ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Επισκόπηση μεθόδων και τεχνικών για την ανάπτυξη εφαρμογών ιστού

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της/του

Γανίτη Παρίση

Επιβλέπων: Κρητικός Κυριάκος

Μέλη εξεταστικής επιτροπής: Κοκολάκης Σπύρος , Χαραλαμπίδης Γιάννης

Σάμος, Οκτώβριος 2026

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πρόλογος και ευχαριστίες

Αντικείμενό της είναι η συστηματοποιημένη επισκόπηση των μεθόδων, τεχνικών και εργαλείων που έχουν προταθεί για την ανάπτυξη εφαρμογών ιστού, με έμφαση σε προσεγγίσεις που καλύπτουν περισσότερα του ενός στάδια του κύκλου ζωής του λογισμικού.

Η επιλογή του θέματος προέκυψε από τη διαπίστωση ενός χάσματος που συναντά κανείς τόσο στη βιβλιογραφία όσο και στην καθημερινή πράξη: ενώ οι μέθοδοι, οι τεχνικές και τα εργαλεία ανάπτυξης εφαρμογών ιστού είναι πολυάριθμα και συχνά εξαιρετικά επεξεργασμένα, καθένα από αυτά αντιμετωπίζει το πρόβλημα από τη σκοπιά ενός περιορισμένου τμήματος του κύκλου ζωής. Η ανάγκη για μια ενοποιημένη θεώρηση, που να τα αντιπαραβάλλει με κοινά κριτήρια, αποτέλεσε το κίνητρο της εργασίας αυτής.

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή κ. Κυριάκο Κρητικό για την καθοδήγηση, την υπομονή, τις εύστοχες παρατηρήσεις και τη διαρκή υποστήριξή του σε όλη τη διάρκεια της εκπόνησης. Οι επισημάνσεις του βελτίωσαν ουσιωδώς τη μεθοδολογική αυστηρότητα και τη σαφήνεια του κειμένου. Ευχαριστώ επίσης τα μέλη της εξεταστικής επιτροπής, κ. Σπύρο Κοκολάκη και κ. Γιάννη Χαραλαμπίδη, για τον χρόνο που αφιέρωσαν στην αξιολόγηση της εργασίας.

Τέλος, οφείλω ένα ευχαριστώ στην οικογένειά μου και στους ανθρώπους που με στήριξαν σε αυτή τη διαδρομή.

© 2026
του
ΓΑΝΙΤΗ ΠΑΡΙΣΗ
Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων
ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πίνακας περιεχομένων

1 Εισαγωγή....1
1.1 Εφαρμογές Ιστού: Ορισμός και Σημασία....1
1.2 Υφιστάμενες Επισκοπήσεις και τα Μειονεκτήματά τους....2
1.2.1 Επισκοπήσεις Δοκιμών Εφαρμογών Ιστού....2
1.2.2 Επισκοπήσεις Μεθοδολογιών Web Engineering....3
1.2.3 Επισκοπήσεις Τεχνολογιών Υλοποίησης και Σύγχρονων Πρακτικών Ανάπτυξης....3
1.2.4 Επισκοπήσεις Ασφάλειας Εφαρμογών Ιστού....4
1.2.5 Σύνοψη των Μειονεκτημάτων....4
1.3 Η Συμβολή της Παρούσας Εργασίας....4
1.4 Δομή της Εργασίας....5
2. Υπόβαθρο....6
2.1 Ορισμός και Χαρακτηριστικά Εφαρμογών Ιστού....6
2.1.1 Ορισμός....6
2.1.2 Διαφορές από τις Παραδοσιακές Εφαρμογές....6
2.1.3 Κύρια Χαρακτηριστικά....7
2.1.4 Το Μοντέλο Ποιότητας ISO/IEC 25010....7
2.2 Αρχιτεκτονική Εφαρμογών Ιστού....8
2.2.1 Τριεπίπεδη Αρχιτεκτονική....8
2.2.2 Σύγχρονες Αρχιτεκτονικές....8
2.2.3 REST και Αρχιτεκτονικές Οδηγούμενες από API....8
2.2.4 Στρατηγικές Απόδοσης Περιεχομένου και JAMstack....9
2.3 Στάδια Ανάπτυξης Εφαρμογών Ιστού....10
2.3.1 Ανάλυση Απαιτήσεων και Σχεδίαση....10
2.3.2 Υλοποίηση....11
2.3.3 Δοκιμές και Διασφάλιση Ποιότητας....11
2.3.4 Διάταξη (Deployment)....12
2.3.5 Συντήρηση και Λειτουργία....13
2.3.6 Μοντέλα Οργάνωσης των Σταδίων....14
2.4 Πεδία Εφαρμογής....14
2.5 Ο Ρόλος του Υπολογιστικού Νέφους....15
2.5.1 Θεμελιώδη Χαρακτηριστικά και Μοντέλα Ανάπτυξης....15
2.6 Μέθοδοι, Τεχνικές και Εργαλεία Ανάπτυξης....16
2.6.1 Παραδοσιακές και Agile Οικογένειες Μεθόδων....16
2.6.2 Model-Driven Web Engineering....17
2.6.3 DevOps και Συνεχής Ενσωμάτωση / Παράδοση....18
2.6.4 Test-Driven Development και Αυτοματοποιημένες Δοκιμές....19
2.6.5 Αρχιτεκτονική Microservices....19
2.6.6 Αρχιτεκτονικές Προσανατολισμένες σε Υπηρεσίες: SOA, REST και API-Driven....20
2.6.7 Γλώσσες Προγραμματισμού....21
2.6.8 Frameworks....23
2.6.8.1 Front-End Frameworks....23
2.6.8.2 Back-End Frameworks....23
2.6.9 Συστήματα Διαχείρισης Περιεχομένου και Ασφάλεια....24

2.6.10 Συνοπτική Αντιστοίχιση Μεθόδων και Σταδίων..... 25
3. Μεθοδολογία Επισκόπησης..... 28
3.1 Ερευνητικές Ερωτήσεις..... 28
3.2 Στρατηγική Αναζήτησης..... 30
3.2.1 Πηγές Βιβλιογραφίας..... 30
3.2.2 Μοτίβα Επερώτησης..... 31
3.2.3 Χρονικά και Γλωσσικά Φίλτρα..... 32
3.3 Κριτήρια Επιλογής και Ποιότητας..... 32
3.3.1 Κριτήρια Επιλογής..... 32
3.3.2 Αξιολόγηση Ποιότητας..... 33
3.4 Το Τελικό Σύνολο Άρθρων..... 34
3.5 Στατιστικά Στοιχεία του Τελικού Συνόλου..... 36
3.5.1 Ποσοστό Διατήρησης..... 36
3.5.2 Αναλογία ανά Τύπο Άρθρου..... 37
3.5.3 Αναλογία ανά Εκδότη..... 38
3.5.4 Χρονική Κατανομή και Κάλυψη Σταδίων..... 39
3.6 Εξαγωγή Δεδομένων..... 40
3.7 Αντιπροσωπευτικά Άρθρα προς Ανάλυση Βάθους..... 40
3.8 Απειλές Εγκυρότητας..... 41
4. Ανάλυση..... 43
4.1 Κατηγοριοποίηση των Επιλεγμένων Άρθρων (RQ1)..... 43
4.2 Ανάλυση Αντιπροσωπευτικών Άρθρων (RQ1, RQ7)..... 45
4.2.1 Model-Driven Web Engineering..... 46
4.2.2 Agile & Scrum..... 47
4.2.3 DevOps & CI/CD..... 49
4.2.4 Microservices..... 50
4.2.5 TDD & Automated Testing..... 52
4.2.6 Frameworks, Γλώσσες & Ασφάλεια..... 53
4.3 Κριτήρια Σύγκρισης ανά Στάδιο SDLC (RQ2)..... 55
4.4 Αποτελέσματα της Σύγκρισης (RQ2–RQ5)..... 58
4.5 Ανάλυση των Αποτελεσμάτων (RQ2–RQ8)..... 60
4.5.1 Γενική Εικόνα της Κάλυψης SDLC..... 60
4.5.2 Κάλυψη ανά Στάδιο και η Θεμελιώδης Πόλωση..... 61
4.5.3 Η Ασφάλεια ως Εγκάρσιο Ζήτημα (RQ3)..... 62
4.5.4 Σχέση με το Νέφος και τα Σύγχρονα Εργαλεία (RQ4, RQ8)..... 64
4.5.5 Frameworks, Γλώσσες και Εργαλεία (RQ5)..... 64
4.5.6 Εμπειρική Αξιολόγηση (RQ7)..... 66
4.5.7 Χρονική Εξέλιξη των Μεθόδων (RQ8)..... 67
4.5.8 Σύνοψη: Τέσσερα Κεντρικά Ευρήματα..... 67
5. Ερευνητικά Κενά και Μελλοντικές Κατευθύνσεις..... 69
5.1 RG1 - Έλλειψη Ολιστικών Οικογενειών Μεθόδων που Καλύπτουν το Σύνολο του SDLC..... 69
5.1.1 Κατευθύνσεις Αντιμετώπισης..... 70
5.2 RG2 - Παραμέληση της Ασφάλειας ως Εγκάρσιου Ζητήματος..... 70
5.2.1 Κατευθύνσεις Αντιμετώπισης..... 71

5.3 RG3 - Χάσμα Θεωρίας-Πράξης: Έλλειψη Κοινών Βιομηχανικών Πρακτικών στην Έρευνα..... 71
5.3.1 Κατευθύνσεις Αντιμετώπισης.....72
5.4 RG4 - Περιορισμένη Ενσωμάτωση της Συντήρησης και της Παρατηρησιμότητας στις Μεθόδους Ανάπτυξης..... 73
5.4.1 Κατευθύνσεις Αντιμετώπισης.....73
5.5 RG5 - Ενσωμάτωση της Τεχνητής Νοημοσύνης στον Κύκλο Ζωής..... 74
5.5.1 Κατευθύνσεις Αντιμετώπισης.....74
5.6 RG6 - Έλλειψη Εμπειρικής Επαλήθευσης σε Βιομηχανικά Περιβάλλοντα..... 75
5.6.1 Κατευθύνσεις Αντιμετώπισης.....75
5.7 RG7 - Βιωσιμότητα και Green Software Engineering για Εφαρμογές Ιστού.....76
5.7.1 Κατευθύνσεις Αντιμετώπισης.....77
5.8 Συνοπτικός Πίνακας Κενών, Κατευθύνσεων και Ορίζοντα Υλοποίησης.....77
6. Συμπεράσματα..... 81
6.1 Κύρια Συνεισφορά της Εργασίας.....81
6.2 Κύρια Αποτελέσματα της Ανάλυσης..... 82
6.2.1 Τέσσερα Κεντρικά Ευρήματα.....82
6.2.2 Απαντήσεις στις Ερευνητικές Ερωτήσεις.....82
6.3 Εμπειρίες που Αποκομίστηκαν..... 84
6.4 Σημαντικές Ερευνητικές Κατευθύνσεις για το Άμεσο Μέλλον.....85
6.4.1 Ολιστικές Υβριδικές Μέθοδοι Ανάπτυξης..... 85
6.4.2 Security-by-Design ως Εγκάρσια Αρχή..... 85
6.4.3 Αξιολόγηση της AI-Ενισχυμένης Ανάπτυξης..... 85
6.4.4 Εμπειρική Έρευνα για CMS και Low-Code.....85
6.5 Κλείσιμο.....85
Βιβλιογραφία.....87
Παράρτημα Α - Πλήρης Κατάλογος Άρθρων (Corpus).....91

Λίστα Σχημάτων

Σχήμα 3.1: Ροόγραμμα της διαδικασίας επιλογής άρθρων.....35
Σχήμα 3.2: Χρονική κατανομή των 27 άρθρων.....40
Σχήμα 4.1: Ολιστική κάλυψη SDLC ανά οικογένεια μεθόδων.....61
Σχήμα 4.2: Κάλυψη του κριτηρίου ασφάλειας (C12) ανά οικογένεια μεθόδων.....63

Λίστα Πινάκων

Πίνακας 2.1: Συγκριτική επισκόπηση κύριων γλωσσών ανάπτυξης....22
Πίνακας 2.2: Συγκριτική επισκόπηση κύριων front-end frameworks....23
Πίνακας 2.3: Συγκριτική επισκόπηση κύριων back-end frameworks....24
Πίνακας 2.4: Ενδεικτική αντιστοίχιση μεθόδων και τεχνικών με στάδια SDLC....26
Πίνακας 3.1: Οι οκτώ ερευνητικές ερωτήσεις....29
Πίνακας 3.2: Ενδεικτικά μοτίβα επερώτησης ανά ερευνητική ερώτηση....31
Πίνακας 3.3: Κατανομή του τελικού συνόλου ανά οικογένεια μεθόδων....35
Πίνακας 3.4: Αναλογία άρθρων ανά τύπο δημοσίευσης....37
Πίνακας 3.5: Αναλογία άρθρων ανά εκδότη....38
Πίνακας 3.6: Κατανομή των άρθρων ως προς τον αριθμό σταδίων SDLC....39
Πίνακας 4.1: Κατανομή των 27 άρθρων ανά οικογένεια μεθόδων και υποκατηγορία....44
Πίνακας 4.2: Τα είκοσι κριτήρια σύγκρισης ανά στάδιο SDLC....56
Πίνακας 4.3: Σκορ των δώδεκα αντιπροσωπευτικών άρθρων ανά στάδιο....58
Πίνακας 4.4: Σκορ ανά οικογένεια μεθόδων και στάδιο SDLC....59
Πίνακας 4.5: Πίνακας 4.5: Κάλυψη του κριτηρίου ασφάλειας (C12) ανά οικογένεια μεθόδων....62
Πίνακας 4.6: Εργαλεία ανά οικογένεια μεθόδων....65
Πίνακας 5.1: Σύνοψη κενών, κατευθύνσεων και ορίζοντα υλοποίησης....77
Πίνακας 6.1: Σύνοψη των απαντήσεων στις οκτώ ερευνητικές ερωτήσεις....83
Πίνακας Α.1: Πλήρης κατάλογος των 27 άρθρων....91

Περίληψη

Η παρούσα διπλωματική εργασία αποτελεί μια δομημένη επισκόπηση των μεθόδων και τεχνικών που έχουν προταθεί για την ανάπτυξη εφαρμογών ιστού, με ιδιαίτερη έμφαση σε προσεγγίσεις που καλύπτουν περισσότερα του ενός στάδια του κύκλου ζωής ανάπτυξης λογισμικού (SDLC). Αφετηρία της αποτελεί μια διαπίστωση που επανέρχεται στη βιβλιογραφία: οι περισσότερες υφιστάμενες επισκοπήσεις κινούνται μονοδιάστατα - άλλες εστιάζουν αποκλειστικά στις τεχνικές δοκιμών, άλλες στις μεθοδολογίες σχεδίασης, άλλες στα εργαλεία υλοποίησης - χωρίς να προσφέρουν μια ολιστική, συγκριτική θεώρηση που να καλύπτει ταυτόχρονα ολόκληρο το φάσμα των σταδίων ανάπτυξης.

Με βάση αυτό το κενό, η εργασία ορίζει ένα σύνολο οκτώ ερευνητικών ερωτήσεων που εξετάζουν ποιες μέθοδοι και τεχνικές καλύπτουν πολλαπλά στάδια SDLC (Σχεδίαση, Υλοποίηση, Δοκιμές και Διασφάλιση Ποιότητας, Διάθεση, Συντήρηση και Λειτουργία), πώς αντιμετωπίζουν τα ζητήματα ασφάλειας, ποια είναι η σχέση τους με σύγχρονα εργαλεία και περιβάλλοντα νέφους, και ποια ερευνητικά κενά παραμένουν ανοιχτά. Για να απαντηθούν, συγκεντρώθηκε ένα σύνολο 27 αντιπροσωπευτικών, υψηλής ποιότητας δημοσιεύσεων μέσα από στοχευμένη αναζήτηση σε κύριες επιστημονικές βάσεις δεδομένων, οι οποίες αξιολογήθηκαν βάσει ενός δομημένου πλαισίου κριτηρίων οργανωμένου ανά στάδιο SDLC.

Τα κύρια ευρήματα δείχνουν ότι καμία από τις εξεταζόμενες οικογένειες μεθόδων - Μοντελοκεντρική Μηχανική Ιστού (Model-Driven Web Engineering), Ευέλικτες Μέθοδοι (Agile/Scrum), Ενοποίηση Ανάπτυξης & Λειτουργίας (DevOps και CI/CD), Μικρο-υπηρεσίες (Microservices), Ανάπτυξη Καθοδηγούμενη από Δοκιμές (Test-Driven Development) και Ανάπτυξη Βασισμένη σε Πλαίσια (Framework-Based Development) - δεν καλύπτει πλήρως και ισόρροπα το σύνολο των σταδίων. Ουσιαστικά αναδεικνύεται μια έντονη πόλωση: οι οικογένειες μεθόδων σχεδίασης υπολείπονται στη φάση της υλοποίησης, της διάταξης (deployment) και της συντήρησης, ενώ οι προσεγγίσεις DevOps και Microservices παραμελούν τη φάση της σχεδίασης. Ιδιαίτερα σημαντική είναι επιπλέον η διαπίστωση ότι η ασφάλεια αντιμετωπίζεται ως δευτερεύον ζήτημα από την πλειονότητα των προσεγγίσεων, παρά την κρίσιμη σημασία της για τις σύγχρονες εφαρμογές ιστού.

Η εργασία καταλήγει σε ένα σύνολο ερευνητικών κενών και υποσχόμενων κατευθύνσεων, με κυριότερες την ανάγκη για ολιστικά, υβριδικά πλαίσια που να γεφυρώνουν τη σχεδίαση με την ανάπτυξη, την ενσωμάτωση της τεχνητής νοημοσύνης στον κύκλο ζωής, και την υιοθέτηση προσεγγίσεων security-by-design από τα πρώτα κιόλας στάδια του SDLC.

Λέξεις Κλειδιά: εφαρμογές ιστού, μέθοδοι ανάπτυξης, τεχνικές ανάπτυξης, κύκλος ζωής λογισμικού (SDLC), Μοντελοκεντρική Μηχανική Ιστού (Model-Driven Web Engineering), Ευέλικτες Μέθοδοι (Agile), Ενοποίηση Ανάπτυξης & Λειτουργίας (DevOps), CI/CD, Μικρο-υπηρεσίες (Microservices), Ανάπτυξη Καθοδηγούμενη από Δοκιμές (Test-Driven Development), ασφάλεια, επισκόπηση βιβλιογραφίας, ερευνητικά κενά, ανάπτυξη βασισμένη σε πλαίσια (frameworks).

Abstract

This diploma thesis presents a structured survey of the methods and techniques proposed for the development of web applications, with particular emphasis on approaches that cover more than one stage of the Software Development Life Cycle (SDLC). Its starting point is a recurring observation in the relevant literature: most existing surveys are one-dimensional - some focus exclusively on testing techniques, others on design methodologies, others on implementation tools - without offering a holistic, comparative account spanning the full range of development stages.

Building on this gap, the thesis defines a set of eight research questions examining which methods and techniques span multiple SDLC stages, how they address related security issues, how they relate to modern tooling and cloud environments, and which research gaps remain open. To answer them, a curated corpus of 27 representative, high-quality publications was gathered through a targeted search of main scientific databases, assessed against a structured framework of criteria organised by SDLC stage.

The main findings show that none of the examined method families - Model-Driven Web Engineering, Agile/Scrum, DevOps and CI/CD, Microservices, Test-Driven Development and Framework-Based Development - covers the full SDLC in a complete and balanced way. In fact a clear polarisation emerges: design-oriented methods underperform in implementation, deployment and maintenance, whereas DevOps and Microservices approaches neglect design. Notably, security is treated as a secondary concern by most approaches, despite its critical importance for modern web applications.

The thesis concludes with a set of research gaps and promising directions, the most prominent being the need for holistic, hybrid frameworks bridging design and development, the integration of artificial intelligence into the development life cycle, and the adoption of security-by-design practices from the earliest SDLC stages.

Keywords: web applications, development methods, development techniques, Software Development Life Cycle (SDLC), Model-Driven Web Engineering, Agile, DevOps, CI/CD, Microservices, Test-Driven Development, security, literature survey, research gaps, Framework-Based Development

1 Εισαγωγή

1.1 Εφαρμογές Ιστού: Ορισμός και Σημασία

Οι εφαρμογές ιστού αποτελούν σήμερα έναν από τους θεμελιώδεις πυλώνες της ψηφιακής οικονομίας και της καθημερινής αλληλεπίδρασης των ανθρώπων με τις ψηφιακές υπηρεσίες (Li et al., 2024). Στην κλασική διατύπωση του Jazayeri (2007), μια εφαρμογή ιστού ορίζεται ως λογισμικό που διευκολύνει την αλληλεπίδραση του χρήστη μέσα από τον συνδυασμό στοιχείων front-end και back-end, προσπελαύνεται μέσω ενός προγράμματος περιήγησης και εκτελείται σε απομακρυσμένους διακομιστές, προσφέροντας ανεξαρτησία από την πλατφόρμα και κεντρική διαχείριση των ενημερώσεων (Doyle & Lopes, 2005). Η αρχιτεκτονική αυτή, όπου η λογική και τα δεδομένα βρίσκονται στον διακομιστή και η παρουσίαση φτάνει στον χρήστη μέσα από το πρόγραμμα περιήγησης, είναι που διαφοροποιεί ουσιωδώς τις εφαρμογές ιστού από το παραδοσιακό λογισμικό επιφάνειας εργασίας (desktop software).

Η κλίμακα και η σημασία αυτού του είδους εφαρμογών δύσκολα υπερεκτιμώνται. Από τις πλατφόρμες ηλεκτρονικού εμπορίου και τις τραπεζικές υπηρεσίες έως τα συστήματα υγειονομικής περίθαλψης και τα εργαλεία τηλεκπαίδευσης, οι εφαρμογές ιστού έχουν αναδιαμορφώσει ολόκληρους κλάδους της οικονομίας. Η ταχεία εξέλιξή τους - από τις απλές στατικές ιστοσελίδες σε Ευφυείς Διαδικτυακές Εφαρμογές (Rich Internet Applications), σε Εφαρμογές Μίας Σελίδας (Single Page Applications) και σε Προοδευτικές Εφαρμογές Ιστού (Progressive Web Apps) - έχει φέρει διαρκώς νέες τεχνολογικές προκλήσεις, καθώς κάθε νέα γενιά εφαρμογών ιστού αυξάνει τις απαιτήσεις σε διαδραστικότητα, απόδοση και εμπειρία χρήστη (Jazayeri, 2007). Ο αριθμός των ενεργών ιστότοπων παγκοσμίως μετριέται πλέον σε δισεκατομμύρια, γεγονός που υπογραμμίζει την κεντρική θέση που κατέχει ο Παγκόσμιος Ιστός (World Wide Web) στη σύγχρονη ψηφιακή υποδομή.

Η πορεία αυτή αντανακλά μια συνεχή τεχνολογική εξέλιξη που ξεκίνησε με την εμφάνιση του ιστού το 1991. Όπως περιγράφουν οι Li et al. (2024), τα πρώτα έγγραφα του ιστού ήταν απλές σελίδες HTML· η προσθήκη των Cascading Style Sheets (CSS) εμπλούτισε την παρουσίαση και τη διάταξη, ενώ η JavaScript εισήγαγε το δυναμικό και διαδραστικό περιεχόμενο. Η ανάπτυξη τεχνολογιών από την πλευρά του διακομιστή και των βάσεων δεδομένων επέτρεψε τη δημιουργία σύνθετων εφαρμογών, επεκτείνοντας τη λειτουργικότητα του ιστού πέρα από τη στατική παράδοση περιεχομένου. Καθοριστικός σταθμός υπήρξε η μετάβαση στο λεγόμενο Web 2.0 στα μέσα της δεκαετίας του 2000: όπως τεκμηρίωσε ο O'Reilly (2007) στο κείμενο που πρώτο επιχείρησε να ορίσει την έννοια, ο ιστός μετατράπηκε σε «πλατφόρμα», με λογισμικό που προσφέρεται ως διαρκώς ενημερωμένη υπηρεσία και με μια «αρχιτεκτονική συμμετοχής» που στηρίζεται στο περιεχόμενο που παράγουν οι ίδιοι οι χρήστες. Η τεχνική AJAX, που επέτρεψε την ανανέωση τμημάτων μιας σελίδας χωρίς πλήρη επαναφόρτωση, άνοιξε τον δρόμο για τις πλούσιες, διαδραστικές εφαρμογές και, στη συνέχεια, για τα Single Page Applications και τα Progressive Web Apps, τα οποία φέρνουν στον ιστό δυνατότητες που προηγουμένως ήταν διαθέσιμες μόνο σε εγγενώς αναπτυγμένες εφαρμογές (Biørn-Hansen, Majchrzak & Grønli, 2017).

Η ανάπτυξη εφαρμογών ιστού παρουσιάζει ιδιαιτερότητες που τη διαφοροποιούν από την παραδοσιακή ανάπτυξη λογισμικού. Καταρχάς, η κατανεμημένη αρχιτεκτονική πελάτη-διακομιστή, με την ασύγχρονη επικοινωνία μέσω του πρωτοκόλλου HTTP, εισάγει πολυπλοκότητα στη διαχείριση της κατάστασης και στον συγχρονισμό των δεδομένων ανάμεσα στον πελάτη και τον διακομιστή. Δεύτερον, η τεχνολογική ετερογένεια - με HTML, CSS και JavaScript στο front-end και γλώσσες, όπως η Python, η Java, η Node.js και η PHP στο back-end -

απαιτεί εξειδικευμένη γνώση σε πολλαπλά επίπεδα της τεχνολογικής στοίβας (Doğan, Betin-Can & Garousi, 2014). Και τρίτον, η δυναμική φύση των εφαρμογών, με τις συχνές ενημερώσεις περιεχομένου και τις απαιτήσεις πραγματικού χρόνου, καθιστά τη δοκιμή και τη συντήρησή τους ιδιαίτερα απαιτητικές, καθώς κάθε αλλαγή πρέπει να ελέγχεται και να διατίθεται χωρίς διακοπή της υπηρεσίας.

Η υποστήριξη της ανάπτυξης με συστηματικές μεθόδους και τεχνικές είναι κρίσιμη για δύο κυρίως λόγους. Αφενός, η ποιότητα και η ασφάλεια αυτών των συστημάτων επηρεάζουν άμεσα τους χρήστες τους, ευπάθειες, όπως το Cross-Site Scripting και το SQL Injection, παραμένουν στις κορυφαίες απειλές σύμφωνα με το OWASP Top 10 (2021), και μια μεμονωμένη παραβίαση τους μπορεί να έχει σοβαρές συνέπειες για εκατομμύρια χρήστες. Αφετέρου, η αποτελεσματικότητα στην ανάπτυξη και τη συντήρηση καθορίζει τον χρόνο διάθεσης στην αγορά, παράγοντα ζωτικής σημασίας για την ανταγωνιστικότητα των επιχειρήσεων. Η σημασία των σύγχρονων πρακτικών DevOps για την ταχύτητα και την αξιοπιστία της παράδοσης έχει τεκμηριωθεί εκτενώς, τόσο στο θεμελιώδες έργο των Forsgren, Humble και Kim (2018) όσο και στις ετήσιες εκθέσεις State of DevOps του οργανισμού DORA (DORA, 2024), από τις οποίες προκύπτουν τα εντυπωσιακά μεγέθη βελτίωσης στη συχνότητα της ανάπτυξης εφαρμογών και στον χρόνο αποκατάστασης που παρατηρούνται στους οργανισμούς υψηλής επίδοσης.

1.2 Υφιστάμενες Επισκοπήσεις και τα Μειονεκτήματά τους

Η επιστημονική κοινότητα έχει αφιερώσει σημαντικούς πόρους στην επισκόπηση της βιβλιογραφίας για την ανάπτυξη εφαρμογών ιστού. Οι υφιστάμενες εργασίες, ωστόσο, μοιράζονται ένα κοινό χαρακτηριστικό: εστιάζουν σε μεμονωμένες πτυχές ή φάσεις του κύκλου ζωής ανάπτυξης, αντί να προσφέρουν μια ολοκληρωμένη, συγκριτική θεώρηση που να καλύπτει ταυτόχρονα μεθόδους, τεχνικές και εργαλεία σε όλα τα στάδια. Στη συνέχεια εξετάζονται οι κύριες κατηγορίες αυτών των επισκοπήσεων και τα αντίστοιχα μειονεκτήματά τους, ώστε να αναδειχθεί με σαφήνεια το κενό που η παρούσα εργασία φιλοδοξεί να καλύψει.

1.2.1 Επισκοπήσεις Δοκιμών Εφαρμογών Ιστού

Ένα σημαντικό μέρος της βιβλιογραφίας εστιάζει αποκλειστικά στη δοκιμή εφαρμογών ιστού, αναγνωρίζοντας την κρισιμότητα της ποιότητας για αυτά τα συστήματα. Ήδη από το 2013, οι Garousi, Mesbah, Betin-Can και Mirshokraie (2013) πραγματοποίησαν την πρώτη συστηματική μελέτη χαρτογράφησης (systematic mapping study) του χώρου, αναλύοντας 79 μελέτες της περιόδου 2000–2011 και δομώντας το σώμα γνώσης γύρω από τις τεχνικές δοκιμής, τα εργαλεία και τα ανοιχτά προβλήματα. Η εργασία αυτή αποτελεί τον προπομπό της πληρέστερης συστηματικής ανασκόπησης των Doğan, Betin-Can και Garousi (2014), οι οποίοι ανέλυσαν 95 μελέτες παρέχοντας μια εκτενή κατηγοριοποίηση των τεχνικών και των εργαλείων δοκιμής. Και οι δύο εργασίες, παρά τη μεθοδολογική τους αρτιότητα, περιορίστηκαν αποκλειστικά στη φάση των δοκιμών, αφήνοντας εκτός τις προσεγγίσεις σχεδίασης, υλοποίησης και συντήρησης.

Πιο πρόσφατα, η επισκόπηση των Li et al. (2024), που ανέλυσε 314 πρωτογενείς μελέτες της περιόδου 2014–2023, ανέδειξε την αυξανόμενη χρήση τεχνικών μηχανικής μάθησης για την αυτοματοποίηση της δοκιμής και τον σημαντικό ρόλο της ασφάλειας, με το 29,94% των μελετών να εστιάζει στην ανίχνευση ευπαθειών. Παρ' όλα αυτά, και αυτή η εργασία παραμένει προσηλωμένη στη φάση της δοκιμής. Παράλληλα, η ανεξάρτητη και εξίσου πρόσφατη επισκόπηση των Kertusha, Assress, Duman και Arcuri (2025), που εξέτασε 214 άρθρα της περιόδου 2014–2024 μέσα από δεκατέσσερα ερευνητικά ερωτήματα, επιβεβαιώνει ότι το Selenium παραμένει το κυρίαρχο εργαλείο, ενώ η βιομηχανική υιοθέτηση των νεότερων τεχνικών δοκιμής παραμένει περιορισμένη. Το κοινό στοιχείο και των εργασιών αυτών είναι η

αναμφισβήτητη αξία τους για τη φάση των δοκιμών, αλλά και ο εγγενής περιορισμός τους σε αυτήν, χωρίς σύνδεση με τα υπόλοιπα στάδια του κύκλου ζωής.

1.2.2 Επισκοπήσεις Μεθοδολογιών Web Engineering

Μια δεύτερη κατηγορία επισκοπήσεων εστιάζει σε μεθόδους Model-Driven Web Engineering (MWDE), όπως οι OOHDM, UWE και WebML. Η εκτενές επισκόπηση (survey) των Schwinger et al. (2008) συνέκρινε επτά MWDE προσεγγίσεις για ubiquitous εφαρμογές ιστού, χρησιμοποιώντας έναν λεπτομερή κατάλογο άνω των τριάντα κριτηρίων αξιολόγησης. Οι Wakil και Jawawi (2015) παρείχαν αργότερα ένα εκτενές συστηματικό mapping study του χώρου, χαρτογραφώντας τη μεγάλη ποικιλία των μοντελοκεντρικών προσεγγίσεων και των εφαρμογών τους. Ειδικά για τη φάση των απαιτήσεων, οι Escalona και Koch (2004) είχαν ήδη επισημάνει ότι οι περισσότερες MDWE μέθοδοι εστιάζουν στη σχεδίαση δίνοντας ελάχιστη προσοχή στη μηχανική απαιτήσεων - μια πρώιμη διατύπωση του προβλήματος της ανισοβαρούς κάλυψης των σταδίων που αναδεικνύεται και στην παρούσα εργασία.

Πιο πρόσφατα, οι Linaje, Preciado και Morales-Chaparro (2019) συνέκριναν μεθοδολογίες MDWE συνδυάζοντας συστηματική ανασκόπηση με βιομηχανική έρευνα, αποκαλύπτοντας ότι στην πράξη οι εταιρείες προτιμούν υβριδικές ή εμπειρικές προσεγγίσεις έναντι των καθαρά ακαδημαϊκών μεθόδων - μια ισχυρή ένδειξη του χάσματος ανάμεσα στη θεωρία και την πράξη. Το κοινό μειονέκτημα όλων αυτών των εργασιών είναι η εστίασή τους κυρίως στη φάση της σχεδίασης και της μοντελοποίησης, χωρίς επαρκή κάλυψη της υλοποίησης, των δοκιμών και της διάταξης (deployment), καθώς και η συχνή απουσία εμπειρικής επαλήθευσης σε πραγματικά έργα.

Παράλληλα με τις κλασικές MDWE μεθόδους, μια σειρά επισκοπήσεων εστίασε στις σύγχρονες αρχιτεκτονικές τεχνοτροπίες. Χαρακτηριστική είναι η συστηματική μελέτη χαρτογράφησης των Di Francesco, Lago και Malavolta (2019), η οποία ανέλυσε 103 πρωτογενείς μελέτες για την αρχιτεκτονική εφαρμογών ιστού με microservices, αξιολογώντας τις τάσεις δημοσίευσης, την εστίαση της έρευνας και το δυναμικό βιομηχανικής υιοθέτησης. Το βασικό της εύρημα είναι ότι, παρότι οι σχεδιαστικές αρχές των microservices έχουν αναγνωριστεί, εξειδικευμένες λύσεις λείπουν ιδίως για τη διαχείριση της πολυπλοκότητας στη σχεδίαση, την παρακολούθηση και τη δοκιμή των συστημάτων αυτών. Ωστόσο, και εδώ η εστίαση περιορίζεται στην αρχιτεκτονική σχεδίαση, χωρίς ολοκληρωμένη σύνδεση με τις φάσεις της δοκιμής, της διάταξης και της συντήρησης.

1.2.3 Επισκοπήσεις Τεχνολογιών Υλοποίησης και Σύγχρονων Πρακτικών Ανάπτυξης

Μια τρίτη κατηγορία επισκοπήσεων εστιάζει στις τεχνολογίες υλοποίησης και στις σύγχρονες πρακτικές ανάπτυξης. Οι Doyle και Lopes (2005) πρόσφεραν μια εκτενή, για την εποχή της, επισκόπηση των τεχνολογιών ανάπτυξης ιστού· η εργασία αυτή, όμως, δεν καλύπτει σύγχρονα πλαίσια (frameworks), όπως τα React, Angular και Vue.js, ούτε τις cloud-native αρχιτεκτονικές και το containerization που κυριαρχούν σήμερα. Στον χώρο των πρακτικών, οι Leite, Rocha, Kon, Milojicic και Meirelles (2019) πραγματοποίησαν μια ευρέως αναφερόμενη εννοιολογική επισκόπηση του DevOps, χαρτογραφώντας τις έννοιες και τις προκλήσεις του σε επίπεδο κουλτούρας, αυτοματοποίησης και μέτρησης. Πιο πρόσφατα, οι El Aouni, Moumane, Idri και Najib (2024) εξέτασαν την ενσωμάτωση Agile, Cloud και DevOps, επισημαίνοντας ότι η νοοτροπία και οι ανθρώπινοι παράγοντες αποτελούν τους πρωταρχικούς παράγοντες επιτυχίας.

Παρά την αξία τους, οι εργασίες αυτές δεν εστιάζουν ειδικά στις εφαρμογές ιστού και στις μοναδικές τεχνικές τους ανάγκες. Επιπλέον, δεν παρέχουν συγκριτική ανάλυση ανάμεσα σε διαφορετικές μεθόδους και εργαλεία ανά στάδιο SDLC, ούτε αντιμετωπίζουν την ασφάλεια ως ένα εγκάρσιο ζήτημα που διαπερνά όλα τα στάδια, περιοριζόμενες συχνά σε μία μόνο διάσταση του προβλήματος.

1.2.4 Επισκοπήσεις Ασφάλειας Εφαρμογών Ιστού

Ιδιαίτερη μνεία αξίζει σε μια τέταρτη ομάδα επισκοπήσεων που πραγματεύεται την ασφάλεια των εφαρμογών ιστού - ένα ζήτημα κεντρικής σημασίας για την παρούσα εργασία. Οι Deepa και Thilagam (2016) πραγματοποίησαν μια εκτενή ανασκόπηση των προσεγγίσεων προστασίας από ευπάθειες έγχυσης (injection) και ευπάθειες επιχειρησιακής λογικής (business logic), με έμφαση στο SQL Injection και το Cross-Site Scripting ως τις κορυφαίες απειλές έγχυσης. Σε ευρύτερο πλαίσιο, οι Alazmi και De Leon (2022), σε συστηματική ανασκόπηση πρωτογενών μελετών για τους σαρωτές ευπαθειών εφαρμογών ιστού, καταλήγουν ότι η αποτελεσματικότητα των διαθέσιμων τεχνικών ανίχνευσης παραμένει ανομοιογενής και ότι δεν έχει καθιερωθεί τυποποιημένη τεχνική για την ανάπτυξη ασφαλών εφαρμογών ιστού, ενώ το OWASP Top 10 (OWASP, 2021) παραμένει το κυρίαρχο σημείο αναφοράς. Το κρίσιμο μειονέκτημα όλων αυτών των εργασιών, ωστόσο, είναι ότι αντιμετωπίζουν την ασφάλεια απομονωμένα - ως αντικείμενο εξειδικευμένης ανάλυσης ή ως στάδιο δοκιμών - και όχι ως εγκάρσια αρχή (security-by-design) ενσωματωμένη σε ολόκληρο τον κύκλο ζωής, ακριβώς το κενό που η παρούσα επισκόπηση αναδεικνύει.

1.2.5 Σύνοψη των Μειονεκτημάτων

Συνοψίζοντας, οι υφιστάμενες επισκοπήσεις παρουσιάζουν πέντε διακριτά μειονεκτήματα. Το πρώτο είναι το περιορισμένο εύρος: επικεντρώνονται σε μία ή δύο φάσεις του κύκλου ζωής - κυρίως στη δοκιμή ή στη σχεδίαση - χωρίς ολοκληρωμένη κάλυψη ολόκληρου του SDLC. Το δεύτερο είναι η απουσία συγκριτικής ανάλυσης: καμία υφιστάμενη επισκόπηση δεν εξετάζει ταυτόχρονα μεθόδους, τεχνικές και εργαλεία, αξιολογώντας τα με κοινά κριτήρια ανά στάδιο. Το τρίτο είναι το συχνά ξεπερασμένο περιεχόμενο, καθώς οι παλαιότερες επισκοπήσεις δεν καλύπτουν σύγχρονες τεχνολογίες, όπως το Docker, το Kubernetes, το serverless και τα AI-ενισχυμένα εργαλεία. Το τέταρτο είναι η παραμέληση της ασφάλειας, που αντιμετωπίζεται ως δευτερεύον ζήτημα - και μόνο στις επισκοπήσεις δοκιμών - αντί ως εγκάρσια ανάγκη. Και το πέμπτο είναι το διαρκές χάσμα θεωρίας και πράξης, με τις ακαδημαϊκές μεθοδολογίες να υιοθετούνται σπάνια ως έχουν στη βιομηχανία, η οποία προτιμά υβριδικές ή εμπειρικές προσεγγίσεις (Linaje et al., 2019). Τα πέντε αυτά μειονεκτήματα ορίζουν, ουσιαστικά, το ερευνητικό κενό που η παρούσα εργασία καλείται να καλύψει.

1.3 Η Συμβολή της Παρούσας Εργασίας

Η παρούσα εργασία έρχεται να καλύψει τα παραπάνω κενά, προσφέροντας μια ολοκληρωμένη επισκόπηση μεθόδων, τεχνικών και εργαλείων για την ανάπτυξη εφαρμογών ιστού, με αποκλειστική εστίαση σε προσεγγίσεις που καλύπτουν περισσότερα του ενός στάδια του κύκλου ζωής. Σε αντίθεση με τις προηγούμενες εργασίες, η συμβολή της είναι πενταπλή.

Πρώτον, ενοποιεί μεθόδους, τεχνικές και εργαλεία σε ένα ενιαίο αναλυτικό πλαίσιο, εξετάζοντας ταυτόχρονα οικογένειες μεθόδων ανάπτυξης, όπως το Agile, η MDWE και το DevOps, τεχνικές όπως το Test Driven Development (TDD), το CI/CD και τα microservices, και σύγχρονα εργαλεία/πλαίσια, όπως το Docker, το Kubernetes και το React. Δεύτερον, εφαρμόζει συγκριτική ανάλυση ανά στάδιο SDLC, αναπτύσσοντας συγκεκριμένα κριτήρια αξιολόγησης για κάθε στάδιο - σχεδίαση, υλοποίηση, δοκιμή, διάταξη και συντήρηση - που επιτρέπουν μια αντικειμενική και τεκμηριωμένη σύγκριση. Τρίτον, καλύπτει τις σύγχρονες τεχνολογίες και πρακτικές, ενσωματώνοντας τις εξελίξεις έως το 2025, περιλαμβανομένων των cloud-native αρχιτεκτονικών, του DevSecOps και της AI-ενισχυμένης ανάπτυξης. Τέταρτον, αντιμετωπίζει την ασφάλεια ως εγκάρσιο ζήτημα, εξετάζοντας πώς κάθε μέθοδος και τεχνική τη διαχειρίζεται σε κάθε στάδιο, με βάση το OWASP Top 10 (2021). Και πέμπτον, προσδιορίζει συστηματικά τα

ανοιχτά ερευνητικά κενά και προτείνει συγκεκριμένες κατευθύνσεις για την αντιμετώπιση τους, σχετιζόμενες με τη δημιουργία ολιστικών, ασφαλών και βιώσιμων πλαισίων ανάπτυξης.

Με τον τρόπο αυτό, η εργασία συνεισφέρει στην επιστημονική κοινότητα μια ενημερωμένη, ολοκληρωμένη επισκόπηση, χρήσιμη τόσο για ερευνητές που αναζητούν ανοιχτά προβλήματα όσο και για επαγγελματίες μηχανικούς λογισμικού και εκπαιδευτές του τομέα που χρειάζονται μια συνολική εικόνα του πεδίου.

1.4 Δομή της Εργασίας

Η υπόλοιπη δομή της εργασίας οργανώνεται ως εξής. Το Κεφάλαιο 2 παρέχει το απαραίτητο θεωρητικό υπόβαθρο: τον ορισμό και τα χαρακτηριστικά των εφαρμογών ιστού, τα αρχιτεκτονικά πρότυπα, τα στάδια του κύκλου ζωής, τα πεδία εφαρμογής, τον ρόλο του υπολογιστικού νέφους, καθώς και μια επισκόπηση των κύριων μεθόδων, τεχνικών και εργαλείων. Το Κεφάλαιο 3 παρουσιάζει τη μεθοδολογία της επισκόπησης - τις ερευνητικές ερωτήσεις, τις πηγές που διερευνήθηκαν, τα μοτίβα επερώτησης, τα κριτήρια επιλογής και αξιολόγησης, καθώς και τα στατιστικά στοιχεία του τελικού συνόλου. Το Κεφάλαιο 4 αποτελεί τον πυρήνα της εργασίας, με την κατηγοριοποίηση των άρθρων, την εις βάθος ανάλυση των αντιπροσωπευτικών εργασιών και τη συγκριτική αξιολόγηση. Τα είκοσι κριτήρια σύγκρισης παρατίθενται σε έναν ενιαίο πίνακα, ομαδοποιημένα ανά στάδιο του κύκλου ζωής, και τα αποτελέσματα της εφαρμογής τους σε δύο πίνακες: έναν ανά άρθρο και έναν ανά κατηγορία. Το Κεφάλαιο 5 εντοπίζει τα ερευνητικά κενά που προκύπτουν από την ανάλυση και προτείνει υποσχόμενες μελλοντικές κατευθύνσεις. Τέλος, το Κεφάλαιο 6 συνοψίζει τη συνεισφορά της εργασίας, τα κύρια αποτελέσματα και τις σημαντικότερες ερευνητικές κατευθύνσεις για το άμεσο μέλλον.

2. Υπόβαθρο

Το παρόν κεφάλαιο παρέχει το θεωρητικό υπόβαθρο που είναι απαραίτητο για την κατανόηση της ανάλυσης που ακολουθεί. Ξεκινά με τον ορισμό της εφαρμογής ιστού και τη διαφοροποίησή της από το παραδοσιακό λογισμικό, συνεχίζει με τα αρχιτεκτονικά πρότυπα και τα στάδια του κύκλου ζωής ανάπτυξης, εξετάζει τα πεδία εφαρμογής και τον ρόλο του υπολογιστικού νέφους, και ολοκληρώνεται με μια επισκόπηση των κύριων οικογενειών μεθόδων, τεχνικών και εργαλείων που χρησιμοποιούνται στον χώρο και που θα αποτελέσουν το αντικείμενο της συγκριτικής ανάλυσης του Κεφαλαίου 4.

2.1 Ορισμός και Χαρακτηριστικά Εφαρμογών Ιστού

2.1.1 Ορισμός

Μια εφαρμογή ιστού είναι λογισμικό που εκτελείται σε έναν ή περισσότερους διακομιστές ιστού και προσπελαύνεται από τον χρήστη μέσα από ένα πρόγραμμα περιήγησης, τον φυλλομετρητή (browser), μέσω του διαδικτύου (Jazayeri, 2007). Η λογική και τα δεδομένα βρίσκονται κατά κύριο λόγο στην πλευρά του διακομιστή, ενώ ο πελάτης -το πρόγραμμα περιήγησης- αναλαμβάνει την παρουσίαση και την αλληλεπίδραση με τον χρήστη. Σε αντίθεση με τις παραδοσιακές εφαρμογές επιφάνειας εργασίας που εγκαθίστανται τοπικά σε κάθε συσκευή, οι εφαρμογές ιστού φιλοξενούνται κεντρικά και είναι προσβάσιμες από οποιαδήποτε συσκευή διαθέτει σύνδεση στο διαδίκτυο και ένα συμβατό πρόγραμμα περιήγησης (Doyle & Lopes, 2005). Χαρακτηριστικά παραδείγματα αποτελούν τα συστήματα ηλεκτρονικού εμπορίου (e-commerce), οι πλατφόρμες κοινωνικής δικτύωσης (social networking platforms), τα συστήματα ηλεκτρονικής τραπεζικής (online banking) και τα εργαλεία συνεργασίας (collaboration tools), όπως τα Google Docs και το Microsoft Teams.

2.1.2 Διαφορές από τις Παραδοσιακές Εφαρμογές

Οι εφαρμογές ιστού διαφέρουν από τις παραδοσιακές εφαρμογές λογισμικού σε αρκετές θεμελιώδεις διαστάσεις. Ως προς την αρχιτεκτονική, το παραδοσιακό λογισμικό ακολουθεί συχνά ένα μονολιθικό, μονο-επίπεδο μοντέλο εγκατεστημένο στη συσκευή του χρήστη, ενώ οι εφαρμογές ιστού υιοθετούν πολυ-επίπεδες, κατανεμημένες αρχιτεκτονικές με σαφή διαχωρισμό της παρουσίασης, της επιχειρησιακής λογικής και των δεδομένων. Ως προς την εγκατάσταση και τις ενημερώσεις, οι παραδοσιακές εφαρμογές απαιτούν χειροκίνητη ενημέρωση σε κάθε συσκευή ξεχωριστά, ενώ οι εφαρμογές ιστού ενημερώνονται κεντρικά στον διακομιστή, με άμεση πρόσβαση όλων των χρηστών στην τελευταία έκδοση χωρίς καμία ενέργεια από την πλευρά τους. Τέλος, ως προς την ανεξαρτησία πλατφόρμας, αξιοποιώντας τα ανοικτά πρότυπα HTML, CSS και JavaScript, οι εφαρμογές ιστού είναι εγγενώς ανεξάρτητες από το λειτουργικό σύστημα και τη συσκευή του χρήστη - ένα πλεονέκτημα που το παραδοσιακό λογισμικό, δεσμευμένο σε συγκεκριμένες πλατφόρμες, δεν διαθέτει.

Η κατανεμημένη φύση των εφαρμογών ιστού, ωστόσο, εισάγει και προκλήσεις που απουσιάζουν από το παραδοσιακό λογισμικό: τη διαχείριση της κατάστασης πάνω από ένα άνευ μνήμης (stateless) πρωτόκολλο, όπως το HTTP, την ταυτόχρονη εξυπηρέτηση πολλών χρηστών, την ασφάλεια της επικοινωνίας μέσω δικτύου, και την ανάγκη για κλιμακωσιμότητα υπό μεταβαλλόμενο φορτίο (Doyle & Lopes, 2005).

Οι διαφορές αυτές αναγνωρίστηκαν νωρίς από την επιστημονική κοινότητα και αποτέλεσαν τον λόγο θεμελίωσης ενός ξεχωριστού κλάδου, της Μηχανικής Ιστού (Web Engineering). Ήδη από το 1998, ο Pressman διατύπωσε το ερώτημα κατά πόσο οι εφαρμογές του διαδικτύου μπορούν πράγματι να «μηχανικοποιηθούν», επισημαίνοντας ότι διακρίνονται από μια συνεχή, ασταμάτητη εξέλιξη (continual change), η οποία καθιστά ανεπαρκή την άμεση μεταφορά των

παραδοσιακών μοντέλων λογισμικού. Λίγο αργότερα, οι Ginige και Murugesan (2001) τεκμηρίωσαν ότι οι εφαρμογές ιστού είναι ταυτόχρονα έντονα εξαρτημένες από το δίκτυο (network-intensive), καθοδηγούμενες από το περιεχόμενο (content-driven) και υποκείμενες σε ασαφείς και μεταβαλλόμενες απαιτήσεις, με αποτέλεσμα τη διαρκή πίεση για συντήρηση και επικαιροποίηση. Αυτή ακριβώς η διαφοροποίηση - η συνύπαρξη λογισμικού, περιεχομένου και διεπαφής υπό συνθήκες αβεβαιότητας - είναι που δικαιολογεί την ανάγκη για εξειδικευμένες μεθόδους και τεχνικές, δηλαδή το αντικείμενο της παρούσας επισκόπησης.

2.1.3 Κύρια Χαρακτηριστικά

Τέσσερα γνωρίσματα καθορίζουν τη φύση των εφαρμογών ιστού. Το πρώτο είναι η *αρχιτεκτονική πελάτη-διακομιστή*, όπου το πρόγραμμα περιήγησης αρχικοποιεί αιτήματα και ο διακομιστής τα επεξεργάζεται και αποκρίνεται, συχνά με ασύγχρονο τρόπο ώστε η διεπαφή να παραμένει αποκριτική. Το δεύτερο είναι η *τεχνολογική ετερογένεια*, με διαφορετικές γλώσσες και τεχνολογίες στο front-end και το back-end, γεγονός που απαιτεί από τους προγραμματιστές γνώση σε πολλαπλά επίπεδα της τεχνολογικής στοίβας (Doğan, Betin-Can & Garousi, 2014). Το τρίτο είναι *η δυναμικότητα και η διαδραστικότητα*, που επιτρέπουν την επεξεργασία δεδομένων σε πραγματικό χρόνο, την αυθεντικοποίηση χρηστών και την προσαρμογή του περιεχομένου, ανάλογα με τον ρόλο του χρήστη και τις αλληλεπιδράσεις με αυτόν. Τέλος, το τέταρτο είναι η *κατανεμημένη φύση τους*, που εισάγει τις προκλήσεις της αξιοπιστίας, της κλιμακωσιμότητας, της απόδοσης και της ασφάλειας ως κεντρικά ζητήματα σχεδίασης.

Τα γνωρίσματα αυτά συνάδουν με την κανονική περιγραφή που διατύπωσαν οι Murugesan, Deshpande, Hansen και Ginige (2001) στη θεμελιώδη εργασία που καθιέρωσε τη Μηχανική Ιστού ως αυτόνομο επιστημονικό κλάδο. Οι συγγραφείς επισημαίνουν ότι οι εφαρμογές ιστού συνδυάζουν χαρακτηριστικά που σπανίως συνυπάρχουν στο παραδοσιακό λογισμικό: είναι ταυτόχρονα φορείς περιεχομένου και λειτουργικότητας, απευθύνονται σε ένα ευρύ, ανώνυμο και ετερογενές κοινό χρηστών, και αναπτύσσονται υπό ασφυκτικές χρονικές πιέσεις («web time») που δυσχεραίνουν την εφαρμογή αυστηρών μεθοδολογιών. Επιπλέον, η ποιότητα - ιδίως ως προς τη χρηστικότητα, την απόδοση, την ασφάλεια και τη διαθεσιμότητα - αποκτά κρισιμότητα μεγαλύτερη από ό,τι σε πολλές κατηγορίες παραδοσιακών εφαρμογών, καθώς οι εναλλακτικές λύσεις για τον χρήστη βρίσκονται συχνά σε απόσταση ενός μόνο συνδέσμου. Η κατανόηση αυτών των χαρακτηριστικών αποτελεί την αφετηρία για την ανάλυση των μεθόδων και τεχνικών που ακολουθεί στα επόμενα κεφάλαια.

2.1.4 Το Μοντέλο Ποιότητας ISO/IEC 25010

Η ποιότητα μιας εφαρμογής ιστού δεν είναι μια αόριστη έννοια, αλλά ένα μετρήσιμο μέγεθος που μπορεί να αποτυπωθεί μέσα από τυποποιημένα χαρακτηριστικά. Το διεθνές πρότυπο ISO/IEC 25010, μέρος της σειράς SQuaRE (Systems and software Quality Requirements and Evaluation) που ορίζεται από το ISO/IEC 25000:2014, αποτελεί σήμερα το πλέον διαδεδομένο πλαίσιο για τον σκοπό αυτό, και για τον λόγο αυτό χρησιμοποιείται και ως ένας από τους άξονες διαμόρφωσης των κριτηρίων σύγκρισης του Κεφαλαίου 4. Στην έκδοση του 2011, το πρότυπο ορίζει ένα μοντέλο ποιότητας προϊόντος με οκτώ χαρακτηριστικά: την λειτουργική καταλληλότητα (functional suitability), την αποδοτικότητα επιδόσεων (performance efficiency), τη συμβατότητα (compatibility), τη χρηστικότητα (usability), την αξιοπιστία (reliability), την ασφάλεια (security), τη συντηρησιμότητα (maintainability) και τη φορητότητα (portability), καθένα από τα οποία αναλύεται περαιτέρω σε επιμέρους υπο-χαρακτηριστικά. Μάλιστα, η ασφάλεια και η συμβατότητα αναδείχθηκαν σε αυτόνομα χαρακτηριστικά ήδη από το 2011 -σε αντίθεση με το προγενέστερο πρότυπο ISO/IEC 9126, όπου η ασφάλεια αντιμετωπιζόταν ως απλό υπο-χαρακτηριστικό της λειτουργικότητας- γεγονός που αντανακλά την αυξανόμενη κρισιμότητά τους για τα σύγχρονα δικτυακά συστήματα.

Η αναθεώρηση του 2023 (ISO/IEC 25010:2023) επεκτείνει το μοντέλο σε εννέα χαρακτηριστικά, προσθέτοντας την προστασία (safety) και μετονομάζοντας τη χρηστικότητα σε «ικανότητα αλληλεπίδρασης» (interaction capability) και τη φορητότητα σε «ευελιξία» (flexibility), με νέα υπο-χαρακτηριστικά όπως η συμπεριληπτικότητα (inclusivity) και η κλιμακωσιμότητα (scalability). Για τις εφαρμογές ιστού, η αξία του προτύπου έγκειται στο ότι προσφέρει ένα κοινό λεξιλόγιο για την προδιαγραφή, τη μέτρηση και την αξιολόγηση της ποιότητας σε όλα τα στάδια του κύκλου ζωής: κάθε απαίτηση ποιότητας μπορεί να αντιστοιχηθεί σε ένα ή περισσότερα ποιοτικά χαρακτηριστικά, με συγκεκριμένους στόχους (για παράδειγμα, χρόνο απόκρισης, διαθεσιμότητα ή ποσοστό σφαλμάτων), οι οποίοι στη συνέχεια καθοδηγούν τη σχεδίαση και τον έλεγχο. Είναι, ωστόσο, σημαντικό να σημειωθεί ότι το πρότυπο παρέχει εννοιολογικούς ορισμούς και όχι έτοιμες μετρικές (metrics), η μετατροπή των χαρακτηριστικών σε ποσοτικά μεγέθη απαιτεί επιπλέον μοντελοποίηση, γεγονός που εξηγεί γιατί στην πράξη διαφορετικές μελέτες - όπως η συγκριτική αξιολόγηση των front-end πλαισίων που εξετάζεται στην Ενότητα 2.6.8 (Loja & Maita, 2024) - εφαρμόζουν το πρότυπο με ελαφρώς διαφορετικό τρόπο.

2.2 Αρχιτεκτονική Εφαρμογών Ιστού

2.2.1 Τριεπίπεδη Αρχιτεκτονική

Η πλέον διαδεδομένη αρχιτεκτονική για εφαρμογές ιστού είναι η τρι-επίπεδη, η οποία διαχωρίζει την εφαρμογή σε τρία διακριτά λογικά επίπεδα. Το επίπεδο παρουσίασης περιλαμβάνει τη διεπαφή του χρήστη και αναπτύσσεται με HTML, CSS, JavaScript και σύγχρονα πλαίσια (frameworks) όπως τα React, Angular και Vue.js. Το επίπεδο εφαρμογής, ή επίπεδο επιχειρησιακής λογικής, επεξεργάζεται τα αιτήματα, εφαρμόζει τους επιχειρησιακούς κανόνες της εφαρμογής και συντονίζει την επικοινωνία ανάμεσα στην παρουσίαση και τα δεδομένα· υλοποιείται με γλώσσες όπως η Java, η C#, η Python, η Ruby, η PHP ή η JavaScript μέσω του περιβάλλοντος Node.js. Τέλος, το επίπεδο δεδομένων φιλοξενεί το σύστημα διαχείρισης βάσης δεδομένων -για παράδειγμα MySQL, PostgreSQL ή MongoDB- που αποθηκεύει, ανακτά και διαχειρίζεται τα δεδομένα με μόνιμο τρόπο. Ο διαχωρισμός αυτός προσφέρει συντηρησιμότητα, καθώς αλλαγές σε ένα επίπεδο δεν επηρεάζουν αναγκαστικά τα υπόλοιπα, καθώς και ανεξάρτητη επεκτασιμότητα κάθε επιπέδου.

2.2.2 Σύγχρονες Αρχιτεκτονικές

Πέρα από το κλασικό τρι-επίπεδο μοντέλο, οι σύγχρονες εφαρμογές ιστού υιοθετούν ολοένα και συχνότερα εξελιγμένα αρχιτεκτονικά πρότυπα. Η πολυεπίπεδη αρχιτεκτονική επεκτείνει το τρι-επίπεδο μοντέλο με εξειδικευμένα επίπεδα για caching, εξισορρόπηση φορτίου και αυθεντικοποίηση. Η αρχιτεκτονική microservices διασπά την εφαρμογή σε μικρότερες, ανεξάρτητα αναπτυσσόμενες υπηρεσίες, καθεμία με την δική της τεχνολογική στοίβα και τη δική της βάση δεδομένων, επικοινωνώντας μέσω ελαφρών διεπαφών (Velepucha & Flores, 2023). Από την άλλη μεριά, οι serverless αρχιτεκτονικές επιτρέπουν την εκτέλεση κώδικα ως αντίδραση σε συμβάντα, χωρίς τη διαχείριση διακομιστών από τον προγραμματιστή και με χρέωση βάσει της πραγματικής χρήσης. Οι σύγχρονες αυτές προσεγγίσεις αναλύονται εκτενέστερα στην Ενότητα 2.6, στο πλαίσιο των αντίστοιχων οικογενειών μεθόδων, καθώς και στο Κεφάλαιο 4.

2.2.3 REST και Αρχιτεκτονικές Οδηγούμενες από API

Εγκάρσια προς όλες τις παραπάνω αρχιτεκτονικές λειτουργεί ένα καθοριστικό στυλ επικοινωνίας: η αρχιτεκτονική REST (Representational State Transfer). Το στυλ αυτό, το οποίο όρισε ο Fielding (2000) στη διδακτορική του διατριβή, θεμελίωσε τον τρόπο με τον οποίο τα κατανεμημένα συστήματα του ιστού επικοινωνούν μέσω του πρωτοκόλλου HTTP, με έμφαση στην

επεκτασιμότητα των αλληλεπιδράσεων, τη γενικότητα των διεπαφών, τη διαστρωματωμένη οργάνωση (layered system) και την ανεξάρτητη ανάπτυξη των συστατικών. Η υιοθέτηση των αρχών REST είχε βαθιά επίδραση στον σύγχρονο σχεδιασμό: επέτρεψε τον σαφή διαχωρισμό του front-end από το back-end, με το επίπεδο παρουσίασης - ιδίως στα Single Page Applications και στα Progressive Web Apps - να επικοινωνεί με τη λογική του διακομιστή αποκλειστικά μέσω τυποποιημένων διεπαφών προγραμματισμού εφαρμογών (APIs). Η προσέγγιση αυτή, γνωστή και ως API-driven ή headless αρχιτεκτονική, αποτελεί το συνδετικό ιστό των microservices, καθώς κάθε ανεξάρτητη υπηρεσία εκθέτει τη λειτουργικότητά της μέσω ενός καλά ορισμένου API. Νεότερα πρότυπα, όπως το GraphQL, έρχονται να αντιμετωπίσουν ορισμένους περιορισμούς του κλασικού REST στην ανάκτηση δεδομένων, χωρίς ωστόσο να ανατρέπουν τη θεμελιώδη αρχή της αποσύζευξης μέσω διεπαφών που εισήγαγε το REST.

2.2.4 Στρατηγικές Απόδοσης Περιεχομένου και JAMstack

Μια αρχιτεκτονική απόφαση με άμεσο αντίκτυπο στην απόδοση, την εμπειρία χρήστη και τη βελτιστοποίηση για μηχανές αναζήτησης είναι το πού και το πότε παράγεται το HTML που φτάνει στον χρήστη. Διακρίνονται τρεις θεμελιώδεις στρατηγικές απόδοσης περιεχομένου. Στην απόδοση από την πλευρά του πελάτη (Client-Side Rendering, CSR), ο διακομιστής αποστέλλει έναν ελάχιστο σκελετό HTML μαζί με μια δέσμη JavaScript, και το πρόγραμμα περιήγησης αναλαμβάνει να κατασκευάσει δυναμικά το περιεχόμενο· η προσέγγιση αυτή, τυπική των μονοσέλιδων εφαρμογών (Single Page Applications, SPA), προσφέρει πλούσια διαδραστικότητα και ομαλές μεταβάσεις μετά την αρχική φόρτωση, αλλά μπορεί να καθυστερήσει την πρώτη εμφάνιση του περιεχομένου και δυσχεραίνει την ευρετηρίαση από μηχανές αναζήτησης. Στην απόδοση από την πλευρά του διακομιστή (Server-Side Rendering, SSR), ο διακομιστής παράγει το πλήρες HTML της σελίδας πριν το αποστείλει στο πρόγραμμα περιήγησης. Στο παραδοσιακό μοντέλο των πολυσέλιδων εφαρμογών (Multi-Page Applications, MPA) αυτό συμβαίνει σε κάθε πλοήγηση, με πλήρη επαναφόρτωση της σελίδας. Τα σύγχρονα πλαίσια (frameworks), όπως το Next.js, το Nuxt και το Angular SSR, υιοθετούν ένα υβριδικό μοντέλο: ο διακομιστής αποδίδει την αρχική σελίδα, η οποία στη συνέχεια ενυδατώνεται (hydration) στο πρόγραμμα περιήγησης, και οι επόμενες πλοηγήσεις εκτελούνται στην πλευρά του πελάτη, όπως σε μια SPA. Το SSR επιταχύνει την εμφάνιση ουσιαστικού περιεχομένου (First Contentful Paint) και βελτιώνει τη συμπεριφορά ως προς τη βελτιστοποίηση για μηχανές αναζήτησης (Search Engine Optimization, SEO), καθώς οι μηχανές αναζήτησης λαμβάνουν έτοιμο περιεχόμενο, με αντίτιμο τον αυξημένο υπολογιστικό φόρτο στον διακομιστή, αφού η απόδοση γίνεται ανά αίτημα. Τέλος, στη στατική παραγωγή ιστοσελίδων (Static Site Generation, SSG), το HTML παράγεται εκ των προτέρων κατά τον χρόνο κατασκευής (build time) και διανέμεται μέσω δικτύου διανομής περιεχομένου (Content Delivery Network, CDN), προσφέροντας πολύ υψηλή ταχύτητα και μειωμένη επιφάνεια επίθεσης, αλλά ενδείκνυται κυρίως για περιεχόμενο που δεν μεταβάλλεται συχνά (τεκμηρίωση λογισμικού, ιστολόγια, σελίδες προϊόντων και προωθητικοί ιστότοποι).

Στην πράξη, οι στρατηγικές αυτές δεν είναι αμοιβαία αποκλειόμενες. Τα σύγχρονα μετα-πλαίσια (meta-frameworks), όπως το Next.js και το Nuxt, επιτρέπουν την επιλογή στρατηγικής ανά σελίδα και συνδυάζουν τα πλεονεκτήματα των στρατηγικών αυτών. Στο πλαίσιο αυτό, η ενυδάτωση (hydration) λειτουργεί ως η συνδετική τεχνική ανάμεσα στη στατική ή από τον διακομιστή παραγωγή και τη διαδραστικότητα του πελάτη: ο κώδικας JavaScript αναλαμβάνει το ήδη στατικά αποδομένο ή δυναμικά δημιουργημένο από τον διακομιστή περιεχόμενο και το καθιστά διαδραστικό, χωρίς να το αναπαράγει. Συμπληρωματικά, η σταδιακή στατική αναγέννηση (Incremental Static Regeneration, ISR) επιτρέπει την εκ των προτέρων παραγωγή στατικών σελίδων με περιοδική ανανέωσή τους. Η ανανέωση αυτή δεν γίνεται με ασύγχρονες κλήσεις από τον πελάτη, αλλά στον διακομιστή: μετά την πάροδο ενός καθορισμένου χρονικού ορίου επικύρωσης, το επόμενο αίτημα που φτάνει πυροδοτεί την εκ νέου παραγωγή της σελίδας στο

παρασκήνιο, ενώ οι χρήστες εξακολουθούν να εξυπηρετούνται από την προηγούμενη εκδοχή της. Η τεχνική ενδείκνυται για περιεχόμενο που μεταβάλλεται τακτικά αλλά όχι ανά αίτημα, όπως κατάλογοι προϊόντων, άρθρα ειδήσεων ή σελίδες με τιμές. Η επιλογή της κατάλληλης στρατηγικής αποτελεί έτσι μια ισορροπία ανάμεσα στην αρχική ταχύτητα φόρτωσης, τη διαδραστικότητα, την επιβάρυνση του διακομιστή και τις απαιτήσεις ευρετηρίασης.

Η λογική της εκ των προτέρων παραγωγής και της αποσύζευξης του front-end από το back-end αποκρυσταλλώνεται στην αρχιτεκτονική JAMstack, όρος που εισήγαγε ο Mathias Biilmann της Netlify γύρω στο 2015 (Biilmann & Hawksworth, 2019). Το ακρωνύμιο αναφέρεται στα τρία δομικά στοιχεία - JavaScript, APIs και Markup - και περιγράφει μια προσέγγιση όπου ολόκληρο το front-end προπαράγεται σε βελτιστοποιημένες στατικές σελίδες κατά τον χρόνο κατασκευής, διανέμεται μέσω CDN, και αντλεί δυναμική λειτουργικότητα από επαναχρησιμοποιήσιμα APIs και serverless υπηρεσίες. Τα οφέλη της προσέγγισης είναι η υψηλή απόδοση, η ενισχυμένη ασφάλεια και η ευκολότερη κλιμάκωση. Η βελτίωση της ασφάλειας δεν οφείλεται στην κατάργηση της λογικής από την πλευρά του διακομιστή - τα APIs και οι serverless συναρτήσεις εξακολουθούν φυσικά να υφίστανται - αλλά στο ότι ο διακομιστής που εξυπηρετεί τις σελίδες δεν εκτελεί πλέον λογική εφαρμογής ούτε διατηρεί άμεση σύνδεση με τη βάση δεδομένων· η λογική μετατοπίζεται σε απομονωμένες, ανεξάρτητα ελεγχόμενες υπηρεσίες, περιορίζοντας έτσι την επιφάνεια επίθεσης. Αξίζει να σημειωθεί ότι η ίδια η Netlify απέσυρε τον όρο ως εμπορική ετικέτα το 2023, προς όφελος του ευρύτερου όρου «συνθέσιμη αρχιτεκτονική» (composable architecture)· ωστόσο οι θεμελιώδεις αρχές της - προπαραγωγή, προτεραιότητα στα APIs, διανομή μέσω CDN και αποσυζευγμένο front-end - εξακολουθούν να ενσωματώνονται στα σύγχρονα headless CMS και στα υβριδικά μετα-πλαίσια απόδοσης.

2.3 Στάδια Ανάπτυξης Εφαρμογών Ιστού

Η ανάπτυξη εφαρμογών ιστού ακολουθεί τον Κύκλο Ζωής Ανάπτυξης Λογισμικού (Software Development Life Cycle, SDLC), ο οποίος αποτελείται από διακριτά στάδια που παραμένουν σταθερά ανεξαρτήτως της επιμέρους μεθόδου που υιοθετείται. Η βιβλιογραφία διακρίνει συνήθως έξι τέτοια στάδια, αντιμετωπίζοντας την ανάλυση απαιτήσεων και τη σχεδίαση ως αυτοτελείς φάσεις με χωριστά παραδοτέα. Η παρούσα εργασία χρησιμοποιεί ένα μοντέλο πέντε σταδίων, συγχωνεύοντας τις δύο αυτές φάσεις σε μία για τους λόγους που εξηγούνται στην Ενότητα 2.3.1· το μοντέλο αυτό αποτελεί και τον άξονα των κριτηρίων σύγκρισης του Κεφαλαίου 4. Το σχήμα αυτό δεν αποτελεί αυστηρά τυποποιημένο μοντέλο, αλλά τη συνήθη, ευρέως αποδεκτή διάρθρωση των δραστηριοτήτων του κύκλου ζωής, όπως αποτυπώνεται σε καθιερωμένα εγχειρίδια της μηχανικής λογισμικού (Sommerville, 2016) και, σε πολύ λεπτομερέστερο επίπεδο ανάλυσης, στο διεθνές πρότυπο ISO/IEC/IEEE 12207 για τις διεργασίες του κύκλου ζωής λογισμικού (ISO/IEC/IEEE, 2026).

2.3.1 Ανάλυση Απαιτήσεων και Σχεδίαση

Το πρώτο στάδιο του κύκλου ζωής θέτει τα θεμέλια ολόκληρου του έργου και περιλαμβάνει δύο στενά συνδεδεμένες δραστηριότητες: τη μηχανική απαιτήσεων και τη σχεδίαση. Η μηχανική απαιτήσεων (requirements engineering) περιλαμβάνει την εκμαίευση, την ανάλυση, την τεκμηρίωση και την επικύρωση των αναγκών των ενδιαφερομένων μερών. Σε αρκετά μοντέλα του κύκλου ζωής οι δύο αυτές δραστηριότητες παρουσιάζονται ως διακριτές φάσεις, με χωριστά παραδοτέα και εμπλεκόμενους ρόλους. Στην παρούσα εργασία εξετάζονται από κοινού, επειδή στην ανάπτυξη εφαρμογών ιστού - με τις συχνά ασαφείς και ταχέως μεταβαλλόμενες απαιτήσεις - η εκμαίευση των απαιτήσεων και η σχεδίαση εξελίσσονται παράλληλα και ανατροφοδοτούν η μία την άλλη. Οι απαιτήσεις διακρίνονται παραδοσιακά σε λειτουργικές, οι οποίες περιγράφουν τι πρέπει να κάνει το σύστημα, και σε μη λειτουργικές, οι οποίες περιγράφουν ποιοτικά

χαρακτηριστικά, όπως η απόδοση, η ασφάλεια, η χρηστικότητα και η επεκτασιμότητα. Για τις εφαρμογές ιστού, οι μη λειτουργικές απαιτήσεις αποκτούν ιδιαίτερη βαρύτητα, δεδομένης της ανάγκης για συνεχή διαθεσιμότητα, ταχεία απόκριση υπό μεταβαλλόμενο φορτίο και ανθεκτικότητα απέναντι σε επιθέσεις σε ένα εκτεθειμένο, δημόσιο περιβάλλον.

Η σημασία της ορθής διαχείρισης αυτού του σταδίου τεκμηριώνεται από μια θεμελιώδη αρχή της μηχανικής λογισμικού: το κόστος διόρθωσης ενός σφάλματος αυξάνεται καθώς αυτό εντοπίζεται σε μεταγενέστερα στάδια. Η αρχή αυτή, που ανάγεται στις πρώιμες μελέτες του Boehm τη δεκαετία του 1970 και του 1980 (Boehm, 1981) και επιβεβαιώθηκε δύο δεκαετίες αργότερα από τους Boehm και Basili (2001), υποστηρίζει ότι ένα ελάττωμα στις απαιτήσεις που διαφεύγει έως την παραγωγή μπορεί να κοστίσει τάξεις μεγέθους περισσότερο από ό,τι αν είχε εντοπιστεί έγκαιρα· αξίζει να σημειωθεί ότι τα ακριβή πολλαπλάσια έχουν αμφισβητηθεί στη νεότερη βιβλιογραφία (Menzies et al., 2017) και ποικίλλουν σημαντικά ανάλογα με το είδος του έργου.

Σε ό,τι αφορά τη σχεδίαση, αυτή διακρίνεται στην αρχιτεκτονική σχεδίαση, που καθορίζει τη δομή και τα επίπεδα του συστήματος, στη λεπτομερή σχεδίαση, όπου προσδιορίζεται η συμπεριφορά του συστήματος μέσα από μοντέλα αλληλεπίδρασης και ροής ελέγχου, τη σχεδίαση των δεδομένων και τον ορισμό των διεπαφών προγραμματισμού (APIs), και στη σχεδίαση της διεπαφής και της εμπειρίας χρήστη (UI/UX), η οποία υλοποιείται συχνά μέσα από wireframes και διαδραστικά πρωτότυπα (prototypes) πριν από τη συγγραφή κώδικα, επιτρέποντας την έγκαιρη επικύρωση των σχεδιαστικών επιλογών.

2.3.2 Υλοποίηση

Το στάδιο της υλοποίησης αφορά τη μετατροπή των σχεδιαστικών προδιαγραφών σε εκτελέσιμο κώδικα. Πέρα από την καθαυτό συγγραφή κώδικα, το στάδιο αυτό διέπεται από ένα σύνολο καθιερωμένων πρακτικών που διασφαλίζουν την ποιότητα και τη συντηρησιμότητα. Κεντρικό ρόλο διαδραματίζει ο έλεγχος εκδόσεων (version control), με κυρίαρχο εργαλείο το Git, ο οποίος επιτρέπει την παράλληλη εργασία πολλών προγραμματιστών, την ιχνηλάτηση των αλλαγών και τη διαχείριση διακλαδώσεων (branches). Στις καθιερωμένες πρακτικές συγκαταλέγονται η επιθεώρηση κώδικα (code review) μέσω αιτημάτων συγχώνευσης, η τήρηση προτύπων κωδικοποίησης και η αρχή του καθαρού κώδικα (clean code), δηλαδή ένα σύνολο συμβάσεων συγγραφής που στοχεύουν στην αναγνωσιμότητα και την κατανοησιμότητα του κώδικα από άλλους προγραμματιστές - περιγραφική ονοματοδοσία, σύντομες συναρτήσεις με έναν σαφή σκοπό, αποφυγή επανάληψης και ελαχιστοποίηση της πολυπλοκότητας (Martin, 2008). Οι πρακτικές αυτές συμβάλλουν στη μείωση του τεχνικού χρέους (technical debt), όρου που περιγράφει το μελλοντικό κόστος αναδιάρθρωσης, το οποίο συσσωρεύεται όταν επιλέγεται μια ταχύτερη αλλά υποδεέστερη τεχνική λύση αντί της ορθότερης· όπως και το οικονομικό χρέος, το τεχνικό χρέος «τοκίζεται», καθιστώντας κάθε επόμενη αλλαγή στον κώδικα πιο δύσκολη και δαπανηρή (Cunningham, 1992). Στις σύγχρονες προσεγγίσεις, η υλοποίηση συνδέεται άρρηκτα με τη συνεχή ενσωμάτωση (continuous integration), όπου κάθε αλλαγή ενσωματώνεται συχνά στον κοινό κορμό του κώδικα και επικυρώνεται αυτόματα. Η επικύρωση αυτή δεν περιορίζεται στη μεταγλώττιση, αλλά περιλαμβάνει και την εκτέλεση της αυτοματοποιημένης σουίτας δοκιμών, γεγονός που καθιστά το στάδιο της υλοποίησης άμεσα συνδεδεμένο με εκείνο των δοκιμών (Ενότητα 2.3.3)· με αυτό τον τρόπο, τα σφάλματα ενσωμάτωσης εντοπίζονται σε πρώιμο στάδιο, πριν συσσωρευτούν και γίνουν δαπανηρά στη διόρθωσή τους.

2.3.3 Δοκιμές και Διασφάλιση Ποιότητας

Το στάδιο των δοκιμών στοχεύει στη συστηματική διασφάλιση της ποιότητας και οργανώνεται σε πολλαπλά επίπεδα, τα οποία αποτυπώνονται συχνά στη μεταφορά της «πυραμίδας δοκιμών» (test

pyramid), όπως τη διατύπωσε ο Cohn (2009). Στη βάση της πυραμίδας βρίσκονται οι πολυάριθμες δοκιμές μονάδας (unit tests), που ελέγχουν μεμονωμένες συναρτήσεις ή συστατικά σε απομόνωση και εκτελούνται γρήγορα· στο ενδιάμεσο επίπεδο βρίσκονται οι δοκιμές ολοκλήρωσης (integration tests), που επαληθεύουν τη συνεργασία πολλαπλών συστατικών· και στην κορυφή, οι λιγότερες αλλά πιο δαπανηρές δοκιμές από άκρο σε άκρο (end-to-end tests), που προσομοιώνουν την πλήρη αλληλεπίδραση του χρήστη μέσω της χρήσης εργαλείων, όπως το Selenium και το Cypress. Η λογική της πυραμίδας είναι ότι το μεγαλύτερο βάρος πρέπει να δίνεται στα ταχύτερα και φθηνότερα κατώτερα επίπεδα. Αυτό δεν υποβαθμίζει τη σημασία των δοκιμών από άκρο σε άκρο, οι οποίες παραμένουν οι μόνες που επαληθεύουν το σύστημα ως σύνολο υπό ρεαλιστικές συνθήκες χρήσης· η λογική είναι ότι τα κατώτερα επίπεδα απορροφούν το μεγαλύτερο μέρος του κινδύνου πριν φτάσει εκεί. Οι δοκιμές μονάδας εντοπίζουν τα σφάλματα της εσωτερικής λογικής κάθε συστατικού και οι δοκιμές ολοκλήρωσης τα σφάλματα των συμβολαίων μεταξύ των συστατικών - δηλαδή ακριβώς τις κατηγορίες ελαττωμάτων που θα εκδηλώνονταν αργότερα ως αποτυχίες από άκρο σε άκρο, αλλά με πολύ δυσκολότερη διάγνωση, αφού σε εκείνο το επίπεδο μια αποτυχία δεν υποδεικνύει την αιτία της. Έτσι, οι δοκιμές από άκρο σε άκρο μπορούν να περιοριστούν σε ένα μικρότερο σύνολο κρίσιμων διαδρομών του χρήστη, χωρίς αυτό να συνεπάγεται απώλεια εμπιστοσύνης στο σύστημα.

Πέρα από τη λειτουργική ορθότητα, οι εφαρμογές ιστού απαιτούν εξειδικευμένες μορφές μη λειτουργικού ελέγχου: δοκιμές απόδοσης και φόρτου, που αξιολογούν τη συμπεριφορά του συστήματος υπό υψηλή κίνηση, δοκιμές χρηστικότητας και προσβασιμότητας, καθώς και δοκιμές ασφαλείας, οι οποίες είναι κρίσιμες λόγω της έκθεσης των εφαρμογών σε δημόσια δίκτυα. Ένα σημαντικό μεθοδολογικό ρεύμα που διασυνδέει το στάδιο των δοκιμών με εκείνο της υλοποίησης είναι η ανάπτυξη καθοδηγούμενη από δοκιμές (Test-Driven Development), την οποία εισήγαγε συστηματικά ο Beck (2002), όπου η συγγραφή της δοκιμής προηγείται της συγγραφής του κώδικα. Το κύριο πλεονέκτημα της πρακτικής αυτής είναι ότι η δοκιμή λειτουργεί ως εκτελέσιμη προδιαγραφή: ο προγραμματιστής αναγκάζεται να διατυπώσει με ακρίβεια την αναμενόμενη συμπεριφορά πριν υλοποιήσει, γεγονός που οδηγεί σε απλούστερο και καλύτερα αποσυζευγμένο σχεδιασμό, εξασφαλίζει εξ ορισμού υψηλή κάλυψη δοκιμών και παρέχει ένα δίχτυ ασφαλείας που επιτρέπει τη μετέπειτα αναδιάρθρωση του κώδικα με εμπιστοσύνη. Η αυτοματοποίηση των δοκιμών, ενταγμένη σε αγωγούς (pipelines) συνεχούς ενσωμάτωσης, αποτελεί σήμερα προϋπόθεση για τη συχνή και αξιόπιστη παράδοση λογισμικού (software delivery), δηλαδή για τη διαδικασία με την οποία μια επικυρωμένη έκδοση του κώδικα φτάνει σε κατάσταση έτοιμη προς διάταξη.

2.3.4 Διάταξη (Deployment)

Το στάδιο της διάταξης αφορά τη μεταφορά της εφαρμογής σε περιβάλλον παραγωγής, ώστε να καταστεί διαθέσιμη στους τελικούς χρήστες. Στις σύγχρονες προσεγγίσεις, η διαδικασία αυτή αυτοματοποιείται μέσω αγωγών (pipelines) συνεχούς ενσωμάτωσης και παράδοσης (CI/CD), οι οποίοι καλύπτουν δύο διακριτές αλλά συνεχόμενες δραστηριότητες: τη συνεχή παράδοση (continuous delivery), που φέρνει κάθε επικυρωμένη έκδοση σε κατάσταση έτοιμη προς διάταξη, και την καθαυτό διάταξη, που την εγκαθιστά και την ενεργοποιεί στο περιβάλλον παραγωγής. Η διαδικασία βασίζεται σε τεχνολογίες δοχειοποίησης (containerization), όπως το Docker, και ενορχήστρωσης δοχείων, όπως το Kubernetes, οι οποίες εξασφαλίζουν συνέπεια μεταξύ των περιβαλλόντων ανάπτυξης, δοκιμών και παραγωγής. Παράλληλα, η αρχή της υποδομής ως κώδικα (Infrastructure as Code) επιτρέπει την περιγραφή και τη διαχείριση της υποδομής με δηλωτικά αρχεία, καθιστώντας τις διατάξεις αναπαραγώγιμες και ελεγχόμενες.

Για την ελαχιστοποίηση του κινδύνου και του χρόνου μη διαθεσιμότητας κατά τη διάταξη, έχουν καθιερωθεί εξειδικευμένες στρατηγικές. Στη διάταξη blue-green διατηρούνται δύο πανομοιότυπα περιβάλλοντα, ένα ενεργό και ένα σε αναμονή· η νέα έκδοση εγκαθίσταται και

επαληθεύεται στο δεύτερο και, αφού ληφθεί η απόφαση μετάβασης, η κίνηση μεταφέρεται ακαριαία από το ένα στο άλλο, προσφέροντας άμεση δυνατότητα επαναφοράς (rollback) με κόστος τη διατήρηση διπλής υποδομής. Αντιθέτως, στη σταδιακή διάταξη canary, η νέα έκδοση εκτίθεται αρχικά σε ένα μικρό ποσοστό της κίνησης και το ποσοστό αυτό αυξάνεται σταδιακά, εφόσον δεν παρατηρηθούν προβλήματα, περιορίζοντας την έκταση μιας ενδεχόμενης αστοχίας με πιο αποδοτική χρήση της υποδομής. Η επιλογή στρατηγικής εξαρτάται από την ανοχή στον κίνδυνο, τους διαθέσιμους πόρους και την ανάγκη για επικύρωση σε πραγματικές συνθήκες.

2.3.5 Συντήρηση και Λειτουργία

Το τελευταίο - χρονικά αλλά όχι σε σημασία - στάδιο του κύκλου ζωής περιλαμβάνει δύο αλληλένδετες δραστηριότητες: τη συντήρηση, δηλαδή τη διαρκή εξέλιξη του λογισμικού μετά τη διάταξή του, και τη λειτουργία, δηλαδή τη διασφάλιση της ομαλής εκτέλεσής του στην παραγωγή. Οι δύο αυτές δραστηριότητες εξετάζονται από κοινού επειδή, όπως αναλύεται παρακάτω, στα σύγχρονα μοντέλα DevOps τα όριά τους έχουν ουσιαστικά καταργηθεί: η παρακολούθηση της λειτουργίας είναι εκείνη που υπαγορεύει τις ενέργειες συντήρησης (Amaro, Pereira & Mira da Silva, 2025). Το στάδιο αυτό συχνά αντιπροσωπεύει το μεγαλύτερο μέρος του συνολικού κόστους μιας εφαρμογής κατά τη διάρκεια ζωής της (Sommerville, 2016).

Η βιβλιογραφία διακρίνει παραδοσιακά τέσσερις τύπους συντήρησης: τη διορθωτική (διόρθωση σφαλμάτων), την προσαρμοστική (προσαρμογή σε νέα περιβάλλοντα εκτέλεσης ή σε νέες εξαρτήσεις), τη βελτιωτική (προσθήκη νέων δυνατοτήτων αλλά και βελτίωση των υφιστάμενων, για παράδειγμα ως προς την απόδοση ή τη χρηστικότητα, χωρίς να υπάρχει κάποιο σφάλμα προς διόρθωση) και την προληπτική (αναδιάρθρωση του κώδικα για τη μείωση μελλοντικών προβλημάτων). Και οι τέσσερις τύποι εφαρμόζονται στις εφαρμογές ιστού, με χαρακτηριστικά όμως που προκύπτουν από τη συνεχή και δημόσια λειτουργία τους: η διορθωτική συντήρηση εκτελείται υπό πίεση χρόνου, καθώς ένα σφάλμα είναι αμέσως ορατό σε όλους τους χρήστες· η προσαρμοστική κυριαρχείται από τη διαχείριση των εξαρτήσεων και τη συμβατότητα με τις εκδόσεις των προγραμμάτων περιήγησης· η βελτιωτική υλοποιείται σταδιακά, σε σύντομους κύκλους, χωρίς διακοπή της υπηρεσίας· ενώ η προληπτική περιλαμβάνει την έγκαιρη εφαρμογή ενημερώσεων ασφαλείας και την αντιμετώπιση του τεχνικού χρέους, πριν αυτά εξελιχθούν σε ευπάθειες ή σε εμπόδια στην εξέλιξη της εφαρμογής. Εγκάρσια προς όλους τους τύπους διατρέχει η συνεχής παρακολούθηση της απόδοσης και της ασφάλειας.

Κρίσιμη έννοια για το στάδιο αυτό είναι η παρατηρησιμότητα (observability), δηλαδή η δυνατότητα κατανόησης της εσωτερικής κατάστασης ενός συστήματος μέσα από τα εξωτερικά του σήματα - τα αρχεία καταγραφής (logs), τις μετρικές (metrics) και την ιχνηλάτηση (tracing). Στα σύγχρονα μοντέλα DevOps, η αυστηρή διάκριση της συντήρησης ως ξεχωριστής, τελικής φάσης εξασθενεί: η λειτουργία, η παρακολούθηση και η βελτίωση εντάσσονται σε μια διαρκή, κυκλική ροή, όπου η ανατροφοδότηση από την παραγωγή τροφοδοτεί άμεσα τον επόμενο κύκλο ανάπτυξης (Shahin, Babar & Zhu, 2017). Η προσέγγιση αυτή αποκρυσταλλώνεται στη μηχανική αξιοπιστίας υπηρεσιών (Site Reliability Engineering, SRE), όρος που καθιερώθηκε από τη Google και όπου το «site» παραπέμπει στην εν λειτουργία διαδικτυακή υπηρεσία και όχι στο λογισμικό ως προϊόν. Η κεντρική της αρχή είναι ότι οι λειτουργικές ανάγκες - η παρακολούθηση, η κλιμάκωση, η απόκριση σε συμβάντα - αντιμετωπίζονται με τα εργαλεία και τις μεθόδους της μηχανικής λογισμικού, δηλαδή με αυτοματοποίηση και κώδικα, αντί με επαναλαμβανόμενες χειροκίνητες ενέργειες· έτσι, το λειτουργικό φορτίο δεν αυξάνεται γραμμικά με το μέγεθος του συστήματος.

2.3.6 Μοντέλα Οργάνωσης των Σταδίων

Πρέπει να τονιστεί ότι τα πέντε αυτά στάδια αποτελούν λογικές δραστηριότητες και όχι κατ' ανάγκη διακριτές, διαδοχικές φάσεις. Ο τρόπος με τον οποίο οργανώνονται στον χρόνο εξαρτάται από το μοντέλο διαδικασίας ανάπτυξης (development process model) που υιοθετείται. Τα καθιερωμένα μοντέλα διαδικασίας είναι το μοντέλο του καταρράκτη (waterfall), το V-Model, το αυξητικό (incremental), το επαναληπτικό (iterative), το σπειροειδές (spiral) και η προτυποποίηση (prototyping), όπως παρουσιάζονται συστηματικά στα καθιερωμένα εγχειρίδια της μηχανικής λογισμικού (Sommerville, 2016).Το πρώτο δύο από αυτά είναι αυστηρά γραμμικά: εκτελούν τα στάδια ως σαφώς οριοθετημένες, ακολουθιακές βαθμίδες, προσέγγιση κατάλληλη για έργα με σταθερές και καλά κατανοητές απαιτήσεις. Τα υπόλοιπα εισάγουν σταδιακά τη λογική της επανάληψης και της προοδευτικής παράδοσης: το αυξητικό μοντέλο παραδίδει το σύστημα σε διαδοχικά τμήματα λειτουργικότητας, το επαναληπτικό εμπλουτίζει μια αρχική εκδοχή σε διαδοχικούς κύκλους, το σπειροειδές οργανώνει τους κύκλους αυτούς γύρω από τη διαχείριση του κινδύνου, και η προτυποποίηση αξιοποιεί πρώιμα πρωτότυπα για την αποσαφήνιση ασαφών απαιτήσεων.

Διακριτές από τα μοντέλα διαδικασίας είναι οι ευέλικτες (agile) μέθοδοι και τα αντίστοιχα πλαίσια, όπως το Scrum (Cohn, 2009), ο ακραίος προγραμματισμός (eXtreme Programming) και το Kanban (Al-Saqqa, Sawalha & AbdelNabi, 2020). Τα τελευταία δεν συνιστούν μοντέλα διαδικασίας με τη στενή έννοια, αλλά πρακτικές οργάνωσης της εργασίας που στηρίζονται σε μια επαναληπτική και αυξητική λογική: αναδιατάσσουν τα ίδια στάδια σε σύντομους, επαναλαμβανόμενους κύκλους, ώστε η ανάλυση, η υλοποίηση και η δοκιμή να συμβαίνουν σχεδόν ταυτόχρονα - προσέγγιση ιδιαίτερα ταιριαστή με τη δυναμική, ταχέως μεταβαλλόμενη φύση των εφαρμογών ιστού. Στη σύγχρονη πρακτική, τα όρια μεταξύ των σταδίων θολώνουν ακόμη περισσότερο υπό την οπτική του DevOps, όπου η υλοποίηση, η διάταξη και η συντήρηση συγχωνεύονται σε μια αδιάλειπτη ροή (Pastrana et al., 2025).

2.4 Πεδία Εφαρμογής

Οι εφαρμογές ιστού καλύπτουν ένα εξαιρετικά ευρύ φάσμα βιομηχανιών και τομέων της ανθρώπινης δραστηριότητας. Στο ηλεκτρονικό εμπόριο, πλατφόρμες όπως το Amazon και το Shopify προσφέρουν καταλόγους προϊόντων, καλάθια αγορών, πύλες πληρωμών και συστήματα διαχείρισης αποθεμάτων. Στην υγειονομική περίθαλψη, συναντάμε πλατφόρμες τηλεϊατρικής, ηλεκτρονικά ιατρικά αρχεία και πύλες εξυπηρέτησης ασθενών. Στην εκπαίδευση κυριαρχούν οι πλατφόρμες τηλεκπαίδευσης, τα συστήματα διαχείρισης μάθησης και τα εργαλεία σύγχρονης διδασκαλίας. Ακολουθούν τα κοινωνικά δίκτυα με τις πλατφόρμες αλληλεπίδρασης και διαμοιρασμού περιεχομένου, οι χρηματοοικονομικές υπηρεσίες με την ηλεκτρονική τραπεζική και τις πλατφόρμες επενδύσεων, τα εργαλεία εργασιακής συνεργασίας, καθώς και οι πλατφόρμες ψυχαγωγίας και διαχείρισης περιεχομένου. Οι κλάδοι που αποκομίζουν τα μεγαλύτερα οφέλη από τις εφαρμογές ιστού είναι κατεξοχήν η τεχνολογία, το λιανικό εμπόριο, η υγεία, η χρηματοδότηση, η εκπαίδευση και η ψυχαγωγία, καθένας εκ των οποίων επιβάλλει και τις δικές του ιδιαίτερες απαιτήσεις σε απόδοση, ασφάλεια και διαθεσιμότητα.

Η διαπίστωση ότι κάθε πεδίο εφαρμογής επιβάλλει διαφορετικές προτεραιότητες δεν είναι απλώς διαισθητική, αλλά τεκμηριώνεται εμπειρικά. Οι Eckhardt, Vogelsang και Méndez Fernández (2016), αναλύοντας 530 μη-λειτουργικές απαιτήσεις από 11 βιομηχανικές προδιαγραφές, διαπίστωσαν ότι η κατανομή των ποιοτικών χαρακτηριστικών διαφέρει σημαντικά ανάλογα με το πεδίο του συστήματος - με την ασφάλεια να κυριαρχεί στα επιχειρησιακά πληροφοριακά συστήματα και την αξιοπιστία στα ενσωματωμένα - και κατέληξαν σε ένα ισχυρό επιχείρημα υπέρ της ανά-πεδίο διαχείρισης των απαιτήσεων αυτών. Στο πλαίσιο των εφαρμογών ιστού, αυτό σημαίνει ότι μια πλατφόρμα ηλεκτρονικής τραπεζικής θέτει ως ύψιστη προτεραιότητα

την ασφάλεια και τη συμμόρφωση με κανονιστικά πλαίσια, μια πλατφόρμα ηλεκτρονικού εμπορίου την επεκτασιμότητα και την απόδοση υπό αιχμές φορτίου, ενώ μια πλατφόρμα υγείας τόσο την προστασία ευαίσθητων δεδομένων όσο και τη διαθεσιμότητα. Η ετερογένεια αυτή των απαιτήσεων εξηγεί γιατί δεν υπάρχει μία και μοναδική «βέλτιστη» μέθοδο ανάπτυξης, αλλά ένα φάσμα μεθόδων και τεχνικών, η επιλογή των οποίων εξαρτάται από τη φύση του εκάστοτε πεδίου - ζήτημα κεντρικό για την παρούσα επισκόπηση.

2.5 Ο Ρόλος του Υπολογιστικού Νέφους

Το υπολογιστικό νέφος έχει μεταμορφώσει ριζικά τον τρόπο ανάπτυξης, διάταξης και λειτουργίας των εφαρμογών ιστού. Ορίζεται ως η παροχή υπολογιστικών υπηρεσιών - διακομιστών, αποθήκευσης, βάσεων δεδομένων, δικτύωσης και λογισμικού - μέσω του διαδικτύου, με μοντέλο χρέωσης βάσει της πραγματικής χρήσης (Mell & Grance, 2011). Το ίδιο πρότυπο ορίζει και τα τρία μοντέλα υπηρεσιών που υποστηρίζουν τις εφαρμογές ιστού. Το μοντέλο υποδομής ως υπηρεσίας (Infrastructure as a Service, IaaS) παρέχει πρόσβαση σε υπολογιστική υποδομή κατ' απαίτηση, αφήνοντας στον προγραμματιστή τη διαχείριση του λειτουργικού συστήματος και των εφαρμογών. Το μοντέλο πλατφόρμας ως υπηρεσίας (Platform as a Service, PaaS) προσφέρει ένα πλήρες περιβάλλον ανάπτυξης και εκτέλεσης, χωρίς την ανάγκη διαχείρισης της υποκείμενης υποδομής. Τέλος, το μοντέλο λογισμικού ως υπηρεσίας (Software as a Service, SaaS) διαθέτει ολοκληρωμένες εφαρμογές απευθείας μέσω του προγράμματος περιήγησης. Αξίζει να τονιστεί ότι η πρόσβαση μέσω του προγράμματος περιήγησης δεν είναι ο μόνος τρόπος κατανάλωσης μιας εφαρμογής SaaS: κατά κανόνα, η λειτουργικότητά της εκτίθεται και προγραμματιστικά, μέσω RESTful APIs ή διαδικτυακών υπηρεσιών γενικότερα, επιτρέποντας σε τρίτους να τη συνθέτουν με άλλες υπηρεσίες και να κατασκευάζουν νέες εφαρμογές προστιθέμενης αξίας. Η δυνατότητα αυτή συνδέεται άμεσα με τις API-driven αρχιτεκτονικές της Ενότητας 2.2.3 και αποτελεί έναν από τους βασικούς λόγους για τους οποίους το νέφος λειτούργησε ως καταλύτης για τη σύνθεση κατανεμημένων εφαρμογών ιστού.

2.5.1 Θεμελιώδη Χαρακτηριστικά και Μοντέλα Ανάπτυξης

Πέρα από τα τρία μοντέλα υπηρεσιών, ο κανονικός ορισμός του υπολογιστικού νέφους, όπως διατυπώθηκε από τους Mell και Grance (2011) στην έκδοση NIST SP 800-145, στηρίζεται σε πέντε θεμελιώδη χαρακτηριστικά που διακρίνουν μια πραγματική υπηρεσία νέφους από την απλή απομακρυσμένη φιλοξενία. Το πρώτο είναι η αυτοεξυπηρέτηση κατ' απαίτηση (on-demand self-service), όπου ο χρήστης προμηθεύεται υπολογιστικούς πόρους αυτόματα, χωρίς ανθρώπινη διαμεσολάβηση. Το δεύτερο είναι η ευρεία πρόσβαση μέσω δικτύου (broad network access), που στη νεότερη βιβλιογραφία αποδίδεται και ως πανταχού παρούσα πρόσβαση (ubiquitous access), με τους πόρους διαθέσιμους μέσα από τυποποιημένους μηχανισμούς και ετερογενείς συσκευές. Το τρίτο είναι η συγκέντρωση πόρων (resource pooling), όπου οι φυσικοί και εικονικοί πόροι μοιράζονται δυναμικά μεταξύ πολλών χρηστών. Το τέταρτο είναι η ταχεία ελαστικότητα (rapid elasticity), που επιτρέπει την άμεση κλιμάκωση προς τα πάνω ή προς τα κάτω. Το πέμπτο και τελευταίο χαρακτηριστικό είναι η μετρούμενη υπηρεσία (measured service), όπου η χρήση παρακολουθείται και χρεώνεται με διαφάνεια. Η ελαστικότητα και η μετρούμενη χρέωση είναι ιδιαίτερα κρίσιμες για τις εφαρμογές ιστού, οι οποίες αντιμετωπίζουν συχνά απρόβλεπτες αιχμές φορτίου.

Το ίδιο πρότυπο ορίζει και τέσσερα μοντέλα ανάπτυξης, τα οποία καθορίζουν το ποιος έχει πρόσβαση στην υποδομή. Το δημόσιο νέφος (public cloud) διατίθεται ανοιχτά μέσω ενός παρόχου και προσφέρει τη μέγιστη οικονομία κλίμακας. Το ιδιωτικό νέφος (private cloud) προορίζεται αποκλειστικά για έναν οργανισμό, προσφέροντας μεγαλύτερο έλεγχο και ασφάλεια. Το κοινοτικό νέφος (community cloud) μοιράζεται μεταξύ οργανισμών με κοινές ανάγκες, ενώ το υβριδικό

νέφος (hybrid cloud) συνδυάζει δύο ή περισσότερα από τα παραπάνω, επιτρέποντας, για παράδειγμα, τη διατήρηση ευαίσθητων δεδομένων σε ιδιωτική υποδομή και την αξιοποίηση του δημόσιου νέφους για τη διαχείριση αιχμών φορτίου. Για τις σύγχρονες εφαρμογές ιστού, η επιλογή του μοντέλου ανάπτυξης δεν είναι απλώς τεχνική αλλά και στρατηγική απόφαση, καθώς επηρεάζει άμεσα το κόστος, τη συμμόρφωση με κανονιστικά πλαίσια, το επίπεδο ασφάλειας και την κλιμακωσιμότητα του συστήματος.

Οι κεντρικές τεχνολογίες νέφους για τις εφαρμογές ιστού περιλαμβάνουν τα δοχεία (containers), που προσφέρουν φορητή και αυτοτελή συσκευασία των υπηρεσιών, και την ενορχήστρωσή τους μέσω του Kubernetes, που αυτοματοποιεί την κατανομή, την κλιμάκωση και τη διαχείριση του κύκλου ζωής τους, το serverless computing για την εκτέλεση κώδικα χωρίς διαχείριση διακομιστών, τις διαχειριζόμενες βάσεις δεδομένων, καθώς και τα δίκτυα διανομής περιεχομένου για τη γρήγορη παράδοση στους τελικούς χρήστες. Η ολιστική θεώρηση της cloud-native στοίβας, από την κατασκευή έως τη λειτουργία και τη συντήρηση, αποτυπώνεται στην εκτενή επισκόπηση των Deng et al. (2024), η οποία αποδομεί τον κύκλο ζωής των cloud-native εφαρμογών σε τέσσερις καταστάσεις: κατασκευή (building), ενορχήστρωση (orchestration), λειτουργία (operate) και συντήρηση (maintenance). Η διάκριση ανάμεσα στην ενορχήστρωση και τη λειτουργία αφορά το επίπεδο στο οποίο αναφέρεται καθεμία: η ενορχήστρωση καλύπτει τη διαχείριση των ίδιων των πόρων - την τοποθέτηση των δοχείων, την κατανομή της υπολογιστικής ισχύος, την αυτόματη κλιμάκωση και την αποκατάσταση μετά από αστοχία - ενώ η λειτουργία αφορά τη συμπεριφορά της εφαρμογής ως υπηρεσίας προς τον χρήστη, δηλαδή την παρακολούθηση, τη διαχείριση της κίνησης, την τήρηση των στόχων επιπέδου υπηρεσίας και την απόκριση σε συμβάντα. Η εργασία αυτή αποτελεί μία από τις πληρέστερες του τελικού συνόλου των 27 άρθρων της παρούσας επισκόπησης ως προς την κάλυψη των σταδίων υλοποίησης, διάταξης και συντήρησης, αν και η φάση της ανάλυσης απαιτήσεων και σχεδίασης καλύπτεται με μικρότερη λεπτομέρεια.

2.6 Μέθοδοι, Τεχνικές και Εργαλεία Ανάπτυξης

Η παρούσα ενότητα παρουσιάζει τις κύριες μεθόδους, τεχνικές και εργαλεία που χρησιμοποιούνται στην ανάπτυξη εφαρμογών ιστού, αποτελώντας το θεωρητικό υπόβαθρο για τη συγκριτική ανάλυση του Κεφαλαίου 4. Η παρουσίαση οργανώνεται σε έξι οικογένειες μεθόδων, οι οποίες διακρίνονται με βάση τον κύριο προσανατολισμό τους και αποτελούν τη μονάδα ανάλυσης του Κεφαλαίου 4: τις παραδοσιακές και ευέλικτες μεθόδους, με διαδικαστικό προσανατολισμό (Ενότητα 2.6.1), τη μοντελοκεντρική μηχανική ιστού (Model-Driven Web Engineering, Ενότητα 2.6.2), το DevOps και τη συνεχή ενσωμάτωση και παράδοση, με λειτουργικό προσανατολισμό (Ενότητα 2.6.3), την ανάπτυξη καθοδηγούμενη από δοκιμές και τις αυτοματοποιημένες δοκιμές, με πρακτικοκεντρικό προσανατολισμό (Ενότητα 2.6.4), τις αρχιτεκτονικές προσεγγίσεις των microservices και των υπηρεσιοστρεφών αρχιτεκτονικών (Ενότητες 2.6.5 και 2.6.6) και, τέλος, το τεχνολογικό επίπεδο των γλωσσών, των πλαισίων και της ασφάλειας (Ενότητες 2.6.7 έως 2.6.9). Οι εφαρμογές ιστού παρουσιάζουν ιδιαίτερες ανάγκες -ταχεία μεταβολή απαιτήσεων, συνεχή διαθεσιμότητα, ετερογένεια τεχνολογιών, απαιτήσεις ασφάλειας- που έχουν οδηγήσει στην ανάπτυξη εξειδικευμένων μεθοδολογικών προσεγγίσεων. Οι προσεγγίσεις αυτές δεν είναι αμοιβαία αποκλειόμενες· αντιθέτως, στην πράξη συχνά συνδυάζονται, με μια ομάδα να υιοθετεί, για παράδειγμα, Agile διαχείριση, microservices αρχιτεκτονική και DevOps πρακτικές ταυτόχρονα.

2.6.1 Παραδοσιακές και Agile Οικογένειες Μεθόδων

Η μέθοδος Waterfall αποτέλεσε την πρώτη δομημένη προσέγγιση ανάπτυξης λογισμικού. Δύο χαρακτηριστικά την ορίζουν: πρώτον, οι φάσεις - απαιτήσεις, σχεδίαση, υλοποίηση, δοκιμές,

συντήρηση - εκτελούνται ακολουθιακά, όπου η καθεμία ολοκληρώνεται πλήρως πριν ξεκινήσει η επόμενη και με παραδοτέο που λειτουργεί ως είσοδος για εκείνη που ακολουθεί· δεύτερον, ολόκληρος ο κύκλος πρέπει να διανυθεί μέχρι τέλους πριν ξεκινήσει ένας νέος. Το πρώτο χαρακτηριστικό καθιστά την προσέγγιση αποτελεσματική σε έργα με σταθερές και σαφώς καθορισμένες απαιτήσεις, καθώς επιτρέπει αυστηρό προγραμματισμό και τεκμηρίωση. Το δεύτερο, όμως, είναι η πηγή της ακαμψίας της: μια αλλαγή στις απαιτήσεις που προκύπτει μετά την ολοκλήρωση της αντίστοιχης φάσης δεν μπορεί να ενσωματωθεί παρά μόνο στον επόμενο πλήρη κύκλο, με αποτέλεσμα ο χρόνος απόκρισης σε μια αλλαγή να ισούται με τη διάρκεια ενός ολόκληρου κύκλου. Στα δυναμικά και ταχέως μεταβαλλόμενα περιβάλλοντα του ιστού, όπου οι απαιτήσεις μεταβάλλονται εντός ορισμένων εβδομάδων, η καθυστέρηση αυτή καθιστά τη μέθοδο πρακτικά ακατάλληλη.

Ως απάντηση σε αυτόν τον περιορισμό, οι Agile μέθοδοι άλλαξαν τον τρόπο ανάπτυξης, θέτοντας ως πρωταρχικές αξίες τη συνεργασία, την ανταπόκριση στις αλλαγές και την τακτική παράδοση λειτουργικού λογισμικού σε σύντομους κύκλους. Το Scrum, το πλέον διαδεδομένο Agile πλαίσιο, οργανώνει την ανάπτυξη σε επαναληπτικούς κύκλους σταθερής διάρκειας, τα sprints, με σαφείς ρόλους και τελετουργίες.

Η εμπειρική βάση των Agile μεθόδων είναι ισχυρή. Η εκτενής συστηματική ανασκόπηση των Dybå και Dingsøyr (2008), που συνέθεσε τα ευρήματα από τριάντα έξι εμπειρικές μελέτες, τεκμηρίωσε τα οφέλη αλλά και τους περιορισμούς των Agile προσεγγίσεων, ενώ η εμπειρική μελέτη των Begel και Nagappan (2007) σε βιομηχανικό περιβάλλον επιβεβαίωσε ότι το Scrum είναι η Agile μέθοδος με τον υψηλότερο βαθμό υιοθέτησης μεταξύ των προγραμματιστών που συμμετείχαν στην έρευνα. Ειδικά για τις εφαρμογές ιστού, η προσαρμοστικότητα των Agile μεθόδων τις καθιστά ιδιαίτερα κατάλληλες, δεδομένης της συχνότητας με την οποία μεταβάλλονται οι απαιτήσεις των χρηστών.

2.6.2 Model-Driven Web Engineering

Οι μέθοδοι Model-Driven Web Engineering (MDWE) εισάγουν έναν μοντελοκεντρικό τρόπο σχεδίασης, όπου η εφαρμογή περιγράφεται μέσα από χωριστά, υψηλού επιπέδου μοντέλα για τα δεδομένα, την πλοήγηση, την παρουσίαση και την επιχειρησιακή λογική, από τα οποία μπορεί στη συνέχεια να παραχθεί αυτόματα κώδικας. Στις κυριότερες μεθόδους συγκαταλέγονται η OOHDM (Object-Oriented Hypermedia Design Method), η οποία οργανώνει τη σχεδίαση σε τέσσερις διαδοχικές φάσεις - εννοιολογική σχεδίαση, σχεδίαση της πλοήγησης, σχεδίαση της αφηρημένης διεπαφής και υλοποίηση (Schwabe & Rossi, 1998)· η UWE (UML-based Web Engineering), η οποία δεν εισάγει νέα γλώσσα μοντελοποίησης αλλά αξιοποιεί ένα εξειδικευμένο προφίλ της UML, ώστε τα μοντέλα να παραμένουν συμβατά με τα καθιερωμένα εργαλεία της μηχανικής λογισμικού (Koch & Kraus, 2002)· και η WebML (Web Modeling Language), μια οπτική γλώσσα για τον ορισμό του περιεχομένου και της πλοήγησης δεδομενοκεντρικών εφαρμογών (Ceri, Fraternali & Bongio, 2000), η οποία εξελίχθηκε αργότερα στο IFML (Interaction Flow Modeling Language), πρότυπο που υιοθετήθηκε επίσημα από τον οργανισμό OMG το 2014 και γενίκευσε τη λογική της πέρα από τον ιστό, σε εφαρμογές επιφάνειας εργασίας και κινητών συσκευών.

Η εκτενές επισκόπηση των Schwinger et al. (2008) συνέκρινε επτά τέτοιες προσεγγίσεις με βάση έναν λεπτομερή κατάλογο άνω των τριάντα κριτηρίων. Το συμπέρασμά του δεν αναδεικνύει μία μέθοδο ως καθολικά ανώτερη: καθεμία υπερέχει σε διαφορετικό υποσύνολο κριτηρίων - ορισμένες καλύπτουν πληρέστερα τη μοντελοποίηση του περιεχομένου και της πλοήγησης, άλλες τη σχεδίαση της παρουσίασης, και άλλες τις πτυχές της προσαρμοστικότητας στο εκάστοτε πλαίσιο χρήσης - ενώ καμία δεν καλύπτει ισόρροπα το σύνολό τους. Η διαπίστωση αυτή προοικονομεί το κεντρικό εύρημα της παρούσας επισκόπησης σε επίπεδο σταδίων SDLC.

Συμπληρωματικά, η ιστορική επισκόπηση των Rossi et al. (2016) χαρτογράφησε την εξέλιξη είκοσι πέντε ετών του πεδίου.

Παρά τη σημαντική ακαδημαϊκή παραγωγή, η βιομηχανική υιοθέτηση των MDWE μεθόδων παραμένει περιορισμένη. Η έρευνα των Linaje et al. (2019) έδειξε ότι στην πράξη οι εταιρείες προτιμούν υβριδικές ή εμπειρικές προσεγγίσεις έναντι των αυστηρών ακαδημαϊκών μεθόδων, αναδεικνύοντας το χάσμα ανάμεσα στη θεωρία και την πράξη, το οποίο αποτελεί ένα από τα κεντρικά ερευνητικά κενά της παρούσας εργασίας.

2.6.3 DevOps και Συνεχής Ενσωμάτωση / Παράδοση

Το DevOps δεν αποτελεί απλώς μια μέθοδο αλλά μια ευρύτερη κουλτούρα συνεργασίας ανάμεσα στις ομάδες ανάπτυξης και λειτουργίας, με στόχο τη συνεχή και αξιόπιστη παράδοση λογισμικού (Leite et al., 2019). Στον πυρήνα του βρίσκεται ο αγωγός (pipeline) συνεχούς ενσωμάτωσης και παράδοσης, όπου κάθε αλλαγή στον κώδικα ενεργοποιεί αυτόματα τη μεταγλώττιση και την εκτέλεση των δοκιμών. Εφόσον τα βήματα αυτά ολοκληρωθούν επιτυχώς, η έκδοση προωθείται σε περιβάλλον staging και καθίσταται έτοιμη προς διάταξη. Στο σημείο αυτό πρέπει να διευκρινιστεί η διάκριση ανάμεσα σε δύο συχνά συγχεόμενες έννοιες: στη συνεχή παράδοση (continuous delivery), η διαδικασία αυτοματοποιείται μέχρι το σημείο όπου κάθε επικυρωμένη έκδοση είναι ανά πάσα στιγμή διαθέσιμη προς εγκατάσταση, αλλά η τελική μετάβαση στην παραγωγή παραμένει μια συνειδητή, ανθρώπινη απόφαση· στη συνεχή διάταξη (continuous deployment), το τελευταίο αυτό βήμα αυτοματοποιείται επίσης, και κάθε αλλαγή που περνά τους ελέγχους φτάνει αυτόματα στους τελικούς χρήστες, χωρίς ενδιάμεση έγκριση.

Οι συνιστώσες ενός τυπικού αγωγού περιλαμβάνουν τη συνεχή ενσωμάτωση, τη συνεχή παράδοση, τη διαχείριση της υποδομής ως κώδικα και τη συνεχή παρακολούθηση σε παραγωγή. Οι δύο τελευταίες αξίζουν ιδιαίτερης μνείας, καθώς είναι εκείνες που καθιστούν τον αγωγό αξιόπιστο στην πράξη. Η υποδομή ως κώδικας (Infrastructure as Code) συνίσταται στην περιγραφή του συνόλου της υποδομής - διακομιστών, δικτύων, βάσεων δεδομένων και παραμετροποιήσεων - σε δηλωτικά αρχεία που τηρούνται στο ίδιο σύστημα ελέγχου εκδόσεων με τον κώδικα της εφαρμογής. Η πρακτική αυτή είναι σήμερα απαραίτητη επειδή εξαλείφει την απόκλιση ανάμεσα στα περιβάλλοντα ανάπτυξης, δοκιμών και παραγωγής, που αποτελεί κλασική πηγή σφαλμάτων, και καθιστά κάθε αλλαγή της υποδομής ιχνηλατήσιμη, επιθεωρήσιμη και αναστρέψιμη, όπως ακριβώς και μια αλλαγή κώδικα. Η συνεχής παρακολούθηση σε παραγωγή, με τη σειρά της, κλείνει τον βρόχο ανατροφοδότησης: χωρίς αυτήν, η ομάδα δεν μπορεί να γνωρίζει αν μια διάταξη πέτυχε ή απέτυχε παρά μόνο όταν το αναφέρουν οι χρήστες - γεγονός που καθιστά αδύνατη τόσο την ταχεία αποκατάσταση όσο και τη μέτρηση των ίδιων των επιδόσεων της παράδοσης. Η πλέον αναφερόμενη συστηματική ανασκόπηση του χώρου, των Shahin, Babar και Zhu (2017), χαρτογράφησε εκτενώς τις προσεγγίσεις, τα εργαλεία και τις προκλήσεις του CI/CD.

Η σημασία των πρακτικών DevOps για την ταχύτητα και την αξιοπιστία της παράδοσης έχει τεκμηριωθεί στο θεμελιώδες έργο των Forsgren, Humble και Kim (2018), το οποίο εισήγαγε τις τέσσερις βασικές μετρικές επίδοσης (DORA metrics). Τα συγκεκριμένα ποσοτικά μεγέθη βελτίωσης - όπως η δραματικά υψηλότερη συχνότητα διατάξεων και ο μικρότερος χρόνος αποκατάστασης που παρατηρούνται στους ώριμους οργανισμούς - προέρχονται από τις ετήσιες εκθέσεις State of DevOps του οργανισμού DORA, οι οποίες αποτελούν διακριτή πηγή από το ίδιο το βιβλίο.

Οι τέσσερις αυτές μετρικές, γνωστές ως DORA metrics από την ερευνητική ομάδα DevOps Research and Assessment, ομαδοποιούνται σε δύο άξονες που αποτυπώνουν την ισορροπία μεταξύ ταχύτητας (velocity) και ευστάθειας (stability). Στον άξονα της ταχύτητας ανήκουν η συχνότητα διατάξεων (deployment frequency), δηλαδή το πόσο συχνά ένας

οργανισμός διαθέτει επιτυχώς κώδικα στην παραγωγή, και ο χρόνος υλοποίησης αλλαγών (lead time for changes), δηλαδή ο χρόνος που μεσολαβεί από την υποβολή μιας αλλαγής έως τη λειτουργία της στην παραγωγή. Στον άξονα της ευστάθειας ανήκουν το ποσοστό αποτυχημένων αλλαγών (change failure rate), δηλαδή το ποσοστό των διατάξεων που προκαλούν αστοχία στην παραγωγή, και ο χρόνος αποκατάστασης της υπηρεσίας (time to restore service ή mean time to recovery), δηλαδή το πόσο γρήγορα επανέρχεται σε κανονική λειτουργία η πληττόμενη υπηρεσία μετά από μια αστοχία. Η ιδιαίτερη αξία του μοντέλου έγκειται στο ότι οι δύο άξονες αντιμετωπίζονται από κοινού: οι οργανισμοί υψηλών επιδόσεων επιτυγχάνουν ταυτόχρονα υψηλή ταχύτητα και υψηλή ευστάθεια, αναιρώντας την παραδοσιακή αντίληψη ότι η μία επιτυγχάνεται εις βάρος της άλλης. Για τις εφαρμογές ιστού, όπου η συχνή και αξιόπιστη παράδοση είναι κρίσιμη, οι μετρικές αυτές έχουν καθιερωθεί ως το κοινό λεξιλόγιο αξιολόγησης της αποδοτικότητας της παράδοσης.

2.6.4 Test-Driven Development και Αυτοματοποιημένες Δοκιμές

Το Test-Driven Development είναι μια τεχνική ανάπτυξης όπου ο προγραμματιστής γράφει πρώτα τις αυτόματες δοκιμές και στη συνέχεια τον ελάχιστο κώδικα που τις ικανοποιεί, ακολουθώντας τον επαναληπτικό κύκλο «κόκκινο - πράσινο - αναδιάρθρωση». Ο κύκλος αυτός αποτελείται από τρία διακριτά βήματα: στο πρώτο, ο προγραμματιστής γράφει μια δοκιμή για μια λειτουργικότητα που δεν έχει ακόμη υλοποιηθεί, η οποία αναπόφευκτα αποτυγχάνει (κόκκινο)· στο δεύτερο, γράφει τον απλούστερο δυνατό κώδικα που την ικανοποιεί, χωρίς να ενδιαφέρεται για την κομψότητα της λύσης, ώστε η δοκιμή να περάσει (πράσινο)· και στο τρίτο, προβαίνει σε αναδιάρθρωση (refactoring) του κώδικα που μόλις έγραψε, βελτιώνοντας τη δομή και αφαιρώντας τις επαναλήψεις, έχοντας πλέον ως δίχτυ ασφαλείας τη δοκιμή που εξασφαλίζει ότι η συμπεριφορά παραμένει αμετάβλητη. Ο κύκλος επαναλαμβάνεται για κάθε νέα λειτουργικότητα, σε πολύ σύντομα διαστήματα.

Η εμπειρική επίδραση του TDD στην ποιότητα και την παραγωγικότητα έχει μελετηθεί εκτενώς: η μετα-ανάλυση των Rafique και Mišić (2013) και η συστηματική ανασκόπηση των Bissi, Neto και Emer (2016) συγκλίνουν στο συμπέρασμα ότι το TDD τείνει να βελτιώνει την εξωτερική ποιότητα του λογισμικού, με πιο σύνθετη και λιγότερο ξεκάθαρη εικόνα ως προς την παραγωγικότητα. Για τις εφαρμογές ιστού, τα επίπεδα δοκιμών εκτείνονται από τις δοκιμές μονάδας, που ελέγχουν μεμονωμένες συναρτήσεις, έως τις από άκρο σε άκρο δοκιμές που προσομοιώνουν την αλληλεπίδραση του χρήστη με εργαλεία, όπως το Selenium και το Cypress, καθώς και τις εξειδικευμένες δοκιμές ασφαλείας.

2.6.5 Αρχιτεκτονική Microservices

Η αρχιτεκτονική microservices δεν συνιστά αυτή καθαυτή μέθοδο ανάπτυξης, αλλά ένα αρχιτεκτονικό στυλ. Εξετάζεται ωστόσο στην παρούσα επισκόπηση επειδή, σε αντίθεση με ένα απλό σχεδιαστικό πρότυπο, η υιοθέτησή της επιβάλλει συγκεκριμένες πρακτικές σε όλα σχεδόν τα στάδια του κύκλου ζωής: υπαγορεύει τον τρόπο οργάνωσης των ομάδων, τη στρατηγική δοκιμών, τον μηχανισμό διάταξης και τις απαιτήσεις παρακολούθησης. Η βιβλιογραφία εξάλλου, την πραγματεύεται συστηματικά μαζί με τις μεθόδους ανάπτυξης - χαρακτηριστικά, οι Waseem, Liang και Shahin (2020) χαρτογραφούν τη σχέση της με το DevOps ως ενιαίο αντικείμενο μελέτης.

Η αρχιτεκτονική αυτή διασπά μια εφαρμογή σε ένα σύνολο μικρών, ανεξάρτητων υπηρεσιών, καθεμία από τις οποίες υλοποιεί μια συγκεκριμένη επιχειρησιακή δυνατότητα, αναπτύσσεται ανεξάρτητα και επικοινωνεί με τις υπόλοιπες μέσω ελαφρών διεπαφών, συνήθως RESTful API βασισμένων στο HTTP (Velepucha & Flores, 2023). Τα βασικά πλεονεκτήματα της προσέγγισης είναι η ανεξάρτητη κλιμάκωση κάθε υπηρεσίας ανάλογα με το φορτίο της, η ανεξαρτησία τεχνολογικής στοίβας που επιτρέπει σε κάθε ομάδα να επιλέγει τα καταλληλότερα

εργαλεία, και η ανθεκτικότητα, καθώς η αστοχία μιας υπηρεσίας δεν καταρρέει αναγκαστικά ολόκληρο το σύστημα. Ωστόσο, η εμπειρική μελέτη των Zhou et al. (2023) σε είκοσι πραγματικές εταιρείες ανέδειξε ότι η διαχείριση των κατανεμημένων δεδομένων και η παρατηρησιμότητα - η δυνατότητα παρακολούθησης και κατανόησης της συμπεριφοράς ενός κατανεμημένου συστήματος - αποτελούν τα μεγαλύτερα πρακτικά εμπόδια στην υιοθέτηση της αρχιτεκτονικής των μικρο-υπηρεσιών.

Πέρα από τη διάσπαση σε υπηρεσίες, η αρχιτεκτονική microservices εξαρτάται καθοριστικά από τον τρόπο επικοινωνίας μεταξύ τους. Η επικοινωνία μπορεί να είναι σύγχρονη, τυπικά μέσω REST κλήσεων πάνω από HTTP, όπου η μία υπηρεσία περιμένει την απάντηση της άλλης, ή ασύγχρονη, μέσω ουρών μηνυμάτων και μεσισμικού διαμεσολάβησης (message brokers), όπου οι υπηρεσίες ανταλλάσσουν μηνύματα χωρίς να μπλοκάρουν η μία την άλλη. Για τη διαχείριση της πολυπλοκότητας που προκύπτει από το πλήθος των υπηρεσιών, έχουν καθιερωθεί ορισμένα υποστηρικτικά μοτίβα σχεδίασης: η πύλη API (API gateway) λειτουργεί ως ενιαίο σημείο εισόδου που δρομολογεί τα αιτήματα προς τις κατάλληλες υπηρεσίες, και αναλαμβάνει διατμηματικά ζητήματα, όπως ο έλεγχος ταυτότητας· η ανακάλυψη υπηρεσιών (service discovery) επιτρέπει σε κάθε υπηρεσία να εντοπίζει δυναμικά τις υπόλοιπες σε ένα μεταβαλλόμενο περιβάλλον· και τα μοτίβα ανθεκτικότητας, όπως ο διακόπτης κυκλώματος (circuit breaker), αποτρέπουν την αλυσιδωτή διάδοση των αστοχιών. Το τελευταίο μοτίβο είναι ιδιαίτερα κρίσιμο ακριβώς λόγω της κατανεμημένης φύσης του συστήματος: όταν μια υπηρεσία καθυστερεί να αποκριθεί, οι υπηρεσίες που την καλούν δεσμεύουν πόρους περιμένοντας, εξαντλούν σταδιακά τις δεξαμενές συνδέσεών τους και καθίστανται και οι ίδιες μη διαθέσιμες, μεταδίδοντας την αστοχία προς τα πάνω σε ολόκληρη την αλυσίδα κλήσεων. Ο διακόπτης κυκλώματος ανιχνεύει το πρόβλημα, διακόπτει προσωρινά τις κλήσεις προς την προβληματική υπηρεσία και επιστρέφει άμεσα μια προκαθορισμένη απόκριση, περιορίζοντας έτσι την αστοχία στο σημείο προέλευσής της.

Η ανάπτυξη και ενορχήστρωση των υπηρεσιών αυτών βασίζεται σήμερα κατά κανόνα σε τεχνολογίες δοχειοποίησης (containers) και σε πλατφόρμες ενορχήστρωσης όπως το Kubernetes. Οι πλατφόρμες αυτές δεν καλύπτουν ολόκληρο τον κύκλο ζωής των υπηρεσιών, αλλά εστιάζουν στα στάδια της διάταξης και της λειτουργίας: αυτοματοποιούν την κατανομή των υπηρεσιών στους διαθέσιμους πόρους, την κλιμάκωσή τους υπό φορτίο, την αντικατάσταση των αστοχούντων στιγμιοτύπων και τη σταδιακή αναβάθμισή τους, αφήνοντας εκτός εμβέλειας τα στάδια της ανάλυσης απαιτήσεων, της σχεδίασης και της υλοποίησης.

2.6.6 Αρχιτεκτονικές Προσανατολισμένες σε Υπηρεσίες: SOA, REST και API-Driven

Στον πυρήνα της επικοινωνίας των σύγχρονων εφαρμογών ιστού βρίσκεται η έννοια της διεπαφής προγραμματισμού εφαρμογών (Application Programming Interface, API). Το ευρύτερο πλαίσιο μέσα στο οποίο καθιερώθηκε η έννοια αυτή είναι η αρχιτεκτονική προσανατολισμένη σε υπηρεσίες (Service-Oriented Architecture, SOA), η οποία προτείνει τη δόμηση ενός συστήματος ως συνόλου αυτοτελών, επαναχρησιμοποιήσιμων υπηρεσιών που εκθέτουν τη λειτουργικότητά τους μέσα από καλά ορισμένα συμβόλαια, ανεξάρτητα από την τεχνολογία υλοποίησής τους. Η SOA γνώρισε δύο βασικές υλοποιήσεις. Η πρώτη, η λεγόμενη «βαριά» ή WS-* προσέγγιση, στηρίζεται στο πρωτόκολλο SOAP και σε ένα εκτενές σύνολο συνοδευτικών προτύπων, με τις διεπαφές να περιγράφονται τυπικά σε WSDL και την επικοινωνία να διαμεσολαβείται συχνά από έναν δίαυλο επιχειρησιακών υπηρεσιών (Enterprise Service Bus). Η δεύτερη στηρίζεται στο REST και, χάρη στην απλότητά της, επικράτησε στον χώρο των εφαρμογών ιστού. Η αρχιτεκτονική των μικρο-υπηρεσιών της προηγούμενης ενότητας μπορεί να ιδωθεί ως η εξέλιξη της SOA σε λεπτότερη κοκκίωση (granularity), με αποκεντρωμένη διακυβέρνηση και χωρίς κεντρικό δίαυλο διαμεσολάβησης.

Το REST (Representational State Transfer) εισήγαγε ο Roy Fielding στη διδακτορική του διατριβή το 2000 (Fielding, 2000). Δεν είναι πρωτόκολλο αλλά αρχιτεκτονικό στυλ, οριζόμενο από ένα σύνολο περιορισμών: τον διαχωρισμό πελάτη-διακομιστή, την απουσία κατάστασης (statelessness), όπου κάθε αίτημα περιέχει όλη την πληροφορία που χρειάζεται ο διακομιστής, τη δυνατότητα προσωρινής αποθήκευσης (cacheability), την ενιαία διεπαφή, το πολυεπίπεδο σύστημα και τον προαιρετικό κώδικα κατ' απαίτηση. Σε μια RESTful διεπαφή, κάθε πόρος αναπαρίσταται από μια διεύθυνση URL και οι πελάτες αλληλεπιδρούν με αυτόν μέσω των τυποποιημένων μεθόδων HTTP (GET, POST, PUT, DELETE), οι οποίες αντιστοιχούν φυσικά στις λειτουργίες CRUD (Create, Read, Update, Delete), δηλαδή στις τέσσερις θεμελιώδεις πράξεις διαχείρισης δεδομένων: δημιουργία, ανάγνωση, ενημέρωση και διαγραφή. Η απλότητα, η επεκτασιμότητα και η φιλικότητα προς την προσωρινή αποθήκευση καθιέρωσαν το REST ως το de facto πρότυπο για τις διαδικτυακές υπηρεσίες επί δύο δεκαετίες.

Παρά την κυριαρχία του, το REST εμφανίζει ορισμένους περιορισμούς που έγιναν εντονότεροι με την άνοδο των σύνθετων, δεδομενοκεντρικών εφαρμογών και της κινητής χρήσης, με κυριότερους την υπερβολική ανάκτηση δεδομένων (over-fetching), όπου ο πελάτης λαμβάνει περισσότερα δεδομένα από όσα χρειάζεται, και την ελλιπή ανάκτηση (under-fetching), όπου απαιτούνται πολλαπλά αιτήματα σε διαφορετικά σημεία για τη συγκέντρωση της απαραίτητης πληροφορίας. Ως απάντηση σε αυτούς τους περιορισμούς, το GraphQL, το οποίο δημοσιοποιήθηκε ως γλώσσα επερώτησης από τη Facebook το 2015, προσφέρει μια εναλλακτική προσέγγιση: εκθέτει ένα ενιαίο σημείο πρόσβασης, όπου ο πελάτης προσδιορίζει με ακρίβεια τα πεδία που χρειάζεται μέσω μιας ισχυρά τυποποιημένης επερώτησης, και ο διακομιστής επιστρέφει ακριβώς αυτά τα δεδομένα σε ένα και μόνο αίτημα. Πρέπει να τονιστεί ότι η απλοποίηση αυτή αφορά την πλευρά του πελάτη και δεν καταργεί την υποκείμενη πολυπλοκότητα: ο διακομιστής GraphQL λειτουργεί ως επίπεδο σύνθεσης, το οποίο αναλύει την επερώτηση και εκτελεί εσωτερικά τις απαραίτητες κλήσεις - συχνά προς τα ίδια RESTful APIs ή προς επιμέρους μικρο-υπηρεσίες που διαχειρίζονται τμήματα των ζητούμενων δεδομένων - συναρμολογώντας στη συνέχεια το αποτέλεσμα. Η πολυπλοκότητα, δηλαδή, μετατοπίζεται από τον πελάτη στον διακομιστή αντί να εξαλείφεται. Οι δύο προσεγγίσεις δεν είναι αμοιβαία αποκλειόμενες· στην πράξη, πολλές ομάδες χρησιμοποιούν το REST για σταθερές διεπαφές πόρων όπου η προσωρινή αποθήκευση HTTP προσφέρει ουσιαστικό όφελος, και το GraphQL για την ευέλικτη συγκέντρωση δεδομένων που απαιτούν διαφορετικοί τύποι πελατών. Η επιλογή μεταξύ τους αποτελεί μία από τις καθοριστικές αρχιτεκτονικές αποφάσεις στη σχεδίαση σύγχρονων εφαρμογών ιστού.

2.6.7 Γλώσσες Προγραμματισμού

Στο επίπεδο της υλοποίησης, η επιλογή της γλώσσας προγραμματισμού είναι καθοριστική. Η JavaScript κατέχει μια μοναδική θέση, καθώς είναι η μόνη γλώσσα που εκτελείται εγγενώς σε κάθε πρόγραμμα περιήγησης και, μέσω της πλατφόρμας Node.js, επεκτείνεται και στην πλευρά του διακομιστή, επιτρέποντας την ανάπτυξη ολόκληρης της εφαρμογής με μία γλώσσα. Σύμφωνα με την επισκόπηση Stack Overflow Developer Survey (2024), η JavaScript παραμένει η πλέον χρησιμοποιούμενη γλώσσα, με ποσοστό περίπου 62%, ακολουθούμενη από τα HTML/CSS με 53% και την Python με 51%.

Στενά συνδεδεμένη με τη JavaScript είναι η TypeScript, η οποία δεν αποτελεί ανεξάρτητη γλώσσα αλλά αυστηρό υπερσύνολό της: κάθε έγκυρο πρόγραμμα JavaScript είναι και έγκυρο πρόγραμμα TypeScript, ενώ η τελευταία προσθέτει ένα προαιρετικό στατικό σύστημα τύπων, το οποίο ελέγχεται κατά τη μεταγλώττιση και στη συνέχεια αφαιρείται, παράγοντας απλή JavaScript προς εκτέλεση. Η δημοτικότητά της αυξάνεται σταθερά σε βάρος της σκέτης JavaScript, ιδίως σε έργα μεγάλης κλίμακας: στην ίδια επισκόπηση Stack Overflow Developer Survey (2024) καταγράφεται ως η πέμπτη πλέον χρησιμοποιούμενη γλώσσα, με ποσοστό περίπου 38%, ενώ

αρκετά σύγχρονα πλαίσια - με χαρακτηριστικότερο το Angular - την υιοθετούν ως προεπιλεγμένη γλώσσα ανάπτυξης.

Ο ακόλουθος πίνακας συνοψίζει τα χαρακτηριστικά των κύριων γλωσσών ανάπτυξης εφαρμογών ιστού, διακρίνοντας τις τυπικές περιοχές χρήσης από τα αντιπροσωπευτικά πλαίσια ανάπτυξης κάθε γλώσσας.

Πίνακας 2.1: Συγκριτική επισκόπηση κύριων γλωσσών ανάπτυξης εφαρμογών ιστού. Πηγή: επεξεργασία του συγγραφέα βάσει Stack Overflow (2024) και Doğan et al. (2014).

Γλώσσα	**Πλευρά**	**Τυπικές Χρήσεις**	**Αντιπροσωπευτικά Πλαίσια**	**Χαρακτηριστικά**
JavaScript	Front-end & Back-end	Διαδραστικές διεπαφές, APIs	React, Angular, Vue.js, Node.js/Express	Εγγενής στο πρόγραμμα περιήγησης, ασύγχρονη
TypeScript	Front-end & Back-end	Εφαρμογές μεγάλης κλίμακας	Angular, React, NestJS	Στατικό σύστημα τύπων πάνω στη JavaScript
Python	Back-end	Δεδομενοκεντρικές εφαρμογές, υπηρεσίες ανάλυσης	Django, Flask, FastAPI	Αναγνωσιμότητα, ταχεία ανάπτυξη
Java	Back-end	Επιχειρησιακά συστήματα μεγάλης κλίμακας	Spring Boot, Jakarta EE	Κλιμακωσιμότητα, ισχυρή τυποποίηση
PHP	Back-end	Συστήματα διαχείρισης περιεχομένου, ιστότοποι	Laravel, Symfony, WordPress	Ευρεία διάδοση σε CMS

Ένα ενδιαφέρον ερώτημα αφορά την επίδραση της ίδιας της γλώσσας στην ποιότητα του κώδικα. Η συστηματική σύγκριση των Bogner και Merkel (2022), που ανέλυσε 604 έργα στο GitHub με πάνω από δεκαέξι εκατομμύρια γραμμές κώδικα, διαπίστωσε ότι οι εφαρμογές σε TypeScript εμφανίζουν καλύτερη ποιότητα και κατανοησιμότητα κώδικα - λιγότερα code smells και χαμηλότερη γνωσιακή πολυπλοκότητα - σε σχέση με τις αντίστοιχες σε JavaScript. Το εύρημα αυτό, ωστόσο, χρειάζεται προσεκτική ερμηνεία: η ίδια μελέτη παρατήρησε υψηλότερο λόγο commits διόρθωσης σφαλμάτων στα έργα TypeScript, γεγονός που υποδηλώνει ότι η σχέση ανάμεσα στο στατικό σύστημα τύπων και την αποφυγή σφαλμάτων είναι πιο σύνθετη από μια απλή, γραμμική βελτίωση.

2.6.8 Frameworks

2.6.8.1 Front-End Frameworks

Στην πλευρά της παρουσίασης κυριαρχούν τρία frameworks: το React, που αναπτύχθηκε από τη Meta και αξιοποιεί έναν εικονικό DOM για αποδοτική ενημέρωση της διεπαφής· το Angular, ένα ολοκληρωμένο framework της Google βασισμένο στην TypeScript· και το Vue.js, γνωστό για την ήπια καμπύλη εκμάθησης και την προοδευτική υιοθέτησή του. Η συγκριτική αξιολόγηση των τριών με βάση το πρότυπο ISO/IEC 25010, στην εργασία των Loja και Maita (2024), έδειξε ότι κάθε framework παρουσιάζει διακριτά πλεονεκτήματα και αδυναμίες ως προς τη χρονική συμπεριφορά και την αξιοποίηση των πόρων, χωρίς κάποιο να υπερτερεί καθολικά - η καταλληλότητά του εξαρτάται από τις συγκεκριμένες απαιτήσεις απόδοσης της εφαρμογής.

Πίνακας 2.2: Συγκριτική επισκόπηση κύριων front-end frameworks. Πηγή: επεξεργασία του συγγραφέα βάσει Loja & Maita (2024).

Framework	**Φορέας**	**Βασικό Χαρακτηριστικό**	**Καταλληλότητα**
React	Meta	Virtual DOM, component-based	Σύνθετες, δυναμικές διεπαφές
Angular	Google	Ολοκληρωμένο, TypeScript-first	Μεγάλες enterprise εφαρμογές
Vue.js	Ανοιχτή κοινότητα	Προοδευτικό, ήπια καμπύλη	Μικρά έως μεσαία έργα

2.6.8.2 Back-End Frameworks

Στην πλευρά του διακομιστή, η επιλογή του framework συνδέεται στενά με τη γλώσσα υλοποίησης και καθορίζει τον τρόπο οργάνωσης της επιχειρησιακής λογικής, της πρόσβασης στα δεδομένα και της έκθεσης των διεπαφών προγραμματισμού. Παρά την ποικιλία των διαθέσιμων επιλογών, διακρίνονται δύο βασικές φιλοσοφίες σχεδίασης.

Η πρώτη είναι εκείνη των ολοκληρωμένων frameworks, όπως το Django για την Python, το Spring Boot για την Java, το Laravel για την PHP και το Ruby on Rails. Τα frameworks αυτά παρέχουν εξ ορισμού ένα πλήρες σύνολο συστατικών - αντιστοίχιση αντικειμένων-σχέσεων (object-relational mapping), σύστημα αυθεντικοποίησης και εξουσιοδότησης, μηχανισμούς μετανάστευσης της βάσης δεδομένων, διαχείριση διαμόρφωσης και μηχανή προτύπων - ακολουθώντας την αρχή της σύμβασης έναντι της παραμετροποίησης (convention over configuration). Η μηχανή προτύπων συνδυάζει πρότυπα παρουσίασης με δεδομένα και παράγει το τελικό HTML στον διακομιστή, αντιστοιχώντας έτσι στην απόδοση από την πλευρά του διακομιστή και στο παραδοσιακό πολυσέλιδο μοντέλο της Ενότητας 2.2.4· στις μονοσέλιδες εφαρμογές ο ρόλος αυτός μετατοπίζεται στο πρόγραμμα περιήγησης, και το framework του διακομιστή περιορίζεται στην έκθεση διεπαφών δεδομένων. Η προσέγγιση αυτή επιταχύνει σημαντικά την αρχική ανάπτυξη και επιβάλλει μια ενιαία δομή στον κώδικα, με αντίτιμο μια πιο απότομη καμπύλη εκμάθησης και μικρότερη ελευθερία στις αρχιτεκτονικές επιλογές.

Η δεύτερη είναι εκείνη των ελαφρών ή μινιμαλιστικών frameworks, όπως το Express.js για το περιβάλλον Node.js, καθώς και το Flask και το FastAPI για την Python. Τα frameworks αυτά παρέχουν κυρίως τον πυρήνα δρομολόγησης των αιτημάτων και έναν μηχανισμό ενδιάμεσου λογισμικού (middleware), αφήνοντας στην ομάδα ανάπτυξης την επιλογή των υπόλοιπων

συνιστωσών. Η ευελιξία αυτή τα καθιστά ιδιαίτερα διαδεδομένα στην ανάπτυξη μικρο-υπηρεσιών και API-driven αρχιτεκτονικών, όπου κάθε υπηρεσία είναι σκόπιμα μικρή, εξειδικευμένη και φέρει μόνο τις εξαρτήσεις που πραγματικά χρειάζεται. Αξίζει να σημειωθεί ότι ορισμένα από αυτά, όπως το FastAPI, ενσωματώνουν εγγενώς την αυτόματη παραγωγή τεκμηρίωσης κατά το πρότυπο OpenAPI, ευθυγραμμιζόμενα έτσι με τη λογική των API-driven αρχιτεκτονικών της Ενότητας 2.6.6· αντίστοιχη υποστήριξη, μέσω καθιερωμένων συνοδευτικών βιβλιοθηκών, παρέχουν και τα ολοκληρωμένα frameworks, όπως το Spring Boot, οπότε η δυνατότητα αυτή δεν αποτελεί αποκλειστικό χαρακτηριστικό των ελαφρών επιλογών.

Πίνακας 2.3: Συγκριτική επισκόπηση κύριων back-end frameworks. Πηγή: επεξεργασία του συγγραφέα βάσει της επίσημης τεκμηρίωσης κάθε framework και του Stack Overflow Developer Survey (2024).

Framework	**Φορέας**	**Βασικό Χαρακτηριστικό**	**Καταλληλότητα**
Django	Django Software Foundation	Python, ολοκληρωμένο, "batteries included"	Δεδομενοκεντρικές εφαρμογές με πλούσιο περιεχόμενο
Spring Boot	VMware / Broadcom	Java, ολοκληρωμένο, έγχυση εξαρτήσεων	Επιχειρησιακά συστήματα μεγάλης κλίμακας
Laravel	Ανοιχτή κοινότητα	PHP, σύμβαση έναντι παραμετροποίησης	Εφαρμογές και ιστότοποι μεσαίας κλίμακας
Express.js	OpenJS Foundation	Node.js, ελαφρύ, βασισμένο σε middleware	APIs και μικρο-υπηρεσίες
FastAPI	Ανοιχτή κοινότητα	Python, ελαφρύ, εγγενής υποστήριξη OpenAPI	Ασύγχρονα APIs υψηλής απόδοσης

Η επιλογή ανάμεσα στις δύο φιλοσοφίες αντανακλά, τελικά, μια ανταλλαγή ανάμεσα στην ταχύτητα της αρχικής υλοποίησης και στον βαθμό ελέγχου επί της αρχιτεκτονικής: τα ολοκληρωμένα frameworks ευνοούν τις εφαρμογές με τυποποιημένες, δεδομενοκεντρικές ανάγκες, ενώ τα ελαφρά ευνοούν τις κατανεμημένες αρχιτεκτονικές. Η διάκριση αυτή δεν είναι απόλυτη: ορισμένα ολοκληρωμένα frameworks, με χαρακτηριστικό παράδειγμα το Spring Boot, συνοδεύονται από υποσυστήματα ειδικά σχεδιασμένα για κατανεμημένες εφαρμογές και χρησιμοποιούνται ευρέως σε αρχιτεκτονικές μικρο-υπηρεσιών. Η επιλογή αυτή δεν είναι ανεξάρτητη από τη μέθοδο ανάπτυξης που υιοθετείται: μια ομάδα που ακολουθεί αρχιτεκτονική μικρο-υπηρεσιών και πρακτικές DevOps τείνει προς τα ελαφρά frameworks, ακριβώς επειδή η ανεξαρτησία της τεχνολογικής στοίβας ανά υπηρεσία αποτελεί ρητό σχεδιαστικό στόχο της προσέγγισης αυτής (Velepucha & Flores, 2023).

2.6.9 Συστήματα Διαχείρισης Περιεχομένου και Ασφάλεια

Τα συστήματα διαχείρισης περιεχομένου (Content Management Systems, CMS), με κυρίαρχο το WordPress, κινούν ένα πολύ μεγάλο μέρος του πραγματικού ιστού, επιτρέποντας την ανάπτυξη εφαρμογών με ελάχιστο ή καθόλου προγραμματισμό. Αυτό καθίσταται δυνατό επειδή ένα CMS

παρέχει έτοιμο, και παραμετροποιήσιμο, το σύνολο σχεδόν της τυπικής στοίβας μιας εφαρμογής ιστού. Στην πλευρά του διακομιστή, προσφέρει ένα προκατασκευασμένο σχήμα βάσης δεδομένων για το περιεχόμενο, σύστημα αυθεντικοποίησης και ρόλων χρηστών, διαχείριση αρχείων πολυμέσων και ένα περιβάλλον διαχείρισης μέσα από το οποίο ο μη τεχνικός χρήστης δημιουργεί και επεξεργάζεται περιεχόμενο. Στην πλευρά της παρουσίασης, η εμφάνιση καθορίζεται από επιλέξιμα θέματα (themes), δηλαδή έτοιμα σύνολα προτύπων που μετατρέπουν το αποθηκευμένο περιεχόμενο σε HTML, ώστε η σχεδίαση να γίνεται με επιλογή και παραμετροποίηση αντί για συγγραφή κώδικα. Η όποια επιπλέον λειτουργικότητα - φόρμες επικοινωνίας, ηλεκτρονικό κατάστημα, βελτιστοποίηση για μηχανές αναζήτησης - προστίθεται μέσω επεκτάσεων (extensions ή plugins), οι οποίες συνδέονται στον πυρήνα της πλατφόρμας μέσα από προκαθορισμένα σημεία επέκτασης. Ο ρόλος του χρήστη μετατοπίζεται έτσι από τη συγγραφή κώδικα στη σύνθεση και παραμετροποίηση έτοιμων συνιστωσών.

Ακριβώς αυτή η αρχιτεκτονική, ωστόσο, μεταθέτει και το κέντρο βάρους του κινδύνου ασφάλειας. Παρά τη βιομηχανική τους κυριαρχία, τα CMS παραμένουν σημαντικά υποεκπροσωπημένα στην ακαδημαϊκή βιβλιογραφία - ένα χάσμα που η παρούσα επισκόπηση αναγνωρίζει ρητά ως ερευνητικό κενό. Ένα κρίσιμο ζήτημα ασφάλειας για τις πλατφόρμες αυτές αναδεικνύεται στη μελέτη των Niemietz, Korth, Mainka και Somorovsky (2021), οι οποίοι ανέλυσαν τα δέκα κορυφαία επίσημα extensions του Joomla και εντόπισαν ευπάθεια Cross-Site Scripting και στα δέκα, καθώς και SQL Injection στα τρία από αυτά. Δύο μάλιστα από τα εξεταζόμενα extensions διατίθενται και σε WordPress ή Drupal, μεταφέροντας εκεί τις ίδιες ευπάθειες. Το εύρημα υπογραμμίζει ότι η ασφάλεια των CMS εξαρτάται κρίσιμα από το οικοσύστημα των extensions και όχι μόνο από τον πυρήνα της πλατφόρμας - μια παρατήρηση που ευθυγραμμίζεται με την κατηγορία A06 του OWASP Top 10 (2021), «Vulnerable and Outdated Components» (ευπαθή και ξεπερασμένα στοιχεία).

2.6.10 Συνοπτική Αντιστοίχιση Μεθόδων και Σταδίων

Ο Πίνακας 2.4 προσφέρει μια πρώτη, εποπτική αντιστοίχιση των κύριων μεθόδων και τεχνικών με τα στάδια του SDLC, με βάση τη βιβλιογραφία που παρουσιάστηκε στις προηγούμενες ενότητες. Έχει αυστηρά ενδεικτικό και ποιοτικό χαρακτήρα: αποσκοπεί στο να δώσει στον αναγνώστη μια συνολική εικόνα πριν από τη συστηματική ανάλυση, και όχι να προκαταλάβει τα αποτελέσματά της. Η πλήρης, ποσοτικά τεκμηριωμένη αποτίμηση, βάσει είκοσι ρητών κριτηρίων αξιολόγησης, παρουσιάζεται στους Πίνακες 4.3 και 4.4.

Δύο διευκρινίσεις είναι απαραίτητες για την ανάγνωση του πίνακα. Πρώτον, η ανάλυση απαιτήσεων και η σχεδίαση παρουσιάζονται εδώ ως χωριστές στήλες, καθώς αρκετές μέθοδοι τις καλύπτουν σε αισθητά διαφορετικό βαθμό - χαρακτηριστικά, οι μεθοδολογίες MDWE επικεντρώνονται στη σχεδίαση δίνοντας ελάχιστη προσοχή στη μηχανική απαιτήσεων, όπως επισήμαναν ήδη οι Escalona και Koch (2004). Στη συγκριτική ανάλυση του Κεφαλαίου 4 τα δύο αυτά στάδια συγχωνεύονται σε ένα, για τους λόγους που εκτέθηκαν στην Ενότητα 2.3.1, όπου η στενή τους αλληλεξάρτηση στην ανάπτυξη εφαρμογών ιστού δικαιολογεί την ενιαία τους αντιμετώπιση. Δεύτερον, η βαθμολόγηση λαμβάνει υπόψη όχι μόνο αν μια μέθοδος καλύπτει εννοιολογικά ένα στάδιο, αλλά και αν προσφέρει εξειδικευμένες τεχνικές για αυτό. Έτσι, μια μέθοδος που περιγράφει τι πρέπει να γίνει σε ένα στάδιο χωρίς να παρέχει αντίστοιχα τεχνικά μέσα βαθμολογείται ως ουσιαστική (ν) και όχι ως πλήρης (νν) κάλυψη - διάκριση που αφορά κατεξοχήν τις ευέλικτες μεθόδους, οι οποίες καλύπτουν περιγραφικά σχεδόν ολόκληρο τον κύκλο ζωής.

Πίνακας 2.4: Ενδεικτική αντιστοίχιση μεθόδων και τεχνικών με στάδια SDLC. Υπόμνημα: νν = πλήρης κάλυψη, με εξειδικευμένες τεχνικές· ν = ουσιαστική κάλυψη· ~ = μερική ή έμμεση κάλυψη· Χ = ουσιαστική απουσία. Πηγές τεκμηρίωσης: Escalona & Koch (2004), Linaje et al. (2019), Rossi et al. (2016), Shahin et al. (2017), Velepucha & Flores (2023), Zhou et al. (2023).

Μέθοδος/ Τεχνική	**Ανάλυση Απαιτήσεων**	**Σχεδίαση**	**Υλοποίηση**	**Δοκιμή**	**Διάταξη**	**Συντήρηση**
MDWE (OOHDM, UWE, WebML)	ν	νν	~	~	~	~
Agile / Scrum	ν	ν	ν	ν	~	ν
DevOps & CI/CD	~	~	νν	νν	νν	ν
Microservices	~	ν	ν	~	νν	νν
TDD & Automated Testing	Χ	ν	ν	νν	~	~
Frameworks & Tools (cloud-native)	~	ν	νν	νν	νν	ν

Ο πίνακας αποτυπώνει ένα κεντρικό μήνυμα που διατρέχει ολόκληρη την εργασία: καμία μέθοδος, από μόνη της, δεν καλύπτει πλήρως και ισόρροπα το σύνολο των σταδίων. Διαγράφεται μια σαφής διαφοροποίηση ανάμεσα στις προσεγγίσεις που αντλούν τη δύναμή τους από τα πρώιμα στάδια, με ισχυρή παρουσία στη σχεδίαση αλλά περιορισμένη και κατά κανόνα έμμεση κάλυψη της διάταξης, και σε εκείνες που επικεντρώνονται στην υλοποίηση και τη λειτουργία, με την ακριβώς αντίστροφη εικόνα. Η ανάγνωση του πίνακα κατά στήλη αποκαλύπτει, επιπλέον, ότι τα στάδια δεν καλύπτονται ομοιόμορφα από το σύνολο των οικογενειών. Η σχεδίαση και η υλοποίηση καλύπτονται τουλάχιστον ουσιαστικά από πέντε από τις έξι οικογένειες και αποτελούν έτσι το κοινό έδαφος σχεδόν όλων των προσεγγίσεων, ενώ η δοκιμή και η διάταξη συγκεντρώνουν τις περισσότερες ενδείξεις πλήρους κάλυψης με εξειδικευμένες τεχνικές, με τρεις οικογένειες η καθεμία. Η διάταξη, ωστόσο, παρουσιάζει μια πολωμένη εικόνα: καλύπτεται πλήρως από τις οικογένειες με λειτουργικό, αρχιτεκτονικό ή τεχνολογικό προσανατολισμό, αλλά μερικά ή έμμεσα από τις υπόλοιπες, χωρίς ενδιάμεσες βαθμίδες. Στο αντίθετο άκρο, η ανάλυση απαιτήσεων αποτελεί το ασθενέστερο σημείο του πίνακα, καθώς καμία οικογένεια δεν της προσφέρει πλήρη κάλυψη και είναι το μόνο στάδιο στο οποίο καταγράφεται ουσιαστική απουσία. Η εικόνα αυτή υποδηλώνει ότι η μεθοδολογική ανάπτυξη συγκεντρώθηκε στα στάδια που επιδέχονται τεχνική τυποποίηση και αυτοματοποίηση, αφήνοντας λιγότερο υποστηριγμένα τα δύο άκρα του κύκλου ζωής: τη διατύπωση των απαιτήσεων και τη μακροπρόθεσμη εξέλιξη του

συστήματος. Η διαφοροποίηση αυτή αποτελεί το θεμελιώδες ερευνητικό κενό που εξετάζεται συστηματικά στα Κεφάλαια 4 και 5.

3. Μεθοδολογία Επισκόπησης

Το παρόν κεφάλαιο περιγράφει τον τρόπο με τον οποίο συγκεντρώθηκε, επιλέχθηκε και αναλύθηκε η βιβλιογραφία που στηρίζει την επισκόπηση. Θα ήταν παραπλανητικό να παρουσιαστεί η διαδικασία αυτή ως μια πλήρης, εξαντλητική συστηματική ανασκόπηση (systematic review) με την αυστηρή έννοια των καθιερωμένων πρωτοκόλλων· κάτι τέτοιο θα απαιτούσε ομάδα αξιολογητών, καταμέτρηση χιλιάδων εγγραφών και μετρήσεις συμφωνίας μεταξύ κριτών που ξεπερνούν το πλαίσιο μιας διπλωματικής εργασίας.

Για τον ακριβή χαρακτηρισμό της διαδικασίας αξιοποιείται η τυπολογία των Grant και Booth (2009), οι οποίοι διακρίνουν δεκατέσσερις τύπους επισκόπησης. Σύμφωνα με αυτή, η παρούσα εργασία συνιστά συστηματοποιημένη επισκόπηση (systematised review): μια προσέγγιση που υιοθετεί τα βασικά στοιχεία της συστηματικής ανασκόπησης - ρητά ερευνητικά ερωτήματα, δηλωμένη στρατηγική αναζήτησης σε πολλαπλές βάσεις, σαφή κριτήρια επιλογής και αποκλεισμού, και δομημένη αξιολόγηση ποιότητας - χωρίς όμως να φτάνει στην πληρότητα και στην αυστηρότητα της τελευταίας, ιδίως ως προς την εξαντλητική κάλυψη της βιβλιογραφίας και την ανεξάρτητη αξιολόγηση από δεύτερο κριτή. Πρόκειται για τον τύπο επισκόπησης που οι Grant και Booth αναγνωρίζουν ως χαρακτηριστικό των μεταπτυχιακών εργασιών, όπου η έρευνα διεξάγεται από έναν ερευνητή. Η επιλογή αυτής της μορφής δεν είναι αδυναμία αλλά συνειδητή μεθοδολογική απόφαση, καθώς ταιριάζει με τον διερευνητικό χαρακτήρα του θέματος και με το εύρος του, που καλύπτει ταυτόχρονα μεθόδους, τεχνικές και εργαλεία.

Η επισκόπηση διεξήχθη σε έξι διαδοχικές φάσεις: τον ορισμό των ερευνητικών ερωτημάτων, τη διαμόρφωση και εκτέλεση της στρατηγικής αναζήτησης, την εφαρμογή των κριτηρίων επιλογής και αποκλεισμού, την αξιολόγηση της ποιότητας των υποψήφιων εργασιών, την εξαγωγή των δεδομένων από το τελικό σύνολο και, τέλος, τη σύνθεση και ανάλυσή τους. Το παρόν κεφάλαιο καλύπτει τις πέντε πρώτες φάσεις, ακολουθώντας τη σειρά με την οποία εκτελέστηκαν: η Ενότητα 3.1 παρουσιάζει τα ερευνητικά ερωτήματα, η 3.2 τη στρατηγική αναζήτησης, η 3.3 τα κριτήρια επιλογής και ποιότητας, οι 3.4 και 3.5 το τελικό σύνολο άρθρων και τα στατιστικά του χαρακτηριστικά, και η 3.6 τη διαδικασία εξαγωγής δεδομένων· η 3.7 προσδιορίζει το υποσύνολο των εργασιών που αναλύονται σε βάθος και η 3.8 συζητά τις απειλές εγκυρότητας. Η έκτη φάση, η σύνθεση και η ανάλυση των δεδομένων, αποτελεί το αντικείμενο του Κεφαλαίου 4.

Η διαφάνεια για τη φύση της διαδικασίας είναι, εξάλλου, προϋπόθεση της επιστημονικής εντιμότητας: κάθε αριθμός που παρατίθεται στη συνέχεια αντιστοιχεί σε πραγματικά άρθρα του τελικού συνόλου, τα οποία παρατίθενται πλήρη - με τίτλο, συγγραφείς, venue και αναγνωριστικό DOI- στο Παράρτημα Α.

3.1 Ερευνητικές Ερωτήσεις

Οι ερευνητικές ερωτήσεις ορίζουν τι αναζητείται, πώς αξιολογούνται τα άρθρα και πώς παρουσιάζονται τα αποτελέσματα. Διατυπώθηκαν οκτώ ερωτήσεις, με βάση τον τίτλο και τους στόχους της εργασίας, τα κενά της βιβλιογραφίας που αναδείχθηκαν στο Κεφάλαιο 1, το θεωρητικό υπόβαθρο του Κεφαλαίου 2 και τις ανάγκες του πεδίου.

Πίνακας 3.1: Οι οκτώ ερευνητικές ερωτήσεις της επισκόπησης. Πηγή: επεξεργασία του συγγραφέα βάσει της ανάλυσης κενών του Κεφαλαίου 1.

ID	**Ερευνητική Ερώτηση**	**Σημείο και Τρόπος Απάντησης**
RQ1	Ποιες αντιπροσωπευτικές μέθοδοι και τεχνικές έχουν προταθεί για την ανάπτυξη εφαρμογών ιστού που καλύπτουν περισσότερα του ενός στάδια SDLC, και πώς μπορούν να κατηγοριοποιηθούν;	Η Ενότητα 4.1 κατατάσσει τα άρθρα του τελικού συνόλου σε έξι οικογένειες μεθόδων· η Ενότητα 4.2 αναλύει εις βάθος δύο αντιπροσωπευτικές εργασίες ανά κατηγορία.
RQ2	Πώς μπορούν να συγκριθούν οι εξεταζόμενες μέθοδοι ανά στάδιο ανάπτυξης και υπό ποια κριτήρια;	Η Ενότητα 4.3 ορίζει τα είκοσι κριτήρια σύγκρισης ανά στάδιο SDLC· η Ενότητα 4.4 παρουσιάζει τα αποτελέσματα της βαθμολόγησης (Πίνακες 4.3 και 4.4)· η Ενότητα 4.5 ερμηνεύει τα αποτελέσματα αυτά.
RQ3	Πώς αντιμετωπίζουν οι εξεταζόμενες μέθοδοι και τεχνικές τα ζητήματα ασφάλειας;	Το κριτήριο C12 (Ενότητα 4.3) αποτυπώνει την κάλυψη της ασφάλειας· η Ενότητα 4.5.3 αναλύει τα σχετικά αποτελέσματα και το Σχήμα 4.2 τα απεικονίζει ανά άρθρο.
RQ4	Ποια είναι η σχέση των μεθόδων αυτών με σύγχρονα εργαλεία και περιβάλλοντα νέφους;	Τα κριτήρια C8 και C13–C15 (Ενότητα 4.3) αποτιμούν τη συμβατότητα με το νέφος, τη διάταξη και τη δοχειοποίηση· η Ενότητα 4.5.4 συζητά τα ευρήματα.
RQ5	Ποια frameworks, γλώσσες και εργαλεία χρησιμοποιούνται συχνότερα;	Η Ενότητα 4.5.5 συγκεντρώνει τα εργαλεία που αναφέρονται στο τελικό σύνολο, οργανωμένα ανά στάδιο SDLC.
RQ6	Ποια ερευνητικά κενά παραμένουν ανοιχτά και ποιες κατευθύνσεις αντιμετώπισής τους προτείνονται;	Το Κεφάλαιο 5 τεκμηριώνει επτά ερευνητικά κενά και προτείνει για καθένα συγκεκριμένες κατευθύνσεις, συνοψισμένες στον Πίνακα 5.1.
RQ7	Ποιες μέθοδοι έχουν	Η Ενότητα 4.2 καταγράφει τον

	αξιολογηθεί εμπειρικά και με ποια αποτελέσματα;	τύπο και τα αποτελέσματα της εμπειρικής αξιολόγησης κάθε αντιπροσωπευτικής εργασίας· η Ενότητα 4.5.6 συνοψίζει την εικόνα για το σύνολο, κατατάσσοντας τις εργασίες ανά τύπο εμπειρικής τεκμηρίωσης.
RQ8	Πώς έχει εξελιχθεί διαχρονικά η ερευνητική εστίαση στις μεθόδους ανάπτυξης εφαρμογών ιστού;	Η Ενότητα 4.5.7 ανασυνθέτει τη χρονική εξέλιξη του πεδίου σε τέσσερις γενιές μεθόδων, με βάση το έτος πρώτης εμφάνισης κάθε μεθόδου και τη χρονολογική κατανομή των άρθρων του συνόλου (Σχήμα 3.2).

3.2 Στρατηγική Αναζήτησης

3.2.1 Πηγές Βιβλιογραφίας

Η αναζήτηση διεξήχθη σε πέντε επιστημονικές πηγές, επιλεγμένες για την κάλυψή τους στη μηχανική λογισμικού και την ανάπτυξη εφαρμογών ιστού: IEEE Xplore, ACM Digital Library, ScienceDirect της Elsevier, SpringerLink και Google Scholar. Η τελευταία λειτούργησε συμπληρωματικά, για τον εντοπισμό preprints από το αποθετήριο arXiv και άρθρων ανοικτής πρόσβασης, όπως εκείνων της MDPI, που ενδέχεται να μην ευρετηριάζονται πλήρως στις υπόλοιπες βάσεις. Επιπλέον, αξιοποιήθηκε η τεχνική του snowballing (Wohlin, 2014): από τις αναφορές των αρχικά εντοπισμένων επισκοπήσεων προέκυψαν επιπλέον σχετικές μελέτες, ιδίως θεμελιώδεις εργασίες MDWE και πρόσφατες συστηματικές ανασκοπήσεις για microservices και DevOps.

Η συμπερίληψη preprints απαιτεί ρητή αιτιολόγηση, καθώς πρόκειται για εργασίες που δεν έχουν υποβληθεί σε κριτική αξιολόγηση από ομοτίμους και, συνεπώς, δεν φέρουν την εγγύηση ποιότητας των δημοσιευμένων άρθρων. Η απόφαση αυτή ελήφθη για έναν συγκεκριμένο λόγο: το πεδίο εξελίσσεται με ρυθμό που υπερβαίνει τον κύκλο της κριτικής αξιολόγησης, και ο αποκλεισμός των preprints θα σήμαινε την παράλειψη των πλέον πρόσφατων επισκοπήσεων σε θέματα αιχμής, όπως η AI-ενισχυμένη δοκιμή εφαρμογών ιστού. Για να περιοριστεί ο κίνδυνος ως προς την ποιότητα, εφαρμόστηκαν τρεις σωρευτικοί περιορισμοί, οι οποίοι αποτυπώνονται ως προϋποθέσεις του κριτηρίου IC3 της Ενότητας 3.3.1. Πρώτον, τα preprints έγιναν δεκτά μόνο κατ' εξαίρεση, όταν κάλυπταν πολύ πρόσφατες εξελίξεις που δεν αποτυπώνονταν σε κάποια αξιολογημένη εργασία του συνόλου. Δεύτερον, υποβλήθηκαν στην ίδια ακριβώς αξιολόγηση ποιότητας με τις υπόλοιπες εργασίες (Ενότητα 3.3.2), χωρίς καμία χαλάρωση του κατωφλίου αποδοχής. Τρίτον, απαιτήθηκε οι συγγραφείς να διαθέτουν προηγούμενες αξιολογημένες δημοσιεύσεις στο ίδιο πεδίο. Το τελικό σύνολο περιλαμβάνει δύο μόνο preprints, ποσοστό 7,4%, έναντι 85,2% άρθρων περιοδικού, γεγονός που περιορίζει αντικειμενικά την επίδρασή τους στα συμπεράσματα· η επιλογή αυτή καταγράφεται και στις απειλές εγκυρότητας της Ενότητας 3.8.

Ο συνδυασμός αναζήτησης σε πολλαπλές βάσεις με τη μέθοδο του snowballing δεν είναι τυχαίος, αλλά στηρίζεται σε τεκμηριωμένη μεθοδολογική πρακτική. Οι Jalali και Wohlin (2012),

συγκρίνοντας συστηματικά τις δύο βασικές στρατηγικές αναζήτησης - την αναζήτηση σε βάσεις δεδομένων και το backward snowballing από τις λίστες αναφορών - κατέληξαν στο ότι καμία από τις δύο δεν υπερτερεί απόλυτα, καθώς και οι δύο οδηγούν σε ουσιαστικά συγκλίνοντα σύνολα άρθρων και παρόμοια συμπεράσματα. Το εύρημα αυτό υποδεικνύει ότι ο συνδυασμός τους -αντί της αποκλειστικής χρήσης μίας μόνο - ενισχύει την πληρότητα και μειώνει τον κίνδυνο συστηματικής παράλειψης σημαντικών εργασιών, ιδίως θεμελιωδών μελετών που ενδέχεται να μην εντοπίζονται με τους τρέχοντες όρους αναζήτησης. Η αξιοποίηση πέντε διακριτών πηγών, καθεμιάς με διαφορετική εκδοτική κάλυψη (η IEEE και η ACM για συνέδρια και περιοδικά μηχανικής, η Elsevier και η Springer για εδραιωμένα επιστημονικά περιοδικά, και η Google Scholar για ανοικτής πρόσβασης και preprint υλικό), υπηρετεί ακριβώς αυτόν τον στόχο της ευρύτερης δυνατής κάλυψης εντός των ορίων μιας συστηματοποιημένης επισκόπησης.

3.2.2 Μοτίβα Επερώτησης

Για κάθε ερευνητική ερώτηση διαμορφώθηκε ένα δομημένο boolean μοτίβο, συνδυάζοντας όρους για το αντικείμενο, τη μέθοδο ή τεχνική, και το ζητούμενο αποτέλεσμα. Τα μοτίβα προσαρμόστηκαν στη σύνταξη κάθε βάσης. Ο Πίνακας 3.2 παραθέτει ενδεικτικά τέσσερα από τα μοτίβα που χρησιμοποιήθηκαν, εκείνα που αντιστοιχούν στις ερευνητικές ερωτήσεις με διακριτό αντικείμενο αναζήτησης. Οι υπόλοιπες τέσσερις ερωτήσεις δεν διαθέτουν δικό τους μοτίβο, για δύο διαφορετικούς λόγους. Η RQ2, που αφορά τα κριτήρια σύγκρισης των μεθόδων, δεν απαντάται μέσα από αναζήτηση αλλά μέσα από την ανάλυση του ήδη επιλεγμένου συνόλου: τα κριτήρια διαμορφώθηκαν από τη βιβλιογραφία, το πρότυπο ISO/IEC 25010 και το ίδιο το περιεχόμενο των άρθρων, όπως περιγράφεται στην Ενότητα 4.3. Αντιστοίχως, οι RQ6, RQ7 και RQ8 - που αφορούν τα ερευνητικά κενά, την εμπειρική αξιολόγηση και τη διαχρονική εξέλιξη των μεθόδων - απαντώνται με βάση το σύνολο των εργασιών που εντοπίστηκαν μέσω των μοτίβων της RQ1, RQ3, RQ4 και RQ5, με φιλτράρισμα βάσει περιεχομένου κατά το στάδιο της επιλογής και της εξαγωγής δεδομένων.

Πίνακας 3.2: Ενδεικτικά μοτίβα επερώτησης ανά ερευνητική ερώτηση.

RQ	**Ενδεικτικό μοτίβο επερώτησης**
RQ1	("web application" OR "web app" OR "web platform") AND ("development method*") AND ("SDLC" OR "life cycle" OR "multi-stage" OR "multi-phase")
RQ3	("web application") AND ("security" OR "OWASP" OR "vulnerability" OR "DevSecOps") AND ("method*" OR "technique*")
RQ4	("web application development") AND ("cloud" OR "cloud-native" OR "microservices" OR "container*" OR "Kubernetes") AND ("method*" OR "framework" OR "tool*")
RQ5	("web development") AND ("framework" OR "programming language" OR "tool*") AND

	("comparison" OR "empirical" OR "survey" OR "review")

3.2.3 Χρονικά και Γλωσσικά Φίλτρα

Κατά το στάδιο της αναζήτησης εφαρμόστηκαν τρία αυστηρά φίλτρα, τα οποία λειτούργησαν ως απόλυτοι περιορισμοί στις ίδιες τις επερωτήσεις, χωρίς εξαιρέσεις. Το πρώτο, χρονικό, όρισε ως κατώτερο όριο το έτος 2007 και ως ανώτερο το 2025. Το δεύτερο, γλωσσικό, περιόρισε την αναζήτηση σε αγγλόφωνες δημοσιεύσεις. Και το τρίτο, τυπολογικό, περιόρισε τα αποτελέσματα σε άρθρα επιστημονικών περιοδικών και συνεδρίων, καθώς και σε preprints επιστημονικών αποθετηρίων, αποκλείοντας βιβλία, διατριβές, τεχνικές αναφορές και υλικό μη επιστημονικού χαρακτήρα.

Η περαιτέρω διαφοροποίηση εντός των ορίων αυτών δεν αποτέλεσε φίλτρο αναζήτησης αλλά κριτήριο επιλογής, εφαρμοζόμενο στο επόμενο στάδιο της διαδικασίας (Ενότητα 3.3). Συγκεκριμένα, μεταξύ των εργασιών που πληρούσαν τα παραπάνω φίλτρα, δόθηκε προτεραιότητα σε δύο κατευθύνσεις: χρονικά, στις δημοσιεύσεις της περιόδου 2015-2025, ώστε να καλυφθούν επαρκώς οι σύγχρονες τεχνολογίες όπως το υπολογιστικό νέφος, τα microservices και το DevOps, με τις παλαιότερες εργασίες να επιλέγονται μόνο εφόσον αφορούσαν θεμελιώδεις μεθόδους ή ευρέως αναφερόμενες μετα-αναλύσεις· και τυπολογικά, στα άρθρα που έχουν υποβληθεί σε κριτική αξιολόγηση από ομοτίμους, με τα preprints να γίνονται δεκτά μόνο κατ' εξαίρεση, υπό τις προϋποθέσεις που αναλύθηκαν στην Ενότητα 3.2.1.

3.3 Κριτήρια Επιλογής και Ποιότητας

3.3.1 Κριτήρια Επιλογής

Ένα άρθρο συμπεριλαμβάνεται στο τελικό σύνολο εφόσον πληροί όλα τα ακόλουθα κριτήρια (inclusion criteria, IC):

- Συνάφεια αντικειμένου (IC1). Το άρθρο προτείνει, αξιολογεί ή συγκρίνει μεθόδους, τεχνικές, frameworks ή εργαλεία για την ανάπτυξη εφαρμογών ιστού.
- Κάλυψη σταδίων (IC2). Η εξεταζόμενη προσέγγιση καλύπτει τουλάχιστον δύο από τα πέντε στάδια του κύκλου ζωής, σύμφωνα με τον δηλωμένο στόχο της επισκόπησης. Κατ' εξαίρεση γίνονται δεκτές εργασίες που εστιάζουν σε ένα μόνο στάδιο, εφόσον αποτελούν τις πλέον εκτενείς και αντιπροσωπευτικές επισκοπήσεις της θεματικής τους κατηγορίας και η παράλειψή τους θα άφηνε την κατηγορία αυτή υποεκπροσωπημένη.
- Εγκυρότητα πηγής (IC3). Η εργασία έχει δημοσιευθεί, μετά από κριτική αξιολόγηση, σε αναγνωρισμένο επιστημονικό venue (IEEE, ACM, Springer, Elsevier, MDPI). Κατ' εξαίρεση γίνεται δεκτό preprint επιστημονικού αποθετηρίου, εφόσον πληρούνται σωρευτικά τρεις προϋποθέσεις: πρώτον, να μην υπάρχει στο σύνολο αξιολογημένη εργασία που να καλύπτει το ίδιο αντικείμενο με αντίστοιχη πληρότητα και επικαιρότητα· δεύτερον, οι συγγραφείς να διαθέτουν προηγούμενες αξιολογημένες δημοσιεύσεις στο ίδιο πεδίο· και τρίτον, η εργασία να υπερβαίνει το κατώφλι της αξιολόγησης ποιότητας της Ενότητας 3.3.2, χωρίς καμία χαλάρωση των κριτηρίων.
- Χρονική συνάφεια (IC4). Η εργασία έχει δημοσιευθεί εντός της περιόδου 2015-2025. Κατ' εξαίρεση γίνονται δεκτές παλαιότερες εργασίες, έως το 2007, εφόσον πρόκειται για

θεμελιώδεις μεθόδους του πεδίου ή για ευρέως αναφερόμενες μετα-αναλύσεις και συστηματικές ανασκοπήσεις που εξακολουθούν να αποτελούν σημείο αναφοράς.

- Γλώσσα (IC5). Το άρθρο είναι γραμμένο στην αγγλική γλώσσα.
- Διαθεσιμότητα (IC6). Το πλήρες κείμενο είναι προσβάσιμο.
- Πρωτοτυπία συνεισφοράς (IC7). Έλεγχος αν από τον τίτλο και την περίληψη προκύπτει πρωτότυπη συνεισφορά, με τη μορφή νέας μεθόδου, εμπειρικής αξιολόγησης ή συστηματικής σύγκρισης.
- Μη επικάλυψη (IC8). Το άρθρο δεν αποτελεί προγενέστερη εκδοχή εργασίας που έχει ήδη ενταχθεί στο σύνολο σε πληρέστερη μορφή.

Τα κριτήρια αυτά εφαρμόστηκαν με βάση τον τίτλο, την περίληψη και τα βιβλιογραφικά στοιχεία κάθε εργασίας· η αποτίμηση που απαιτεί ανάγνωση του πλήρους κειμένου ανήκει στο επόμενο στάδιο, εκείνο της αξιολόγησης ποιότητας. Δεν διατυπώνεται χωριστό σύνολο κριτηρίων αποκλεισμού. Καθώς η συμπερίληψη απαιτεί την ταυτόχρονη ικανοποίηση όλων των παραπάνω κριτηρίων, η μη ικανοποίηση οποιουδήποτε από αυτά συνιστά από μόνη της λόγο αποκλεισμού· ένα χωριστό σύνολο θα επαναλάμβανε τα ίδια κριτήρια με αντεστραμμένη διατύπωση.

3.3.2 Αξιολόγηση Ποιότητας

Κάθε άρθρο που πληρούσε τα κριτήρια επιλογής αξιολογήθηκε, με βάση το πλήρες κείμενό του, ως προς πέντε κριτήρια ποιότητας, το καθένα βαθμολογούμενο από μηδέν έως δύο, με μέγιστο συνολικό σκορ τις δέκα μονάδες. Τα κριτήρια αφορούν τη σαφήνεια του ερευνητικού στόχου, την επάρκεια της περιγραφής της μεθόδου, την ύπαρξη εμπειρικής αξιολόγησης, το εύρος της κάλυψης των σταδίων του κύκλου ζωής, και την καταλληλότητα του άρθρου ως προς τις οκτώ ερευνητικές ερωτήσεις της παρούσας επισκόπησης, όπως αυτές ορίστηκαν στην Ενότητα 3.1. Το τελευταίο κριτήριο δεν τιμωρεί την παλαιότητα καθαυτή: θεμελιώδεις ιστορικές συνεισφορές, όπως οι πρώιμες μέθοδοι MDWE, αξιολογούνται ως προς τη διαρκή τους σημασία, ώστε να αποφευχθεί μια συστηματική μεροληψία εις βάρος τους.

Η τριβάθμια κλίμακα βαθμολόγησης εφαρμόστηκε ομοιόμορφα σε καθένα από τα πέντε κριτήρια. Μια βαθμολογία μηδέν υποδηλώνει ότι το κριτήριο δεν πληρείται - για παράδειγμα, ένα άρθρο χωρίς καμία εμπειρική αποτίμηση της προτεινόμενης μεθόδου. Μια βαθμολογία ένα αντιστοιχεί σε μερική ικανοποίηση, όπως μια περιγραφή μεθόδου που παρέχει τη γενική ιδέα αλλά όχι επαρκείς λεπτομέρειες για αναπαραγωγή, ή μια αξιολόγηση περιορισμένης κλίμακας. Και μια βαθμολογία δύο σηματοδοτεί πλήρη ικανοποίηση του κριτηρίου, όπως έναν σαφώς διατυπωμένο ερευνητικό στόχο ή μια αυστηρή εμπειρική μελέτη με επαρκές δείγμα.

Για την αποδοχή ενός άρθρου τέθηκαν δύο σωρευτικές προϋποθέσεις, σύμφωνα με την καθιερωμένη πρακτική των ανασκοπήσεων. Η πρώτη είναι ποσοτική: το άθροισμα των πέντε επιμέρους βαθμών να ανέρχεται τουλάχιστον σε έξι από τις δέκα μονάδες. Η δεύτερη είναι περιοριστική ανά κριτήριο: κανένα από τα πέντε κριτήρια να μη βαθμολογείται με μηδέν, ώστε να αποκλείονται άρθρα που, παρά το ικανοποιητικό συνολικό σκορ, παρουσιάζουν πλήρη έλλειψη σε κάποια θεμελιώδη διάσταση ποιότητας.

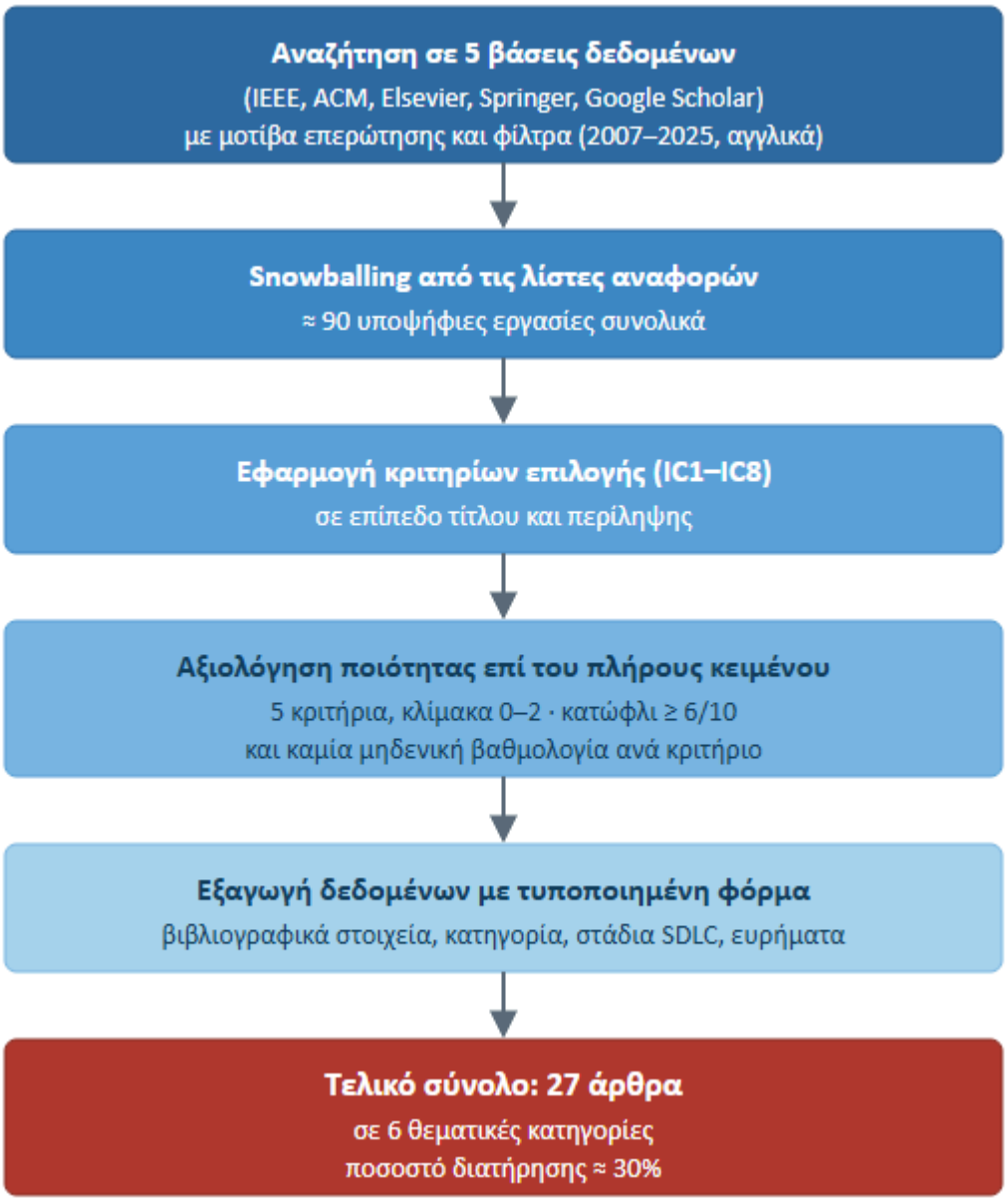
Η επιλογή του κατωφλίου στις έξι μονάδες, αντί του αυστηρότερου ορίου των επτά που συναντάται συχνά στη βιβλιογραφία, υπαγορεύεται από τη σύνθεση του συγκεκριμένου συνόλου. Η πλειονότητα των εργασιών αποτελούν δευτερογενείς μελέτες-συστηματικές ανασκοπήσεις, μελέτες χαρτογράφησης και μετα-αναλύσεις - οι οποίες εξ ορισμού βαθμολογούνται χαμηλότερα στο κριτήριο της πρωτογενούς εμπειρικής αξιολόγησης, χωρίς αυτό να αντανακλά έλλειψη μεθοδολογικής αυστηρότητας. Ένα κατώφλι επτά μονάδων θα απέκλειε συστηματικά εργασίες υψηλής ποιότητας για έναν λόγο που δεν αφορά την ποιότητά τους, εισάγοντας μεροληψία υπέρ

των πρωτογενών μελετών. Η επιλογή αυτή δεν οδήγησε, στην πράξη, σε σύνολο χαμηλής ποιότητας: ο μέσος όρος του συνολικού σκορ των είκοσι επτά άρθρων ανέρχεται σε 7,9 στις δέκα μονάδες, με το χαμηλότερο σκορ του συνόλου να είναι έξι και το υψηλότερο εννέα. Με τον συνδυασμό των δύο προϋποθέσεων, το τελικό σύνολο χαρακτηρίζεται όχι μόνο από θεματική συνάφεια αλλά και από τεκμηριωμένη μεθοδολογική ποιότητα.

3.4 Το Τελικό Σύνολο Άρθρων

Η συνολική ροή της διαδικασίας επιλογής αποτυπώνεται στο Σχήμα 3.1. Η αναζήτηση στις πέντε βάσεις, με την εφαρμογή των μοτίβων επερώτησης και των φίλτρων της Ενότητας 3.2, συμπληρώθηκε από την τεχνική του snowballing και απέδωσε ένα σύνολο της τάξης των ενενήντα υποψήφιων εργασιών. Η εξέταση σε επίπεδο τίτλου και περίληψης, με βάση τα κριτήρια επιλογής της Ενότητας 3.3.1, και στη συνέχεια η αξιολόγηση ποιότητας επί του πλήρους κειμένου, κατέληξαν στο τελικό σύνολο των 27 άρθρων - ποσοστό διατήρησης της τάξης του 30% επί των υποψηφίων. Τα άρθρα αυτά κατανέμονται στις έξι οικογένειες μεθόδων που δομούν την ανάλυση του Κεφαλαίου 4. Σημειώνεται ότι οι αριθμοί των ενδιάμεσων σταδίων δεν καταγράφηκαν συστηματικά κατά την εκτέλεση της διαδικασίας και αναφέρονται, για τον λόγο αυτό, ως τάξεις μεγέθους· ο περιορισμός αυτός συζητείται στην Ενότητα 3.8.

Ο αριθμός αυτός αντικατοπτρίζει τη φύση της επισκόπησης: αντί για μια εξαντλητική καταμέτρηση, επιδιώχθηκε η επιλογή, ανά οικεγένεια μεθόδων, των πλέον αντιπροσωπευτικών και υψηλής ποιότητας μελετών, κατά κανόνα συστηματικών ανασκοπήσεων, μετα-αναλύσεων και εκτενών επισκοπήσεων που η καθεμία συνθέτει δεκάδες ή εκατοντάδες πρωτογενείς μελέτες.



Σχήμα 3.1: Ροόγραμμα της διαδικασίας επιλογής άρθρων.

Σημείωση: Ενδεικτικά, μόνο οι συστηματικές ανασκοπήσεις του τελικού συνόλου συνθέτουν αθροιστικά πάνω από χίλιες πρωτογενείς μελέτες - για παράδειγμα, οι Doğan et al. (95 μελέτες), οι Li et al. (314), οι Kertusha et al. (214), οι Dybå & Dingsøyr (36 εμπειρικές), οι Wakil & Jawawi (289), οι Alazmi & De Leon (90) και οι Pastrana et al. (111). Έτσι, το σύνολο των 27 άρθρων ενσωματώνει έμμεσα τη γνώση ενός πολύ ευρύτερου σώματος βιβλιογραφίας.

Πίνακας 3.3: Πίνακας 3.3: Κατανομή του τελικού συνόλου ανά οικογένεια μεθόδων. Πηγή: επεξεργασία του συγγραφέα (βλ. Παράρτημα Α).

Οικογένεια Μεθόδων	**Προσανατολισμός**	**Άρθρα**	**Ποσοστό**
Model-Driven Web	Μοντελοκεντρικός	3	11,1%

Engineering	(model-oriented)		
Agile & Scrum για Web	Διαδικαστικός (process-oriented)	6	22,2%
DevOps & CI/CD	Λειτουργικός (development-operations-oriented)	4	14,8%
Microservices Architecture	Αρχιτεκτονικός (architecture-oriented)	5	18,5%
TDD & Automated Testing	Πρακτικοκεντρικός (practice-oriented)	7	25,9%
Frameworks, Γλώσσες & Ασφάλεια	Τεχνολογικός (technology/framework-oriented)	2	7,4%
Σύνολο		*27*	*100%*

Η ταξινόμηση αυτή προέκυψε επαγωγικά, από το ίδιο το περιεχόμενο των άρθρων του συνόλου, και όχι από κάποια προϋπάρχουσα ταξινομία. Δύο επιλογές χρήζουν αιτιολόγηση. Πρώτον, η κατηγορία των ευέλικτων μεθόδων θα μπορούσε να αντιμετωπιστεί ως μέρος μιας γενικότερης διαδικαστικής κατηγορίας, η οποία θα περιλάμβανε και τα ακολουθιακά, τα επαναληπτικά και τα συνεχή μοντέλα - εντάσσοντας έτσι και το DevOps ως υποκατηγορία. Η παρούσα εργασία διατηρεί τις δύο κατηγορίες χωριστές, επειδή η βιβλιογραφία του πεδίου τις πραγματεύεται ως διακριτά σώματα γνώσης, με διαφορετικές ερευνητικές ομάδες και σαφώς διαφορετική εστίαση στα στάδια του κύκλου ζωής· η συγχώνευσή τους θα επισκίαζε ακριβώς τη διαφοροποίηση που αναδεικνύει η ανάλυση του Κεφαλαίου 4. Δεύτερον, δεν ορίζεται χωριστή κατηγορία για τις AI-ενισχυμένες προσεγγίσεις, παρότι αυτές αποτελούν σαφή αναδυόμενη τάση. Ο λόγος είναι ότι καμία εργασία του συνόλου δεν έχει ως κύριο αντικείμενο μια AI-βασισμένη μέθοδο ανάπτυξης· η τεχνητή νοημοσύνη εμφανίζεται ως επιμέρους διάσταση εντός άλλων κατηγοριών, κυρίως στις επισκοπήσεις δοκιμών. Η απουσία αυτή δεν αγνοείται, αλλά καταγράφεται ρητά ως ερευνητικό κενό στην Ενότητα 5.5.

3.5 Στατιστικά Στοιχεία του Τελικού Συνόλου

Τα ακόλουθα στοιχεία υπολογίστηκαν απευθείας από τα 27 άρθρα του Παραρτήματος Α και αποτελούν ακριβείς καταμετρήσεις επί του τελικού συνόλου. Η διευκρίνιση αυτή είναι αναγκαία, καθώς το ποσοστό διατήρησης της Ενότητας 3.5.1 αποτελεί τη μοναδική εξαίρεση: αφορά τα ενδιάμεσα στάδια της επιλογής, οι αριθμοί των οποίων δεν καταγράφηκαν συστηματικά και δίνονται ως τάξεις μεγέθους.

3.5.1 Ποσοστό Διατήρησης

Η στοχευμένη φύση της αναζήτησης σημαίνει ότι το ποσοστό διατήρησης δεν έχει την ίδια έννοια με εκείνη μιας εξαντλητικής συστηματικής ανασκόπησης, όπου καταμετρώνται χιλιάδες αρχικές εγγραφές. Η διήθηση των υποψήφιων εργασιών διεξήχθη σε δύο διαδοχικά βήματα. Στο πρώτο,

εφαρμόστηκαν τα κριτήρια επιλογής της Ενότητας 3.3.1 σε επίπεδο τίτλου, περίληψης και βιβλιογραφικών στοιχείων, ώστε να αποκλειστούν οι εργασίες που δεν πληρούσαν τις τυπικές προϋποθέσεις συνάφειας, κάλυψης σταδίων, εγκυρότητας πηγής και χρονικού εύρους. Στο δεύτερο, οι εργασίες που επέζησαν του πρώτου βήματος μελετήθηκαν στο πλήρες κείμενό τους και αξιολογήθηκαν με τα πέντε κριτήρια ποιότητας της Ενότητας 3.3.2· όσες δεν υπερέβαιναν το κατώφλι των έξι μονάδων ή παρουσίαζαν μηδενική βαθμολογία σε κάποιο κριτήριο αποκλείστηκαν, ακόμη και όταν πληρούσαν τα τυπικά κριτήρια επιλογής.

Από ένα σύνολο της τάξης των ενενήντα υποψήφιων εργασιών, όπως αυτό διαμορφώθηκε μετά την αναζήτηση στις πέντε βάσεις και το snowballing, η διαδικασία των δύο βημάτων κατέληξε στα 27 τελικά άρθρα, ποσοστό διατήρησης της τάξης του 30%. Οι αριθμοί αυτοί δίνονται ρητά ως τάξεις μεγέθους και όχι ως ακριβείς καταμετρήσεις, καθώς τα ενδιάμεσα μεγέθη δεν καταγράφηκαν συστηματικά κατά την εκτέλεση - περιορισμός που αναγνωρίζεται στην Ενότητα 3.8. Η επιλεκτικότητα αυτή είναι, πάντως, εσκεμμένη: στόχος δεν ήταν η πληρότητα αλλά η αντιπροσωπευτικότητα και η ποιότητα του τελικού συνόλου.

3.5.2 Αναλογία ανά Τύπο Άρθρου

Πίνακας 3.4: Αναλογία άρθρων ανά τύπο δημοσίευσης.

Τύπος Δημοσίευσης	**Άρθρα**	**Ποσοστό**
Άρθρο περιοδικού (journal)	23	85.2%
Άρθρο συνεδρίου (conference)	2	7,4%
Preprint (arXiv)	2	7,4%
Σύνολο	*27*	*100%*

Η κυριαρχία των άρθρων περιοδικού, που αντιστοιχούν στο 85,2% του συνόλου, ενισχύει την αξιοπιστία του, καθώς οι εργασίες αυτές υπόκεινται σε αυστηρότερη κριτική αξιολόγηση. Οι υπόλοιπες τέσσερις εργασίες δεν επιλέχθηκαν τυχαία, αλλά καλύπτουν δύο συγκεκριμένα κενά του συνόλου. Τα δύο άρθρα συνεδρίου εισφέρουν εμπειρικό υλικό που δεν ήταν διαθέσιμο σε μορφή ανασκόπησης περιοδικού: μια μελέτη υιοθέτησης των ευέλικτων μεθόδων σε βιομηχανικό περιβάλλον (Begel & Nagappan, 2007) και μια μελέτη εξόρυξης αποθετηρίων για τη σύγκριση της JavaScript με την TypeScript (Bogner & Merkel, 2022). Τα δύο preprints, αντιστοίχως, καλύπτουν τις πλέον πρόσφατες εξελίξεις στη δοκιμή εφαρμογών ιστού και, ειδικότερα, τον ρόλο της τεχνητής νοημοσύνης σε αυτήν (Li et al., 2024· Kertusha et al., 2025), θεματική για την οποία καμία αξιολογημένη εργασία του συνόλου δεν προσφέρει αντίστοιχη επικαιρότητα - γεγονός που αποτελεί και τη ρητή προϋπόθεση αποδοχής τους κατά το κριτήριο IC3 της Ενότητας 3.3.1.

3.5.3 Αναλογία ανά Εκδότη

Πίνακας 3.5: Αναλογία άρθρων ανά εκδότη. Πηγή: Παράρτημα Α.

Εκδότης / Venue	**Άρθρα**	**Ποσοστό**
Elsevier (Inf. & Softw. Tech., JSS)	9	33,3%
IEEE (Access, TSE, Proc., ESEM)	7	25,9%
MDPI (Applied Sciences, Information)	3	11,1%
Λοιπά περιοδικά ανοικτής πρόσβασης (CLEI e-Journal, e-Informatica SEJ, iJIM)	3	11,1%
arXiv (preprints)	2	7,4%
Springer (Empirical Software Eng.)	1	3,7%
Wiley (Softw. Practice & Experience)	1	3,7%
ACM (MSR)	1	3,7%
Σύνολο	*27*	*100%*

Η κατανομή ανά εκδότη δείχνει συγκέντρωση στους δύο μεγαλύτερους διεθνείς εκδοτικούς οίκους της μηχανικής λογισμικού, την Elsevier και την IEEE, που μαζί καλύπτουν περίπου τα τρία πέμπτα του συνόλου. Η εικόνα αυτή είναι αναμενόμενη, καθώς τα περιοδικά των δύο αυτών οίκων - ιδίως τα Information and Software Technology, Journal of Systems and Software και IEEE Transactions on Software Engineering - αποτελούν τα καθιερωμένα βήματα δημοσίευσης συστηματικών ανασκοπήσεων στο πεδίο. Το υπόλοιπο σύνολο κατανέμεται σε περισσότερους και μικρότερους φορείς: περίπου το ένα πέμπτο σε περιοδικά ανοικτής πρόσβασης, τόσο καθιερωμένων εκδοτών όπως η MDPI όσο και ανεξάρτητων επιστημονικών περιοδικών, και τα εναπομείναντα άρθρα σε Springer, Wiley και ACM, καθώς και στο αποθετήριο preprints arXiv. Η παρουσία των περιοδικών ανοικτής πρόσβασης αντανακλά τη στροφή μέρους της πρόσφατης βιβλιογραφίας προς το μοντέλο αυτό δημοσίευσης και, παράλληλα, περιορίζει τη μονομέρεια του συνόλου.

Η περιορισμένη εκπροσώπηση της ACM, με ένα μόνο άρθρο, δεν αντανακλά απουσία σχετικής έρευνας αλλά τη δομή του εκδοτικού της καταλόγου σε συνδυασμό με τα κριτήρια επιλογής: η ACM δημοσιεύει στη μηχανική λογισμικού κυρίως πρακτικά συνεδρίων, ενώ το τυπολογικό κριτήριο της Ενότητας 3.2.3 έδωσε προτεραιότητα στα άρθρα περιοδικού, στα οποία

δημοσιεύεται και το μεγαλύτερο μέρος των συστηματικών ανασκοπήσεων του πεδίου. Χαρακτηριστικά, η επισκόπηση των Leite et al. (2019) στο ACM Computing Surveys αξιοποιείται ως αναφορά υποβάθρου στο Κεφάλαιο 2 χωρίς να εντάσσεται στο αναλυόμενο σύνολο, καθώς δεν εστιάζει ειδικά στις εφαρμογές ιστού.

Η συγκέντρωση στους δύο κύριους εκδότες συνιστά, ωστόσο, και έναν περιορισμό: ενδέχεται να ευνοεί ορισμένες ερευνητικές παραδόσεις και κοινότητες έναντι άλλων, ζήτημα που συζητείται στις απειλές εγκυρότητας της Ενότητας 3.8.

3.5.4 Χρονική Κατανομή και Κάλυψη Σταδίων

Ως προς τη χρονική κατανομή, το σύνολο εκτείνεται από το 2007 έως το 2025, με το 33,3% των άρθρων να ανήκει στην πιο πρόσφατη περίοδο 2023–2025 και συνολικά πάνω από το ήμισυ (59,2%) από το 2020 και έπειτα, γεγονός που αντανακλά την έμφαση σε σύγχρονες προσεγγίσεις. Ως προς την κάλυψη σταδίων, μόλις πέντε από τα 27 άρθρα (18,5%) καλύπτουν τέσσερα ή περισσότερα στάδια του κύκλου ζωής· η μεγαλύτερη επιμέρους ομάδα (33,3%) καλύπτει δύο στάδια. Παρότι το ποσοστό αυτό υπολογίζεται σε επιλεγμένο, όχι τυχαίο σύνολο, επιβεβαιώνει ποιοτικά την κεντρική υπόθεση του Κεφαλαίου 1: η ολιστική κάλυψη του κύκλου ζωής παραμένει σπάνια, ακόμη και στις πλέον αντιπροσωπευτικές μελέτες του πεδίου.

Η συνανάγνωση των τριών κατανομών - ανά τύπο, ανά εκδότη και ανά έτος - προσφέρει μια συνεκτική εικόνα της ποιότητας και του προσανατολισμού του συνόλου. Η κυριαρχία των άρθρων περιοδικού (85,2%), σε συνδυασμό με τη συγκέντρωση σε εδραιωμένους εκδότες μηχανικής λογισμικού όπως η Elsevier (33,3%) και η IEEE (25,9%), υποδηλώνει ένα σύνολο με υψηλή μέση αξιοπιστία, καθώς οι πηγές αυτές υπόκεινται σε αυστηρή κριτική αξιολόγηση. Παράλληλα, η έντονη μετατόπιση προς την πρόσφατη περίοδο - με το 59,2% των άρθρων να έχει δημοσιευθεί από το 2020 και έπειτα (Σχήμα 3.2) - αντανακλά τη συνειδητή επιλογή να δοθεί προτεραιότητα στις σύγχρονες τεχνολογικές τάσεις (νέφος, microservices, DevOps), χωρίς ωστόσο να παραληφθούν οι θεμελιώδεις ιστορικές συνεισφορές που καλύπτουν το φάσμα από το 2007. Η ταυτόχρονη παρουσία και των δύο άκρων του χρονικού φάσματος επιτρέπει στην ανάλυση του Κεφαλαίου 4 να αποτυπώσει τόσο την εξέλιξη όσο και τη σύγχρονη αιχμή του πεδίου.

Πίνακας 3.6: Κατανομή των άρθρων του τελικού συνόλου ως προς τον αριθμό σταδίων SDLC που καλύπτουν.

Αριθμός σταδίων	**Άρθρα**	**Ποσοστό**
1 στάδιο	5	18,5%
2 στάδια	9	33,3%
3 στάδια	8	29,6%
4 στάδια	3	11,1%
5 στάδια	2	7,4%
Σύνολο	*27*	*100%*

Σχήμα 3.2: Χρονική κατανομή των 27 άρθρων του τελικού συνόλου.

3.6 Εξαγωγή Δεδομένων

Για κάθε άρθρο εξήχθησαν συστηματικά, μέσω μιας τυποποιημένης φόρμας, τα ακόλουθα μετα-δεδομένα: τα βιβλιογραφικά στοιχεία, η θεματική κατηγορία της μεθόδου ή τεχνικής - με βάση τις έξι κατηγορίες που ορίστηκαν στην Ενότητα 3.4 και παρουσιάζονται στον Πίνακα 3.3 - τα στάδια του κύκλου ζωής που καλύπτει, η κύρια συνεισφορά, τα πλεονεκτήματα και οι αδυναμίες, ο τύπος και τα αποτελέσματα της εμπειρικής αξιολόγησης, τα εργαλεία και frameworks που αναφέρονται, και ο τρόπος αντιμετώπισης της ασφάλειας. Η φόρμα δοκιμάστηκε πιλοτικά σε τρία άρθρα πριν την πλήρη εφαρμογή της, ώστε να επιβεβαιωθεί ότι τα πεδία της ήταν επαρκή και σαφώς ορισμένα.

3.7 Αντιπροσωπευτικά Άρθρα προς Ανάλυση Βάθους

Και τα 27 άρθρα του τελικού συνόλου πληρούν τα κριτήρια επιλογής και ποιότητας και αξιοποιούνται στην ανάλυση. Για την αναλυτική συγκριτική σύνθεση, ωστόσο, επιλέχθηκαν δύο αντιπροσωπευτικές εργασίες από καθεμία από τις έξι θεματικές κατηγορίες, σχηματίζοντας ένα εστιακό υποσύνολο δώδεκα άρθρων. Ο περιορισμός αυτός υπαγορεύεται από το βάθος της ανάλυσης: η βαθμολόγηση κάθε εργασίας ως προς είκοσι κριτήρια απαιτεί πλήρη και λεπτομερή μελέτη του κειμένου της, διαδικασία που δεν θα ήταν εφικτή με την ίδια αυστηρότητα για το σύνολο των 27 άρθρων, δεδομένων των περιορισμών χρόνου και διαθέσιμων πόρων μιας ατομικής διπλωματικής εργασίας. Η επιλογή δύο άρθρων ανά κατηγορία εξασφαλίζει, παράλληλα, ισόρροπη εκπροσώπηση των κατηγοριών στη σύγκριση, ανεξάρτητα από το πλήθος των άρθρων που αντιστοιχεί σε καθεμία.

Η επιλογή των δώδεκα έγινε με τρία διαδοχικά, ρητά κριτήρια, εφαρμοζόμενα εντός κάθε κατηγορίας. Πρώτο κριτήριο ήταν το σκορ ποιότητας της Ενότητας 3.3.2: προκρίθηκαν οι εργασίες με την υψηλότερη βαθμολογία. Δεύτερο κριτήριο, σε περίπτωση ισοβαθμίας, ήταν το εύρος κάλυψης των σταδίων του κύκλου ζωής, όπως καταγράφεται στη στήλη «Στ.» του Παραρτήματος Α: προκρίθηκαν οι εργασίες που καλύπτουν περισσότερα στάδια, καθώς προσφέρουν πληρέστερη βάση για τη σύγκριση ανά στάδιο. Τρίτο και τελευταίο κριτήριο, όπου τα δύο πρώτα δεν επαρκούσαν για να ξεχωρίσουν δύο εργασίες, ήταν η συμπληρωματικότητα:

μεταξύ ισοδύναμων εργασιών προτιμήθηκε εκείνη που πρόσθετε διαφορετική οπτική στο ήδη επιλεγμένο άρθρο της κατηγορίας - για παράδειγμα, μια εμπειρική μελέτη δίπλα σε μια συστηματική ανασκόπηση - ώστε το ζεύγος να καλύπτει ευρύτερο φάσμα της κατηγορίας.

Τα υπόλοιπα δεκαπέντε άρθρα διατηρήθηκαν ως σύνολο επικύρωσης. Ο ρόλος τους δεν είναι διακοσμητικός: αξιοποιούνται για να ελεγχθεί αν τα ευρήματα και οι στατιστικές κατανομές που προκύπτουν από το εστιακό υποσύνολο συμφωνούν με την ευρύτερη βάση τεκμηρίων. Η επικύρωση αυτή διενεργήθηκε με δύο τρόπους. Αφενός, οι περιγραφικές κατανομές της Ενότητας 3.5 - ανά τύπο δημοσίευσης, εκδότη, έτος και κάλυψη σταδίων - υπολογίστηκαν επί του πλήρους συνόλου των 27 άρθρων και όχι επί των δώδεκα, ώστε τα ποσοτικά χαρακτηριστικά του πεδίου να μην εξαρτώνται από την εστιακή επιλογή. Αφετέρου, τα δεκαπέντε άρθρα του συνόλου επικύρωσης αξιοποιήθηκαν για τον έλεγχο επιλεγμένων, ελέγξιμων ευρημάτων της Ενότητας 4.5 - όπως η παρουσία ή απουσία συγκεκριμένων πρακτικών, εργαλείων και σταδίων κάλυψης - και όχι για πλήρη βαθμολόγηση ως προς τα είκοσι κριτήρια. Ο έλεγχος αυτός στηρίχθηκε στα μετα-δεδομένα που είχαν εξαχθεί κατά την Ενότητα 3.6 και, όπου το εύρημα δεν ήταν αποτιμήσιμο από αυτά, σε στοχευμένη ανάγνωση των σχετικών ενοτήτων του πλήρους κειμένου. Πρόκειται, συνεπώς, για μερική και όχι για ισοδύναμη αποτίμηση: τα δεκαπέντε άρθρα λειτουργούν ως έλεγχος συνέπειας των ευρημάτων και όχι ως δεύτερο, ανεξάρτητο σύνολο βαθμολόγησης. Η ασυμμετρία αυτή αναγνωρίζεται ως περιορισμός της παρούσας επισκόπησης: ενδέχεται μια πλήρης αποτίμηση και των δεκαπέντε, να μετέβαλε οριακά τις μέσες βαθμολογίες, χωρίς πάντως να αναιρεί τη σχετική κατάταξη των οικογενειών, η οποία ελέγχθηκε ως προς τη συνέπειά της με το ευρύτερο σύνολο. Τυχόν αποκλίσεις σημειώνονται ρητά στην ανάλυση. Ο πλήρης κατάλογος και των 27 άρθρων, με επισήμανση των δώδεκα εστιακών, παρατίθεται στο Παράρτημα Α.

3.8 Απειλές Εγκυρότητας

Κάθε μεθοδολογία έχει περιορισμούς, και η ειλικρινής αναγνώρισή τους είναι εξίσου σημαντική με την περιγραφή των δυνατών σημείων.

Ως προς τη μεροληψία επιλογής, η επισκόπηση δεν είναι εξαντλητική και το σύνολο επιλέχθηκε στοχευμένα, οπότε ενδέχεται σχετικές μελέτες να μην συμπεριλήφθηκαν· ο κίνδυνος αυτός μετριάζεται με τη χρήση πέντε πηγών, την τεχνική του snowballing και την προτεραιότητα σε συστηματικές ανασκοπήσεις, όπου η καθεμία συνθέτει ευρύ σώμα πρωτογενούς βιβλιογραφίας. Στην ίδια κατηγορία εντάσσεται και η συμπερίληψη ενός preprint, το οποίο δεν έχει υποβληθεί σε κριτική αξιολόγηση από ομοτίμους και επομένως δεν φέρει την ίδια εγγύηση ποιότητας με τα δημοσιευμένα άρθρα· η επιλογή αιτιολογείται στην Ενότητα 3.2.1 και ο κίνδυνος περιορίζεται από το ελάχιστο μερίδιό του στο τελικό σύνολο (3,7%) και από την εφαρμογή σε αυτό των ίδιων κριτηρίων ποιότητας. Σημειώνεται, επίσης, η συγκέντρωση των δύο τρίτων του συνόλου σε δύο μόνο εκδοτικούς οίκους, η οποία ενδέχεται να ευνοεί ορισμένες ερευνητικές παραδόσεις έναντι άλλων.

Ως προς την υποκειμενικότητα της αξιολόγησης, η κατηγοριοποίηση, η βαθμολόγηση ποιότητας και η εξαγωγή δεδομένων έγιναν από έναν ερευνητή, χωρίς δεύτερο αξιολογητή ή μέτρηση συμφωνίας μεταξύ κριτών - έναν ουσιώδη περιορισμό που μετριάζεται με τη χρήση διαφανών, ρητά διατυπωμένων κριτηρίων.

Ως προς τη γενικευσιμότητα, πρέπει να διευκρινιστεί σε ποιο σύνολο στηρίζεται κάθε τύπος ευρήματος. Οι περιγραφικές κατανομές της Ενότητας 3.5 και η κατηγοριοποίηση της Ενότητας 4.1 υπολογίζονται επί του πλήρους συνόλου των 27 άρθρων. Αντιθέτως, η αναλυτική συγκριτική σύνθεση των Ενοτήτων 4.2 έως 4.5, και ιδίως η βαθμολόγηση ως προς τα είκοσι κριτήρια, στηρίζεται στο εστιακό υποσύνολο των δώδεκα αντιπροσωπευτικών άρθρων, με τα

υπόλοιπα δεκαπέντε να λειτουργούν ως σύνολο επικύρωσης κατά τα οριζόμενα στην Ενότητα 3.7. Τα συμπεράσματα της σύγκρισης πρέπει, συνεπώς, να διαβάζονται ως ενδείξεις που προκύπτουν από αυτό το επιλεγμένο υποσύνολο και όχι ως καθολικές γενικεύσεις για το σύνολο της βιβλιογραφίας.

Ως προς τη διαφάνεια της διαδικασίας επιλογής, οι ακριβείς αριθμοί εγγραφών σε κάθε ενδιάμεσο στάδιο - μετά την αναζήτηση, μετά το snowballing και μετά τον έλεγχο τίτλου και περίληψης - δεν καταγράφηκαν συστηματικά κατά την εκτέλεση και αναφέρονται ως τάξεις μεγέθους. Η πρακτική αυτή αποκλίνει από τα πρότυπα των πλήρων συστηματικών ανασκοπήσεων και περιορίζει την αναπαραγωγιμότητα της διαδικασίας, χωρίς ωστόσο να επηρεάζει την επαληθευσιμότητα του ίδιου του τελικού συνόλου, το οποίο παρατίθεται πλήρες στο Παράρτημα Α.

Τέλος, ως προς τον χρονικό περιορισμό, η αναζήτηση καλύπτει δημοσιεύσεις έως το 2025, οπότε μεταγενέστερες εξελίξεις δεν συμπεριλαμβάνονται και αποτελούν αντικείμενο μελλοντικής επικαιροποίησης.

Ολοκληρώνοντας, το κεφάλαιο αυτό όρισε μια διαφανή διαδικασία: οκτώ ερευνητικές ερωτήσεις, ρητά κριτήρια επιλογής και ποιότητας, και ένα τελικό σύνολο 27 πραγματικών, επαληθεύσιμων άρθρων, από τα οποία δώδεκα αναλύονται σε βάθος στο Κεφάλαιο 4 και δεκαπέντε λειτουργούν ως σύνολο επικύρωσης των ευρημάτων.

4. Ανάλυση

Το παρόν κεφάλαιο αποτελεί τον πυρήνα της επισκόπησης και οργανώνεται σε πέντε ενότητες: την κατηγοριοποίηση των άρθρων, την εις βάθος ανάλυση δώδεκα αντιπροσωπευτικών εργασιών, την παρουσίαση των κριτηρίων σύγκρισης, τα αποτελέσματα της σύγκρισης βάσει ενός αναλυτικού πίνακα, και τέλος την ερμηνεία των αποτελεσμάτων.

Οι ενότητες αυτές δεν στηρίζονται όλες στο ίδιο σύνολο δεδομένων, και η διάκριση είναι σημαντική για την ορθή ανάγνωση των ευρημάτων. Η κατηγοριοποίηση της Ενότητας 4.1 καλύπτει και τα 27 άρθρα που εντοπίστηκαν μέσω της μεθοδολογίας του Κεφαλαίου 3. Αντιθέτως, η εις βάθος ανάλυση της Ενότητας 4.2 και η συγκριτική αποτίμηση των Ενοτήτων 4.3 έως 4.5 στηρίζονται στο εστιακό υποσύνολο των δώδεκα αντιπροσωπευτικών εργασιών, δύο ανά κατηγορία, που επιλέχθηκαν με τα κριτήρια της Ενότητας 3.7· τα υπόλοιπα δεκαπέντε άρθρα λειτουργούν ως σύνολο επικύρωσης των ευρημάτων.

Μέσα από τη δομή αυτή απαντώνται οι επτά από τις οκτώ ερευνητικές ερωτήσεις της Ενότητας 3.1. Εξαίρεση αποτελεί η RQ6, που αφορά τα ανοιχτά ερευνητικά κενά και τις προτεινόμενες κατευθύνσεις αντιμετώπισής τους· η ερώτηση αυτή απαντάται στο Κεφάλαιο 5, το οποίο στηρίζεται ακριβώς στα ευρήματα της ανάλυσης που ακολουθεί.

4.1 Κατηγοριοποίηση των Επιλεγμένων Άρθρων (RQ1)

Τα άρθρα του τελικού συνόλου κατηγοριοποιήθηκαν σε έξι θεματικές ομάδες, με βάση την κύρια μέθοδο ή τεχνική που πραγματεύονται, και κάθε ομάδα αναλύθηκε περαιτέρω σε υποκατηγορίες που προέκυψαν επαγωγικά από το ίδιο το περιεχόμενο των άρθρων. Σε αντίθεση με την επισκόπηση του Κεφαλαίου 2, η οποία παρουσιάζει τις οικογένειες μεθόδων όπως τις αποτυπώνει η καθιερωμένη βιβλιογραφία, η ταξινόμηση που ακολουθεί προκύπτει από τη μελέτη των συγκεκριμένων εργασιών του συνόλου.

Τα δύο επίπεδα της ταξινόμησης έχουν, ωστόσο, διαφορετική φύση και διαφορετικό ρόλο, και η διάκριση αυτή είναι απαραίτητη για την ορθή ανάγνωση του Πίνακα 4.1. Το πρώτο επίπεδο, οι έξι οικογένειες, συνιστά κατηγοριοποίηση των ίδιων των μεθόδων και τεχνικών ανάπτυξης και είναι εκείνο που απαντά στην πρώτη ερευνητική ερώτηση. Το δεύτερο επίπεδο, οι υποκατηγορίες, δεν ταξινομεί τις μεθόδους αλλά τις εργασίες που τις επισκοπούν, ανάλογα με τη φύση και την εστίαση της καθεμιάς - ιστορική επισκόπηση, εμπειρική αποτίμηση, συστηματική χαρτογράφηση. Παρατίθεται επειδή προσφέρει στον αναγνώστη εικόνα του τι είδους τεκμήρια στηρίζουν κάθε οικογένεια, αλλά δεν αποτελεί απάντηση στη RQ1. Μια πλήρης ταξινόμηση των μεθόδων εντός κάθε οικογένειας θα απαιτούσε τη σύνθεση των επιμέρους ταξινομιών που προτείνει κάθε επισκόπηση του συνόλου, εγχείρημα που υπερβαίνει την εμβέλεια της παρούσας εργασίας και καταγράφεται ως κατεύθυνση μελλοντικής έρευνας.

Η κατηγοριοποίηση σε έξι οικογένειες παρέχει, τέλος, τον οργανωτικό άξονα για την υπόλοιπη ανάλυση.

Πίνακας 4.1: Πίνακας 4.1: Κατανομή των 27 άρθρων ανά οικογένεια μεθόδων και υποκατηγορία. Πηγή: επεξεργασία του συγγραφέα (βλ. Παράρτημα Α).

Κατηγορία	**Υποκατηγορία**	**Άρθρα**	**Ποσοστό**	**Κύρια Εστίαση**
Model-Driven Web Engineering	Ιστορική επισκόπηση και εξέλιξη του πεδίου	2	7,4%	Μοντελοκεντρική σχεδίαση, εξέλιξη προς IFML
	Εμπειρική αποτίμηση υιοθέτησης	1	3,7%	Χάσμα ακαδημαϊκής θεωρίας και βιομηχανικής πρακτικής
Agile & Scrum για Web	Επαναληπτικές μέθοδοι και εμπειρική τους τεκμηρίωση	4	14,8%	Sprints, ευελιξία, αποτίμηση οφελών
	Ενσωμάτωση με πλαίσια ωριμότητας και υβριδικές προσεγγίσεις	2	7,4%	Agile σε συνδυασμό με CMMI, νέφος και DevOps
DevOps & CI/CD	Αγωγοί συνεχούς ενσωμάτωσης, παράδοσης και διάταξης	2	7,4%	Αυτοματοποίηση αγωγού, συνεχής παράδοση
	Αντιστοίχιση δυνατοτήτων DevOps στον κύκλο ζωής	2	7,4%	Οφέλη, οργανωσιακές δυνατότητες, κάλυψη σταδίων
Microservices Architecture	Αρχές, μοτίβα και προκλήσεις μετάβασης	3	11,1%	Αποσύνθεση υπηρεσιών, ανεξάρτητη διάταξη
	Εμπειρική και αρχιτεκτονική αποτίμηση σε πραγματικά συστήματα	2	7,4%	Βιομηχανική πρακτική, εξέλιξη αρχιτεκτονικής
TDD & Automated Testing	Αποτίμηση της επίδρασης του TDD στην ποιότητα	3	11,1%	Μετα-αναλύσεις, πειράματα, παραγωγικότητα
	Δοκιμή εφαρμογών ιστού και ασφάλεια	4	14,8%	Αυτοματοποιημένες δοκιμές,

				σαρωτές ευπαθειών, AI
Frameworks, Γλώσσες & Ασφάλεια	Τεχνολογίες νέφους και cloud-native στοίβα	1	3,7%	Κύκλος ζωής cloud-native εφαρμογών
	Συγκριτική αποτίμηση γλωσσών και ποιότητας κώδικα	1	3,7%	JavaScript έναντι TypeScript, code smells
Σύνολο		*27*	*100%*	-

Η κατανομή δείχνει ότι η δοκιμή λογισμικού, το Agile και τα microservices συγκεντρώνουν το μεγαλύτερο ενδιαφέρον, ενώ οι μέθοδοι MDWE, παρότι ιστορικά θεμελιώδεις, εμφανίζονται πλέον με μικρότερη συχνότητα στην πρόσφατη βιβλιογραφία.

Δύο επιφυλάξεις είναι απαραίτητες ως προς την ερμηνεία της κατανομής αυτής. Πρώτον, πρόκειται για εύρημα που προκύπτει από το συγκεκριμένο, στοχευμένα επιλεγμένο σύνολο της παρούσας επισκόπησης και όχι από εξαντλητική καταμέτρηση της βιβλιογραφίας· η επιβεβαίωσή του απαιτεί πλήρη συστηματική ανασκόπηση με εξαντλητική κάλυψη των βάσεων, η οποία αποτελεί προφανή κατεύθυνση επέκτασης της εργασίας. Δεύτερον, δύο κατηγορίες - η Model-Driven Web Engineering και η ομάδα των frameworks, γλωσσών και ασφάλειας - εκπροσωπούνται από λιγότερα άρθρα σε σχέση με τις υπόλοιπες, γεγονός που περιορίζει τη στατιστική βαρύτητα των συμπερασμάτων που τις αφορούν. Για την πρώτη, η υποεκπροσώπηση αποτελεί εν μέρει το ίδιο το εύρημα, καθώς αντανακλά τη φθίνουσα παρουσία του πεδίου στην πρόσφατη βιβλιογραφία· για τη δεύτερη, οφείλεται κυρίως στην ετερογένεια του αντικειμένου, που δυσχεραίνει τον εντοπισμό εργασιών οι οποίες να το πραγματεύονται ενιαία. Η ενίσχυση και των δύο κατηγοριών, ώστε κάθε ομάδα να στηρίζεται σε τουλάχιστον τέσσερις εργασίες, καταγράφεται ως περιορισμός της παρούσας επισκόπησης. Στη συνέχεια, για κάθε κατηγορία αναλύονται δύο αντιπροσωπευτικά άρθρα, τα οποία επιλέχθηκαν από το τελικό σύνολο βάσει της διαδικασίας που προσδιορίστηκε στην Ενότητα 3.7.

4.2 Ανάλυση Αντιπροσωπευτικών Άρθρων (RQ1, RQ7)

Για κάθε κατηγορία επιλέχθηκαν δύο αντιπροσωπευτικά άρθρα, με τα κριτήρια που περιγράφηκαν στην Ενότητα 3.7. Η μορφή της ανάλυσης καθορίζεται από τη φύση του ίδιου του συνόλου: η πλειονότητα των εργασιών δεν αποτελούν πρωτογενείς μελέτες αλλά δευτερογενείς - συστηματικές ανασκοπήσεις, μελέτες χαρτογράφησης και μετα-αναλύσεις - καθεμία από τις οποίες συνθέτει ήδη δεκάδες ή εκατοντάδες πρωτογενείς εργασίες. Για τις εργασίες αυτές, το ζητούμενο δεν είναι η αποτίμηση μιας νέας μεθόδου που προτείνουν, αλλά η ανάδειξη της γνώσης που μεταφέρουν για το πεδίο.

Κάθε άρθρο παρουσιάζεται, συνεπώς, σε τρία σκέλη. Το πρώτο συνοψίζει το αντικείμενο και τη δηλωμένη εμβέλεια της εργασίας. Το δεύτερο αποτιμά κριτικά τι καλύπτεται στην πραγματικότητα και με ποιο βάθος, σε αντιδιαστολή με τα δηλωμένα, καθώς και τι παραμένει εκτός, στοιχείο που τροφοδοτεί την ανάλυση των ερευνητικών κενών του Κεφαλαίου 5. Το τρίτο, που αποτελεί και τον πυρήνα της ανάλυσης, συνθέτει τα ουσιώδη ευρήματα της εργασίας και επισημαίνει σε ποια από τα ερευνητικά ερωτήματα της Ενότητας 3.1 συνεισφέρει. Όπου το άρθρο αποτελεί πρωτογενή μελέτη - όπως στα πειράματα και στις εμπειρικές συγκρίσεις του συνόλου - η

ανάλυση εστιάζει αντιστοίχως στον σχεδιασμό της μελέτης και στην εγκυρότητα των αποτελεσμάτων της.

4.2.1 Model-Driven Web Engineering

Rossi et al. (2016) - 25 Years of Model-Driven Web Engineering

Η εργασία των Rossi et al. (2016) προσφέρει μια ιστορική επισκόπηση εικοσιπέντε ετών εξέλιξης των μεθόδων MDWE, από τα πρώτα εννοιολογικά μοντέλα έως το τυποποιημένο πρότυπο IFML (Interaction Flow Modeling Language). Οι συγγραφείς τεκμηριώνουν πώς κάθε γενιά μεθόδων επέκτεινε σταδιακά την κάλυψη των σταδίων του κύκλου ζωής, αναδεικνύοντας τη μετάβαση από τη WebML στο IFML ως το σημαντικότερο ορόσημο τυποποίησης του πεδίου, και συνδέουν την εξέλιξη αυτή με τις ευρύτερες τάσεις της μηχανικής λογισμικού.

Το κύριο πλεονέκτημα της εργασίας είναι η εκτενής και εις βάθος κάλυψη της ιστορικής εξέλιξης, με σαφή χαρτογράφηση των δυνατών και των αδύναμων σημείων κάθε μεθόδου και χρήσιμες συστάσεις για την ενσωμάτωσή τους με Agile πρακτικές. Ως προς τους περιορισμούς, πρέπει να γίνει μια διάκριση. Η επιφανειακή ανάλυση των σταδίων της διάταξης και της συντήρησης, καθώς και το χάσμα με τις σύγχρονες agile και cloud πρακτικές, αποτελούν περιορισμούς της ίδιας της επισκόπησης, η οποία επιλέγει να εστιάσει στη μοντελοποίηση και στη σχεδίαση. Αντιθέτως, η απουσία εμπειρικής τεκμηρίωσης δεν είναι αδυναμία της επισκόπησης αλλά διαπίστωσή της για το σύνολο του πεδίου: οι ίδιοι οι συγγραφείς καταγράφουν ότι οι μέθοδοι MDWE δεν έχουν υποβληθεί σε αυστηρή εμπειρική αποτίμηση σε πραγματικά περιβάλλοντα ανάπτυξης.

Πέρα από την ιστορική αφήγηση, η εργασία προσφέρει μια ουσιώδη διάγνωση για το ίδιο το πεδίο. Οι συγγραφείς επισημαίνουν ότι η καθιέρωση της IFML ως προτύπου από τον οργανισμό OMG σηματοδότησε τη μετάβαση από μια αμιγώς ιστοκεντρική λογική (WebML) σε ένα γενικευμένο πλαίσιο μοντελοποίησης διεπαφών, βασισμένο στο πρότυπο Model-View-Controller και εφαρμόσιμο και σε εφαρμογές επιφάνειας εργασίας και κινητών συσκευών. Το πλέον καίριο εύρημά τους, ωστόσο, αποτελεί ομολογία αδυναμίας του πεδίου: παρά την επιστημονική ωριμότητα και τη διάδοση σε δεκάδες πανεπιστήμια, οι μέθοδοι MDWE δεν έχουν υιοθετηθεί ευρέως στη βιομηχανία. Ως προς τα ερευνητικά ερωτήματα της παρούσας επισκόπησης, η εργασία συνεισφέρει κυρίως στη RQ1, καθώς καταγράφει τη διαδοχή των γενεών μεθόδων της οικογένειας - από τα πρώιμα εννοιολογικά μοντέλα και τη WebML έως την τυποποίηση με την IFML - και την αντίστοιχη επέκταση της κάλυψης σταδίων ανά γενιά· οι επιμέρους μέθοδοι της οικογένειας, όπως οι OOHDM και UWE, παρουσιάζονται στην Ενότητα 2.6.2, και στη RQ7, καθώς τεκμηριώνει ρητά την έλλειψη εμπειρικής αποτίμησης στο σύνολο του ρεύματος. Ως προς τη RQ3, η συνεισφορά της είναι ουσιαστικά αρνητική αλλά ενδεικτική: η ασφάλεια δεν αντιμετωπίζεται ως αυτοτελής διάσταση σε καμία από τις μεθόδους που εξετάζει. Τέλος, η διαπίστωση της περιορισμένης βιομηχανικής υιοθέτησης τροφοδοτεί την ανάλυση των ερευνητικών κενών του Κεφαλαίου 5.

Linaje et al. (2019) - Comparison of development methodologies in web applications

Η εργασία των Linaje et al. (2019) συνδυάζει μια συστηματική ανασκόπηση με βιομηχανική έρευνα ερωτηματολογίου, προκειμένου να συγκρίνει τις μεθόδους MDWE όχι μόνο θεωρητικά αλλά και ως προς την πραγματική τους υιοθέτηση. Το βασικό της εύρημα είναι ότι στην πράξη οι εταιρείες χρησιμοποιούν υβριδικές ή εμπειρικές προσεγγίσεις αντί για αυστηρές ακαδημαϊκές μεθόδους, με την UWE να εμφανίζεται ως η πλέον υιοθετημένη μεταξύ αυτών.

Το σημαντικότερο πλεονέκτημα της εργασίας είναι ο σπάνιος συνδυασμός θεωρητικής ανασκόπησης και εμπειρικής έρευνας, που της επιτρέπει να αναδείξει ρητά το χάσμα ανάμεσα στη θεωρία και την πράξη. Οι περιορισμοί της είναι ο γεωγραφικός περιορισμός του δείγματος της

έρευνας, ο οποίος επηρεάζει τη γενικευσιμότητα, και η σχεδόν πλήρης παράλειψη των σταδίων των δοκιμών, της διάταξης και της συντήρησης.

Πέρα από τη διάγνωση του χάσματος ανάμεσα στη θεωρία και την πράξη, η εργασία προσφέρει και μια χρήσιμη ποιοτική διαβάθμιση των ίδιων των μεθόδων. Οι συγγραφείς καταγράφουν ότι, μεταξύ των καθιερωμένων MDWE προσεγγίσεων, η UWE εμφανίζεται ως η πλέον αναγνωρίσιμη και υιοθετημένη στην πράξη, ακολουθούμενη από την OOHDM και τη WebML/IFML, ενώ ταυτόχρονα η πλειονότητα των εταιρειών δηλώνει ότι στηρίζεται σε προσαρμοσμένες, εσωτερικές παραλλαγές αντί για αυστηρή εφαρμογή οποιασδήποτε ακαδημαϊκής μεθόδου. Το εύρημα αυτό είναι κρίσιμο για την παρούσα επισκόπηση, διότι δεν αρκείται στη διαπίστωση ότι η θεωρία και η πράξη αποκλίνουν, αλλά δείχνει και τον μηχανισμό της απόκλισης: οι ομάδες δεν απορρίπτουν τις ιδέες των MDWE, αλλά τις αποδομούν και κρατούν επιλεκτικά όσα στοιχεία εξυπηρετούν τις ανάγκες τους. Ως προς τα ερευνητικά ερωτήματα, η εργασία συνεισφέρει στη RQ1, προσθέτοντας στη χαρτογράφηση των μεθόδων μια διαβάθμιση κατά βαθμό υιοθέτησης, και κυρίως στη RQ7, καθώς αποτελεί μία από τις ελάχιστες εργασίες του συνόλου που στηρίζει τα συμπεράσματά της σε πρωτογενή εμπειρικά δεδομένα από τη βιομηχανία. Η διαπίστωση ότι η πράξη κινείται προς υβριδικά, προσαρμόσιμα σχήματα αντί για «καθαρές» μεθόδους αποτελεί, βασικό τεκμήριο για την ανάλυση των ερευνητικών κενών του Κεφαλαίου 5.

4.2.2 Agile & Scrum

Torrecilla-Salinas et al. (2016) - Agile, Web Engineering and CMMI

Η εργασία των Torrecilla-Salinas et al. (2016) αναλύει, μέσα από μια συστηματική ανασκόπηση, τη συμβατότητα των Agile πρακτικών με τα επίπεδα ωριμότητας του μοντέλου CMMI, ειδικά για το πλαίσιο των εφαρμογών ιστού. Καλύπτει τη σχεδίαση, την υλοποίηση, τη δοκιμή και τη διαχείριση του έργου, δείχνοντας ότι οι Agile προσεγγίσεις μπορούν να επιτύχουν υψηλά επίπεδα ωριμότητας χωρίς να θυσιάσουν την ευελιξία τους.

Το πλεονέκτημα της εργασίας είναι ότι γεφυρώνει τον κόσμο του Agile με το CMMI ειδικά για τον ιστό, με σαφή και τεκμηριωμένη χαρτογράφηση των πρακτικών. Ως προς τους περιορισμούς, χρειάζεται διάκριση ανάμεσα στην κάλυψη και στην οπτική. Η εργασία καλύπτει τρία στάδια του κύκλου ζωής - τη σχεδίαση, την υλοποίηση και τη δοκιμή - αλλά τα προσεγγίζει από τη σκοπιά της οργανωσιακής ωριμότητας και της διαχείρισης του έργου: εξετάζει δηλαδή πώς οργανώνονται και τεκμηριώνονται οι πρακτικές κάθε σταδίου, και όχι τις τεχνικές που εφαρμόζονται εντός του. Η μηχανική απαιτήσεων εξετάζεται μόνο στο μέτρο που αφορά την οργάνωση και την τεκμηρίωση των πρακτικών, συνεξεταζόμενη με τη σχεδίαση σύμφωνα με το μοντέλο της Ενότητας 2.3.1, ενώ εκτός εμβέλειας παραμένουν η διάταξη, οι τεχνολογίες νέφους και η ασφάλεια. Πρέπει να τονιστεί ότι πρόκειται για περιορισμό της εμβέλειας της συγκεκριμένης ανασκόπησης και όχι για διαπίστωση ότι οι ευέλικτες μέθοδοι αγνοούν τα ζητήματα αυτά. Η κάλυψη των τεχνολογιών νέφους και της ασφάλειας δεν αποτιμάται, εξάλλου, αφηγηματικά και κατά περίπτωση: αποτελεί αντικείμενο ρητών κριτηρίων σύγκρισης - των C8 και C13 έως C15 για το νέφος και τη διάταξη και του C12 για την ασφάλεια - τα οποία εφαρμόζονται ομοιόμορφα σε όλες τις εργασίες του εστιακού υποσυνόλου στην Ενότητα 4.3.

Η μεθοδολογική συνεισφορά της εργασίας αξίζει ιδιαίτερης μνείας. Οι συγγραφείς δεν περιορίζονται σε μια απλή ανασκόπηση, αλλά ορίζουν ένα σχήμα χαρακτηρισμού (characterization schema) με το οποίο συγκρίνουν συστηματικά τις υπάρχουσες προσεγγίσεις ως προς την κάλυψη των επιπέδων ωριμότητας του CMMI, καταλήγοντας στη διαπίστωση ότι οι ευέλικτες πρακτικές μπορούν να αντιστοιχηθούν έως και στα υψηλότερα επίπεδα, αρκεί να συνοδεύονται από κατάλληλα τεκμηριωμένες διαδικασίες. Η ίδια ερευνητική ομάδα προχώρησε αργότερα, εκτός του συνόλου της παρούσας εργασίας, στην πρόταση του πλαισίου NDT-Agile (Torrecilla-Salinas et al., 2017), μιας CMMI-συμβατής ευέλικτης μεθόδου για τη μηχανική ιστού -

γεγονός που επιβεβαιώνει ότι η ανασκόπηση του 2016 αποτέλεσε το θεμέλιο για την ανάπτυξη ενός εφαρμόσιμου πλαισίου. Ως προς τα ερευνητικά ερωτήματα, η εργασία συνεισφέρει στη RQ1, καθώς τεκμηριώνει ότι οι ευέλικτες μέθοδοι καλύπτουν στην πράξη περισσότερα του ενός στάδια του κύκλου ζωής, και στη RQ2, προσφέροντας ένα έτοιμο σχήμα κριτηρίων σύγκρισης που επηρέασε τον σχεδιασμό των κριτηρίων της Ενότητας 4.3. Αντιθέτως, δεν προσφέρει υλικό για τη RQ3 και τη RQ4, καθώς η ασφάλεια και το νέφος βρίσκονται εκτός της εμβέλειάς της. Η διαπίστωση ότι η γεφύρωση μεταξύ ευελιξίας και πειθαρχίας είναι εφικτή αποτελεί πρώιμη απόπειρα προς την κατεύθυνση των ολιστικών, υβριδικών πλαισίων που συζητούνται στο Κεφάλαιο 5.

Pastrana et al. (2025) - Best Practices Evidenced for Software Development Based on DevOps and Scrum

Η εργασία των Pastrana et al. (2025) πραγματοποιεί θεματική ανάλυση 111 άρθρων των βάσεων Scopus και Web of Science, για την περίοδο 2014–2023, με στόχο τον εντοπισμό των βέλτιστων πρακτικών που συνδυάζουν το Scrum με το DevOps. Εστιάζει ειδικότερα στις πολύ μικρές οντότητες (Very Small Entities, VSEs), οι οποίες συχνά στερούνται πόρων και εξειδικευμένης τεχνογνωσίας - εστίαση που καθιστά τα ευρήματά της ιδιαίτερα σχετικά με τη μεγάλη πλειονότητα των ομάδων ανάπτυξης εφαρμογών ιστού. Σημειώνεται ότι η εργασία τοποθετείται στο όριο μεταξύ της παρούσας κατηγορίας και εκείνης του DevOps· εντάσσεται εδώ επειδή η οπτική της είναι διαδικαστική, με το DevOps να εξετάζεται ως συμπλήρωμα του Scrum και όχι ως αυτοτελές αντικείμενο.

Τα πλεονεκτήματα της εργασίας είναι το σύγχρονο και εκτενές σώμα βιβλιογραφίας που καλύπτει, τα ποσοτικά τεκμηριωμένα ευρήματά της και η ταυτόχρονη κάλυψη Scrum και DevOps. Ως προς τους περιορισμούς, η εργασία δεν εστιάζει αποκλειστικά στις εφαρμογές ιστού, ενώ η διάσταση της ασφάλειας καλύπτεται ανεπαρκώς. Όπως και στην προηγούμενη εργασία, η έλλειψη αυτή αφορά την εμβέλεια της συγκεκριμένης ανασκόπησης και δεν θεμελιώνει το συμπέρασμα ότι η ασφάλεια απουσιάζει από την οικογένεια των ευέλικτων μεθόδων συνολικά· η συστηματική αποτίμηση της διάστασης αυτής γίνεται στην Ενότητα 4.5, με βάση το κριτήριο C12 και το σύνολο των εργασιών.

Τα ευρήματα της εργασίας συνοψίζονται σε δύο πεντάδες: πέντε κρίσιμοι παράγοντες επιτυχίας για το Scrum - ενεργός εμπλοκή των εμπλεκομένων μερών, έγκαιρη και συνεχής ανατροφοδότηση, διαφανές κανάλι επικοινωνίας, διαρκής μέτρηση της ποιότητας και ποιοτικό παραδοτέο - και πέντε θεμελιώδεις πρακτικές DevOps: διαχείριση εκδόσεων, αυτοματοποίηση του αγωγού, κουλτούρα συνεργασίας, συνεχής ενσωμάτωση και αυτοματοποιημένες δοκιμές, με τη συνεχή ενσωμάτωση να αναδεικνύεται ως η πλέον κυρίαρχη και επαναλαμβανόμενη πρακτική στη βιβλιογραφία. Ως προς τα ερευνητικά ερωτήματα, η εργασία συνεισφέρει στη RQ1, δείχνοντας ότι ο συνδυασμός ευέλικτων και λειτουργικών πρακτικών καλύπτει το μεγαλύτερο μέρος του κύκλου ζωής. Η συνεισφορά της στη RQ7 είναι έμμεση: δεν διενεργεί η ίδια εμπειρική αποτίμηση, αλλά συναθροίζει τη συχνότητα με την οποία οι πρωτογενείς μελέτες αναφέρουν κάθε πρακτική - με τη συνεχή ενσωμάτωση να αναδεικνύεται ως η πλέον επαναλαμβανόμενη - προσφέροντας έτσι δευτερογενή ένδειξη για το ποιες πρακτικές τεκμηριώνονται στην πράξη. Η συνολική εικόνα της εμπειρικής τεκμηρίωσης, ανά τύπο και ανά οικογένεια, συνθέτεται στην Ενότητα 4.5.6. Η περιορισμένη κάλυψη της ασφάλειας, τέλος, αποτελεί ένδειξη που τροφοδοτεί την ανάλυση των ερευνητικών κενών του Κεφαλαίου 5.

4.2.3 DevOps & CI/CD

Shahin et al. (2017) - Continuous Integration, Delivery and Deployment: A Systematic Review

Η εργασία των Shahin, Babar και Zhu (2017) αποτελεί την πλέον αναφερόμενη συστηματική ανασκόπηση στον χώρο του CI/CD. Αναλύει περίπου εβδομήντα πρωτογενείς μελέτες, χαρτογραφώντας συστηματικά τις προσεγγίσεις, τα εργαλεία και τις προκλήσεις της συνεχούς ενσωμάτωσης, παράδοσης και διάταξης, όπως οι ασταθείς δοκιμές και οι μεγάλοι χρόνοι μεταγλώττισης.

Το πλεονέκτημα της εργασίας είναι η αυστηρή της μεθοδολογία, η σαφής σύνδεση των πρακτικών CI/CD με τα στάδια του κύκλου ζωής, και η πλούσια καταγραφή εργαλείων και προκλήσεων. Ως προς τους περιορισμούς, η σχεδίαση και η ανάλυση απαιτήσεων μένουν εκτός εμβέλειας. Επιπλέον, το τεχνολογικό τοπίο στο οποίο στηρίζεται έχει εν μέρει μεταβληθεί: οι μελέτες που συνθέτει προηγούνται της ευρείας καθιέρωσης της ενορχήστρωσης δοχείων και των cloud-native πρακτικών, εξέλιξη που αποτυπώνεται στις νεότερες εργασίες του παρόντος συνόλου. Συγκεκριμένα, οι Deng et al. (2024) καταγράφουν έναν κύκλο ζωής cloud-native εφαρμογών όπου η ενορχήστρωση αποτελεί αυτοτελές στάδιο, ενώ οι Amaro et al. (2025) και El Aouni et al. (2024) αντιστοιχούν τις δυνατότητες DevOps σε ευρύτερο φάσμα σταδίων απ' ό,τι η ανασκόπηση του 2017. Η διαπίστωση αυτή δεν ακυρώνει τα ευρήματά της, αλλά οριοθετεί το χρονικό τους πλαίσιο.

Η επισκόπηση δομείται γύρω από τέσσερα ρητά ερευνητικά ερωτήματα: ποιες προσεγγίσεις και ποια εργαλεία υποστηρίζουν τις συνεχείς πρακτικές, ποια εργαλεία χρησιμοποιούνται ειδικότερα για τον σχεδιασμό και την υλοποίηση των αγωγών διάταξης, ποιες προκλήσεις εμποδίζουν την υιοθέτησή τους και ποιες πρακτικές οδηγούν σε επιτυχή εφαρμογή. Η γνώση που μεταφέρει στην παρούσα επισκόπηση είναι κυρίως εννοιολογική και ταξινομική: αποσαφηνίζει τη σχέση μεταξύ συνεχούς ενσωμάτωσης, παράδοσης και διάταξης - τριών όρων που συχνά χρησιμοποιούνται ασαφώς - και συνδέει κάθε πρακτική με συγκεκριμένα εργαλεία και οργανωσιακές προϋποθέσεις. Ως προς τα ερευνητικά ερωτήματα, η συνεισφορά της είναι άμεση και διπλή ως προς τη RQ1: αφενός προσδιορίζει τις τεχνικές που συγκροτούν την οικογένεια - τη συνεχή ενσωμάτωση, τη συνεχή παράδοση και τη συνεχή διάταξη - και τις διακρίνει ρητά μεταξύ τους, προσφέροντας έτσι την ταξινομική βάση της κατηγορίας· αφετέρου τεκμηριώνει ότι οι πρακτικές αυτές καλύπτουν ενιαία την υλοποίηση, τη δοκιμή και τη διάταξη. Συνεισφέρει επίσης στη RQ5, καταγράφοντας εργαλεία που εκτείνονται από τα πλαίσια μοναδιαίων δοκιμών, όπως τα JUnit και NUnit, έως τα εργαλεία διαχείρισης της διαμόρφωσης της υποδομής, όπως το Chef· η συνολική καταγραφή των εργαλείων όλων των εργασιών του συνόλου, οργανωμένη ανά στάδιο, παρατίθεται στην Ενότητα 4.5.5. Συνεισφέρει, τέλος, στη RQ4, καθώς η συνεχής διάταξη προϋποθέτει υποδομή νέφους· η σχέση του νέφους με τον κύκλο ζωής δεν περιορίζεται πάντως στη διάταξη, όπως δείχνουν και άλλες εργασίες του συνόλου: οι Deng et al. (2024) τεκμηριώνουν cloud-native κάλυψη και της υλοποίησης και της συντήρησης, ενώ οι Li et al. (2024) καταγράφουν τα περιβάλλοντα δοκιμών στο νέφος - διαστάσεις που αποτιμώνται συστηματικά στην Ενότητα 4.5.4. Αντιθέτως, η ασφάλεια (RQ3) εμφανίζεται μόνο περιφερειακά: μόλις δύο από τις εξήντα εννέα μελέτες την πραγματεύονται, ως επιμέρους τεχνικό ζήτημα και όχι ως οργανωτική διάσταση του αγωγού.

El Aouni et al. (2024) - A Systematic Literature Review on Agile, Cloud, and DevOps integration

Η εργασία των El Aouni et al. (2024) εξετάζει, μέσα από συστηματική ανασκόπηση τριάντα ενός άρθρων, την ενσωμάτωση των τριών παραδειγμάτων του Agile, του Cloud και του DevOps. Συνθέτει τα ευρήματα των μελετών αυτών σε ένα ενιαίο πλαίσιο ενσωμάτωσης, αναδεικνύοντας

τη νοοτροπία και τους ανθρώπινους παράγοντες ως τους πρωταρχικούς παράγοντες επιτυχίας, και καλύπτει τέσσερα από τα πέντε στάδια του κύκλου ζωής. Εξ ορισμού του αντικειμένου της, η εργασία τέμνει τρεις από τις κατηγορίες της παρούσας επισκόπησης - τις ευέλικτες μεθόδους, το DevOps και την κατηγορία των frameworks, γλωσσών και ασφάλειας, στην οποία εντάσσονται οι τεχνολογίες νέφους· εντάσσεται στην παρούσα κατηγορία επειδή το DevOps αποτελεί τον συνδετικό άξονα της ανάλυσής της και το σημείο όπου εντοπίζει τις ουσιώδεις προκλήσεις.

Τα πλεονεκτήματα της εργασίας είναι ο σύγχρονος χαρακτήρας της, η εμπειρική επαλήθευση του προτεινόμενου πλαισίου και η προσοχή που δίνει στους ανθρώπινους και οργανωσιακούς παράγοντες. Οι περιορισμοί της είναι το σχετικά μικρό σώμα βιβλιογραφίας που αναλύει και η ελλιπής κάλυψη της αρχιτεκτονικής σχεδίασης - περιορισμοί που αφορούν την εμβέλεια της συγκεκριμένης ανασκόπησης και όχι την οικογένεια των πρακτικών DevOps συνολικά.

Σε βαθύτερη ανάγνωση, η συνεισφορά της εργασίας έγκειται στην ανάδειξη της πολιτισμικής και όχι απλώς της τεχνικής διάστασης της ενσωμάτωσης. Οι συγγραφείς εντοπίζουν ως κυρίαρχες προκλήσεις τον πολλαπλασιασμό των εργαλείων (tool proliferation), την κατακερματισμένη ροή εργασίας και την πολυπλοκότητα της αλυσίδας εργαλείων, και προτείνουν ως απάντηση την ευθυγράμμιση με το μοντέλο CALMS (Culture, Automation, Lean, Measurement, Sharing), υποστηριζόμενη από κοινά εργαλεία και διαλειτουργικές ομάδες Scrum. Το πλέον ενδιαφέρον εύρημά τους είναι ότι η νοοτροπία και η οργανωσιακή συνοχή αναδεικνύονται ως ισχυρότεροι παράγοντες επιτυχίας από τα ίδια τα τεχνικά εργαλεία. Η εργασία επισημαίνει επίσης τη δυσκολία αξιολόγησης λόγω της απουσίας υβριδικών μετρικών και την πρόσθετη πολυπλοκότητα που εισάγει η ενσωμάτωση της ασφάλειας (DevSecOps). Ως προς τα ερευνητικά ερωτήματα, η εργασία συνεισφέρει στη RQ1, τεκμηριώνοντας πώς η συνδυασμένη υιοθέτηση των τριών παραδειγμάτων διευρύνει την κάλυψη του κύκλου ζωής· στη RQ4, καθώς το υπολογιστικό νέφος αποτελεί ένα από τα τρία παραδείγματα που εξετάζει, με αποτέλεσμα η εργασία να πραγματεύεται ρητά τη σχέση των ευέλικτων και των λειτουργικών πρακτικών με τα περιβάλλοντα νέφους και όχι μόνο την οργάνωση της ανάπτυξης· και στη RQ3, καθώς είναι από τις ελάχιστες του συνόλου που αντιμετωπίζουν την ασφάλεια ως διακριτή διάσταση ενσωμάτωσης και όχι ως επιμέρους πρόκληση. Η διαπίστωση ότι η ενσωμάτωση της ασφάλειας παραμένει ανοιχτό ζήτημα τροφοδοτεί την ανάλυση των ερευνητικών κενών του Κεφαλαίου 5.

4.2.4 Microservices

Velepucha & Flores (2023) - A Survey on Microservices Architecture

Η εργασία των Velepucha και Flores (2023) προσφέρει μια ολοκληρωμένη επισκόπηση των αρχών, των αρχιτεκτονικών προτύπων και των προκλήσεων μετανάστευσης προς την αρχιτεκτονική microservices. Τα πρότυπα αυτά δεν περιορίζονται στη σχεδίαση, καθώς καθένα τους συνοδεύεται από συγκεκριμένες συνέπειες στα επόμενα στάδια: η πύλη API και η ανακάλυψη υπηρεσιών προδιαγράφουν τον τρόπο υλοποίησης της επικοινωνίας, ο διακόπτης κυκλώματος απαιτεί δοκιμή της συμπεριφοράς του συστήματος υπό αστοχία, και η ανεξαρτησία των υπηρεσιών προϋποθέτει δοχειοποίηση και ενορχήστρωση κατά τη διάταξη. Με τον τρόπο αυτό η εργασία συνδέει τα πρότυπα με τα στάδια της σχεδίασης, της υλοποίησης, της δοκιμής και της διάταξης, και παρέχει έναν πρακτικό οδικό χάρτη για τη μετάβαση από μονολιθικές σε κατανεμημένες αρχιτεκτονικές.

Το πλεονέκτημα της εργασίας είναι η πλούσια κάλυψη τεσσάρων σταδίων του κύκλου ζωής, με πρότυπα και παραδείγματα αντλημένα από πραγματικές εταιρείες. Η κύρια αδυναμία της είναι ότι η συντήρηση και η παρατηρησιμότητα καλύπτονται ανεπαρκώς, παρότι αποτελούν κρίσιμες πτυχές της λειτουργίας των microservices. Το πρώτο στάδιο του κύκλου ζωής, εξάλλου, προσεγγίζεται αποκλειστικά από την πλευρά της σχεδίασης: η εργασία δεν πραγματεύεται τη μηχανική απαιτήσεων, δηλαδή τον τρόπο με τον οποίο οι απαιτήσεις οδηγούν στην επιλογή και

στην οριοθέτηση των υπηρεσιών. Η διάσταση αυτή δεν αποτιμάται κατά περίπτωση αλλά συστηματικά, μέσω των κριτηρίων C1 έως C3 της Ενότητας 4.3, τα οποία εφαρμόζονται ομοιόμορφα σε όλες τις εργασίες του εστιακού υποσυνόλου.

Χαρακτηριστικό της συνεισφοράς της εργασίας είναι ότι δεν περιορίζεται στην περιγραφή της αρχιτεκτονικής, αλλά θεμελιώνει τα microservices στις αρχές του αντικειμενοστρεφούς παραδείγματος - τη λειτουργική και την αντικειμενοστρεφή αποσύνθεση και την απόκρυψη πληροφορίας - δείχνοντας πώς οι ίδιες αυτές αρχές διέπουν τόσο τις μονολιθικές όσο και τις κατανεμημένες αρχιτεκτονικές. Η αξία της διαπίστωσης αυτής δεν έγκειται στην ίδια τη διατύπωση των αρχών, οι οποίες είναι γνωστές και προκύπτουν σε μεγάλο βαθμό από τον ορισμό της αρχιτεκτονικής, αλλά στη ρητή τεκμηρίωση της συνέχειας: οι συγγραφείς δείχνουν ότι η μετανάστευση σε microservices δεν συνιστά ρήξη με τις θεμελιώδεις αρχές σχεδίασης λογισμικού, αλλά εφαρμογή τους σε διαφορετικό επίπεδο κοκκίωσης. Στο πλαίσιο αυτό εντοπίζουν και ένα κενό στο σώμα της γνώσης: ενώ η βιβλιογραφία της μετανάστευσης είναι πλούσια σε τεχνικές και εργαλεία, η μεθοδολογική καθοδήγηση για το ποιες αρχές οφείλουν να υιοθετούν οι ομάδες κατά την υλοποίηση παραμένει ασαφής. Ως προς τα ερευνητικά ερωτήματα, η εργασία συνεισφέρει στη RQ1 σε δύο επίπεδα: αναγνωρίζει και ονοματίζει τα αρχιτεκτονικά πρότυπα που συγκροτούν την οικογένεια, ταξινομώντας τα κατά τον ρόλο τους - επικοινωνία, ανθεκτικότητα, ανακάλυψη και διάταξη - και τεκμηριώνει την κάλυψη τεσσάρων σταδίων από αυτά· στη RQ4, μέσα από τη σύνδεση με δοχεία και ενορχήστρωση· και στη RQ5, μέσω της καταγραφής των σχετικών εργαλείων. Η παρατήρηση ότι η αρχιτεκτονική τεκμηρίωση προηγείται της μεθοδολογικής καθοδήγησης συζητείται περαιτέρω στην Ενότητα 4.5 και τροφοδοτεί την ανάλυση των ερευνητικών κενών του Κεφαλαίου 5.

Zhou et al. (2023) - Revisiting the practices and pains of microservice architecture in reality

Η εργασία των Zhou et al. (2023) αποτελεί ποιοτική εμπειρική μελέτη σε είκοσι εταιρείες, με στόχο την αντιπαραβολή των ακαδημαϊκά διατυπωμένων αρχών των microservices με τη βιομηχανική πρακτική. Το βασικό της εύρημα είναι ότι οι αρχές αυτές γίνονται μεν αποδεκτές σε επίπεδο δηλωμένης πρόθεσης, αλλά η εφαρμογή τους προσκρούει σε συγκεκριμένα εμπόδια, με κυριότερα τις κατανεμημένες συναλλαγές (distributed transactions) και την παρατηρησιμότητα· η αποδοχή, δηλαδή, δεν συνεπάγεται υιοθέτηση.

Το σημαντικότερο πλεονέκτημα της εργασίας είναι η σπάνια εμπειρική τεκμηρίωση σε πραγματικά βιομηχανικά περιβάλλοντα, που της προσδίδει υψηλή πρακτική αξία, καθώς και τα πλούσια ποιοτικά δεδομένα της. Οι περιορισμοί της είναι η απουσία ποσοτικών μετρήσεων και η πιθανή μεροληψία στην επιλογή του δείγματος των εταιρειών.

Από τη συνθετική κωδικοποίηση των συνεντεύξεων, οι συγγραφείς εξάγουν οκτώ ζεύγη «πρακτικής και πόνου» (practices and pains), αναδεικνύοντας ότι κάθε όφελος της αρχιτεκτονικής συνοδεύεται από αντίστοιχο κόστος: η ανεξαρτησία των υπηρεσιών φέρνει πολυπλοκότητα στη διαχείριση των κατανεμημένων δεδομένων, η ευελιξία της διάταξης φέρνει δυσκολία στην παρατηρησιμότητα, και αντιστοίχως για τα υπόλοιπα ζεύγη. Επιπλέον, εντοπίζουν πέντε αποφάσεις που οι μηχανικοί οφείλουν να σταθμίσουν - την αποτίμηση των κινήτρων της μετάβασης, το μέγεθος των υπηρεσιών, τα πρωτόκολλα επικοινωνίας, την τεχνολογική διαφοροποίηση και τη διαχείριση των βάσεων δεδομένων - και πέντε ερευνητικές κατευθύνσεις, μεταξύ των οποίων η συστηματική αξιολόγηση, ο οργανωσιακός μετασχηματισμός και η διάγνωση σφαλμάτων σε κατανεμημένα περιβάλλοντα.

Ιδιαίτερα διαφωτιστικό είναι το εύρημα ότι οι εταιρείες εξακολουθούν να χρησιμοποιούν παραδοσιακές αντικειμενοστρεφείς τεχνικές μοντελοποίησης και τις ίδιες υποκειμενικές μεθόδους εκτίμησης μεγέθους και προσπάθειας που εφάρμοζαν στις μονολιθικές αρχιτεκτονικές. Το εύρημα αυτό χρήζει προσεκτικής ερμηνείας: δεν σημαίνει ότι απουσιάζουν προσεγγίσεις σχεδίασης και

μετανάστευσης - η βιβλιογραφία είναι πλούσια σε αυτές, όπως δείχνει και η προηγούμενη εργασία - αλλά ότι οι προσεγγίσεις αυτές δεν έχουν υιοθετηθεί στην πράξη. Πρόκειται, συνεπώς, για ζήτημα υιοθέτησης και όχι διαθεσιμότητας. Οι ίδιοι οι συγγραφείς δεν αναλύουν συστηματικά τα αίτια, καταγράφουν ωστόσο μια σαφή ένδειξη: όσοι επιχείρησαν να εφαρμόσουν δομημένες μεθόδους, όπως το Domain-Driven Design, τις χαρακτήρισαν υπερβολικά αφηρημένες και χωρίς συγκεκριμένο οδηγό εφαρμογής, με αποτέλεσμα οι αποφάσεις αποσύνθεσης να λαμβάνονται τελικά εμπειρικά.

Ως προς τα ερευνητικά ερωτήματα, η εργασία συνεισφέρει κυρίως στη RQ7, αποτελώντας μία από τις ελάχιστες πρωτογενείς εμπειρικές μελέτες του συνόλου, και στη RQ1, καθώς αποτυπώνει ποια στάδια καλύπτονται πράγματι στη βιομηχανική πρακτική. Η διάσταση μεταξύ διαθέσιμων και υιοθετημένων μεθόδων αποτελεί βασικό τεκμήριο για την ανάλυση του Κεφαλαίου 5.

4.2.5 TDD & Automated Testing

Rafique & Mišić (2013) - The Effects of TDD on External Quality and Productivity: A Meta-Analysis

Η εργασία των Rafique και Mišić (2013) είναι μια μετα-ανάλυση είκοσι επτά ελεγχόμενων πειραμάτων σχετικά με το Test-Driven Development, με στόχο τη συστηματική αποτίμηση της επίδρασής του στην ποιότητα και την παραγωγικότητα. Το συνολικό της συμπέρασμα είναι ότι το TDD έχει μικρή θετική επίδραση στην εξωτερική ποιότητα του λογισμικού και σχεδόν μηδενική διακριτή επίδραση στην παραγωγικότητα, στηριζόμενη σε ισχυρή στατιστική βάση.

Το πλεονέκτημα της εργασίας είναι η αυστηρή στατιστική μεθοδολογία της, με ξεχωριστή ανάλυση των ακαδημαϊκών και των βιομηχανικών μελετών. Οι αδυναμίες της είναι ο σχετικά μικρός αριθμός των ελεγχόμενων πειραμάτων που συνθέτει, ιδίως κατά τον επιμερισμό τους σε υποομάδες, και το γεγονός ότι τα εργαλεία που αφορούν ειδικά τις εφαρμογές ιστού δεν αναλύονται ξεχωριστά.

Το βαθύτερο ενδιαφέρον της μετα-ανάλυσης βρίσκεται στην ανάλυση υποομάδων (subgroup analysis), η οποία αποκαλύπτει ότι η συνολική, μέτρια θετική εικόνα του TDD κρύβει σημαντικές διαφοροποιήσεις. Οι συγγραφείς διαπιστώνουν ότι τόσο η βελτίωση της ποιότητας όσο και η πτώση της παραγωγικότητας είναι αισθητά εντονότερες στις βιομηχανικές μελέτες σε σχέση με τις ακαδημαϊκές. Η μεγαλύτερη μείωση παραγωγικότητας παρατηρείται εκεί όπου η διαφορά στην προσπάθεια δοκιμών μεταξύ της ομάδας TDD και της ομάδας ελέγχου είναι σημαντική - εύρημα που υποδεικνύει ότι το κόστος του TDD σε χρόνο προκύπτει κυρίως από τον πρόσθετο όγκο δοκιμών και όχι από την πρακτική καθαυτή. Ιδιαίτερα διαφωτιστική είναι, τέλος, η στατιστικά σημαντική θετική συσχέτιση μεταξύ του μεγέθους της εργασίας και του μεγέθους της βελτίωσης στην ποιότητα: το όφελος του TDD αυξάνεται με την κλίμακα του έργου, γεγονός που εξηγεί εν μέρει γιατί τα αποτελέσματα των ακαδημαϊκών πειραμάτων, τα οποία στηρίζονται σε μικρές εργασίες, υποεκτιμούν συστηματικά την επίδρασή του.

Ως προς τα ερευνητικά ερωτήματα, η εργασία συνεισφέρει κυρίως στη RQ7, καθώς αποτελεί τη μεθοδολογικά αυστηρότερη εμπειρική αποτίμηση ολόκληρου του συνόλου, με ποσοτική σύνθεση ελεγχόμενων πειραμάτων· και στη RQ1, καθώς τεκμηριώνει ότι η επίδραση του TDD εκτείνεται πέρα από το στάδιο της δοκιμής, στην ίδια την υλοποίηση. Δεν προσφέρει υλικό για τη RQ3, τη RQ4 και τη RQ5, αφού η ασφάλεια, το νέφος και τα εργαλεία για εφαρμογές ιστού βρίσκονται εκτός της εμβέλειάς της. Η διαπίστωση, τέλος, ότι ακόμη και οι πλέον μελετημένες τεχνικές εξετάζονται μεμονωμένα, χωρίς ενσωμάτωση σε ένα ολοκληρωμένο πλαίσιο ανάπτυξης, τροφοδοτεί την ανάλυση των ερευνητικών κενών του Κεφαλαίου 5.

Li et al. (2024) - A Survey on Web Application Testing: A Decade of Evolution

Η εργασία των Li et al. (2024) αποτελεί μια εκτενή επισκόπηση 314 πρωτογενών μελετών για τη δοκιμή εφαρμογών ιστού την περίοδο 2014–2023. Καταγράφει ότι το Selenium χρησιμοποιείται ευρέως για την αυτοματοποίηση των λειτουργικών δοκιμών σε διαφορετικά προγράμματα περιήγησης, ότι το 29,94% των μελετών εστιάζει στην ανίχνευση ευπαθειών ασφάλειας, και αναδεικνύει την αυξανόμενη αξιοποίηση τεχνικών μηχανικής μάθησης για την αυτοματοποίηση των δοκιμών. Σημειώνεται ότι το ποσοστό του 29,94% αφορά τον θεματικό προσανατολισμό των μελετών που πραγματεύονται την ανίχνευση ευπαθειών και όχι τη χρήση κάποιου συγκεκριμένου εργαλείου· η επισκόπηση καταγράφει χωριστά τα εργαλεία που αξιοποιούνται ανά κατηγορία δοκιμών.

Τα πλεονεκτήματα της εργασίας είναι το μεγάλο και σύγχρονο σώμα βιβλιογραφίας που καλύπτει, καθώς και η ανάδειξη των αναδυόμενων τάσεων της AI-ενισχυμένης δοκιμής. Ο κύριος περιορισμός της είναι η αποκλειστική εστίασή της στο στάδιο των δοκιμών, χωρίς κάλυψη των υπόλοιπων σταδίων του κύκλου ζωής. Πρόκειται, και εδώ, για περιορισμό της εμβέλειας της συγκεκριμένης επισκόπησης και όχι για χαρακτηριστικό της οικογένειας μεθόδων στην οποία εντάσσεται: η ανάπτυξη καθοδηγούμενη από δοκιμές επηρεάζει εξ ορισμού και τη σχεδίαση και την υλοποίηση, αφού η δοκιμή προηγείται του κώδικα και διαμορφώνει τη δομή του, ενώ οι αυτοματοποιημένες δοκιμές εντάσσονται στους αγωγούς συνεχούς ενσωμάτωσης - κάλυψη που αποτυπώνεται και στον Πίνακα 2.4.

Η μεθοδολογική βάση της μελέτης είναι ιδιαίτερα ισχυρή: οι συγγραφείς οργανώνουν την επισκόπηση γύρω από τα βασικά βήματα της δοκιμής εφαρμογών ιστού - τη δημιουργία και εκτέλεση δοκιμαστικών περιπτώσεων (test-case generation and execution) και την αξιολόγηση και αποτίμησή τους. Ένα αξιοσημείωτο εύρημα είναι η αναδυόμενη αξιοποίηση περιβαλλόντων δοκιμών βασισμένων στο νέφος - με πλατφόρμες όπως AWS, Google Cloud Platform και Microsoft Azure να επιτρέπουν την κατ' απαίτηση, γεωγραφικά κατανεμημένη εκτέλεση δοκιμών υπό ρεαλιστικές συνθήκες φόρτου - η οποία, ωστόσο, παραμένει περιορισμένη έναντι της συντριπτικά επικρατούσας τοπικής εκτέλεσης. Η εργασία δεν πρέπει να συγχέεται με την παρεμφερή επισκόπηση των Kertusha et al. (2025), η οποία αναλύει 214 άρθρα της περιόδου 2014–2024 με έμφαση στον ρόλο της τεχνητής νοημοσύνης· και οι δύο, ωστόσο, συγκλίνουν στο συμπέρασμα ότι το Selenium παραμένει το κυρίαρχο εργαλείο δοκιμών, ενώ η βιομηχανική υιοθέτηση των ερευνητικών προσεγγίσεων παραμένει συγκριτικά περιορισμένη.

Ως προς τα ερευνητικά ερωτήματα, η εργασία συνεισφέρει στη RQ5, μέσα από την εκτενέστερη καταγραφή εργαλείων δοκιμής ολόκληρου του συνόλου· στη RQ3, καθώς σχεδόν το ένα τρίτο των μελετών που συνθέτει αφορά την ανίχνευση ευπαθειών, προσφέροντας ποσοτική τεκμηρίωση για το βάρος της ασφάλειας στο στάδιο της δοκιμής· και στη RQ4, μέσω της καταγραφής των περιβαλλόντων δοκιμών στο νέφος. Η συνεισφορά της στη RQ1 είναι μερική αλλά ουσιαστική: η εργασία αναγνωρίζει τις τεχνικές δοκιμής και τις ομαδοποιεί κατά βήμα της διαδικασίας - τη δημιουργία και εκτέλεση δοκιμαστικών περιπτώσεων και την αξιολόγησή τους - διακρίνοντας παράλληλα τις κλασικές από τις αναδυόμενες, βασισμένες στη μηχανική μάθηση, προσεγγίσεις. Ο περιορισμός της δεν αφορά, συνεπώς, την αναγνώριση και την ομαδοποίηση των μεθόδων αλλά την κάλυψη των σταδίων, αφού εστιάζει σε ένα μόνο στάδιο του κύκλου ζωής.

4.2.6 Frameworks, Γλώσσες & Ασφάλεια

Deng et al. (2024) - Cloud-Native Computing: A Survey From the Perspective of Services

Η εργασία των Deng et al. (2024) προσφέρει μια ολοκληρωμένη επισκόπηση του cloud-native υπολογισμού από τη σκοπιά των υπηρεσιών. Παρότι ο τίτλος της δεν αναφέρεται ρητά στις εφαρμογές ιστού, η συνάφειά της με το αντικείμενο της παρούσας επισκόπησης είναι άμεση: οι συγγραφείς χαρακτηρίζουν το cloud-native υπόδειγμα ως την πλέον επιδραστική αρχή ανάπτυξης

για τις σύγχρονες εφαρμογές ιστού, και η ανάλυσή τους στηρίζεται στην αρχιτεκτονική των microservices, την οποία παρουσιάζουν ως την πλέον διαδεδομένη εξέλιξη της αρχιτεκτονικής προσανατολισμένης σε υπηρεσίες. Το γεγονός αυτό συνεπάγεται και μια αναγνωρισμένη επικάλυψη με την κατηγορία των microservices της Ενότητας 4.2.4· η εργασία εντάσσεται εδώ επειδή το αντικείμενό της δεν είναι η αρχιτεκτονική καθαυτή, αλλά η τεχνολογική στοίβα και τα εργαλεία που την υποστηρίζουν σε όλο το εύρος του κύκλου ζωής.

Οι συγγραφείς αποδομούν τον κύκλο ζωής των cloud-native εφαρμογών σε τέσσερις καταστάσεις - κατασκευή (building), ενορχήστρωση (orchestration), λειτουργία (operate) και συντήρηση (maintenance) - και χαρτογραφούν για καθεμία τους ερευνητικούς τομείς, τις θεμελιώδεις αναγκαιότητες και τους δείκτες απόδοσης. Η επισκόπηση καλύπτει τεχνολογίες, όπως το Docker, το Kubernetes, τα service meshes και το serverless, και αποτελεί την ολιστικότερη εργασία ολόκληρου του συνόλου ως προς την κάλυψη των σταδίων της υλοποίησης, της διάταξης και της συντήρησης.

Το πλεονέκτημα της εργασίας είναι η εξαιρετική κάλυψη του νέφους και των σχετικών εργαλείων. Οι περιορισμοί της είναι ότι δεν εστιάζει αποκλειστικά στις εφαρμογές ιστού και ότι η φάση της σχεδίασης καλύπτεται με λιγότερη λεπτομέρεια σε σχέση με εκείνες της υλοποίησης και της λειτουργίας. Πρόκειται, και εδώ, για περιορισμούς της εμβέλειας της συγκεκριμένης επισκόπησης και όχι για διαπίστωση ότι οι cloud-native προσεγγίσεις αγνοούν τη σχεδίαση.

Ως προς τη θέση της στην παρούσα επισκόπηση, η εργασία είναι ενδεικτική μιας ευρύτερης τάσης: οι πληρέστερες σύγχρονες επισκοπήσεις προσεγγίζουν τον κύκλο ζωής όχι από τη σκοπιά της εφαρμογής ιστού καθαυτής, αλλά από τη σκοπιά της υποκείμενης cloud-native υποδομής. Η ίδια η επισκόπηση αναγνωρίζει ρητά την απουσία ενός σαφούς ερευνητικού οδικού χάρτη για το cloud-native πεδίο ως το βασικό κενό που έρχεται να καλύψει - διαπίστωση που αφορά τα στάδια στα οποία η ίδια εστιάζει. Η παρούσα εργασία προεκτείνει την παρατήρηση αυτή: αν ακόμη και στα στάδια της υλοποίησης, της διάταξης και της λειτουργίας, όπου η cloud-native βιβλιογραφία είναι πλουσιότερη, απουσιάζει ένας ενοποιημένος οδικός χάρτης, τότε η ολιστική κάλυψη ολόκληρου του κύκλου ζωής για εφαρμογές ιστού - συμπεριλαμβανομένων των σταδίων της ανάλυσης απαιτήσεων και της σχεδίασης - παραμένει κατά μείζονα λόγο ανοιχτό ζήτημα. Ως προς τα ερευνητικά ερωτήματα, η εργασία συνεισφέρει καθοριστικά στη RQ4, αποτελώντας την πληρέστερη πηγή του συνόλου για τη σχέση των μεθόδων με τα περιβάλλοντα νέφους· στη RQ5, μέσα από την συστηματική χαρτογράφηση εργαλείων ανά κατάσταση του κύκλου ζωής· και στη RQ1, καθώς αναγνωρίζει τις πρακτικές και τα υποδείγματα που συγκροτούν την cloud-native προσέγγιση - τη δοχειοποίηση, την ενορχήστρωση, τα service meshes και το serverless ως εναλλακτικό μοντέλο εκτέλεσης - τεκμηριώνοντας παράλληλα κάλυψη τεσσάρων διακριτών σταδίων. Η συνεισφορά της στη RQ6 είναι έμμεση αλλά σημαντική, καθώς το κενό που η ίδια δηλώνει συνομιλεί άμεσα με τα ερευνητικά κενά του Κεφαλαίου 5.

Alazmi & De Leon (2022) - A Systematic Literature Review on Web Application Vulnerability Scanners

Η εργασία των Alazmi και De Leon (2022) είναι μια συστηματική ανασκόπηση των χαρακτηριστικών και της αποτελεσματικότητας των σαρωτών ευπαθειών εφαρμογών ιστού. Αναλύει ενενήντα μελέτες, με ιδιαίτερη έμφαση σε ευπάθειες όπως το SQL Injection και το Cross-Site Scripting, και αξιολογεί κατά πόσο τα υπάρχοντα εργαλεία τις εντοπίζουν αποτελεσματικά.

Το πλεονέκτημα της εργασίας είναι η αυστηρή ανασκόπηση με σαφή σύνδεση προς τις κατηγορίες του OWASP, γεγονός που την καθιστά ιδιαίτερα χρήσιμη για τη διάσταση της ασφάλειας. Ο κύριος περιορισμός της είναι ότι η πρωταρχική της εστίαση παραμένει το στάδιο των δοκιμών ασφαλείας· τα στάδια της διάταξης και της συντήρησης καλύπτονται κυρίως

έμμεσα, μέσα από την ένταξη των σαρωτών σε αγωγούς συνεχούς ενσωμάτωσης και τη συνεχή λειτουργία τους, ενώ η σχεδίαση και η υλοποίηση της ίδιας της εφαρμογής καλύπτονται μόνο περιφερειακά, υπό το πρίσμα της ασφάλειας.

Τα ποσοτικά ευρήματα της εργασίας είναι ιδιαίτερα αποκαλυπτικά για την κατάσταση της αξιολόγησης της ασφάλειας. Από τα ενενήντα άρθρα που εξετάστηκαν, οι συγγραφείς εντόπισαν τριάντα σαρωτές ευπαθειών, εκ των οποίων μόλις δώδεκα διέθεταν έστω μία ποσοτική αποτίμηση αποτελεσματικότητας, τεκμηριωμένη συνολικά από δεκαπέντε πρωτότυπες μελέτες αξιολόγησης. Το πλέον ανησυχητικό εύρημα είναι ότι οι μελέτες αυτές εξέτασαν κατά κανόνα μόλις δύο από τις δέκα κατηγορίες του OWASP Top Ten - το SQL Injection και το Cross-Site Scripting - ενώ μία μόνο εργασία κάλυψε έξι κατηγορίες, και μάλιστα για έναν μόνο σαρωτή. Επιπλέον, τα αναφερόμενα ποσοστά ανίχνευσης ήταν εξαιρετικά ανομοιογενή μεταξύ των αξιολογήσεων, γεγονός που καθιστά αδύνατη μια αντικειμενική σύγκριση. Οι συγγραφείς καταλήγουν σε μια σαφή σύσταση: την ανάγκη για τυποποιημένες εφαρμογές αναφοράς (benchmark applications) που θα καλύπτουν ολόκληρο το OWASP Top Ten, ώστε η αποτελεσματικότητα των εργαλείων ασφαλείας να γίνει επιτέλους συγκρίσιμη και αναπαραγώγιμη.

Ως προς τη σύνδεση με τα υπόλοιπα στάδια του κύκλου ζωής, η εργασία τοποθετεί τους σαρωτές στο πλαίσιο της αυτοματοποιημένης αποτίμησης ασφάλειας, αναγνωρίζοντας ότι η πρακτική τους αξία εξαρτάται από την ένταξή τους στον αγωγό συνεχούς ενσωμάτωσης: ένας σαρωτής που εκτελείται χειροκίνητα και σποραδικά δεν προστατεύει μια εφαρμογή που διατίθεται πολλές φορές την εβδομάδα. Η διάσταση αυτή, ωστόσο, δεν αναλύεται συστηματικά - δεν εξετάζεται ποια εργαλεία προσφέρουν ενσωμάτωση σε αγωγούς CI/CD, ούτε πώς η ενσωμάτωση αυτή επηρεάζει τα στάδια της υλοποίησης και της διάταξης. Το κενό αυτό είναι χαρακτηριστικό της συνολικής εικόνας που αναδεικνύει η παρούσα επισκόπηση, όπου η ασφάλεια αντιμετωπίζεται ως αυτοτελής δραστηριότητα δοκιμής και όχι ως εγκάρσια διάσταση του κύκλου ζωής.

Ως προς τα ερευνητικά ερωτήματα, η εργασία συνεισφέρει καθοριστικά στη RQ3, αποτελώντας την πληρέστερη πηγή του συνόλου για την αντιμετώπιση της ασφάλειας, και μάλιστα με ποσοτική τεκμηρίωση της ανεπάρκειας των υφιστάμενων εργαλείων· στη RQ5, μέσω της καταγραφής τριάντα σαρωτών· και στη RQ7, καθώς αποτιμά ρητά τον βαθμό εμπειρικής τεκμηρίωσης κάθε εργαλείου. Η σύσταση για τυποποιημένες εφαρμογές αναφοράς, τέλος, τροφοδοτεί άμεσα την ανάλυση των ερευνητικών κενών του Κεφαλαίου 5.

4.3 Κριτήρια Σύγκρισης ανά Στάδιο SDLC (RQ2)

Τα κριτήρια σύγκρισης αναπτύχθηκαν με βάση τρεις άξονες. Ο πρώτος είναι τα σχήματα σύγκρισης που χρησιμοποιούν σχετικές επισκοπήσεις: το σχήμα χαρακτηρισμού των Torrecilla-Salinas et al. (2016), ο κατάλογος κριτηρίων των Schwinger et al. (2008) για τις μεθόδους MDWE, η συγκριτική μεθοδολογία των Linaje et al. (2019) και η ταξινομία πρακτικών και εργαλείων των Shahin et al. (2017) για το CI/CD. Ο δεύτερος είναι τα χαρακτηριστικά ποιότητας του προτύπου ISO/IEC 25010, από το οποίο αντλούνται ιδίως οι διαστάσεις της ασφάλειας, της συντηρησιμότητας και της προσαρμοστικότητας. Ο τρίτος είναι οι ανάγκες που αναδύθηκαν από τις ερευνητικές ερωτήσεις της Ενότητας 3.1.

Τρεις από τις πηγές αυτές ανήκουν στο τελικό σύνολο των αναλυόμενων άρθρων, ενώ οι Schwinger et al. (2008) αποτελούν υποστηρικτική αναφορά εκτός συνόλου. Η μερική επικάλυψη είναι αναμενόμενη, καθώς οι αντιπροσωπευτικότερες επισκοπήσεις ενός πεδίου διαθέτουν κατά κανόνα και τα πιο επεξεργασμένα σχήματα σύγκρισης· αντιμετωπίστηκε με το να μην αντληθεί κανένα κριτήριο αποκλειστικά από μία πηγή και με έλεγχο ότι η τελική διατύπωση δεν ευνοεί συστηματικά την οικογένεια μεθόδων στην οποία ανήκει η πηγή του.

Ορίστηκαν είκοσι κριτήρια, τέσσερα ανά στάδιο. Μολονότι η ανάλυση απαιτήσεων και η σχεδίαση αντιμετωπίζονται ως ένα στάδιο, για τους λόγους της Ενότητας 2.3.1, τα κριτήρια τις διακρίνουν στο εσωτερικό του: το C2 αποτιμά αποκλειστικά τη μηχανική απαιτήσεων, ενώ τα C1 και C3 την αρχιτεκτονική και τη σχεδίαση διεπαφής.

Πίνακας 4.2: Τα είκοσι κριτήρια σύγκρισης ανά στάδιο SDLC. Πηγές: Schwinger et al. (2008), Torrecilla-Salinas et al. (2016), Linaje et al. (2019), Shahin et al. (2017) και το πρότυπο ISO/IEC 25010, σε συνδυασμό με τις ερευνητικές ερωτήσεις της Ενότητας 3.1.

Στάδιο	**Κριτήριο**	**Σημασιολογία**
Σχεδίαση	C1 Αρχιτεκτονικά πρότυπα	Καθοδήγηση για την επιλογή και εφαρμογή αρχιτεκτονικών προτύπων
	C2 Μοντελοποίηση απαιτήσεων	Υποστήριξη της εκμαίευσης και τυπικής αποτύπωσης λειτουργικών και μη λειτουργικών απαιτήσεων
	C3 Σχεδίαση διεπαφής και εμπειρίας χρήστη	Υποστήριξη της σχεδίασης του επιπέδου παρουσίασης, που στις εφαρμογές ιστού αποτελεί διακριτή σχεδιαστική πρόκληση· η σχεδίαση του back-end αποτιμάται από το C1
	C4 Εργαλειακή υποστήριξη σχεδίασης	Διαθεσιμότητα εργαλείων ειδικά για τις δραστηριότητες σχεδίασης
Υλοποίηση	C5 Frameworks και γλώσσες	Καθοδήγηση για την επιλογή τεχνολογικής στοίβας και τα αντίστοιχα εργαλεία
	C6 Αυτοματοποιημένη παραγωγή κώδικα	Παραγωγή εκτελέσιμου κώδικα από μοντέλα ή πρότυπα, χωρίς χειρωνακτική συγγραφή
	C7 Επαναχρησιμοποιησιμότητα	Δομημένη επαναχρησιμοποίηση συνιστωσών ή υπηρεσιών
	C8 Cloud-native χαρακτήρας και διαλειτουργικότητα APIs	Καταλληλότητα για εκτέλεση σε νέφος και ανεξαρτησία από συγκεκριμένο πάροχο (cross-cloud φορητότητα) μέσω τυποποιημένων διεπαφών
Δοκιμή	C9 Εύρος τύπων δοκιμών	Πλήθος και ποικιλία των καλυπτόμενων τύπων (μονάδας, ολοκλήρωσης, άκρο σε άκρο, απόδοσης, χρηστικότητας)
	C10 Αυτοματοποίηση δοκιμών	Αυτόματη εκτέλεση των δοκιμών και τα σχετικά εργαλεία
	C11 Κάλυψη κώδικα	Αποτίμηση ή διασφάλιση του βαθμού κάλυψης του κώδικα από τις δοκιμές
	C12 Δοκιμές ασφαλείας	Η ασφάλεια ως διακριτή δραστηριότητα δοκιμής, με τεχνικές ανίχνευσης ευπαθειών
Διάταξη	C13 Αυτοματοποιημένη διάταξη σε νέφος	Αυτοματοποιημένη εγκατάσταση και ενεργοποίηση της εφαρμογής σε

		υποδομή νέφους
	C14 Ενσωμάτωση σε αγωγούς CI/CD	Ένταξη της προσέγγισης σε αγωγό συνεχούς ενσωμάτωσης και παράδοσης
	C15 Δοχειοποίηση και ενορχήστρωση	Τεχνολογικά μέσα συσκευασίας και διαχείρισης των εκτελέσιμων μονάδων κατά τη διάταξη
	C16 Στρατηγικές διάταξης	Στρατηγικές μείωσης κινδύνου κατά τη διάταξη (blue-green, canary, rolling)
Συντήρηση	C17 Παρατηρησιμότητα	Κατανόηση της κατάστασης του συστήματος μέσω αρχείων καταγραφής, μετρικών και ιχνηλάτησης
	C18 Διαχείριση ενημερώσεων	Συστηματική διάθεση διορθώσεων, ενημερώσεων ασφαλείας και νέων εκδόσεων στην παραγωγή
	C19 Τεκμηρίωση	Η τεκμηρίωση ως παραδοτέο που υποστηρίζει τη συντήρηση
	C20 Προσαρμοστικότητα	Ικανότητα απορρόφησης μεταβολών απαιτήσεων και τεχνολογιών μετά τη διάταξη

Ορισμένα κριτήρια έχουν κατ' ανάγκη διασταδιακό χαρακτήρα, καταχωρίζονται όμως στο στάδιο όπου εκδηλώνεται πρωτίστως η συνεισφορά τους, ώστε να αποφευχθεί η διπλή προσμέτρηση. Έτσι, το C4 αποτιμά αποκλειστικά τα εργαλεία σχεδίασης, ενώ τα εργαλεία της υλοποίησης, της δοκιμής και της διάταξης καλύπτονται από τα C5, C10 και C15, αντιστοίχως. Για το στάδιο της συντήρησης δεν ορίζεται χωριστό κριτήριο εργαλείων: η εργαλειακή του διάσταση αποτιμάται μέσα από το C17, η σημασιολογία του οποίου παραπέμπει ρητά στα αρχεία καταγραφής, στις μετρικές και στην ιχνηλάτηση, δηλαδή στα μέσα με τα οποία υλοποιείται η παρατηρησιμότητα· η αναλυτική καταγραφή των εργαλείων συντήρησης γίνεται στην Ενότητα 4.5.5. Αντίστοιχα, η σχέση των C8 και C15 είναι συμπληρωματική και όχι επικαλυπτόμενη: το C8 αποτιμά τον cloud-native χαρακτήρα σε επίπεδο σχεδίασης και υλοποίησης, το C15 τα συγκεκριμένα τεχνολογικά μέσα κατά τη διάταξη. Ανάλογη διάκριση ισχύει και για το ζεύγος C18 και C20: το C18 αποτιμά τον μηχανισμό παράδοσης της αλλαγής, δηλαδή τον τρόπο με τον οποίο διορθώσεις, ενημερώσεις ασφαλείας και νέες εκδόσεις φτάνουν συστηματικά στην παραγωγή, ενώ το C20 αποτιμά την ικανότητα της προσέγγισης να απορροφά μεταβολές απαιτήσεων και τεχνολογιών. Η μεταβολή των απαιτήσεων, συνεπώς, αποτιμάται στο C20 και όχι στο C18.

Κάθε κριτήριο βαθμολογείται σε κλίμακα τεσσάρων τιμών, με ενιαίο κανόνα: δύο μονάδες για πλήρη κάλυψη, όταν η προσέγγιση προσφέρει ρητό μηχανισμό, τεχνική ή εργαλείο για το κριτήριο· μία μονάδα για ουσιαστική κάλυψη, όταν το πραγματεύεται συστηματικά χωρίς να προσδιορίζει συγκεκριμένα μέσα· μισή μονάδα για μερική ή έμμεση κάλυψη, όταν το αναφέρει ακροθιγώς ή ως προϋπόθεση· και μηδέν για απουσία.

Το αντικείμενο της αποτίμησης χρειάζεται ρητή διευκρίνιση. Η βαθμολόγηση δεν αφορά την ποιότητα των άρθρων καθαυτών - αυτή αποτιμήθηκε χωριστά, με τα κριτήρια της Ενότητας 3.3.2 - αλλά τον βαθμό στον οποίο η οικογένεια μεθόδων που πραγματεύεται κάθε άρθρο καλύπτει το αντίστοιχο κριτήριο. Το άρθρο λειτουργεί ως τεκμήριο: η βαθμολογία προκύπτει από τη γνώση που αυτό μεταφέρει για τις μεθόδους που αναλύει, όχι από τη δική του συγγραφική αρτιότητα. Για τον λόγο αυτό, μονάδα ανάλυσης των αποτελεσμάτων είναι η οικογένεια μεθόδων:

ο Πίνακας 4.3 παρουσιάζει τα επιμέρους σκορ ως ενδιάμεσο, ελέγξιμο βήμα, ενώ η συγκριτική ερμηνεία της Ενότητας 4.5 στηρίζεται στα συγκεντρωτικά σκορ ανά οικογένεια του Πίνακα 4.4.

Ο τρόπος παραγωγής των συγκεντρωτικών τιμών είναι ο ακόλουθος. Κάθε κριτήριο βαθμολογείται στην κλίμακα 0–2, οπότε το άθροισμα των τεσσάρων κριτηρίων ενός σταδίου αποδίδει βαθμολογία σταδίου με μέγιστο τις οκτώ μονάδες, και το άθροισμα των πέντε σταδίων αποδίδει συνολική βαθμολογία άρθρου με μέγιστο τις σαράντα μονάδες. Επειδή κάθε οικογένεια εκπροσωπείται από δύο άρθρα, η βαθμολογία της οικογένειας προκύπτει αθροιστικά, με μέγιστο τις ογδόντα μονάδες. Η επιλογή του αθροίσματος αντί του μέσου όρου δεν επηρεάζει τη σύγκριση: καθώς ο αριθμός των άρθρων είναι σταθερός και ίσος με δύο για όλες τις οικογένειες, το άθροισμα αποτελεί σταθερό πολλαπλάσιο του μέσου όρου και η σχετική κατάταξη παραμένει η ίδια. Για να είναι, πάντως, οι τιμές συγκρίσιμες μεταξύ σταδίων και οικογενειών ανεξάρτητα από την κλίμακα, οι βαθμολογίες παρατίθενται στην Ενότητα 4.5 και ως ποσοστά επί του αντίστοιχου μεγίστου, παράλληλα με τις απόλυτες μονάδες.

Σημειώνεται, τέλος, ότι ο Πίνακας 4.3 παρουσιάζει τις βαθμολογίες συγκεντρωμένες ανά στάδιο και όχι ανά επιμέρους κριτήριο, για λόγους αναγνωσιμότητας. Η ανάγνωση σε επίπεδο κριτηρίου δεν παραλείπεται, ωστόσο: η Ενότητα 4.5 εντοπίζει τα κριτήρια που εμφανίζουν συστηματικά χαμηλή κάλυψη σε όλες τις οικογένειες - με χαρακτηριστικότερο το C12 για τις δοκιμές ασφαλείας, το οποίο στην Ενότητα 4.5.3 αποδεικνύεται το κριτήριο με τη χαμηλότερη βαθμολογία ολόκληρου του πίνακα - ακριβώς επειδή τα σημεία αυτά υποδεικνύουν στοχευμένες δυνατότητες βελτίωσης που δεν θα ήταν ορατές από τα συγκεντρωτικά αθροίσματα.

4.4 Αποτελέσματα της Σύγκρισης (RQ2–RQ5)

Τα είκοσι κριτήρια εφαρμόστηκαν στα δώδεκα αντιπροσωπευτικά άρθρα. Ο Πίνακας 4.3 παρουσιάζει το σκορ κάθε άρθρου ανά στάδιο, καθώς και το συνολικό του σκορ στα σαράντα, ομαδοποιημένα κατά την οικογένεια μεθόδων στην οποία ανήκει το άρθρο.

Πίνακας 4.3: Σκορ των δώδεκα αντιπροσωπευτικών άρθρων ανά στάδιο SDLC (μέγιστο 8 ανά στάδιο, 40 συνολικά). Πηγή: υπολογισμός του συγγραφέα βάσει των είκοσι κριτηρίων.

Οικογένεια Μεθόδων	**Άρθρο**	**Σχεδ.**	**Υλοπ.**	**Δοκ.**	**Deploy**	**Συντ.**	**Σύνολο/40**
MDWE	Rossi et al. (MDWE)	7,5	4,0	0,5	0,0	4,5	16,5
MDWE	Linaje et al. (MDWE)	7,5	3,5	0,0	0,0	0,5	11,5
Agile & Scrum	Torrecilla-S. (Agile)	4,0	3,0	4,5	2,0	2,0	15,5
Agile & Scrum	Pastrana (Agile)	2,5	3,0	4,0	5,0	5,0	19,5

DevOps & CI/CD	Shahin et al. (DevOps)	2,0	4,0	6,0	8,0	4,0	24,0
DevOps & CI/CD	El Aouni (DevOps)	2,0	4,0	6,0	7,0	4,5	23,5
Microservices	Velepucha (Micro)	4,0	6,5	2,0	7,0	4,0	23,5
Microservices	Zhou et al. (Micro)	3,0	6,0	2,5	8,0	5,0	24,5
TDD & Testing	Rafique (TDD)	0,5	2,5	6,5	1,0	0,5	11,0
TDD & Testing	Li et al. (Testing)	0,5	2,5	7,0	2,0	2,5	14,5
Frameworks & Ασφάλεια	Deng et al. (Frameworks)	3,0	5,0	3,0	8,0	3,5	22,5
Frameworks & Ασφάλεια	Alazmi (Ασφάλεια)	4,0	5,0	5,0	7,0	5,5	26,5

Συγκεντρώνοντας τα σκορ των δύο άρθρων κάθε οικογένειας μεθόδων, προκύπτει ο Πίνακας 4.4, όπου η μονάδα αποτίμησης είναι πλέον η ίδια η οικογένεια μεθόδων και όχι το μεμονωμένο άρθρο. Κάθε στάδιο βαθμολογείται με μέγιστο τις δεκαέξι μονάδες και κάθε οικογένεια με μέγιστο τις ογδόντα.

Πίνακας 4.4: Πίνακας 4.4: Σκορ ανά οικογένεια μεθόδων και στάδιο SDLC (άθροισμα ζεύγους άρθρων· μέγιστο 16 ανά στάδιο, 80 ανά οικογένεια). Πηγή: υπολογισμός του συγγραφέα.

Οικογένεια	**Σχεδ.**	**Υλοπ.**	**Δοκ.**	**Διάταξη**	**Συντ.**	**Σύνολο/80**
MDWE	15,0	7,5	0,5	0,0	5,0	28,0
Agile/Scrum	6,5	6,0	8,5	7,0	7,0	35,0
DevOps & CI/CD	4,0	8,0	12,0	15,0	8,5	47,5

Microservices	7,0	12,5	4,5	15,0	9,0	48,0
TDD & Testing	1,0	5,0	13,5	3,0	3,0	25,5
Frameworks & Ασφάλεια	7,0	10,0	8,0	15,0	9,0	49,0
Μέση τιμή	*6,8*	*8,2*	*7,8*	*9,2*	*6,9*	*38,8*

4.5 Ανάλυση των Αποτελεσμάτων (RQ2–RQ8)

Πριν από την παρουσίαση των ευρημάτων είναι αναγκαία μια διευκρίνιση για το τι αποτυπώνουν τα σκορ των Πινάκων 4.3 και 4.4 και, κυρίως, για το τι δεν αποτυπώνουν. Η βαθμολόγηση προκύπτει από τη γνώση που μεταφέρουν δύο αντιπροσωπευτικές εργασίες ανά οικογένεια μεθόδων. Τα σκορ, συνεπώς, αποτιμούν τον βαθμό στον οποίο η βιβλιογραφία κάθε οικογένειας τεκμηριώνει την κάλυψη κάθε σταδίου, και όχι το θεωρητικό δυναμικό της οικογένειας καθαυτό. Μια οικογένεια ενδέχεται να καλύπτει ένα στάδιο καλύτερα από όσο υποδηλώνει η βαθμολογία της, εφόσον η κάλυψη αυτή δεν αποτυπώνεται στις επισκοπήσεις που τη πραγματεύονται· η μέτρηση αποτελεί, δηλαδή, ένδειξη δεύτερης τάξεως.

Δύο μεθοδολογικές επιλογές χρειάζονται ρητή δήλωση. Πρώτον, η βαθμολογία κάθε οικογένειας σε ένα στάδιο προκύπτει ως άθροισμα των δύο αντιπροσωπευτικών εργασιών, ακολουθώντας δηλαδή τη λογική της μέσης και όχι της βέλτιστης επίδοσης. Η επιλογή είναι συνειδητή: όταν ένα στάδιο καλύπτεται από μία μόνο προσέγγιση μιας οικογένειας, ενώ οι υπόλοιπες το αγνοούν, δεν μπορεί να θεωρηθεί ότι η οικογένεια το καλύπτει ως σύνολο. Η αποτίμηση κατά βέλτιστη επίδοση θα έδινε συστηματικά υψηλότερα σκορ και θα συσκότιζε τις ανομοιογένειες που η ανάλυση επιδιώκει να αναδείξει· όπου η διάκριση έχει ουσιώδεις συνέπειες, σημειώνεται στις επιμέρους υποενότητες. Δεύτερον, το γεγονός ότι δύο εργασίες ανά οικογένεια αποτελούν περιορισμένη βάση για γενικεύσεις καταγράφεται στις απειλές εγκυρότητας της Ενότητας 3.8· η επέκταση σε περισσότερες, συμπληρωματικές μεταξύ τους εργασίες ανά οικογένεια αποτελεί προφανή κατεύθυνση επέκτασης της εργασίας.

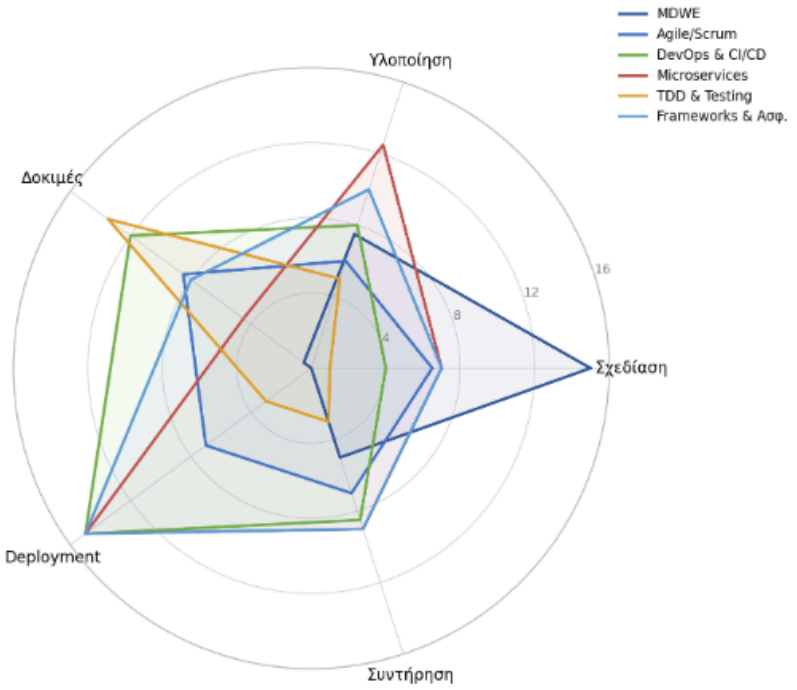
4.5.1 Γενική Εικόνα της Κάλυψης SDLC

Το πρώτο και κεντρικό εύρημα είναι ότι καμία οικογένεια μεθόδων δεν επιτυγχάνει πλήρη και ισόρροπη κάλυψη όλων των σταδίων. Όπως αποτυπώνεται στο Σχήμα 4.1, την υψηλότερη ολιστική κάλυψη σημειώνει η οικογένεια των Frameworks και της Ασφάλειας με 49,0 στα 80, ποσοστό που αντιστοιχεί σε 61,3% της μέγιστης δυνατής βαθμολογίας, ακολουθούμενη στενά από τα Microservices με 48,0 και το DevOps με 47,5. Στον αντίποδα, η MDWE, παρά την ιστορική της πρωτοκαθεδρία, περιορίζεται σε 28,0 στα 80, κυρίως λόγω της απουσίας της από το στάδιο της διάταξης. Το εύρος των σκορ εκτείνεται από 25,5, για την οικογένεια των δοκιμών, έως 49,0.

Το εύρημα χρειάζεται προσεκτική διατύπωση. Δεν σημαίνει ότι οι οικογένειες αγνοούν ολόκληρα στάδια: με εξαίρεση τη MDWE, η οποία σημειώνει μηδενική βαθμολογία στη διάταξη, όλες οι υπόλοιπες παρουσιάζουν μη μηδενική κάλυψη σε κάθε στάδιο. Το ζήτημα είναι η ομοιομορφία και το βάθος της κάλυψης και όχι η εμβέλειά της: το γεγονός ότι ούτε η κορυφαία

οικογένεια δεν ξεπερνά το 61,3% της μέγιστης βαθμολογίας σημαίνει ότι, ακόμη και στις πληρέστερες περιπτώσεις, περίπου τα δύο πέμπτα των κριτηρίων παραμένουν ακάλυπτα ή καλύπτονται μόνο έμμεσα. Το έλλειμμα αυτό, ωστόσο, δεν κατανέμεται ομοιόμορφα στα πέντε στάδια. Κάθε οικογένεια παρουσιάζει στάδια σχεδόν πλήρους κάλυψης και στάδια ελάχιστης: η οικογένεια των δοκιμών, για παράδειγμα, συγκεντρώνει 13,5 από τις 16 δυνατές μονάδες του σταδίου της δοκιμής, αλλά μόλις 1,0 στη σχεδίαση. Το συνολικό ποσοστό, συνεπώς, δεν περιγράφει μια διάχυτη αδυναμία αλλά το αποτέλεσμα μιας συστηματικής ανισοκατανομής, η οποία αναλύεται ανά στάδιο στην Ενότητα 4.5.2.

Η παρατήρηση αυτή υποδεικνύει και την κατεύθυνση της λύσης. Οι επιδόσεις των οικογενειών εμφανίζονται συμπληρωματικές: η MDWE σημειώνει 15,0 στη σχεδίαση έναντι 4,0 του DevOps, ενώ το DevOps σημειώνει 15,0 στη διάταξη έναντι μηδέν της MDWE. Η πλήρης κάλυψη του κύκλου ζωής δεν προϋποθέτει, συνεπώς, αναγκαστικά μια νέα καθολική οικογένεια μεθόδων, αλλά ενδέχεται να επιτευχθεί με τον τεκμηριωμένο συνδυασμό δύο ή περισσότερων υφιστάμενων, καθεμιάς στα στάδια όπου υπερέχει. Ο έλεγχος αυτής της συμπληρωματικότητας και η διερεύνηση των προϋποθέσεων υπό τις οποίες ο συνδυασμός είναι εφικτός αναπτύσσονται στο Κεφάλαιο 5.



Σχήμα 4.1: Ολιστική κάλυψη των σταδίων SDLC ανά κατηγορία (μέγιστο 16 ανά στάδιο).

4.5.2 Κάλυψη ανά Στάδιο και η Θεμελιώδης Πόλωση

Εξετάζοντας τα στάδια ξεχωριστά, αναδεικνύεται μια θεμελιώδης πόλωση. Η σχεδίαση καλύπτεται πληρέστερα από την MDWE, με 15,0 στα 16, ενώ η ίδια οικογένεια απουσιάζει εντελώς από τη διάταξη, με μηδενική βαθμολογία. Αντιστρόφως, το DevOps κυριαρχεί στη διάταξη με 15,0 στα 16, αλλά υπολείπεται σημαντικά στη σχεδίαση, με μόλις 4,0. Η πόλωση είναι εντονότερη στη διάταξη, όπου οι τιμές εκτείνονται από 0,0 έως 15,0, και ασθενέστερη στην υλοποίηση και τη συντήρηση, που αποτελούν τα δύο πλέον ομοιόμορφα καλυμμένα στάδια, με εύρη 5,0–12,5 και 3,0–9,0 αντίστοιχα. Η ομοιομορφία τους, ωστόσο, είναι ποιοτικά διαφορετική: η υλοποίηση καλύπτεται ομοιόμορφα σε υψηλό επίπεδο, με μέση τιμή 8,2 και με τα microservices να προηγούνται, ενώ η συντήρηση ομοιόμορφα σε χαμηλό, με μέση τιμή 6,9 και καμία οικογένεια να μην ξεπερνά το 9,0. Η ποιοτική αυτή διάκριση δεν στηρίζεται στη διαφορά των μέσων τιμών, η

οποία είναι μικρή - 1,3 μονάδες στις 16 - αλλά στη θέση του ανώτατου ορίου κάθε σταδίου. Στην υλοποίηση δύο οικογένειες φτάνουν ή ξεπερνούν τις δέκα από τις δεκαέξι μονάδες, με τα microservices να αγγίζουν τις 12,5, δηλαδή το 78% του μεγίστου, και να λειτουργούν ως τεκμηριωμένο σημείο αναφοράς για το τι είναι εφικτό· στη συντήρηση καμία οικογένεια δεν ξεπερνά τις 9,0, δηλαδή το 56%, και κανένα αντίστοιχο υπόδειγμα δεν υπάρχει. Στην πρώτη περίπτωση η ομοιομορφία σημαίνει κοινή επάρκεια, στη δεύτερη κοινή ανεπάρκεια.

Η συντήρηση αναδεικνύεται έτσι, μαζί με τη σχεδίαση, ως ένα από τα δύο πλέον παραμελημένα στάδια, με μέση τιμή 6,9 στα 16, οριακά πάνω από εκείνη της σχεδίασης (6,8). Οι δύο περιπτώσεις διαφέρουν ουσιωδώς: η χαμηλή τιμή της σχεδίασης προκύπτει από την πόλωση, ενώ εκείνη της συντήρησης από το ότι καμία οικογένεια δεν την καλύπτει ικανοποιητικά. Η πόλωση ανάμεσα στις προσεγγίσεις που εστιάζουν στη σχεδίαση και σε εκείνες που εστιάζουν στην υλοποίηση και τη λειτουργία εξηγεί γιατί, στην πράξη, οι εταιρείες υιοθετούν υβριδικές προσεγγίσεις.

4.5.3 Η Ασφάλεια ως Εγκάρσιο Ζήτημα (RQ3)

Η ασφάλεια, όπως αποτυπώνεται στο κριτήριο C12, είναι το κριτήριο με τη χαμηλότερη μέση βαθμολογία σε ολόκληρο τον πίνακα: μόλις 0,63 στις δύο μονάδες ανά άρθρο, δηλαδή συνολικό άθροισμα 7,5 στις 24 δυνατές μονάδες της γραμμής. Πλήρη κάλυψη παρέχει ένα μόνο άρθρο, εκείνο με τον σαφή προσανατολισμό στην ασφάλεια των εφαρμογών ιστού· ουσιαστική κάλυψη εμφανίζουν τρία, μερική πέντε, και τρία αγνοούν εντελώς το ζήτημα.

Η εικόνα αυτή αποκτά το πραγματικό της νόημα όταν ανατεθεί στις οικογένειες μεθόδων, που αποτελούν και τη μονάδα ανάλυσης της παρούσας επισκόπησης. Το Σχήμα 4.2 και ο Πίνακας 4.5 παρουσιάζουν τα σκορ ομαδοποιημένα κατά οικογένεια, ώστε να φαίνεται όχι μόνο ποιο άρθρο καλύπτει την ασφάλεια αλλά και πώς συμπεριφέρεται το ζεύγος που εκπροσωπεί κάθε οικογένεια.

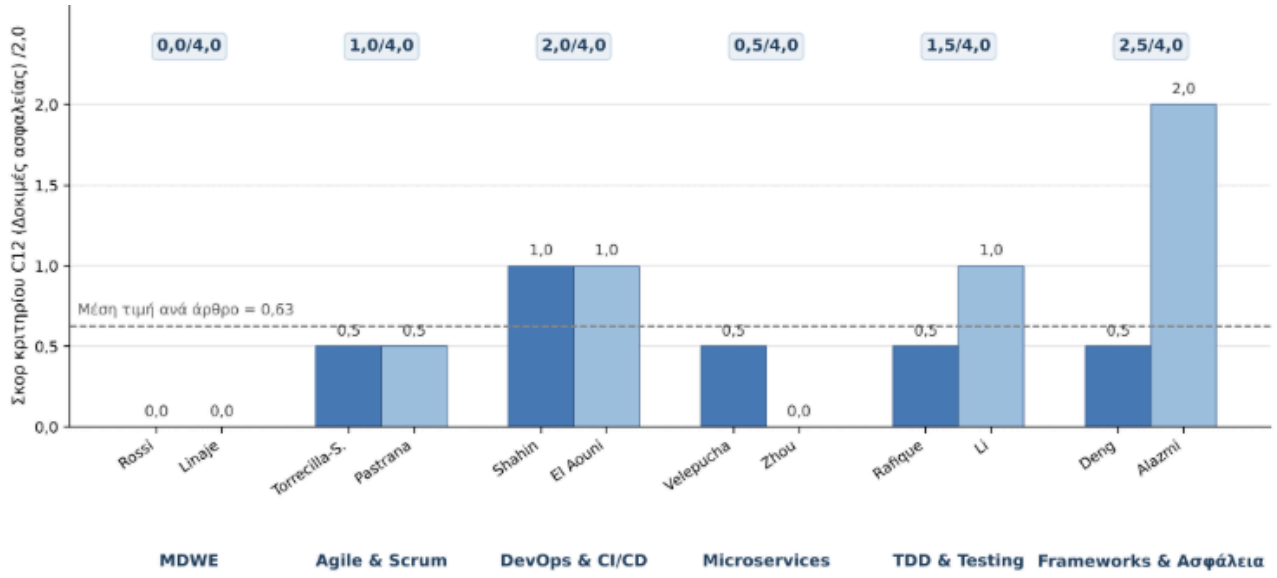
Πίνακας 4.5: Κάλυψη του κριτηρίου ασφάλειας (C12) ανά οικογένεια μεθόδων (άθροισμα ζεύγους άρθρων· μέγιστο 4,0 ανά οικογένεια). Πηγή: υπολογισμός του συγγραφέα.

Οικογένεια μεθόδων	**C12 / 4,0**	**Ποσοστό**
Frameworks, Γλώσσες & Ασφάλεια	2,5	62,5%
DevOps & CI/CD	2,0	50,0%
TDD & Automated Testing	1,5	37,5%
Agile & Scrum	1,0	25,0%
Microservices	0,5	12,5%
Model-Driven Web Engineering	0,0	0,0%
Σύνολο	*7,5 / 24,0*	*31,3%*

Η ανάγνωση σε επίπεδο οικογένειας αναδεικνύει δύο πράγματα που δεν φαίνονται στο επίπεδο του άρθρου. Πρώτον, καμία οικογένεια δεν ξεπερνά τα δύο τρίτα της δυνατής κάλυψης, και μόνο δύο φτάνουν ή υπερβαίνουν το ήμισυ· η ασφάλεια, δηλαδή, δεν αποτελεί τυποποιημένο συστατικό καμίας από τις εξεταζόμενες προσεγγίσεις. Δεύτερον, η κάλυψη δεν κατανέμεται τυχαία αλλά ακολουθεί τον προσανατολισμό κάθε οικογένειας: οι υψηλότερες τιμές εμφανίζονται στις τεχνολογικά και λειτουργικά προσανατολισμένες οικογένειες, όπου η ασφάλεια συνδέεται με τον αγωγό και τα εργαλεία, ενώ η μοντελοκεντρική οικογένεια σημειώνει μηδενική βαθμολογία, γεγονός που επιβεβαιώνει ότι η ασφάλεια δεν αντιμετωπίζεται ως σχεδιαστική απαίτηση.

Η ερμηνεία του ευρήματος απαιτεί προσοχή, καθώς η κατανομή αυτή αποτυπώνει πρωτίστως την εστίαση των επισκοπήσεων και όχι, κατ' ανάγκη, την πραγματική αντιμετώπιση της ασφάλειας από τις οικογένειες μεθόδων που αυτές πραγματεύονται. Το ζήτημα γίνεται σαφές στην περίπτωση του DevOps: το DevSecOps, με τη λογική του shift-left και την ενσωμάτωση των δοκιμών ασφαλείας στον αγωγό, αποτελεί τη μόνη από τις εξεταζόμενες προσεγγίσεις που αντιμετωπίζει την ασφάλεια ως διαρκή, εγκάρσια απαίτηση και όχι ως εκ των υστέρων έλεγχο. Εφόσον το DevSecOps συνιστά εξέλιξη του DevOps, η οικογένεια αυτή θα βαθμολογούνταν σημαντικά υψηλότερα στο C12 αν η αποτίμηση γινόταν κατά τη βέλτιστη επίδοση των μεθόδων της. Η παρούσα ανάλυση ακολουθεί, για τους λόγους που εκτέθηκαν στην εισαγωγή της Ενότητας 4.5, τη λογική της μέσης επίδοσης: το DevSecOps δεν αποτελεί ακόμη τυπικό συστατικό κάθε υλοποίησης DevOps, και οι δύο αντιπροσωπευτικές επισκοπήσεις της οικογένειας το πραγματεύονται χωρίς να το αναδεικνύουν σε κεντρικό άξονα της ανάλυσής τους. Η βαθμολογία δηλώνει, συνεπώς, ότι η ασφάλεια δεν αποτελεί τυποποιημένο συστατικό των υφιστάμενων προσεγγίσεων, όχι ότι δεν υπάρχουν προσεγγίσεις που την αντιμετωπίζουν.

Η διάκριση αυτή δεν αποδυναμώνει το εύρημα αλλά το εξειδικεύει: το πρόβλημα δεν είναι η απουσία κατάλληλων πρακτικών, αλλά η περιορισμένη ενσωμάτωσή τους στις καθιερωμένες μεθόδους ανάπτυξης και, κατ' επέκταση, στη βιβλιογραφία που τις πραγματεύεται. Παρότι το OWASP Top 10 (2021) παραμένει το βασικό σημείο αναφοράς για τις ευπάθειες των εφαρμογών ιστού, ελάχιστες από τις εξεταζόμενες προσεγγίσεις το αντιμετωπίζουν ως σχεδιαστική απαίτηση. Η συστηματική ενσωμάτωση της ασφάλειας σε όλα τα στάδια του κύκλου ζωής αποτελεί, για τον λόγο αυτό, κατεύθυνση που αναδεικνύεται ρητά στο Κεφάλαιο 5.



Σχήμα 4.2: Κάλυψη του κριτηρίου ασφάλειας (C12) ανά αντιπροσωπευτικό άρθρο, ομαδοποιημένα κατά οικογένεια μεθόδων.Πάνω από κάθε ζεύγος σημειώνεται το συνολικό σκορ της οικογένειας.

4.5.4 Σχέση με το Νέφος και τα Σύγχρονα Εργαλεία (RQ4, RQ8)

Η ανάλυση αποκαλύπτει μια σαφή χρονική πόλωση. Τα άρθρα που δημοσιεύτηκαν πριν το 2018 παρουσιάζουν σχεδόν μηδενική κάλυψη του νέφους και της δοχειοποίησης, ενώ εκείνα από το 2019 και έπειτα ενσωματώνουν εκτενώς τεχνολογίες, όπως το Docker, το Kubernetes, το serverless και το GitOps. Η συμβατότητα με το νέφος και τα API, όπως αποτυπώνεται στο κριτήριο C8, εμφανίζεται με πλήρη κάλυψη αποκλειστικά στις οικογένειες του DevOps, των Microservices και των Frameworks - σε πέντε από τις έξι εργασίες που τις εκπροσωπούν - ενώ απουσιάζει εντελώς από τις υπόλοιπες τρεις οικογένειες. Η εργασία των Deng et al. (2024) αποτελεί την ολιστικότερη καταγραφή της cloud-native στοίβας σε ολόκληρο το σύνολο.

Η RQ4, ωστόσο, δεν εξαντλείται στο C8. Όπως δηλώνεται στην Ενότητα 3.1, η σχέση των μεθόδων με τα περιβάλλοντα νέφους αποτιμάται και από τα κριτήρια C13 έως C15, τα οποία αφορούν την αυτοματοποιημένη διάταξη σε νέφος, την ένταξη σε αγωγούς CI/CD και τη δοχειοποίηση με ενορχήστρωση, και εντάσσονται στο στάδιο της διάταξης. Η στήλη της διάταξης του Πίνακα 4.4 επιβεβαιώνει και διευρύνει την εικόνα του C8: οι ίδιες τρεις οικογένειες - DevOps, Microservices και Frameworks - συγκεντρώνουν από 15,0 στις 16 δυνατές μονάδες, δηλαδή σχεδόν πλήρη κάλυψη, ενώ οι ευέλικτες μέθοδοι περιορίζονται στις 7,0, η οικογένεια των δοκιμών στις 3,0 και η MDWE σημειώνει μηδέν. Η σύγκλιση δύο ανεξάρτητων ομάδων κριτηρίων - του C8 σε επίπεδο σχεδίασης και υλοποίησης και των C13 έως C15 σε επίπεδο διάταξης - ενισχύει το εύρημα: η cloud-native διάσταση δεν είναι απλώς ισχυρότερη σε ορισμένες οικογένειες, αλλά απουσιάζει συστηματικά από εκείνες που εστιάζουν στα πρώιμα στάδια.

Η πόλωση αυτή χρειάζεται προσεκτική ερμηνεία. Αποτυπώνει τη χρονική μετατόπιση της εστίασης της βιβλιογραφίας και όχι, κατ' ανάγκη, τη χρονική στιγμή κατά την οποία οι οικογένειες μεθόδων άρχισαν να ενσωματώνουν τεχνολογίες νέφους. Το Docker κυκλοφόρησε ως λογισμικό ανοικτού κώδικα τον Μάρτιο του 2013, ενώ το Kubernetes έφτασε στην πρώτη του σταθερή έκδοση τον Ιούλιο του 2015· η απουσία τους από τις προγενέστερες επισκοπήσεις δεν συνεπάγεται, επομένως, απουσία τους από τη βιομηχανική πρακτική της ίδιας περιόδου, αλλά αντανακλά πιθανότερα μια υστέρηση της τάξης των τριών ετών, με την οποία η ερευνητική κοινότητα αποτύπωσε συστηματικά τις εξελίξεις αυτές. Η οριστική διάκριση ανάμεσα στις δύο ερμηνείες θα απαιτούσε χρονολογική μελέτη της υιοθέτησης, η οποία υπερβαίνει την εμβέλεια της παρούσας επισκόπησης και καταγράφεται ως κατεύθυνση περαιτέρω διερεύνησης.

Ανεξάρτητα από την ερμηνεία, το εύρημα παραμένει χρήσιμο για τη RQ4: η συμβατότητα με το νέφος εμφανίζεται πλήρης μόνο στις οικογένειες των οποίων το αντικείμενο συνδέεται εξ ορισμού με την υποδομή εκτέλεσης, ενώ απουσιάζει από εκείνες που εστιάζουν στα πρώιμα στάδια του κύκλου ζωής. Η παρατήρηση συνάδει με τη θεμελιώδη πόλωση της Ενότητας 4.5.2 και υποδεικνύει ότι η cloud-native διάσταση δεν έχει ακόμη ενσωματωθεί στις μεθόδους σχεδίασης.

4.5.5 Frameworks, Γλώσσες και Εργαλεία (RQ5)

Από την ανάλυση προκύπτει μια σαφής εικόνα της τεχνολογικής στοίβας ανά στάδιο. Σημειώνεται ότι η εικόνα συνθέτει δύο πηγές: τα εργαλεία που καταγράφουν ρητά τα άρθρα του τελικού συνόλου και εκείνα που αποτυπώθηκαν στην επισκόπηση του Κεφαλαίου 2, καθώς ορισμένες κατηγορίες - ιδίως τα πλαίσια ανάπτυξης της πλευράς του πελάτη - δεν αποτελούν αντικείμενο των αναλυόμενων εργασιών. Η αντιστοίχιση κάθε εργαλείου στην εργασία που το αναφέρει δίνεται στις επιμέρους αναλύσεις της Ενότητας 4.2, όπου καταγράφονται τα εργαλεία που εντοπίζει κάθε αντιπροσωπευτικό άρθρο· η παρούσα ενότητα τα συγκεντρώνει ανά στάδιο, ώστε να προκύψει ενιαία εικόνα της τεχνολογικής στοίβας. Στη σχεδίαση κυριαρχούν τα περιβάλλοντα μοντελοποίησης UML, που αποτελούν τη βάση των μεθόδων MDWE, καθώς και τα συνεργατικά

εργαλεία σχεδίασης διεπαφών, με χαρακτηριστικότερο το Figma - μια πλατφόρμα που εκτελείται στο πρόγραμμα περιήγησης και επιτρέπει σε σχεδιαστές και προγραμματιστές να δημιουργούν wireframes και διαδραστικά πρωτότυπα, συνεργαζόμενοι ταυτόχρονα στο ίδιο αρχείο. Στην υλοποίηση, τα React, Angular και Vue.js στην πλευρά του πελάτη, και τα Node.js, Django και Spring Boot στην πλευρά του διακομιστή, με τη JavaScript ως de facto κοινή βάση, ενώ η στατική ανάλυση ποιότητας στηρίζεται σε εργαλεία όπως το SonarQube. Στη δοκιμή, τα JUnit και NUnit για τις μοναδιαίες δοκιμές, τα Selenium και Cypress για τις δοκιμές από άκρο σε άκρο, το Selenium Grid για την κατανεμημένη εκτέλεσή τους, τα OWASP ZAP και Burp Suite για τις δοκιμές ασφαλείας, και τα JMeter και LoadRunner για τις δοκιμές απόδοσης και φόρτου. Στη διάταξη, τα Docker και Kubernetes για τη δοχειοποίηση και την ενορχήστρωση, το Helm για τη διαχείριση των πακέτων εγκατάστασης, τα Istio και Envoy για τη διαχείριση της επικοινωνίας μεταξύ υπηρεσιών, το Chef για τη διαχείριση διαμόρφωσης, και τα Jenkins, GitLab CI/CD, Bamboo και TeamCity ως μηχανές (CI/CD) αγωγών. Στη συντήρηση, τέλος, τα Prometheus και Grafana για τη συλλογή και απεικόνιση μετρικών και το Kibana για τη διαχείριση των αρχείων καταγραφής.

Η καταγραφή αυτή αναδεικνύει μια αξιοσημείωτη γενεαλογική υστέρηση ως προς τα εργαλεία της διάταξης και της συντήρησης. Οι εργασίες του συνόλου αναγνωρίζουν την αυτοματοποιημένη διαχείριση της υποδομής ως πρακτική και αναφέρουν εργαλεία της προηγούμενης γενιάς, όπως το Chef, χωρίς όμως να χρησιμοποιούν τον καθιερωμένο πλέον όρο «υποδομή ως κώδικας» ούτε να καταγράφουν τα σημερινά αντιπροσωπευτικά του εργαλεία, όπως το Terraform και το Ansible· αντίστοιχα, οι GitOps πρακτικές και τα εργαλεία τους, όπως το Argo CD και το Flux, απουσιάζουν εντελώς. Παράλληλα, η κατανεμημένη ιχνηλάτηση αναγνωρίζεται ως κρίσιμη απαίτηση - οι Zhou et al. (2023) την εντοπίζουν ρητά ως ένα από τα δύο μεγαλύτερα πρακτικά εμπόδια - χωρίς να συνοδεύεται από καταγραφή των αντίστοιχων εργαλείων. Η έλλειψη αυτή δεν οφείλεται σε ανωριμότητα της τεχνολογίας: το OpenTelemetry αποτελεί σήμερα το καθιερωμένο πρότυπο συλλογής ιχνών και μετρικών, με ευρέως διαδεδομένες υλοποιήσεις όπως το Jaeger και το Zipkin. Κανένα από τα εργαλεία αυτά δεν αναφέρεται στις εργασίες του τελικού συνόλου, γεγονός που επιβεβαιώνει ότι η απόκλιση εντοπίζεται στην ερευνητική αποτύπωση και όχι στη διαθεσιμότητα των μέσων. Η εικόνα, συνεπώς, δεν είναι απουσίας αλλά χρονικής υστέρησης: η ακαδημαϊκή βιβλιογραφία αποτυπώνει την εργαλειακή διάσταση της διάταξης και της λειτουργίας με καθυστέρηση μιας γενιάς σε σχέση με τη βιομηχανική πρακτική. Ανάλογη, αν και ηπιότερη, είναι η εικόνα στις μοναδιαίες δοκιμές, όπου εργαλεία όπως το JUnit αναφέρονται περιστασιακά ως συνοδευτικά, χωρίς καμία εργασία να χαρτογραφεί συστηματικά τα πλαίσια μοναδιαίων δοκιμών ανά τεχνολογική στοίβα. Η αντιστοίχιση των εργαλείων στις οικογένειες μεθόδων συνοψίζεται στον Πίνακα 4.6, ο οποίος παραθέτει τα εργαλεία που καταγράφει κάθε οικογένεια μεθόδων μέσα από τις αντιπροσωπευτικές της εργασίες. Ο πίνακας καλύπτει το εστιακό υποσύνολο των δώδεκα άρθρων, για τα οποία η καταγραφή των εργαλείων έγινε σε επίπεδο πλήρους κειμένου· για τα δεκαπέντε άρθρα του συνόλου επικύρωσης η εξαγωγή δεν περιέλαβε συστηματική καταγραφή εργαλείων, περιορισμός που αναγνωρίζεται ρητά.

Πίνακας 4.6: Εργαλεία που καταγράφονται ανά οικογένεια μεθόδων, με βάση τις αντιπροσωπευτικές εργασίες της Ενότητας 4.2.

Οικογένεια μεθόδων	**Εργαλεία που καταγράφονται**	**Στάδια**
Model-Driven Web Engineering	Καμία ρητή καταγραφή εργαλείων· αναφέρονται μόνο γλώσσες και μέθοδοι μοντελοποίησης (WebML, IFML)	-
Agile & Scrum	Καμία ρητή καταγραφή εργαλείων	-

DevOps & CI/CD	JUnit, NUnit· Chef και εργαλεία διαχείρισης διαμόρφωσης	Δοκιμή, Διάταξη
Microservices	Δοχεία και πλατφόρμες ενορχήστρωσης, χωρίς ονομαστική καταγραφή	Διάταξη
TDD & Automated Testing	Selenium· περιβάλλοντα δοκιμών σε νέφος (AWS, Google Cloud Platform, Microsoft Azure)	Δοκιμή
Frameworks, Γλώσσες & Ασφάλεια	Docker, Kubernetes, service meshes, serverless· τριάντα σαρωτές ευπαθειών, χωρίς ονομαστική παράθεση	Υλοποίηση, Διάταξη, Συντήρηση

Η εικόνα του πίνακα δεν είναι ουδέτερη: η εργαλειακή καταγραφή συγκεντρώνεται στις οικογένειες με λειτουργικό, αρχιτεκτονικό και τεχνολογικό προσανατολισμό, ενώ οι μοντελοκεντρικές και οι διαδικαστικές οικογένειες δεν καταγράφουν εργαλεία καθόλου - αναφέρονται μόνο σε γλώσσες, μεθόδους και πρακτικές. Η παρατήρηση αυτή ενισχύει τη θεμελιώδη πόλωση της Ενότητας 4.5.2 και από την πλευρά των εργαλείων, και εξηγεί γιατί η συγκεντρωτική εικόνα της στοίβας χρειάστηκε να συμπληρωθεί από την επισκόπηση του Κεφαλαίου 2 στα στάδια της σχεδίασης και της υλοποίησης.

4.5.6 Εμπειρική Αξιολόγηση (RQ7)

Η RQ7 δεν αντιστοιχεί σε κάποιο από τα είκοσι κριτήρια σύγκρισης, τα οποία αποτιμούν την κάλυψη των σταδίων του κύκλου ζωής και όχι τον βαθμό εμπειρικής τεκμηρίωσης. Η απάντησή της προκύπτει από το τρίτο κριτήριο της αξιολόγησης ποιότητας (Ενότητα 3.3.2), που αφορά ρητά την ύπαρξη εμπειρικής αξιολόγησης, σε συνδυασμό με την ποιοτική ανάλυση των αντιπροσωπευτικών εργασιών της Ενότητας 4.2.

Το ερώτημα διακρίνεται σε δύο επίπεδα. Το πρώτο αφορά τις ίδιες τις εργασίες του τελικού συνόλου: διενεργούν αυτές εμπειρική αξιολόγηση; Η απάντηση είναι καταφατική για έξι από τις είκοσι επτά εργασίες, δηλαδή για το 22,2% του συνόλου. Τέσσερις εργασίες του συνόλου στηρίζονται σε πρωτογενή εμπειρικά δεδομένα: οι Zhou et al. (2023) σε ποιοτικές συνεντεύξεις σε είκοσι εταιρείες, οι Linaje et al. (2019) σε έρευνα ερωτηματολογίου σε τριάντα τέσσερις εταιρείες, οι Begel και Nagappan (2007) σε βιομηχανική μελέτη, και οι Bogner και Merkel (2022) σε ανάλυση εξακοσίων τεσσάρων αποθετηρίων κώδικα. Δύο επιπλέον συνθέτουν ποσοτικά πρωτογενείς μελέτες: η μετα-ανάλυση των Rafique και Mišić (2013), επί είκοσι επτά ελεγχόμενων πειραμάτων, και η οικογένεια πειραμάτων των Santos et al. (2021). Από αυτές, οι Zhou et al. (2023), οι Linaje et al. (2019) και οι Rafique και Mišić (2013) ανήκουν στο εστιακό υποσύνολο των δώδεκα, ενώ οι υπόλοιπες στο σύνολο επικύρωσης. Οι λοιπές εργασίες αποτελούν συστηματικές ανασκοπήσεις ή μελέτες χαρτογράφησης, οι οποίες δεν παράγουν νέα εμπειρικά δεδομένα αλλά συνθέτουν υπάρχοντα.

Το δεύτερο επίπεδο αφορά το αν υπάρχουν πρωτογενείς μελέτες που τεκμηριώνουν εμπειρικά τις οικογένειες μεθόδων, ανεξάρτητα από το αν τις διενεργούν οι επισκοπήσεις του συνόλου. Η εικόνα εδώ είναι έντονα ανομοιογενής. Το TDD διαθέτει την πλουσιότερη εμπειρική βάση ολόκληρου του πεδίου, με δεκάδες ελεγχόμενα πειράματα που συνθέτονται σε μετα-αναλύσεις και επιτρέπουν ποσοτικά συμπεράσματα για την επίδραση στην ποιότητα και την παραγωγικότητα. Οι ευέλικτες μέθοδοι διαθέτουν επίσης σημαντική εμπειρική τεκμηρίωση, όπως δείχνει η ανασκόπηση των Dybå και Dingsøyr (2008) επί τριάντα έξι εμπειρικών μελετών. Για τα microservices και το DevOps, η τεκμηρίωση είναι κυρίως ποιοτική και προέρχεται από

βιομηχανικές μελέτες και συνεντεύξεις, χωρίς ελεγχόμενες συγκρίσεις. Για τις μεθόδους MDWE, τέλος, η εμπειρική βάση είναι η ασθενέστερη: οι Rossi et al. (2016) διαπιστώνουν ρητά την απουσία αυστηρής εμπειρικής τεκμηρίωσης των οφελών τους σε πραγματικά περιβάλλοντα.

Αξιοσημείωτο είναι ότι, παρά την ύπαρξη πρωτογενών μελετών ανά τεχνική, κανένα από τα άρθρα του συνόλου - ούτε κάποια από τις πρωτογενείς μελέτες που αυτά αναφέρουν - δεν παρέχει πλήρως ελεγχόμενο, τυχαιοποιημένο πείραμα για μια ολόκληρη μέθοδο ανάπτυξης εφαρμογών ιστού. Τα υπάρχοντα πειράματα αφορούν μεμονωμένες τεχνικές, κατά κανόνα σε επίπεδο σταδίου, και όχι τη συνολική επίδραση μιας οικογένειας μεθόδων στον κύκλο ζωής. Ο περιορισμός αυτός είναι γνωστός και εν μέρει αναπόφευκτος, καθώς ένα τέτοιο πείραμα θα απαιτούσε παράλληλη ανάπτυξη του ίδιου συστήματος από ισοδύναμες ομάδες με διαφορετικές μεθόδους, σε κλίμακα και διάρκεια που σπανίως είναι εφικτές. Ενδεικτικό της απαιτούμενης προσπάθειας είναι ότι η αποτίμηση μιας και μόνο τεχνικής, του TDD, χρειάστηκε συντονισμένη οικογένεια πειραμάτων σε πολλαπλές ερευνητικές ομάδες και τοποθεσίες (Santos et al., 2021). Αναγνωρίζεται, ωστόσο, ως ερευνητικό κενό στο Κεφάλαιο 5.

4.5.7 Χρονική Εξέλιξη των Μεθόδων (RQ8)

Πριν από την παρουσίαση της εξέλιξης, χρειάζεται μια διευκρίνιση ως προς το αντικείμενό της. Η ανάλυση που ακολουθεί αποτυπώνει τη μετατόπιση της ερευνητικής εστίασης, δηλαδή το ποια οικογένεια μεθόδων συγκέντρωνε το κύριο ενδιαφέρον της βιβλιογραφίας σε κάθε περίοδο, και όχι τη χρονολογία της βιομηχανικής υιοθέτησης. Η διάκριση είναι ουσιώδης, καθώς, όπως διαπιστώθηκε στην Ενότητα 4.5.4, η βιβλιογραφία αποτυπώνει τις τεχνολογικές εξελίξεις με υστέρηση περίπου μιας γενιάς· η παράλληλη χαρτογράφηση της βιομηχανικής υιοθέτησης θα απαιτούσε διαφορετικό τύπο δεδομένων, όπως οι ετήσιες έρευνες του κλάδου, και υπερβαίνει την εμβέλεια της παρούσας επισκόπησης.

Με βάση τη χρονική κατανομή του Σχήματος 3.2 και το αντικείμενο των εργασιών κάθε περιόδου, αναδεικνύεται μια εξέλιξη τεσσάρων γενεών ερευνητικής εστίασης. Η πρώτη, έως περίπου το 2010, κυριαρχείται από τις μεθόδους MDWE και εστιάζει στη σχεδίαση και τη μοντελοποίηση. Η δεύτερη, έως το 2015, φέρνει την άνοδο των ευέλικτων μεθόδων για τον ιστό και την καθιέρωση της συνεχούς ενσωμάτωσης. Η τρίτη, έως το 2020, σηματοδοτεί τη μετάβαση στο DevOps και τα microservices, με το Docker και το Kubernetes να καθιερώνονται ως de facto πρότυπα. Η τέταρτη, τέλος, έως το 2025, χαρακτηρίζεται από τις cloud-native προσεγγίσεις, το GitOps, την AI-ενισχυμένη ανάπτυξη και την αρχή του security-by-design.

Σημειώνεται ότι οι γενεές αυτές αναφέρονται στην περίοδο κατά την οποία κάθε οικογένεια μεθόδων αναδείχθηκε ως κύριο αντικείμενο έρευνας και όχι στο έτος δημοσίευσης των αντίστοιχων άρθρων του συνόλου· οι επισκοπήσεις των μεθόδων MDWE, για παράδειγμα, δημοσιεύονται από το 2015 και έπειτα, αποτιμώντας αναδρομικά ένα ρεύμα που αναπτύχθηκε την προηγούμενη δεκαετία. Η διαδοχή των γενεών, εξάλλου, δεν συνεπάγεται αντικατάσταση: καμία οικογένεια δεν εγκαταλείπεται, αλλά η ερευνητική προσοχή μετατοπίζεται, αφήνοντας τις παλαιότερες σε καθεστώς σταδιακής ωρίμανσης, με αισθητά μειωμένο ρυθμό νέων συνεισφορών - εικόνα που αποτυπώνεται ξεκάθαρα στην περιορισμένη πρόσφατη παρουσία της MDWE.

4.5.8 Σύνοψη: Τέσσερα Κεντρικά Ευρήματα

Η ανάλυση αποδίδει τέσσερα κεντρικά ευρήματα. Το πρώτο είναι η έλλειψη ολιστικής κάλυψης: καμία οικογένεια μεθόδων δεν ξεπερνά το 61,3% της μέγιστης δυνατής βαθμολογίας, με την κατηγορία των Frameworks και το DevOps να πλησιάζουν περισσότερο χωρίς να το επιτυγχάνουν.

Το δεύτερο είναι η πόλωση ανάμεσα στη σχεδίαση και τη διάταξη, με την MDWE να κυριαρχεί στη σχεδίαση αλλά να απουσιάζει από τη διάταξη, και το DevOps να ακολουθεί την αντίστροφη πορεία.

Το τρίτο είναι η περιορισμένη κάλυψη της ασφάλειας, που αποτελεί το λιγότερο καλυμμένο κριτήριο ολόκληρου του πίνακα. Το εύρημα αυτό, όπως αναλύθηκε στην Ενότητα 4.5.3, δεν σημαίνει ότι απουσιάζουν κατάλληλες προσεγγίσεις: το DevSecOps, το οποίο αποτελεί εξέλιξη του DevOps και όχι αυτοτελή οικογένεια, αντιμετωπίζει την ασφάλεια ως εγκάρσια απαίτηση μέσα από τη λογική του shift-left. Η οικογένεια του DevOps διαθέτει, συνεπώς, τα μέσα για την κάλυψη του κριτηρίου· η χαμηλή βαθμολογία αποτυπώνει το ότι τα μέσα αυτά δεν αποτελούν ακόμη τυποποιημένο συστατικό κάθε υλοποίησης DevOps, ούτε αντιμετωπίζονται ως τέτοιο από τη βιβλιογραφία που την πραγματεύεται.

Τέλος, το τέταρτο εύρημα είναι το χάσμα ανάμεσα στη θεωρία και την πράξη, το οποίο τεκμηριώνεται με ασφάλεια από την πλευρά της MDWE: παρά τη μεγαλύτερη ακαδημαϊκή παραγωγή του πεδίου, η οικογένεια αυτή υιοθετείται ελάχιστα στη βιομηχανία, όπως επιβεβαιώνουν εμπειρικά οι Linaje et al. (2019), με το 59% των εταιρειών του δείγματός τους να δηλώνει χρήση υβριδικών, εσωτερικών προσεγγίσεων. Η αντίστροφη όψη του χάσματος - το κατά πόσο τεχνολογίες με ευρεία βιομηχανική διάδοση, όπως τα συστήματα διαχείρισης περιεχομένου και οι πλατφόρμες low-code, υποεκπροσωπούνται στην έρευνα - δεν μπορεί να τεκμηριωθεί με βεβαιότητα από το παρόν σύνολο, καθώς η περιορισμένη τους παρουσία σε αυτό ενδέχεται να οφείλεται στα μοτίβα επερώτησης της Ενότητας 3.2.2 και όχι σε πραγματικό ερευνητικό κενό· διατυπώνεται, για τον λόγο αυτό, ως υπόθεση που απαιτεί στοχευμένη διερεύνηση.

Εγκάρσια προς τα τέσσερα αυτά ευρήματα διατρέχει μια ακόμη διαπίστωση, που αναδείχθηκε στην Ενότητα 4.5.6: καμία εργασία του συνόλου, ούτε κάποια από τις πρωτογενείς μελέτες που αυτές αναφέρουν, δεν αποτιμά πειραματικά μια ολόκληρη οικογένεια μεθόδων - η εμπειρική τεκμηρίωση αφορά μεμονωμένες τεχνικές σε επίπεδο σταδίου. Η παρατήρηση αυτή οριοθετεί τη βεβαιότητα με την οποία μπορούν να συγκριθούν οι οικογένειες μεταξύ τους. Τα ευρήματα αυτά τροφοδοτούν άμεσα τα ερευνητικά κενά του Κεφαλαίου 5.

5. Ερευνητικά Κενά και Μελλοντικές Κατευθύνσεις

Το παρόν κεφάλαιο παρουσιάζει τα ερευνητικά κενά που αναδύθηκαν από τη μελέτη της βιβλιογραφίας και από τη συγκριτική ανάλυση του Κεφαλαίου 4, απαντώντας άμεσα στην έκτη ερευνητική ερώτηση. Η ταυτοποίηση των κενών δεν προέκυψε αυθαίρετα, αλλά μέσα από μια συστηματική διαδικασία: για κάθε κατηγορία μεθόδων και για κάθε στάδιο του κύκλου ζωής, εντοπίστηκαν οι περιοχές όπου είτε η ανάλυση κατέγραψε χαμηλή βαθμολογία κάλυψης, είτε η ίδια η βιβλιογραφία επισημαίνει ρητά ανοιχτά προβλήματα. Για κάθε κενό παρέχεται πρώτα η τεκμηρίωσή του και στη συνέχεια, σε ξεχωριστή υποενότητα, ένα σύνολο υποσχόμενων ερευνητικών κατευθύνσεων για την αντιμετώπισή του. Τα κενά δεν είναι ανεξάρτητα μεταξύ τους· αντιθέτως, αλληλοσυνδέονται, και η αντιμετώπιση του ενός συχνά διευκολύνει και την αντιμετώπιση των υπολοίπων.

5.1 RG1 - Έλλειψη Ολιστικών Οικογενειών Μεθόδων που Καλύπτουν το Σύνολο του SDLC

Το πρώτο και κεντρικότερο κενό είναι η απουσία οικογενειών μεθόδων που να καλύπτουν ολόκληρο τον κύκλο ζωής της ανάπτυξης με ισόρροπο τρόπο. Η τεκμηρίωσή του είναι διπλή, αντλούμενη τόσο από τη βιβλιογραφία όσο και από τα ποσοτικά δεδομένα της ανάλυσης. Από την πλευρά της βιβλιογραφίας, οι Rossi et al. (2016) επισημαίνουν ότι ακόμη και οι ώριμες μέθοδοι MDWE, παρά την εικοσιπενταετή τους εξέλιξη, αποτυγχάνουν να καλύψουν επαρκώς τα μεταγενέστερα στάδια του κύκλου ζωής, ιδίως τη διάταξη και τη συντήρηση. Παράλληλα, οι Linaje et al. (2019) επιβεβαιώνουν εμπειρικά, μέσα από τη βιομηχανική τους έρευνα, ότι οι εταιρείες καταφεύγουν σε υβριδικές προσεγγίσεις ακριβώς επειδή καμία ενιαία οικογένεια μεθόδων δεν επαρκεί για το σύνολο των αναγκών τους.

Από την πλευρά των δεδομένων, η ανάλυση του Κεφαλαίου 4 ποσοτικοποιεί με σαφήνεια το κενό. Η βαθμολογία ολιστικής κάλυψης ανά οικογένεια κυμαίνεται μεταξύ 25,5 και 49,0 στις ογδόντα δυνατές μονάδες, χωρίς καμία να προσεγγίζει το ανώτατο όριο. Ακόμη και η κορυφαία, εκείνη των Frameworks και της Ασφάλειας, φτάνει μόλις στο 61,3% της μέγιστης δυνατής κάλυψης.

Η ρίζα του προβλήματος είναι ιστορική: κάθε οικογένεια μεθόδων αναπτύχθηκε για να αντιμετωπίσει το πρόβλημα ενός περιορισμένου υποσυνόλου σταδίων και διατηρεί έκτοτε το κέντρο βάρους της εκεί. Η εξειδίκευση αυτή δεν σημαίνει, πάντως, μονοσταδιακή εστίαση. Οι μέθοδοι MDWE εστιάζουν στη σχεδίαση, αλλά ο τελικός τους στόχος είναι η υλοποίηση: η μοντελοποίηση αποτελεί το μέσο μέσα από το οποίο παράγεται αυτόματα ο εκτελέσιμος κώδικας, και η οικογένεια καλύπτει συνεπώς δύο στάδια και όχι ένα - εξ ου και η βαθμολογία 7,5 στις δεκαέξι μονάδες της υλοποίησης, έναντι μηδενικής στη διάταξη. Αντίστοιχα, το DevOps δεν περιορίζεται στη διάταξη: οι αγωγοί συνεχούς ενσωμάτωσης και παράδοσης καλύπτουν εξίσου την αυτοματοποιημένη δοκιμή και την επικύρωση του λογισμικού, ενώ η λειτουργική διάσταση επεκτείνει την κάλυψη και στη συντήρηση· τα σκορ της οικογένειας το επιβεβαιώνουν, με 12,0 στη δοκιμή και 8,5 στη συντήρηση πέραν του 15,0 της διάταξης. Η πόλωση που εντοπίστηκε στην ανάλυση αφορά, συνεπώς, την κατανομή του βάρους σε δύο ή τρία στάδια και όχι τον αποκλεισμό όλων των υπολοίπων· αυτό που λείπει σε κάθε περίπτωση είναι η ισόρροπη κάλυψη του συνόλου.

Στο κενό αυτό συμβάλλει και ένας βαθύτερος παράγοντας, που αναδείχθηκε ήδη στο υπόβαθρο της εργασίας: η ετερογένεια των απαιτήσεων ανά πεδίο εφαρμογής. Όπως τεκμηρίωσαν εμπειρικά οι Eckhardt, Vogelsang και Méndez Fernández (2016), η κατανομή των ποιοτικών χαρακτηριστικών διαφέρει σημαντικά ανάλογα με τον τύπο του συστήματος, γεγονός που δυσχεραίνει τη διαμόρφωση μιας ενιαίας, καθολικά εφαρμόσιμης οικογένειας μεθόδων. Μια εφαρμογή ηλεκτρονικής τραπεζικής, μια πλατφόρμα υγείας και ένα ηλεκτρονικό κατάστημα

θέτουν ριζικά διαφορετικές προτεραιότητες ως προς την ασφάλεια, τη διαθεσιμότητα και την επεκτασιμότητα - και επομένως καμία μοναδική μέθοδος δεν μπορεί να είναι ταυτόχρονα βέλτιστη για όλα τα δυνατά πεδία εφαρμογής. Η διαπίστωση αυτή δεν αναιρεί την ανάγκη για ολιστικές προσεγγίσεις, αλλά υποδεικνύει ότι αυτές οφείλουν να είναι προσαρμόσιμες (tailorable) και όχι άκαμπτες, ώστε να μπορούν να διαμορφώνονται ανάλογα με το εκάστοτε πλαίσιο.

5.1.1 Κατευθύνσεις Αντιμετώπισης

Η αντιμετώπιση του κενού αυτού απαιτεί μια στροφή από την εξειδίκευση προς την ολοκλήρωση, με αντικείμενο τη μέθοδο ανάπτυξης καθαυτή και όχι κάποιο επιμέρους τεχνολογικό εργαλείο.

Μια πρώτη υποσχόμενη κατεύθυνση είναι η σύνθεση υβριδικών μεθόδων ανάπτυξης. Η λογική δεν είναι η επινόηση μιας νέας μεθόδου από μηδενική βάση, αλλά η επιλογή συγκεκριμένων πρακτικών και τεχνικών από διαφορετικές οικογένειες και η οργανική τους ενσωμάτωση σε μια ενιαία μέθοδο που καλύπτει το σύνολο των σταδίων: η μοντελοκεντρική σχεδίαση και η αυτόματη παραγωγή κώδικα από την MDWE για τα πρώιμα στάδια, οι ευέλικτες πρακτικές για την οργάνωση της υλοποίησης, και οι πρακτικές DevOps για τη δοκιμή, τη διάταξη και τη λειτουργία - καθεμία αξιοποιούμενη εκεί όπου η ανάλυση του Κεφαλαίου 4 έδειξε ότι υπερέχει. Το κρίσιμο ερευνητικό ερώτημα δεν είναι αν τα συστατικά αυτά είναι χρήσιμα, αλλά υπό ποιες προϋποθέσεις είναι συμβατά μεταξύ τους και πώς τεκμηριώνεται η συμβατότητά τους.

Μια δεύτερη κατεύθυνση, προαπαιτούμενη της πρώτης, είναι η συστηματική αντιστοίχιση των σταδίων του κύκλου ζωής σε τεχνικές και εργαλεία. Το ζητούμενο είναι μια τεκμηριωμένη χαρτογράφηση που, για κάθε στάδιο, καταγράφει τις διαθέσιμες τεχνικές, τα εργαλεία που τις υλοποιούν και τις προϋποθέσεις εφαρμογής τους. Μια τέτοια χαρτογράφηση δεν συνιστά, όπως εύλογα θα μπορούσε να αντιταχθεί, ενιαία αρχιτεκτονική που φιλοδοξεί να φιλοξενήσει κάθε δυνατό εργαλείο - κάτι τέτοιο θα ήταν ανέφικτο, καθώς τα εργαλεία κάθε σταδίου είναι συχνά ανταγωνιστικά ή αμοιβαία ασύμβατα. Αποτελεί, αντιθέτως, χάρτη επιλογών με ρητά κριτήρια: ποιες τεχνικές καλύπτουν ποιο στάδιο, ποιοι συνδυασμοί τους είναι συμβατοί και ποιοι όχι. Η αντιστοίχιση αυτή μπορεί στη συνέχεια να αξιοποιηθεί με τρεις τρόπους - για την κατασκευή νέας μεθόδου που καλύπτει όλα τα στάδια, για την επέκταση μιας υφιστάμενης ώστε να καλύψει τα στάδια που της λείπουν, ή για τον συνδυασμό δύο συμπληρωματικών μεθόδων όπου αυτό είναι εφικτό.

Τέλος, μια τρίτη κατεύθυνση είναι η εμπειρική αξιολόγηση των μεθόδων που θα προκύψουν, σε πραγματικά έργα και όχι σε ελεγχόμενες ακαδημαϊκές συνθήκες. Η κατεύθυνση αυτή είναι κρίσιμη ώστε να τεκμηριωθεί η αποτελεσματικότητά τους πέρα από το θεωρητικό επίπεδο και να αποφευχθεί η επανάληψη του σφάλματος των μεθόδων που, παρά την ακαδημαϊκή τους ωριμότητα, παρέμειναν αναξιοποίητες στη βιομηχανία.

5.2 RG2 - Παραμέληση της Ασφάλειας ως Εγκάρσιου Ζητήματος

Το δεύτερο κενό αφορά την ανεπαρκή και αποσπασματική αντιμετώπιση της ασφάλειας. Η ανάλυση ανέδειξε το σχετικό κριτήριο ως το λιγότερο καλυμμένο ολόκληρου του πίνακα σύγκρισης, με μέση βαθμολογία μόλις 0,58 στις δύο μονάδες ανά άρθρο. Η διαπίστωση αυτή δεν είναι μεμονωμένη, αλλά ενισχύεται από πολλαπλές πηγές της βιβλιογραφίας. Οι Alazmi και De Leon (2022) δείχνουν ότι, παρότι υπάρχει πληθώρα εξειδικευμένων εργαλείων σάρωσης ευπαθειών, αυτά εφαρμόζονται κατά κανόνα ως ξεχωριστό βήμα μετά την υλοποίηση, και όχι ως ενσωματωμένο μέρος της διαδικασίας ανάπτυξης. Οι Niemietz et al. (2021), από την πλευρά τους, αναδεικνύουν πόσο διαδεδομένες παραμένουν στην πράξη ευπάθειες όπως το Cross-Site Scripting και το SQL Injection, οι οποίες βρίσκονται σταθερά στην κορυφή του OWASP Top 10 (2021) εδώ και χρόνια.

Η βαθύτερη αιτία του κενού είναι πολιτισμική και ιστορική: η ασφάλεια αντιμετωπιζόταν παραδοσιακά ως μια αρμοδιότητα εξειδικευμένων ομάδων, ξεχωριστή από την ανάπτυξη, και όχι ως μια ευθύνη που διαπερνά κάθε στάδιο και κάθε ρόλο. Το παράδειγμα του DevSecOps επιχειρεί να αντιστρέψει αυτή τη λογική, η ενσωμάτωσή του όμως στις καθιερωμένες μεθόδους ανάπτυξης παραμένει περιορισμένη, όπως επισημαίνουν οι El Aouni et al. (2024), οι οποίοι καταγράφουν την πρόσθετη πολυπλοκότητα που εισάγει η ενσωμάτωση της ασφάλειας στους αγωγούς CI/CD.

Η αποσπασματικότητα αυτή επιβεβαιώνεται και από την ειδικότερη βιβλιογραφία της ασφάλειας εφαρμογών ιστού. Οι Deepa και Thilagam (2016), στην εκτενή ανασκόπησή τους για τις ευπάθειες injection και logic, καταλήγουν ότι δεν υπάρχει μια καθιερωμένη, τυποποιημένη τεχνική για την ανάπτυξη ασφαλών εφαρμογών ιστού, αλλά μάλλον ένα σύνολο ετερογενών, επιμέρους λύσεων που εφαρμόζονται εκ των υστέρων. Το εύρημα αυτό συγκλίνει απόλυτα με τη διαπίστωση της παρούσας ανάλυσης: η ασφάλεια αντιμετωπίζεται ως τεχνικό ζήτημα ανίχνευσης και διόρθωσης, και όχι ως αρχιτεκτονική αρχή που καθοδηγεί τον σχεδιασμό από την αρχή. Ελλείψει μιας μεθόδου που να ενσωματώνει την ασφάλεια σε κάθε στάδιο, οι οργανισμοί εξαρτώνται από τη διακριτική ευχέρεια και την εξειδίκευση μεμονωμένων ομάδων, με αποτέλεσμα η κάλυψη να παραμένει ανομοιογενής και, όπως δείχνει η επιμονή των ίδιων ευπαθειών στην κορυφή του OWASP Top 10, συχνά ανεπαρκής.

5.2.1 Κατευθύνσεις Αντιμετώπισης

Η θεραπεία του κενού περνά μέσα από την καθιέρωση της αρχής security-by-design. Μια πρώτη κατεύθυνση είναι η ανάπτυξη μεθοδολογικών κατευθυντήριων γραμμών που θα ενσωματώνουν τους ελέγχους του OWASP σε κάθε στάδιο, από τη μοντελοποίηση απαιτήσεων ασφαλείας στη σχεδίαση έως τη συνεχή παρακολούθηση στη συντήρηση. Μια δεύτερη είναι η περαιτέρω αυτοματοποίηση των δοκιμών ασφαλείας μέσα στον αγωγό συνεχούς ενσωμάτωσης, ακολουθώντας και επεκτείνοντας το παράδειγμα του DevSecOps με στατική και δυναμική ανάλυση κώδικα. Η κατεύθυνση αυτή προϋποθέτει και τη δοχειοποίηση, η οποία αφενός διευκολύνει τη δυναμική ανάλυση προσφέροντας απομονωμένα, αναπαραγώγιμα περιβάλλοντα εκτέλεσης, αφετέρου εισάγει νέες κατηγορίες ευπαθειών - στις εικόνες των δοχείων, στις διαμορφώσεις της ενορχήστρωσης και στην αλυσίδα εφοδιασμού των εξαρτήσεων - που απαιτούν πρόσθετους, εξειδικευμένους ελέγχους εντός του αγωγού. Τέλος, μια τρίτη, αναδυόμενη κατεύθυνση είναι η διερεύνηση της αξιοποίησης των μεγάλων γλωσσικών μοντέλων - όχι για τη συγγραφή κώδικα δοκιμών, καθώς η ανάλυση ασφαλείας διενεργείται μέσω εξειδικευμένων εργαλείων, αλλά για την παραγωγή δοκιμαστικών εισόδων προσαρμοσμένων στη λογική της εφαρμογής, την υποβοήθηση στη διαμόρφωση των αγωγών και των κανόνων των εργαλείων, και τη διαλογή των ευρημάτων από τα ψευδώς θετικά αποτελέσματα.

5.3 RG3 - Χάσμα Θεωρίας-Πράξης: Έλλειψη Κοινών Βιομηχανικών Πρακτικών στην Έρευνα

Το τρίτο κενό αφορά την αναντιστοιχία ανάμεσα στο τι μελετά η ακαδημαϊκή κοινότητα και στο τι χρησιμοποιεί η βιομηχανία. Η τεκμηρίωσή του προέρχεται από την εμπειρική παρατήρηση των Linaje et al. (2019): το 59% των εταιρειών του δείγματός τους χρησιμοποιεί υβριδικές, εσωτερικά διαμορφωμένες προσεγγίσεις αντί για τις καθιερωμένες ακαδημαϊκές μεθόδους. Το χάσμα έχει δύο διακριτές κατευθύνσεις. Η πρώτη είναι η δυσχερής μεταφορά των ερευνητικών αποτελεσμάτων στην πράξη, όπως διαπιστώνουν οι Di Francesco, Lago και Malavolta (2019) μέσα από τη συστηματική χαρτογράφηση 103 μελετών για τα microservices. Η δεύτερη, αντίστροφη και αυτή που συνιστά το προκείμενο κενό, είναι η περιορισμένη αποτύπωση των καθιερωμένων βιομηχανικών πρακτικών στην ερευνητική ατζέντα.

Η ανάλυση παρέχει συγκεκριμένα παραδείγματα της δεύτερης κατεύθυνσης. Όπως διαπιστώθηκε στην Ενότητα 4.5.5, καμία εργασία του συνόλου δεν καταγράφει τα εργαλεία της υποδομής ως κώδικα, όπως το Terraform και το Ansible, ούτε τις GitOps πρακτικές, παρότι αποτελούν τυπικά συστατικά ενός σύγχρονου αγωγού· η βιβλιογραφία αναφέρεται σε εργαλεία διαχείρισης διαμόρφωσης της προηγούμενης γενιάς. Αντίστοιχα, η κατανεμημένη ιχνηλάτηση αναγνωρίζεται από τους Zhou et al. (2023) ως κρίσιμο εμπόδιο, χωρίς καταγραφή των εργαλείων που την υλοποιούν. Στην ίδια κατηγορία πιθανολογείται ότι εντάσσονται και τα συστήματα διαχείρισης περιεχομένου και οι πλατφόρμες low-code, η διαπίστωση όμως αυτή δεν μπορεί να τεκμηριωθεί από το παρόν σύνολο και διατυπώνεται ως υπόθεση που απαιτεί στοχευμένη διερεύνηση.

Τα αίτια δεν προκύπτουν από την ανάλυση και διατυπώνονται ως υποθέσεις. Μια πρώτη εξήγηση είναι τα κίνητρα της ακαδημαϊκής έρευνας, η οποία αποτιμάται με βάση τη θεωρητική πρωτοτυπία και θεωρεί τη μελέτη ευρέως διαδεδομένων αλλά τεχνικά συμβατικών εργαλείων λιγότερο ελκυστική· μια δεύτερη είναι η περιορισμένη ορατότητα των βιομηχανικών πρακτικών, καθώς οι εσωτερικές διαδικασίες των εταιρειών σπανίως δημοσιοποιούνται. Το αποτέλεσμα, σε κάθε περίπτωση, είναι ότι οι μηχανικοί στερούνται τεκμηριωμένων κατευθύνσεων - όχι για τη χρήση των εργαλείων καθαυτών, που καλύπτεται από την τεχνική τους τεκμηρίωση, αλλά για την ένταξή τους σε μια συνεκτική μέθοδο ανάπτυξης. Η γεφύρωση του χάσματος απαιτεί, συνεπώς, έρευνα διαφορετικού προσανατολισμού, στραμμένη ρητά προς τα ερωτήματα που θέτει η βιομηχανική πραγματικότητα. Η στροφή αυτή, ωστόσο, καλύπτει μόνο τη μία κατεύθυνση. Το χάσμα είναι αμφίδρομο και η αντιμετώπισή του προϋποθέτει δύο ακόμη μηχανισμούς. Ο πρώτος αφορά την αντίστροφη ροή, δηλαδή την υιοθέτηση των ερευνητικών προτάσεων από τη βιομηχανία: η πιθανότητα υιοθέτησης αυξάνεται όταν οι προτάσεις συνοδεύονται από αξιολόγηση σε πραγματικά περιβάλλοντα και από αναφορές υλοποίησης, και όταν τα οφέλη τους διατυπώνονται σε μεγέθη που αποτιμά η βιομηχανία - χρόνος παράδοσης, κόστος συντήρησης, συχνότητα αστοχιών - και όχι μόνο σε όρους θεωρητικής πληρότητας. Ο δεύτερος αφορά την ορατότητα της ίδιας της βιομηχανικής πρακτικής, η οποία, όπως σημειώθηκε, σπανίως δημοσιοποιείται: η ανάδειξή της απαιτεί μεθόδους που δεν στηρίζονται στην επιστημονική βιβλιογραφία, όπως οι δομημένες συνεντεύξεις και οι έρευνες ερωτηματολογίου σε επαγγελματίες (Begel & Nagappan, 2007· Zhou et al., 2023), η συστηματική μελέτη της γκρίζας βιβλιογραφίας - τεχνικών αναφορών, τεκμηρίωσης εργαλείων και αναφορών εμπειρίας - και η εξόρυξη δημόσιων αποθετηρίων κώδικα και διαμορφώσεων αγωγών, κατά το πρότυπο των Bogner και Merkel (2022).

5.3.1 Κατευθύνσεις Αντιμετώπισης

Η γεφύρωση του χάσματος απαιτεί μια συνειδητή στροφή της έρευνας προς τη βιομηχανική πραγματικότητα. Μια πρώτη κατεύθυνση αφορά τις πρακτικές που η παρούσα ανάλυση εντόπισε ως υποεκπροσωπημένες: τη συστηματική μελέτη της υποδομής ως κώδικα και των GitOps πρακτικών, τόσο ως προς τα εργαλεία που τις υλοποιούν όσο και ως προς τη θέση τους μέσα σε μια συνεκτική μέθοδο ανάπτυξης, καθώς και την εργαλειακή διάσταση της παρατηρησιμότητας και της κατανεμημένης ιχνηλάτησης, η οποία αναγνωρίζεται ως κρίσιμη απαίτηση χωρίς να συνοδεύεται από αντίστοιχη ερευνητική κάλυψη.

Μια δεύτερη κατεύθυνση αφορά τις πλατφόρμες low-code και no-code. Η παρούσα επισκόπηση δεν τεκμηριώνει ούτε τον βαθμό της βιομηχανικής τους διάδοσης ούτε την έκταση της ερευνητικής τους κάλυψης, καθώς καμία εργασία του τελικού συνόλου δεν τις πραγματεύεται· η αποτίμηση και των δύο διαστάσεων συνιστά, ακριβώς για τον λόγο αυτό, το περιεχόμενο της προτεινόμενης κατεύθυνσης. Προαπαιτούμενο βήμα αποτελεί ο έλεγχος της πλέον πρόσφατης βιβλιογραφίας, ώστε να διαπιστωθεί αν η απουσία τους από το παρόν σύνολο αντανακλά

πραγματικό ερευνητικό κενό ή απλώς το χρονικό εύρος και τα μοτίβα επερώτησης της Ενότητας 3.2.2.

Τέλος, μια τρίτη κατεύθυνση είναι οι εθνογραφικές μελέτες των υβριδικών μεθόδων που πράγματι χρησιμοποιούνται στην πράξη, ώστε η θεωρία να αντλήσει από την πραγματικότητα αντί να την αγνοεί. Η κατεύθυνση αυτή απαντά άμεσα στο εύρημα των Linaje et al. (2019) για την επικράτηση εσωτερικά διαμορφωμένων προσεγγίσεων και αποτελεί, ενδεχομένως, την πλέον παραγωγική, καθώς δεν προϋποθέτει τη διατύπωση νέας μεθόδου αλλά την καταγραφή και συστηματοποίηση εκείνων που ήδη εφαρμόζονται.

5.4 RG4 - Περιορισμένη Ενσωμάτωση της Συντήρησης και της Παρατηρησιμότητας στις Μεθόδους Ανάπτυξης

Το τέταρτο κενό απαιτεί προσεκτική διατύπωση, ώστε να μην υπερβαίνει αυτό που τα δεδομένα τεκμηριώνουν. Η ανάλυση ανέδειξε τη συντήρηση ως ένα από τα δύο πλέον παραμελημένα στάδια, με μέση βαθμολογία 6,9 στις δεκαέξι μονάδες και με καμία από τις έξι οικογένειες που εξετάστηκαν να μην την έχει ως πρωταρχική εστίαση. Το εύρημα αυτό δεν σημαίνει ότι η συντήρηση αγνοείται από τη βιβλιογραφία της μηχανικής λογισμικού. Υπάρχουν, αντιθέτως, ολόκληρες οικογένειες μεθόδων αφιερωμένες ρητά σε αυτήν - μέθοδοι επανασχεδίασης και εκσυγχρονισμού λογισμικού, όπως εκείνες που χαρτογραφούν οι Abgaz et al. (2023) για τη μετάβαση από μονολιθικά συστήματα σε αρχιτεκτονικές microservices, καθώς και μέθοδοι συντήρησης και εξέλιξης, οι διεργασίες των οποίων τυποποιούνται στο πρότυπο ISO/IEC 14764 - οι οποίες βρίσκονται εκτός της εμβέλειας της παρούσας επισκόπησης, εστιασμένης στις μεθόδους ανάπτυξης εφαρμογών ιστού. Το εύρημα αφορά, συνεπώς, αποκλειστικά τις οικογένειες που εξετάστηκαν και δεν γενικεύεται στο σύνολο του πεδίου.

Το κενό που τεκμηριώνεται είναι ειδικότερο: η συντήρηση δεν είναι ενσωματωμένη στις μεθόδους ανάπτυξης εφαρμογών ιστού, αλλά αντιμετωπίζεται ως μεταγενέστερη, διακριτή δραστηριότητα, που ανατίθεται σε χωριστές μεθόδους και χωριστές ομάδες. Η ίδια διαπίστωση ισχύει και για την παρατηρησιμότητα, με μια σημαντική διαφοροποίηση. Η διάσταση αυτή καλύπτεται ουσιαστικά από το DevOps, από τη μηχανική αξιοπιστίας υπηρεσιών και από τη cloud-native προσέγγιση, όπως επιβεβαιώνουν και οι βαθμολογίες των αντίστοιχων οικογενειών στο στάδιο της συντήρησης, 8,5 και 9,0 στις δεκαέξι μονάδες, από τις υψηλότερες του πίνακα. Αυτό που απουσιάζει δεν είναι, επομένως, η κάλυψη της παρατηρησιμότητας αλλά η αντιμετώπισή της ως σχεδιαστικής απαίτησης από τα πρώιμα στάδια: οι Deng et al. (2024) χαρτογραφούν τις σχετικές τεχνολογίες, όπως το Prometheus, το Grafana και το Kibana, χωρίς να προσφέρουν μεθοδολογική καθοδήγηση για την ένταξή τους στη σχεδίαση, ενώ οι Zhou et al. (2023) διαπιστώνουν εμπειρικά ότι η παρατηρησιμότητα και η διαχείριση των κατανεμημένων δεδομένων παραμένουν τα μεγαλύτερα πρακτικά εμπόδια στην υιοθέτηση των microservices - ακριβώς επειδή αντιμετωπίζονται εκ των υστέρων.

Η συνέπεια είναι πρακτική: εφόσον οι εφαρμογές ιστού λειτουργούν συνεχώς και εξελίσσονται διαρκώς, το μεγαλύτερο μέρος του κόστους και της προσπάθειας του κύκλου ζωής τους δαπανάται μετά την αρχική διάταξη, σε ένα στάδιο που οι μέθοδοι ανάπτυξης δεν προβλέπουν συστηματικά.

5.4.1 Κατευθύνσεις Αντιμετώπισης

Οι κατευθύνσεις που ακολουθούν δεν αφορούν την εισαγωγή της συντήρησης ως νέας δραστηριότητας, η οποία υποστηρίζεται ήδη από υπάρχουσες μεθόδους και εργαλεία, αλλά τη βελτίωση της ενσωμάτωσής της στις μεθόδους ανάπτυξης - και ειδικότερα στο πλαίσιο των

κατανεμημένων εφαρμογών και της αρχιτεκτονικής microservices, όπου η ανάλυση ανέδειξε τα εντονότερα προβλήματα.

Μια πρώτη κατεύθυνση είναι ο εμπλουτισμός των μεθόδων ανάπτυξης, ώστε να ενσωματώνουν τους στόχους επιπέδου υπηρεσίας (Service Level Objectives, SLO) ήδη από τη φάση της σχεδίασης, αντιμετωπίζοντας την παρατηρησιμότητα ως απαίτηση και όχι ως εκ των υστέρων προσθήκη. Δεν πρόκειται για νέα, αυτοτελή οικογένεια μεθόδων, αλλά για επέκταση των υφιστάμενων προς δύο κατευθύνσεις: αφενός των μοντελοκεντρικών μεθόδων, ώστε οι στόχοι λειτουργίας να αποτυπώνονται στα ίδια τα μοντέλα σχεδίασης μαζί με τα δεδομένα και την πλοήγηση· αφετέρου των ευέλικτων πρακτικών, ώστε οι στόχοι αυτοί να διατυπώνονται ως μετρήσιμα κριτήρια αποδοχής ήδη κατά τη σύνταξη των απαιτήσεων. Η λογική αυτή δεν είναι άγνωστη στη μηχανική λογισμικού: αποτελεί επέκταση της φιλοσοφίας του Test-Driven Development, όπου η απαίτηση μετατρέπεται σε εκτελέσιμη δοκιμή πριν από την υλοποίηση. Η διαφορά είναι ότι το TDD εφαρμόζεται κατά κανόνα σε λειτουργικές απαιτήσεις και σε επίπεδο μονάδας κώδικα, ενώ εδώ το ζητούμενο είναι η αντίστοιχη μεταχείριση των στόχων λειτουργίας του συστήματος στην παραγωγή - διαθεσιμότητας, καθυστέρησης, ποσοστού σφαλμάτων - οι οποίοι θα όφειλαν να διατυπώνονται ως μετρήσιμα κριτήρια κατά τη σχεδίαση και να επαληθεύονται συνεχώς κατά τη λειτουργία.

Μια δεύτερη κατεύθυνση είναι η συστηματική μελέτη του GitOps ως μεθόδου διαχείρισης της συντήρησης και των ενημερώσεων με δηλωτικό τρόπο. Το GitOps έχει ήδη αναφερθεί στην Ενότητα 5.3, εκεί όμως ως παράδειγμα καθιερωμένης βιομηχανικής πρακτικής που δεν αποτυπώνεται στην ερευνητική ατζέντα· εδώ το ζητούμενο είναι διαφορετικό και αφορά τη λειτουργία του ως μεθόδου συντήρησης, δηλαδή το πώς η δηλωτική διαχείριση της επιθυμητής κατάστασης του συστήματος επηρεάζει την αξιοπιστία, τον χρόνο αποκατάστασης και τη δυνατότητα επαναφοράς μετά από αστοχία. Η υποεκπροσώπηση των σχετικών πρακτικών στη βιβλιογραφία, όπως διαπιστώθηκε στην Ενότητα 4.5.5, καθιστά και τις δύο οπτικές ανοιχτές.

Τέλος, μια τρίτη κατεύθυνση είναι η ένταξη της κατανεμημένης ιχνηλάτησης ως ζητήματος πρώτης προτεραιότητας στη σχεδίαση των κατανεμημένων εφαρμογών, ώστε η δυνατότητα διάγνωσης προβλημάτων να είναι εγγενής και όχι επίκτητη.

5.5 RG5 - Ενσωμάτωση της Τεχνητής Νοημοσύνης στον Κύκλο Ζωής

Το πέμπτο κενό είναι αναδυόμενο αλλά ταχύτατα αναπτυσσόμενο, και σχετίζεται με την ενσωμάτωση των εργαλείων τεχνητής νοημοσύνης στην ανάπτυξη εφαρμογών ιστού. Η εμφάνιση εργαλείων AI-ενισχυμένης ανάπτυξης δημιουργεί νέες δυνατότητες σε κάθε στάδιο, αλλά και νέα, ανοιχτά ερευνητικά ερωτήματα. Η τεκμηρίωση του κενού προέρχεται από την επισκόπηση των Li et al. (2024), οι οποίοι κατέγραψαν την αυξανόμενη αξιοποίηση της μηχανικής μάθησης στη δοκιμή εφαρμογών ιστού. Ωστόσο, η εφαρμογή των τεχνικών αυτών στις ειδικές ιδιαιτερότητες των εφαρμογών ιστού - τη σχεδίαση της πλοήγησης, τις responsive διεπαφές, τα ζητήματα ασφάλειας - παραμένει σε μεγάλο βαθμό ανεξερεύνητη. Εξίσου ανοιχτά παραμένουν τα ζητήματα της ποιότητας και της ασφάλειας του κώδικα που παράγεται αυτόματα από τέτοια εργαλεία.

5.5.1 Κατευθύνσεις Αντιμετώπισης

Μια πρώτη κατεύθυνση είναι η εμπειρική αξιολόγηση της ποιότητας και της ασφάλειας του κώδικα που παράγεται από εργαλεία τεχνητής νοημοσύνης. Η αξιολόγηση αυτή οφείλει να διενεργηθεί με αυστηρή μεθοδολογία και σε πραγματικά περιβάλλοντα ανάπτυξης, και όχι μόνο σε ελεγχόμενες συνθήκες, ώστε να αποτιμηθεί η επίδραση των εργαλείων στον πλήρη κύκλο ζωής: αν ο παραγόμενος κώδικας είναι συντηρήσιμος, αν εισάγει ευπάθειες, και αν το κέρδος σε ταχύτητα υλοποίησης αντισταθμίζεται από πρόσθετο κόστος στα μεταγενέστερα στάδια.

Μια δεύτερη κατεύθυνση είναι η ανάπτυξη σημείων αναφοράς (benchmarks) ειδικά προσαρμοσμένων στις τεχνολογίες του ιστού. Η ανάγκη αυτή συνδέεται άμεσα με το εύρημα των Alazmi και De Leon (2022) για την αδυναμία αντικειμενικής σύγκρισης των εργαλείων ασφαλείας, λόγω της ανομοιογένειας των αναφερόμενων ποσοστών ανίχνευσης· χωρίς κοινές, τυποποιημένες εφαρμογές αναφοράς, η αποτίμηση κάθε νέου εργαλείου παραμένει μη συγκρίσιμη και μη αναπαραγώγιμη.

Μια τρίτη και ευρύτερη κατεύθυνση είναι η μεθοδολογική καθοδήγηση για την αξιοποίηση της τεχνητής νοημοσύνης σε ολόκληρο τον κύκλο ζωής και όχι μόνο στο στάδιο της δοκιμής. Οι δυνατότητες εκτείνονται σε πολλαπλά στάδια: στην ανάλυση απαιτήσεων, για την εξαγωγή και την αποσαφήνιση απαιτήσεων από κείμενο· στη σχεδίαση, για τη πρόταση αρχιτεκτονικών επιλογών και την ανίχνευση αντιπροτύπων· στην υλοποίηση, για την παραγωγή και την επιθεώρηση κώδικα· στη δοκιμή, για την έξυπνη επιλογή και ιεράρχηση των δοκιμών μέσα στον αγωγό, ώστε να μειωθεί ο χρόνος και το κόστος χωρίς απώλεια αξιοπιστίας· και στη συντήρηση, για την ανίχνευση ανωμαλιών και τη διάγνωση σφαλμάτων. Το ζητούμενο, συνεπώς, δεν είναι η μεμονωμένη αξιοποίηση σε κάποιο στάδιο, αλλά η ένταξη των δυνατοτήτων αυτών σε συνεκτική μεθοδολογική μορφή - είτε με την επέκταση υφιστάμενων μεθόδων ώστε να προβλέπουν ρητά τη θέση και τους περιορισμούς των AI εργαλείων σε κάθε στάδιο, είτε με τη διαμόρφωση υβριδικών μεθόδων κατά τη λογική που περιγράφηκε στην Ενότητα 5.1.1. Δεδομένου ότι πρόκειται για πεδίο που εξελίσσεται με εξαιρετική ταχύτητα, οι σχετικοί ισχυρισμοί οφείλουν να τεκμηριώνονται με ιδιαίτερη προσοχή σε επικαιροποιημένες πηγές.

5.6 RG6 - Έλλειψη Εμπειρικής Επαλήθευσης σε Βιομηχανικά Περιβάλλοντα

Το έκτο κενό αφορά τη μεθοδολογική αυστηρότητα της ίδιας της έρευνας στο πεδίο. Όπως αναδείχθηκε στην ανάλυση της εμπειρικής αξιολόγησης, κανένα από τα αντιπροσωπευτικά άρθρα δεν παρέχει ένα πλήρως ελεγχόμενο, τυχαιοποιημένο πείραμα για μια ολόκληρη μεθοδολογία ανάπτυξης εφαρμογής ιστού. Η τεκμηρίωση προέρχεται από τη φύση των ίδιων των εργασιών του συνόλου: ακόμη και οι μεθοδολογικά αυστηρότερες εργασίες, όπως η μετα-ανάλυση των Rafique και Mišić (2013), εξετάζουν μεμονωμένες τεχνικές, όπως το TDD, και όχι ολοκληρωμένες μεθόδους σε πραγματικές συνθήκες παραγωγής. Το κενό αυτό είναι εν μέρει εγγενές, καθώς τα ελεγχόμενα πειράματα σε πραγματικά βιομηχανικά έργα είναι δαπανηρά και δύσκολα στην οργάνωσή τους.

5.6.1 Κατευθύνσεις Αντιμετώπισης

Μια πρώτη κατεύθυνση είναι η διεξαγωγή διαχρονικών μελετών περίπτωσης σε εταιρείες ανάπτυξης λογισμικού, οι οποίες θα παρακολουθούν την εφαρμογή μιας μεθόδου σε βάθος χρόνου και θα καταγράφουν τα πραγματικά της αποτελέσματα. Η προσέγγιση αυτή είναι μία ρεαλιστική εναλλακτική του ελεγχόμενου πειράματος για ολόκληρες μεθόδους: αντί να επιχειρεί την παράλληλη ανάπτυξη του ίδιου συστήματος από ισοδύναμες ομάδες, κάτι πρακτικά ανέφικτο, παρακολουθεί την ίδια ομάδα και το ίδιο έργο πριν και μετά την υιοθέτηση μιας μεθόδου, αξιοποιώντας μετρήσιμους δείκτες, όπως εκείνους του μοντέλου DORA.

Μια δεύτερη κατεύθυνση είναι η δημιουργία ανοικτών συνόλων δεδομένων αναφοράς, που θα επιτρέπουν τη συγκρίσιμη και αναπαραγώγιμη αξιολόγηση μεθόδων και εργαλείων. Όπως ανέδειξε η ανάλυση, η απουσία κοινής βάσης σύγκρισης καθιστά τα αποτελέσματα των επιμέρους αξιολογήσεων μη αντιπαραβλητά μεταξύ τους, με συνέπεια κάθε νέα μελέτη να ξεκινά από μηδενική βάση και τα συσσωρευμένα ευρήματα του πεδίου να παραμένουν κατακερματισμένα.

Τέλος, μια τρίτη κατεύθυνση είναι η ανάπτυξη δομημένων προγραμμάτων συνεργασίας ανάμεσα στη βιομηχανία και την ακαδημαϊκή κοινότητα. Η κατεύθυνση αυτή απαντά και στη δεύτερη από τις υποθέσεις που διατυπώθηκαν στην Ενότητα 5.3 για τα αίτια του χάσματος θεωρίας και πράξης, δηλαδή στην περιορισμένη ορατότητα των βιομηχανικών πρακτικών: μέσα από θεσμοθετημένα σχήματα συνεργασίας, η έρευνα αποκτά πρόσβαση σε πραγματικά δεδομένα και, αντίστροφα, τα ευρήματά της επιστρέφουν στην πράξη με τεκμηριωμένο τρόπο.

5.7 RG7 - Βιωσιμότητα και Green Software Engineering για Εφαρμογές Ιστού

Το έβδομο και τελευταίο κενό είναι επίσης αναδυόμενο και αφορά την περιβαλλοντική βιωσιμότητα. Η σημασία του ζητήματος προκύπτει από δύο δεδομένα: ο τομέας των τεχνολογιών πληροφορικής ευθύνεται για περίπου 2% των παγκόσμιων εκπομπών άνθρακα, ενώ οι εφαρμογές ιστού λειτουργούν αδιάλειπτα και σε κλίμακα, με αποτέλεσμα κάθε σχεδιαστική αναποτελεσματικότητα να πολλαπλασιάζεται επί εκατομμύρια αιτήματα. Κρίσιμο είναι, επιπλέον, ότι οι αποφάσεις που καθορίζουν την κατανάλωση - η στρατηγική απόδοσης περιεχομένου, η πολιτική προσωρινής αποθήκευσης, η κοκκίωση των υπηρεσιών - λαμβάνονται κατά τη σχεδίαση και την υλοποίηση, δηλαδή εντός της εμβέλειας των μεθόδων ανάπτυξης. Ουσιαστικά, η απάντηση στο ερώτημα αν μια εφαρμογή ιστού μπορεί να σχεδιαστεί και να υλοποιηθεί ώστε να καταναλώνει λιγότερη ενέργεια είναι, κατ' αρχήν, καταφατική, και οι μοχλοί βρίσκονται σε αποφάσεις που έχουν ήδη περιγραφεί στην παρούσα εργασία: η στρατηγική απόδοσης περιεχομένου της Ενότητας 2.2.4, όπου η εκ των προτέρων παραγωγή στατικού περιεχομένου μεταθέτει υπολογισμό εκτός του χρόνου εκτέλεσης· η προσωρινή αποθήκευση και η χρήση δικτύων παράδοσης περιεχομένου, που περιορίζουν τον όγκο και την απόσταση των μεταφερόμενων δεδομένων· η ελαστική κλιμάκωση των υπηρεσιών, που αποτρέπει τη δέσμευση πόρων χωρίς αντίστοιχο φορτίο· και η ακρίβεια στην ανάκτηση δεδομένων μέσω των διεπαφών, που περιορίζει την υπερβολική ανάκτηση της Ενότητας 2.6.6. Το κενό, συνεπώς, δεν αφορά την τεχνική δυνατότητα αλλά την απουσία μεθοδολογικής καθοδήγησης και μετρήσιμων κριτηρίων: καμία εργασία του συνόλου δεν αποτιμά την επίδραση των αποφάσεων αυτών στην κατανάλωση, ώστε ο μηχανικός να μπορεί να επιλέξει τεκμηριωμένα.

Παρά τη σημασία αυτή, η εφαρμογή των αρχών του green software engineering στις μεθόδους ανάπτυξης εφαρμογών ιστού παραμένει σχεδόν αδιερεύνητη: κανένα από τα άρθρα του συνόλου δεν αναφέρεται στην ενεργειακή αποδοτικότητα ως κριτήριο σχεδίασης ή αξιολόγησης. Το κενό δεν είναι μόνο παρατήρηση της παρούσας επισκόπησης αλλά τεκμηριώνεται και στην ευρύτερη βιβλιογραφία: η πρόσφατη συστηματική ανασκόπηση των Danushi, Forti και Soldani (2024), η οποία ανέλυσε εξήντα πέντε πρωτογενείς μελέτες με ταξινόμηση βασισμένη στα ερωτήματα 5W1H, καταλήγει ρητά ότι οι υπάρχουσες προσεγγίσεις για τον σχεδιασμό ενεργειακά αποδοτικού λογισμικού παραμένουν διάσπαρτες και ότι η ενσωμάτωση της βιωσιμότητας σε όλα τα στάδια του κύκλου ζωής αποτελεί ανοιχτή πρόκληση.

Σημειώνεται, τέλος, ότι η ενεργειακή αποδοτικότητα ανταγωνίζεται συχνά την απόδοση: τεχνικές που μειώνουν την κατανάλωση ενδέχεται να αυξήσουν την καθυστέρηση, ενώ η επιθετική προσωρινή αποθήκευση και η γεωγραφική κατανομή βελτιώνουν την απόδοση αυξάνοντας το αποτύπωμα. Το ζητούμενο δεν είναι, συνεπώς, η μεμονωμένη βελτιστοποίηση της ενεργειακής αποδοτικότητας αλλά η ρητή διαχείριση της αντιστάθμισης μεταξύ αυτής και της απόδοσης, η οποία ανήκει στη σχεδίαση.

5.7.1 Κατευθύνσεις Αντιμετώπισης

Προτείνεται η ενσωμάτωση εργαλείων ενεργειακής αποτίμησης μέσα στον αγωγό της ανάπτυξης, ώστε η κατανάλωση ενέργειας να γίνεται μετρήσιμη και ορατή· η επέκταση των μεθόδων, ιδίως των MDWE, ώστε να περιλαμβάνουν την ενεργειακή αποδοτικότητα ως ρητή μη λειτουργική απαίτηση ήδη από τη φάση της σχεδίασης· και η ελαχιστοποίηση του αποτυπώματος άνθρακα στις cloud αρχιτεκτονικές, με αξιοποίηση carbon-aware τεχνικών χρονοδρομολόγησης που εκτελούν τις εργασίες όταν και όπου η ενέργεια είναι καθαρότερη. Η τελευταία κατεύθυνση υλοποιείται μεν σε επίπεδο υποδομής, προϋποθέτει όμως σχεδιαστικές επιλογές που ανήκουν στη μέθοδο ανάπτυξης: για να μπορεί μια εργασία να μετατεθεί χρονικά ή γεωγραφικά, η εφαρμογή πρέπει να έχει σχεδιαστεί με ασύγχρονη λογική, να διακρίνει ρητά τις χρονικά κρίσιμες από τις ανελαστικές λειτουργίες, και να διαθέτει υπηρεσίες χωρίς κατάσταση και ιδιοδύναμες πράξεις (idempotent operations). Η ενσωμάτωση των απαιτήσεων αυτών στη σχεδίαση αποτελεί, συνεπώς, το μεθοδολογικό αντίστοιχο της τεχνικής αυτής.

5.8 Συνοπτικός Πίνακας Κενών, Κατευθύνσεων και Ορίζοντα Υλοποίησης

Ο ακόλουθος πίνακας συνοψίζει τα επτά ερευνητικά κενά και το σύνολο των κατευθύνσεων αντιμετώπισης που προτάθηκαν στις προηγούμενες ενότητες. Κάθε κατεύθυνση χαρακτηρίζεται ως βασική ή δευτερεύουσα και συνοδεύεται από εκτιμώμενο ορίζοντα υλοποίησης.

Οι τρεις ορίζοντες ορίζονται ως εξής. Άμεσος (ένα έως δύο έτη): η κατεύθυνση μπορεί να υλοποιηθεί με τα υφιστάμενα εργαλεία και μεθόδους, χωρίς να απαιτείται νέα θεωρητική συνεισφορά, και αφορά κενό με άμεσες πρακτικές συνέπειες. Μεσοπρόθεσμος (τρία έως πέντε έτη): η κατεύθυνση απαιτεί νέα μεθοδολογική ή εννοιολογική συνεισφορά, η οποία όμως μπορεί να στηριχθεί σε υπάρχουσα τεχνολογική βάση. Μακροπρόθεσμος (πέντε έως δέκα έτη): η κατεύθυνση απαιτεί συσσώρευση εμπειρικών δεδομένων σε βάθος χρόνου, θεσμικές αλλαγές ή τεχνολογίες που δεν έχουν ακόμη ωριμάσει. Σημειώνεται ότι ο ορίζοντας αποδίδεται σε κάθε κατεύθυνση χωριστά και όχι στο κενό συνολικά, και αποτυπώνει τον χρόνο στον οποίο η κατεύθυνση μπορεί να ξεκινήσει και να αποδώσει τα πρώτα αποτελέσματα, όχι τον χρόνο ολοκλήρωσής της. Δύο κατευθύνσεις του ίδιου κενού ενδέχεται, συνεπώς, να διαφέρουν: όταν η μία μπορεί να στηριχθεί στα υφιστάμενα δεδομένα και εργαλεία, ενώ η άλλη προϋποθέτει νέα συλλογή εμπειρικού υλικού.

Πίνακας 5.1: Σύνοψη των επτά ερευνητικών κενών, του συνόλου των κατευθύνσεων αντιμετώπισης, της σημαντικότητας καθεμιάς και του εκτιμώμενου ορίζοντα υλοποίησης.

ID	**Ερευνητικό Κενό**	**Κατευθύνσεις Αντιμετώπισης**	**Ορίζοντας**
RG1	Έλλειψη ολιστικών οικογενειών μεθόδων	Αντιστοίχιση σταδίων σε τεχνικές και εργαλεία (βασική)	Άμεσος
		Σύνθεση υβριδικών μεθόδων ανάπτυξης (βασική)	Μεσοπρόθεσμος
		Εμπειρική	Μακροπρόθεσμος

		αξιολόγηση των νέων μεθόδων σε πραγματικά έργα (βασική)	
RG2	Παραμέληση της ασφάλειας ως εγκάρσιου ζητήματος	Αυτοματοποίηση δοκιμών ασφαλείας στον αγωγό, με ελέγχους δοχείων (βασική)	Άμεσος
		Κατευθυντήριες γραμμές ενσωμάτωσης των ελέγχων OWASP σε κάθε στάδιο (βασική)	Μεσοπρόθεσμος
		Αξιοποίηση γλωσσικών μοντέλων για δοκιμές και διαλογή ευρημάτων (δευτερεύουσα)	Μεσοπρόθεσμος
RG3	Έλλειψη κοινών βιομηχανικών πρακτικών στην έρευνα	Συστηματική μελέτη IaC, GitOps και των εργαλείων παρατηρησιμότητας (βασική)	Άμεσος
		Εθνογραφικές μελέτες των υβριδικών μεθόδων που εφαρμόζονται στη βιομηχανία (βασική)	Μεσοπρόθεσμος
		Έρευνα για τις πλατφόρμες low-code και no-code (δευτερεύουσα)	Μεσοπρόθεσμος
RG4	Περιορισμένη ενσωμάτωση συντήρησης και παρατηρησιμότητας	Ένταξη της κατανεμημένης ιχνηλάτησης στη σχεδίαση (βασική)	Άμεσος
		Μέθοδοι που ενσωματώνουν στόχους επιπέδου	Μεσοπρόθεσμος

		υπηρεσίας από τη σχεδίαση (βασική)	
		Μελέτη του GitOps ως μεθόδου διαχείρισης ενημερώσεων (δευτερεύουσα)	Άμεσος
RG5	Ενσωμάτωση της Τεχνητής Νοημοσύνης στον Κύκλο Ζωής	Εμπειρική αξιολόγηση ποιότητας και ασφάλειας του παραγόμενου κώδικα (βασική)	Άμεσος
		Σημεία αναφοράς προσαρμοσμένα στις τεχνολογίες του ιστού (βασική)	Μεσοπρόθεσμος
		Μεθοδολογική καθοδήγηση για AI σε όλα τα στάδια του κύκλου ζωής (βασική)	Μεσοπρόθεσμος
RG6	Έλλειψη εμπειρικής επαλήθευσης ολόκληρων μεθόδων	Ανοικτά σύνολα δεδομένων αναφοράς (βασική)	Μακροπρόθεσμος
		Διαχρονικές μελέτες περίπτωσης σε εταιρείες ανάπτυξης (βασική)	Μακροπρόθεσμος
		Δομημένα προγράμματα συνεργασίας βιομηχανίας και ακαδημίας (δευτερεύουσα)	Μεσοπρόθεσμος
RG7	Βιωσιμότητα και green software engineering	Εργαλεία ενεργειακής αποτίμησης στον αγωγό ανάπτυξης (βασική)	Άμεσος
		Επέκταση των	Μεσοπρόθεσμος

		μεθόδων με την ενεργειακή αποδοτικότητα ως απαίτηση (βασική)	
		Σχεδιαστικές προϋποθέσεις για carbon-aware εκτέλεση (δευτερεύουσα)	Μακροπρόθεσμος

Από την κατανομή των ορίζοντων προκύπτει ένας οδικός χάρτης τριών φάσεων για την επόμενη δεκαετία. Η πρώτη φάση, του άμεσου ορίζοντα, αφορά κατευθύνσεις που αξιοποιούν υφιστάμενα μέσα: τη χαρτογράφηση των τεχνικών ανά στάδιο, την αυτοματοποίηση των δοκιμών ασφαλείας, τη συστηματική μελέτη των υποεκπροσωπημένων βιομηχανικών πρακτικών, και την αποτίμηση των AI εργαλείων. Η δεύτερη φάση, του μεσοπρόθεσμου ορίζοντα, στηρίζεται στα αποτελέσματα της πρώτης και αφορά τη διατύπωση νέων ή επεκταμένων μεθόδων - υβριδικών, security-by-design, με ενσωματωμένους στόχους λειτουργίας και ενεργειακά κριτήρια. Η τρίτη φάση, του μακροπρόθεσμου ορίζοντα, αφορά την εμπειρική τους επικύρωση και τις θεσμικές προϋποθέσεις της, οι οποίες απαιτούν συσσώρευση δεδομένων σε βάθος χρόνου και σταθερή συνεργασία με τη βιομηχανία.

Η κοινή συνισταμένη των επτά κενών είναι η ανάγκη για ολοκλήρωση: των σταδίων μεταξύ τους, της ασφάλειας σε κάθε φάση, της θεωρίας με την πράξη, και των νέων παραδειγμάτων της τεχνητής νοημοσύνης και της βιωσιμότητας με τις καθιερωμένες μεθόδους. Τα κυριότερα από αυτά, εκείνα με τον πιο άμεσο ορίζοντα, επανέρχονται στα συμπεράσματα του επόμενου κεφαλαίου.

6. Συμπεράσματα

Το παρόν κεφάλαιο ολοκληρώνει την εργασία, συνοψίζοντας την κύρια συνεισφορά της, τα κύρια αποτελέσματα της ανάλυσης, τις εμπειρίες που αποκομίστηκαν κατά την εκπόνησή της και τις σημαντικότερες ερευνητικές κατευθύνσεις που αξίζει να υιοθετηθούν στο άμεσο μέλλον. Στόχος του δεν είναι απλώς η επανάληψη των ευρημάτων, αλλά η ένταξή τους σε μια συνολική εικόνα που αναδεικνύει τη σημασία τους για την έρευνα και την πράξη.

6.1 Κύρια Συνεισφορά της Εργασίας

Η παρούσα διπλωματική πραγματοποίησε μια συστηματοποιημένη επισκόπηση (systematised review) των μεθόδων, τεχνικών και εργαλείων ανάπτυξης εφαρμογών ιστού, με ιδιαίτερη έμφαση σε προσεγγίσεις που καλύπτουν περισσότερα του ενός στάδια του κύκλου ζωής. Η συνεισφορά της αναπτύσσεται σε πέντε άξονες.

Πρώτον, η εργασία ανέλυσε συστηματικά ένα σύνολο 27 επιλεγμένων, πραγματικών και επαληθεύσιμων άρθρων, κατηγοριοποιημένων σε έξι οικογένειες μεθόδων και σε επιμέρους υποκατηγορίες που προέκυψαν επαγωγικά από το περιεχόμενο των ίδιων των εργασιών, προσφέροντας μια ενοποιημένη χαρτογράφηση που καλύπτει ταυτόχρονα μεθόδους, τεχνικές και εργαλεία. Όπως αναδείχθηκε στο Κεφάλαιο 1, μια τέτοια ενοποιημένη θεώρηση απουσίαζε από τις υφιστάμενες επισκοπήσεις, οι οποίες περιορίζονταν σε μεμονωμένες πτυχές.

Δεύτερον, ανέπτυξε ένα πλαίσιο είκοσι κριτηρίων, κατανεμημένων ισόποσα στα πέντε στάδια του κύκλου ζωής, με ρητά διατυπωμένη σημασιολογία και κανόνα αποτίμησης για καθένα, και το εφάρμοσε σε δώδεκα αντιπροσωπευτικά άρθρα - δύο ανά οικογένεια - με τα υπόλοιπα δεκαπέντε να λειτουργούν ως σύνολο επικύρωσης των ευρημάτων. Η αποτίμηση δεν αφορά την ποιότητα των άρθρων καθαυτών αλλά τον βαθμό στον οποίο η οικογένεια μεθόδων που πραγματεύεται κάθε άρθρο καλύπτει το αντίστοιχο κριτήριο, παράγοντας έτσι μια ποσοτικά θεμελιωμένη συγκριτική αποτίμηση ανά οικογένεια. Η αποτίμηση αυτή διενεργήθηκε από έναν ερευνητή και στηρίζεται, συνεπώς, σε διαφανή αλλά όχι ανεξάρτητα επικυρωμένα κριτήρια, όπως σημειώνεται στις απειλές εγκυρότητας.

Τρίτον, απάντησε συστηματικά και στις οκτώ ερευνητικές ερωτήσεις που τέθηκαν στο Κεφάλαιο 3, με τις επτά να απαντώνται στην ανάλυση του Κεφαλαίου 4 και εκείνη που αφορά τα ανοιχτά ερευνητικά κενά στο Κεφάλαιο 5.

Τέταρτον, εντόπισε και τεκμηρίωσε επτά ερευνητικά κενά, προτείνοντας για το καθένα συγκεκριμένες κατευθύνσεις με εκτιμώμενο ορίζοντα υλοποίησης.

Πέμπτον, ενσωμάτωσε τις πρόσφατες εξελίξεις έως το 2025, αντιμετωπίζοντας παράλληλα την ασφάλεια ως εγκάρσιο ζήτημα που διαπερνά όλα τα στάδια, αντί ως απομονωμένη ανησυχία.

Η συνεισφορά αυτή έχει διακριτή χρησιμότητα για δύο κοινά. Για την ερευνητική κοινότητα, η εργασία προσφέρει δύο αξιοποιήσιμα προϊόντα: αφενός ένα πλαίσιο είκοσι κριτηρίων με ρητή σημασιολογία και ενιαίο κανόνα βαθμολόγησης, το οποίο μπορεί να επαναχρησιμοποιηθεί για την αποτίμηση και άλλων οικογενειών μεθόδων ή νεότερων εργασιών, πέρα από το παρόν σύνολο· αφετέρου επτά τεκμηριωμένα ερευνητικά κενά με κατευθύνσεις και ορίζοντα υλοποίησης, που λειτουργούν ως συγκεκριμένη ερευνητική ατζέντα και υποδεικνύουν πού η υπάρχουσα τεκμηρίωση είναι ασθενέστερη - ιδίως στη μηχανική απαιτήσεων, στη συντήρηση και στην ασφάλεια.

Για τους επαγγελματίες μηχανικούς, η αξία της έγκειται στη συγκριτική εικόνα ανά στάδιο: η τεκμηριωμένη διαπίστωση ότι καμία οικογένεια δεν καλύπτει ισόρροπα ολόκληρο τον κύκλο ζωής, σε συνδυασμό με τη συμπληρωματικότητα των επιδόσεών τους, μετατρέπει την επιλογή μεθόδου από ζήτημα προτίμησης σε τεκμηριωμένη απόφαση συνδυασμού, ενώ η διαπίστωση ότι η ασφάλεια δεν αποτελεί τυποποιημένο συστατικό καμίας οικογένειας λειτουργεί

ως ρητή προειδοποίηση για τον σχεδιασμό των δικών τους διαδικασιών. Χρήσιμη, τέλος, είναι και για τη διδασκαλία του πεδίου, καθώς προσφέρει μια ενιαία, ενημερωμένη εικόνα που διαφορετικά θα απαιτούσε τη μελέτη δεκάδων επιμέρους επισκοπήσεων.

6.2 Κύρια Αποτελέσματα της Ανάλυσης

6.2.1 Τέσσερα Κεντρικά Ευρήματα

Τα αποτελέσματα της ανάλυσης συνοψίζονται σε τέσσερα κεντρικά ευρήματα, καθένα από τα οποία τεκμηριώνεται από τα ποσοτικά δεδομένα του Κεφαλαίου 4.

Το πρώτο εύρημα είναι ότι καμία οικογένεια μεθόδων δεν είναι ολοκληρωμένη. Καμία από τις έξι οικογένειες που εξετάστηκαν δεν επιτυγχάνει πλήρη και ισόρροπη κάλυψη όλων των σταδίων: η υψηλότερη βαθμολογία, εκείνη των Frameworks και της Ασφάλειας με 49,0 στα 80, αντιστοιχεί σε μόλις 61,3% της μέγιστης δυνατής. Το εύρημα αυτό επιβεβαιώνει εμπειρικά την κεντρική υπόθεση εργασίας που διατυπώθηκε στο Κεφάλαιο 1.

Το δεύτερο εύρημα είναι μια σαφής πόλωση ως προς την κατανομή του βάρους στα στάδια. Οι μέθοδοι MDWE κυριαρχούν στη σχεδίαση, με 15,0 στα 16, και επεκτείνονται στην υλοποίηση μέσω της αυτόματης παραγωγής κώδικα, με 7,5, αλλά απουσιάζουν εντελώς από τη διάταξη, με μηδενική βαθμολογία. Αντιστρόφως, οι προσεγγίσεις DevOps και CI/CD κυριαρχούν στη διάταξη, με 15,0 στα 16, καλύπτουν ουσιαστικά και τη δοκιμή με 12,0 μέσω των αυτοματοποιημένων αγωγών, υστερούν όμως σημαντικά στη σχεδίαση, με μόλις 4,0. Η πόλωση δεν συνίσταται, συνεπώς, στον αποκλεισμό όλων των υπόλοιπων σταδίων αλλά στη συγκέντρωση του βάρους σε δύο ή τρία, σε αντίθετα άκρα του κύκλου ζωής - και αυτό ακριβώς εξηγεί γιατί οι εταιρείες υιοθετούν στην πράξη υβριδικά σχήματα, συνδυάζοντας μεθόδους από διαφορετικές οικογένειες.

Το τρίτο εύρημα είναι ότι η ασφάλεια αποτελεί το μεγαλύτερο κενό. Είναι συστηματικά το λιγότερο καλυμμένο κριτήριο, με μέση τιμή μόλις 0,58 στις δύο μονάδες, και αντιμετωπίζεται ως δευτερεύον ζήτημα από τις οικογένειες MDWE, TDD και Agile, παρότι το OWASP Top 10 αφορά κατεξοχήν τις εφαρμογές ιστού. Το DevSecOps, ως εξέλιξη του DevOps, προσφέρει τη λογική για την ουσιαστική ενσωμάτωσή της· η χαμηλή βαθμολογία αποτυπώνει ότι η λογική αυτή δεν έχει ακόμη καθιερωθεί ως τυποποιημένο συστατικό των μεθόδων ανάπτυξης, και όχι ότι απουσιάζουν κατάλληλες προσεγγίσεις.

Το τέταρτο εύρημα, τέλος, είναι το χάσμα ανάμεσα στη θεωρία και την πράξη, το οποίο τεκμηριώνεται με ασφάλεια από την πλευρά της MDWE: η οικογένεια με τη μεγαλύτερη ακαδημαϊκή παραγωγή υιοθετείται ελάχιστα στη βιομηχανία, με το 59% των εταιρειών του δείγματος των Linaje et al. (2019) να δηλώνει χρήση υβριδικών, εσωτερικά διαμορφωμένων προσεγγίσεων. Η αντίστροφη όψη του χάσματος, δηλαδή η υποεκπροσώπηση στην έρευνα πρακτικών με ευρεία βιομηχανική διάδοση, τεκμηριώνεται από την απουσία των εργαλείων υποδομής ως κώδικα και των GitOps πρακτικών από το σύνολο των εργασιών, ενώ για τα συστήματα διαχείρισης περιεχομένου και τις πλατφόρμες low-code διατυπώνεται ως υπόθεση που απαιτεί στοχευμένη διερεύνηση.

6.2.2 Απαντήσεις στις Ερευνητικές Ερωτήσεις

Ο ακόλουθος πίνακας συνοψίζει τις απαντήσεις σε καθεμία από τις οκτώ ερευνητικές ερωτήσεις, όπως προέκυψαν από την ανάλυση του Κεφαλαίου 4 και, ως προς τη RQ6, από τα ερευνητικά κενά του Κεφαλαίου 5.

Πίνακας 6.1: Σύνοψη των απαντήσεων στις οκτώ ερευνητικές ερωτήσεις.

RQ	**Ερώτηση**	**Κύριο Εύρημα**
RQ1	Ποιες μέθοδοι καλύπτουν περισσότερα του ενός στάδια και πώς κατηγοριοποιούνται;	Έξι οικογένειες μεθόδων με επιμέρους υποκατηγορίες· μόλις 5 στα 27 άρθρα τεκμηριώνουν κάλυψη τεσσάρων ή περισσότερων σταδίων.
RQ2	Πώς συγκρίνονται οι μέθοδοι ανά στάδιο και υπό ποια κριτήρια;	Είκοσι κριτήρια με ρητή σημασιολογία και κανόνα αποτίμησης· καμία οικογένεια δεν ξεπερνά το 61,3% της μέγιστης κάλυψης, με τη σχεδίαση και τη συντήρηση τα λιγότερο καλυμμένα στάδια.
RQ3	Πώς αντιμετωπίζεται η ασφάλεια;	Το λιγότερο καλυμμένο κριτήριο (0,58 στις 2)· η λογική του DevSecOps, ως εξέλιξη του DevOps, προσφέρει τα μέσα, χωρίς να έχει καθιερωθεί ως τυποποιημένο συστατικό των μεθόδων.
RQ4	Ποια είναι η σχέση με το νέφος και τα σύγχρονα εργαλεία;	Χρονική πόλωση: τα άρθρα μετά το 2019 καλύπτουν εκτενώς τις cloud-native τεχνολογίες· η κάλυψη εμφανίζεται πλήρης μόνο στις οικογένειες που συνδέονται με την υποδομή εκτέλεσης.
RQ5	Ποια frameworks και εργαλεία χρησιμοποιούνται συχνότερα;	Selenium και Cypress στη δοκιμή, Docker και Kubernetes στη διάταξη, Prometheus και Grafana στη συντήρηση· απουσιάζουν τα εργαλεία υποδομής ως κώδικα και οι GitOps πρακτικές.
RQ6	Ποια ερευνητικά κενά	Επτά κενά, με κεντρικά την

	παραμένουν ανοιχτά;	απουσία ολιστικών οικογενειών, την ασφάλεια και το χάσμα θεωρίας και πράξης. Για καθένα προτείνονται κατευθύνσεις αντιμετώπισης, χαρακτηρισμένες ως βασικές ή δευτερεύουσες και με εκτιμώμενο ορίζοντα υλοποίησης, οι οποίες συγκροτούν ερευνητικό οδικό χάρτη (Κεφάλαιο 5, Πίνακας 5.1).
RQ7	Ποιες μέθοδοι έχουν αξιολογηθεί εμπειρικά;	Ανομοιογενής εικόνα: ισχυρή εμπειρική βάση για το TDD και τις ευέλικτες μεθόδους, κυρίως ποιοτική για microservices και DevOps, ασθενής για την MDWE· κανένα ελεγχόμενο πείραμα ολόκληρης μεθόδου.
RQ8	Πώς εξελίχθηκε διαχρονικά η ερευνητική εστίαση;	Τέσσερις γενιές: MDWE, ευέλικτες μέθοδοι, DevOps και microservices, cloud-native και τεχνητή νοημοσύνη.

6.3 Εμπειρίες που Αποκομίστηκαν

Πέρα από τα επιστημονικά αποτελέσματα, η εκπόνηση της εργασίας δίδαξε ορισμένα ουσιώδη μαθήματα. Το πρώτο αφορά τη δυσκολία της σύνθεσης: η ενοποίηση βιβλιογραφίας από έξι διαφορετικές οικογένειες μεθοδολογιών, καθεμία με τη δική της ορολογία, τους δικούς της στόχους και τα δικά της μεθοδολογικά πλαίσια, αποδείχθηκε ιδιαίτερα απαιτητική και καθόρισε εξαρχής τη δομή των κριτηρίων σύγκρισης. Χωρίς ένα κοινό πλαίσιο αξιολόγησης, η σύγκριση τόσο ετερογενών προσεγγίσεων θα ήταν αδύνατη.

Το δεύτερο μάθημα αφορά τη σημασία της εστίασης: ο περιορισμός σε προσεγγίσεις που καλύπτουν πολλαπλά στάδια αποδείχθηκε κρίσιμος για τη διαχειρισιμότητα του εγχειρήματος, καθώς το πεδίο είναι τεράστιο και μια εξαντλητική κάλυψη θα ήταν πρακτικά ανέφικτη. Το τρίτο, και ίσως το πιο διδακτικό, αφορά την πραγματικότητα του χάσματος θεωρίας και πράξης: η διαπίστωση ότι τεχνολογίες τεράστιας βιομηχανικής παρουσίας απουσιάζουν σχεδόν εντελώς από την ακαδημαϊκή βιβλιογραφία υπήρξε από τα πιο απροσδόκητα και προβληματίζοντα ευρήματα. Το τέταρτο αφορά την ταχύτητα εξέλιξης του πεδίου, η οποία υπογραμμίζει ότι κάθε επισκόπηση αποτελεί αναγκαστικά ένα στιγμιότυπο στον χρόνο. Και το πέμπτο, μεθοδολογικό, αφορά τη θεμελιώδη σημασία της αυστηρής επαλήθευσης κάθε πηγής και κάθε αριθμού· η εμπειρία έδειξε ότι η αξιοπιστία μιας επισκόπησης εξαρτάται απόλυτα από την ακρίβεια των δεδομένων στα οποία στηρίζεται.

6.4 Σημαντικές Ερευνητικές Κατευθύνσεις για το Άμεσο Μέλλον

Από το σύνολο των ερευνητικών κενών, τέσσερις κατευθύνσεις ξεχωρίζουν ως οι πλέον επείγουσες και υποσχόμενες για το άμεσο μέλλον.

6.4.1 Ολιστικές Υβριδικές Μέθοδοι Ανάπτυξης

Η πρώτη και σημαντικότερη κατεύθυνση είναι η σύνθεση ολιστικών, υβριδικών μεθόδων ανάπτυξης που θα γεφυρώνουν τη σχεδίαση με την υλοποίηση και τη λειτουργία. Μια τέτοια μέθοδος θα μπορούσε να αντλεί τη μοντελοκεντρική αυστηρότητα από την MDWE, την ευελιξία από τις ευέλικτες μεθόδους και την αυτοματοποίηση από το DevOps, καλύπτοντας τη πόλωση που ανέδειξε η ανάλυση. Το ζητούμενο δεν είναι μια ακόμη μεμονωμένη μέθοδος προσθετικά στις υπάρχουσες, αλλά η τεκμηριωμένη επιλογή και ενσωμάτωση συγκεκριμένων πρακτικών από διαφορετικές οικογένειες, καθεμιάς στα στάδια όπου αποδεδειγμένα υπερέχει, όπως αναλύθηκε στην Ενότητα 5.1.1.

6.4.2 Security-by-Design ως Εγκάρσια Αρχή

Η δεύτερη κατεύθυνση είναι η καθιέρωση του security-by-design ως εγκάρσιας αρχής σε κάθε στάδιο του κύκλου ζωής. Αντί η ασφάλεια να αντιμετωπίζεται ως εκ των υστέρων έλεγχος, θα πρέπει να ενσωματώνεται από τη μοντελοποίηση των απαιτήσεων έως τη συντήρηση, με αυτοματοποιημένους ελέγχους σε κάθε βήμα του pipeline, επεκτείνοντας τη λογική του DevSecOps σε όλες τις μεθοδολογίες.

6.4.3 Αξιολόγηση της AI-Ενισχυμένης Ανάπτυξης

Η τρίτη κατεύθυνση είναι η εμπειρική αξιολόγηση της AI-ενισχυμένης ανάπτυξης, με ιδιαίτερη έμφαση στην ποιότητα και την ασφάλεια του κώδικα που παράγεται αυτόματα. Καθώς τα εργαλεία αυτά υιοθετούνται ραγδαία στη βιομηχανία, η επιστημονική τεκμηρίωση των δυνατοτήτων και των κινδύνων τους καθίσταται επείγουσα, ώστε η υιοθέτησή τους να γίνεται με επίγνωση και όχι τυφλά.

6.4.4 Εμπειρική Έρευνα για CMS και Low-Code

Η τέταρτη κατεύθυνση είναι η στροφή της ακαδημαϊκής έρευνας προς τις τεχνολογίες που πραγματικά χρησιμοποιεί η βιομηχανία, όπως τα συστήματα διαχείρισης περιεχομένου και οι πλατφόρμες low-code. Η γεφύρωση του χάσματος θεωρίας και πράξης δεν ωφελεί μόνο τη βιομηχανία, αλλά εμπλουτίζει και την ίδια την έρευνα με πραγματικά, σημαντικά προβλήματα προς επίλυση.

6.5 Κλείσιμο

Οι εφαρμογές ιστού αποτελούν τον πυρήνα της σύγχρονης ψηφιακής οικονομίας, και η ανάπτυξή τους είναι ένα πολυδιάστατο εγχείρημα που απαιτεί συστηματικές μεθόδους για κάθε στάδιο του κύκλου ζωής τους. Η παρούσα εργασία έδειξε ότι, παρά τον εντυπωσιακό όγκο της ακαδημαϊκής παραγωγής, η ολιστική κάλυψη του κύκλου ζωής παραμένει ένα ανοιχτό πρόβλημα, και ότι η απόσταση ανάμεσα σε ό,τι προτείνει η ακαδημαϊκή κοινότητα και σε ό,τι χρησιμοποιεί η βιομηχανία εξακολουθεί να είναι σημαντική.

Αν υπάρχει ένα ενιαίο νήμα που διατρέχει το σύνολο των ευρημάτων της εργασίας, αυτό είναι η ένταση ανάμεσα στην εξειδίκευση και την ολοκλήρωση. Κάθε οικογένεια μεθόδων που εξετάστηκε - από τη μοντελοκεντρική σχεδίαση της MDWE και την ευελιξία του Agile, έως την

αυτοματοποίηση του DevOps και την αυστηρότητα του TDD - αναπτύχθηκε για να λύσει ένα συγκεκριμένο, καλά οριοθετημένο πρόβλημα, και σε αυτό διαπρέπει. Η πρόκληση του πεδίου, όπως αναδείχθηκε τόσο από τη συγκριτική ανάλυση όσο και από τα επτά ερευνητικά κενά, δεν είναι η δημιουργία ακόμη περισσότερων εξειδικευμένων εργαλείων, αλλά η ενορχήστρωσή τους σε ένα συνεκτικό σύνολο που θα υπηρετεί ολόκληρο τον κύκλο ζωής. Τα εγκάρσια ζητήματα - η ασφάλεια, η συντήρηση, η βιωσιμότητα - είναι ακριβώς εκείνα που δεν ανήκουν σε κανένα μεμονωμένο στάδιο και γι' αυτό παραμένουν τα λιγότερο καλυμμένα· η αντιμετώπισή τους απαιτεί μια συνθετική, και όχι κατακερματισμένη, οπτική της ανάπτυξης.

Ο δρόμος προς τα εμπρός περνά μέσα από τη σύνθεση: ολιστικά πλαίσια που γεφυρώνουν τη σχεδίαση με την ανάπτυξη, ασφάλεια ενσωματωμένη ως εγκάρσια ανάγκη, εμπειρική επαλήθευση σε πραγματικά περιβάλλοντα, και ακαδημαϊκή προσοχή στις τεχνολογίες που πράγματι κυριαρχούν στην πράξη. Η κάλυψη αυτών των κενών δεν αποτελεί μόνο ερευνητική πρόκληση, αλλά και προϋπόθεση για την ανάπτυξη εφαρμογών ιστού που θα είναι ταυτόχρονα ποιοτικές, ασφαλείς και βιώσιμες - εφαρμογών αντάξιων του κεντρικού ρόλου που διαδραματίζουν στη ζωή μας.

Βιβλιογραφία

Οι αναφορές παρατίθενται σε ενιαίο σύστημα APA (7ης έκδοσης). Ο κατάλογος περιλαμβάνει δύο κατηγορίες πηγών: αφενός τα 27 άρθρα του τελικού συνόλου (corpus) που αναλύθηκαν συστηματικά και παρατίθενται με τους κωδικούς S01–S27 στο Παράρτημα Α, και αφετέρου τις υποστηρικτικές αναφορές υποβάθρου -θεμελιώδεις εργασίες, μεθοδολογικές πηγές και πρότυπα, όπως οι Fielding (2000), Ginige και Murugesan (2001) ή το OWASP Top 10 (2021)- οι οποίες τεκμηριώνουν το θεωρητικό πλαίσιο χωρίς να αποτελούν μέλη του αναλυόμενου συνόλου. Το πλήρες corpus παρατίθεται με τους αντίστοιχους κωδικούς DOI στο Παράρτημα Α.

Abgaz, Y., McCarren, A., Elger, P., Solan, D., Lapuz, N., Bivol, M., Jackson, G., Yilmaz, M., Buckley, J., & Clarke, P. (2023). Decomposition of Monolith Applications Into Microservices Architectures: A Systematic Review. IEEE Transactions on Software Engineering, 49(8), 4213–4242. https://doi.org/10.1109/TSE.2023.3287297

Al-Saqqa, S., Sawalha, S., & AbdelNabi, H. (2020). Agile Software Development: Methodologies and Trends. International Journal of Interactive Mobile Technologies, 14(11), 246–270. https://doi.org/10.3991/ijim.v14i11.13269

Alazmi, S., & De Leon, D. C. (2022). A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanners. IEEE Access, 10, 33200–33219. https://doi.org/10.1109/ACCESS.2022.3161522

Amaro, R., Pereira, R., & Mira da Silva, M. (2025). Mapping DevOps capabilities to the software life cycle: A systematic literature review. Information and Software Technology, 177, 107583. https://doi.org/10.1016/j.infsof.2024.107583

Beck, K. (2002). Test-Driven Development: By Example. Addison-Wesley Professional.

Begel, A., & Nagappan, N. (2007). Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study. In Proceedings of the 1st International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), 255–264. https://doi.org/10.1109/ESEM.2007.85

Bissi, W., Neto, A. G. S. S., & Emer, M. C. F. P. (2016). The effects of test-driven development on internal quality, external quality and productivity: A systematic review. Information and Software Technology, 74, 45–54. https://doi.org/10.1016/j.infsof.2016.02.004

Bogner, J., & Merkel, M. (2022). To Type or Not to Type? A Systematic Comparison of the Software Quality of JavaScript and TypeScript Applications on GitHub. In Proceedings of the 19th International Conference on Mining Software Repositories (MSR '22), 658–669. https://doi.org/10.1145/3524842.3528454

Bushong, V., Abdelfattah, A. S., Maruf, A. A., Das, D., Lehman, A., Jaroszewski, E., Coffey, M., Cerny, T., Frajtak, K., Tisnovsky, P., & Bures, M. (2021). On Microservice Analysis and Architecture Evolution: A Systematic Mapping Study. Applied Sciences, 11(17), 7856. https://doi.org/10.3390/app11177856

Cohn, M. (2009). Succeeding with Agile: Software Development Using Scrum. Addison-Wesley Professional.

Danushi, O., Forti, S., & Soldani, J. (2024). Carbon-Efficient Software Design and Development: A Systematic Literature Review. arXiv preprint arXiv:2407.19901.

Deepa, G., & Thilagam, P. S. (2016). Securing web applications from injection and logic vulnerabilities: Approaches and challenges. Information and Software Technology, 74, 160–180. https://doi.org/10.1016/j.infsof.2016.02.005

Deng, S., Zhao, H., Huang, B., Zhang, C., Chen, F., Deng, Y., Yin, J., Dustdar, S., & Zomaya, A. Y. (2024). Cloud-Native Computing: A Survey From the Perspective of Services. Proceedings of the IEEE, 112(1), 12–46. https://doi.org/10.1109/JPROC.2024.3352584

Di Francesco, P., Lago, P., & Malavolta, I. (2019). Architecting with microservices: A systematic mapping study. Journal of Systems and Software, 150, 77–97. https://doi.org/10.1016/j.jss.2019.01.001

Doğan, S., Betin-Can, A., & Garousi, V. (2014). Web application testing: A systematic literature review. Journal of Systems and Software, 91, 174–201. https://doi.org/10.1016/j.jss.2014.01.004

Doyle, B. J., & Lopes, C. V. (2005). Survey of Technologies for Web Application Development. arXiv:0801.2618.

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. Information and Software Technology, 50(9–10), 833–859. https://doi.org/10.1016/j.infsof.2008.01.006

Eckhardt, J., Vogelsang, A., & Méndez Fernández, D. (2016). Are "Non-functional" Requirements really Non-functional? An Investigation of Non-functional Requirements in Practice. In 38th International Conference on Software Engineering (ICSE) (pp. 832–842). IEEE/ACM. https://doi.org/10.1145/2884781.2884788

El Aouni, F., Moumane, K., Idri, A., & Najib, M. (2024). A systematic literature review on Agile, Cloud, and DevOps integration: Challenges, benefits. Information and Software Technology, 177, 107569. https://doi.org/10.1016/j.infsof.2024.107569

Escalona, M. J., & Koch, N. (2004). Requirements Engineering for Web Applications – A Comparative Study. Journal of Web Engineering, 2(3), 193–212.

Faustino, J., Adriano, D., Amaro, R., Pereira, R., & Mira da Silva, M. (2022). DevOps benefits: A systematic literature review. Software: Practice and Experience, 52(9), 1905–1926. https://doi.org/10.1002/spe.3096

Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures [Doctoral dissertation, University of California, Irvine].

Forsgren, N., Humble, J., & Kim, G. (2018). Accelerate: The Science of Lean Software and DevOps. IT Revolution Press.

Garousi, V., Mesbah, A., Betin-Can, A., & Mirshokraie, S. (2013). A systematic mapping study of web application testing. Information and Software Technology, 55(8), 1374–1396. https://doi.org/10.1016/j.infsof.2013.02.006

Ginige, A., & Murugesan, S. (2001). Web Engineering: An Introduction. IEEE MultiMedia, 8(1), 14–18. https://doi.org/10.1109/93.923949

Grant, M. J., & Booth, A. (2009). A typology of reviews: an analysis of 14 review types and associated methodologies. Health Information & Libraries Journal, 26(2), 91–108. https://doi.org/10.1111/j.1471-1842.2009.00848.x

Jalali, S., & Wohlin, C. (2012). Systematic Literature Studies: Database Searches vs. Backward Snowballing. In Proceedings of the 6th ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (pp. 29–38). ACM. https://doi.org/10.1145/2372251.2372257

Jazayeri, M. (2007). Some Trends in Web Application Development. In Future of Software Engineering (FOSE '07), 199–213. IEEE. https://doi.org/10.1109/FOSE.2007.26

Kertusha, F., Assress, A., Duman, O., & Arcuri, A. (2025). A Survey on Web Testing: On the Rise of AI and Applications in Industry. arXiv preprint arXiv:2503.05378.

Leite, L., Rocha, C., Kon, F., Milojicic, D., & Meirelles, P. (2019). A Survey of DevOps Concepts and Challenges. ACM Computing Surveys, 52(6), Article 127. https://doi.org/10.1145/3359981

Li, T., Huang, R., Cui, C., Towey, D., Ma, L., Li, Y.-F., & Xia, W. (2024). A Survey on Web Application Testing: A Decade of Evolution. arXiv preprint arXiv:2412.10476.

Linaje, M., Preciado, J. C., & Morales-Chaparro, R. (2019). Comparison of development methodologies in web applications. Information and Software Technology, 109, 93–109. https://doi.org/10.1016/j.infsof.2019.02.008

Loja, A., & Maita, T. (2024). Comparative Evaluation of Performance Efficiency in Terms of Temporal Behavior and Resource Utilization, According to the ISO/IEC 25010 Model, in a Web Application Developed with Angular, React.js, and Vue.js. In XVIII Multidisciplinary International Congress on Science and Technology. Springer. https://doi.org/10.1007/978-3-031-52255-0_21

Mell, P., & Grance, T. (2011). The NIST Definition of Cloud Computing (NIST Special Publication 800-145). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-145

Molina-Ríos, J., & Pedreira-Souto, N. (2019). Approach of Agile Methodologies in the Development of Web-Based Software. Information, 10(10), 314. https://doi.org/10.3390/info10100314

Murugesan, S., Deshpande, Y., Hansen, S., & Ginige, A. (2001). Web Engineering: A New Discipline for Development of Web-Based Systems. In Web Engineering (LNCS 2016, pp. 3–13). Springer. https://doi.org/10.1007/3-540-45144-7_2

Niemietz, M., Korth, M., Mainka, C., & Somorovsky, J. (2021). Over 100 Bugs in a Row: Security Analysis of the Top-Rated Joomla Extensions. arXiv preprint arXiv:2102.03131.

O'Reilly, T. (2007). What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. Communications & Strategies, 65, 17–37.

OWASP. (2021). OWASP Top 10 – 2021. The Open Worldwide Application Security Project. https://owasp.org/Top10/

Pastrana, M., Ordoñez, H., Cobos-Lozada, C. A., & Muñoz, M. (2025). Best Practices Evidenced for Software Development Based on DevOps and Scrum: A Literature Review. Applied Sciences, 15(10), 5421. https://doi.org/10.3390/app15105421

Pressman, R. S. (1998). Can Internet Applications be Engineered? IEEE Software, 15(5), 104–110. https://doi.org/10.1109/52.714825

Rafique, Y., & Mišić, V. B. (2013). The Effects of Test-Driven Development on External Quality and Productivity: A Meta-Analysis. IEEE Transactions on Software Engineering, 39(6), 835–856. https://doi.org/10.1109/TSE.2012.28

Rossi, G., Urbieta, M., Distante, D., Rivero, J. M., & Firmenich, S. (2016). 25 Years of Model-Driven Web Engineering: What we achieved, What is missing. CLEI Electronic Journal, 19(3). https://doi.org/10.19153/cleiej.19.3.1

Santos, A., Vegas, S., Dieste, O., Uyaguari, F., Tosun, A., Fucci, D., Turhan, B., Scanniello, G., Romano, S., Karac, I., Kuhrmann, M., Mandić, V., Ramač, R., Pfahl, D., Engblom, C., Kyykka, J., Rungi, K., Palomeque, C., Spisak, J., … Juristo, N. (2021). A family of experiments on test-driven development. Empirical Software Engineering, 26, 43. https://doi.org/10.1007/s10664-020-09895-8

Schwinger, W., Retschitzegger, W., Schauerhuber, A., Kappel, G., Wimmer, M., Pröll, B., Cachero Castro, C., Casteleyn, S., De Troyer, O., Fraternali, P., Garrigós, I., Garzotto, F., Ginige, A., Houben, G.-J., Koch, N., Moreno, N., Pastor, O., Paolini, P., Pelechano, V., … Zhang, G. (2008).

A survey on web modeling approaches for ubiquitous web applications. International Journal of Web Information Systems, 4(3), 234–305. https://doi.org/10.1108/17440080810901089

Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices. IEEE Access, 5, 3909–3943. https://doi.org/10.1109/ACCESS.2017.2685629

Stack Overflow. (2024). Stack Overflow Developer Survey 2024. https://survey.stackoverflow.co/2024/

Torrecilla-Salinas, C. J., Sedeño, J., Escalona, M. J., & Mejías, M. (2016). Agile, Web Engineering and Capability Maturity Model Integration: A Systematic Literature Review. Information and Software Technology, 71, 92–107. https://doi.org/10.1016/j.infsof.2015.11.002

Velepucha, V., & Flores, P. (2023). A Survey on Microservices Architecture: Principles, Patterns and Migration Challenges. IEEE Access, 11, 88339–88358. https://doi.org/10.1109/ACCESS.2023.3305687

Wakil, K., & Jawawi, D. N. A. (2015). Model Driven Web Engineering: A Systematic Mapping Study. e-Informatica Software Engineering Journal, 9(1), 107–142. https://doi.org/10.5277/e-Inf150106

Waseem, M., Liang, P., & Shahin, M. (2020). A systematic mapping study on microservices architecture in DevOps. Journal of Systems and Software, 170, 110798. https://doi.org/10.1016/j.jss.2020.110798

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. In Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (EASE '14), Article 38. https://doi.org/10.1145/2601248.2601268

Zhou, X., Peng, X., Xie, T., Sun, J., Ji, C., Li, W., & Ding, D. (2023). Revisiting the practices and pains of microservice architecture in reality: An industrial inquiry. Journal of Systems and Software, 195, 111521. https://doi.org/10.1016/j.jss.2022.111521

Boehm, B. W. (1981). Software engineering economics. Prentice-Hall.

Boehm, B., & Basili, V. R. (2001). Software defect reduction top 10 list. Computer, 34(1), 135–137. https://doi.org/10.1109/2.962984

Cunningham, W. (1992). The WyCash portfolio management system. In Addendum to the Proceedings on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA ‘92) (pp. 29–30). ACM. https://doi.org/10.1145/157709.157715

Martin, R. C. (2008). Clean code: A handbook of agile software craftsmanship. Prentice Hall.

Menzies, T., Nichols, W., Shull, F., & Layman, L. (2017). Are delayed issues harder to resolve? Revisiting cost-to-fix of defects throughout the lifecycle. Empirical Software Engineering, 22(4), 1903–1935. https://doi.org/10.1007/s10664-016-9469-x

Sommerville, I. (2016). Software engineering (10th ed.). Pearson.

Παράρτημα Α - Πλήρης Κατάλογος Άρθρων (Corpus)

Ο πίνακας παραθέτει και τα 27 άρθρα του τελικού συνόλου. Όλα είναι πραγματικές, επαληθεύσιμες δημοσιεύσεις· ο κωδικός DOI ή το αναγνωριστικό arXiv επιτρέπει τον άμεσο εντοπισμό κάθε εργασίας. Η στήλη «Στ.» δηλώνει τον αριθμό των σταδίων SDLC που καλύπτει το άρθρο και η στήλη «QC» το σκορ ποιότητας (μέγιστο 10).

Πίνακας Α.1: Πλήρης κατάλογος των 27 άρθρων του τελικού συνόλου. Πηγή: επεξεργασία του συγγραφέα.

ID	Συγγραφείς	Τίτλος	Venue	DOI / arXiv	Στ.	QC
S01	Rossi et al. (2016)	25 Years of Model-Driven Web Engineering	CLEI E. Journal 19(3)	10.19153/cleiej.19.3.1	3	8
S02	Linaje et al. (2019)	Comparison of development methodologies in web applications	Inf. & Softw. Tech. 109	10.1016/j.infsof.2019.02.008	2	8
S03	Wakil & Jawawi (2015)	Model Driven Web Engineering: A Systematic Mapping Study	e-Informatica SEJ 9(1)	10.5277/e-Inf150106	2	7
S04	Torrecilla-Salinas et al. (2016)	Agile, Web Engineering and CMMI: A SLR	Inf. & Softw. Tech. 71	10.1016/j.infsof.2015.11.002	3	9
S05	Pastrana et al. (2025)	Best Practices for Software Development Based on DevOps and Scrum	Applied Sciences 15(10),5421	10.3390/app15105421	3	8

S06	Dybå & Dingsøyr (2008)	Empirical studies of agile software development: A systematic review	Inf. & Softw. Tech. 50(9-10)	10.1016/j.infsof.2008.01.006	2	9
S07	Molina-Ríos & Pedreira-Souto (2019)	Approach of Agile Methodologies in the Development of Web-Based Software	Information 10(10),314	10.3390/info10100314	2	7
S08	Al-Saqqa et al. (2020)	Agile Software Development: Methodologies and Trends	IJIM 14(11)	10.3991/ijim.v14i11.13269	2	6
S09	Begel & Nagappan (2007)	Usage and Perceptions of Agile Software Development	ESEM 2007,255-264	10.1109/ESEM.2007.85	1	7
S10	Shahin et al. (2017)	Continuous Integration, Delivery and Deployment: A Systematic Review	IEEE Access 5	10.1109/ACCESS.2017.2685629	3	9
S11	El Aouni et al. (2024)	A SLR on Agile, Cloud, and DevOps integration	Inf. & Softw. Tech. 177,107569	10.1016/j.infsof.2024.107569	4	8
S12	Amaro et al. (2025)	Mapping DevOps capabilities to the	Inf. & Softw. Tech. 177,10758	10.1016/j.infsof.2024.107583	5	8

		software life cycle	3			
S13	Faustino et al. (2022)	DevOps benefits: A systematic literature review	Softw. Pract. Exper. 52(9)	10.1002/spe.3096	3	7
S14	Velepucha & Flores (2023)	A Survey on Microservices Architecture	IEEE Access 11	10.1109/ACCESS.2023.3305687	4	9
S15	Zhou et al. (2023)	Revisiting the practices and pains of microservice architecture	J. Syst. Softw. 195,111521	10.1016/j.jss.2022.111521	4	8
S16	Waseem et al. (2020)	A systematic mapping study on microservices architecture in DevOps	J. Syst. Softw. 170,110798	10.1016/j.jss.2020.110798	3	8
S17	Bushong et al. (2021)	On Microservice Analysis and Architecture Evolution	Applied Sciences 11(17),7856	10.3390/app11177856	3	7
S18	Abgaz et al. (2023)	Decomposition of Monolith Applications Into Microservices	IEEE TSE 49(8)	10.1109/TSE.2023.3287297	2	8

S19	Deng et al. (2024)	Cloud-Native Computing: A Survey From the Perspective of Services	Proc. IEEE 112(1)	10.1109/JPROC.2024.3352584	5	9
S20	Rafique & Mišić (2013)	The Effects of TDD on External Quality and Productivity: A Meta-Analysis	IEEE TSE 39(6)	10.1109/TSE.2012.28	2	9
S21	Bissi et al. (2016)	The effects of TDD on internal/external quality and productivity	Inf. & Softw. Tech. 74	10.1016/j.infsof.2016.02.004	2	8
S22	Santos et al. (2021)	A family of experiments on test-driven development	Empir. Softw. Eng. 26,43	10.1007/s10664-020-09895-8	2	8
S23	Doğan et al. (2014)	Web application testing: A systematic literature review	J. Syst. Softw. 91	10.1016/j.jss.2014.01.004	1	8
S24	Li et al. (2024)	A Survey on Web Application Testing: A Decade of Evolution	arXiv:2412.10476	arXiv:2412.10476	1	7
S25	Alazmi & De Leon (2022)	A SLR on Web Applicatio	IEEE Access 10	10.1109/ACCESS.2022.316152	3	8

		n Vulnerability Scanners		2		
S26	Kertusha et al. (2025)	A Survey on Web Testing: On the Rise of AI and Applications in Industry	arXiv:2503.05378	arXiv:2503.05378	1	7
S27	Bogner & Merkel (2022)	To Type or Not to Type? JS vs TypeScript on GitHub	MSR '22,658-669	10.1145/3524842.3528454	1	8