή εσφαλμένη. Οι εργασίες αυτής της κατηγορίας προτείνουν τεχνικές, όπως η χρήση εναλλακτικών μοντέλων, στατιστικών κριτηρίων, κανόνων, ή ακόμα και άλλων νευρωνικών δικτύων, προκειμένου να εκτιμηθεί η αξιοπιστία ή η ορθότητα της παραγόμενης εξόδου. Παρότι ο όρος oracle construction εμφανίζεται συχνά και σε προσεγγίσεις όπου η τεχνητή νοημοσύνη υποστηρίζει τη δοκιμή λογισμικού, στο παρόν πλαίσιο αναφέρεται στην αντίστροφη κατεύθυνση: στην αξιοποίηση αρχών και τεχνικών της δοκιμής λογισμικού για την αντιμετώπιση του προβλήματος επαλήθευσης της ορθότητας των αποτελεσμάτων που παράγουν τα συστήματα AI, σε περιβάλλοντα όπου δεν υπάρχουν σαφώς καθορισμένες αναμενόμενες τιμές. Στη συνέχεια αναλύουμε μια προσέγγιση που εντοπίσαμε και υιοθετεί αυτή την στρατηγική.

4.2.3.3.1 A Method and Experiment to evaluate Deep Neural Networks as Test Oracles for Scientific Software

Το άρθρο (Santiago Júnior, 2022) ασχολείται με τη δυσκολία δοκιμής επιστημονικών συστημάτων λογισμικού, τα οποία συχνά εμφανίζουν μη ντετερμινιστική συμπεριφορά και παράγουν πολύπλοκα αποτελέσματα, όπως εικόνες, καθιστώντας το πρόβλημα του test oracle ιδιαίτερα δύσκολο για την αυτοματοποίηση των δοκιμών. Για την αντιμετώπιση αυτού του ζητήματος, οι συγγραφείς προτείνουν τη χρήση CNN ως διαδικασιών test oracle για τη δοκιμή επιστημονικού λογισμικού. Εισάγουν τη μέθοδο TOrC, η οποία συνδυάζει δοκιμές συνδυαστικής αλληλεπίδρασης (combinatorial interaction testing), συστηματική δημιουργία συνόλων δεδομένων, μεταφορά μάθησης (transfer learning) και ταξινόμηση βάσει CNN για την κατασκευή και αξιολόγηση αυτοματοποιημένων test oracles. Εκτός από τη βασική μέθοδο δοκιμών, οι συγγραφείς προτείνουν την τεχνική Feature and Neighborhood-based Analysis (FNA) ως μέρος της πειραματικής τους μελέτης για να εξηγήσουν τις λανθασμένες ταξινομήσεις εικόνων που παράγονται από τα CNN oracles. Οι συγγραφείς στοχεύουν έτσι να παράσχουν

τόσο μια πρακτική προσέγγιση όσο και μια εμπειρική κατανόηση της βιωσιμότητας των CNN test oracles στην επιστημονική δοκιμή λογισμικού. Να σημειωθεί πως από το σύνολο των μελετών, η συγκεκριμένη ίσως αποτελεί την «πεμπουσία» της συνέργειας μεταξύ τεχνητής νοημοσύνης και δοκιμής λογισμικού, καθώς χρησιμοποιούνται DNNs ως test oracles, των οποίων τελικά η επίδοση αξιολογείται, και για αυτό εντάσσεται στην κατεύθυνση συνεισφοράς της δοκιμής λογισμικού στην τεχνητή νοημοσύνη και όχι στην κατεύθυνση συνεισφοράς της τεχνητής νοημοσύνης στη δοκιμή λογισμικού.

Το πλαίσιο TOrC καθιερώνει μια συστηματική διαδικασία για την κατασκευή αυτοματοποιημένων test oracles για επιστημονικό λογισμικό χρησιμοποιώντας deep learning. Η διαδικασία ξεκινά με τη δημιουργία συνόλων δεδομένων εκπαίδευσης, επικύρωσης και δοκιμών μέσω συνδυαστικών δοκιμών αλληλεπίδρασης, οι οποίες εφαρμόζονται τόσο στο αρχικό επιστημονικό πρόγραμμα όσο και στις μεταλλαγμένες εκδόσεις του (στις οποίες εισάγονται τεχνητά σφάλματα), προκειμένου να παραχθούν ποικίλες και αντιπροσωπευτικές εικόνες εξόδου. Αυτά τα σύνολα δεδομένων αποτελούν τη βάση για την “επανεκπαίδευση” (fine - tuning) ενός προ-εκπαιδευμένου CNN που θα αξιοποιηθεί στη συνέχεια ως test oracle. Το CNN δεν εκπαιδεύεται εξ αρχής στο υπό εξέταση πρόβλημα, αλλά βασίζεται σε τεχνικές transfer learning. Πιο συγκεκριμένα, οι συγγραφείς χρησιμοποιούν προ-εκπαιδευμένα CNNs, των οποίων οι αρχικές παράμετροι έχουν προσδιοριστεί μέσω εκπαίδευσης σε μεγάλης κλίμακας δεδομένα γενικού σκοπού από το χώρο της υπολογιστικής όρασης. Στη συνέχεια, εφαρμόζεται διαδικασία fine tuning επί των ίδιων δικτύων, χρησιμοποιώντας δεδομένα που παράγονται από το υπό δοκιμή επιστημονικό λογισμικό, προκειμένου να προσαρμοστεί το μοντέλο στα ιδιαίτερα χαρακτηριστικά του συγκεκριμένου πεδίου εφαρμογής. Κατά συνέπεια, η αρχική εκπαίδευση του μοντέλου δεν σχετίζεται άμεσα με το domain του υπό δοκιμής συστήματος, ενώ η εξειδίκευσή του επιτυγχάνεται μέσω της επανεκπαίδευσης σε domain-specific δεδομένα. Για να βελτιώσουν την αποτελεσματικότητα και την απόδοση της εκπαίδευσης, οι συγγραφείς βασίζονται στη μεταφορά μάθησης, προσαρμόζοντας τις προ-εκπαιδευμένες αρχιτεκτονικές CNN στα συγκεκριμένα χαρακτηριστικά των επιστημονικών εικόνων εξόδου. Το CNN χρησιμοποιείται ως μηχανισμός λήψης αποφάσεων που χαρακτηρίζει ως επιτυχημένα ή αποτυχημένα τα νέα αποτελέσματα που παράγονται από το υπό δοκιμή σύστημα, αφού προηγηθεί η διαδικασία fine-tuning του προ-εκπαιδευμένου CNN, αξιοποιώντας τα νέα δεδομένα εκπαίδευσης που δημιουργούνται στο πλαίσιο της προσέγγισης όπως περιγράφεται

παραπάνω. Μέσω αυτής της δομημένης ροής εργασίας, οι συγγραφείς είναι σε θέση να διερευνήσουν πώς διαφορετικές επιλογές σχεδιασμού (το βάθος του δικτύου, η σύνθεση του συνόλου δεδομένων και η στρατηγική μάθησης) επηρεάζουν την ακρίβεια και την αξιοπιστία των test oracles που βασίζονται σε CNN. Στο πλαίσιο της πειραματικής αξιολόγησης της προτεινόμενης μεθόδου, οι συγγραφείς εξετάζουν 7 διαφορετικές αρχιτεκτονικές CNN με διαφορετικό βάθος δικτύου, με σκοπό να αξιολογήσουν την απόδοσή τους ως test oracles σε σύνολα εικόνων που παράγονται από ορθές και μεταλλαγμένες εκδόσεις επιστημονικών προγραμμάτων. Η απόδοση των μοντέλων μετράται ως προς την ακρίβεια ταξινόμησης των παραγόμενων εικόνων.

Ως προς τα θετικά στοιχεία της μελέτης, οι συγγραφείς παρέχουν εμπειρικές πληροφορίες σχετικά με τη συμπεριφορά των test oracles που βασίζονται σε CNN, δείχνοντας ότι τα δίκτυα μεγαλύτερου βάθους δεν επιτυγχάνουν απαραίτητα καλύτερη απόδοση και ότι η επιλογή της αρχιτεκτονικής του δικτύου παίζει σημαντικό ρόλο στην ακρίβεια του test oracle. Οι συγγραφείς στο πλαίσιο της προσέγγισης αναδεικνύουν τη σημασία μιας εστιασμένης στα δεδομένα προοπτικής, δηλώνοντας ρητά ότι η “data-centric AI” αντιπροσωπεύει μια πολλά υποσχόμενη κατεύθυνση για τη βελτίωση της αποτελεσματικότητας των test oracles, καθώς αρκεί η βελτίωση των δεδομένων εκπαίδευσης για να επιφέρει βελτίωση στην απόδοση των αντίστοιχων εκπαιδευόμενων μοντέλων. Τέλος, η διερεύνηση της σχέσης μεταξύ των τεχνητά εισαγόμενων ελαττωμάτων στο λογισμικό και των σφαλμάτων των test oracles (εσφαλμένες ταξινομήσεις) δείχνει ότι δεν υπάρχει σαφής συσχέτιση ή πως η σχέση αυτή είναι γραμμική, προσφέροντας σημαντικές πληροφορίες σχετικά με τη συμπεριφορά των CNN test oracles. Στην ίδια κατεύθυνση συνεισφέρει η εισαγωγή της τεχνικής FNA ως ένα πρακτικό μέσο για την εξήγηση των εσφαλμένων ταξινομήσεων, ενισχύοντας τη δυνατότητα ερμηνείας της συμπεριφοράς του test oracle αλλά και τη συνολική αναλυτική συμβολή της μελέτης. Τα πειραματικά αποτελέσματα δείχνουν ότι η απόδοση των CNN ως test oracles κυμαίνεται σημαντικά ανάλογα με την αρχιτεκτονική του μοντέλου, τις παραμέτρους και τα σύνολα δεδομένων που χρησιμοποιούνται, με την υψηλότερη ακρίβεια να φτάνει το 80% (CNN DenseNet-161, SS data set profile). Παράλληλα, η προτεινόμενη τεχνική FNA για την ανάλυση των εσφαλμένων ταξινομήσεων επιτυγχάνει υψηλή ακρίβεια 96.55%.

Να σημειωθεί πως ιδανικά ένας test oracle θα πρέπει να είναι απολύτως ακριβής. Ωστόσο, σε πολλές κατηγορίες συστημάτων, συμπεριλαμβανομένου και του επιστημονικού λογισμικού κατά τους συγγραφείς, δεν είναι γνωστό το αναμενόμενο σωστό αποτέλεσμα (oracle problem). Στη συγκεκριμένη μελέτη, τα CNN λειτουργούν ως προσεγγιστικοί μηχανισμοί ελέγχου, οι οποίοι παρότι όχι αλάνθαστοι, μπορούν να υποστηρίξουν την ανίχνευση πιθανών σφαλμάτων σε περιβάλλοντα όπου δεν υπάρχει διαθέσιμος παραδοσιακός μηχανισμός test oracle απόλυτης ακρίβειας.

Ως προς τα αρνητικά στοιχεία της μελέτης, οι συγγραφείς αναγνωρίζουν ρητά ότι τα πειραματικά συμπεράσματα υπόκεινται σε ζητήματα γενίκευσης, σημειώνοντας ότι απαιτείται μελλοντική εργασία για την επικύρωση της προσέγγισης σε περισσότερους τύπους δεδομένων και πειραματικών πλαισίων, γεγονός που υποδηλώνει ότι το τρέχον πεδίο αξιολόγησης είναι περιορισμένο. Παρατηρούν περαιτέρω ότι, αν και η μεταφορά γνώσης βελτιώνει την απόδοση, ενδέχεται να είναι απαραίτητη η επέκταση των υφιστάμενων αρχιτεκτονικών μοντέλων για την επίτευξη καλύτερων αποτελεσμάτων, αποκαλύπτοντας περιορισμούς στις τρέχουσες διαμορφώσεις μοντέλων. Επιπλέον, ενώ η προτεινόμενη τεχνική FNA επιδεικνύει υψηλή ακρίβεια στην εξήγηση των εσφαλμένων ταξινομήσεων, οι συγγραφείς τονίζουν ότι θα πρέπει να συγκριθεί εκτενέστερα με υφιστάμενες τεχνικές, όπως η κάλυψη νευρώνων, το DeepGauge (Ma et al., 2018) και το SADL (Kim et al., 2019), υπογραμμίζοντας ότι η αξιολόγησή της παραμένει ατελής. Συνολικά, αυτοί οι περιορισμοί υποδηλώνουν ότι, αν και η προτεινόμενη μέθοδος είναι αποτελεσματική στο πλαίσιο της μελέτης, απαιτείται περαιτέρω έρευνα για την ενίσχυση της ανθεκτικότητας και της δυνατότητας εφαρμογής της σε ευρύτερους επιστημονικούς τομείς λογισμικού.

4.2.3. BOUNDARY/ADVERSARIAL ORIENTED STRATEGY

Η βασική αρχή αυτής της στρατηγικής προέρχεται από την παραδοσιακή θεωρία δοκιμών λογισμικού, σύμφωνα με την οποία τα σφάλματα εμφανίζονται συχνότερα στα όρια των επιτρεπτών τιμών ή σε ακραίες καταστάσεις. Στο πλαίσιο της τεχνητής νοημοσύνης, η ιδέα αυτή μεταφράζεται στην αναζήτηση εισόδων που βρίσκονται κοντά στα όρια απόφασης του μοντέλου ή που προκαλούν παραπλανητικές ταξινομήσεις, γνωστές ως adversarial examples (εχθρικά/αντιφατικά παραδείγματα). Οι μελέτες που εντάσσονται σε αυτή την κατηγορία επικεντρώνονται κυρίως στην παραγωγή, ανάλυση ή συστηματική εξερεύνηση τέτοιων οριακών

περιπτώσεων, με στόχο την αποκάλυψη ευθραυστότητας, αστάθειας ή έλλειψης γενίκευσης του μοντέλου.

4.2.3.4.1 Automated Boundary Identification for Machine Learning Classifiers

Το άρθρο *Automated Boundary Identification for Machine Learning Classifiers* (Dobslaw & Feldt, 2024) προτείνει μια αυτοματοποιημένη προσέγγιση για τον εντοπισμό των ορίων απόφασης (decision boundaries) μοντέλων μηχανικής μάθησης, με στόχο τη συστηματική δοκιμή και καλύτερη κατανόηση της συμπεριφοράς τους.

Εισάγεται ο αλγόριθμος ML-BSA (Machine Learning Boundary Search Algorithm) για να εντοπίσει τα όρια ως πρόβλημα βελτιστοποίησης. Η μέθοδος αυτή αναζητά ζεύγη εισόδων πολύ κοντά μεταξύ τους που όμως οδηγούν σε διαφορετικές προβλέψεις. Τα ζεύγη αυτά λειτουργούν ως boundary candidates, αποκαλύπτοντας κρίσιμες περιοχές κοντά στα όρια απόφασης. Για την καλύτερη κάλυψη των ορίων ενσωματώνεται στρατηγική διαφοροποίησης (diversification strategy), η οποία βασίζεται στη μεγιστοποίηση της απόστασης κάθε υποψηφίου ζεύγους ορίου από τους δύο κοντινότερους γείτονές του, οι οποίοι επίσης αποτελούν ζεύγη ορίων, με στόχο τη διασφάλιση πως τα υποψήφια ζεύγη ορίων δε συγκεντρώνονται τοπικά. Η μέθοδος αξιολογείται σε 7 σύνολα δεδομένων, όπου η εφαρμογή της στρατηγικής διαφοροποίησης οδηγεί σε μεγαλύτερη κάλυψη των ορίων. Για παράδειγμα, στο σύνολο δεδομένων iris η μέση απόσταση μεταξύ boundary candidates αυξάνεται από $2,23 \pm 0,79$ σε $5,22 \pm 0,22$ μετά την εφαρμογή στρατηγικής διαφοροποίησης, καλύπτοντας έτσι πιο εκτεταμένα το όριο απόφασης.

Η προτεινόμενη προσέγγιση για τον εντοπισμό ορίων απόφασης παρουσιάζει ορισμένα θετικά στοιχεία για τη δοκιμή μοντέλων μηχανικής μάθησης. Καταρχάς, η ενσωμάτωση της διάστασης/κριτηρίου diversity οδηγεί συστηματικά σε μεγαλύτερες αποστάσεις μεταξύ των υποψηφίων σημείων και μικρότερη διακύμανση, γεγονός που υποδηλώνει ευρύτερη και πιο ισοκατανεμημένη κάλυψη του χώρου εισόδου. Επιπλέον, τα παραγόμενα boundary candidates αποτελούν ζεύγη εισόδων με μικρή απόσταση μεταξύ τους, τα οποία οδηγούν το μοντέλο σε διαφορετικές ταξινομήσεις, υποδηλώνοντας την ύπαρξη ορίου απόφασης μεταξύ τους. Οπότε, τα σημεία αυτά μπορούν να αξιοποιηθούν άμεσα ως περιπτώσεις δοκιμής, είτε μέσω της απευθείας εκτέλεσής τους, είτε μέσω περαιτέρω τοπικής διερεύνησης με μικρές μεταβολές, με στόχο την αξιολόγηση της σταθερότητας και της ορθότητας της συμπεριφοράς του μοντέλου. Ιδιαίτερα

σημαντική είναι η καταλληλότητά της για black-box ή μη διαφορίσιμα μοντέλα⁴⁹, όπως δένδρα αποφάσεων (decision trees) και random forest, όπου δεν είναι διαθέσιμες εσωτερικές πληροφορίες. Επιπλέον, το πλαίσιο μπορεί να αναδεικνύει περιοχές χαμηλής εμπιστοσύνης που απαιτούν ανθρώπινη διερεύνηση και, σε περιβάλλοντα χαμηλής διάστασης, να υποστηρίξει οπτική αναπαράσταση του ορίου. Η προτεινόμενη τεχνική επιδιώκει την ευρύτερη κάλυψη σημαντικών τμημάτων του decision boundary, κάτι που επιβεβαιώνεται και από την εκτεταμένη επαναληπτική δειγματοληψία που πραγματοποιήθηκε στα πειράματα.

Παρά τα θετικά στοιχεία της προσέγγισης, οι συγγραφείς αναγνωρίζουν ορισμένους περιορισμούς. Η προσέγγιση απαιτεί επαναλαμβανόμενες εκτελέσεις κατά την αναζήτηση boundary candidates, οπότε ενδέχεται να επιφέρει αυξημένο υπολογιστικό κόστος σε παραδοσιακά συστήματα. Επιπλέον, η παρούσα μελέτη δεν υποστηρίζει κατηγορικές εισόδους, αν και επισημαίνεται ότι κάτι τέτοιο θα μπορούσε να ενσωματωθεί σχετικά εύκολα. Ένας ακόμη περιορισμός αφορά την κλιμάκωση της μεθόδου σε προβλήματα πολλαπλών κλάσεων, όπου ο θεωρητικός αριθμός πιθανών ορίων απόφασης αυξάνεται σημαντικά, χωρίς να έχει διερευνηθεί εμπειρικά πόσα boundary candidates είναι απαραίτητα στην πράξη. Τέλος, η προτεινόμενη έννοια της αβεβαιότητας που βασίζεται στην απόσταση από τα όρια παρουσιάζεται ενδεικτικά, χωρίς συστηματική σύγκριση με άλλες καθιερωμένες μετρικές εμπιστοσύνης.

4.3 ΠΕΡΙΛΗΨΗ

Παρατηρώντας την κατανομή των άρθρων που μελετήθηκαν ανά κατεύθυνση συνεισφοράς, παρατηρούνται κατευθύνσεις οι οποίες φαίνεται πως έχουν «εξερευνηθεί» περισσότερο, σε αντίθεση με κάποιες που η κατανομή υποδεικνύει πως υπάρχει περιθώριο περαιτέρω έρευνας και συνεισφοράς.

Αναφορικά με τη συνεισφορά της τεχνητής νοημοσύνης στη δοκιμή λογισμικού, προκύπτει πως μια ερευνητική κατεύθυνση που παρουσιάζει δυσανάλογο αριθμό άρθρων σε σχέση με τις προοπτικές της είναι η αλγοριθμική συνεισφορά. Μέσω αυτής, μπορούν να προκύψουν νέοι αλγόριθμοι ή η βελτιστοποίηση υφιστάμενων. Οι αλγόριθμοι που θα προκύψουν μπορούν να χρησιμοποιηθούν ως επιμέρους στοιχεία πλαισίων και μοντέλων, συμβάλλοντας στην περαιτέρω εξέλιξη και στις αντίστοιχες κατευθύνσεις συνεισφοράς που

⁴⁹ Ως μη διαφορίσιμα μοντέλα ορίζονται μοντέλα μηχανικής μάθησης των οποίων η συνάρτηση πρόβλεψης δεν είναι διαφορίσιμη (δεν ορίζεται η παράγωγος αυτής) ως προς τις εισόδους ή τις παραμέτρους της.

αφορούν την παροχή πλαισίων (methodological/framework contribution) ή μοντέλων (model/learning based contribution).

Αντίστοιχα, αναφορικά με τη συνεισφορά της δοκιμής λογισμικού στην τεχνητή νοημοσύνη, προκύπτει από την κατανομή των άρθρων πως έχει ήδη παγιωθεί ως κατεύθυνση συνεισφοράς το coverage-driven exploration (φαίνεται πως μάλλον ήταν η κύρια ερευνητική τάση για τη συνεισφορά της δοκιμής λογισμικού στην τεχνητή νοημοσύνη το 2019), ωστόσο παρουσιάζεται ως ανερχόμενη κατεύθυνση η metamorphic validation, καθώς υπάρχουν μελέτες (Improving Early Fault Detection in Machine Learning Systems, Srinivasan & Kanewala, 2024) που υποδεικνύουν πως προσεγγίσεις metamorphic testing μπορούν να λειτουργήσουν εξαιρετικά πιο αποδοτικά σε σχέση με αντίστοιχες coverage-driven για την αξιολόγηση μοντέλων νευρωνικών δικτύων. Ενδιαφέρον παρουσιάζουν επίσης οι κατευθύνσεις συνεισφοράς oracle construction και boundary/adversarial analysis, παρόλο που στη βιβλιογραφία που μελετήθηκε και αναλύθηκε ισοβαθμούν τελευταίες στον αριθμό άρθρων. Η oracle construction γιατί εκφράζει ίσως καλύτερα από όλες τις υπόλοιπες τη διάδραση μεταξύ τεχνητής νοημοσύνης και δοκιμής λογισμικού, καθώς πέρα από την αξιοποίηση των αρχών του testing για την αξιολόγηση αποτελεσμάτων συστημάτων τεχνητής νοημοσύνης, μπορούν και συστήματα τεχνητής νοημοσύνης να υποστηρίξουν μηχανισμούς test oracle. Η boundary/adversarial analysis γιατί επεκτείνει μια παραδοσιακή μεθοδολογία/νοοτροπία (έλεγχος ακραίων περιπτώσεων) σε σύγχρονα περιβάλλοντα συστημάτων τεχνητής νοημοσύνης οπότε μπορεί να αναδείξει αδυναμίες τους. Αν για παράδειγμα το υπό δοκιμή σύστημα τεχνητής νοημοσύνης επιδεικνύει υψηλή ευαισθησία σε μη σημασιολογικές μεταβολές στα δεδομένα εισόδου, η υψηλή αυτή ευαισθησία αποτελεί ένδειξη πως το μοντέλο δεν έχει εκπαιδευθεί με κατάλληλα δεδομένα ώστε να παράγει ποιοτικές εξόδους, αλλά βασίζεται σε επιφανειακά μοτίβα.

5. ΕΡΕΥΝΗΤΙΚΑ ΚΕΝΑ ΚΑΙ ΠΡΟΤΕΙΝΟΜΕΝΕΣ ΕΡΕΥΝΗΤΙΚΕΣ ΚΑΤΕΥΘΥΝΣΕΙΣ

Σκοπός του κεφαλαίου αυτού είναι να επεξηγήσει τα ερευνητικά κενά που αναδείχθηκαν από την ανάλυση των άρθρων καθώς και τις προτεινόμενες ερευνητικές κατευθύνσεις για την αντιμετώπιση αυτών. Ο παρακάτω πίνακας παρουσιάζει την αντιστοίχιση των ερευνητικών προκλήσεων (ανά κατεύθυνση/πεδίο συνεισφοράς) με συγκεκριμένες κατευθύνσεις για την αντιμετώπισή τους.

Πίνακας 6. Ερευνητικά κενά και αντίστοιχες προτεινόμενες ερευνητικές κατευθύνσεις

Κατεύθυνση συνεισφοράς	Ερευνητικά κενά	Προτεινόμενες ερευνητικές κατευθύνσεις
AI in Software testing	Εγγενείς αδυναμίες	Δημιουργία «τυποποιημένων» εννοιολογικών πλαισίων, μηχανισμών ενσωμάτωσης αυτών στη χρήση των LLM για ορισμένο σκοπό και μηχανισμών επαλήθευσης των παραγομένων αποτελεσμάτων.
	Πληρότητα συνόλων δεδομένων και μετρικών	Εστίαση στην προσεκτική σχεδίαση των συνόλων δεδομένων ανά τομέα, ώστε να προκύψουν πρότυπα αντιπροσωπευτικά σύνολα.
	Αδυναμία επανασχεδιασμού ροής	Ανάπτυξη πλαισίων δοκιμών με ενσωματωμένους πράκτορες για την επίτευξη προσαρμοστικής δυναμικής ροής.
	Ασαφής δυνατότητα επέκτασης χρήσης DNN ως test oracles σε ευρύτερα πλαίσια	Δημιουργία αξιόπιστων μεθόδων και εργαλείων της απόδοσης των DNN ως test oracles (Santiago Júnior, 2022).
Software testing in AI	Αμφισβητήσιμες μετρικές κάλυψης ως απόδειξη λειτουργικής ορθότητας μοντέλων	Κατασκευή πλήρων benchmarks για την ολοκληρωμένη εμπειρική αξιολόγηση των νέων μετρικών.
	Καθορισμός ενδεδειγμένου αριθμού metamorphic relations (MRs)	Μείωση αριθμού (μέσω φιλτραρίσματος, σύνθεσης ή συσταδοποίησης) / βελτίωση ποιότητας MRs.
Και οι δύο (AI in Software testing & Software	Οι προτεινόμενες προσεγγίσεις να είναι κυρίως διαγνωστικού και όχι διορθωτικού χαρακτήρα	Αξιοποίηση επαναληπτικών τεχνικών (regression testing, coverage-guided fuzzing) σε συνδυασμό με συστήματα agents, με σκοπό οι τελικές προσεγγίσεις να επεκτείνονται και στην επιδιόρθωση των ελαττωμάτων που ανιχνεύονται.

testing in AI)		
----------------	--	--

5.1 ΕΓΓΕΝΕΙΣ ΑΔΥΝΑΜΙΕΣ ΤΩΝ LLM

Στο άρθρο *A method for IoT devices test case generation using language models* (Kumar et al., 2025) οι συγγραφείς επισημαίνουν κάποιες εγγενείς αδυναμίες που παρατηρούνται κατά τη χρήση των LLMs στην τρέχουσα μορφή τους. Αυτές περιλαμβάνουν:

- Hallucinations⁵⁰ ή ανακριβείς προτάσεις που δεν ευθυγραμμίζονται με τις προδιαγραφές του συστήματος.
- Ελλιπή λογική επικύρωσης αν η προτροπή (prompt) προς το LLM υστερεί σαφήνειας ή το εννοιολογικό πλαίσιο δεν είναι αυστηρά καθορισμένο.
- Κενά ασφαλείας, όταν αντίστοιχη τεκμηρίωση (documentation) για το υπό δοκιμή σύστημα δεν αναφέρει ρητά λεπτομέρειες για την αυθεντικοποίηση και την κρυπτογράφηση.

Προτεινόμενη ερευνητική κατεύθυνση: Δημιουργία «τυποποιημένων» εννοιολογικών πλαισίων, μηχανισμών ενσωμάτωσης αυτών στη χρήση των LLM για ορισμένο σκοπό και μηχανισμών επαλήθευσης των παραγόμενων αποτελεσμάτων. Ενδεικτικά, οι Kumar et al. (2025) προτείνουν ένα πλαίσιο που εφαρμόζει όλα τα σκέλη της προτεινόμενης ερευνητικής κατεύθυνσης, αφενός αντλώντας σχετικές πληροφορίες για τα υπό δοκιμή συστήματα από διανυσματικές βάσεις δεδομένων συναφών με το υπό μελέτη σύστημα, αφετέρου ορίζοντας σημεία ελέγχου (checkpoints), στα οποία γίνεται διασταύρωση των παραγόμενων αποτελεσμάτων του LLM με τη βοήθεια εξειδικευμένου προσωπικού (όπως είναι ένας μηχανικός λογισμικού ή μηχανικός ποιότητας). Θα μπορούσαν δηλαδή με αντίστοιχο τρόπο να δημιουργηθούν αντίστοιχα πλαίσια για τον εκάστοτε τομέα ενδιαφέροντος, προς το παρόν με τη βοήθεια εξειδικευμένου ανθρώπινου δυναμικού. Στη συνέχεια, όταν το πλήθος των μελετών θα είναι επαρκές, θα μπορούσαν να δημιουργηθούν αντίστοιχες βάσεις δεδομένων αναμενόμενων αποτελεσμάτων, με στόχο την αυτοματοποίηση της διαδικασίας, ενδεχομένως με την εφαρμογή

⁵⁰ Με τον όρο hallucinations αναφερόμαστε στην παραγωγή περιεχομένου από LLMs, το οποίο, παρά την φαινομενική του αξιοπιστία, περιέχει πραγματολογικά σφάλματα, ανύπαρκτες αναφορές ή αυθαίρετες συσχετίσεις.

επιπλέον τεχνικών τεχνικής νοημοσύνης (π.χ. με τη χρήση agent που εκμεταλλεύεται τεχνικές Reinforcement Learning για να συγκρίνει τα παραγόμενα αποτελέσματα του LLM με τα αναμενόμενα στην αντίστοιχη βάση δεδομένων που θα έχει παραχθεί από εξειδικευμένο προσωπικό).

5.2 ΠΛΗΡΟΤΗΤΑ ΣΥΝΟΛΩΝ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΜΕΤΡΙΚΩΝ

Αναφέρεται στη βιβλιογραφία (Dewangan et al., 2022, Ouédraogo et al., 2025) πως χρησιμοποιήθηκαν σύνολα δεδομένων, που ναι μεν θεωρούνται ενδεικτικά, δεν καλύπτουν ωστόσο αντιπροσωπευτικά το σύνολο του χώρου εξερεύνησης του εκάστοτε προβλήματος. Αυτό είναι ένα ζήτημα το οποίο επεκτείνεται σε κάθε προσέγγιση που αξιοποιεί σύνολα δεδομένων, καθώς αν αυτά δεν είναι αντιπροσωπευτικά του εκάστοτε προβλήματος, θα επηρεάζεται αρνητικά η επίδοσή τους σε πραγματικά περιβάλλοντα.

Προτεινόμενη ερευνητική κατεύθυνση: Εστίαση στην προσεκτική σχεδίαση των συνόλων δεδομένων ανά τομέα, ώστε να προκύψουν πρότυπα σύνολα, πλήρως αντιπροσωπευτικά, τα οποία στη συνέχεια θα μπορούσαν να χρησιμοποιηθούν τόσο για την εκπαίδευση μοντέλων όσο και για την αξιολόγησή τους.

5.3 ΔΥΝΑΜΙΑ ΕΠΑΝΑΣΧΕΔΙΑΣΜΟΥ ΡΟΗΣ

Στο άρθρο *TestLoter: A logic-driven framework for automated unit test generation and error repair using large language models* (Yang et al., 2025), αναφέρεται πως ένα ολόκληρο πλαίσιο καθίσταται μη λειτουργικό στις περιπτώσεις που το στάδιο δημιουργίας περιπτώσεων δοκιμών δεν επιτυγχάνει την παραγωγή βιώσιμων περιπτώσεων δοκιμών ή η φάση επιδιόρθωσης δεν είναι σε θέση να αντιμετωπίσει σφάλματα σύνταξης και εκτέλεσης. Η ισχυρή αυτή αλληλεξάρτηση μεταξύ των επιμέρους φάσεων υποδηλώνει έλλειψη αυτονομίας και περιορισμένες δυνατότητες προσαρμογής και ανάκαμψης ενός πλαισίου.

Προτεινόμενη ερευνητική κατεύθυνση: Ανάπτυξη πλαισίων δοκιμών στα οποία θα ενσωματωθούν αυτόνομοι πράκτορες (με ενσωματωμένα μοντέλα μηχανικής μάθησης), που θα μπορούν να συνεργάζονται, να αναγνωρίζουν αστοχίες και να επανακαθορίζουν δυναμικά τη στρατηγική δοκιμής, ενισχύοντας τη βιωσιμότητα, προσαρμοστικότητα και την ευελιξία του (εκάστοτε) πλαισίου.

5.4 ΜΕΤΡΙΚΕΣ ΚΑΛΥΨΗΣ ΩΣ ΑΠΟΔΕΙΞΗ ΛΕΙΤΟΥΡΓΙΚΗΣ ΟΡΘΟΤΗΤΑΣ ΜΟΝΤΕΛΩΝ

Οι μετρικές κάλυψης αξιοποιούνται ως ενδείξεις της αποτελεσματικότητας της εξερεύνησης του χώρου εισόδων και όχι ως απόδειξη λειτουργικής ορθότητας των μοντέλων. Αυτό έχει αναδειχθεί σε μελέτες που εμπίπτουν στην κατηγορία coverage-driven exploration, όπου παρατηρείται πως οι παραδοσιακές μετρικές αξιολόγησης λογισμικού, όπως η κάλυψη κώδικα, δεν είναι αποτελεσματικές στην αξιολόγηση συστημάτων με νευρωνικά δίκτυα. Ήδη για την αντιμετώπιση του ζητήματος, κάποιες από τις μελέτες εισάγουν νέες μετρικές αξιολόγησης, όπως η μετρική κάλυψης νευρώνων στο άρθρο *DeepXplore: Automated White box Testing of Deep Learning Systems* (Pie et al., 2019) και τα κριτήρια κάλυψης στο άρθρο *Structural Test Coverage Criteria for Deep Neural Networks* (Sun et al., 2019). Όμως, πολλές από τις νέες μετρικές υστερούν επαρκούς εμπειρικής τεκμηρίωσης. Ήδη υπάρχουν στη βιβλιογραφία μελέτες που έχουν αποπειραθεί να αναπτύξουν κάποιο εργαλείο αξιολόγησης με σκοπό να συγκρίνουν τις μετρικές κάλυψης, ωστόσο φαίνεται να απαιτείται περαιτέρω προσπάθεια σε αυτήν την κατεύθυνση, καθώς δεν επιτυγχάνεται τελικά κάποια κατάταξη που να κατηγοριοποιεί τις μετρικές με τέτοιο τρόπο που να μπορεί να αξιοποιηθεί σε ευρύτερη κλίμακα (Usman et al., 2022).

Προτεινόμενη ερευνητική κατεύθυνση: Διεξαγωγή μελετών με τη χρήση διαφορετικών ειδών νευρωνικών δικτύων και συνόλων δεδομένων εκπαίδευσης, στο πλαίσιο εμπειρικής τεκμηρίωσης των νέων μετρικών που προτείνονται στη σύγχρονη βιβλιογραφία με στόχο τη παραγωγή ολοκληρωμένων benchmarks. Η πιθανή αποτυχία εμπειρικής τεκμηρίωσης κάποιων εκ των νέων μετρικών μπορεί να αναδείξει την αναγκαιότητα πιο αποτελεσματικών και κατάλληλων μετρικών από τις προτεινόμενες στη σύγχρονη βιβλιογραφία.

5.5 ΑΣΑΦΗΣ ΔΥΝΑΤΟΤΗΤΑ ΕΠΕΚΤΑΣΗΣ ΧΡΗΣΗΣ DNN ΩΣ TEST ORACLES ΣΕ ΕΥΡΥΤΕΡΑ ΠΛΑΙΣΙΑ

Η βιβλιογραφία (Santiago Júnior, 2022) δεν έχει καταλήξει στο κατά πόσο τα μοντέλα βαθιάς μάθησης μπορούν να λειτουργήσουν με συνέπεια ως υποκατάστατα των παραδοσιακών oracles, καθώς εγείρονται ζητήματα γενίκευσης λόγω ανεπαρκούς εμπειρικής τεκμηρίωσης σε περισσότερους τύπους δεδομένων και πειραματικά πλαίσια. Προκύπτει έτσι ένα ουσιαστικό ερευνητικό κενό που πρέπει να καλυφθεί.

Προτεινόμενη ερευνητική κατεύθυνση: Συνδυαστική αξιοποίηση πολλαπλών ανεξάρτητα εκπαιδευμένων DNN. Η σύγκριση και η συμφωνία ή απόκλιση μεταξύ των προβλέψεων του κάθε επιμέρους DNN θα μπορούσε να δίνει ενδείξεις για πιθανές αστοχίες του συστήματος υπό δοκιμή. Αναφέρεται ως παράδειγμα ο εντοπισμός ενός corner case μόνο από ένα και όχι από όλα τα DNN που απαρτίζουν το μηχανισμό test oracle. Με τον τρόπο αυτό δημιουργείται ένας συλλογικός μηχανισμός test oracle. Παράλληλα, τεχνικές transfer learning θα μπορούσαν να αξιοποιηθούν για την προσαρμογή των DNN σε διαφορετικά αλλά συναφή περιβάλλοντα εφαρμογών (Pie et al, 2019, Santiago Júnior, 2022).

5.6 ΚΑΘΟΡΙΣΜΟΣ ΕΝΔΕΛΕΙΓΜΕΝΟΥ ΑΡΙΘΜΟΥ METAMORPHIC RELATIONS

Στο άρθρο *Improving Early Fault Detection in Machine Learning Systems Using Data Diversity-Driven Metamorphic Relation Prioritization* (Srinivasan & Kanewala, 2024) επισημαίνεται πως η αύξηση των MR (ειδικά όταν αυτές δεν είναι ανεξάρτητες μεταξύ τους) κατά μικρό αριθμό μπορεί να επιφέρει εκθετική αύξηση των παραγόμενων περιπτώσεων δοκιμών, πολλές από τις οποίες μπορεί να είναι περιττό να εκτελεστούν ή να είναι λιγότερο σημαντικές σε σύγκριση με άλλες. Συνεπώς, αυτό μπορεί να οδηγήσει και στην σημαντική αύξηση του χρόνου δοκιμών. Επομένως, υπάρχει η ανάγκη για την ανίχνευση και φιλτράρισμα περιττών δοκιμών (άρα πχ. το φιλτράρισμα των MRs) ή για την παραγωγή πιο ποιοτικών και ανεξάρτητων μεταξύ τους MRs.

Προτεινόμενη ερευνητική κατεύθυνση 1: Δημιουργία μηχανισμών φιλτραρίσματος ή σύνθεσης MRs. Οι μηχανισμοί «φιλτραρίσματος» των MR θα αναγνωρίζουν τις περιπτώσεις δοκιμής που προκύπτουν και είναι περιττό να εκτελεστούν, μειώνοντας έτσι την προσπάθεια και το αντίστοιχο κόστος των δοκιμών λογισμικού και επιτρέποντας την εστίαση στις πιο σημαντικές MR. Οι μηχανισμοί αυτοί μπορεί να περιλαμβάνουν τη προτεραιοποίηση των MRs με βάση συγκεκριμένες μετρικές, την ανάλυση εξαρτήσεων για την αφαίρεση περιττών MRs ή ακόμη και τη σύνθεση επιμέρους MRs σε ενιαίες πιο σύνθετες MRs καθώς και τη συσταδοποίηση MRs με σκοπό τον περιορισμό του αριθμού τους, επιλέγοντας μία προς εκτέλεση από κάθε συστάδα.

Προτεινόμενη ερευνητική κατεύθυνση 2: Διερεύνηση μεθόδων για τη συστηματική παραγωγή MRs που να είναι ανεξάρτητες, σημασιολογικές και αποτελεσματικές στην ανακάλυψη σφαλμάτων, με στόχο τη μείωση πλεοναζουσών πληροφοριών και βελτίωση της αποδοτικότητας του metamorphic testing.

5.7 ΕΣΤΙΑΣΗ ΣΤΗ ΔΙΑΓΝΩΣΗ ΚΑΙ ΟΧΙ ΣΤΗ ΔΙΟΡΘΩΣΗ

Μέχρι στιγμής, οι τρέχουσες προσεγγίσεις είναι αμιγώς διαγνωστικού χαρακτήρα και όχι διορθωτικού. Αυτό ισχύει και για τα δύο πεδία συνεισφοράς. Αυτό σημαίνει πως η ομάδα ενός έργου λογισμικού θα πρέπει να σπαταλά σημαντικό χρόνο για να αξιολογεί την ορθότητα των διαγνώσεων, τον εντοπισμό των αντίστοιχων προβληματικών μερών του κώδικα και την διόρθωσή τους. Συνεπώς, απαιτείται περαιτέρω αυτοματοποίηση της διαδικασίας ώστε να μειωθεί τόσο ο φόρτος εργασίας όσο και το αντίστοιχο κόστος καθώς και ο χρόνος ανάπτυξης.

Προτεινόμενη ερευνητική κατεύθυνση: Αξιοποίηση επαναληπτικών τεχνικών (regression testing, coverage-guided fuzzing) σε συνδυασμό με συστήματα agents, με σκοπό οι τελικές προσεγγίσεις να επεκτείνονται και στην επιδιόρθωση των ελαττωμάτων που ανιχνεύονται.

6. ΣΥΜΠΕΡΑΣΜΑΤΑ

6.1 ΣΥΝΕΙΣΦΟΡΑ

Η συνεισφορά της παρούσας μελέτης αφορά την αποτύπωση της τρέχουσας κατάστασης της έρευνας σε δύο πεδία εφαρμογής: (α) στην εφαρμογή μεθόδων και τεχνικών τεχνητής νοημοσύνης στη δοκιμή λογισμικού και (β) στην εφαρμογή μεθόδων δοκιμής λογισμικού σε συστήματα τεχνητής νοημοσύνης, μέσω συστηματικής βιβλιογραφικής ανασκόπησης. Η αποτύπωση της τρέχουσας κατάστασης ανά πεδίο εφαρμογής αναγνώρισε κύριες κατευθύνσεις συνεισφοράς, οδήγησε στην ταξινόμησή των σχετικών προσεγγίσεων βάσει ιεραρχικής κατηγοριοποίησης και στην βαθιά ανάλυση των προσεγγίσεων με ανάδειξη των πλεονεκτημάτων και μειονεκτημάτων τους καθώς και στον καθορισμό των καθολικών δυνατών σημείων και αδυναμιών. Επιπλέον, η παρούσα μελέτη ανέδειξε τα κενά που προέκυψαν από την ανάλυση της βιβλιογραφίας και πρότεινε αντίστοιχες ερευνητικές κατευθύνσεις για την αντιμετώπιση αυτών.

Τα βασικά οφέλη των παραπάνω συνεισφορών της εργασίας μας είναι πως η εργασία αυτή χαρτογραφεί και τα δύο πεδία συνεισφορών μεταξύ ΑΙ και δοκιμής λογισμικού καθώς και αναδεικνύει τα σημεία εστίασης της μελλοντικής έρευνας. Μπορεί κατά συνέπεια να αποτελέσει οδηγό για μελλοντικές μελέτες, υποδεικνύοντας εκείνους τους τομείς για τους οποίους υπάρχει αναγκαιότητα περαιτέρω διεξαγωγής πρωτογενούς έρευνας με σκοπό να οδηγήσει στην εξέλιξη και των δυο συνεισφοράς.

6.2 ΕΜΠΕΙΡΙΑ

Στο πλαίσιο εκπόνησης της συγκεκριμένης μελέτης, αναπτύχθηκαν δεξιότητες διεξαγωγής βιβλιογραφικής ανασκόπησης. Επιπλέον, ήταν φυσικό επακόλουθο η εμβάθυνση στους τομείς της τεχνητής νοημοσύνης και της δοκιμής λογισμικού καθώς και των συνεργειών τους, καθώς εμπλουτίστηκε το εννοιολογικό υπόβαθρο από τη μελέτη της σχετικής βιβλιογραφίας.

6.3 ΑΝΑΛΥΣΗ OPPORTUNITIES - THREATS

Παρατίθενται παρακάτω πίνακες – αναλύσεις ευκαιριών-απειλών για την τεχνητή νοημοσύνη στη δοκιμή λογισμικού και τη δοκιμή λογισμικού στην τεχνητή νοημοσύνη.

Πίνακας 7. Ανάλυση ευκαιριών & απειλών της τεχνητής νοημοσύνης στη δοκιμή λογισμικού

ΕΥΚΑΙΡΙΕΣ	ΑΠΕΙΛΕΣ
Εμπειρική τεκμηρίωση αποτελεσματικότητας προτεινόμενων μεθόδων στη σύγχρονη βιβλιογραφία σε ευρύτερα σύνολα δεδομένων και μετρικών	Υπερβολική εξάρτηση από μηχανισμούς τεχνητής νοημοσύνης
Μετάβαση σε μοντέλα βασισμένα σε πράκτορες για την προσθήκη δυνατοτήτων δυναμικού επανακαθορισμού της στρατηγικής των δοκιμών	Περιορισμένη πρόσβαση σε ποιοτικά δεδομένα εκπαίδευσης ή υψηλό κόστος πρόσβασης σε αυτά
Έρευνα για τη δυνατότητα ενσωμάτωσης μηχανισμών σαφούς καθορισμού εννοιολογικού πλαισίου και μηχανισμών επαλήθευσης των παραγόμενων αποτελεσμάτων σε LLMs (στο πρότυπο της μελέτης από τους Kumar et al., 2025).	Κίνδυνος overfitting στα δεδομένα εκπαίδευσης ML/DL μοντέλων (Braga et al., 2018, Dewangan et al., 2022)
Αξιοποίηση τεχνικών transfer learning για την εκπαίδευση συστημάτων τεχνητής νοημοσύνης εξαλείφοντας την ανάγκη για επιπλέον δεδομένα εκπαίδευσης	Εγγενείς αδυναμίες των LLMs στην παρούσα μορφή τους
Εφαρμογή data driven τεχνικών για την βελτίωση της απόδοσης μοντέλων DNN ως test oracles	
Συνδυαστική αξιοποίηση πολλαπλών data -driven μοντέλων DNN με στόχο να υποκαταστήσουν παραδοσιακούς μηχανισμούς test oracle	

Πίνακας 8. Ανάλυση ευκαιριών & απειλών δοκιμής λογισμικού στην τεχνητή νοημοσύνη

ΕΥΚΑΙΡΙΕΣ	ΑΠΕΙΛΕΣ
Διερεύνηση μεθόδων για τη συστηματική παραγωγή MRs που να είναι ανεξάρτητες, σημασιολογικές και αποτελεσματικές στην ανακάλυψη σφαλμάτων.	Απαντώνται συγκεκριμένα κοινά σύνολα δεδομένων που έχουν χρησιμοποιηθεί σε εύρος μελετών, γεγονός που πιθανώς υποβαθμίζει τη δυνατότητα γενίκευσης των αποτελεσμάτων καθώς τα σύνολα ενδέχεται να μην είναι τόσο αντιπροσωπευτικά ή πλήρη (MNIST, Fashion-MNIST, CIFAR-10, ImageNet).
Προσπάθεια δημιουργίας μηχανισμών φιλτραρίσματος MRs (προτεραιοποίηση, ανάλυση εξαρτήσεων, συσταδοποίηση, σύνθεση) ώστε να αποκλείονται εκείνες που οδηγούν σε περιττές δοκιμές.	Επειδή οι προσεγγίσεις κάλυψης υστερούν ευρύτερης εμπειρικής τεκμηρίωσης, τα αποτελέσματά τους λειτουργούν περισσότερο ως μετρικές εξερεύνησης της δομής των υπό δοκιμή μοντέλων και λιγότερο ως

	εγγυήσεις της λειτουργικής τους ορθότητας.
Διεξαγωγή μελετών με σκοπό την εμπειρική τεκμηρίωση της συσχέτισης των μετρικών κάλυψης που απαντώνται στη σύγχρονη βιβλιογραφία με τη λειτουργική ορθότητα των υπό δοκιμή μοντέλων.	
Σύγκριση coverage-based και MR προσεγγίσεων, για να διερευνηθεί αν η υπεροχή των τελευταίων (Srinivasan & Kanewala, 2024) αφορά συγκεκριμένες περιπτώσεις ή είναι γενική.	

Στο πεδίο της εφαρμογής της τεχνητής νοημοσύνης στη δοκιμή λογισμικού, οι κυριότερες απειλές αφορούν την περιορισμένη πρόσβαση σε ποιοτικά δεδομένα εκπαίδευσης⁵¹, ο κίνδυνος υπερβολικής προσαρμογής στα δεδομένα εκπαίδευσης⁵², η υπερβολική εξάρτηση σε μηχανισμούς τεχνητής νοημοσύνης για να αξιολογήσουν την αποτελεσματικότητα συστημάτων λογισμικού, η λειτουργία των οποίων μπορεί να βασίζεται επίσης στη χρήση τεχνητής νοημοσύνης (καθώς η υπερβολική εμπιστοσύνη χωρίς την παρουσία ανθρώπινου παράγοντα για αντιπαραβολή των αποτελεσμάτων των δοκιμών ενδέχεται να αποβεί προβληματική) και οι εγγενείς αδυναμίες των LLMs στην παρούσα μορφή/κατάστασή τους (π.χ. hallucinations), καθώς ήδη αποτελούν δημοφιλή μορφή τεχνητής νοημοσύνης και στο μέλλον ακόμη περισσότερες προσεγγίσεις μπορεί να στηριχθούν στη χρήση τους.

Οι πιο σημαντικές ερευνητικές κατευθύνσεις προς υιοθέτηση στο άμεσο μέλλον φαίνονται να είναι η αξιοποίηση τεχνικών transfer learning με σκοπό τη μείωση της ανάγκης για νέα δεδομένα εκπαίδευσης, η αντιμετώπιση των εγγενών αδυναμιών των LLMs με την εφαρμογή προσεγγίσεων στο πρότυπο της μελέτης των Kumar et al. (2025) και η εμπειρική τεκμηρίωση αποτελεσματικότητας προτεινόμενων μεθόδων στη σύγχρονη βιβλιογραφία σε ευρύτερα σύνολα δεδομένων και μετρικών, καθώς φαίνεται πως αποτελεί κοινό περιορισμό για το εύρος μεθόδων που μελετήθηκε. Μια επίσης χρήσιμη ερευνητική κατεύθυνση, είναι η

⁵¹ Στις περιπτώσεις που μελετήθηκαν, μπορεί ως δεδομένα εκπαίδευσης να νοούνται δεδομένα αλληλεπίδρασης χρηστών με τα υπό δοκιμή συστήματα, με αποτέλεσμα αν ένα σύστημα έχει αναπτυχθεί πρόσφατα, απλώς να μην υπάρχουν ακόμη διαθέσιμα δεδομένα εκπαίδευσης (A machine learning approach to generate test oracles, Braga et al., 2018),

⁵² Πιθανή ένδειξη είναι η επίτευξη ποσοστών accuracy που προσεγγίζει το 100%, A machine learning approach to generate test oracles, Braga et al., 2018, Code Smell Detection Using Ensemble Machine Learning Algorithms, Dewangan et al., 2022)

αξιοποίηση data-centric τεχνικών, με σκοπό τη βελτίωση της απόδοσης DNN ως test oracles, καθώς αυτή τους η χρήση είναι κοινή σε πολλές προσεγγίσεις.

Στο πεδίο της εφαρμογής μεθόδων και τεχνικών δοκιμής λογισμικού στην τεχνητή νοημοσύνη, οι κυριότερες απειλές αφορούν τη χρήση συγκεκριμένων συνόλων δεδομένων σε εύρος μελετών, γεγονός που πιθανώς υποβαθμίζει τη δυνατότητα γενίκευσης των αποτελεσμάτων, και την έλλειψη επαρκούς εμπειρικής τεκμηρίωσης της χρήσης των coverage-based μετρικών.

Οι πιο σημαντικές ερευνητικές κατευθύνσεις που πρέπει να υιοθετηθούν στο άμεσο μέλλον, αφορούν τη διεξαγωγή μελετών με σκοπό την εμπειρική τεκμηρίωση της συσχέτισης των μετρικών κάλυψης που απαντώνται στη σύγχρονη βιβλιογραφία με τη λειτουργική ορθότητα των υπό δοκιμή μοντέλων αλλά και τη σύγκριση coverage-based και metamorphic relations προσεγγίσεων, για να αναδειχθεί αν οι τελευταίες υπερέχουν των πρώτων (όπως προκύπτει από τη μελέτη των Srinivasan & Kanewala (2024) σε κάθε περίπτωση ή μόνο σε συγκεκριμένες περιπτώσεις. Ακριβώς επειδή οι προσεγγίσεις που βασίζονται σε MRs φαίνονται πολλά υποσχόμενες, μια ερευνητική κατεύθυνση που θα πρέπει να υιοθετηθεί είναι η προσπάθεια βελτιστοποίησης των προσεγγίσεων αυτών. Αυτή η προσπάθεια μπορεί να περιλαμβάνει τη διερεύνηση μεθόδων για τη συστηματική παραγωγή MRs που να είναι ανεξάρτητες, σημασιολογικές και αποτελεσματικές στην ανακάλυψη σφαλμάτων αλλά και για το «φιλτράρισμα» των MRs (προτεραιοποίηση, ανάλυση εξαρτήσεων για την αφαίρεση περιττών MRs, συσταδοποίηση και επιλογή μιας από κάθε συστάδα, σύνθεση επιμέρους MRs). Μια επίσης χρήσιμη ερευνητική κατεύθυνση, είναι η ανάπτυξη μεθόδων και εργαλείων για την αποτελεσματική δοκιμή και αξιολόγηση συστημάτων DNN ως test oracles.

Τέλος, ένα ίσως κοινό ερευνητικό κενό, είναι πως μέχρι στιγμής, δεν γίνεται εκμεταλλεύσιμη στο έπακρο η δυνατότητα αλληλοσυμπλήρωσης AI και software testing, με αποτέλεσμα οι προτεινόμενες προσεγγίσεις να είναι κυρίως διαγνωστικού και όχι διορθωτικού χαρακτήρα. Οι μελέτες των Braga et al. (2018), Yang et al. (2025), Romdhana et al. (2022) και Liu et al. (2025) υποδεικνύουν μία ερευνητική κατεύθυνση, όπου μπορούν να αξιοποιηθούν επαναληπτικές τεχνικές (regression testing, coverage-guided fuzzing) σε συνδυασμό με συστήματα agents, με σκοπό οι τελικές προσεγγίσεις που θα προκύπτουν να μην περιορίζονται στη διάγνωση αστοχιών, αλλά να επεκτείνονται και στην επιδιόρθωση αυτών, εφαρμόζοντας

έτσι στην πράξη και στο έπακρο την συνέργεια (interplay) μεταξύ Τεχνητής Νοημοσύνης και Δοκιμής Λογισμικού.

7. ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Awad, A. I., Fouda, M. M., Khashaba, M. M., Mohamed, E. R., & Hosny, K. M. (2023). Utilization of mobile edge computing on the Internet of Medical Things: A survey. *ICT Express*, 9(3), 473–485. <https://doi.org/10.1016/j.ict.2022.05.006>
2. Baqar, M., & Khanda, R. (2024). The future of software testing: AI-powered test case generation and validation. *arXiv*. <https://doi.org/10.48550/arXiv.2409.05808>
3. Barr, E. T., Harman, M., McMin, P., Shahbaz, M., & Yoo, S. (2015). The oracle problem in software testing: A survey. *IEEE Transactions on Software Engineering*, 41(5), 507–525. <https://doi.org/10.1109/TSE.2014.2372785>
4. Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer*, 21(5), 61–72. <https://doi.org/10.1109/2.59>
5. Borah, S., Aliliele, K. C., Rakshit, S., & Vajjhala, N. R. (2022). Applications of artificial intelligence in software testing. In P. K. Mallick, A. K. Bhoi, P. Barsocchi, & V. H. C. de Albuquerque (Eds.), *Cognitive informatics and soft computing* (Lecture Notes in Networks and Systems, Vol. 375, pp. 727–736). Springer. https://doi.org/10.1007/978-981-16-8763-1_60
6. Braga, R., Santos Neto, P., Rabêlo, R., Santiago, J., & Souza, M. (2018). A machine learning approach to generate test oracles. In *Proceedings of the XXXII Brazilian Symposium on Software Engineering (SBES 2018)* (pp. 142–151). Association for Computing Machinery. <https://doi.org/10.1145/3266237.3266273>
7. Chen, T. Y., Kuo, F.-C., Liu, H., Poon, P.-L., Towey, D., Tse, T. H., & Zhou, Z. Q. (2018). Metamorphic testing: A review of challenges and opportunities. *ACM Computing Surveys*, 51(1), Article 4. <https://doi.org/10.1145/3143561>
8. Chen, Y., Hu, Z., Zhi, C., Han, J., Deng, S., & Yin, J. (2024). ChatUniTest: A framework for LLM-based test generation. In *Companion proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering* (pp. 572–576). Association for Computing Machinery. <https://doi.org/10.48550/arXiv.2305.04764>
9. De Moura, L., & Bjørner, N. (2008). Z3: An efficient SMT solver. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)* (pp. 337–340). https://doi.org/10.1007/978-3-540-78800-3_24

10. Dewangan, S., Rao, R. S., Mishra, A., & Gupta, M. (2022). Code smell detection using ensemble machine learning algorithms. *Applied Sciences*, 12(20), Article 10321. <https://doi.org/10.3390/app122010321>
11. Fontes, A., & Gay, G. (2021). Using machine learning to generate test oracles: A systematic literature review. In *Proceedings of the 1st International Workshop on Test Oracles (TORACLE '21)* (pp. 1–10). Association for Computing Machinery. <https://doi.org/10.1145/3472675.3473974>
12. Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3, 1289–1305.
13. Fraser, G., & Arcuri, A. (2011). EvoSuite: Automatic test suite generation for object-oriented software. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering* (pp. 416–419). Association for Computing Machinery. <https://doi.org/10.1145/2025113.2025179>
14. Just, R., Jalali, D., & Ernst, M. D. (2014). Defects4J: A database of existing faults to enable controlled testing studies for Java programs. In *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA)* (pp. 437–440). ACM. <https://doi.org/10.1145/2610384.2628055>
15. Kumar, S., Nitin, & Yadav, M. (2023). Finite state GUI testing with test case prioritization using Z-BES and GK-GRU. *Applied Sciences*, 13(19), Article 10569. <https://doi.org/10.3390/app131910569>
16. Liu, W., Luo, S., Pan, L., & Zhang, Z. (2025). DeepCNP: An efficient white-box testing of deep neural networks by aligning critical neuron paths. *Information and Software Technology*, 179, Article 107640. <https://doi.org/10.1016/j.infsof.2024.107640>
17. Ma, L., Juefei-Xu, F., Sun, J., Chen, C., Su, T., Li, L., Liu, Y., Zhao, J., & Wang, Y. (2018). **DeepGauge: Multi-granularity testing criteria for deep learning systems.** In *Proceedings of the 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2018)* (pp. 120–131). IEEE. <https://doi.org/10.1145/3238147.3238202>
18. Malinowski, M., & Fritz, M. (2014). A multi-world approach to question answering about real-world scenes based on uncertain input. In *Advances in Neural Information Processing Systems (NeurIPS)* (Vol. 27, pp. 1682–1690).

19. Mao, K., Harman, M., & Jia, Y. (2016). Sapienz: Multi-objective automated testing for Android applications. In *Proceedings of the 25th International Symposium on Software Testing and Analysis (ISSTA)* (pp. 94-105). <https://doi.org/10.1145/2931037.2931054>
20. Mishra, A., & Alzoubi, Y. I. (2023). Structured software development versus agile software development: A comparative analysis. *International Journal of Systems Assurance Engineering and Management*, 14(4), 1504–1522. <https://doi.org/10.1007/s13198-023-01958-5>
21. Mo, R., Cai, Y., Kazman, R., & Xiao, L. (2016). Hotspot patterns: The formal definition and automatic detection of architecture smells. In *Proceedings of the IEEE/IFIP Working Conference on Software Architecture (WICSA)*. IEEE. <https://doi.org/10.1109/WICSA.2015.12>
22. Odena, A., Olsson, C., Andersen, D. G., & Goodfellow, I. (2019). TensorFuzz: Debugging neural networks with coverage-guided fuzzing. In *Proceedings of the 36th International Conference on Machine Learning* (Vol. 97, pp. 4901–4911). PMLR.
23. Ouédraogo, W. C., Plein, L., Kaboré, K., Habib, A., Klein, J., Lo, D., & Bissyandé, T. F. (2025). Enriching automatic test case generation by extracting relevant test inputs from bug reports. *Empirical Software Engineering*, 30, Article 85. <https://doi.org/10.1007/s10664-025-10635-z>
24. Pei, K., Cao, Y., Yang, J., & Jana, S. (2019). DeepXplore: Automated whitebox testing of deep learning systems. *Communications of the ACM*, 62(11), 137–145. <https://doi.org/10.1145/3361566>
25. Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 2383–2392). Association for Computational Linguistics. <https://doi.org/10.18653/v1/D16-1264>
26. Riccio, V., Jahangirova, G., Stocco, A., Humbatova, N., Weiss, M., & Tonella, P. (2020). Testing machine learning-based systems: A systematic mapping. *Empirical Software Engineering*, 25, 5193–5254. <https://doi.org/10.1007/s10664-020-09881-0>
27. Romdhana, A., Merlo, A., Ceccato, M., & Tonella, P. (2022). Deep reinforcement learning for black-box testing of Android apps. *ACM Transactions on Software Engineering and Methodology*, 31(4), Article 65. <https://doi.org/10.1145/3502868>

28. Russell, S., & Norvig, P. (2020). *Artificial intelligence: A modern approach* (4th ed.). Pearson.
29. Sabagh, A., Keyvanpour, M. R., & Parsa, S. (2020). FCCI: A fuzzy expert system for identifying coincidental correct test cases. *Journal of Systems and Software*, 168, Article 110635. <https://doi.org/10.1016/j.jss.2020.110635>
30. Shrivathsan, A. D., Ravichandran, K. S., Krishankumar, R., Sangeetha, V., Kar, S., Ziemba, P., & Jankowski, J. (2019). Novel fuzzy clustering methods for test case prioritization in software projects. *Symmetry*, 11(11), Article 1400. <https://doi.org/10.3390/sym11111400>
31. Soares, L. B., FitzGerald, N., Ling, J., & Kwiatkowski, T. (2019). Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)* (pp. 2895–2905). <https://doi.org/10.18653/v1/P19-1279>
32. Sommerville, I. (2016). *Software engineering* (10th ed.). Pearson.
33. Srinivasan, M., & Kanewala, U. (2024). Improving early fault detection in machine learning systems using data diversity-driven metamorphic relation prioritization. *Electronics*, 13(17), Article 3380. <https://doi.org/10.3390/electronics13173380>
34. Stocks, P., & Carrington, D. (1996). A framework for specification-based testing. *IEEE Transactions on Software Engineering*, 22(11), 777–793. <https://doi.org/10.1109/32.553698>
35. Sun, Y., Ding, Z., Huang, H., Zou, S., & Jiang, M. (2023). Metamorphic testing of relation extraction models. *Algorithms*, 16(2), Article 102. <https://doi.org/10.3390/a16020102>
36. Sun, Y., Huang, X., Kroening, D., Sharp, J., Hill, M., & Ashmore, R. (2019). Structural test coverage criteria for deep neural networks. *ACM Transactions on Embedded Computing Systems*, 18(5s), Article 94. <https://doi.org/10.1145/3358233>
37. Wang, R., Sato, Y., & Liu, S. (2021). Mutated specification-based test data generation with a genetic algorithm. *Mathematics*, 9(4), Article 331. <https://doi.org/10.3390/math9040331>
38. Xie, X., Ma, L., Juefei-Xu, F., Xue, M., Chen, H., Liu, Y., Zhao, J., Li, B., Yin, J., & See, S. (2019). DeepHunter: A coverage-guided fuzz testing framework for deep neural

- networks. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '19)* (pp. 146–157). Association for Computing Machinery. <https://doi.org/10.1145/3293882.3330579>
39. Yamada, I., Asai, A., Shindo, H., Takeda, H., & Matsumoto, Y. (2020). LUKE: Deep contextualized entity representations with entity-aware self-attention. *arXiv*. <https://arxiv.org/abs/2010.01057>
40. Yang, R., Xu, X., & Wang, R. (2025). TestLoter: A logic-driven framework for automated unit test generation and error repair using large language models. *Journal of Computer Languages*, 84, Article 101348. <https://doi.org/10.1016/j.cola.2025.101348>
41. Yas, Q. M., Alazzawi, A., & Rahmatullah, B. (2023). A comprehensive review of software development life cycle methodologies: Pros, cons, and future directions. *Iraqi Journal for Computer Science and Mathematics*, 4(4), Article 14. <https://doi.org/10.52866/ijcsm.2023.04.04.014>
42. Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8(3), 338–353. [https://doi.org/10.1016/S0019-9958\(65\)90241-X](https://doi.org/10.1016/S0019-9958(65)90241-X)
43. Zavadskas, E. K., Turskis, Z., Antucheviciene, J., & Zakarevicius, A. (2012). Optimization of weighted aggregated sum product assessment. *Engineering Economics*, 23(2), 122–130. doi:[10.5755/j01.eee.122.6.1810](https://doi.org/10.5755/j01.eee.122.6.1810)
44. Zhou, W., & Chen, M. (2021). Learning from noisy labels for entity-centric information extraction. *arXiv*. <https://arxiv.org/abs/2104.08656>
45. Zubair, K., Farooq, S. U., & Khan, D. A. (2022). Artificial intelligence in software testing: Impact, problems, challenges and prospect. *arXiv*. <https://doi.org/10.48550/arXiv.2201.05371>