ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΙΓΑΙΟΥ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Συνέργεια μεταξύ Τεχνητής Νοημοσύνης και Τεχνολογίας Λογισμικού (Synergy between Artificial Intelligence and Software Engineering)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Στεφανίδου Κωνσταντίνα

Επιβλέπων: Κρητικός Κυριάκος (Αναπληρωτής Καθηγητής)
Μέλη εξεταστικής επιτροπής: Χαραλαμπίδης Ιωάννης (Καθηγητής), Συμεωνίδης Παναγιώτης (Αναπληρωτής Καθηγητής)

Σάμος, 2024

Περιεχόμενα

Λίστα Εικόνων

Ακρωνύμια:

ΤΛ	Τεχνολογία Λογισμικού
ΤΝ	Τεχνητή Νοημοσύνη
ΜΜ	Μηχανική Μάθηση

Acronyms:

SE	Software Engineering
AI	Artificial Intelligence
SNARC	Stochastic Neural Analog Reinforcement Calculator
IEEE	Institute of Electrical and Electronics Engineers
AGI	Artificial General Intelligence
MLPs	Master Limited Partnerships
CNNs	Convolutional Neural Networks
RNNs	Recurrent Neural Networks
NLP	Natural Language Processing
ML	Machine learning

GPT	Generative Pre-trained Transformer
AGV	Automated Guided Vehicle
UAVs	Unmanned aerial Vehicles
AUVs	Autonomous Underwater Vehicles
DBMS	Database Management system
NATO	North Atlantic Treaty
HTTP	HyperText Transfer Protocol
AWS	Amazon Web Services
SDLC	Software Development Life Cycle
SRS	Software Requirements Specification
UAT	User Acceptance Testing
SLR	Systematic Literature Review
GUI	Graphical User Interface
KBS	Knowledge Based Systems

Περίληψη

Η διπλωματική εργασία αποσκοπεί στη διερεύνηση των προοπτικών συνεργασίας και από τις δύο δυνατές κατευθύνσεις μεταξύ των τομέων της Τεχνητής Νοημοσύνης (Artificial intelligence) και της Τεχνολογίας Λογισμικού (Software Engineering). Αφενός, έχει ως στόχο να διερευνήσει πώς η Τεχνολογία Λογισμικού μπορεί να συμβάλλει στη βελτίωση, επέκταση και αυτοματοποίηση των μεθόδων, τεχνικών και εφαρμογών της Τεχνητής Νοημοσύνης. Αφετέρου, στοχεύει να διερευνήσει πώς η Τεχνητή Νοημοσύνη μπορεί να συμβάλλει στη βελτίωση και αυτοματοποίηση των δραστηριοτήτων που συνδέονται με την ανάπτυξη έργων λογισμικού οποιουδήποτε είδους. Θα διερευνηθούν οι θετικές αλληλεπιδράσεις μεταξύ των δύο τομέων αλλά και οι πιθανές αρνητικές, με τη χρήση πραγματικών παραδειγμάτων από παγκόσμια ή τοπικά περιστατικά/γεγονότα. Η διερεύνηση βασίζεται τόσο στη βιβλιογραφία όσο και στην κοινή λογική, εκμεταλλευόμενη τις ικανότητες και τη συμπληρωματικότητα των δύο τομέων. Επίσης, εξετάζονται προηγούμενες έρευνες, σχετικές μελέτες που έχουν αναδείξει την αμοιβαία επίδραση της Τεχνητής Νοημοσύνης και της Τεχνολογίας Λογισμικού, με σκοπό να διεξαχθούν προσωπικά συμπεράσματα και να αναλυθεί η εμπειρία που αποκομίσθηκε μέσω της έρευνας. Τέλος, μέσω της παρούσας μελέτης εντοπίζονται ερευνητικά κενά και υποσχόμενες ερευνητικές κατευθύνσεις που μπορούν να καλύψουν αυτά τα κενά, με απώτερο σκοπό την πραγματοποίηση της οραματιζόμενης συνέργειας μεταξύ των 2 τομέων.

Λέξεις κλειδιά: αυτοματοποίηση, αλληλεπίδραση, συμπληρωματικότητα, επίδραση.

Abstract

This thesis aims to investigate the prospects of cooperation between the fields of Artificial Intelligence (AI) and Software Engineering (SE) from both possible directions. On one hand, it aims to explore how SE can contribute to the improvement, extension and automation of AI methods, techniques and applications. On the other hand, it aims to investigate how AI can contribute to the improvement and automation of activities associated with the development of software projects of any kind. Positive interactions between the two areas and potential negative ones will be explored, using real examples from global or local cases/events. The exploration will be based on both the existing literature as well as the common sense, taking advantage of the capabilities and complementarity of the two sectors. A reference is also made to previous research and related studies that have highlighted the mutual impact of AI and SE,in order to conduct personal conclusions and to analyze the gained experience through the survey. Through this study, research gaps and promising research directions that fill these gaps are identified, with the ultimate goal of realizing the envisioned synergy between the two domains.

Keywords: automation, interaction, complementarity, impact.

Κεφάλαιο 1: Εισαγωγή

1.1 Εισαγωγή

Η Τεχνητή Νοημοσύνη αναφέρεται στον τομέα της επιστήμης και της τεχνολογίας που αναζητά τρόπους σχεδίασης και αναπαραγωγής ανθρώπινων γνωστικών λειτουργιών με τη χρήση μηχανών. Στοχεύει στην ανάπτυξη υπολογιστικών συστημάτων που μπορούν να επιδείξουν έξυπνη συμπεριφορά, δηλαδή να έχουν την ικανότητα της αντίληψης, της εκμάθησης, της σκέψης.

Η Τεχνολογία Λογισμικού αναφέρεται στον τομέα της επιστήμης και της τεχνολογίας που εστιάζει στην ανάπτυξη, στον σχεδιασμό, στην υλοποίηση και στη συντήρηση εφαρμογών λογισμικού. Στοχεύει στη βέλτιστη και αξιόπιστη λειτουργικότητα των εφαρμογών, αλλά και στη μέγιστη δυνατή ασφάλεια και απόδοση του λογισμικού.

Κατά τη διάρκεια των τελευταίων δεκαετιών, αυτοί οι δύο τομείς ανέπτυξαν ξεχωριστά τις έρευνες τους, έχοντας περιορισμένη ανταλλαγή επιστημονικών ευρημάτων μεταξύ τους. Στο παρόν, αλλά και στο άμεσο μέλλον τίθεται θέμα αλληλεπίδρασης των δύο πεδίων καθώς, η διασύνδεση τους μπορεί να ανοίξει νέους ορίζοντες για την ανθρώπινη πρόοδο, ανατρέποντας τα παραδοσιακά πλαίσια της τεχνολογίας και επιφέροντας ταχύτερη ανάπτυξη και προσαρμογή στους σύγχρονους ρυθμούς ζωής. Η ΤΛ αποτελεί τη βάση για ανάπτυξη και εξέλιξη της ΤΝ, παρέχοντας πολλαπλές δυνατότητες, εργαλεία που απαιτούνται για τη βελτιστοποίηση της απόδοσης των αλγορίθμων και άλλων πρακτικών εφαρμογών. Αντίστοιχα, με την ανάπτυξη και την ενσωμάτωση της ΤΝ στον κόσμο του λογισμικού, παρουσιάζονται νέες δυνατότητες για τη βελτίωση της αποτελεσματικότητας, της αξιοπιστίας, της ευφυΐας των εφαρμογών λογισμικού.

Ουσιαστικά, η τεχνητή νοημοσύνη αφορά την ευφυΐα των μηχανών, ενώ η τεχνολογία λογισμικού αποσκοπεί στην υλοποίηση των προγραμμάτων και εφαρμογών. Η συνέργεια μεταξύ τους έχει αρχίσει να αλλάζει τον τρόπο αξιοποίησης και χρήσης του λογισμικού. Η εφαρμογή της ΤΝ στον τομέα του λογισμικού δίνει τη δυνατότητα δημιουργίας προϊόντων (λογισμικού), τα οποία θα έχουν την ικανότητα να μαθαίνουν, να προσαρμόζονται ανάλογα με τις συνθήκες και να αυτοβελτιώνονται με βάση την εμπειρία και χρήση τους. Ετσι, με τη χρήση της ευφυούς συμπεριφοράς το λογισμικό εξελίσσεται και γίνεται πιο ευέλικτο και προσαρμοστικό στις ανάγκες των χρηστών του.

Παράλληλα, η συνέργεια μεταξύ της ΤΝ και της Τεχνολογίας Λογισμικού δημιουργεί προκλήσεις και ηθικά ζητήματα. Τα προβλεπόμενα συστήματα που θα αναπτυχθούν μελλοντικά θα συνδυάζουν τεχνητή νοημοσύνη με προηγμένο λογισμικό τα οποία έγκειται να επηρεάσουν θετικά αλλά και αρνητικά την ανθρώπινη διαβίωση. Βασικό ζήτημα αποτελούν οι απρόοπτες επιπτώσεις των αποφάσεων που θα λαμβάνονται και των διεργασιών που θα εκτελούνται από τα υπολογιστικά συστήματα. Τα συστήματα αυτά θα λειτουργούν αυτόματα, έτσι όσο αφορά τη συμπεριφορά τους, τίθενται θέματα μεροληψίας, δικαιοσύνης, διαφάνειας και λήψης ευθυνών. Επιπλέον, μία μηχανή τεχνητής νοημοσύνης εφόσον διαθέτει σύστημα λογισμικού, διαθέτει και (υποκείμενη) βάση δεδομένων, η οποία χρήζει ασφάλειας και

ιδιωτικοποίησης. Έτσι, προκύπτει ζήτημα προστασίας δεδομένων και έλεγχος πρόσβασης όσον αφορά εμπιστευτικές και προσωπικές πληροφορίες. Ένα ακόμη ζήτημα που προκύπτει είναι ότι όσο τα ευφυή συστήματα Τεχνητής Νοημοσύνης εκτελούν εργασίες αυτοματοποιημένα, ταχύτατα και με περισσότερη ακρίβεια, τόσο οι αρμοδιότητες του ανθρώπου θα λιγοστεύουν. Ένα μελλοντικό σενάριο λοιπόν είναι η αύξηση της ανεργίας και της οικονομικής ανισότητας.

Συνεπώς η παρούσα ερευνητική εργασία έχει ως σκοπό τον προσδιορισμό και την ανάλυση των όρων της τεχνολογίας λογισμικού και της τεχνητής νοημοσύνης. Δινεται έμφαση στην δυνητική εξέλιξη των δύο τομέων και διεξάγεται μελέτη του ευρύ φάσματος των εφαρμογών τους σε διάφορους τεχνολογικούς τομείς. Στη συνέχεια αναφέρονται τα πλεονεκτήματα και μειονεκτήματα της συνέργειας των δύο επιστημονικών πεδίων αποτυπώνονται τα αποτελέσματα και η μελλοντική τους ενδεχόμενη αλληλεπίδραση και επέκταση σύμφωνα με σύγχρονη βιβλιογραφία και προσωπικά συμπεράσματα που προέκυψαν από την έρευνα.

1.2 Δομή Εργασίας

Η υλοποίηση της παρούσας διπλωματικής εργασίας πραγματοποιήθηκε μέσω της ανασκόπησης διάφορων βιβλιογραφικών πηγών, η οποία βασίστηκε σε αναζήτηση, φιλτράρισμα και ανάλυση πληροφοριών από άρθρα, μελέτες, έρευνες που είναι διαθέσιμα στο διαδίκτυο.

Η εργασία διαχωρίζεται σε 5 επόμενα κεφάλαια, τα οποία διαρθρώνονται ως εξής:

- Στο Κεφάλαιο 2 αναλύεται το θεωρητικό υπόβαθρο, μέσα από το οποίο παρέχονται πρωταρχικές γνώσεις που απαιτούνται για την καλύτερη κατανόηση του υπόλοιπου μέρους της διπλωματικής.

- Στο Κεφάλαιο 3 αναλύεται η μεθοδολογία βιβλιογραφικής ανασκόπησης και τα αποτελέσματά της με τη παροχή στατιστικών.

- Στο Κεφάλαιο 4 γίνεται η ανάλυση των σχετικών άρθρων που εντοπίστηκαν με βάση την ανάγκη παροχής απαντήσεων στα ερευνητικά ερωτήματα που τέθηκαν στο Κεφάλαιο 3.

- Στο Κεφάλαιο 5 εξετάζονται με βάση την προηγούμενη ανάλυση, ορισμένες από τις βασικές προκλήσεις που αφορούν την τρέχουσα έρευνα και τις κύριες κατευθύνσεις που αυτή πρέπει να ακολουθήσει για να αντιμετωπίσει τις προκλήσεις αυτές.

❖ Στο Κεφάλαιο 6 παρατίθενται τα συμπεράσματα της βιβλιογραφικής έρευνας που επιτελέστηκε, αναφέρεται η προσωπική εμπειρία που αποκομίσθηκε και προτείνεται μια μελλοντική επέκταση της εργασίας.

Κεφάλαιο 2: Θεωρητικό Υπόβαθρο

Στο δεύτερο Κεφάλαιο παρέχεται η θεωρητική ανάλυση της Τεχνητής Νοημοσύνης μέσα από την Ενότητα 2.1 και της Τεχνολογίας Λογισμικού μέσα από την Ενότητα 2.2. Το κεφάλαιο αυτό προσεγγίζει σε βάθος τους δύο επιστημονικούς τομείς, προσδιορίζοντας βασικές έννοιες, χαρακτηριστικά και είδη, δίνοντας τις κατάλληλες βάσεις για την συνέχεια. Σκοπός του θεωρητικού υποβάθρου είναι η καλύτερη κατανόηση του υπόλοιπου μέρους της διπλωματικής και της έρευνας που διεξήχθη όσον αφορά την συνέργεια των δύο τομέων.

2.1 Τεχνητή Νοημοσύνη

Στη πρώτη αυτή ενότητα του τρέχοντος κεφαλαίου παρατίθεται η αναλυτική περιγραφή της έννοιας “Τεχνητή Νοημοσύνη”. Περιγράφεται η φιλοσοφία της, γίνεται αναδρομή στην ιστορική της εξέλιξη, αναλύονται τα είδη καθώς και οι κύριες εφαρμογές της.

2.1.1 Ορισμός Τεχνητής Νοημοσύνης

Ο όρος “Τεχνητή Νοημοσύνη” (ΤΝ) αναφέρεται στην ικανότητα των μηχανών να εκτελούν εργασίες που συνδέονται με την ανθρώπινη νοημοσύνη, όπως η εκμάθηση και επίλυση προβλημάτων. Αποτελεί ένα διακριτό τομέα της πληροφορικής, ο οποίος ασχολείται με την ανάπτυξη υπολογιστικών συστημάτων που προσομοιώνουν την ανθρώπινη αντίληψη και σκέψη, έχοντας ως στόχο ακόμα και να τις υπερβαίνουν. Η ΤΝ απαιτεί χρήση αλγορίθμων και μοντέλων που επιτρέπουν στα υπολογιστικά συστήματα να λειτουργούν αποτελεσματικά και με έξυπνο τρόπο.

Η ΤΝ συναντάται σε λογισμικό ενσωματωμένο σε μηχανές αλλά και ανεξάρτητο:

- Λογισμικό Τεχνητής Νοημοσύνης: αφορά λογισμικό, προγράμματα και εφαρμογές που αναπτύσσονται για να επιτελούν κάποιο είδος νοημοσύνης ή κατανόησης σε υπολογιστικές πλατφόρμες. Αυτά τα είδη λογισμικού βασίζονται σε αλγόριθμους και μαθηματικά μοντέλα που επιτρέπουν στους υπολογιστές να αναλύουν δεδομένα, να κατανοούν πληροφορίες, να αναπαράγουν πληροφορία, να λαμβάνουν αποφάσεις και να λύνουν προβλήματα. Παραδείγματα τέτοιου λογισμικού ΤΝ είναι: οι μηχανές αναζήτησης, συστήματα αναγνώρισης προσώπου και ομιλίας, καθώς και συστήματα ανάλυσης εικόνας.

- ΤΝ ενσωματωμένη σε μηχανές: αναφέρεται σε ενσωματωμένες λειτουργίες, ικανότητες, και δεξιότητες βασιζόμενες σε ΤΝ (μέσω λογισμικού) τόσο μηχανών όσο και άλλων ειδών συσκευών. Οι προκείμενες μηχανές αποκτούν δυνατότητα λήψης αποφάσεων σε ζητήματα που προκύπτουν, μαθαίνουν πως να λειτουργούν με βάση

πληροφορίες που αντλούν από το περιβάλλον τους και αποκτούν εμπειρία ως προς τη συμπεριφορά που θα πρέπει να επιδεικνύουν. Παραδείγματα μηχανών που διαθέτουν ενσωματωμένη τεχνητή νοημοσύνη είναι: τα ρομπότ, τα έξυπνα τηλέφωνα (smartphones), τα αυτόνομα οχήματα (autonomous vehicles), τα τηλεκατευθυνόμενα αεροσκάφη (drones), και οι έξυπνες ηλεκτρικές συσκευές σπιτιού (smart home appliances).

Το λογισμικό αναπτύσσεται ανεξάρτητα, αντιπροσωπεύει προγράμματα και αλγορίθμους, μπορεί να ενσωματωθεί σε διάφορες εφαρμογές, όπως και μηχανές. Η ενσωμάτωση του λογισμικού στις μηχανές γίνεται με σκοπό την ενίσχυση της λειτουργικότητας και των δυνατότητων τους, επιδεικνύοντας ευφυή συμπεριφορά. Ο συνδυασμός αυτός, δηλαδή λογισμικού και υλικού επιτυγχάνει τη δημιουργία μιας έξυπνης μηχανής τεχνητής νοημοσύνης. Χάρη στην ευελιξία του λογισμικού ΤΝ είναι εφικτή η ενσωμάτωσή του σε ένα ευρύ φάσμα μηχανών, από τα πιο απλά ηλεκτρονικά είδη έως και τις πιο σύνθετες μηχανές που χρησιμοποιούνται στη βιομηχανία.

2.1.2 Ιστορική Εξέλιξη

- Οι αρχαίοι Έλληνες, προσπαθώντας να ερμηνεύσουν φυσικά αλλά και ανεξήγητα φαινόμενα για την εποχή, χρησιμοποιούσαν τη λογική και διάφορα πειράματα για να επιβεβαιώσουν ή αποδείξουν τις υποθέσεις και τους υπολογισμούς τους. Οι απαρχές της Τεχνητής Νοημοσύνης ανάγονται από τους συλλογισμούς του Αριστοτέλη (384-322 π.Χ.), οι οποίοι παρείχαν πρότυπα εκφράσεων που έδιναν πάντα σωστά συμπεράσματα από σωστές υποθέσεις βασισμένα στην Αριστοτέλεια Συλλογιστική **[6]**. Στη συνέχεια, έπεται μια ιστορική αναδρομή των πιο σημαντικών επιτευγμάτων της σύγχρονης εποχής, που οδήγησαν και έθεσαν τις σωστές βάσεις για την ανάπτυξη της ΤΝ.

- Το 1854 ο George Boole έθεσε τις βάσεις της προτασιακής λογικής (propositional logic) **[5]**.

- Το 1879 ο Gottlieb Frege πρότεινε ένα σύστημα αυτοματοποιημένης συλλογιστικής και έθεσε τις βάσεις του κατηγορηματικού λογισμού (predicate calculus) **[5]**.

- Το 1943 οι McCulloch και Pitts πρότειναν ένα μοντέλο τεχνητών νευρώνων (artificial neurons) που είχε τη δυνατότητα να μαθαίνει και να υπολογίζει κάθε υπολογίσιμη συνάρτηση **[5]**.

- Το 1950 ο Alan Turing εμπνεύστηκε το τεστ της μίμησης (τεστ Τούρινγκ), δοκιμάζοντας την ικανότητα μιας μηχανής να επιδεικνύει ευφυή συμπεριφορά παρόμοια με την ανθρώπινη **[7]**.

- Το 1951 οι Minsky και Edmonts υλοποίησαν το πρώτο νευρωνικό δίκτυο (neural network), το SNARC, το οποίο είχε 40 νευρώνες και χρησιμοποιούσε 3000 λυχνίες **[5]**.

- Η χρήση του όρου "Artificial Intelligence" ("A.I.") (ΤΝ) έγινε πρώτη φορά από τον John McCarthy το 1956 σε ένα συνέδριο με βασικό θέμα, "*Οι μηχανές που σκέφτονται"*. Η ΤΝ ερμηνεύτηκε ως: <<Η ικανότητα της μηχανής να μπορεί να σκέφτεται και να μιμείται την ανθρώπινη συμπεριφορά και ευφυΐα, αλλά να μην την αντικαθιστά>> **[6]**.

- Το 1958 ο McCarthy δημιούργησε τη γλώσσα προγραμματισμού Lisp **[7]**.

- Το 1965 Ο Έντουαρτ Φάιγκενμπαουμ δημιούργησε το πρώτο έμπειρο σύστημα (expert system) (το οποίο είναι ένα υπολογιστικό σύστημα, που μιμείται τις ικανότητες ενός ειδικού, διαθέτει εξειδικευμένες γνώσεις και λαμβάνει αποφάσεις και λύσεις πολύπλοκων προβλημάτων) ονόματι Dendral, μια δεκαετή προσπάθεια ανάπτυξης λογισμικού που συμπέρανε τη μοριακή δομή οργανικών ενώσεων χρησιμοποιώντας ενδείξεις επιστημονικών οργάνων **[7]**.

- Το 1966 ο Weizenbaum δημιούργησε το ELIZA, το πρώτο αυτόνομο πρόγραμμα συνομιλίας μέσω υπολογιστή **[8]**.

- Το 1970 αναπτύχθηκε το Planner (μια εντυπωσιακή επίδειξη αλληλεπίδρασης μεταξύ ανθρώπου και υπολογιστή), το οποίο χρησιμοποιήθηκε στο SHRDLU (ένα πρώιμο πρόγραμμα υπολογιστή κατανόησης φυσικής γλώσσας). Στο πρόγραμμα αυτό, ο χρήστης μπορούσε να συνομιλήσει με υπολογιστή, να μετακινήσει αντικείμενα, να δώσει όνομα σε συλλογές και να ρωτήσει την κατάσταση του εικονικού κουτιού **[7]**.

- Το 1971 ξεκίνησε η εργασία πάνω στο σύστημα αυτόματης απόδειξης θεωρημάτων (automated theorem proval) Boyer-Moore στο Εδιμβούργο **[7]**.

- Το 1972 Οι Alain Colmerauer και Philippe Roussel από το Πανεπιστήμιο της Μασσαλίας σε συνεργασία με τον Robert Kowalski από το Πανεπιστήμιο του Εδιμβούργου δημιούργησαν και εφάρμοσαν τη γλώσσα λογικού προγραμματισμού Prolog. Η Prolog ήταν μια από τις πρώτες λογικές γλώσσες προγραμματισμού, η οποία χρησιμοποιήθηκε για απόδειξη θεωρημάτων, για αυτοματοποιημένο σχεδιασμό, για επαναγραφή όρων, σε έμπειρα συστήματα, καθώς και σε συστήματα τύπων, έχοντας ως αρχική χρήση την επεξεργασία φυσικής γλώσσας **[7]**.

- Το 1973 στο Εδιμβούργο αναπτύχθηκε το ρομπότ συναρμολόγησης "Φρέντι" (Freddy), το οποίο είναι ένα ευπροσάρμοστο σύστημα συναρμολόγησης που ελέγχεται από υπολογιστές **[7]**.

- Το 1974 ο Τέντ Σόρτλιφ συνέγραψε τη διατριβή του σχετικά με το πρόγραμμα MYCIN στο πανεπιστήμιο Στάνφορντ, το οποίο κατέδειξε μια πολύ πρακτική προσέγγιση στην ιατρική διάγνωση που βασίζεται σε κανόνες, ενώ λειτουργεί ακόμα και με παρουσία

αβεβαιότητας. Παρά το γεγονός ότι το δανείστηκε από το DENDRAL, οι δικές του συνεισφορές επηρέασαν έντονα το μέλλον των έμπειρων συστημάτων **[7]**.

- Τη δεκαετία του '80 επήλθε η αναγέννηση των τεχνητών νευρωνικών δικτύων (artificial neural networks).

- Το 1986 οι Rumelhart and McClelland περιέγραψαν τη δημιουργία προσομοιώσεων της αντίληψης στον υπολογιστή.

- Το 1987 διεξήχθη το 1ο Διεθνές Συνέδριο για τα Νευρωνικά Δίκτυα του οργανισμού IEEE.

- Το 1991 η εφαρμογή σχεδίασης ενεργειών DART χρησιμοποιήθηκε αποτελεσματικά στον Α' Πόλεμο του Κόλπου και αντάμειψε 30 χρόνια έρευνας στην ΤΝ του Αμερικανικού Στρατού **[7]**.

- Το 1994 οι Ντίκμαννς και Ντάιμλερ-Μπενζ κατάφεραν να επιδείξουν τις δυνατότητες της αυτόνομης οδήγησης, οδηγώντας αυτόνομο αυτοκίνητο για περισσότερο από 1.000 χλμ. σε μια εθνική οδό του Παρισιού, υπό συνθήκες βαρείας κυκλοφορίας και με ταχύτητα ως και 130 χλμ./ώρα **[7]**.

- Το 1997 ο παγκόσμιος πρωταθλητή σκακιού Γκάρι Κασπάροφ νικήθηκε από τον υπολογιστή Deep blue της IBM **[7]**.

- Το 1998 έγινε η πρώτη επιτυχημένη εμφάνιση ΤΝ σε οικιακό περιβάλλον με την κυκλοφορία του Φέρμπι (Furby) της Tiger Electronics, το οποίο είναι ένα ηλεκτρονικό ρομποτικό παιχνίδι που μοιάζει με χάμστερ **[7]**.

- Το 1999 η Sony λάνσαρε ένα από τα πρώτα αυτόνομα κατοικίδια ΤΝ ονόματι AIBO **[7]**.

- Το 2000 το ρομπότ Nomad εξερεύνησε απομακρυσμένες περιοχές στην Ανταρκτική, αναζητώντας δείγματα μετεωριτών **[7]**.

- Το 2002 η εταιρεία iRobot παρουσίασε τη πρώτη γενιά αυτόνομων ηλεκτρικών σκουπών Roomba, οι οποίες χρησιμοποιούν ένα σύνολο αισθητήρων για τη πλοήγηση στην επιφάνεια του δαπέδου ενός σπιτιού.

- Το 2004 η DARPA ξεκίνησε το πρόγραμμα DARPA Grand Challenge, προκαλώντας συμμετέχοντες να δημιουργήσουν αυτόνομα οχήματα παρέχοντας ως αντάλλαγμα χρηματικό έπαθλο **[7]**.

- Το 2005 το project Blue Brain επιχείρησε να αναπτύξει μια ψηφιακή ανακατασκευή του εγκεφάλου του ποντικιού **[7]**.

- Το 2009 δημιουργήθηκε το πρώτο αυτο-οδηγούμενο αυτοκίνητο από την Google **[7]**.

- Το 2010 δημιουργήθηκε η Siri, η πρώτη έξυπνη προσωπική βοηθός που μπορεί να εκτελέσει ένα ευρύ φάσμα εντολών του χρήστη, η οποία αρχικά ξεκίνησε ως εφαρμογή για iOS. Αργότερα, το 2011,ενσωματώθηκε στο iPhone 4s της Apple χωρίς να αποτελεί πλέον ξεχωριστή εφαρμογή **[9]**.

- Το 2011 αναπτύχθηκε το σύστημα υπολογιστών IBM Watson, το οποίο έχει τη δυνατότητα να απαντά σε ερωτήσεις που τίθενται από τον χρήστη σε φυσική γλώσσα **[10]**.

- Το 2014 η Amazon κυκλοφόρησε ένα έξυπνο ηχείο ονόματι Echo, το οποίο διαθέτει ενσωματωμένη την έξυπνη υπηρεσία προσωπικού βοηθού Alexa τύπου Siri. Η Alexa ακούει φωνητικές εντολές των χρηστών και εκτελεί τις ανάλογες λειτουργίες, όπως να παρέχει πληροφορίες, μουσική, ειδήσεις, και προγνώσεις καιρού **[11]**.

- Το 2016 δημιουργήθηκε το κοινωνικό ανθρωποειδές ρομπότ Σοφία (Sophia) από την εταιρεία Hanson Robotics και έκανε τη πρώτη της εμφάνιση στο South by Southwest στο Ώστιν (Austin) του Τέξας. Αποκαλείται “κοινωνικό ρομπότ” διότι μπορεί να μιμηθεί κοινωνική συμπεριφορά και να προκαλέσει συναισθήματα, κάτι το οποίο αποδείχθηκε και μέσω των συνεντεύξεων που έδωσε σε μέσα ενημέρωσης. Το 2017 μάλιστα πήρε και την υπηκοότητα της Σαουδικής Αραβίας και έγινε το πρώτο ρομπότ με νομική προσωπικότητα **[12].**

- Το 2017 κυκλοφόρησε το τραγούδι “Break Free” της Taryn Southern σε συνεργασία με το Amper the AI, το οποίο είναι σύστημα ΤΝ που μπορεί να παράγει επαγγελματική μουσική. Το σύστημα αυτό αναπτύχθηκε από μια ομάδα επαγγελματιών μουσικών και ειδικών της τεχνολογίας και είναι το πρώτο σύστημα αυτου του είδους που συνέθεσε και παρήγαγε ένα ολόκληρο μουσικό άλμπουμ με το όνομα IAMAI **[13]**.

- Το 2022 η OpenAI λάνσαρε το ChatGPT 3, το οποίο είναι ένα σύστημα τεχνητής νοημοσύνης, σχεδιασμένο να αλληλοεπιδρά με τους χρήστες μέσω γραπτού λόγου σε φυσική γλώσσα. Μπορεί να ανταποκριθεί σε ερωτήσεις, να παράγει κείμενο σύμφωνα με τις γνώσεις που διαθέτει σε ευρεία γκάμα θεμάτων, να υλοποιήσει προγράμματα ηλεκτρονικού υπολογιστή σε διάφορες γλώσσες προγραμματισμού και άλλα **[14]**.

2.1.3 Τεχνολογική Μοναδικότητα

Στη φιλοσοφία της τεχνητής νοημοσύνης αναλύεται και επεξηγείται ένας σημαντικός όρος, “Τεχνολογική Μοναδικότητα” ή “Μοναδικότητα” (“Technological Singularity” ή “Singularity”), με τον οποίο εκφράζεται η ανεξέλεγκτη και μη προβλέψιμη τεχνολογική

ανάπτυξη, την οποία κανένας ανθρώπινος νους δεν θα μπορεί να προβλέψει ή αποτρέψει. Η ιδέα αυτή αποτελεί ένα μελλοντικό και ίσως φανταστικό ενδεχόμενο πολλών επιστημόνων.

Ο μαθηματικός John von Neumann γνωστός για τη συνεισφορά του σε πολλούς τομείς της επιστήμης, μελέτησε την εξέλιξη της τεχνολογίας και των υπολογιστών. Η έννοια της Μοναδικότητας φημολογείται ότι προέκυψε από τις ιδέες του σχετικά με τις αλλαγές που θα επιφέρει η επιταχυνόμενη πρόοδος της τεχνολογίας στον τρόπο ζωής του ανθρώπου.

Ο όρος “Singularity” αρχικά διαδόθηκε από τον μαθηματικό και πληροφορικό Vernor Vinge κατά το έτος 1993 σε ένα άρθρο στο οποίο υποστήριζε ότι μόλις οι άνθρωποι δημιουργήσουν νοημοσύνη μεγαλύτερη από τη δική τους, θα υπάρξει μια τεχνολογική και κοινωνική μετάβαση παρόμοια από κάποια άποψη με "τον δεμένο χωροχρόνο στο κέντρο μιας μαύρης τρύπας”. Μετέπειτα στο δοκίμιο του *The Coming Technological Singularity* περιέγραψε το τέλος της ανθρώπινης εποχής, καθώς η νέα υπερνοημοσύνη θα συνεχιζόταν να αναβαθμίζεται και θα προχωρούσε τεχνολογικά με ακατανόητο ρυθμό **[16]**. Προέβλεψε επίσης ότι <<Μέσα σε τριάντα χρόνια, θα έχουμε τα τεχνολογικά μέσα για να δημιουργήσουμε υπεράνθρωπη νοημοσύνη. Λίγο αργότερα, η ανθρώπινη εποχή θα τελειώσει>> και πιο συγκεκριμένα <<Θα εκπλαγώ αν αυτό το γεγονός συμβεί πριν από το 2005 ή μετά το 2030>> **[17].**

Ο Stephen Hawking εξέφρασε και τη δική του ανησυχία για την τεχνητή υπερνοημοσύνη, η οποία θα μπορούσε να προκαλέσει ακόμη και την εξαφάνιση του ανθρώπινου είδους.

Γενικότερα, οι επιπτώσεις της Μοναδικότητας έχουν συζητηθεί από πολλούς επιστήμονες και προκύπτουν διάφορα ερωτήματα, όπως για το πως θα επηρεάσει τη μελλοντική κοινωνία, οικονομία και τι θα προκύψει στη συνέχεια, προκαλώντας έτσι ποικίλες αντιδράσεις. Παράλληλα, θέμα σημερινής συζήτησης αποτελεί ο σχηματισμός πλαισίου που θα διέπει και θα ελέγχει την εξέλιξη της ΤΝ, ώστε να μη τελεστεί εις βάρος της ανθρωπότητας, διασφαλίζοντας την ιδιωτικότητα, την ασφάλεια και τη διαφάνεια.

2.1.4 Είδη Τεχνητής Νοημοσύνης

Η τεχνητή νοημοσύνη χωρίζεται σε διάφορα είδη, καθένα από τα οποία εκφράζει μια διαφορετική προσέγγιση της εξελισσόμενης αυτής τεχνολογίας. Παρακάτω αναλύονται τα βασικά είδη ΤΝ, τα οποία μέλλεται να διευρυνθούν και να εξελιχθούν.

- Στενή Τεχνητή Νοημοσύνη (Narrow AI): θεωρείται “αδύναμη” ή στενή εξαιτίας της περιορισμένης λειτουργικότητάς της. Ειδικότερα, είναι σχεδιασμένη να μπορεί να εκτελεί εργασίες ή λειτουργίες οδηγούμενες από πολύπλοκους αλγόριθμους και νευρωνικά δίκτυα, ωστόσο εντοπίζεται σε περιορισμένο κλειστό πεδίο. Ουσιαστικά, στερείται προσαρμοστικότητα, δηλαδή δεν έχει την ικανότητα να επεκτείνει τις γνώσεις και τις δεξιότητές της εκτός του συγκεκριμένου πεδίου για το οποίο έχει εκπαιδευτεί.

Παραδείγματα στενής τεχνητής νοημοσύνης είναι: η αναγνώριση φωνής, η αναγνώριση προσώπου, η επεξεργασία γραπτού κειμένου κ.λ.π.

- Γενική Τεχνητή Νοημοσύνη (Artificial General Intelligence - AGI): θεωρείται διακεκριμένη ΤΝ και αναφέρεται σε υπολογιστικά συστήματα που έχουν ικανότητες παρόμοιες με την ανθρώπινη νοημοσύνη, δηλαδή την ικανότητα να κατανοούν, να αναλύουν και να αντιδρούν στον κόσμο γύρω τους, αντιγράφοντας την ανθρώπινη νοημοσύνη. Προς το παρών συστήματα AGI δεν είναι υπαρκτά, καθώς αποτελούν έναν ενεργό τομέα έρευνας που δεν είναι σίγουρο ότι θα επιτευχθεί μελλοντικά. Ωστόσο, τέτοια συστήματα μέλλει να φέρουν τεράστια αλλαγή, η οποία αποκαλείται ως το τελικό σημείο από πολλούς ειδικούς της τεχνητής νοημοσύνης.

- Τεχνητή Υπερνοημοσύνη (Superintelligent AI): αποτελεί ένα υποθετικό κλάδο τεχνητής νοημοσύνης που δεν μιμείται απλώς την ανθρώπινη νοημοσύνη αλλά την υπερβαίνει σε ικανότητες. Σε αυτό το σενάριο, τα συστήματα θα έχουν τη δυνατότητα να εκτελούν εργασίες με υπερβατική απόδοση, διαθέτοντας τη μέγιστη δυνατή μνήμη καθώς και ταχύτερη ικανότητα επεξεργασίας και ανάλυσης δεδομένων και ερεθισμάτων. Μία τέτοια υπερφυές μηχανή ΤΝ θα μπορεί να προβλέψει και να αποφασίσει τη βέλτιστη επιλογή για προβλήματα που θα προκύπτουν. Επιπλέον, μέλλεται οι υπερφυείς μηχανές ΤΝ να διαθέτουν συναισθήματα, να έχουν ανθρώπινες ανάγκες αλλά ακόμη και να επιδεικνύουν ικανότητα αυτοσυντήρησης.

2.1.5 Κατηγορίες και Εφαρμογές ΤΝ

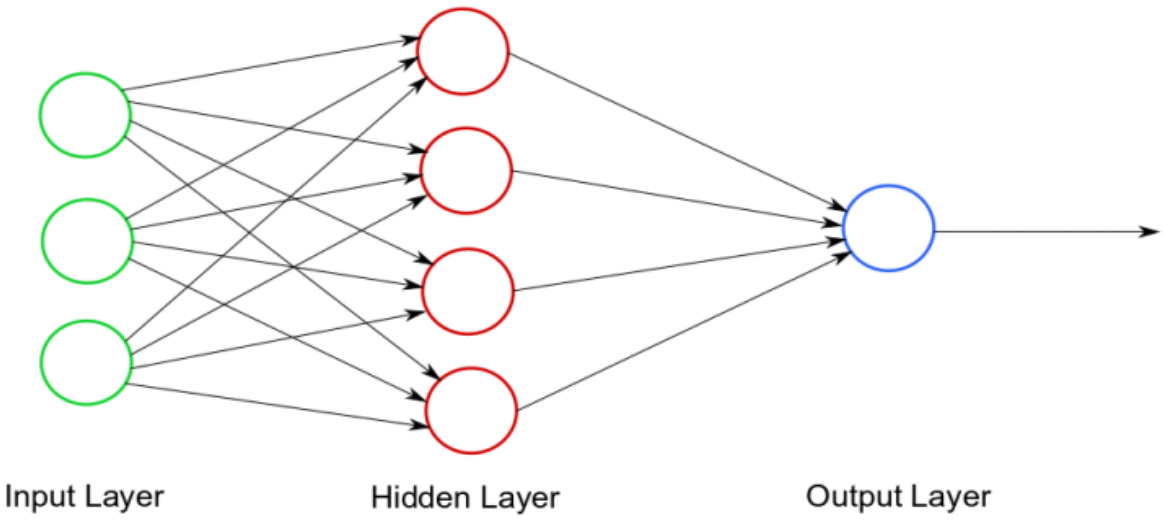
Βασικός στόχος των υπολογιστικών συστημάτων τεχνητής νοημοσύνης είναι η μίμηση στοιχείων ευφυίας που παρατηρούνται στον άνθρωπο, όπως είναι η ικανότητα της γνώσης, αντίληψης, μνήμης, μάθησης, προσαρμογής στο περιβάλλον και απόκτησης εμπειρίας. Αναπτύχθηκαν λοιπόν, αλγόριθμοι και μέθοδοι που βασίζονται στον συλλογισμό ενός ανθρώπου όσο αφορά την επίλυση πολύπλοκων προβλημάτων και την εκτέλεση εργασιών.

Παρακάτω αναλύονται οι κυριότερες κατηγορίες και εφαρμογές εκτέλεσης “έξυπνων” εργασιών που εφαρμόζονται στον χώρο της τεχνητής νοημοσύνης.

2.1.5.1 Νευρωνικά Δίκτυα (Neural Networks)

Το τεχνητό νευρωνικό δίκτυο ή απλούστερα νευρωνικό δίκτυο είναι ένα δίκτυο που αποτελείται από απλούς υπολογιστικούς κόμβους, οι οποίοι ονομάζονται νευρώνες και είναι διασυνδεδεμένοι μεταξύ τους. Αυτό το δίκτυο προσομοιάζει το Κεντρικό Νευρικό Σύστημα του ανθρώπου, αντιγράφοντας την δομή και λειτουργία του ανθρώπινου εγκεφάλου.

Όσον αφορά τη λειτουργία του δικτύου, η διαδικασία που ακολουθείται χωρίζεται σε τρία επίπεδα-στρώματα (όπως φαίνεται στην Εικόνα 1).



Εικόνα 1 - Απλό Νευρωνικό Δίκτυο [31]

Στο στρώμα εισόδου (Input Layer), κάθε νευρώνας λαμβάνει πολλές εισόδους (inputs) με πληροφορίες από εξωτερικούς παράγοντες, τις οποίες επεξεργάζεται, αναλύει και μεταβιβάζει στο επόμενο επίπεδο. Το κρυμμένο στρώμα (Hidden Layer), αποτελεί το ενδιάμεσο στάδιο ανάμεσα στο επίπεδο εισόδου και εξόδου, όπου οι υπολογιστικοί νευρώνες εκτελούν υπολογισμούς και λαμβάνουν αποφάσεις. Στο στρώμα εξόδου (Output Layer), παράγεται το τελικό αποτέλεσμα που προκύπτει από την εσωτερική επεξεργασία και καθορίζεται η έξοδος (output) του νευρωνικού δικτύου.

Να σημειωθεί ότι, ένα νευρωνικό δίκτυο που έχει μόνο τρία στρώματα, δηλ. μόνο ένα κρυφό, χαρακτηρίζεται ως απλό νευρωνικό δίκτυο. Ενώ, αν αποτελείται από πολλά κρυφά στρώματα (πχ. τρία ή τέσσερα), θεωρείται βαθύ νευρωνικό δίκτυο που επιτελεί βαθιά μάθηση.

Η τριμερής αυτή σχεδιαστική δομή επιτρέπει στα νευρωνικά δίκτυα να λειτουργούν με ευφυία, λαμβάνοντας έξυπνες αποφάσεις και επιλύοντας πολύπλοκα προβλήματα. Αυτό επιτυγχάνεται χάρη στη δυνατότητα εκπαίδευσης τους σε μεγάλα σύνολα δεδομένων και στην ικανότητα μοντελοποίησης πολλαπλών σχέσεων μεταξύ των δεδομένων εισόδου και εξόδου, παράγοντας έτσι συμπεράσματα με μεγαλύτερη ακρίβεια.

Υπάρχουν τρία κύρια είδη νευρωνικών δικτύων: τα νευρωνικά δίκτυα τροφοδοσίας (Master Limited Partnerships - MLPs), τα συνελικτικά νευρωνικά δίκτυα (Convolutional Neural Networks - CNNs), και τα επαναλαμβανόμενα νευρωνικά δίκτυα (Recurrent Neural Networks - RNNs).

- Τα νευρωνικά δίκτυα τροφοδοσίας ή αλλιώς τα πολυστρωματικά perceptrons (MLPs), είναι ο απλούστερος τύπος νευρωνικού δικτύου. Είναι η μικρότερη

μονάδα νευρωνικού δικτύου, η οποία εκτελεί υπολογισμούς με σκοπό την ανίχνευση χαρακτηριστικών ευφυίας στα δεδομένα εισόδου. Αποτελείται από το στρώμα εισόδου, το κρυμμένο στρώμα και το στρώμα εξόδου. Τα νευρωνικά δίκτυα αυτού του είδους χρησιμοποιούνται για προβλήματα κατηγοριοποίησης και πρόβλεψης.

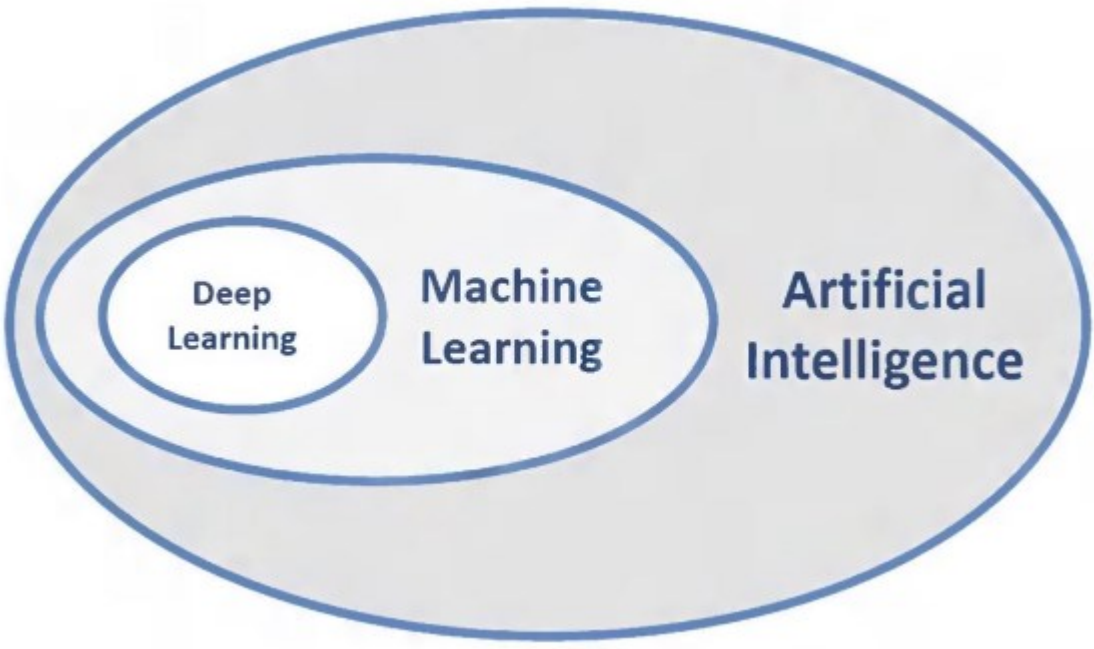
- Τα συνελικτικά νευρωνικά δίκτυα αξιοποιούν έναν τύπο πολυστρωματικού perceptron και περιλαμβάνουν ένα ή περισσότερα συνελικτικά στρώματα, τα οποία είναι ομαδοποιημένα (pooled) ή εξ ολοκλήρου συνδεδεμένα (fully-connected). Τα στρώματα αυτά αποτελούνται από φίλτρα με δυνατότητα εκμάθησης που χρησιμοποιούνται πάνω στα δεδομένα για την εξαγωγή τοπικών μοτίβων ή χαρακτηριστικών σε διαφορετικές χωρικές τοποθεσίες. Το στοίβαγμα των στρωμάτων αυτών έπειτα επιτρέπει την εκμάθηση ιεραρχικών αναπαραστάσεων των δεδομένων εισόδου, συλλαμβάνοντας αυξανόμενα πολύπλοκα και αφηρημένα χαρακτηριστικά. Αυτός ο τύπος νευρωνικού δικτύου χρησιμοποιείται κυρίως σε λειτουργίες μηχανικής όρασης όπως αναγνώρισης εικόνας, προσώπου, χαρακτηριστικών, ταξινόμησης εικόνας καθώς και σε λειτουργίες επεξεργασίας φυσικής γλώσσας.

- Τα επαναλαμβανόμενα νευρωνικά δίκτυα είναι σχεδιασμένα για να επεξεργάζονται διαδοχικά δεδομένα, όπως δεδομένα χρονοσειρών ή συμβολοσειρών. Διαθέτουν συνδέσεις μεταξύ των νευρώνων που σχηματίζουν κατευθυνόμενους κύκλους έτσι ώστε να επιδεικνύουν δυναμική χρονική συμπεριφορά. Με αυτό τον τρόπο είναι σε θέση να ανιχνεύουν μακροπρόθεσμες εξαρτήσεις μεταξύ δεδομένων μεταβλητού μήκους που τους επιτρέπουν να εκτελούν λειτουργίες πρόβλεψης , όπως οικονομικές προβλέψεις, προβλέψεις εξέλιξης τιμών καθώς και λειτουργίες δημιουργίας ακολουθιών (sequence generation) και επεξεργασίας φυσικής γλώσσας.

2.1.5.2 Μηχανική Μάθηση (Machine Learning)

Είναι ένα από τα κυριότερα υποσύνολα της τεχνητής νοημοσύνης (όπως φαίνεται παρακάτω στην Εικόνα 1). Αποτελεί μια μέθοδο εκμάθησης βάσει δεδομένων, που ασχολείται με την δημιουργία και εκπαίδευση αλγορίθμων που μπορούν να μαθαίνουν από τα δεδομένα, να αναγνωρίζουν μοτίβα και να λαμβάνουν αποφάσεις χρησιμοποιώντας ελάχιστη ανθρώπινη παρέμβαση. Αναλυτικότερα, η μηχανή συλλέγει δεδομένα, τα οποία υπόκεινται σε επεξεργασία, έπειτα επιλέγεται ο τύπος αλγορίθμου μάθησης που θα χρησιμοποιηθεί, δηλαδή επιβλεπόμενη (Supervised learning) ή μη επιβλεπόμενη (Unsupervised learning) ή ενισχυτική μάθηση (Reinforcement learning). Με τη χρήση των επεξεργασμένων δεδομένων εκπαιδεύεται το μοντέλο (μηχανικής μάθησης), το οποίο μπορεί έπειτα να χρησιμοποιηθεί για να προβλέψει αποτελέσματα, να καταλήξει σε συμπεράσματα και να προτείνει λύσεις στο πρόβλημα στο οποίο έχει εκπαιδευτεί.

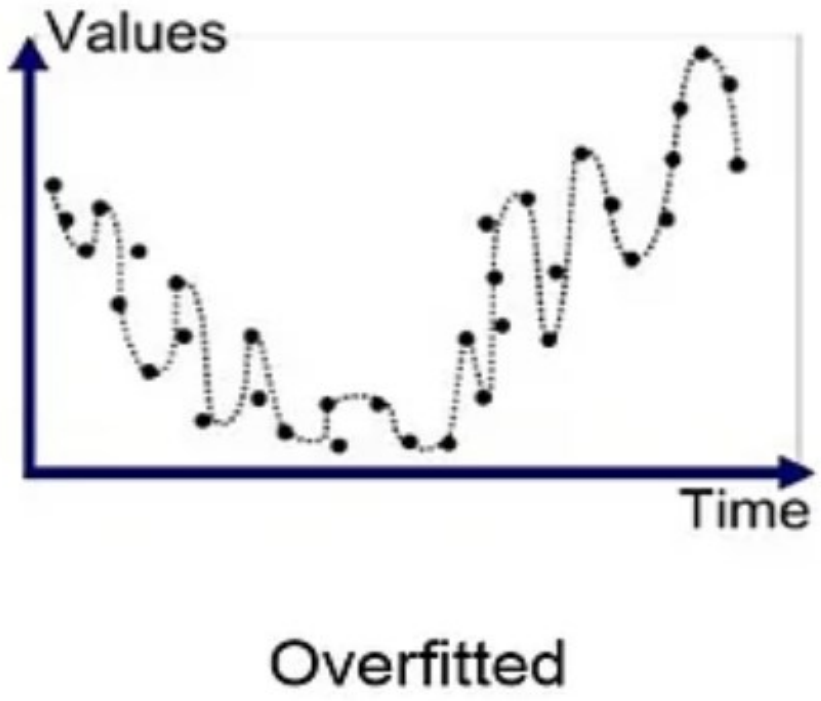
Σημειώνεται ότι με τη χρήση διαφορετικών δεδομένων προς εκπαίδευση, ο ίδιος αλγόριθμος μάθησης μπορεί να παράγει διαφορετικά μοντέλα. Επίσης, όσα περισσότερα δεδομένα παρέχονται στον αλγόριθμο, τόσο περισσότερο αυτός μαθαίνει. Συνεπώς, προκύπτει ότι, η ποσότητα και η ποιότητα των δεδομένων που χρησιμοποιούνται κατά την εκπαίδευση επηρεάζουν την ικανότητα του αλγορίθμου να κατανοεί και να αντιμετωπίζει νέα δεδομένα. Απαιτείται, λοιπόν, σωστή συλλογή και επεξεργασία δεδομένων για όλες τις καταστάσεις που μπορεί να προκύψουν, ώστε ο αλγόριθμος μάθησης να παράγει καλά/ακριβή αποτελέσματα.



Εικόνα 2 - Σχέση μεταξύ AI-ML-DL [19]

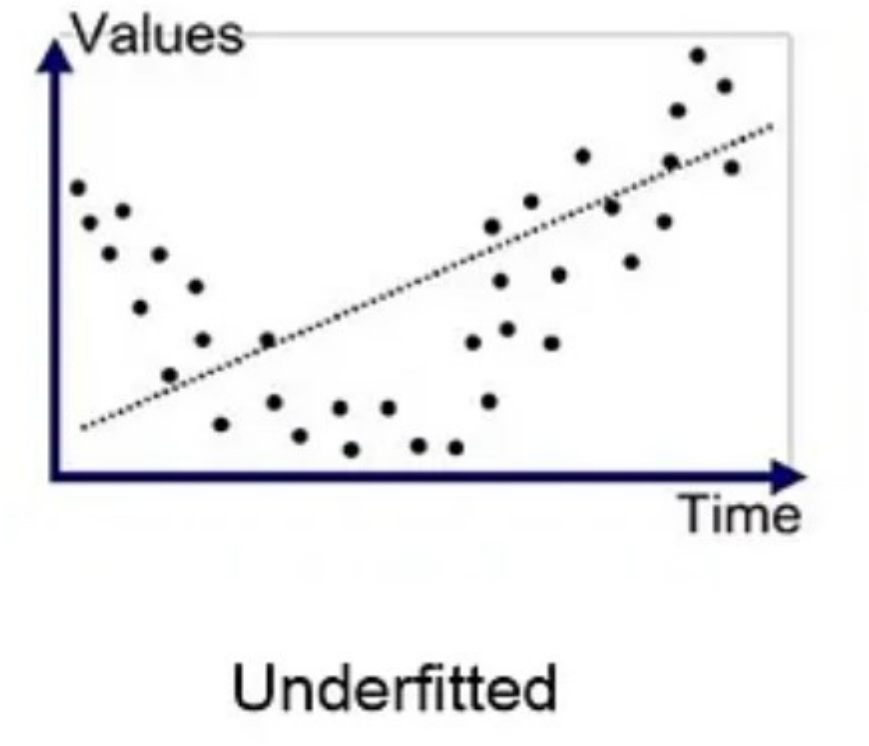
Δύο συνηθισμένα προβλήματα στον τομέα της μηχανικής μάθησης είναι η υπερπροσαρμογή (overfitting) και η υποπροσαρμογή (underfitting), τα οποία συμβάλλουν στην κακή απόδοση του μοντέλου.

Η υπερπροσαρμογή αναφέρεται όταν σε ένα μοντέλο που μαθαίνει τόσο καλά τις λεπτομέρειες και τον θόρυβο στα δεδομένα εκπαίδευσης, που επηρεάζεται αρνητικά η απόδοσή του σε νέα δεδομένα. Το μοντέλο αυτό χαρακτηρίζεται ως περίπλοκο, διαθέτει πολύ καλή απόδοση σε δεδομένα εκπαίδευσης, αλλά αποτυγχάνει στο να προβλέψει ακριβή αποτελέσματα για νέα δεδομένα. Όσο περισσότερο εκπαιδευτεί το μοντέλο, τόσες περισσότερες πιθανότητες υπάρχουν να εμφανιστεί το υπερπροσαρμοσμένο μοντέλο. Το υπερπροσαρμοσμένο μοντέλο έχει χαμηλή προκατάληψη και υψηλή διακύμανση. Όπως παρατηρείται στη παρακάτω εικόνα, το μοντέλο καλύπτει όλα τα σημεία δεδομένων που υπάρχουν στο διάγραμμα διασποράς, συμπεριλαμβανομένων των σημείων που είναι θόρυβος και ακραία σημεία, έχοντας ως στόχο να βρει την καλύτερη προβλεπόμενη γραμμή, το οποίο είναι ανέφικτο σε αυτή τη περίπτωση, άρα προκύπτουν σφάλματα πρόβλεψης.



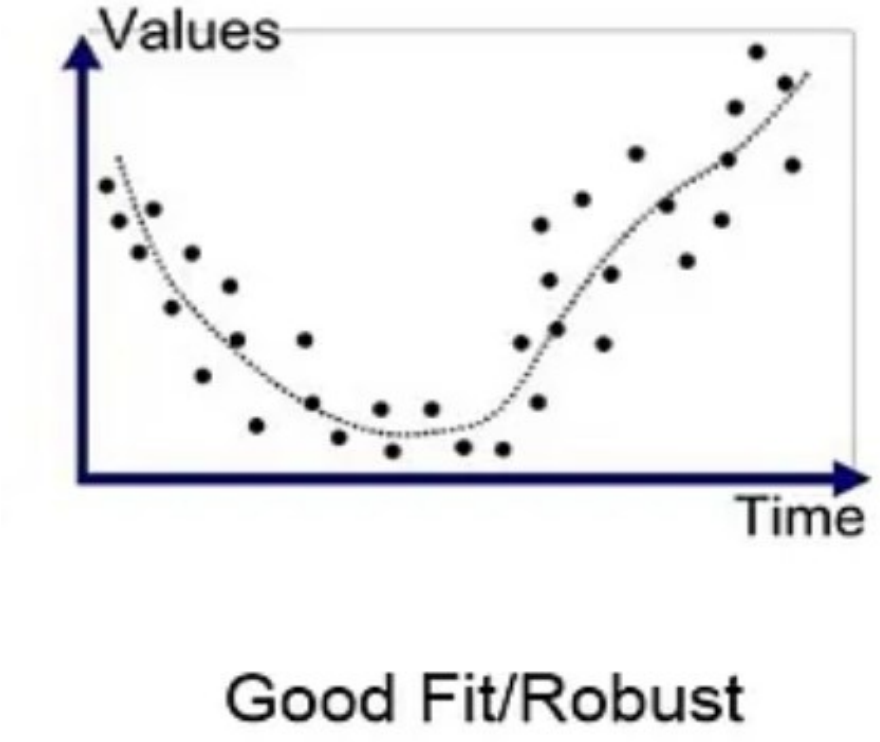
Εικόνα 3 - Υπερπροσαρμογή [20]

Η υποπροσαρμογή συμβαίνει όταν ένα μοντέλο δεν έχει μάθει καλά τα μοτίβα, τις σχέσεις στα δεδομένα εκπαίδευσης και δεν μπορεί να γενικεύσει και άρα να δημιουργήσει προβλέψεις για νέα δεδομένα. Ένα υποπροσαρμοσμένο μοντέλο χαρακτηρίζεται ως απλό και έχει κακή απόδοση τόσο σε δεδομένα εκπαίδευσης όσο και σε νέα. Ένα μη προσαρμοσμένο μοντέλο έχει υψηλή προκατάληψη και χαμηλή διακύμανση. Όπως παρατηρείται στη παρακάτω εικόνα, το μοντέλο αδυνατεί να συλλάβει όλα τα σημεία δεδομένων που υπάρχουν στο διάγραμμα.



Εικόνα 4 - Υποπροσαρμογή [20]

Ο στόχος του μοντέλου μηχανικής μάθησης είναι να επιτύχει καλή εκπαίδευση, με ακρίβεια δοκιμής, να αποφύγει την υπερπροσαρμογή ή την υποπροσαρμογή, έτσι ώστε να αποδίδει καλά, κάνοντας σωστή γενίκευση των δεδομένων. Η καλή απόδοση του μοντέλου επιτυγχάνεται όταν το μοντέλο δεν είναι ούτε πολύ απλό, ώστε να μη μπορεί να περιγράψει τον στόχο, αλλά ούτε και πολύ περίπλοκο, ώστε να μη μπορεί να εκφράσει τον στόχο. Ενα μοντέλο καλής προσαρμογής διαθέτει ισορροπία μεταξύ μεροληψίας και διακύμανσης. Παρατηρώντας τη παρακάτω εικόνα, απεικονίζεται μια αρκετά καλή και προβλεπόμενη γραμμή.



Εικόνα 5 - Καλή προσαρμογή [20]

Διακρίνονται τρεις μορφές μηχανικής μάθησης, οι οποίες αναλύονται παρακάτω:

- η Επιτηρούμενη Μάθηση (ή αλλιώς Επιβλεπόμενη Μάθηση ή Μάθηση με Επίβλεψη) (Supervised Learning), όπου το υπολογιστικό πρόγραμμα δέχεται τα δεδομένα εισόδου (input data), αλλά και τα αναμενόμενα/σωστά αποτελέσματα (output data) από τον "δάσκαλο" (teacher). Ο στόχος είναι να επιτευχθεί η μέγιστη δυνατή ακριβής πρόβλεψη αποτελεσμάτων σε εφαρμογές με άγνωστα στοιχεία (δηλαδή νέα δεδομένα) χρησιμοποιώντας τον κατάλληλο αλγόριθμο πρόβλεψης. Η παρούσα διαδικασία χωρίζεται σε δύο κατηγορίες ανάλογα με τον αλγόριθμο που επιλέγεται να χρησιμοποιηθεί. Αν η πρόβλεψη αφορά συνεχείς μεταβλητές τότε ακολουθείται παλινδρόμηση (regression), ενώ αν αφορά διακριτές μεταβλητές ακολουθείται ταξινόμηση (classification). Στην παλινδρόμηση στόχος είναι να προβλεφθεί μια συνεχής αριθμητική έξοδος, με βάση μία ή περισσότερες εισόδους. Στην ταξινόμηση στόχος είναι τα δεδομένα εξόδου να χωρίζονται σε διαφορετικές κατηγορίες (ετικέτες), με βάση τα χαρακτηριστικά των δεδομένων εισόδου.

- η Μη Επιτηρούμενη Μάθηση (ή αλλιώς Μη Επιβλεπόμενη Μάθηση ή Μάθηση χωρίς Επίβλεψη) (Unsupervised Learning), όπου το υπολογιστικό πρόγραμμα λαμβάνει μόνο τα δεδομένα εισόδου, ενώ το αποτέλεσμα είναι άγνωστο (δηλ. δεν παρέχονται τα σωστά/αναμενόμενα αποτελέσματα). Ο στόχος είναι να αναγνωρίζονται αυτόματα μοτίβα, δομές, ή ομάδες στα δεδομένα. Μία από τις πιο συνηθισμένες εφαρμογές

αυτού του είδους μηχανικής μάθησης είναι η ομαδοποίηση (clustering), όπου το σύστημα δημιουργεί συστάδες (clusters) από τα δεδομένα εισόδου, ομαδοποιώντας με αυτό τον τρόπο τα δεδομένα που έχουν κοινά στοιχεία.

- η Ενισχυτική Μάθηση (Reinforcement Learning), όπου το υπολογιστικό πρόγραμμα αλληλοεπιδρά με το περιβάλλον του, χωρίς τη μεσολάβηση κάποιου “δασκάλου” (χωρίς ανθρώπινη καθοδήγηση), έτσι ώστε να επιτευχθεί ένας συγκεκριμένος στόχος. Διάφορα λογισμικά και μηχανήματα τη χρησιμοποιούν, με σκοπό να βρεθεί η καλύτερη δυνατή συμπεριφορά ή διαδρομή που πρέπει να ακολουθηθεί σε μια συγκεκριμένη κατάσταση. Ο αλγόριθμος ενισχυτικής μάθησης μαθαίνει από τα αποτελέσματα και αποφασίζει ποια ενέργεια έπεται να εκτελεστεί. Μετά από κάθε ενέργεια, ο αλγόριθμος ανατροφοδοτείται και προσδιορίζει εάν η επιλογή που ακολούθησε ήταν σωστή, ουδέτερη ή εσφαλμένη. Αποτελεί καλή τεχνική για αυτοματοποιημένα συστήματα που καλούνται να λαμβάνουν πολλές μικρές αποφάσεις.

2.1.5.2.1 Βαθιά μάθηση (Deep Learning)

Είναι ένα υποσύνολο της μηχανικής μάθησης (όπως φαίνεται στην Εικόνα 1), το οποίο χρησιμοποιεί τεχνητά νευρωνικά δίκτυα για την εκμάθηση και την ανάλυση δεδομένων. Η τεχνολογία αυτή βασίζεται στον τρόπο λειτουργίας του ανθρώπινου εγκεφάλου, μιμώντας το πρότυπο της επεξεργασίας των οπτικών πληροφοριών.

Στη βαθιά μάθηση, τα μοντέλα τεχνητών νευρώνων έχουν πολλαπλά επίπεδα (γεγονός που δίνει τον όρο "βαθιά"), τα οποία συνεργάζονται με σκοπό την επεξεργασία πολύπλοκων δεδομένων. Το σύστημα μαθαίνει να αναπαριστά διαφορετικά χαρακτηριστικά των δεδομένων, με σκοπό να επιτύχει την κατανόηση και επίλυση του προβλήματος. Η εκπαίδευση βαθιών νευρωνικών δικτύων απαιτεί μεγάλο όγκο δεδομένων και υπολογιστικών πόρων, αλλά επιτυγχάνει μεγαλύτερη ακρίβεια πρόβλεψης συγκριτικά με την “απλή” μηχανική μάθηση. Πιο συγκεκριμένα, έχει καταφέρει εντυπωσιακά αποτελέσματα σε πολλούς τομείς (εφαρμογών), όπως η αναγνώριση εικόνας, η φωνητική αναγνώριση, η ανάλυση κειμένου, η μετάφραση γλωσσών και άλλα.

2.1.5.3 Επεξεργασία Φυσικής Γλώσσας (Natural Language Processing)

Η επεξεργασία φυσικής γλώσσας ασχολείται με τον τρόπο αλληλεπίδρασης των υπολογιστών με την ανθρώπινη γλώσσα. Στοχεύει στο να επιτρέψει στις μηχανές να κατανοούν, να αναλύουν και να επεξεργάζονται μεγάλες ποσότητες φυσικής γλώσσας. Η τεχνολογία αυτή μπορεί να χρησιμοποιηθεί για αναγνώριση ομιλίας, ανάλυση και κατανόηση λόγου, συντακτική ανάλυση, εξαγωγή πληροφοριών, αυτόματη μετάφραση κειμένου, αναζήτηση με λέξεις-κλειδιά, παραγωγή φυσικής γλώσσας και πολλά άλλα.

Σύγχρονα εργαλεία επεξεργασίας φυσικής γλώσσας αποτελούν τα chatbot, τα οποία είναι εφαρμογές λογισμικού που προσομοιώνουν και επεξεργάζονται ανθρώπινη συνομιλία γραπτή ή προφορική, επιτρέποντας την αλληλεπίδραση του ανθρώπου με ψηφιακές συσκευές. Χρησιμοποιώντας κυρίως κανόνες NLP (Natural Language Processing) αλλά και λίγο ML (Machine learning), δημιουργούν μια μορφή συνομιλίας δίνοντας αυτοματοποιημένες απαντήσεις σε ερωτήματα που θέτουν οι χρήστες. Η χρήση τους σκοπεύει στην αυτοεξυπηρέτηση των χρηστών χωρίς να απαιτείται ανθρώπινη παρέμβαση, στην αυτοματοποίηση και στην γρήγορη εκτέλεση διαδικασιών.

Το πιο ευρέως γνωστο chatbot είναι το ChatGPT της OpenAI, το οποίο καθοδηγείται από τεχνολογία AI και χρησιμοποιεί την αρχιτεκτονική GPT (Generative Pre-trained Transformer - Γενετικός Προ-εκπαιδευμένος Μετασχηματιστής). Το μοντέλο εκπαιδεύεται αντλώντας δεδομένα από το διαδίκτυο, επανεκπαιδεύεται και εξελίσσεται με τη χρήση αλλά και με νέα δεδομένα που εισάγονται συνεχώς στο σύστημα και παράγει αποτελέσματα μέσω απαντήσεων σε συνομιλία με τον χρήστη. Διαθέτει πολλές χρήσεις, έχοντας ως πιο συνήθεις την απάντηση σε ερωτήσεις, την δημιουργία, την μετάφραση, τη σύνοψη κειμένου, τη συγγραφή και την διόρθωση προγραμμάτων υπολογιστών σε διάφορες γλώσσες προγραμματισμού, όπως Python, Java, PHP, Ruby και πολλές άλλες.

2.1.5.4 Ρομποτική (Robotics)

Η ρομποτική είναι κλάδος της μηχανοηλεκτρικής επιστήμης και αναφέρεται στη μελέτη, τη σχεδίαση, την κατασκευή, τον έλεγχο και την λειτουργία μηχανών που έχουν την ικανότητα να εκτελούν εργασίες με αυτονομία ή με ελάχιστη ανθρώπινη παρέμβαση. Αυτές οι μηχανές αποκαλούνται ρομπότ και έχουν ως στόχο την αντικατάσταση του ανθρώπου σε διάφορες εργασίες, τόσο σε φυσικό όσο και σε επίπεδο λήψης αποφάσεων. <<Το Ρομπότ είναι μία κατασκευή, που μπορεί να εκτελεί προγραμματισμένες εργασίες, αλλά σε αντίθεση με τις απλές μηχανές ακολουθεί τη μέθοδο: αντιλαμβάνομαι, σκέπτομαι, ενεργώ.>> **[39]**. Η λέξη ρομπότ προέρχεται από το σλαβικό robota που σημαίνει εργασία. Καθιερώθηκε ως όρος με την σημερινή του έννοια το 1920 από τον Τσέχο θεατρικό συγγραφέα Karel Čapek στο έργο του "R.U.R." (Rossum's Universal Robots), όπου σατιρίζει την εξάρτηση της κοινωνίας από τους μηχανικούς εργάτες (ρομπότ) της τεχνολογικής εξέλιξης και που τελικά εξοντώνουν τους δημιουργούς τους. Σε πολλές σύγχρονες σλαβικές γλώσσες, όπως στην πολωνική, χρησιμοποιείται σαν έκφραση της καθημερινότητας με την έννοια της σκληρής δουλειάς **[40]**.

Ένα ρομπότ αποτελείται από τρεις συνιστώσες:

1) Ένα μηχανολογικό σύστημα, το οποίο αποτελεί το σώμα και τους μηχανισμούς που επιτρέπουν στο ρομπότ να κινείται στο χώρο. Αυτά τα μηχανικά στοιχεία μπορεί να είναι κινητήρες, ρόδες, αρθρώσεις, όπως πόδια, έλικες, και άλλα.

2) Ένα σύστημα αισθητήρων, με το οποίο το ρομπότ ανιχνεύει στοιχεία του περιβάλλοντος, συγκεντρώνει πληροφορίες για την κατάσταση στην οποία βρίσκεται, επικοινωνεί και αλληλοεπιδρά με το εξωτερικό περιβάλλον. Το σύστημα αυτό μπορεί να

περιλαμβάνει αισθητήρες όρασης, αφής, ήχου, θερμοκρασίας, με τους οποίους καταφέρνει να παρατηρεί το περιβάλλον γύρω του και να αντιδρά ανάλογα με τις συνθήκες.

3) Ένα σύστημα ελέγχου, το οποίο συντονίζει την αίσθηση, την κίνηση και καθοδηγεί το ρομπότ έτσι ώστε να λειτουργεί αποτελεσματικά. Πιο συγκεκριμένα, λαμβάνει αποφάσεις σύμφωνα με τα δεδομένα που έχουν συγκεντρωθεί από τους αισθητήρες, στέλνει μηνύματα στους κινητήρες και ενεργεί ανάλογα.

Οι γενιές ρομπότ μπορούν να διαχωριστούν ανάλογα με το επίπεδο τεχνολογίας που διαθέτουν.

- Στη πρώτη γενιά, ανήκουν τα ρομπότ που αναπτύχθηκαν στα μέσα του 20ου αιώνα και αποκαλούνταν “ρομπότ χειρισμού” με βάση τη βιβλιογραφία. Τα ρομπότ αυτά, ήταν μεγάλου όγκου μηχανές, οι οποίες χρησιμοποιήθηκαν κυρίως σε βιομηχανικά περιβάλλοντα και δεν είχαν δυνατότητα υπολογισμού και αίσθησης. Το σύστημα τους είχε τη δυνατότητα μόνο να εκτελέσει προγραμματισμένες επαναλαμβανόμενες εργασίες που εκτελούνταν χειροκίνητα ή με σταθερό διαδοχικό τρόπο.

- Η δεύτερη γενιά ρομπότ εμφανίστηκε κατά τη δεκαετία του 1980, όπου τα ρομπότ αποκαλούνταν ρομπότ εκμάθησης, αφού μπορούσαν να αυτοβελτιώνονται με βάση τα λάθη και να επαναπρογραμματιστούν. Ωστόσο, στερούνταν ανθρώπινη νοημοσύνη και διέθεταν περιορισμένη υπολογιστική ισχύ.

- Η τρίτη γενιά αφορά ρομπότ με νοημοσύνη που μπορούν να λειτουργήσουν σε μεγάλο βαθμό χωρίς επίβλεψη από κάποιον άνθρωπο ή εξωτερικό υπολογιστή. Διαθέτουν ικανότητες αντίληψης, κίνησης, είναι ικανά να λαμβάνουν αποφάσεις και να αντιμετωπίζουν προβλήματα κατά τη διάρκεια της εργασίας τους.

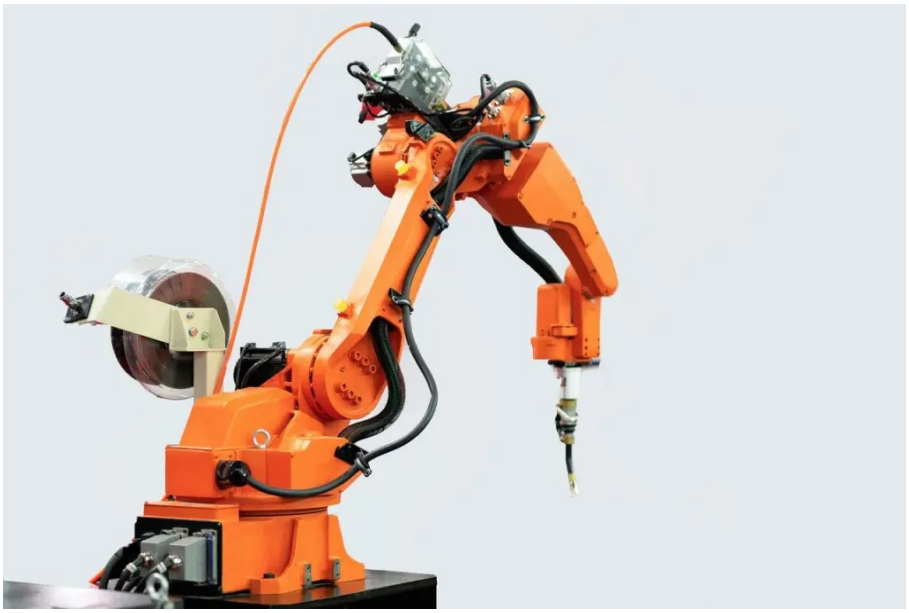
- Η τέταρτη γενιά, η οποία είναι η σημερινή, πραγματοποιεί χρήση προηγμένων τεχνολογιών, όπως η τεχνητή νοημοσύνη, η μηχανική μάθηση, η υπολογιστική όραση και άλλες. Τα σημερινά ρομπότ λειτουργούν πιο αυτόνομα, είναι πιο ευέλικτα, διαθέτουν προηγμένες ικανότητες χάρη στους εξελιγμένους αισθητήρες και επεξεργαστές που χρησιμοποιούν. Μπορούν να εκτελέσουν σύνθετες εργασίες χωρίς ανθρώπινη παρέμβαση και έχουν την δυνατότητα να επαναπρογραμματιστούν για κάθε νέα εργασία συγκριτικά με παλαιότερες γενιές ρομπότ που έπρεπε να διαμορφωθούν εκ νέου για κάθε νέα εργασία. Έτσι, τα μηχανήματα αυτής της γενιάς μπορούν να εκπαιδευτούν ώστε να εκτελούν πολλές διαφορετικές εργασίες ακόμα και ταυτόχρονα, διαθέτοντας επομένως το χαρακτηριστικό multitasking.

2.1.5.4.1 Είδη ρομπότ και χρησιμότητα

Η δημιουργία των ρομπότ είχε σκοπό την αύξηση απόδοσης και παραγωγικότητας σε διάφορους τομείς, καθώς μία κατάλληλα εκπαιδευμένη μηχανή, όπως ένα ρομπότ, μπορεί να προσφέρει εργασία με μεγάλη ακρίβεια, ταχύτητα και αποτελεσματικότητα. Έτσι, γίνεται εξοικονόμηση χρόνου και ενέργειας, αφού τα ρομπότ μπορούν να εργάζονται αδιάλειπτα και αποτελούν φθηνά εργατικά χέρια. Παράλληλα, επιτυγχάνεται η μείωση του ανθρώπινου κινδύνου κατά την εκτέλεση επικίνδυνων εργασιών σε επιβλαβή για την υγεία και ασφάλεια περιβάλλοντα, όπως σε εργοστάσια χημικών ουσιών ή σε περιοχές καταστροφών. Επιπλέον, είναι ευρέως γνωστή και η χρησιμότητα τους στον τομέα της υγείας, αφού διαθέτουν υψηλή ακρίβεια σε ιατρικές εφαρμογές, διαθέτοντας εξειδικευμένες δεξιότητες.

Κάνοντας μια μικρή ανάλυση ως προς τα είδη ρομπότ, τα πιο βασικά που αξίζει να αναφερθούν είναι τα βιομηχανικά (industrial robots), τα κινούμενα (mobile robots), τα τηλεχειριζόμενα (remote-controlled robots) και τα ανθρωποειδή (humanoid robots).

Τα βιομηχανικά ρομπότ αποτελούνται από έναν πολλαπλό βραχίονα που διαθέτει τρεις ή περισσότερους άξονες, ο οποίος είναι ελεγχόμενος και επαναπρογραμματιζόμενος. Υπάρχουν διάφορα είδη βραχιόνων ανάλογα με τη χρήση. Για παράδειγμα, ένας ρομποτικός βραχίονας μπορεί να εξοπλιστεί με εξειδικευμένες λαβές, οι οποίες μπορούν να χειριστούν με ευαισθησία εύθραυστα αντικείμενα, ενώ κάποιο άλλο είδος ρομποτικού βραχίονα μπορεί να διαθέτει λαβές που να έχουν την ικανότητα να πιάσουν και να ανυψώσουν φορτία βάρους πολλών τόνων. Ο ρομποτικός βραχίονας ακόμη, μπορεί να έχει τη δυνατότητα να εξοπλιστεί με ένα σύνολο αισθητήρων και ένα σύστημα όρασης, για να μπορεί να αναγνωρίζει αντικείμενα και εικόνες, αν το απαιτεί η χρήση. Τα βιομηχανικά ρομπότ εξασφαλίζουν διάφορες χρήσεις, όπως η συναρμολόγηση, η τοποθέτηση, η συγκόλληση, ο έλεγχος, ο διαχωρισμός προϊόντων και άλλα.



Εικόνα 6 - Βιομηχανικό Ρομπότ [47]

Ως κινούμενα ρομπότ χαρακτηρίζονται τα ρομπότ που ο μηχανισμός που διαθέτουν τους επιτρέπει τη δυνατότητα κίνησης. Ανάλογα με τον τρόπο κίνησης και τον χώρο εργασίας τους, προκύπτουν τα παρακάτω υπο-είδη:

- Το αυτοματοποιημένο όχημα καθοδήγησης (Automated Guided Vehicle - AGV), το οποίο είναι ένα φορητό ρομπότ με περιορισμένη αυτονομία κίνησης αφού ακολουθεί σημαδεμένες μακριές γραμμές ή καλώδια στο έδαφος. Χρησιμοποιείται για μεταφορά βαρέων υλικών γύρω από βιομηχανικά κτίρια.

Εικόνα 7 - AGV [48]

- Τα έντροχα ρομπότ πλοηγούνται στο έδαφος χρησιμοποιώντας μηχανοκίνητους τροχούς για να κινηθούν. Είναι πιο απλοϊκός ο σχεδιασμός τους και αποτελούν πιο οικονομική κατασκευή. Ένα παράδειγμα τέτοιου ρομπότ είναι η ηλεκτρική σκούπα Roomba που μετακινείται με τροχούς στο χώρο ενώ διαθέτει ενσωματωμένους αισθητήρες για χαρτογράφηση του χώρου και αποφυγή εμποδίων.

Εικόνα 8 - Roomba [49]

- Τα μη επανδρωμένα εναέρια οχήματα (Unmanned aerial Vehicles - UAVs), τα οποία αποτελούν είδη ελικοπτέρων και αεροπλάνων χωρίς επιβάτες ή πιλότο. Αποκαλούνται κοινώς ως drones και χρησιμοποιούνται σε πολλές εφαρμογές, όπως αεροφωτογράφηση, παρακολούθηση πυρκαγιών, επιθεωρήσεις υποδομών αλλά ακόμη και για ψυχαγωγικούς σκοπούς.

Εικόνα 9 - UAV [50]

- Τα αυτόνομα υποβρύχια οχήματα (Autonomous Underwater Vehicles - AUVs), τα οποία είναι μη επανδρωμένα υποβρύχια ρομπότ που ελέγχονται από την επιφάνεια, από κάποιον χειριστή μέσω τηλεχειριστηρίου. Έχουν σχήμα τορπιλών, κινούνται με μεγάλη ταχύτητα και η τροφοδοσία τους γίνεται μέσω ειδικών μπαταριών.

Εικόνα 10 - AUV [51]

- Τα ανθρωποειδή ρομπότ, τα οποία μιμούνται το ανθρώπινο σώμα, έχουν ανθρώπινα χαρακτηριστικά, όπως κεφάλι, κορμό, δύο χέρια και δύο πόδια με τα οποία κινούνται. Χαρακτηρίζονται ως ρομπότ εξυπηρέτησης, διαθέτουν ανθρώπινη συμπεριφορά και αλληλεπίδραση με το περιβάλλον. Χρησιμοποιούνται για ερευνητικούς σκοπούς και για μελέτη της ανθρώπινης δομής και συμπεριφοράς. Σημαντική είναι η συνεισφορά τους στο τομέα της ιατρικής και βιοτεχνολογίας καθώς η μελέτη των μελών τους αποτελεί έναυσμα για την ανάπτυξη σύνθετων προσθετικών για άτομα με σωματικές και κινητικές αναπηρίες. Γενικότερα, μπορούν να αναλάβουν διάφορους ρόλους σε ένα ευρύ φάσμα εφαρμογών, εκτελώντας εργασίες ρουτίνας ή ακόμα και επικίνδυνες αποστολές.

Εικόνα 11 - Robot Apollo [53]

2.2 Τεχνολογία Λογισμικού

Στην δεύτερη ενότητα του παρόντος κεφαλαίου παρουσιάζεται αναλυτικά η έννοια της "Τεχνολογίας Λογισμικού". Περιγράφεται η φιλοσοφία της, ποιους υποτομείς/περιοχές περιλαμβάνει, ποια είναι η κύρια διαδικασία ανάπτυξης λογισμικού που ακολουθείται και οι δραστηριότητές της καθώς και ποιες είναι οι κυριότερες μεθοδολογίες ανάπτυξης που περιλαμβάνει.

2.2.1 Ορισμός Λογισμικού

Λογισμικό (software) ορίζεται το σύνολο εντολών, προγραμμάτων και δεδομένων που καθορίζει τη λειτουργία ενός υπολογιστή δίνοντας οδηγίες εκτέλεσης εργασιών. Ουσιαστικά, το λογισμικό καθοδηγεί τον υπολογιστή στο πώς να λειτουργεί. Ο όρος δημιουργήθηκε για τη διαφοροποίηση αυτών των οδηγιών από το υλικό (hardware), δηλαδή τα φυσικά στοιχεία ενός συστήματος υπολογιστή, τα οποία εκτελούν τις εργασίες **[57]**. Το λογισμικό είναι γραμμένο σε κάποια γλώσσα προγραμματισμού υψηλού επιπέδου (οι πιο δημοφιλείς γλώσσες είναι η C, C++, Java, Python και άλλες), η οποία μεταφράζεται σε κώδικα μηχανής χαμηλού επιπέδου χρησιμοποιώντας έναν μεταγλωττιστή ή διερμηνεύεται από έναν διερμηνέα για να μπορεί ο υπολογιστής να την κατανοήσει. Επίσης, μπορεί να είναι γραμμένο σε γλώσσα assembly χαμηλού επιπέδου, η οποία μεταφράζεται στη γλώσσα μηχανής με τη χρήση ενός assembler.

2.2.2 Τύποι Λογισμικού

Υπάρχουν διάφοροι τύποι λογισμικού ανάλογα με τη λειτουργία και τη χρησιμότητα τους. Παρακάτω αναλύονται οι κύριοι τύποι:

- το Λογισμικό Συστήματος (System Software): διαχειρίζεται την εσωτερική λειτουργία του υλικού παρέχοντας βασικές λειτουργίες. Ελέγχει την αλληλεπίδραση μεταξύ του χρήστη και του υλικού, ενώ δεν χρησιμοποιείται από τους τελικούς χρήστες, καθώς λειτουργεί στο παρασκήνιο ως μεσολαβητής. Αυτός ο τύπος λογισμικού περιλαμβάνει τα παρακάτω:

  - Τα Λειτουργικά Συστήματα (Operating Systems): είναι συλλογές λογισμικού που διαχειρίζονται πόρους υλικού και λογισμικού ενώ προσφέρουν γενικές υπηρεσίες υπολογιστή, όπως διαχείριση χρηστών. Καθώς το σύστημα του υπολογιστή ενεργοποιείται, το πρώτο που φορτώνεται στη μνήμη είναι το λειτουργικό σύστημα. Υπάρχουν διάφοροι τύποι λειτουργικών συστημάτων, όπως Windows, iOS, Unix, Linux και Ubuntu και άλλοι.

  - Τα Προγράμματα Οδήγησης Συσκευών (Device Drivers): ελέγχουν τις συσκευές υλικού που είναι συνδεδεμένες στον υπολογιστή, όπως ένα πληκτρολόγιο, εκτυπωτής, ποντίκι. Με τη σύνδεση μιας νέας συσκευής στο

σύστημα υπολογιστή απαιτείται η εγκατάσταση του προγράμματος οδήγησης για την διαχείριση αυτής της συσκευής.

- ➢ Το Βοηθητικό Λογισμικό (Utility software): περιλαμβάνει προγράμματα υπολογιστών που αφορούν τη διαμόρφωση, τη συντήρηση και τη βελτιστοποίηση ενός υπολογιστή ή των μερών του.

- ❖ το Λογισμικό Εφαρμογής (Application Software): είναι προγράμματα που εξυπηρετούν τους τελικούς χρήστες εκτελώντας εργασίες και ειδικές λειτουργίες που ζητήθηκαν, ικανοποιεί δηλαδή τις απαιτήσεις των χρηστών. Κάποιοι από τους βασικούς τύπους λογισμικού εφαρμογών είναι:

  - ➢ Επεξεργαστές κειμένου (editors): είναι εφαρμογές που χρησιμοποιούνται για συγγραφή περιεχομένου, τεκμηρίωση και επεξεργασία. Αυτές οι εφαρμογές προσφέρουν την δυνατότητα της αποθήκευσης, μορφοποίησης και εκτύπωσης δεδομένων. Διάσημα προγράμματα επεξεργασίες κειμένου είναι το Microsoft Word, το Google Docs και άλλα.

  - ➢ Προγράμματα περιήγησης Ιστού (web browsers): είναι λογισμικό που προσφέρει πρόσβαση και προβολή ιστοσελίδων. Τα πιο γνωστά παραδείγματα είναι το Google Chrome και το Internet Explorer.

  - ➢ Λογισμικό βάσης δεδομένων: αποτελεί το σύστημα διαχείρισης βάσεων δεδομένων (Database Management system - DBMS), το οποίο χρησιμοποιείται για την δημιουργία και διαχείριση μιας βάσης δεδομένων με σκοπό την ανάκτηση και την επεξεργασία δεδομένων. Τα πιο ευρέως χρησιμοποιούμενα συστήματα διαχείρισης βάσεων δεδομένων είναι τα MySQL, Oracle, Microsoft SQL Server και άλλα.

- ❖ το Λογισμικό Προγραμματισμού (Programming Software): δεν χρησιμοποιείται από τους τελικούς χρήστες αλλά από προγραμματιστές για τη συγγραφή, κατασκευή, δοκιμή, και συντήρηση του πηγαίου κώδικα ενός λογισμικού/εφαρμογής. Περιλαμβάνει εργαλεία, βιβλιοθήκες και πόρους που χρησιμοποιούνται για τη δημιουργία εφαρμογών και προγραμμάτων. Το λογισμικό προγραμματισμού λειτουργεί και ως μεταφραστής, δηλαδή παίρνει μια γλώσσα υψηλού επιπέδου, όπως είναι η Java, και τη μεταφράζει σε γλώσσα μηχανής.

2.2.3 Ιδιότητες Καλού Λογισμικού

Με τη χρήση βέλτιστων πρακτικών, ελέγχων και διαδικασιών ένα προϊόν λογισμικού εξετάζεται ως προς την ορθότητα, τη συντηρησιμότητα και την απόδοσή του.

Η ορθότητα του λογισμικού αναφέρεται στην ικανότητα του να λειτουργεί με ακρίβεια και αξιοπιστία χωρίς σφάλματα κατά την εκτέλεση. Η πρόληψη σφαλμάτων μπορεί να

επιτευχθεί με διάφορες πρακτικές κατά το στάδιο της ανάπτυξης, παράγοντας σωστά αποτελέσματα. Το πιο σημαντικό και αποτελεσματικό εργαλείο για τη διασφάλιση της ορθότητας είναι οι δοκιμές. Μέσω μετρικών κάλυψης (coverage metrics) παρέχονται πληροφορίες σχετικά με το ποσοστό του κώδικα που ελέγχθηκε κατά τη διάρκεια των δοκιμών, ώστε να επαληθευθεί η σωστή λειτουργία του λογισμικού στα περισσότερα μέρη του. Αυτές οι μετρήσεις μπορούν να ληφθούν από διάφορους τύπους δοκιμών, με τους πιο ευρέως γνωστούς να είναι οι δοκιμές μονάδας (Unit Testing), οι δοκιμές ολοκλήρωσης (Integration Testing), και οι δοκιμές απόδοσης (Performance Testing).

Η συντηρησιμότητα του λογισμικού σχετίζεται με την ευκολία αντιμετώπισης σφαλμάτων, την ικανότητα τροποποιήσεων του λογισμικού και την υψηλή επεκτασιμότητα του. Κατά τη περίοδο ανάπτυξης η συγγραφή κώδικα χρειάζεται να γίνει με τρόπο ευανάγνωστο, να διαθέτει απλότητα και να έχει τη δυνατότητα επέκτασης. Χρήσιμα εργαλεία αποτελούν η κατάλληλη σχεδίαση, οργάνωση του κώδικα σε μικρά ανεξάρτητα μέρη, τα οποία να έχουν την δυνατότητα διάσπασης, η χρήση στοχευμένων σχολίων επεξήγησης, καθώς και η χρήση σχεδιαστικών μοτίβων (design patterns) για μεγαλύτερη ευελιξία. Αυτές οι πρακτικές συμβάλλουν στην ταχύτερη διαχείριση και αντιμετώπιση προβλημάτων κώδικα, επιτυγχάνοντας εξοικονόμηση χρόνου.

Η απόδοση του λογισμικού χαρακτηρίζει την ποιότητα και την σωστή λειτουργικότητα του. Κατά την αξιολόγηση του λογισμικού όσον αφορά την απόδοση, ελέγχεται αν μπορεί να ανταποκριθεί στον απαιτούμενο χρόνο απόκρισης χωρίς καθυστερήσεις καθώς και στη σωστή διαχείριση και χρήση πόρων. Η βελτίωση της απόδοσης μπορεί να επιτευχθεί με βελτιστοποιήσεις στον κώδικα, στην αλγοριθμική σχεδίαση και με συχνές δοκιμές εκτέλεσης. Ο έλεγχος της απόδοσης μπορεί να γίνει μέσω εργαλείων παρακολούθησης, κάνοντας καταγραφή των μετρήσεων κατανάλωσης πόρων, της ταχύτητας απόκρισης και της αντοχής του συστήματος (π.χ. σε μεγάλο φόρτο εργασίας).

2.2.4 Ορισμός Τεχνολογίας Λογισμικού

Η τεχνολογία λογισμικού (πιο ευρέως χρησιμοποιούμενος όρος για τη μετάφραση του software engineering) ή αλλιώς μηχανική λογισμικού (σωστά αποδομένος όρος), είναι μία συστηματική, πειθαρχημένη και ποσοτικοποιήσιμη προσέγγιση για την κάλυψη όλων των δραστηριοτήτων και διαδικασιών (όπως σχεδίασης, ανάπτυξης, δοκιμής και συντήρησης), που αφορούν την παραγωγή του λογισμικού του λογισμικού. Είναι ένας κλάδος της επιστήμης και μηχανικής των υπολογιστών, ο οποίος δεν περιλαμβάνει μόνο τη τεχνική διαδικασία της ανάπτυξης λογισμικού αλλά και τη διαχείριση του έργου του λογισμικού καθώς και την ανάπτυξη εργαλείων και μεθόδων για τη υποστήριξη της παραγωγής του. Ο κλάδος αυτός εφαρμόζει αρχές, μεθοδολογίες και καλές πρακτικές σε όλο το κύκλο ζωής του λογισμικού με σκοπό να παραχθεί ένα αξιόπιστο, ποιοτικό και κλιμακώσιμο λογισμικό που ικανοποιεί τις ανάγκες των χρηστών και ενδιαφερόμενων μερών του (stakeholders).

2.2.5 Ιστορική Εξέλιξη

- Ο όρος «λογισμικό» με τη σημερινή έννοια χρησιμοποιήθηκε για πρώτη φορά από τον John W. Tukey το 1958. **[60]**

- Η μηχανική λογισμικού υποκινήθηκε μέσω της κρίσης λογισμικού κατά τη δεκαετία του 1960, η οποία χαρακτηρίζεται ως μια περίοδος προβλημάτων όσον αφορά την ανάπτυξη λογισμικού. Όσο τα συστήματα λογισμικού εξελίσσονταν, γινόντουσαν όλο και πιο σύνθετα, με αποτέλεσμα τα έργα ανάπτυξης να καθυστερούν, το κόστος να αυξάνεται και το λογισμικό της εποχής να κρίνεται αναξιόπιστο. Ουσιαστικά, η εξέλιξη του λογισμικού προσπαθούσε να συμβαδίσει με την ραγδαία εξέλιξη του υλικού **[61].**

- Το 1962 κυκλοφόρησε η πρώτη αντικειμενοστραφής γλώσσα προγραμματισμού με όνομα Simula, για τη προσομοίωση συστημάτων.

- Το 1968 έγινε το πρώτο συνέδριο μηχανικής λογισμικού του ΝΑΤΟ που είχε ως σκοπό την αντιμετώπιση ζητημάτων της κρίσης λογισμικού, ορίζοντας κατάλληλες πρακτικές ανάπτυξης που έπρεπε να ακολουθηθούν. **[62]**

- Κατά τη δεκαετία του 1970 επήλθε μια άνοδος, καθώς εμφανίστηκαν λύσεις διαχείρισης της ανάπτυξης λογισμικού.

- Το 1970 δημιουργήθηκε η γλώσσα προγραμματισμού C, από τον Dennis Ritchie, η οποία αποδείχθηκε βέλτιστη επιλογή για πολύπλοκα λογισμικά χειρισμού δεδομένων, προσφέροντας ευελιξία και αποδοτικότητα. Η γλώσσα C αναπτύχθηκε αρχικά στα εργαστήρια Bell από τον Ritchie μεταξύ 1972 και 1973 για την κατασκευή βοηθητικών προγραμμάτων που εκτελούνται σε Unix **[63]**.

- Στη δεκαετία του 1980 η κρίση λογισμικού τελειώνει και η ανάπτυξη του αντικειμενοστραφούς προγραμματισμού αλλάζει τα δεδομένα, συμβάλλοντας στη καλύτερη οργάνωση και διαχείριση πολύπλοκων συστημάτων. Παράλληλα, αναπτύχθηκαν τα πρώτα συστήματα διαχείρισης βάσεων δεδομένων, όπως Oracle & Microsoft SQL Server, για να επιτευχθεί καλύτερη διαχείριση των δεδομένων.

- Το 1980 σχεδιάστηκε η γλώσσα προγραμματισμού Ada, από τον Jean Ichbiah, παρέχοντας δυνατότητες επεξεργασίας σε πραγματικό χρόνο **[64]**.

- Το 1983 βγήκε σε κυκλοφορία ο πρώτος προσωπικός υπολογιστής μαζικής αγοράς με γραφικό περιβάλλον χρήστη **[65]**.

- Το 1985 κυκλοφόρησε η γλώσσα προγραμματισμού C++, με δημιουργό τον Stroustrup, ο οποίος πρόσθεσε νέες (ουσιαστικά αντικειμενοστρεφής) δυνατότητες στη γλώσσα C, δημιουργώντας έτσι μία νέα (αντικειμενοστρεφής) μορφή της με όνομα C++ **[66]**.

- Τη δεκαετία του 1990 με τη χρήση του διαδικτύου επήλθε ταχεία επέκταση των εφαρμογών λογισμικού και ο αντικειμενοστραφής προγραμματισμός άρχισε να αποκτά μεγαλύτερη δημοτικότητα.

- Το 1991 αναπτύχθηκε ο παγκόσμιος ιστός (World Wide Web) από τον Tim Berners-Lee, θέτοντας τα θεμέλια του σημερινού διαδικτύου. Παράλληλα, δημιουργήθηκαν οι πρώτες ιστοσελίδες, το Πρωτόκολλο Μεταφοράς Υπερκειμένου (HyperText Transfer Protocol - HTTP), και η γλώσσα σήμανσης HTML **[67]**. Στο ίδιο έτος ο Linus Torvalds ανέπτυξε ένα νέο, ελεύθερου πυρήνα λειτουργικό σύστημα (Linux) **[68]**.

- Το 1994 ο Andreessen ίδρυσε τη Netscape, δημοσιεύοντας το Netscape Navigator, το πρώτο πρόγραμμα περιήγησης (ιστού) για το ευρύ κοινό **[69]**.

- Το 1995 κυκλοφόρησε η αντικειμενοστραφής γλώσσα προγραμματισμού Java, από τον James Gosling στη Sun Microsystems, παρέχοντας υψηλές δυνατότητες, ακολουθώντας ανοδική πορεία μέχρι και σήμερα **[70]**.

- Η δεκαετία του 2000 επέφερε σημαντική πρόοδο, όπου επήλθε εξέλιξη των δυνατοτήτων λογισμικού, επιτρέποντας την εμφάνιση νέων τεχνολογιών και έξυπνων εφαρμογών.

- Τα μέσα κοινωνικής δικτύωσης αναπτύχθηκαν σε αυτή τη δεκαετία, επηρεάζοντας τη καθημερινότητα τον ανθρώπων μέχρι και σήμερα, αλλάζοντας τον τρόπο επικοινωνίας και κοινωνικοποίησης που υπήρχε μέχρι τότε. Αναλυτικότερα, οι πιο διάσημες πλατφόρμες, όπως το LinkedIn και το Skype κυκλοφόρησαν το 2003, το Facebook το 2004, το YouTube το 2005, το WhatsApp το 2009, το Pinterest και το Instagram το 2010 **[71]**.

- Η ανάπτυξη του κινητού υπολογισμού (mobile computing) εδραιώθηκε με την κυκλοφορία των πρώτων έξυπνων κινητών (smartphones). Το 2001 κυκλοφόρησε το Kyocera 6035, το πρώτο smartphone που συνδύαζε έναν προσωπικό ψηφιακό βοηθό (Personal Digital Assistant - PDA) (το οποίο είναι μία μικρή και εύχρηστη συσκευή, που χρησιμοποιείται με ένα ειδικό στυλό αντί για πληκτρολόγιο και επιτρέπει την αποθήκευση και ανάκτηση πληροφοριών **[74]**) με μία κινητή συσκευή, διαθέτοντας ωστόσο περιορισμένη περιήγηση στο διαδίκτυο. Ακολούθησαν και άλλες εταιρείες λανσάροντας παρόμοιες συσκευές, με την Apple να πρωτοπορεί το 2007 με την κυκλοφορία του iPhone, το πρώτο έξυπνο κινητό με οθόνη αφής (multi-touch) και προηγμένες λειτουργίες **[72]**.

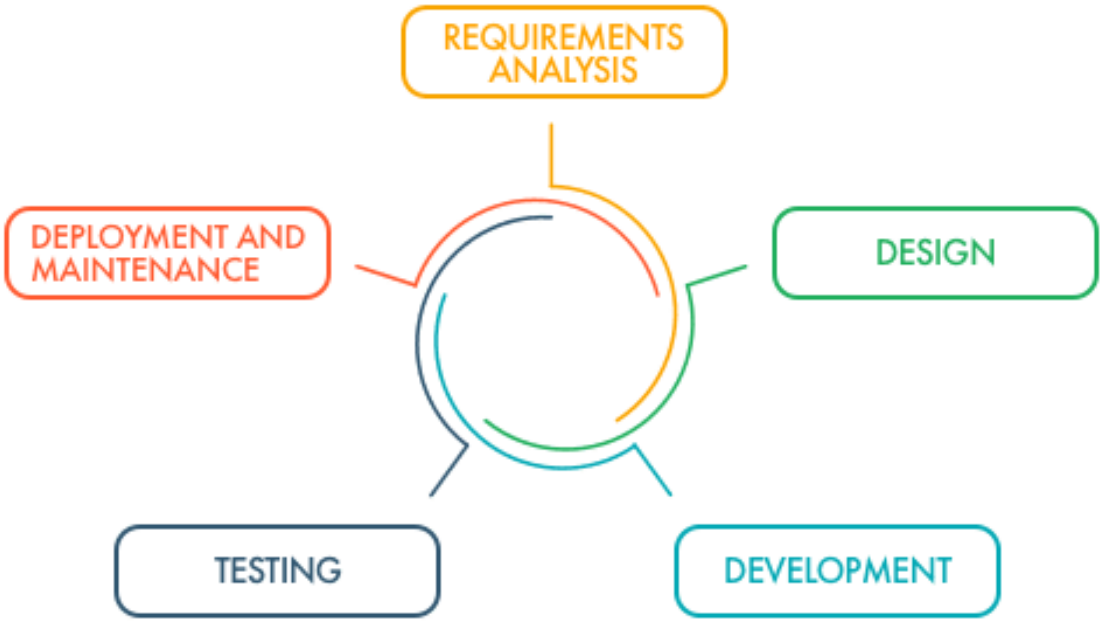
- Το 2002 η Amazon μέσω του Amazon Web Services (AWS), παρείχε υπηρεσίες αποθήκευσης και υπολογισμού μέσω του διαδικτύου και μετέπειτα, το 2006, κυκλοφόρησε την εμπορική υπηρεσία Elastic Compute Cloud (EC2), η οποία τέθηκε προς δημόσια χρήση. Αυτή αποτελούσε μια πρώτη μορφή πλατφόρμας υπολογιστικού νέφους (cloud computing platform), παρέχοντας διάφορα είδη υπηρεσιών νέφους. Το 2009, το Google Play άρχισε να παρέχει την εφαρμογή Cloud Computing Enterprise, το 2010 η Microsoft λανσάρει το Microsoft Azure καθώς και άλλες εταιρείες άρχισαν να

παρέχουν τις υπηρεσίες υπολογιστικού νέφους (cloud) τους, όπως η Alibaba, η IBM, η Oracle, και η HP **[73]**.

- Από την επόμενη δεκαετία και έπειτα, δηλαδή τη σημερινή εποχή, η τεχνολογία λογισμικού εξελίσσεται με ταχείς ρυθμούς, με εξειδικευμένες τεχνολογίες και εφαρμογές σε διάφορους τομείς. Το υπολογιστικό νέφος συνέχισε να επεκτείνεται και να λαμβάνει όλο και περισσότερη δημοτικότητα χάρη στο αυξημένο επίπεδο ασφάλειας δεδομένων που προσφέρει καθώς και τα νέα είδη γενικών & εξειδικευμένων υπηρεσιών που παρέχει, τα οποία καλύπτουν μεγάλη γκάμα προσωπικών και εταιρικών αναγκών. Επιπλέον, αλληλένδετος τομέας αποτελεί η τεχνητή νοημοσύνη καθώς εξελίσσεται παράλληλα με τη τεχνολογία λογισμικού, έτσι ώστε να επιτευχθεί η διαχείριση του πλέον μεγάλου όγκου των δεδομένων, η σωστή λήψη αποφάσεων, καθώς και ο έλεγχος, η συντήρηση και η ενημέρωση των έξυπνων συστημάτων. Παράλληλα, παρατηρείται άνοδος της εικονικής πραγματικότητας, η οποία προκύπτει από τη δημιουργία εξελιγμένου κώδικα που αποδίδει ρεαλιστικά γραφικά σε ένα εικονικό περιβάλλον προσφέροντας καθηλωτική εμπειρία στους χρήστες.

2.2.6 Διαδικασίες Παραγωγής Λογισμικού

Ο Κύκλος Ζωής Ανάπτυξης Λογισμικού (Software Development Life Cycle - SDLC) είναι ένα διεθνές πρότυπο, που περιλαμβάνει όλες τις φάσεις της διαδικασίας παραγωγής λογισμικού μέσω διακριτών βημάτων, με σκοπό την παροχή υψηλής ποιότητας υπηρεσιών. Παρακάτω αναλύονται τα στάδια του μοντέλου SDLC.



Εικόνα 12 - Κύκλος ζωής ανάπτυξης λογισμικού

2.2.6.1 Ανάλυση Απαιτήσεων Λογισμικού (Software Requirements Analysis)

Η Ανάλυση Απαιτήσεων Λογισμικού είναι το αρχικό στάδιο στη διαδικασία ανάπτυξης λογισμικού, το οποίο έχει ως σκοπό την εξέταση, την ανάλυση του έργου λογισμικού και των απαιτήσεων που καλείται να ικανοποιήσει. Έτσι, σύμφωνα με τις προσδοκίες του πελάτη, ορίζονται τα χαρακτηριστικά και οι λειτουργίες, δηλαδή οι λειτουργικές προδιαγραφές που θα διαθέτει το τελικό προϊόν λογισμικού. Η κατηγοριοποίηση των απαιτήσεων σε λειτουργικές, και μη λειτουργικές, στοχεύει στην ιεράρχηση των προτεραιοτήτων και στην σωστή οργάνωση για την αρχική σχεδίαση του έργου.

Κατά την ανάλυση αυτή, δίνεται προσοχή στην αποφυγή ερπυσμού χαρακτηριστικών σε έργα λογισμικού, δηλαδή στην αποφυγή υπερβολικών, περίπλοκων λειτουργιών που καθιστούν ένα λογισμικό δύσχρηστο προς τον χρήστη. Ο ερπυσμός προκύπτει λόγω κακού σχεδιασμού και έλλειψης επικοινωνίας με τον πελάτη, με αποτέλεσμα να παραχθεί ένα μη ποιοτικό και μη λειτουργικό προϊόν.

Απαραίτητο στοιχείο για το πρώτο στάδιο ανάπτυξης αποτελεί το έγγραφο προδιαγραφής απαιτήσεων λογισμικού (Software Requirements Specification - SRS), το οποίο παρέχει μια ολοκληρωμένη περιγραφή του προϊόντος λογισμικού, τις λειτουργίες και τις διαδικασίες που θα διαθέτει, ενώ περιλαμβάνει και ένα σύνολο περιπτώσεων χρήσης, όπως και ποια θα είναι η τελική του μορφή. Σκοπός του είναι η καθοδήγηση κατά τη περαιτέρω διαδικασία ανάπτυξης, η αποφυγή διαφωνιών σχεδίασης, η οριστικοποίηση του έργου, ώστε να αποτραπεί η επανάληψη εργασιών, η δημιουργία προβλημάτων και η απαίτηση αλλαγών ή τροποποιήσεων από τον πελάτη. Έτσι, διασφαλίζεται μέσω γραπτής επικύρωσης, ότι το τελικό προϊόν θα εξυπηρετεί σε ικανοποιητικό βαθμό τις ανάγκες και τις προσδοκίες του πελάτη.

Παράλληλα, καθορίζεται το χρονοδιάγραμμα όλων των φάσεων του έργου, μελετώνται οι πόροι που θα χρειαστούν, δηλαδή ανθρώπινοι, υλικοί και οικονομικοί πόροι. Εντοπίζονται τα πιθανά προβλήματα που ενδέχεται να προκύψουν και προβλέπονται οι τρόποι για την αποφυγή ή οι λύσεις για την αντιμετώπιση τους.

Χρήσιμη είναι η σχεδίαση διαγραμμάτων, καθώς αποτελούν σημαντικά εργαλεία στη διαδικασία της ανάλυσης, καθώς το καθένα παρουσιάζει διαφορετική οπτική του συστήματος, συμβάλλοντας στην κατανόηση των λειτουργικών απαιτήσεων. Τα πιο βασικά διαγράμματα είναι: το διάγραμμα περιβάλλοντος (Context Diagram), το διάγραμμα ροής δεδομένων (Data Flow Diagram) και το διάγραμμα περιπτώσεων χρήσης (Use Case Diagram). Το διάγραμμα περιβάλλοντος παρουσιάζει τις αλληλεπιδράσεις του συστήματος υπό ανάπτυξη με εξωτερικά συστήματα και χρήστες, οριοθετεί επομένως το σύστημα στο περιβάλλον του. Το διάγραμμα ροής δεδομένων απεικονίζει τον τρόπο κίνησης των δεδομένων μέσα στο σύστημα. Δείχνει δηλαδή από που προέρχονται τα δεδομένα, σε τι επεξεργασία υπόκεινται και που πηγαίνουν. Ουσιαστικά λειτουργεί σαν χάρτης του συστήματος που αναπαριστά τη πορεία των δεδομένων. Το διάγραμμα περιπτώσεων χρήσης αποτυπώνει τις συσχετιζόμενες περιπτώσεις χρήσης και δείχνει την αλληλεπίδραση μεταξύ του συστήματος και του εξωτερικού του περιβάλλοντος.

2.2.6.2 Σχεδιασμός Λογισμικού (Software Design/Planning)

Ο σχεδιασμός λογισμικού αποτελεί το επόμενο στάδιο κατά το οποίο, αφού έχουν συγκεντρωθεί οι απαιτήσεις, ακολουθεί ο τρόπος διαμόρφωσης της κατασκευής λογισμικού. Κατά τη διαδικασία αυτή δημιουργείται ένα λεπτομερές σχέδιο σχεδιασμού, σύμφωνα με τις οδηγίες του SRS. Ορίζεται ο αρχιτεκτονικός σχεδιασμός, οι δομές δεδομένων, οι αλγόριθμοι, οι διεπαφές για την ανταλλαγή πληροφοριών του συστήματος και οι μηχανισμοί ελέγχου ροής των δεδομένων που θα χρησιμοποιηθούν κατά την υλοποίηση. Ο σχεδιασμός αυτός πρέπει να είναι κατάλληλα δομημένος, ώστε να είναι δυνατές οι τροποποιήσεις και οι αλλαγές που ενδέχεται να προκύψουν. Ουσιαστικά, η διαδικασία σχεδιασμού ενός λογισμικού συστήματος αποτελεί τη φάση πρόληψης των προκλήσεων, της εκτίμησης και διαχείρισης του χρονικού πλαισίου, με σκοπό την ομαλή εξέλιξη και ολοκλήρωση του λογισμικού έργου.

2.2.6.3 Υλοποίηση Λογισμικού (Software Development)

Η φάση της ανάπτυξης λογισμικού αποτελεί την μετατροπή των διαγραμμάτων σχεδίασης του λογισμικού σε ένα ολοκληρωμένο πρόγραμμα λογισμικού. Το έργο αυτό υλοποιείται βάσει της μελέτης των δύο προηγούμενων σταδίων και σύμφωνα με τις ανάγκες του χρήστη. Συνήθως δεν ακολουθείται κάποια πρότυπη διαδικασία ή μέθοδος υλοποίησης, διότι η υλοποίηση λογισμικού αποτελεί μια δημιουργική δραστηριότητα, όπου κάθε προϊόν αναπτύσσεται διαφορετικά με βάση τις λειτουργίες και τις προδιαγραφές που απαιτούνται.

Μπορούν να χρησιμοποιηθούν διάφορα είδη αρχιτεκτονικής, όπως η αρχιτεκτονική πολλαπλών επιπέδων με πιο γνωστή την τριών επιπέδων (Three-Tier Architecture), η αρχιτεκτονική προσανατολισμένη σε υπηρεσίες (Service-Oriented Architecture) και άλλες. Η τριών επιπέδων περιλαμβάνει το front-end επίπεδο, δηλαδή τη διεπαφή χρήστη, το back-end επίπεδο που περιέχει την κύρια λογική της εφαρμογής/λογισμικού και το επίπεδο της βάσης δεδομένων (database). Η αρχιτεκτονική προσανατολισμένη σε υπηρεσίες χρησιμοποιείται για τον σχεδιασμό εφαρμογών που λειτουργούν ως σύνολο υπηρεσιών και επικοινωνούν μεταξύ τους μέσω πρότυπων μηχανισμών και πρωτοκόλλων.

Αρχικά για την υλοποίηση ενός προϊόντος λογισμικού, επιλέγεται η κατάλληλη γλώσσα προγραμματισμού ανάλογα με τον τύπο του λογισμικού που αναπτύσσεται και την εμπειρία και ικανότητες της ομάδας έργου. Για τη συγγραφή, επεξεργασία και έλεγχο του κώδικα του λογισμικού χρησιμοποιείται κάποιο περιβάλλον ανάπτυξης τύπου Integrated Development Environments. Ο κώδικας οργανώνεται σε μονάδες, όπως συναρτήσεις και κλάσεις, οι οποίες ενσωματώνονται σταδιακά σε ένα σύνολο. Παράλληλα, χρησιμοποιείται κάποιο πλαίσιο (Framework) που μπορεί να εξυπηρετήσει τις ανάγκες του συγκεκριμένου έργου (πχ. Spring Boot για την ανάπτυξη υπηρεσιών), η πλατφόρμα εκτέλεσης (Runtime Platform) του κώδικα, η βάση δεδομένων για αποθήκευση και ανάκτηση πληροφοριών καθώς και άλλα εργαλεία ανάπτυξης (πχ. εργαλεία αυτοματοποίησης κατασκευής όπως το Maven ή εργαλεία διαχείρισης διαμορφώσεων όπως το Git).

2.2.6.4 Έλεγχος Λογισμικού (Software Testing)

Αφού ολοκληρωθεί η υλοποίηση του πηγαίου κώδικα, πραγματοποιείται έλεγχος του μέσω δοκιμών πριν την παράδοση του στον τελικό χρήστη, όπου επαληθεύεται και επικυρώνεται το τελικό προϊόν. Με τη χρήση δοκιμών, αντιμετωπίζονται άμεσα τα σφάλματα κατά τη διάρκεια ανάπτυξης, ικανοποιούνται οι απαιτήσεις των πελατών, καθώς και ελέγχεται η ποιότητα και απόδοση του λογισμικού. Οι δοκιμές αυτές μπορεί να είναι αυτοματοποιημένες ώστε να εκτελούνται κάθε φορά που γίνονται αλλαγές στον κώδικα και έτσι να μειωθεί η διάρκεια της όλης δραστηριότητας της δοκιμής. Η σωστή διαχείριση του κώδικα συμβάλει στην επιτυχή υλοποίηση, η οποία διασφαλίζεται μέσω δοκιμών, διορθώσεων σφαλμάτων, βελτιώσεων της απόδοσης και άλλων μεθόδων. Η διαδικασία αυτή συνεχίζεται μέχρι το λογισμικό να απαλλαγεί πλήρως από σφάλματα (εκτός αν πρόκειται για σφάλμα minor, το οποίο αποτελεί έλλειψη ή κάποια ασυνέπεια που δεν είναι σοβαρή, δεν επηρεάζει δηλαδή τη βασική λειτουργικότητα του λογισμικού και θα μπορούσε να αντιμετωπιστεί σε επόμενη έκδοση του) και μέχρι να επιτευχθεί η επιθυμητή λειτουργικότητα και ποιότητα του λογισμικού.

Υπάρχουν διάφορα είδη δοκιμών που καλύπτουν τόσο το λειτουργικό όσο και μη λειτουργικό μέρος μιας εφαρμογής. Ορισμένα γνωστά είδη δοκιμών είναι τα ακόλουθα:

- Δοκιμές Μονάδας (Unit Testing): είναι μία μέθοδος δοκιμής λογισμικού με την οποία ελέγχονται μεμονωμένες μονάδες του πηγαίου κώδικα (όπως συναρτήσεις και κλάσεις) **[86]**.

- Δοκιμές Ενοποίησης (Integration Testing): αποτελεί μία φάση κατά την οποία ενοποιούνται τα συστατικά του λογισμικού με βαθμιαίο τρόπο και δοκιμάζεται το παραγόμενο προϊόν ως ομάδα, ώστε να εξακριβωθεί αν έχει την σωστή/επιθυμητή συμπεριφορά. **[87]**

- Δοκιμές Ασφάλειας (Security Testing): είναι μια διαδικασία με την οποία εντοπίζονται τρωτότητες ασφάλειας στον κύριο κώδικα και στις εξαρτήσεις του. Μπορεί να είναι στατικής ή δυναμικής φύσης. Στις στατικές δοκιμές ελέγχεται ο πηγαίος κώδικας. Στις δυναμικές δοκιμές ελέγχεται η (εξωτερική) συμπεριφορά του συστήματος από την οπτική ενός εξωτερικού επιτιθέμενου. **[89]**

- Δοκιμές Αντοχής (Stress Testing): είναι μια δραστηριότητα που καθορίζει την ευρωστία του λογισμικού δοκιμάζοντας το πέρα από τα όρια της κανονικής λειτουργίας του. Συνεπώς, είναι ένα τεστ καταπόνησης συστήματος, το οποίο δίνει έμφαση στην ευρωστία, τη διαθεσιμότητα και τον χειρισμό σφαλμάτων υπό βαρύ φορτίο, παρά σε αυτό που θα μπορούσε να θεωρηθεί σωστή συμπεριφορά υπό κανονικές συνθήκες **[90]**. Στόχος είναι να αναδειχθούν προβλήματα στην συμπεριφορά του συστήματος πέρα από τα όρια αντοχής του καθώς και να εξεταστεί αν το σύστημα αποτυγχάνει πλήρως ή μερικώς σε αυτή την περίπτωση. Ιδανικά θα πρέπει να συνεχίσει να λειτουργεί κανονικά ακόμη και σε τέτοιες συνθήκες καταπόνησης.

2.2.6.5 Συντήρηση Λογισμικού (Software Maintenance)

Η συντήρηση λογισμικού αποτελεί μία συνεχής διαδικασία, κατά την οποία διασφαλίζεται ότι το σύστημα παραμένει λειτουργικό με την πάροδο του χρόνου. Είναι το τελευταίο στάδιο ανάπτυξης λογισμικού, όπου το σύστημα υπόκεινται στη προβλεπόμενη συντήρηση ανάλογα με τις δυνατότητες, το είδος και τον προϋπολογισμό του εκάστοτε οργανισμού. Η προβλεπόμενη συντήρηση μπορεί να είναι: διορθωτική, προσαρμοστική, προληπτική και τελειοποιητική συντήρηση.

Μέσω της διορθωτικής συντήρησης (Corrective Maintenance) εντοπίζονται και διορθώνονται σφάλματα που επηρεάζουν την σωστή λειτουργία του λογισμικού σε ορισμένες περιπτώσεις, την εμπειρία χρήστη ή ακόμα και αυτά που μπορεί να αποτελέσουν ευπάθεια ασφάλειας του συστήματος. Ο εντοπισμός των σφαλμάτων μπορεί να γίνει είτε από την ομάδα ανάπτυξης, είτε από τους χρήστες του συστήματος καθώς το χειρίζονται. Τέτοια προβλήματα μπορεί να είναι σημασιολογικά προγραμματιστικά λάθη ή αποτυχίες σύνδεσης χρηστών.

Στην προσαρμοστική συντήρηση (Adaptive Maintenance) λογισμικού διασφαλίζεται ότι το προϊόν παραμένει ενημερωμένο, διαθέτοντας τις πιο πρόσφατες και κατάλληλες τεχνολογίες καθώς και μεθόδους ασφαλείας. Παράλληλα, τροποποιείται για να ταιριάξει με τις νέες ανάγκες του (επιχειρηματικού) περιβάλλοντος και των χρηστών.

Μέσω της προληπτικής συντήρησης (Preventive Maintenance) εντοπίζονται πιθανά ζητήματα που μπορεί να επηρεάσουν το προϊόν μελλοντικά και επιλύονται προληπτικά. Είναι μία μέθοδος προγραμματισμού και παρακολούθησης του λογισμικού για να ανιχνεύονται προβλήματα, όπως πιθανά σφάλματα, ενδεχομένως αυξημένος φόρτος εργασίας, κ.α.

2.2.7 Επαλήθευση και Επικύρωση

Βασικός στόχος της φιλοσοφίας της τεχνολογίας λογισμικού είναι η εξασφάλιση της αξιοπιστίας, της ασφάλειας, της αποδοτικότητας, της διατηρησιμότητας και της ευχρηστίας του λογισμικού. Ο ποιοτικός έλεγχος του λογισμικού διασφαλίζει όλα τα παραπάνω και πραγματοποιείται μέσω των δραστηριοτήτων επαλήθευσης (verification) και επικύρωσης (validation) του λογισμικού **[58]**.

Η διαδικασία της επαλήθευσης αναφέρεται στον έλεγχο του λογισμικού που εκτελείται στη περίοδο ανάπτυξης, εξασφαλίζοντας ότι η υλοποίηση γίνεται σύμφωνα με τις ζητούμενες προδιαγραφές και τις αναμενόμενες απαιτήσεις. Περιλαμβάνει την επιθεώρηση, την δοκιμή του λογισμικού και μέσω της ανάλυσης, ανασκόπησης κώδικα, έχει ως σκοπό την απόδειξη της ορθότητας ενός προγράμματος λογισμικού χωρίς σφάλματα.

Η διαδικασία της επικύρωσης αναφέρεται στην αξιολόγηση του τελικού προϊόντος, όπου βεβαιώνεται ότι ικανοποιεί τις ανάγκες και απαιτήσεις του χρήστη. Ελέγχει ότι το προϊόν που αναπτύσσεται είναι το σωστό κάνοντας δοκιμές επικύρωσης (όπως User Acceptance Testing - UAT), για να διασφαλιστεί ότι το προϊόν είναι έτοιμο προς παράδοση. Μέσω των δοκιμών επικύρωσης εξετάζονται βασικά σενάρια χρήσης του λογισμικού από τους χρήστες, καθώς και άλλα μη λειτουργικά χαρακτηριστικά, όπως η χρηστικότητα.

Η επικύρωση διαφέρει από την επαλήθευση και ως προς τη σειρά εκτέλεσης, καθώς πρώτα γίνεται η δοκιμή επαλήθευσης για να επιβεβαιωθεί ότι η απόδοση και λειτουργικότητα του λογισμικού πληροί τις προκαθορισμένες προδιαγραφές. Αφού γίνει η επαλήθευση, τότε ακολουθεί η διαδικασία επικύρωσης, για να επικυρωθεί ότι αυτές οι προδιαγραφές του προϊόντος καλύπτουν τις ανάγκες των χρηστών. Εν ολίγοις, η επαλήθευση λογισμικού θέτει την ερώτηση: “Κατασκευάζουμε σωστά το προϊόν;” Ενώ η επικύρωση λογισμικού θέτει την ερώτηση: “Κατασκευάζουμε το σωστό προϊόν;” **[59]**. Με την ολοκλήρωση των παραπάνω διαδικασιών, το λογισμικό μπορεί να χαρακτηριστεί ως “fit-for-purpose”, καθώς θεωρείται κατάλληλο για τον σκοπό του.

2.2.8 Μοντέλα/Μεθοδολογίες Ανάπτυξης Λογισμικού

Η μεθοδολογία ανάπτυξης λογισμικού είναι ένα πλαίσιο δομημένων διαδικασιών που χρησιμοποιείται για τη δομή, τον σχεδιασμό και τον έλεγχο της ανάπτυξης ενός συστήματος λογισμικού. Αποτελεί στρατηγική διαχείρισης που επιτυγχάνει την αποτελεσματική οργάνωση, μέσω μιας ακολουθίας βημάτων, προσφέροντας καθοδήγηση σε κάθε στάδιο ανάπτυξης ενός προϊόντος λογισμικού. Υπάρχουν διάφορα είδη μεθοδολογιών, όπου το καθένα διαθέτει τα δικά του πλεονεκτήματα αλλά και μειονεκτήματα. Είναι σημαντική η επιλογή της κατάλληλης μεθοδολογίας ανάλογα με τις προδιαγραφές, τις απαιτήσεις, τους στόχους του έργου λογισμικού, το είδος του και τις ικανότητες της ομάδας έργου ώστε να υπάρχει η κατάλληλη καθοδήγηση κατά την ανάπτυξη του λογισμικού. Έτσι, διασφαλίζεται η καλή επικοινωνία μεταξύ της ομάδας ανάπτυξης αλλά και με τον πελάτη, λαμβάνονται σωστές αποφάσεις. Ακολουθεί μια επισκόπηση των πιο ευρέως χρησιμοποιούμενων και δημοφιλών μεθοδολογιών ανάπτυξης λογισμικού.

- Waterfall (Καταρράκτη): είναι το πιο αναγνωρίσιμο μοντέλο υλοποίησης του SDLC. Αποτελεί μία γραμμική διαδοχική/ακολουθιακή προσέγγιση, η οποία φροντίζει να ολοκληρώνεται ένα στάδιο του SDLC πριν ξεκινήσουν οι εργασίες για το επόμενο στάδιο. Για αυτό, σε έργα με μεταβαλλόμενες απαιτήσεις δεν προτιμάται λόγω αυτής της ανελαστικότητας του. Χρησιμοποιείται κυρίως σε έργα με σαφές πεδίο εφαρμογής και με προβλέψιμα αποτελέσματα, καθώς είναι αδύνατη η επιστροφή σε προηγούμενο στάδιο για τη διαχείριση αλλαγών στη περίπτωση που προκύψουν νέες απαιτήσεις. Η μεθοδολογία αυτή εξοικονομεί χρόνο, λειτουργεί αποτελεσματικότερα για μικρά έργα και λόγω της απλότητας της προτιμάται από αρχάριους προγραμματιστές με μικρή εμπειρία στο αντικείμενο.

- Agile (Ευκίνητο μοντέλο ανάπτυξης): αποτελεί μία βέλτιστη πρακτική για ανάπτυξη λογισμικού, η οποία διαχειρίζεται διάφορα είδη έργων. Στόχος του είναι να δημιουργήσει ένα βιώσιμο προϊόν, το οποίο θα βελτιώνεται συνεχώς μέσω επαναλήψεων (increments), ανάλογα με νέες πληροφορίες που θα προκύπτουν σχετικά με τις ανάγκες και τις προτιμήσεις των χρηστών καθώς και τις λειτουργίες που έχουν προγραμματιστεί να υλοποιηθούν. Οι επαναλήψεις μπορεί να αφορούν διαφορετικές εργασίες, όπως προσθήκη νέων λειτουργιών, τροποποίηση του σχεδιασμού ή της υλοποίησης υπαρχόντων λειτουργιών, και διάφορες δοκιμές, οι οποίες ενδέχεται να γίνονται και ταυτόχρονα. Ουσιαστικά, οι δραστηριότητες ανάπτυξης μπορεί να διαβαλλόμενες. Αποτελεί μια ευέλικτη/ευκίνητη μέθοδο ανάπτυξης καθώς προσαρμόζεται σε νέες απαιτήσεις σύμφωνα με τα σχόλια των χρηστών επιτυγχάνοντας τη βέλτιστη επικοινωνία και αλληλεπίδραση μεταξύ προγραμματιστών και πελάτη. Ωστόσο, επειδή δεν ορίζονται αρχικά οι προδιαγραφές του έργου, το τελικό αποτέλεσμα είναι ασαφές. Ακόμη, εφόσον δεν ορίζεται χρονικό διάγραμμα, ούτε προθεσμίες παράδοσης, είναι αδύνατον να υπολογιστεί η ημερομηνία ολοκλήρωσης του έργου.

- DevOps: είναι ένα μοντέλο που συνδυάζει ομάδες ανάπτυξης και λειτουργιών για την προώθηση της συνεργασίας, τη διασφάλιση της ποιότητας, και της αποτελεσματικότητας στην ανάπτυξη λογισμικού. Αυτά τα δύο τμήματα (ανάπτυξης και λειτουργιών) λειτουργούν ως ενιαία ομάδα για όλες τις διαδικασίες (ανάπτυξης) σε όλο τον κύκλο ζωής. Αποτελεί γρήγορη και εύκολη διαδικασία, καθώς οι δραστηριότητες/λειτουργίες εκτελούνται ταυτόχρονα, συμβάλλοντας στην ταχεία παράδοση του έργου. Έτσι, το μοντέλο DevOps μπορεί να θεωρηθεί ως συμπληρωματικό πρότυπο του Agile. Το μοντέλο αυτό (DevOps) καταφέρνει ακόμη να δημιουργήσει βέλτιστη συνεργασία και αλληλεπίδραση με τον πελάτη μέσω συχνών ενημερώσεων και συνεχών αυτόματων παραδόσεων του έργου σε διάφορες φάσεις κατά τη διάρκεια της υλοποίησης, ανταποκρίνοντας έτσι άμεσα στις απαιτήσεις του.

- Scrum: είναι ένα πλαίσιο βασισμένο στη μεθοδολογία Agile, που περιλαμβάνει τον ιδιοκτήτη του προϊόντος, τον Scrum Master και την ομάδα ανάπτυξης. Ο ιδιοκτήτης είναι σε επικοινωνία με τον πελάτη, ελέγχει την εξέλιξη του έργου και φροντίζει την κάλυψη των απαιτήσεων του πελάτη. Ο Scrum Master διασφαλίζει την εφαρμογή των αρχών Scrum, προωθώντας τη συνεργασία μεταξύ της ομάδας και βοηθώντας την να συνεχίσει απρόσκοπτα το έργο της (μακριά από κάθε είδους παρεμβολές). Η ομάδα ανάπτυξης εργάζεται σε επαναλήψεις που ονομάζονται σπριντ (Sprint), ώστε να παραδώσει κώδικα στο τέλος κάθε επανάληψης. Γίνονται καθημερινές συναντήσεις της ομάδας ανάπτυξης με σκοπό την ενίσχυση της παραγωγικότητας και την παρακολούθηση της προόδου. Η ανάπτυξη Scrum προσφέρει γρήγορη παράδοση, συχνή ενημέρωση της εξέλιξης του έργου, περιλαμβάνει τακτικά σχόλια ενώ οι αλλαγές γίνονται άμεσα. Το προϊόν που παραδίδεται με τη χρήση αυτής της μεθόδου είναι οικονομικό λόγω της ταχείας παράδοσης που είναι εντός του προγραμματισμένου χρόνου. Προτιμάται για έργα μικρά, ταχείας εκτέλεσης και απαιτεί έμπειρα άτομα για τη διαχείριση και εφαρμογή της μεθοδολογίας.

- Lean: η λιτή ανάπτυξη δημιουργεί ένα οικονομικά αποδοτικό λογισμικό, που διαθέτει αρχές λιτής παραγωγής, βελτιστοποιεί τους πόρους χρησιμοποιώντας το ένα τρίτο των τυπικών κεφαλαίων, ανθρώπινων διαδικασιών και χρόνου εργασίας. Αποτελεί μία μινιμαλιστική προσέγγιση μειώνοντας περιττές σπατάλες, όπως συναντήσεις, τεκμηρίωση και επαναλαμβανόμενες εργασίες. Η ανάπτυξη λογισμικού που προσφέρει είναι ταχεία, αποτελεσματική και οικονομική. Η ευελιξία της μεθόδου είναι περιορισμένη με σκοπό την αποφυγή σπατάλης χρόνου. Απαιτεί ομαδική εργασία, πειθαρχία, προηγμένες δεξιότητες και εμπειρία από την ομάδα ανάπτυξης.

- Integration and Configuration: το μοντέλο ενοποίησης και διαμόρφωσης επικεντρώνεται στην επαναχρησιμοποίηση υπαρχόντων στοιχείων λογισμικού σε ένα νέο σύστημα λογισμικού. Αυτά τα επαναχρησιμοποιούμενα στοιχεία μπορούν να διαμορφωθούν ώστε να προσαρμόσουν τη συμπεριφορά και τη λειτουργικότητά τους σύμφωνα με τις απαιτήσεις των χρηστών και του νέου λογισμικού συστήματος. Έτσι, αυτή η διαδικασία της επαναχρησιμοποίησης μειώνει το κόστος και τους κινδύνους ενώ συμβάλει στη γρήγορη παράδοση του έργου.

2.2.9 Συσχέτιση Μηχανικής Συστημάτων & Λογισμικού

Η Μηχανική Συστημάτων (Systems Engineering) είναι ένας επιστημονικός κλάδος που επικεντρώνεται στις πτυχές ανάπτυξης συστημάτων που βασίζονται σε υπολογιστές, συμπεριλαμβανομένου της μηχανικής υλικού, λογισμικού & διαδικασιών. Συνεπώς, η Τεχνολογία Λογισμικού αποτελεί μέρος αυτής της ευρύτερης διαδικασίας.

Σύμφωνα με τους Pyster et al. στο συνέδριο που διεξήχθη το 2014 **[98]** σε ένα εργαστήριο στο Ινστιτούτο Τεχνολογίας του Στίβεν στο Χόμποκεν, διερευνήθηκαν οι αναπτυξιακές προσεγγίσεις των δύο κλάδων, οι τεχνικές σχέσεις, η σχέση μεταξύ των ανθρώπων που ασχολούνται με τους δύο τομείς της μηχανικής, η εκπαίδευση των μηχανικών συστημάτων και των μηχανικών λογισμικού. Οι Pyster et al. ορίζουν μία δισδιάστατη οντολογία, όπου η πρώτη διάσταση περιλαμβάνει την κάθετη και τη οριζόντια κατάσταση ενός συστήματος και η δεύτερη διάσταση περιλαμβάνει τρεις κατηγορίες συστημάτων, δηλαδή τα φυσικά, υπολογιστικά και κυβερνοφυσικά.

Όσον αφορά την πρώτη διάσταση, οι κάθετες διαστάσεις ενός συστήματος αφορούν τις διαμορφωμένες δομές γύρω από τεχνικά εστιασμένους κλάδους μηχανικής, δηλαδή η μηχανολογική μηχανική επικεντρώνεται σε μηχανικές πτυχές. Οι οριζόντιες διαστάσεις αφορούν τις οριζόντιες ανησυχίες σε επίπεδο συστημάτων, όπως για παράδειγμα οι προτιμήσεις πελατών που διαμορφώνουν ανάλογα ένα σύστημα.

Όσον αφορά τη δεύτερη διάσταση, οι τρεις κατηγορίες αντικατοπτρίζουν το τεχνολογικό “κέντρο βάρους” ενός συστήματος. Τα φυσικά συστήματα λειτουργούν και παράγουν ύλη ή ενέργεια. Χρησιμοποιούν τεχνολογίες υπολογισμού και λογισμικού ως

εξαρτήματα, τα οποία δεν κυριαρχούν στην οριζόντια διάσταση της μηχανικής. Αντιθέτως, θεωρούνται κάθετες ανησυχίες.

Τα υπολογιστικά συστήματα διαθέτουν υπολογιστική συμπεριφορά και ανήκουν στην κάθετη μηχανική λογισμικού. Αυτά τα συστήματα έχουν την ικανότητα να λειτουργούν, να παράγουν δεδομένα και να διαθέτουν ανθρώπινα, φυσικά στοιχεία. Δεν αποτελούν όμως πρόκληση για την ανάπτυξη και εξέλιξη των συστημάτων λόγω της πολύπλοκης υλοποίησης των υπολογιστικών συμπεριφορών τους.

Η τρίτη κατηγορία συστημάτων, δηλαδή των κυβερνοφυσικών, προέκυψε με την εξέλιξη των φυσικών και υπολογιστικών συστημάτων, αποκτώντας ένα συνδυαστικό ρόλο στη μηχανική λογισμικού. Αυτά τα συστήματα θέτουν μεγάλες προκλήσεις οριζόντιας μηχανικής σε όλους τους τομείς. Περιλαμβάνουν τα σύγχρονα έξυπνα συστήματα, ρομποτικά συστήματα, το διαδίκτυο των πραγμάτων και άλλα.

Στο συνέδριο ακόμη μελετήθηκε και το κομμάτι της εκπαίδευσης των δύο κλάδων: της μηχανικής συστημάτων και μηχανικής λογισμικού, όπου παρατηρήθηκε ότι αποτελούν ξεχωριστούς κλάδους φοίτησης στα πανεπιστήμια. Για αυτό επισημάνθηκε ότι πρέπει να αναπτυχθούν προγράμματα σπουδών, τα οποία συγχωνεύουν αυτά τα δύο πεδία και παρέχουν πτυχίο μηχανικής συστημάτων-λογισμικού. Έτσι, θα προκύψουν επιστήμονες με ευρείες τεχνολογικές γνώσεις, που θα εμβαθύνουν σε διάφορους τύπους συστημάτων.

Γενικότερα στο έγγραφο συμπεραίνεται ότι η συσχέτιση αυτή αποτελεί ένα έργο σε εξέλιξη, που διαθέτει πολλές προοπτικές στον τρόπο ανάπτυξης των μηχανικών συστημάτων και λογισμικού.

Κεφάλαιο 3: Μεθοδολογία Βιβλιογραφικής Επισκόπησης

Στο κεφάλαιο αυτό προσδιορίζεται η μεθοδολογία της συστηματικής ανασκόπησης της βιβλιογραφίας (Systematic Literature Review - SLR) που πραγματοποιήθηκε. Αρχικά, με σκοπό τον καθορισμό του πεδίου έρευνας, ορίζονται κάποια βασικά ερωτήματα προς εξέταση. Τα ερωτήματα αυτά δίνουν μία κατευθυντήρια γραμμή σχετικά με την μελέτη που θα ακολουθήσει στην συνέχεια, δίνοντας έμφαση σε σημαντικές πτυχές του θέματος. Στη συνέχεια, προσδιορίζονται οι πηγές αναζήτησης και οι λέξεις-κλειδιά που χρησιμοποιήθηκαν για την εύρεση της κατάλληλης βιβλιογραφίας. Έπειτα, ορίζονται τα κριτήρια ένταξης και αποκλεισμού άρθρων, με τα οποία διασφαλίζεται ότι το περιεχόμενο είναι αξιόπιστο και ότι ανταποκρίνεται σε επιστημονικά αναγνωρισμένες μελέτες. Αφού ολοκληρωθεί το παραπάνω ποιοτικό φιλτράρισμα των άρθρων, παρέχονται σχετικά στατιστικά δεδομένα, όπως ο αριθμός και το ποσοστό άρθρων που διατηρήθηκαν, τα ποσοστά ανά είδος και ανά εκδότη.

3.1 Ερευνητικά Ερωτήματα

Στόχος της παρούσας μελέτης είναι να εξεταστεί η συνέργεια μεταξύ της Τεχνητής Νοημοσύνης και της Τεχνολογίας Λογισμικού. Εστιάζοντας λοιπόν στην κατανόηση του θέματος και για την καθοδήγηση της έρευνας, τίθενται κάποια κρίσιμα ερωτήματα προς απάντηση.

Ερώτημα 1: Ποιες τεχνικές/μέθοδοι τεχνητής νοημοσύνης εφαρμόζονται στις διαδικασίες τεχνολογίας λογισμικού;

Ερώτημα 2: Ποια συστήματα/πλαίσια υπάρχουν που να εφαρμόζουν την συνέργεια των δύο τομέων;

Ερώτημα 3: Ποιες είναι οι προσεγγίσεις/πρακτικές της ΤΛ για συστήματα βασισμένα στην ΤΝ;

Ερώτημα 4: Ποια είναι η αλληλεπίδραση των δύο κλάδων;

Ερώτημα 5: Τι προκλήσεις, προβλήματα προκύπτουν από αυτήν την συνέργεια;

3.2 Προσδιορισμός Πηγών Αναζήτησης

Για τη διεξαγωγή της βιβλιογραφικής έρευνας, δημιουργήθηκε μια συμβολοσειρά αναζήτησης με τη χρήση κατάλληλων λέξεων κλειδιών, για την εύρεση σχετικών αποτελεσμάτων. Στόχος της συμβολοσειράς είναι η κατάλληλα προσανατολισμένη διαδικασία εξαγωγής άρθρων και η ανίχνευση χρήσιμων δεδομένων για τη συγγραφή της μελέτης. Χρησιμοποιήθηκε η ακόλουθη συμβολοσειρά με τις εξής λέξεις κλειδιά: “(Artificial Intelligence | AI | Machine Learning | ML) & (Software & (Engineering | Development | Design | Testing | Requirements Engineering | Maintenance)”. Είναι σημαντικό να διασφαλιστεί ότι τα εξαγόμενα δεδομένα είναι έγκυρα και ότι προέρχονται από αξιόπιστες πηγές. Η εφαρμογή της συμβολοσειράς αναζήτησης έγινε σε γνωστές, εγκεκριμένες ψηφιακές βιβλιοθήκες: Elsevier, Springer, IEEE και στο Google Scholar.

3.3 Κριτήρια Ένταξης και Αποκλεισμού Άρθρων

Κατά την αναζήτηση βιβλιογραφίας είναι σημαντικό να τεθούν κάποια κριτήρια αποδοχής και απόρριψης άρθρων, καθώς τα αποτελέσματα ενδέχεται να είναι άπειρα και να ποικίλλουν. Για αυτό, διεξήχθη μια διαδικασία διαλογής, σύμφωνα με τις παρακάτω απαιτήσεις:

Κριτήρια Ένταξης:

I. Αποδοχή άρθρων που είναι διαθέσιμα προς ανάγνωση.

II. Αποδοχή άρθρων που έχουν συνταχθεί σε συγκεκριμένη φυσική γλώσσα (αγγλικά και ελληνικά), ώστε να είναι κατανοητό το περιεχόμενό τους, προκειμένου να επιτευχθεί ορθή επεξεργασία και αξιόπιστη ανάλυση δεδομένων.

III. Αποδοχή άρθρων με ημερομηνία συγγραφής μετά το έτος 2000, ώστε τα δεδομένα που περιέχονται σε αυτά να αφορούν σύγχρονες τεχνολογίες.

IV. Έλεγχος περίληψης και περιεχομένου των άρθρων, με σκοπό να συνάδουν με το πεδίο της έρευνας και να ανταποκρίνονται στις ερωτήσεις που έχουν τεθεί.

V. Αποδοχή επιστημονικών/ερευνητικών άρθρων που έχουν δημοσιευθεί σε περιοδικά ή συνέδρια, βιβλίων ή κεφαλαίων βιβλίων και αναφορών διδακτορικών διατριβών.

Κριτήρια Αποκλεισμού:

I. Απόρριψη άρθρων που δεν διαθέτουν επαρκώς ανεπτυγμένο περιεχόμενο, με αποτέλεσμα να μην προκύπτουν αρκετά στοιχεία προς ανάπτυξη.

II. Απόρριψη άρθρων με δυσνόητο, ασαφές περιεχόμενο, που γενικότερα δεν αποτελούν ποιοτικές προσεγγίσεις.

III. Απόρριψη άρθρων που δεν είναι δωρεάν διαθέσιμα προς ανάγνωση και συλλογή δεδομένων.

IV. Απόρριψη άρθρων που βρέθηκαν παραπάνω από μία φορά, κατά την αναζήτηση σε διαφορετικές πηγές.

3.4 Στατιστικά Αναζήτησης

Στον εκδοτικό οίκο Springer στην κατηγορία Journals με τη χρήση της συμβολοσειρά αναζήτησης στο SpringerOpen (όπου υπάρχουν μόνο ανοικτής πρόσβασης άρθρα), βρέθηκαν 31 αποτελέσματα.

Στον εκδοτικό οίκο Elsevier τα αποτελέσματα της αναζήτησης αφορούσαν 34 επιστημονικά περιοδικά (journals).

Στον εκδοτικό οίκο IEEE αναζητώντας με βάση τον τίτλο και την περίληψη (δεν χρησιμοποιήθηκε η συμβολοσειρά, διότι δεν προέκυπταν αποτελέσματα), τα αποτελέσματα ήταν 19 άρθρα από συνέδρια (conferences) και 10 από περιοδικά.

Στο Google Scholar αναζητώντας στο χρονικό διάστημα 2000-2024 και προσθέτοντας ως φίλτρο να υπάρχουν οι λέξεις κλειδιά στον τίτλο, τα αποτελέσματα ήταν 22.

Οπότε, συλλέχθηκαν συνολικά 117 άρθρα. Κατά το φιλτράρισμα αρχικά ελέγχθηκε αν ο τίτλος και η περίληψη των άρθρων αντιστοιχούσαν με το αντικείμενο της διπλωματικής εργασίας και όχι απλά να αναφέρονται οι όροι μέσα στο περιεχόμενο. Στη συνέχεια, ελέγχοντας τα κριτήρια ένταξης, αποκλεισμού που προαναφέρθηκαν και μέσω ενδελεχούς ανάγνωσης των περιεχομένων, συμπεριλήφθηκαν στη SLR 20 άρθρα (17%).

Έπειτα από ανάλυση των άρθρων που επιλέχθηκαν προέκυψαν κάποια αποτελέσματα που παρουσιάζονται μέσω ποσοστών ανά είδος, ανά εκδότη και ανά περιεχόμενο των άρθρων.

- Σχετικά με το είδος των άρθρων που διατηρήθηκαν, προέκυψαν τα εξής στατιστικά:

I. τα βιβλία αφορούσαν το 10% (δηλαδή συνολικά ήταν 2),

II. τα επιστημονικά περιοδικά (journals) αφορούσαν το 30% (δηλαδή συνολικά ήταν 6),

III. τα συνέδρια (conferences) αφορούσαν το 55% (δηλαδή συνολικά ήταν 11),

IV. οι διπλωματικές διατριβές (thesis) αφορούσαν το 5% (δηλαδή συνολικά ήταν 1) των αποτελεσμάτων.

- Σχετικά με τα ποσοστά ανά εκδότη των άρθρων που διατηρήθηκαν:

I. στο Springer βρέθηκαν 3 αποτελέσματα, δηλαδή 15%,

II. στο Elsevier βρέθηκαν 2 αποτελέσματα, δηλαδή 10%,

III. στο IEEE βρέθηκαν 7 αποτελέσματα, δηλαδή 35%,

IV. από άλλους εκδότες στο Google Scholar βρέθηκαν 8 αποτελέσματα, δηλαδή 40%.

- Σχετικά με το περιεχόμενο προέκυψαν τα εξής στατιστικά:

I. η χρήση των τεχνικών, μεθόδων, εργαλείων τεχνητής νοημοσύνης στην τεχνολογία λογισμικού αναλύεται σε 13 άρθρα, δηλαδή ποσοστό 65%,

II. η αντίστροφη χρήση αναλύεται σε 5 άρθρα, δηλαδή ποσοστό 25%,

III. τα άρθρα που εξετάζουν πλαίσια με ενσωμάτωση των δύο επιστημονικών πεδίων είναι συνολικά 5, δηλαδή 25%.

Κάποια άρθρα (συνολικά 3) συμπεριλαμβάνονται σε δύο κατηγορίες καθώς το περιεχόμενο τους αφορούσε και τις δύο κατευθύνσεις συνέργειας (για αυτό και το άθροισμα των παραπάνω ποσοστών είναι μεγαλύτερο του 100%). Είναι λογικό να υπάρχει αυτή η απόκλιση στα ποσοστά σχετικά με τη πρώτη κατεύθυνση καθώς η ΤΝ περιέχει πολλά υποπεδία, όπως η μηχανική μάθηση, τα νευρωνικά δίκτυα, συνεπώς υπάρχει μεγαλύτερο εύρος επιλογών προς χρήση από την τεχνολογία λογισμικού.

Κεφάλαιο 4: Ανάλυση

Σε αυτό το κεφάλαιο εξετάζονται και αναλύονται έρευνες σχετικές με το αντικείμενο της παρούσας διπλωματικής εργασίας. Σκοπός του κεφαλαίου είναι η εύρεση απαντήσεων στα ερωτήματα που τέθηκαν στην Ενότητα 3.1 και η δημιουργία ενός χάρτη συνεργειών μεταξύ της τεχνητής νοημοσύνης και τεχνολογίας λογισμικού.

4.1 Ανάλυση Άρθρων

Η αρχική ανάγκη για συνέργεια της τεχνητής νοημοσύνης και τεχνολογίας λογισμικού προέκυψε όταν η διαχείριση του τεράστιου όγκου των δεδομένων αποτελούσε πρόβλημα και απαιτούνταν νέοι αυτοματοποιήσιμοι μέθοδοι ανάλυσης δεδομένων, όπως οι αλγόριθμοι ML. Η συνέργεια των δύο τομέων ξεκίνησε μέσω της υλοποίησης αλγορίθμων ΤΝ (π.χ. Mark 1 perceptron), ως μεμονωμένων προγραμμάτων και κατέληξε στη δημιουργία συστημάτων λογισμικού με ενσωματωμένα στοιχεία ΤΝ.

Η αλληλεπίδραση αυτή συμβάλλει στην δημιουργία ισχυρών και αυτόνομων συστημάτων με εξελιγμένες δυνατότητες, όπως διαχείριση έργων, αυτόματη δημιουργία και δοκιμή κώδικα, αυτόματη αποσφαλμάτωση και πολλές άλλες. Η συνέργεια των δύο κύριων τομέων της επιστήμης των υπολογιστών αναμένεται να αυτοματοποιήσει τα πάντα στην τεχνολογία του σήμερα, οδηγώντας στην απόκτηση νέων και ακόμα πιο εξελιγμένων ικανοτήτων στα συστήματα που τις συνδυάζουν (engineering AI systems).

Για να απαντηθούν οι ερευνητικές ερωτήσεις που τέθηκαν, τα ευρήματα από τη σχετική βιβλιογραφία οργανώθηκαν ως εξής: 1) τεχνικές/μέθοδοι ΤΝ που εφαρμόζονται στις διαδικασίες τεχνολογίας λογισμικού, 2) Συστήματα/Πλαίσια που εφαρμόζουν την συνέργεια των δύο τομέων, 3) προσεγγίσεις/πρακτικές της ΤΛ για συστήματα βασισμένα στην ΤΝ. Συνεπώς, υπάρχει 1-1 αντιστοίχιση με τα 3 πρώτα ερευνητικά ερωτήματα. Το ερώτημα 4 βρίσκει απάντηση μέσω του χάρτη συνεργειών και των αποτελεσμάτων της συνέργειας που αναπτύσσονται στην ενότητα 4.2. Το ερώτημα 5 βρίσκει απάντηση στο πέμπτο κεφάλαιο στην ενότητα 5.1, όπου αναλύονται οι προκλήσεις και τα προβλήματα που εντοπίστηκαν στη βιβλιογραφία και προκύπτουν τα ανάλογα συμπεράσματα.

4.1.1 Τεχνικές/μέθοδοι ΤΝ που εφαρμόζονται στις διαδικασίες τεχνολογίας λογισμικού

Η τεχνητή νοημοσύνη μπορεί να συνεισφέρει στη βελτίωση και αυτοματοποίηση διαδικασιών ανάπτυξης λογισμικού, εφαρμόζοντας διάφορες τεχνικές και μεθόδους, σύμφωνα με σχετικές μελέτες και έρευνες.

Σύμφωνα με τον Harman **[100]** ο κλάδος της τεχνολογίας λογισμικού εφαρμόζει τρεις τεχνικές τεχνητής νοημοσύνης για την βελτίωση και υποστήριξη των διαδικασιών ανάπτυξης λογισμικού: 1) τις υπολογιστικές τεχνικές αναζήτησης και βελτιστοποίησης (Search Based Software Engineering), 2) τις ασαφείς και πιθανολογικές μεθόδους για συλλογισμό με παρουσία αβεβαιότητας, 3) την ταξινόμηση, εκμάθηση και πρόβλεψη. Οι τεχνικές αυτές χρησιμοποιούνται σε τρεις αντίστοιχους τομείς:

- στην “μηχανική λογισμικού με βάση την αναζήτηση” (Search Based Software Engineering), όπου τα προβλήματα μηχανικής λογισμικού επαναδιατυπώνονται σε προβλήματα βελτιστοποίησης για να μπορούν να αντιμετωπιστούν με την υπολογιστική αναζήτηση,

- στην “πιθανολογική μηχανική λογισμικού” (Probabilistic Software Engineering), όπου εφαρμόζονται τεχνικές ΤΝ που διαχειρίζονται ασαφή και πιθανολογικά προβλήματα, όπως για παράδειγμα η πιθανολογική συλλογιστική του Bayes που αφορά τη μοντελοποίηση της αξιοπιστίας του λογισμικού,

- στην “ταξινόμηση, εκμάθηση, πρόβλεψη για την μηχανική λογισμικού” (Classification, Learning and Prediction for Software Engineering), που χρησιμοποιείται στον σχεδιασμό έργων για μοντελοποίηση και πρόβλεψη του κόστους ανάπτυξης.

Ο Wangoo **[101]** εξετάζει την ενσωμάτωση τεχνητής νοημοσύνης μέσω της εξόρυξης δεδομένων στη τεχνολογία λογισμικού, με σκοπό την αυτοματοποιημένη επαναχρησιμοποίηση λογισμικού (Automated Software Reuse). Λόγω του μεγάλου όγκου δεδομένων που παράγεται στα στάδια του κύκλου ζωής ανάπτυξης λογισμικού (SDLC), υπάρχει η ανάγκη αυτοματοποίησης της διαδικασίας, για τη διατήρηση της αποτελεσματικότητας του παραγόμενου λογισμικού. Μέσω της εξόρυξης χρήσιμων δεδομένων, ανακαλύπτονται στοιχεία για παραγωγική επαναχρησιμοποίηση. Τα στοιχεία που επιλέγονται για επαναχρησιμοποίηση είναι υψηλής ποιότητας, χωρίς σφάλματα και αφορούν το πρόβλημα τεχνολογίας λογισμικού που προσεγγίζεται για επίλυση. Έτσι, με τη διαδικασία της επαναχρησιμοποίησης, αυτοματοποιούνται οι εργασίες τεχνολογίας λογισμικού και επιτυγχάνεται μείωση του κόστους και του χρόνου κατασκευής.

Ο Pawar **[102]** εξετάζει τεχνικές ΤΝ που εφαρμόζονται για την αυτοματοποίηση δραστηριοτήτων τεχνολογίας λογισμικού. Περιγράφει την συμβολή της εφαρμογής της τεχνητής νοημοσύνης στα εξής στάδια:

I. στην ανίχνευση απαιτήσεων, όπου εφαρμόζεται η νοημοσύνη Swarm (Swarm Intelligence) με τη χρήση φερομονικής (pheromone) επικοινωνίας εστιάζοντας σε όρους και λέξεις στο κείμενο, με σκοπό την εύρεση σχετικών κειμένων,

II. στην προδιαγραφή απαιτήσεων, όπου μέσω εργαλείων ΤΝ αποσαφηνίζονται οι απαιτήσεις φυσικής γλώσσας, με σκοπό την καλύτερη κατανόηση και την εύρεση λύσεων,

III. στη δημιουργία κώδικα, όπου μέσω προκαθορισμένων πολιτικών και κανόνων ακολουθείται η διαδικασία αυτόνομης δημιουργίας κώδικα λογισμικού (Autonomous Software Code Generation), η οποία εκτελείται από τον πράκτορα ανάπτυξης λογισμικού (Software Developer Agent). Μέσω του πράκτορα γίνεται λήψη αποφάσεων σχετικά με την σχεδίαση και υλοποίηση του λογισμικού, σύμφωνα με τη λογική του συστήματος,

IV. στις δοκιμές λογισμικού, όπου χρησιμοποιούνται εργαλεία ΤΝ για τη δημιουργία, την ανάλυση και διαχείριση δοκιμών,

V. στη δοκιμή GUI (Graphical User Interface), όπου αυτοματοποιείται η διαδικασία δημιουργίας περιπτώσεων δοκιμών, ώστε να αναπαράγονται δοκιμές κάθε φορά που αλλάζει το GUI και να κατασκευάζονται αυτόματα μοντέλα πρόβλεψης συμπεριφοράς της διεπαφής χρήστη,

VI. στην εκτίμηση λογισμικού, όπου εντοπίζονται σφάλματα με τη χρήση μεθόδων υπολογιστικής νοημοσύνης και προβλέπεται / εκτιμάται η ποιότητα λογισμικού.

Οι Feldt et al. **[103]** παρουσιάζουν την τρίπτυχη ταξινόμηση τεχνικών τεχνητής νοημοσύνης που εφαρμόζονται σε συστήματα λογισμικού, σύμφωνα με το σημείο εφαρμογής (Point of Application), τον τύπο τεχνολογίας AI που εφαρμόζεται (Type of AI) και το επίπεδο αυτοματοποίησης που προσφέρεται (Level of Automation). Η πρώτη πτυχή, το σημείο εφαρμογής αφορά το “πότε” και “σε τι” εφαρμόζεται η τεχνολογία AI. Περιλαμβάνει τρία επίπεδα: τα δύο αφορούν το στάδιο πριν από την ανάπτυξη του συστήματος λογισμικού (επίπεδο διαδικασίας και προϊόντος) και το τρίτο αφορά το στάδιο μετά την ανάπτυξη (επίπεδο εκτέλεσης). Το επίπεδο διαδικασίας (process level) υποδεικνύει ότι η ΤΝ εφαρμόζεται στη διαδικασία ανάπτυξης λογισμικού, χωρίς να επηρεάζει τον πηγαίο κώδικα (όπως στην ανάλυση δοκιμών, όπου δεν μεταβάλλεται ο κώδικας). Το επίπεδο προϊόντος (product level) υποδεικνύει ότι η ΤΝ επηρεάζει άμεσα τον πηγαίο κώδικα (όπως στην αυτοματοποιημένη αποσφαλμάτωση προγράμματος, όπου τροποποιείται ο κώδικας για τη διόρθωση σφαλμάτων). Το επίπεδο εκτέλεσης (runtime level) αντιπροσωπεύει εφαρμογές ΤΝ που επηρεάζουν το αναπτυγμένο σύστημα λογισμικού κατά τη διάρκεια εκτέλεσης (όπως η online εκμάθηση βέλτιστων βάσεων δεδομένων με βάση τα δεδομένα που αποθηκεύονται κατά τη λειτουργία). Σχετικά με την δεύτερη πτυχή, δηλαδή τον τύπο τεχνολογίας AI, επειδή δεν υπάρχει κάποιο συγκεκριμένο σύνολο επιπέδων, προτείνεται η ταξινόμηση των “πέντε φυλών ΤΝ (the five tribes of AI)” που παρουσίασε ο Domingos **[104]**, δηλαδή Symbolist (πχ. inverse deduction - αντίστροφη επαγωγή), Connectionist (πχ. backpropagation / αντίστροφη διάδοση), Evolutionaries (πχ. genetic programming - γενετικός προγραμματισμός), Bayesians

(πχ. probabilistic inference - πιθανοτική εκτίμηση / συμπερασματολογία), Analogizers (πχ. kernel machines - μηχανές πυρήνα). Η τρίτη πτυχή αφορά το επίπεδο αυτοματοποίησης, το οποίο στοχεύει ή επιτυγχάνει η εφαρμογή AI. Αυτή η πτυχή βασίζεται στα 10 επίπεδα αυτοματοποίησης Sheridan-Verplanck, η οποία είναι μια ταξινόμηση από την έρευνα Automation/HCI που επικεντρώνεται στην λήψη αποφάσεων ανθρώπου-υπολογιστή. Η ταξινόμηση αυτή εκφράζει το πώς θα πρέπει να συνεργάζονται οι ανθρώπινοι φορείς και το τεχνικό σύστημα μοιράζοντας τον έλεγχο προσδιορισμού και επιλογής για την υλοποίηση εργασιών.

Οι Ammar et al. **[105]** προσπαθούν να καλύψουν το κενό μεταξύ έρευνας και πρακτικής, σχετικά με την εφαρμογή τεχνικών ΤΝ στις διαδικασίες τεχνολογίας λογισμικού, επισημαίνοντας τα ανοιχτά προβλήματα. Εξετάζουν τις τρέχουσες τεχνικές ΤΝ που προτείνονται σε άλλες μελέτες και τις συσχετίζουν με τις διαδικασίες λογισμικού που ορίζονται από το πρότυπο IEEE 12207. Το πρότυπο αυτό καθορίζει ένα πλαίσιο, ορίζει διαδικασίες, δραστηριότητες και καθήκοντα που πρέπει να εφαρμόζονται στις φάσεις του κύκλου ζωής ανάπτυξης λογισμικού. Τα ανοιχτά προβλήματα τεχνολογίας λογισμικού που αναφέρονται, στα οποία μπορούν να βοηθήσουν οι τεχνικές ΤΝ είναι: η αποσαφήνιση απαιτήσεων φυσικής γλώσσας, η ανάπτυξη συστημάτων που βασίζονται στη γνώση και σε οντολογίες για τη διαχείριση των απαιτήσεων και τη μοντελοποίηση των πεδίων, καθώς και η χρήση υπολογιστικής νοημοσύνης για την επίλυση των προβλημάτων έλλειψης και ιεράρχησης των απαιτήσεων.

Οι Barenkamp et al. **[106]**, μέσω της ανάλυσης προηγούμενων ερευνών, κατέληξαν σε κάποιες τεχνικές δυνατότητες που διαθέτει η ΤΝ. Πιο συγκεκριμένα, η ΤΝ επιτυγχάνει την αυτοματοποίηση εργασιών ανάπτυξης και ελέγχου λογισμικού με τη χρήση αλγορίθμων, την δομημένη ανάλυση μεγάλου όγκου δεδομένων, εντοπίζοντας μοτίβα και συστάδες, ενώ μέσω των τεχνητών νευρωνικών δικτύων γίνεται συλλογή, ταξινόμηση και οργάνωση δεδομένων σε σύνολα, με σκοπό την εύρεση λύσεων. Συμβάλλει έτσι, στην επιτάχυνση των διαδικασιών ανάπτυξης λογισμικού, στην εξοικονόμηση πόρων, στην μείωση του κόστους ανάπτυξης, στην αύξηση της αποδοτικότητας και ποιότητας των προϊόντων. Τα αποτελέσματα της επικυρωμένης ανάλυσης των Barenkamp et al. βασίζονται σε συνεντεύξεις εμπειρογνωμόνων και αξιολογούνται ανά στάδιο του κύκλου ζωής ανάπτυξης λογισμικού. Σε αυτή την ανάλυση συμπεραίνεται ότι η ΤΝ μπορεί να υποστηρίξει και τα 6 στάδια του κύκλου με τη χρήση των τεχνικών λύσεων που προσφέρει.

Οι Shehab et al. **[108]** μελέτησαν ερευνητικά άρθρα σχετικά με τις προσεγγίσεις της τεχνητής νοημοσύνης σε δραστηριότητες τεχνολογίας λογισμικού. Ένα από τα ευρήματα τους αφορά το λογισμικό αυτόματης κωδικοποίησης (auto-coding) ονόματι DeepCoders, το οποίο χρησιμοποιεί τεχνητή νοημοσύνη για την ανάπτυξη προγράμματος. Το DeepCoders αναμένεται να εξελιχθεί στα επόμενα χρόνια, καθώς προς το παρόν η συγγραφή του δεν υπερβαίνει τις πέντε γραμμές κώδικα. Αναφέρεται ακόμη πως με τη χρήση ΤΝ μπορούν να αντιμετωπιστούν διάφορα προβλήματα ανάπτυξης λογισμικού, όπως απαιτήσεων, σχεδιασμού, συντήρησης και ελέγχου. Ο αυτοματοποιημένος προγραμματισμός ακόμη, ελαχιστοποιεί την εξειδίκευση, απλοποιεί την συγγραφή και την κατανόηση και παράγει λιγότερα σφάλματα κώδικα. Επιπλέον, μέσω της αυτοματοποίησης που προσφέρει η ΤΝ εξοικονομείται χρόνος και ανθρώπινη προσπάθεια για την διαδικασία ανάπτυξης λογισμικού.

Οι Amershi et al. **[109]** πραγματοποίησαν μια μελέτη παρατηρώντας τις ομάδες λογισμικού της Microsoft, καθώς αναπτύσσουν εφαρμογές λογισμικού με χαρακτηριστικά τεχνητής νοημοσύνης. Εξετάζεται ο τρόπος εργασίας των ομάδων μηχανικών λογισμικού σε μια διαδικασία ροής εργασιών εννέα σταδίων για την ενσωμάτωση της μηχανικής μάθησης στην διαδικασία ανάπτυξης εφαρμογών. Τα δεδομένα συλλέχθηκαν σε δύο φάσεις, από ένα σύνολο συνεντεύξεων από 14 μηχανικούς λογισμικού της Microsoft ενώ με βάση τα αποτελέσματα των συνεντεύξεων διεξήχθη μια ευρείας κλίμακας έρευνα. Διαπιστώθηκε ότι <<η τεχνητή νοημοσύνη χρησιμοποιείται σε παραδοσιακούς τομείς, όπως η αναζήτηση, η διαφήμιση, η αυτόματη μετάφραση, η πρόβλεψη των αγορών των πελατών, η αναγνώριση φωνής και η αναγνώριση εικόνας, αλλά και σε νέους τομείς, όπως ο εντοπισμός πελατών, η παροχή συμβουλών σχεδιασμού για παρουσιάσεις και έγγραφα επεξεργασίας κειμένου, η παροχή μοναδικών χαρακτηριστικών σχεδίασης, η υγειονομική περίθαλψη και η βελτίωση του παιχνιδιού>> Παράλληλα, η μηχανική μάθηση χρησιμοποιείται <<σε έργα υποδομής για τη διαχείριση αναφορών περιστατικών, τον εντοπισμό των πιο πιθανών αιτιών για σφάλματα, την παρακολούθηση της δόλιας φορολογικής δραστηριότητας και την παρακολούθηση ροών δικτύου για παραβιάσεις ασφαλείας.>> Όσον αφορά την κατασκευή των εφαρμογών χρησιμοποιήθηκε ένα ευρύ φάσμα προσεγγίσεων ML << από την ταξινόμηση, την ομαδοποίηση, τον δυναμικό προγραμματισμό και τη στατιστική μέχρι τη μοντελοποίηση της συμπεριφοράς των χρηστών, την ανάλυση κοινωνικών δικτύων και το συνεργατικό φιλτράρισμα.>> Με βάση τα ευρήματα της ανάλυσης παρατηρήθηκαν επίσης τρεις θεμελιώδεις διαφορές με προηγούμενους τομείς εφαρμογών ΤΝ που σχετίζονται με: την ανακάλυψη, διαχείριση δεδομένων, την προσαρμογή, επαναχρησιμοποίηση και την αρθρωτότητα ML.

Ο Xie **[116]** κατηγοριοποιεί τέσσερα προβλήματα τεχνολογίας λογισμικού, για τα οποία υπάρχουν λύσεις τεχνητής νοημοσύνης:

1. Η δημιουργία δοκιμών παραδοσιακά γινόταν με τη χρήση ανθρώπινης νοημοσύνης, σχεδιάζοντας εισόδους δοκιμών που ικανοποιούν απαιτήσεις δοκιμών. Ενώ με τη χρήση μεθόδων ΤΝ, γίνεται αυτόματη παραγωγή δοκιμών χρησιμοποιώντας εργαλεία αυτοματοποίησης που ικανοποιούν απαιτήσεις δοκιμών.

2. Ο καθορισμός προδιαγραφών παραδοσιακά γινόταν με ανθρώπινη νοημοσύνη, ενώ με τη χρήση της ΤΝ γίνεται αυτόματη εξαγωγή προδιαγραφών.

3. Η αποσφαλμάτωση παραδοσιακά γινόταν με την ανθρώπινη ικανότητα εντοπισμού και διόρθωσης ελαττωματικού κώδικα, ενώ πλέον χρησιμοποιείται αυτόματη αποσφαλμάτωση με εργαλεία ΤΝ.

4. Η συγγραφή προγραμμάτων για την υλοποίηση λειτουργιών, πλέον γίνεται μέσω αυτόματων προγραμμάτων συγγραφής.

Συνεπώς, καταλήγει πως οι ανθρώπινες ικανότητες αποτελούν αφετηρία και πηγή έμπνευσης για τον σχεδιασμό τεχνικών αυτοματοποίησης ΤΝ.

Οι Meziane και Vadera **[120]** περιγράφουν τις εξής χρήσεις της ΤΝ στις διαδικασίες ανάπτυξης λογισμικού: συστήματα βασισμένα στη γνώση, νευρωνικά δίκτυα, γενετικοί αλγόριθμοι, συλλογισμός με βάση περιπτώσεις και τρόπους χρήσης AI στη μηχανική απαιτήσεων. Αναλυτικότερα:

I. Με βάση τον συλλογισμό ότι με την απόκτηση εμπειρίας σε προηγούμενα έργα, βελτιώνεται η ικανότητα σχεδίασης νέων έργων, προκύπτει μια προσέγγιση συστημάτων που βασίζονται στη γνώση (Knowledge Based Systems - KBS), τα οποία χρησιμοποιούνται για τον σχεδιασμό έργων ανάπτυξης λογισμικού.

II. Τα νευρωνικά δίκτυα (Neural Networks) χρησιμοποιούνται από την τεχνολογία λογισμικού για πρόβλεψη αποτελεσμάτων, πρόβλεψη σφαλμάτων και γενικότερα για την ταξινόμηση δεδομένων με προγνωστικά χαρακτηριστικά εισόδου.

III. Οι γενετικοί αλγόριθμοι (Genetic Algorithms) χρησιμοποιούνται για τον προγραμματισμό των έργων ανάπτυξης λογισμικού, όπως η δημιουργία βέλτιστων χρονοδιαγραμμάτων και η ανάθεση εργασιών.

IV. Ο συλλογισμός με βάση προηγούμενες περιπτώσεις (Case Based Reasoning), συμβάλει στον επιτυχημένο σχεδιασμό, προγραμματισμό και διαχείριση έργων ανάπτυξης λογισμικού. Για την ανάπτυξη ενός νέου έργου, ανακτώνται δεδομένα από προηγούμενες περιπτώσεις, με τη χρήση μεθόδων εξόρυξης δεδομένων, με σκοπό την καθοδήγηση σχεδιασμού του έργου.

V. Τα εργαλεία ΤΝ χρησιμοποιούνται για την αποσαφήνιση των απαιτήσεων της φυσικής γλώσσας, τα συστήματα που βασίζονται σε γνώση και οντολογίες χρησιμοποιούνται για τη διαχείριση των απαιτήσεων και η υπολογιστική νοημοσύνη χρησιμοποιείται για την επίλυση προβλημάτων απαιτήσεων.

Συνοψίζοντας, στα ευρήματα από σχετικές μελέτες υπήρχαν αρκετά κοινά στοιχεία, που σχετίζονταν με τους τρόπους εφαρμογής της τεχνητής νοημοσύνης και με τα οφέλη που προσφέρουν στις διαδικασίες ανάπτυξης λογισμικού. Οι τρόποι εφαρμογής των τεχνικών ΤΝ που εντοπίστηκαν ήταν: μέσω συστημάτων που βασίζονται στη γνώση, μέσω γενετικών αλγορίθμων, μέσω μηχανικής μάθησης και μέσω τεχνητών νευρωνικών δικτύων. Με τη χρήση των τεχνικών αυτών επιτυγχάνεται ο βέλτιστος σχεδιασμός και η μοντελοποίηση έργων, η ανίχνευση και διαχείριση των απαιτήσεων, η ταξινόμηση και οργάνωση των δεδομένων σε σύνολα, η αυτόματη παραγωγή κώδικα, η πρόβλεψη και ο εντοπισμός σφαλμάτων, η συντήρηση και ο αυτοματοποιημένος έλεγχος/δοκιμή λογισμικού. Εν ολίγοις, οι τεχνικές τεχνητής νοημοσύνης μπορούν να υποστηρίξουν και να βελτιστοποιήσουν όλα τα στάδια του κύκλου ζωής ανάπτυξης λογισμικού, προσφέροντας αποτελεσματικές και αυτοματοποιημένες λύσεις.

4.1.2 Συστήματα/Πλαίσια που εφαρμόζουν την συνέργεια των δύο τομέων

Παρακάτω παρουσιάζονται συστήματα/πλαίσια με ενσωματωμένη τεχνητή νοημοσύνη που εντοπίστηκαν στη σχετική βιβλιογραφία. Αναλύεται ο τρόπος λειτουργίας τους, καθώς και ο τρόπος αλληλεπίδρασης και συνεισφοράς της τεχνητής νοημοσύνης στην τεχνολογία λογισμικού μέσω αυτών των συστημάτων.

Οι Vashisht et al. **[107]** προτείνουν ένα πλαίσιο βασισμένο σε τεχνητά νευρωνικά δίκτυα, το οποίο προβλέπει ελαττώματα λογισμικού σε όλο τον κύκλο ανάπτυξης. Το πλαίσιο είναι διαδραστικό, καθώς περιλαμβάνει μια γραφική διεπαφή χρήστη που επιτρέπει την τροφοδότηση δεδομένων εισόδου για την πρόβλεψη ενός εύρους ελαττωμάτων. Η λήψη δεδομένων γίνεται από ολοκληρωμένα έργα χρησιμοποιώντας το μοντέλο ανάπτυξης καταρράκτη. Στο πλαίσιο που κατασκευάστηκε, τα τεχνητά νευρωνικά δίκτυα χρησιμοποιούν το σύνολο δεδομένων που συλλέχθηκε ως δεδομένα εκπαίδευσης για την εκπαίδευση του πλαισίου. Σκοπός του είναι η πρόβλεψη ελαττωμάτων σε νέα έργα. Στο πλαίσιο αυτό οι συνιστώσες εισόδου για το δίκτυο ελέγχουν τις προσπάθειες παραγωγής, πρόληψης, επανεπεξεργασίας και αναθεώρησης. Τα δεδομένα εισάγονται από τους χρήστες στις φάσεις του κύκλου ανάπτυξης λογισμικού και αφού ελεγχθεί αν πληρούν το εύρος επιλέξιμων κριτηρίων, τότε εξετάζονται με το σύστημα εκτίμησης των τεχνητών νευρικών δικτύων. Οι Vashisht et al. τεκμηριώνουν την αποτελεσματικότητα του πλαισίου με τη χρήση 15 έργων πραγματικού χρόνου και συμπεραίνουν ότι τα πραγματικά ελαττώματα είναι εντός του εύρους ελαττωμάτων. Παράλληλα, επισημαίνουν ότι το πλαίσιο έχει επιτύχει την μεγαλύτερη ακρίβεια με ποσοστό έως 90% συγκριτικά με άλλα παρόμοια πλαίσια.

Ο Rahmaniar **[114]** παρουσιάζει ευκαιρίες και προκλήσεις που προκύπτουν από τη συνέργεια του ChatGPT με την τεχνολογία λογισμικού. Το ChatGPT διαθέτει πολλές δυνατότητες, όπως σύνθεση περιεχομένου, ερωτηματικές ανταποκρίσεις με τη μορφή συνομιλίας, και παραγωγή λογισμικού. Συμβάλει στην επιτάχυνση της διαδικασίας κωδικοποίησης, καθώς έχει την ικανότητα να παράγει μεγάλα αποσπάσματα κώδικα κατά παραγγελία. Μπορεί να δημιουργήσει αυτόματη τεκμηρίωση κώδικα με σχόλια που βασίζονται στη δομή και λειτουργικότητα του. Μπορεί επίσης να εκτελέσει αναθεωρήσεις κώδικα, ανιχνεύοντας σφάλματα και να προτείνει τρόπους επίλυσης και βελτιστοποίησης. Αναφέρεται ωστόσο, πως υπάρχουν ηθικές ανησυχίες σχετικά με την ηθική του κώδικα που παράγεται από μοντέλα ΤΝ και πως σε μελλοντικές εκδόσεις του ChatGPT πρέπει να ενσωματωθούν ηθικά χαρακτηριστικά για τη διασφάλιση ότι το παραγόμενο λογισμικό διαθέτει πρότυπα απορρήτου, ασφάλειας και δικαιοσύνης. Προκύπτουν και οικονομικές ανησυχίες, καθώς ενδέχεται να καλυφθούν θέσεις εργασίας προγραμματιστών ή να προκύψουν νέοι ρόλοι με καθήκοντα διαχείρισης, προσαρμογής, εποπτείας των μοντέλων AI και NLP. Άλλη ανησυχία αφορά την ασφάλεια και το ενδεχόμενο ενσωμάτωσης κομματιού κώδικα που περιέχει κακόβουλο λογισμικό. Εκτός από το ChatGPT προτείνονται και άλλα χρήσιμα εργαλεία ΤΝ που εξυπηρετούν μηχανικούς λογισμικού όπως το CodeBERT, Codex, BERT for Code Search και το DeepCode.

Η διδακτορική έρευνα του Islam **[111]** προτείνει την ανάπτυξη ενός πλαισίου μηχανικής διαδικασιών (process engineering) για την αντιμετώπιση των ηθικών προβληματισμών (όπως

ζητημάτων διαφάνειας, δικαιοσύνης και ιδιωτικότητας), που προκύπτουν στην διάρκεια του κύκλου ζωής ανάπτυξης λογισμικού για συστήματα ΤΝ. Επισημαίνεται, λοιπόν, πως απαιτούνται διαδικασίες, εργαλεία και μέθοδοι ΤΛ για να αναπτυχθούν αξιόπιστα συστήματα ΤΝ, που θα διαθέτουν διαφάνεια επιπτώσεων στους ανθρώπους και στο περιβάλλον. Το μεθοδολογικό πλαίσιο που προτείνεται θα διαθέτει πολιτικές σχεδιασμού που θα στοχεύουν στην υπεύθυνη ανάπτυξη συστημάτων ΤΝ. Η ενσωμάτωση των ηθικών αξιών θα γίνεται στην αρχική φάση σχεδιασμού, δηλαδή κατά το στάδιο της προδιαγραφής απαιτήσεων του συστήματος, θα διαχειρίζονται παράλληλα και οι ηθικές απαιτήσεις. Το πλαίσιο αυτό θα μετατρέπει τις μεθόδους προώθησης κοινωνικών, οικονομικών, πολιτιστικών αξιών σε λειτουργικές προδιαγραφές του συστήματος και μετά την ανάλυση, θα χαρτογραφούνται με το SDLC. Αυτά τα συστήματα μηχανικής ΤΝ θα διατηρούν διαφανή διαδικασία ανάπτυξης και θα επαληθευτούν / επικυρωθούν με επίσημες μεθόδους μελλοντικά.

Οι Bosch et al. **[110]** κατηγοριοποιούν τρεις τομείς εφαρμογών ML/DL:

I. Τα φυσικά συστήματα στον κυβερνοχώρο (Cyber physical systems), τα οποία ταξινομούνται σε τρεις υπολογιστικές πλατφόρμες: τη συσκευή άκρης (edge device), στην οποία συλλέγονται τα δεδομένα για την ML/DL, έναν τοπικό διακομιστή (on-premise server) κάποιου είδους και την υποδομή στο νέφος (cloud).

II. Τα συστήματα κρίσιμης ασφάλειας (Safety-critical systems), αφορούν αυτά που η δυσλειτουργία ή αποτυχία τους μπορεί προκαλέσει σωματικές, περιβαλλοντικές ή οικονομικές βλάβες. Ένα τέτοιο παράδειγμα συστήματος κρίσιμης ασφάλειας με τεχνολογία ML/DL είναι τα αυτοκίνητα αυτόνομης οδήγησης.

III. Τα αυτόνομα βελτιωμένα συστήματα (Autonomously improving systems), τα οποία χρησιμοποιούν μοντέλα ML/DL για τη βελτίωση της απόδοσης του συστήματος, μέσω μηχανισμών πειραματισμού και βελτίωσης χωρίς την παρέμβαση ανθρώπινης ενέργειας. Αυτό επιτυγχάνεται με το μοντέλο ML/DL, μέσω της ανάλυσης δεδομένων και της εκπαίδευσης με βάση αυτά.

Ο Jain **[112]** ερευνά ένα πλαίσιο με τρόπους αλληλεπίδρασης των δύο τομέων, το οποίο περιλαμβάνει τέσσερις κύριες κατηγορίες:

I. το περιβάλλον υποστήριξης λογισμικού (Software support system): αυτή η κατηγορία περιλαμβάνει εργασίες μείωσης της πολυπλοκότητας ανάπτυξης λογισμικού για τον μηχανικό λογισμικού μέσω υπολογιστικού συστήματος, το οποίο εκτελεί τη περισσότερη εργασία ανάπτυξης.

II. τα εργαλεία και οι τεχνικές ΤΝ στο συμβατικό λογισμικό: σε αυτή τη κατηγορία περιγράφεται πως οι στρατηγικές τεχνολογίας λογισμικού απαιτούν γνώσεις στο πεδίο εφαρμογής, στην ανάπτυξη λογισμικού και στην ευρετική λήψη αποφάσεων, για αυτό τα συμβατικά συστήματα λογισμικού απαιτούν τη χρήση ισχυρών και αξιόπιστων τεχνικών που προσφέρει η ΤΝ.

III. τη χρήση συμβατικής τεχνολογίας λογισμικού σε συστήματα ΤΝ: σε αυτή τη κατηγορία αναφέρεται πως η ΤΝ μπορεί να εκμεταλλευτεί τον αποδοτικό τρόπο που προσφέρει η ΤΛ για την αποτελεσματική παραγωγή ισχυρών, αξιόπιστων, ποιοτικών λογισμικών ΤΝ.

IV. τις μεθοδολογικές εκτιμήσεις: αυτή η κατηγορία αφορά την αλληλεπίδραση των διαφορετικών μεθοδολογιών ανάπτυξης. Η τεχνολογία λογισμικού περιλαμβάνει τη μηχανική απαιτήσεων, τη σχεδίαση, ανάπτυξη κώδικα, διαχείριση έργων. Οι δραστηριότητες αυτές μπορούν να χρησιμοποιηθούν για τη δημιουργία τεχνητών ευφυών συστημάτων και αντίστοιχα η τεχνητή νοημοσύνη χρησιμοποιεί τεχνικές ανάλυσης δεδομένων, απόκτησης γνώσης, οι οποίες μπορούν να χρησιμοποιηθούν για συστήματα λογισμικού.

Επομένως, με τη παραπάνω μελέτη των συστημάτων συνέργειας αναδεικνύεται μέσω πρακτικών εφαρμογών ο τρόπος ενίσχυσης, βελτιστοποίησης και τεκμηριώνεται η αποτελεσματικότητα της συνέργειας των δύο τομέων. Οι συνδυασμένες προσεγγίσεις που εντοπίστηκαν και ο σκοπός εξυπηρέτησης τους συνοψίζεται ως εξής:

- το πλαίσιο των Vashisht et al. **[107]** χρησιμοποιείται για πρόβλεψη ελαττωμάτων λογισμικού,

- το Chatgpt χρησιμοποιείται για παραγωγή, τεκμηρίωση κώδικα και ανίχνευση σφαλμάτων,

- τα φυσικά συστήματα στον κυβερνοχώρο, τα συστήματα κρίσιμης ασφάλειας, τα αυτόνομα βελτιωμένα συστήματα, χρησιμοποιούν μηχανική/βαθιά μάθηση με σκοπό την αυτοματοποίηση και βελτίωση τους,

- στη τελευταία μελέτη (του Jain **[112]**) παρουσιάζεται ένα γενικότερο πλαίσιο αλληλεπίδρασης της τεχνητής νοημοσύνης και τεχνολογίας λογισμικού για την βελτίωση διαφόρων διαδικασιών των δύο τομέων.

4.1.3 Προσεγγίσεις/πρακτικές της ΤΛ για συστήματα βασισμένα στην ΤΝ

Η παραδοσιακή ανάπτυξη λογισμικού αποτελεί μια σταθερή, διαδοχική διαδικασία, χρησιμοποιεί τυπικές μεθόδους και επικεντρώνεται στην ανάπτυξη συστημάτων με προκαθορισμένες λειτουργίες και ικανότητες. Ωστόσο, η διαδικασία ανάπτυξης λογισμικού τεχνητής νοημοσύνης αποτελεί μια επαναληπτική και ευέλικτη διαδικασία, που βασίζεται στη συλλογή και ανάλυση δεδομένων. Σκοπεύει στην ανάπτυξη ευφυών συστημάτων, τα οποία εκπαιδεύονται από τα δεδομένα, προσαρμόζονται στο περιβάλλον και διαθέτουν αυτοματοποιημένες λειτουργίες. Η τεχνολογία λογισμικού προσεγγίζει τα συστήματα τεχνητής νοημοσύνης με διάφορες πρακτικές και εφαρμογές, με σκοπό να βελτιστοποιήσει τη λειτουργία και την απόδοση των συστημάτων. Οι πρακτικές που εντοπίστηκαν και αναλύθηκαν στη σχετική βιβλιογραφία αφορούν:

- την βελτιστοποίηση των αλγορίθμων μηχανικής μάθησης, μειώνοντας έτσι τον απαιτούμενο χρόνο εκπαίδευσης.

- την αξιολόγηση, την επαλήθευση των μοντέλων, αλγορίθμων τεχνητής νοημοσύνης μέσω μετρικών ποιότητας, απόδοσης (όπως τη μέτρηση χρόνου εκτέλεσης του μοντέλου) και ακρίβειας πρόβλεψης.

- την βελτιστοποίηση και την αυτοματοποίηση επαναληπτικών διαδικασιών εκπαίδευσης και αξιολόγησης των μοντέλων AI, επιτυγχάνοντας εξοικονόμηση χρόνου και πόρων του συστήματος.

- τη χρήση τεχνικών δοκιμών στα συστήματα, στοχεύοντας στην επιθεώρηση του λογισμικού AI και τη χρήση τεχνικών αποσφαλμάτωσης για τον εντοπισμό και διόρθωση σφαλμάτων.

Οι Shehab et al. **[108]** αναφέρουν πως οι μέθοδοι βαθιάς μάθησης είναι ευρέως χρησιμοποιούμενες από διάφορες εφαρμογές, όπως ευφυείς μηχανές, εξόρυξης δεδομένων, επεξεργασίας εικόνας, εξόρυξης κειμένου, τεχνικές επεξεργασίας ομιλίας και θεραπευτικής διάγνωσης. Υποστηρίζουν πως αυτή η επιτυχία των μεθόδων DL οφείλεται στα βαθιά νευρωνικά δίκτυα. Ωστόσο, η απαίτηση γνώσεων για τον έλεγχο του μαύρου κουτιού (black box testing) αφήνει ανοιχτά ζητήματα ασφάλειας και προστασίας, με αποτέλεσμα να χρειάζονται τεχνικές και συστήματα ΤΛ για τη βελτίωση της ποιότητας. Με βάση σχετικές μελέτες που ερευνήθηκαν, συμπεραίνεται πως με τη χρήση της τεχνολογίας λογισμικού μπορούν να προσδιοριστούν οι δυσκολίες, οι αδυναμίες και να διασφαλιστεί η ποιότητα λογισμικού ΤΝ, επιτυγχάνοντας έτσι ασφαλή συστηματα DL.

Ο Xie **[113]** καταγράφει πως σύμφωνα με ερευνητικές προσπάθειες εξετάζονται λύσεις μηχανικής λογισμικού για τη βελτίωση της παραγωγικότητας, της ανάπτυξης και της αξιοπιστίας λογισμικού ΤΝ. Η εξέταση αφορά την ευφυή τεχνολογία λογισμικού (intelligent software engineering) και επικεντρώνεται σε δύο πτυχές:

I. την ενσωμάτωση ευφυίας σε λύσεις για την αντιμετώπιση προβλημάτων τεχνολογίας λογισμικού, όπως με τη δημιουργία εργαλείων αυτοματοποίησης,

II. την παροχή λύσεων τεχνολογίας λογισμικού για ευφυές λογισμικό, καθώς η έλλειψη ασφάλειας και ελέγχου περιορίζει το πεδίο εφαρμογής του, για αυτό χρειάζονται τεχνικές δοκιμής για τη διασφάλιση της αξιοπιστίας.

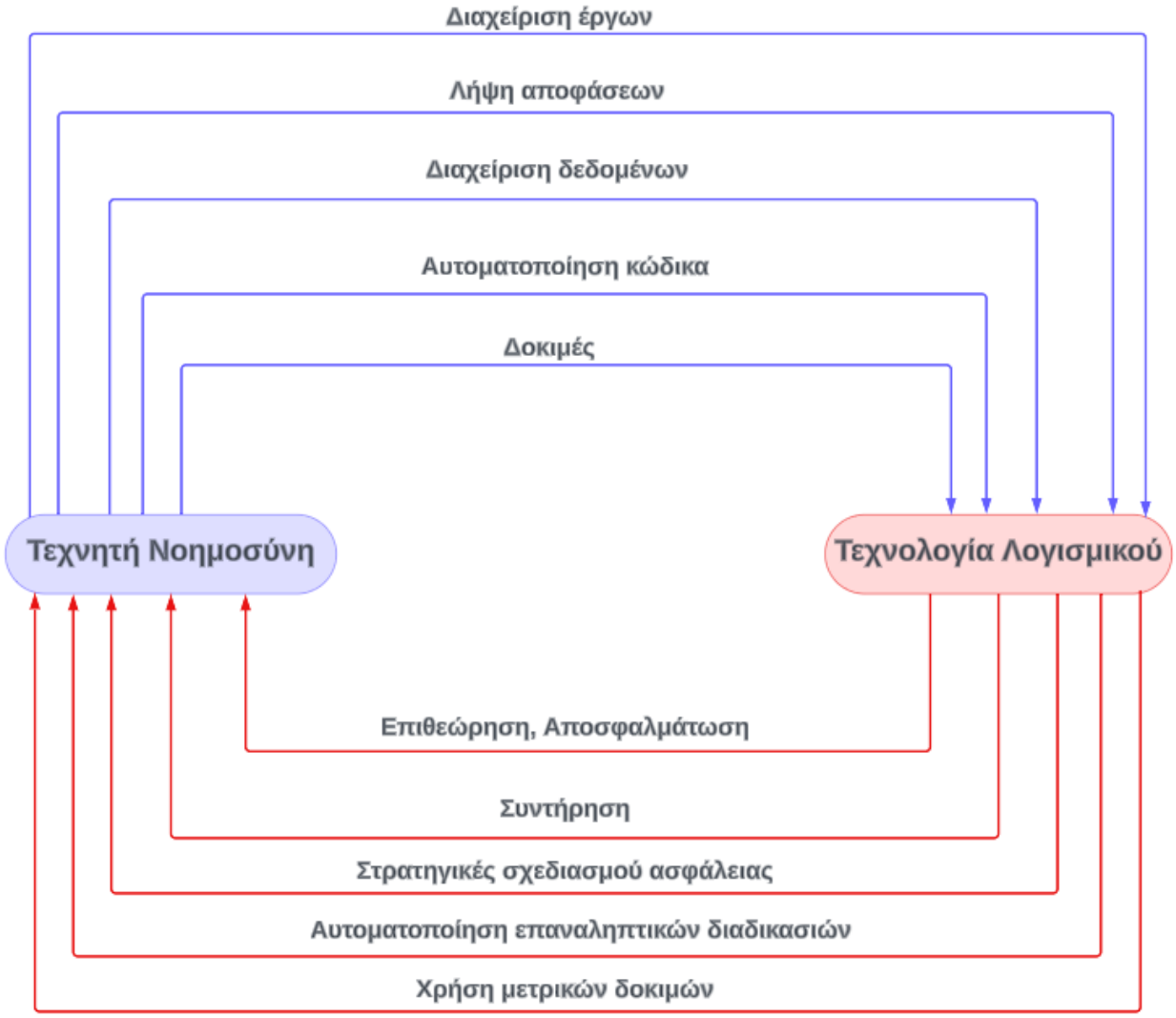
Οι Martinez-Fernandez et al. **[115]** διεξήγαγαν μια επισκόπηση της βιβλιογραφίας και αποτύπωσαν τις προσεγγίσεις τεχνολογίας λογισμικού σε συστήματα βασισμένα στη τεχνητή νοημοσύνη. Σύμφωνα με 17 μελέτες που αφορούσαν στις απαιτήσεις λογισμικού, συμπεραίνεται πως οι περισσότερες από αυτές εστιάζουν μόνο στην διαδικασία των μη λειτουργικών απαιτήσεων για την ΤΝ (Non Functional Requirements for AI), κυρίως για χαρακτηριστικά ποιότητας, ενώ άλλες εστιάζουν στην προδιαγραφή απαιτήσεων (Requirements Engineering). Σύμφωνα με 34 μελέτες που αφορούσαν στον σχεδιασμό

λογισμικού, πολλές από αυτές επικεντρώνονται στις τεχνικές σχεδιασμού ΤΝ για τη βελτίωση ποιότητας των συστημάτων που βασίζονται σε ΤΝ, ενώ άλλες προτείνουν υποδομές, πρότυπα σχεδιασμού ή αρχιτεκτονικές. Σχετικά με την κατασκευή λογισμικού εξετάσθηκαν 23 μελέτες, πολλές εκ των οποίων προσέφεραν εξειδικευμένα εργαλεία για την υποστήριξη ανάπτυξης συστημάτων ΤΝ, ενώ άλλες παρουσίασαν πρακτικές κατασκευής συστημάτων ΤΝ, όπως η χρήση στοιχείων ML για ανίχνευση και διόρθωση σφαλμάτων. Σχετικά με τις δοκιμές λογισμικού για ΤΝ, σύμφωνα με 115 μελέτες, γίνεται εστίαση στον τομέα δημιουργίας περιπτώσεων δοκιμών. Οι περισσότερες μελέτες εφαρμόζουν μεθόδους ελέγχου με τη χρήση διαφόρων τεχνικών (όπως concolic testing) και άλλες σχετίζονται με την διερεύνηση μετρικών δοκιμών για τη μέτρηση της ποιότητας δοκιμών, προτείνοντας κριτήρια κάλυψης. Σχετικά με τη συντήρηση λογισμικού (6 μελέτες), δίνεται έμφαση στον αντίκτυπο της πρόβλεψης σφαλμάτων σε λογισμικό ΤΝ, ενώ παρέχονται επίσης εξειδικευμένες προσεγγίσεις και εργαλεία για την αποσφαλμάτωση λογισμικού ML.

Οι Nascimento et al. **[119]** εξετάζουν πρακτικές τεχνολογίας λογισμικού για την αντιμετώπιση των προκλήσεων της μηχανικής μάθησης. Οι πρακτικές ΤΛ που αναφέρονται για να υποστηρίξουν τις δοκιμές συστημάτων ML αφορούν την ανίχνευση σφαλμάτων, δοκιμές καταπόνησης και ειδικές δοκιμές για νευρωνικά δίκτυα/βαθιά μάθηση. Στις πρακτικές για τη διαχείριση διαμόρφωσης συγκαταλέγονται ο πειραματισμός, η αυτοματοποίηση, οι στρατηγικές για συγκεκριμένους τομείς όπως για ηθικές πτυχές/διαφάνεια/λογοδοσία, οι μέθοδοι επαναχρησιμοποίησης, και η βελτίωση σταθερότητας μέσω αρχιτεκτονικού προτύπου. Οι πρακτικές υποστήριξης του σχεδιασμού συστημάτων ML αποτελούν προτάσεις αρχιτεκτονικού πλαισίου για αυτοματοποιημένο συνεχή πειραματισμό, προτάσεις αρχιτεκτονικών προτύπων για βελτίωση σταθερότητας, προτάσεις στρατηγικών σχεδιασμού ασφάλειας και άλλων ειδικών συστημάτων.

Συνεπώς, η τεχνολογία λογισμικού παρέχει το πλαίσιο (όπως βιβλιοθήκες, εργαλεία ανάπτυξης κώδικα) για τη δημιουργία και υλοποίηση των μοντέλων, των αλγορίθμων τεχνητής νοημοσύνης και μπορεί να την υποστηρίξει με διάφορους τρόπους. Συμβάλει στη συνεχή εξέλιξη των τεχνικών της μέσω βελτιωτικών προσεγγίσεων, με σκοπό τη βελτιστοποίηση των αλγορίθμων και της απόδοσης των μοντέλων ΤΝ, αυτοματοποιώντας επαναληπτικές διαδικασίες. Η τεχνολογία λογισμικού παρέχει τα εργαλεία και τους μηχανισμούς για συνεχή συντήρηση και έλεγχο ασφάλειας των συστημάτων τεχνητής νοημοσύνης, ανιχνεύοντας και διορθώνοντας σφάλματα. Ακόμη, εφαρμόζει διάφορες τεχνικές δοκιμές για την εκτίμηση και αξιολόγηση των συστημάτων ΤΝ.

4.2 Αποτελέσματα Συνέργειας



Εικόνα 13 - Χάρτης συνεργειών

Στην παρούσα ενότητα βρίσκει απάντηση το ερευνητικό ερώτημα 4, καθώς η αλληλεπίδραση των δύο τομέων αναδεικνύεται μέσω του χάρτη συνεργειών και της ανάλυσης των αποτελεσμάτων της συνέργειας που γίνεται παρακάτω.

Οι συνεισφορές της τεχνητής νοημοσύνης στην τεχνολογία λογισμικού που εντοπίστηκαν συνοψίζονται ως εξής:

- Διαχείριση έργων:
  - διαχείριση απαιτήσεων: η προδιαγραφή απαιτήσεων γίνεται μέσω αποσαφήνισης των απαιτήσεων φυσικής γλώσσας χρησιμοποιώντας τεχνικές NLP, ώστε να επιτευχθεί η κατανόηση των απαιτήσεων/αναγκών του χρήστη,

- ➢ προγραμματισμός έργων: με τη χρήση γενετικών αλγορίθμων επιτυγχάνεται ο προγραμματισμός του έργου, όπως με τη δημιουργία βέλτιστων χρονοδιαγραμμάτων, με τη διαχείριση πόρων και την ανάθεση εργασιών.

- ❖ Λήψη αποφάσεων: με την ανάπτυξη συστημάτων που βασίζονται σε γνώση και οντολογίες, ανακτώνται χρήσιμα δεδομένα (μέσω μεθόδων εξόρυξης γνώσης), με σκοπό την βέλτιστη λήψη αποφάσεων και την επίλυση πολύπλοκων προβλημάτων.

- ❖ Διαχείριση δεδομένων: μέσω νευρωνικών δικτύων επιτυγχάνεται η συλλογή, ταξινόμηση και οργάνωση δεδομένων σε σύνολα με σκοπό την δομημένη ανάλυση τους για πρόβλεψη (αποτελεσμάτων, σφαλμάτων) και για εύρεση λύσεων.

- ❖ Αυτοματοποίηση κώδικα:

  - ➢ χρησιμοποιώντας μοντέλα μηχανικής μάθησης, γίνεται αυτόματος εντοπισμός και διόρθωση σφαλμάτων,

  - ➢ με εργαλεία ΤΝ (όπως το ChatGPT) μπορεί να γίνει αυτόματη συγγραφή, τεκμηρίωση και επεξήγηση κώδικα, παρέχοντας πληροφορίες σχετικά με τη λειτουργία του.

- ❖ Δοκιμές: με τη χρήση εργαλείων μηχανικής μάθησης γίνεται αυτόματη παραγωγή και ανάλυση δοκιμών λογισμικού και γραφικών διεπαφών χρήστη.

Οι συνεισφορές της τεχνολογίας λογισμικού στην τεχνητή νοημοσύνη που εντοπίστηκαν αφορούσαν:

- ❖ την επιθεώρηση και αποσφαλμάτωση λογισμικού ΤΝ, μέσω εργαλείων ανάλυσης κώδικα,

- ❖ την συντήρηση λογισμικού ΤΝ, μέσω αναβαθμίσεων, ενημερώσεων και άλλων διαδικασιών στα συστήματα,

- ❖ στρατηγικές σχεδιασμού ασφάλειας, με τη χρήση πρακτικών ασφαλείας, ανάλυσης κινδύνου και ευπαθειών, προστασίας από επιθέσεις, με σκοπό τη διασφάλιση της αξιοπιστίας του ευφυούς λογισμικού,

- ❖ την αυτοματοποίηση επαναληπτικών διαδικασιών, όπως αυτόματη επεξεργασία δεδομένων, αυτόματη εκτέλεση αλγορίθμων μηχανικής μάθησης,

- ❖ τη χρήση μετρικών δοκιμών για μέτρηση ποιότητας, απόδοσης και ακρίβειας καθώς και διαφόρων μη λειτουργικών δοκιμών, όπως καταπόνησης, και ειδικών δοκιμών που αφορούν νευρωνικά δίκτυα και βαθιά μάθηση.

Κεφάλαιο 5: Μελλοντική Έρευνα

5.1 Προκλήσεις

Η συνέργεια μεταξύ τεχνητής νοημοσύνης και τεχνολογίας λογισμικού αντιπροσωπεύει έναν ωφέλιμο συνδυασμό που συνδράμει και στις δύο κατευθύνσεις. Ωστόσο, προκύπτουν διάφορες προκλήσεις προς εξέταση. Σε αυτήν την ενότητα βρίσκει απάντηση το ερευνητικό ερώτημα 5, καθώς αναφέρονται οι προκλήσεις και τα προβλήματα που εντοπίστηκαν στη βιβλιογραφία ενώ αναλύονται και τα σχετικά συμπεράσματα που προκύπτουν. Μέσω αυτής της ανάλυσης δίνεται έναυσμα στην ερευνητική κοινότητα για την εύρεση αξιόπιστων λύσεων, με σκοπό την μείωση των αρνητικών επιπτώσεων που επιφέρει η συνέργεια των δύο τομέων.

Η ανάπτυξη συστημάτων τεχνητής νοημοσύνης παράγει μοντέλα που συχνά χαρακτηρίζονται ως “μαύρα κουτιά” (black boxes), καθώς λόγω της πολυπλοκότητας και της απουσίας διαφάνειας, είναι δύσκολο να κατανοηθεί η λειτουργία και η συμπεριφορά τους από τον άνθρωπο. Ορίζονται ως αδιαπέραστα συστήματα, καθώς οι είσοδοι και λειτουργίες τους (όπως ο τρόπος λήψης αποφάσεων και η παραγωγή αποτελεσμάτων), δεν είναι ορατές στον χρήστη. Ένα παράδειγμα μαύρου κουτιού είναι οι αλγόριθμοι μηχανικής μάθησης που χρησιμοποιούνται για την αναγνώριση εικόνων σε ένα σύστημα αυτόματης αναγνώρισης αντικειμένων, όπως αυτοκίνητα. Αποτελεί πρόκληση η επεξήγηση των αποφάσεων που λαμβάνονται, η ερμηνεία της λειτουργίας, καθώς και η συντήρηση αυτών των συστημάτων.

Οι Hutchins et al. **[118]** μέσω βιβλιογραφικής ανασκόπησης εξετάζουν διάφορα κοινωνικά και ηθικά διλήμματα που προκύπτουν από τον σχεδιασμό και την ανάπτυξη συστημάτων τεχνητής νοημοσύνης καθώς και από τη δημιουργία των αυτόνομων συστημάτων (όπως αυτόνομα αυτοκίνητα). Τα ζητήματα που αποτυπώνουν αφορούν τον τρόπο λήψης αποφάσεων και λειτουργίας των συστημάτων σχετικά με το αν συμβαδίζουν με την ηθική, την ασφάλεια, την ιδιωτικότητα και την διαφάνεια. Επιπλέον, παρατηρείται η δυσκολία ορισμού συνεπειών για εγγενώς κακές ενέργειες που προκαλούν βλάβη σε κάποιο άτομο. Επισημαίνεται πως οφείλεται να δημιουργηθούν νόμοι, κανονισμοί και κατευθυντήριες γραμμές, με σκοπό την διασφάλιση ότι τα συστήματα αυτά, τηρούν τους κανόνες της κοινωνίας και ότι λαμβάνουν διαφανείς ηθικές αποφάσεις.

Παρατηρείται λοιπόν έλλειψη ηθικών απαιτήσεων στη διαδικασία σχεδιασμού έργων τεχνητής νοημοσύνης, σχετικά με το αν πληρούνται πρότυπα δεοντολογίας, ασφάλειας, και προστασίας της ιδιωτικής ζωής. Εντοπίζεται, ακόμη, κίνδυνος ενσωμάτωσης προκαταλήψεων στα μοντέλα τεχνητής νοημοσύνης, οι οποίες μπορεί να οδηγήσουν σε διακρίσεις και αδικίες στις αποφάσεις που λαμβάνονται. Παράλληλα, ζήτημα αφορά η ρίψη ευθύνης στα συστήματα ΤΝ, για βλάβες ή λάθη που ενδέχεται να προκαλέσουν μέσω αυτόματων λειτουργιών που εκτελούνται. Για αυτό κρίνεται απαραίτητη η ύπαρξη ενός χειριστή-επόπτη για τον έλεγχο του

συστήματος, ο οποίος θα φροντίζει την τήρηση της δικαιοσύνης, της διαφάνειας και της ασφάλειας.

Οι Nascimento et al. **[119]** πραγματοποίησαν βιβλιογραφική έρευνα για να αποτυπώσουν προσεγγίσεις της τεχνολογίας λογισμικού που χρησιμοποιούνται για την ανάπτυξη συστημάτων ΤΝ/ΜΜ και να εντοπίσουν τις σχετικές προκλήσεις. Η κατηγοριοποίηση των προκλήσεων γίνεται ανάλογα με τις πτυχές ανάπτυξης ΤΝ/ΜΜ.

I. Μία πτυχή προκλήσεων αφορά την εκτέλεση δοκιμών, καθώς σύμφωνα με μελέτες οι διαδικασίες εκτέλεσης δοκιμών σε συστήματα ML είναι διαφορετικές και πιο περίπλοκες συγκριτικά με τις παραδοσιακές τεχνικές δοκιμών που χρησιμοποιούνται για συστήματα χωρίς ΤΝ/ΜΜ.

II. Η δεύτερη πτυχή αφορά την διασφάλιση ποιότητας και επεκτασιμότητας λογισμικού AI καθώς και την απαίτηση ηθικής της ΤΝ.

III. Η τρίτη πτυχή σχετίζεται με τη διαχείριση του μεγάλου όγκου δεδομένων, καθώς παρατηρείται υψηλό κόστος για τη συλλογή, επεξεργασία, διαθεσιμότητα και την εξασφάλιση της ποιότητας.

IV. Η τέταρτη πτυχή περιλαμβάνει την κατασκευή του μοντέλου μηχανικής μάθησης, η οποία διαθέτει πολλές προκλήσεις, όπως ο μεγάλος αριθμός εισόδων που απαιτείται, η εφαρμογή ηθικής ΤΝ, και η αβεβαιότητα συμπεριφοράς του μοντέλου.

V. Η πέμπτη πτυχή σχετίζεται με την διαχείριση έργου, η οποία περιλαμβάνει προκλήσεις διαχείρισης πόρων καθώς και απαιτήσεις εργαλείων, βιβλιοθηκών και αλγορίθμων για τη κατασκευή του μοντέλου.

VI. Η έκτη πτυχή σχετίζεται με την υποδομή, δηλαδή την απόκτηση, εγκατάσταση, διαμόρφωση και συντήρηση υποδομής για την ανάπτυξη των συστημάτων ΤΝ/ML. Παράλληλα, εντοπίστηκε στη βιβλιογραφία η έλλειψη εργαλείων, μηχανισμών καταγραφής και παρακολούθησης των διαδικασιών ανάπτυξης ΤΝ.

VII. Η έβδομη πτυχή περιλαμβάνει τη μηχανική απαιτήσεων, στην οποία υπάρχει η δυσκολία κατανόησης των αναγκών των πελατών και διαφόρων απαιτήσεων ποιότητας και προδιαγραφών.

Οι Raj et al. **[117]** περιγράφουν την πρόκληση τεχνολογίας λογισμικού, που σχετίζεται με τη διαχείριση δεδομένων σε συστήματα βαθιάς μάθησης στον βιομηχανικό κλάδο. Αναφέρεται πως λόγω της απαίτησης μεγάλου όγκου δεδομένων για την εκπαίδευση και την ανάπτυξη των μοντέλων βαθιάς μάθησης, προκύπτει πρόβλημα διαχείρισης των δεδομένων. Τα προβλήματα που προκύπτουν από την έρευνα αφορούν την συλλογή, εξερεύνηση, προεπεξεργασία δεδομένων, την προετοιμασία του συνόλου δεδομένων, την δοκιμή, την

ανάπτυξη, και το στάδιο μετά την ανάπτυξη. Έτσι, τονίζεται πως πρέπει να ορίζονται πρακτικές διαχείρισης δεδομένων σε όλη τη διάρκεια ανάπτυξης των μοντέλων βαθιάς μάθησης.

Άρα, σημαντική πρόκληση αποτελεί η διαχείριση του μεγάλου όγκου δεδομένων που απαιτούν οι αλγόριθμοι ΤΝ/ΜΜ για εκπαίδευση και αποτελεσματική λειτουργία. Καθώς τα συστήματα συλλέγουν μαζικά πληροφορίες για διάφορους τομείς, με σκοπό την επεξεργασία για δημιουργία μοτίβων και προτύπων προς εξαγωγή συμπερασμάτων, δεν επιτυγχάνεται ποιοτικός έλεγχος και αποθήκευση του συνόλου των δεδομένων. Παρατηρούνται γενικότερα ζητήματα διατήρησης του απορρήτου, διασφάλισης της προστασίας των προσωπικών δεδομένων και της ιδιωτικότητας των πληροφοριών των χρηστών.

Σύμφωνα με μελέτη που διεξήγαγαν οι Bosch et al. **[110]** σε περισσότερες από δώδεκα διεθνείς εταιρείες, οι οποίες έχουν υιοθετήσει και ενσωματώσει στοιχεία ML/DL στα συστήματα τους, παρατήρησαν ότι τα στοιχεία αυτά αποφέρουν κάποιες τεχνικές προκλήσεις. Οι προκλήσεις σχετίζονται με την διαχείριση ποιότητας των δεδομένων, με τις μεθόδους και διαδικασίες σχεδιασμού, με την πρότυπη απόδοση, καθώς και με την ανάπτυξη και συμμόρφωση των μοντέλων. Σχετικά με τη διαχείριση ποιότητας, παρατηρείται πρόβλημα στη δημιουργία συνόλων και ροών δεδομένων με επαρκή ποιότητα, ώστε να επιτευχθεί η εκπαίδευση και η εξαγωγή συμπερασμάτων. Σχετικά με τις μεθόδους και διαδικασίες σχεδιασμού, παρατηρείται πως ενώ η δημιουργία του μοντέλου ML είναι εύκολη διαδικασία, η εφαρμογή του σε κλίμακα και με επαναλαμβανόμενο τρόπο αποτελεί πρόκληση. Η πρότυπη απόδοση των μοντέλων ML/DL, αποτελεί πρόκληση καθώς η διασφάλιση της ακρίβειας της εξαρτάται από πολλούς παράγοντες. Προβλήματα που μπορεί να προκύψουν και να επηρεάσουν την απόδοση αποτελούν η έλλειψη υποστήριξης για ποιοτικά χαρακτηριστικά και η υπερβολική προσαρμογή των μοντέλων. Σχετικά με την ανάπτυξη και συμμόρφωση των μοντέλων, παρατηρούνται διάφορες προκλήσεις, οι οποίες αφορούν την παρακολούθηση και καταγραφή των μοντέλων, την δοκιμή και διάφορες άλλες. Μέσω της καταγραφής των προκλήσεων, σκοπός των Bosch et al. είναι να παρακινήσουν την ερευνητική κοινότητα, ώστε να αντιμετωπιστούν οι προκλήσεις της μηχανικής ΤΝ και να αναπτυχθεί μια νέα, δομημένη τεχνική προσέγγιση για την κατασκευή και την εξέλιξη συστημάτων που περιέχουν στοιχεία ML/DL.

Σύμφωνα με τους Barenkamp et al. **[106]**, ο ρόλος των κλασικών μηχανικών λογισμικού θα μπορούσε να είναι περιττός μελλοντικά, καθώς οι μηχανές θα μπορούν να αναλάβουν τα καθήκοντά τους. Η ΤΝ θα είναι ικανή να παράγει κώδικα και να επιλύει προβλήματα υπολογιστή χωρίς να απαιτείται κάποιος ανθρώπινος παράγοντας. Παράλληλα, όμως, όσοι μηχανικοί αξιοποιήσουν την τεχνολογία ΤΝ, θα ενισχύσουν τις δεξιότητες τους με τη χρήση των αποτελεσματικών εργαλείων ΤΝ και θα εξοικονομούν χρόνο εργασίας μέσω των αυτοματοποιημένων λύσεων που προσφέρονται για εργασίες ρουτίνας σε κάθε στάδιο του κύκλου ζωής της ανάπτυξης λογισμικού.

Προκύπτουν, λοιπόν, οικονομικές ανησυχίες καθώς ενδέχεται να εκλείψουν θέσεις εργασίας λόγω των αυτοματοποιημένων διαδικασιών (όπως ανάπτυξης κώδικα), που προσφέρει η τεχνητή νοημοσύνη. Με την αυτοματοποίηση αυτή, απαξιώνονται οι δεξιότητες μηχανικών λογισμικού, καθώς προτιμάται η ταχύτερη εκτέλεση αυτοματοποιημένων διαδικασιών από συστήματα ΤΝ, οι οποίες οδηγούν σε δημιουργικότερα και αποδοτικότερα αποτελέσματα.

Από την άλλη μεριά, η ενσωμάτωση τμημάτων κώδικα που παράχθηκε από τεχνητή νοημοσύνη σε ένα πρόγραμμα λογισμικού αποτελεί πρόκληση, καθώς μπορεί να περιλαμβάνει σφάλματα, ελλείψεις, οι οποίες μπορεί να επηρεάσουν αρνητικά τη λειτουργία και συμπεριφορά του συστήματος. Επίσης, ο έτοιμος κώδικας μπορεί να αλλάξει το τελικό αποτέλεσμα, το οποίο ενδέχεται να μην ικανοποιεί τις απαιτήσεις/ανάγκες των χρηστών. Ακόμη, ενδέχεται να δημιουργηθούν νέες ευπάθειες ή ακόμη και να εμπεριέχεται κακόβουλο λογισμικό, το οποίο θα προκαλέσει ζημία στο σύστημα. Επομένως, εδώ τονίζεται η ανάγκη εφαρμογής εποπτικού ρόλου από τους μηχανικούς λογισμικού στην επιθεώρηση και ενσωμάτωση έτοιμου κώδικα σε έργα λογισμικού ώστε να εξασφαλιστεί η ικανοποίηση των απαιτήσεων που έχουν τεθεί και άρα η βιωσιμότητα και καταλληλότητα του λογισμικού υπό ανάπτυξη.

5.2 Μελλοντικές Ερευνητικές Κατευθύνσεις

Η συνέργεια της Τεχνητής Νοημοσύνης και της Τεχνολογίας Λογισμικού διαμορφώνει μελλοντικές καινοτομίες και υπερβαίνει τον παραδοσιακό τρόπο ανάπτυξης λογισμικού. Η τεχνολογική εξέλιξη αναμένεται με πολλές προοπτικές και μελλοντικές κατευθύνσεις που θα εστιάζουν στην ασφάλεια, στην υπεύθυνη χρήση και στη βιωσιμότητα των τεχνολογιών.

Μία σημαντική ερευνητική κατεύθυνση που πρέπει να ακολουθεί αφορά την εύρεση ενός κατάλληλου τρόπου διαχείρισης έργων τεχνητής νοημοσύνης. Παρατηρήθηκε πρόκληση σχετικά με το υψηλό κόστος που απαιτείται για την κατασκευή μοντέλων ΤΝ όσον αφορά τις διαδικασίες διαχείρισης δεδομένων, πόρων, εργαλείων, βιβλιοθηκών και αλγορίθμων. Για αυτό, οφείλεται να ληφθούν κατάλληλα μέτρα ελέγχου και διευθέτησης των έργων για να μετριάσουν αυτό το πρόβλημα. Ένα τέτοιο παράδειγμα θα μπορούσε να είναι η μελλοντική ανάπτυξη εργαλείων επαναχρησιμοποίησης δεδομένων, κώδικα και πόρων που έχουν χρησιμοποιηθεί σε προηγούμενα έργα, ώστε να επιτευχθεί η μείωση του κόστους αλλά και η εξοικονόμηση χρόνου αποφεύγοντας την επανασχεδίαση.

Σχετικά με την πρόκληση της ασφάλειας δεδομένων, κρίνεται αναγκαίος ο μελλοντικός σχεδιασμός συστημάτων με ενσωματωμένες τεχνικές και πρωτόκολλα που προάγουν την ασφάλεια, την ιδιωτικότητα και την διαφάνεια. Απαιτείται λοιπόν:

- η χρήση κατάλληλων μέτρων ασφαλείας για τον έλεγχο και τον περιορισμό της πρόσβασης σε προσωπικά και ευαίσθητα δεδομένα,
- η κρυπτογράφηση των πληροφοριών, με σκοπό την ασφαλή αποθήκευση και μεταφορά των δεδομένων,
- η παροχή διαφάνειας σχετικά με την χρήση και επεξεργασία προσωπικών δεδομένων.

Έτσι, ο νέος σχεδιασμός μοντέλων διαχείρισης δεδομένων θα ενισχύει την εμπιστοσύνη των χρηστών και θα μειώσει τα προβλήματα που προκύπτουν.

Μελλοντικά θα δοθεί ιδιαίτερη έμφαση στην ηθική και υπεύθυνη εφαρμογή τεχνητής νοημοσύνης, με σκοπό τη διαφάνεια και εποπτεία στον τρόπο λειτουργίας, λήψης αποφάσεων και δικαιοσύνης, σεβόμενη τα ανθρώπινα δικαιώματα και αξίες. Με την ενσωμάτωση ηθικών κανονισμών που θα συμβαδίζουν με τις κοινωνικές αξίες και την ισχύουσα νομοθεσία, θα διασφαλιστεί η προστασία της ιδιωτικής ζωή και των προσωπικών δεδομένων. Η εποπτεία των συστημάτων ΤΝ θα μπορούσε να επιτευχθεί μελλοντικά μέσω ελεγκτικών μηχανισμών λογισμικού, που θα καταγράφουν και θα παρακολουθούν την συμμόρφωση με το ηθικό πλαίσιο που θα προβλέπεται.

Παρά τις δυνατότητες αυτοματοποίησης και της ταχύτερης εκτέλεσης διαδικασιών που προσφέρει η ΤΝ σε συστήματα λογισμικού, δεν μπορεί να αντικαταστήσει τον ρόλο του προγραμματιστή. Ωστόσο, αυτός ο ρόλος ενδέχεται να τροποποιήσει τα καθήκοντά του στο μέλλον, με σκοπό να ανταποκρίνεται στις απαιτήσεις των συστημάτων ΤΝ, αλλά και να τις κατευθύνει. Μέσω της συνεργασίας αυτής, οι επαναλαμβανόμενες και χρονοβόρες εργασίες θα εκτελούνται αυτόματα με συστήματα ΤΝ, ενώ οι μηχανικοί λογισμικού θα έχουν τον ρόλο του επόπτη και ελεγκτή, βελτιώνοντας έτσι την παραγωγικότητα, την αποτελεσματικότητα και εξοικονομώντας χρόνο εργασίας. Συνεπώς, μελλοντικά θα τροποποιηθούν οι ευθύνες και αρμοδιότητες των μηχανικών λογισμικού και θα απαιτούνται εξελιγμένες δεξιότητες, ευελιξία, καθώς και συνεχής μάθηση για την εξοικείωση με τις λειτουργίες AI.

Το νέο πλαίσιο συνέργειας αποτελεί έναν ψηφιακό μετασχηματισμό με καινοτόμες πρακτικές ασφάλειας, συντήρησης και εξέλιξης των ευφυών συστημάτων λογισμικού. Σημαντική είναι η μελλοντική ενσωμάτωση πρακτικών ασφαλείας με σκοπό τον έλεγχο των συστημάτων τεχνητής νοημοσύνης για τον εντοπισμό τρωτών σημείων και ευπαθειών. Σχετικά με τη συντηρησιμότητα, η τεχνητή νοημοσύνη μπορεί να προσφέρει νέα πρωτόκολλα και βέλτιστες πρακτικές για συνεχή προσαρμογή και ενημέρωση των συστημάτων λογισμικού, εξασφαλίζοντας σταθερή εξέλιξη και δημιουργώντας ανθεκτικό λογισμικό. Ακόμη, ενδέχεται μελλοντικά το λογισμικό να μπορεί να αυτοεξελίσσεται και να αυτοπροσαρμόζεται σε νέες συνθήκες και απαιτήσεις, με τη χρήση τεχνολογιών τεχνητής νοημοσύνης. Αυτό θα επιτυγχάνεται χάρη στην συνεχή εκπαίδευση των μοντέλων σε μεγάλα σύνολα δεδομένων και στη δημιουργία εμπειρίας με βάση αυτά, βελτιώνοντας έτσι τις δεξιότητες και ικανότητες προσαρμογής στο περιβάλλον.

Κεφάλαιο 6: Συμπεράσματα

Σε αυτό το κεφάλαιο θα αναφερθούν τα συμπεράσματα και η συνεισφορά της παρούσας μελέτης, καθώς και μια πιθανή μελλοντική επέκταση της διπλωματικής.

6.1 Συνεισφορά και Συμπεράσματα

Η παρούσα διπλωματική εργασία αποτελεί μια ολοκληρωμένη μελέτη της συνέργειας μεταξύ τεχνητής νοημοσύνης και τεχνολογίας λογισμικού. Παρέχει τεχνικές και πρακτικές που εφαρμόζονται και από τις δύο κατευθύνσεις, με σκοπό να αναδείξει το πώς η τεχνητή νοημοσύνη συνεισφέρει στις διαδικασίες ανάπτυξης λογισμικού αλλά και αντίστροφα, στο πώς η τεχνολογία λογισμικού μπορεί να βελτιστοποιήσει συστήματα/λογισμικά τεχνητής νοημοσύνης. Μάλιστα, αφού διεξήχθη η SLR, σχεδιάστηκε ένας χάρτης συνεργειών ανά κατεύθυνση που απεικονίζει τα ευρήματα της βιβλιογραφικής έρευνας, τα οποία και αναλύθηκαν στη συνέχεια. Προτείνονται ακόμη, κάποια συστήματα που συνδυάζουν τους δύο τομείς, τα οποία εντοπίστηκαν στη βιβλιογραφία ενώ αναλύονται κάποιες προκλήσεις και προβλήματα καθώς και σχετικές ερευνητικές κατευθύνσεις αντιμετώπισής τους.

Τα αποτελέσματα της έρευνας έδειξαν πως η συνέργεια των δύο τομέων, προσφέρει ευρύ φάσμα δυνατοτήτων και λύσεων σε προβλήματα συστημάτων. Ενώ λοιπόν η τεχνητή νοημοσύνη εκσυγχρονίζει και αυτοματοποιεί τις διαδικασίες ανάπτυξης λογισμικού, η τεχνολογία λογισμικού εστιάζει στην επιθεώρηση και εξέλιξη των συστημάτων ΤΝ. Το σύνολο των συνεισφορών αυτής της συνέργειας επιφέρει τα παρακάτω θετικά αποτελέσματα:

- βέλτιστη διαχείριση χρόνου εργασιών και εξοικονόμηση πόρων του συστήματος μέσω του επιτυχούς προγραμματισμού έργων,

- μείωση του κόστους και χρόνου ανάπτυξης, επιτυγχάνοντας ταχύτερη εκτέλεση διαδικασιών μέσω της αυτοματοποίησης (όπως αυτοματοποιημένη λήψη αποφάσεων, αυτοματοποιημένη διόρθωση σφαλμάτων, αυτοματοποιημένος έλεγχος λογισμικού, αυτοματοποιημένες δοκιμές),

- βελτίωση της ποιότητας των συστημάτων προσφέροντας καινοτόμες λύσεις σε προβλήματα, όπως τη διαχείριση μεγάλου όγκου δεδομένων και τη βέλτιστη λήψη αποφάσεων,

- ανάπτυξη αξιόπιστων συστημάτων μέσω της πρόβλεψης καθώς και αντιμετώπισης κινδύνων ασφάλειας και σφαλμάτων κώδικα,

- αύξηση αποδοτικότητας συστημάτων μέσω προηγμένων λειτουργιών (όπως μετρικές δοκιμές), επιτυγχάνοντας μεγαλύτερη ακρίβεια.

Έπειτα από την μελέτη της συνέργειας μεταξύ τεχνητής νοημοσύνης και τεχνολογίας λογισμικού συμπεραίνεται πως προκύπτουν διάφορες προκλήσεις αλλά και νέες προοπτικές για μελλοντικές ερευνητικές κατευθύνσεις στον τομέα του λογισμικού. Επισημαίνεται ότι η άποψη, ότι τα συστήματα τεχνητής νοημοσύνης θα αντικαταστήσουν τους μηχανικούς λογισμικού αποτελεί μύθο, καθώς η παρουσία ενός χειριστή θα συνεχίσει να είναι απαραίτητη. Επίσης, με τα κατάλληλα μέτρα ελέγχου και υπεύθυνης χρήσης μπορεί να διασφαλιστεί η μελλοντική ισορροπία, προσφέροντας μόνο θετική αλληλεπίδραση και εξέλιξη.

6.2 Μελλοντική επέκταση

Η μελλοντική επέκταση της παρούσας εργασίας θα μπορούσε να γίνει μέσω πρακτικής εφαρμογής και μελέτης στατιστικών ευρημάτων σχετικά με τον μετασχηματισμό εκτέλεσης των διαδικασιών ανάπτυξης λογισμικού με τη χρήση μοντέλων τεχνητής νοημοσύνης. Θα μπορούσαν να διεξαχθούν σχετικά πειράματα, ώστε να γίνουν μετρήσεις χρόνου, απόδοσης των συστημάτων και να γίνουν συγκρίσεις με παλαιότερες παραδοσιακές μεθόδους. Ένα τέτοιο παράδειγμα πειράματος θα μπορούσε να είναι η ανάπτυξη της ίδιας εφαρμογής από έναν μηχανικό λογισμικού χωρίς και με την αρωγή ενός chatbot, ώστε να γίνει σύγκριση του παραγόμενου κώδικα των δύο περιπτώσεων ως προς τη ποιότητα (αν περιέχει σφάλματα, αν είναι μεγάλου μήκους), τον χρόνο συγγραφής και άλλα κριτήρια, με σκοπό να αναδειχθεί στη πράξη η χρησιμότητα της τεχνητής νοημοσύνης. Παράλληλα, θα μπορούσαν να γίνουν και άλλες πρακτικές εφαρμογές (και για την αντίστροφη χρησιμότητα, δηλαδή της τεχνολογίας λογισμικού στη τεχνητή νοημοσύνη), ώστε να μελετηθεί εις βάθος η συνέργεια των δύο τομέων.

Βιβλιογραφία

[1] "Artificial intelligence." Wikipedia.
Available:https://en.wikipedia.org/wiki/Artificial_intelligence
[2] "Software engineering." Wikipedia.
Available:https://en.wikipedia.org/wiki/Software_engineering
[3] K. Yasar. "What is software engineering?" TechTarget. Accessed: March, 2023 [Online.]
Available:https://www.techtarget.com/whatis/definition/software-engineering
[4] B.J. Copeland. "Alan Turing and the beginning of AI." Britannica
Available:https://www.britannica.com/technology/artificial-intelligence/Alan-Turing-and-the-beginning-of-AI
[5] Ι. Βλαχάβας, Π. Κεφαλάς, Ν. Βασιλειάδης, Φ. Κόκκορας, Η. Σακελλαρίου. "ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ." Β΄Έκδοση, 2005
[6] D. Stefanidis. "Ιστορία και Εξέλιξη της Τεχνητής Νοημοσύνης – Artificial Intelligence (A.I.)" sapkasgeorge.gr.
Available:https://www.sapkasgeorge.gr/%CE%B9%CF%83%CF%84%CE%BF%CF%81%CE%AF%CE%B1-%CE%BA%CE%B1%CE%B9-%CE%B5%CE%BE%CE%AD%CE%BB%CE%B9%CE%BE%CE%B7-%CF%84%CE%B7%CF%82-%CF%84%CE%B5%CF%87%CE%BD%CE%B7%CF%84%CE%AE%CF%82-%CE%BD%CE%BF%CE%B7%CE%BC/
[7] "Τεχνητή νοημοσύνη." Βικιπαίδεια
Available:https://el.wikipedia.org/wiki/%CE%A4%CE%B5%CF%87%CE%BD%CE%B7%CF%84%CE%AE_%CE%BD%CE%BF%CE%B7%CE%BC%CE%BF%CF%83%CF%8D%CE%BD%CE%B7
[8] "ELIZA." PRONEWS. Accessed: February 22, 2023. [Online.]
Available:https://www.pronews.gr/istoria/eliza-h-efeyresi-pou-tromakse-ton-dimiourgo-tis-to-proto-programma-synomilias-meso-ypologisti-ston-kosmo/
[9] A. Peterson. " How AI Has Advanced During the 21st Century and Where it's Headed" prosapien. Available:https://www.pro-sapien.com/blog/how-ai-has-advanced-during-21st-century-and-where-its-headed/
[10] "IBM Watson." Wikipedia. Available:https://en.wikipedia.org/wiki/IBM_Watson
[11] C. Welch. "Amazon just surprised everyone with a crazy speaker that talks to you." The Verge. Accessed: Nov 6, 2014. [Online.]
Available:https://www.theverge.com/2014/11/6/7167793/amazon-echo-speaker-announced
[12] "Sophia (robot)." Wikipedia. Available:https://en.wikipedia.org/wiki/Sophia_(robot)
[13] L. Plaugic. "Musician Taryn Southern on composing her new album entirely with AI." The Verge. Accessed: Aug 27, 2017. [Online.]
Available:https://www.theverge.com/2017/8/27/16197196/taryn-southern-album-artificial-intelligence-interview
[14] "ChatGPT." Wikipedia. Available:https://el.wikipedia.org/wiki/ChatGPT
[15] N. Duggal. "What is Artificial Intelligence and Why It Matters in 2024? simplilearn. Accessed: Mar 12, 2024. [Online.] Available:https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/what-is-artificial-intelligence
[16] "Technological singularity." Wikipedia.
Available:https://en.wikipedia.org/wiki/Technological_singularity#cite_note-chalmers2010-7

[17] "Vernor Vinge Predicts "The Singularity"" HistoryofInformation.com Available:https://historyofinformation.com/detail.php?entryid=2510
[18] D. Black. "AI singularity: waking nightmare, fool's dream, or an answer to prayers?" cybernews. Accessed: July 05, 2023. [Online.] Available:https://cybernews.com/tech/ai-technological-singularity-explained/
[19] "What is the difference between AI/ML/DL?" HCAI. Available:https://human-centered.ai/2017/11/11/difference-ai-ml/
[20] "What is underfitting and overfitting in machine learning and how to deal with it." medium.com. Accessed: March 11, 2018. [Online.] Available:https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76
[21] "Μηχανική μάθηση." Βικιπαίδεια. Available:https://el.wikipedia.org/wiki/%CE%9C%CE%B7%CF%87%CE%B1%CE%BD%CE%B9%CE%BA%CE%AE_%CE%BC%CE%AC%CE%B8%CE%B7%CF%83%CE%B7
[22] Σ. Ράπτης. "Τι είναι η μηχανική μάθηση (machine learning);" 2science.gr. Accessed: January 11, 2023. [Online.] Available:https://2science.gr/machine-learning-1/
[23] "Τι είναι και πως λειτουργεί η βαθιά μάθηση" iguru,gr. Accessed: September 30, 2021. [Online.] Available: https://iguru.gr/einai-kai-pos-leitourgei-vathia-mathisi/
[24] "what is machine learning", SAP. Available:https://www.sap.com/greece/products/artificial-intelligence/what-is-machine-learning.html
[25] N. Νέλσον. "Τι είναι το Deep Learning;" unite.ai. Accessed: November 28, 2020. [Online.] Available:https://www.unite.ai/el/%CF%84%CE%B9-%CE%B5%CE%AF%CE%BD%CE%B1%CE%B9-%CE%B2%CE%B1%CE%B8%CE%B9%CE%AC-%CE%BC%CE%AC%CE%B8%CE%B7%CF%83%CE%B7/
[26] "Επεξεργασία φυσικής γλώσσας." Βικιπαίδεια. Available:https://el.wikipedia.org/wiki/%CE%95%CF%80%CE%B5%CE%BE%CE%B5%CF%81%CE%B3%CE%B1%CF%83%CE%AF%CE%B1_%CF%86%CF%85%CF%83%CE%B9%CE%BA%CE%AE%CF%82_%CE%B3%CE%BB%CF%8E%CF%83%CF%83%CE%B1%CF%82
[27] S. Krupsky. "Επεξεργασία φυσικής γλώσσας Εισαγωγή: τι είναι η επεξεργασία φυσικής γλώσσας (NLP);" NLP Cloud. Accessed: July 5, 2021. [Online.] Available:https://nlpcloud.com/el/introduction-what-is-nlp-natural-language-processing.html
[28] C. Alexiou. "Τεχνητά νευρωνικά δίκτυα: το μέλλον της υπολογιστικής επιστήμης." techmaniacs.gr. Accessed: September 10, 2016. [Online.] Available:https://techmaniacs.gr/why-neural-networks-are-the-future-of-computing/
[29] "Νευρωνικά Δίκτυα (Neural Networks): Ορισμός & Εφαρμογές." Big Blue Data Academy. Accessed: March 03, 2023. [Online.] Available:https://bigblue.academy/gr/neuronika-diktua
[30] "Βασικά στοιχεία νευρωνικών δικτύων." Elements of AI. Available:https://course.elementsofai.com/el/5/1
[31] M. Mayo. "Neural Network Foundations, Explained: Activation Function" KD nuggets. Accessed: September 13, 2017. [Online.] Available:https://www.kdnuggets.com/2017/09/neural-network-foundations-explained-activation-function.html
[32] "Νευρωνικό δίκτυο." Βικιπαίδεια. Available:https://el.wikipedia.org/wiki/%CE%9D%CE%B5%CF%85%CF%81%CF%89%CE%BD%CE%B9%CE%BA%CF%8C_%CE%B4%CE%AF%CE%BA%CF%84%CF%85%CE%BF
[33] "What is a neural network?" IBM. Available:https://www.ibm.com/topics/neural-networks

[34] "Types of Neural Networks and Definition of Neural Network." Great Learning. Accessed: Nov 23, 2022. [Online.] Available:https://www.mygreatlearning.com/blog/types-of-neural-networks/
[35] H. Ashtari. "What Is a Neural Network? Definition, Working, Types, and Applications in 2022" spiceworks. Accessed: August 3, 2022. [Online.] Available:https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-a-neural-network/
[36] "Ρομποτική." Βικιπαίδεια. Available:https://el.wikipedia.org/wiki/%CE%A1%CE%BF%CE%BC%CF%80%CE%BF%CF%84%CE%B9%CE%BA%CE%AE
[37] Ν. Χασάπη. "Τεχνητή Νοημοσύνη και Ρομποτική." pemptousia.gr. Accessed: Nov. 19, 2021. [Online.] Available:https://www.pemptousia.gr/2021/11/techniti-noimosini-ke-rompotiki/
[38] Μ. Κυδωνάκη. "Η Ιστορία των ρομπότ, η εξέλιξή τους." athinodromio.gr. Accessed: October 19, 2023. [Online.] Available:https://www.athinodromio.gr/%CE%B7-%CE%B9%CF%83%CF%84%CE%BF%CF%81%CE%AF%CE%B1-%CF%84%CF%89%CE%BD-%CF%81%CE%BF%CE%BC%CF%80%CF%8C%CF%84-%CE%B7-%CE%B5%CE%BE%CE%AD%CE%BB%CE%B9%CE%BE%CE%AE-%CF%84%CE%BF%CF%85%CF%82/
[39] Σ. Δρακάκη. "Τι ακριβώς είναι η Ρομποτική." athinodromio.gr. Accessed: Nov. 4, 2016. [Online.] Available:https://www.athinodromio.gr/%CF%84%CE%B9-%CE%B1%CE%BA%CF%81%CE%B9%CE%B2%CF%8E%CF%82-%CE%B5%CE%AF%CE%BD%CE%B1%CE%B9-%CE%B7-%CF%81%CE%BF%CE%BC%CF%80%CE%BF%CF%84%CE%B9%CE%BA%CE%AE/#.WVpxPYiGOM8
[40] "Γνωρίζω τον κόσμο των Ρομπότ." weebly.com. Available:https://4dimkal-robot.weebly.com/alphapi972-tauiota-alphapiomicrontauepsilonlambdaepsilon943taualphaiota-941nualpha-rhoomicronmupi972tau.html
[41] "Robot." Wikipedia. Available:https://en.wikipedia.org/wiki/Robot#Contemporary_uses
[42] "Types of robots: classification, applications and examples." Telefonica. Available:https://www.telefonica.com/en/communication-room/blog/types-of-robots-classification-applications-and-examples/
[43] "What is the characteristic of the fourth generation of robots?" PROBOT Corp. Available:https://www.probotcorp.com/post/what-is-the-characteristic-of-the-fourth-generation-of-robots
[44] F. M. Borges. "The 5 Generations of Robotics." automatismosmundo.com. Accessed: Sept. 3, 2022. [Online.] Available:https://automatismosmundo.com/en/the-5-generations-of-robotics/
[45] "Industrial robot." Wikipedia. Available:https://en.wikipedia.org/wiki/Industrial_robot
[46] "Industrial Robotics." VEX. Available:https://education.vex.com/stemlabs/workcell/stemlab/industrial-robotics/what-are-industrial-robots
[47] "Industrial robots: what they are, how they work, and what types exist." Telefonica. Available:https://www.telefonica.com/en/communication-room/blog/industrial-robots-what-how-work-types/
[48] "Automated guided vehicle." Wikipedia. Available:https://en.wikipedia.org/wiki/Automated_guided_vehicle
[49] "Roomba." Wikipedia. Available:https://en.wikipedia.org/wiki/Roomba

[50] "Unmanned aerial vehicle." Wikipedia.
Available:https://en.wikipedia.org/wiki/Unmanned_aerial_vehicle
[51] "Autonomous underwater vehicle." Wikipedia.
Available:https://en.wikipedia.org/wiki/Autonomous_underwater_vehicle
[52] "Humanoid robot." Wikipedia. Available:https://en.wikipedia.org/wiki/Humanoid_robot
[53] J. Biba. "Top 22 Humanoid Robots in Use Right Now." Built In. Accessed: Mar 01, 2024. [Online.] Available:https://builtin.com/robotics/humanoid-robots
[54] "Service Robots: Humanoid Robots" automate.org.
Available:https://www.automate.org/robotics/service-robots/service-robots-humanoid-robots
[55] "What is Software Engineering? Definition of Software Engineering, Software Engineering Meaning" The Economic Times.
Available:https://economictimes.indiatimes.com/definition/software-engineering
[56] M. Martin. "What is Software Engineering? Definition, Basics, Characteristics" Accessed: February 24, 2024. [Online.] Available:https://www.guru99.com/what-is-software-engineering.html
[57] "Software" Britannica. Available:https://www.britannica.com/technology/software
[58] "Verification and Validation in Software Engineering" GeeksforGeeks.
Available:https://www.geeksforgeeks.org/software-engineering-verification-and-validation/
[59] "Software verification." Wikipedia.
Available:https://en.wikipedia.org/wiki/Software_verification
[60] "Λογισμικό." Βικιπαίδεια.
Available:https://el.wikipedia.org/wiki/%CE%9B%CE%BF%CE%B3%CE%B9%CF%83%CE%BC%CE%B9%CE%BA%CF%8C
[61] "The History of Software Engineering" Institute of Data. Accessed: Sept. 28, 2023. [Online.] Available:https://www.institutedata.com/blog/the-history-of-software-engineering/
[62] K. Juhasz. "The history of coding and software engineering" Galvanize. Accessed: December 9, 2022. [Online.] Available:https://www.galvanize.com/blog/the-history-of-coding-and-software-engineering/
[63] "C (programming language)." Wikipedia.
Available:https://en.wikipedia.org/wiki/C_(programming_language)
[64] "Ada." Wikipedia. Available:https://el.wikipedia.org/wiki/Ada
[65] "Timeline of computing 1980–1989." Wikipedia.
Available:https://en.wikipedia.org/wiki/Timeline_of_computing_1980%E2%80%931989
[66] "C++." Wikipedia. Available:https://en.wikipedia.org/wiki/C%2B%2B
[67] "World Wide Web." Wikipedia. Available:https://en.wikipedia.org/wiki/World_Wide_Web
[68] "History of Linux." Wikipedia. Available:https://en.wikipedia.org/wiki/History_of_Linux
[69] "Η ιστορία των προγραμμάτων περιήγησης" mozilla.org.
Available:https://www.mozilla.org/el/firefox/browsers/browser-history/
[70] "Java." Wikipedia. Available:https://el.wikipedia.org/wiki/Java
[71] "Social media." Wikipedia. Available:https://en.wikipedia.org/wiki/Social_media
[72] "Έξυπνο τηλέφωνο." Βικιπαίδεια.
Available:https://el.wikipedia.org/wiki/%CE%88%CE%BE%CF%85%CF%80%CE%BD%CE%BF_%CF%84%CE%B7%CE%BB%CE%AD%CF%86%CF%89%CE%BD%CE%BF

[73] "History of Cloud Computing." GeeksforGeeks.
Available:https://www.geeksforgeeks.org/history-of-cloud-computing/
[74] "Προσωπικός ψηφιακός οδηγός." Βικιπαίδεια.
Available:https://el.wikipedia.org/wiki/%CE%A0%CF%81%CE%BF%CF%83%CF%89%CF%80%CE%B9%CE%BA%CF%8C%CF%82_%CF%88%CE%B7%CF%86%CE%B9%CE%B1%CE%BA%CF%8C%CF%82_%CE%BF%CE%B4%CE%B7%CE%B3%CF%8C%CF%82
[75] R. Awati. "What is Requirements Analysis?" TechTarget. Accessed: June, 2023. [Online.]
Available:https://www.techtarget.com/searchsoftwarequality/definition/requirements-analysis
[76] "Requirements analysis." Wikipedia.
Available:https://en.wikipedia.org/wiki/Requirements_analysis
[77] "Προδιαγραφή Απαιτήσεων Λογισμικού (SRS): Συμβουλές & Πρότυπο." visure.
Available:https://visuresolutions.com/el/software-requirement-specification-srs-tips-template/
[78] "Requirements Analysis" javatpoint. Available:https://www.javatpoint.com/software-engineering-requirement-analysis
[79] "What Is the Software Development Life Cycle?" LITSLINK. Accessed: Feb. 25, 2023. [Online.]
Available:https://litslink.com/blog/software-development-life-cycle
[80] "Software Design Process – Software Engineering." GeeksforGeeks.
Available:https://www.geeksforgeeks.org/software-engineering-software-design-process/
[81] "Software design." Wikipedia. Available:https://en.wikipedia.org/wiki/Software_design
[82] "Software Design Basics." tutorialspoint.
Available:https://www.tutorialspoint.com/software_engineering/software_design_basics.htm
[83] "What is Software Engineering? History, Key Principles, Best Practices." nexwebsites.com.
Available:https://nexwebsites.com/blog/software-engineering/
[84] https://www.simplilearn.com/tutorials/programming-tutorial/what-is-software-development
[85] "What Is Software Development?" OutSystems.
Available:https://www.outsystems.com/glossary/what-is-software-development/
[86] "Unit testing." Wikipedia. Available:https://en.wikipedia.org/wiki/Unit_testing
[87] "Integration testing." Wikipedia. Available:https://en.wikipedia.org/wiki/Integration_testing
[88] "Software performance testing." Wikipedia.
Available:https://en.wikipedia.org/wiki/Software_performance_testing
[89] "Security testing." Wikipedia. Available:https://en.wikipedia.org/wiki/Security_testing
[90] "Stress testing (software)." Wikipedia.
Available:https://en.wikipedia.org/wiki/Stress_testing_(software)
[91] "Software Maintenance." javatpoint. Available:https://www.javatpoint.com/software-engineering-software-maintenance
[92] J. Gomez. "The Four Types of Software Maintenance." koombea.com. Accessed: Oct. 6, 2023. [Online.] Available:https://www.koombea.com/blog/types-of-software-maintenance/
[93] "What is Software Engineering? History, Key Principles, Best Practices." nexwebsites.com.
Available:https://nexwebsites.com/blog/software-engineering/
[94] "Top 4 software development methodologies." synopsys. Accessed: Mar. 27, 2017. [Online.]
Available:https://www.synopsys.com/blogs/software-security/top-4-software-development-methodologies.html

[95] "Top 12 Software Development Methodologies." TatvaSoft. Accessed: Dec. 25, 2020. [Online.] Available:https://www.tatvasoft.com/blog/top-12-software-development-methodologies-and-its-advantages-disadvantages/

[96] "8 Most Common Software Development Methodologies." upwork. Accessed: July 31, 2023. [Online.] Available:https://www.upwork.com/resources/most-common-software-development-methodologies

[97] A. Oladele. "Top 15 Software Development Methodologies: Benefits and Drawbacks." Velvetech. Accessed: July 27, 2023. [Online.] Available:https://www.velvetech.com/blog/software-development-methodologies/

[98] Pyster, Art, Rick Adcock, Mark Ardis, Rob Cloutier, Devanandham Henry, Linda Laird, Harold 'Bud' Lawson, Michael Pennotti, Kevin Sullivan, και Jon Wade. 'Exploring the Relationship between Systems Engineering and Software Engineering'. Procedia Computer Science 44 (2015): 708–17.

[99] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, 'Systematic Mapping Studies in Software Engineering', presented at the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), Jun. 2008.

[100] M. Harman, 'The role of Artificial Intelligence in Software Engineering', in 2012 First International Workshop on Realizing AI Synergies in Software Engineering (RAISE), Zurich, Switzerland: IEEE, Jun. 2012, pp. 1–6.

[101] D. P. Wangoo, 'Artificial Intelligence Techniques in Software Engineering for Automated Software Reuse and Design', in 2018 4th International Conference on Computing Communication and Automation (ICCCA), Greater Noida, India: IEEE, Dec. 2018, pp. 1–4.

[102] N. Pawar, 'Application of Artificial Intelligence in Software Engineering', Volume 18, Issue 3, Ver. IV, PP 46-51, 2016.

[103] R. Feldt, F. G. de O. Neto, and R. Torkar, 'Ways of Applying Artificial Intelligence in Software Engineering'. arXiv, Feb. 07, 2018.

[104] P. Domingos, The master algorithm: how the quest for the ultimate learning machine will remake our world, First paperback edition. New York: Basic books, a member of the Perseus Book Group, 2018.

[105] H. H Ammar, W. Abdelmoez, M. S. Hamdi, 'Software Engineering Using Artificial Intelligence Techniques: Current State and Open Problems'. 2012.

[106] M. Barenkamp, J. Rebstadt, and O. Thomas, 'Applications of AI in classical software engineering', AI Perspect, vol. 2, no. 1, p. 1, Dec. 2020.

[107] V. Vashisht, M. Lal, and G. S. Sureshchandar, 'A Framework for Software Defect Prediction Using Neural Networks', JSEA, vol. 08, no. 08, pp. 384–394, 2015.

[108] M. Shehab, L. Abualigah, M. I. Jarrah, O. A. Alomari, and M. Sh. Daoud, '(AIAM2019) Artificial Intelligence in Software Engineering and inverse: Review', International Journal of Computer Integrated Manufacturing, vol. 33, no. 10–11, pp. 1129–1144, Nov. 2020.

[109] S. Amershi et al., 'Software Engineering for Machine Learning: A Case Study', in 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), Montreal, QC, Canada: IEEE, May 2019, pp. 291–300.

[110] J. Bosch, I. Crnkovic, and H. H. Olsson, 'Engineering AI Systems: A Research Agenda'. arXiv, Jun. 03, 2020.

[111] Zahoor Ul Islam. 2021. Software Engineering Methods for Responsible Artificial Intelligence. In Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '21). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1814–1815.

[112] Jain, Prince. (2011). Interaction between Software Engineering and Artificial Intelligence-A Review. International Journal on Computer Science and Engineering.

[113] T. Xie, 'Intelligent Software Engineering: Synergy between AI and Software Engineering', in Proceedings of the 11th Innovations in Software Engineering Conference, Hyderabad India: ACM, Feb. 2018, pp. 1–1.

[114] Wahyu Rahmaniar . ChatGPT for Software Development: Opportunities and Challenges. TechRxiv. August 23, 2023. DOI: 10.36227/techrxiv.23993583.v1

[115] Silverio Martínez-Fernández, Justus Bogner, Xavier Franch, Marc Oriol, Julien Siebert, Adam Trendowicz, Anna Maria Vollmer, and Stefan Wagner. 2022. Software Engineering for AI-Based Systems: A Survey. ACM Trans. Softw. Eng. Methodol. 31, 2, Article 37e (April 2022), 59 pages. https://doi.org/10.1145/3487043

[116] T. Xie, "The synergy of human and artificial intelligence in software engineering," *2013 2nd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)*, San Francisco, CA, USA, 2013, pp. 4-6, doi: 10.1109/RAISE.2013.6615197.

[117] A. Munappy, J. Bosch, H. H. Olsson, A. Arpteg and B. Brinne, "Data Management Challenges for Deep Learning," *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Kallithea, Greece, 2019, pp. 140-147, doi: 10.1109/SEAA.2019.00030.

[118] N. Hutchins, Z. Kirkendoll and L. Hook, "Social impacts of ethical artificial intelligence and autonomous system design," *2017 IEEE International Systems Engineering Symposium (ISSE)*, Vienna, Austria, 2017, pp. 1-5, doi: 10.1109/SysEng.2017.8088298.

[119] Nascimento, E.D., Nguyen-Duc, A., Sundbø, I., & Conte, T.U. (2020). Software engineering for artificial intelligence and machine learning software: A systematic literature review. ArXiv, abs/2011.03751.

[120] F. Meziane and S. Vadera, Eds., Artificial intelligence applications for improved software engineering development: new prospects. Hershey, PA: Information Science Reference, 2010.