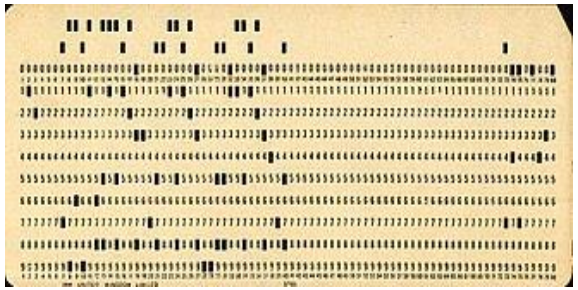
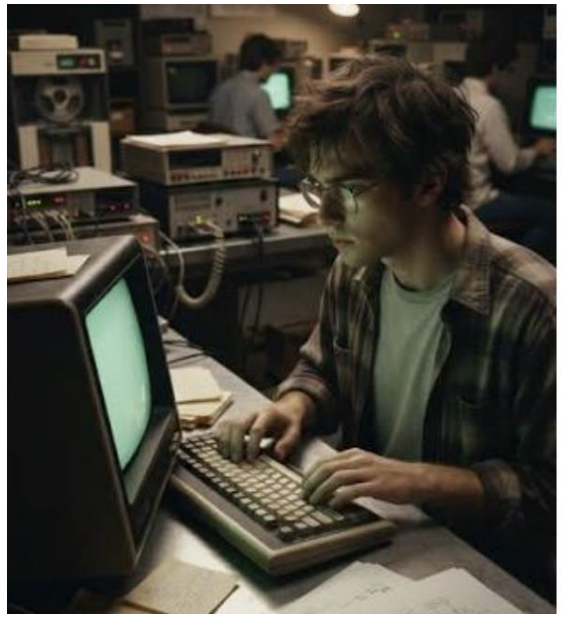
ΤΕΧΝΟΛΟΓΙΑ ΛΟΓΙΣΜΙΚΟΥ





Από την εποχή των διάτρητων καρτών
και της χειρογραφικής συγγραφής κώδικα
στην εποχή της Τεχνητής Νοημοσύνης

Αφροδίτη Τσαλγατίδου



Τι είναι η Τεχνολογία Λογισμικού

Είναι ο κλάδος της Πληροφορικής που

- Ασχολείται με
  - μεθόδους, εργαλεία και τεχνικές

  για τη μελέτη, ανάλυση, σχεδίαση, ανάπτυξη, δοκιμή, συντήρηση και εξέλιξη λογισμικού.

- Έχει στόχο να εξασφαλίσει ότι τα συστήματα λογισμικού είναι
  - αξιόπιστα, αποδοτικά, ασφαλή, επεκτάσιμα και
  - ικανοποιούν τις ανάγκες των χρηστών και των οργανισμών.

Η Εξέλιξη της Ανάπτυξης Λογισμικού

Η Εποχή της Ανθρώπινης Εργασίας (1970 - 2020):

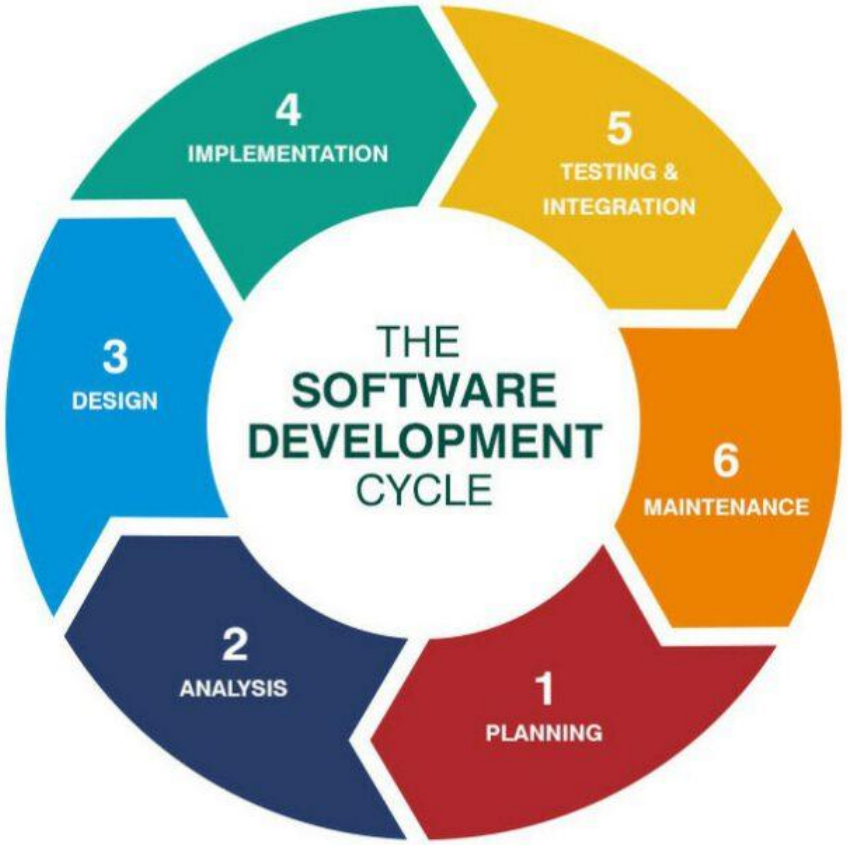
- Από τον πλήρη ανθρώπινο έλεγχο και την πλήρως χειρογραφική συγγραφή κώδικα, σε ημι-αυτόματο έλεγχο και συγγραφή κώδικα

Η Εποχή της Τεχνητής Νοημοσύνης (2021 - σήμερα):

- Auto-complete (Tab-complete), Copilots, chatbots παραγωγής snippet κώδικα.
- Αυτόνομοι AI agents που σχεδιάζουν, εκτελούν και διορθώνουν ολόκληρα features ή repositories.

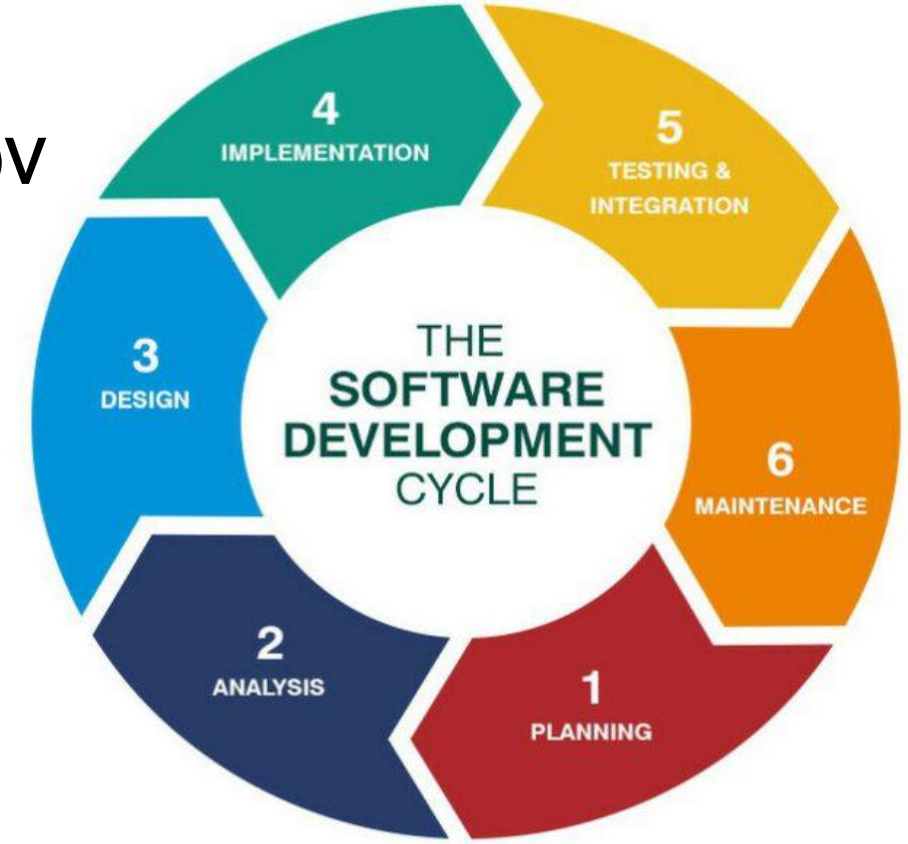
Κύκλος ζωής ανάπτυξης λογισμικού (Software Development Life Cycle SDLC)

- Είναι η διαδικασία που ακολουθούν οι εταιρείες για να παράγουν ποιοτικό λογισμικό με αποτελεσματικό, υποστηρίξιμο και έγκαιρο τρόπο.
- Το SDLC έχει υποστεί πολλές αλλαγές τα τελευταία 50 χρόνια.
- Ωστόσο, εξακολουθεί να αποτελεί ένα βασικό πλαίσιο για την κατανόηση του τρόπου ανάπτυξης και παράδοσης προϊόντων λογισμικού.



Στάδια κύκλου ζωής ανάπτυξης λογισμικού (SDLC)

- Σχεδιασμός έργου (Planning)
- Συλλογή & Ανάλυση Απαιτήσεων
- Σχεδίαση Λογισμικού
- Υλοποίηση Λογισμικού
- Έλεγχος και Ολοκλήρωση
- Συντήρηση



Επομένως, η Τεχνολογία Λογισμικού …

… είναι το

τεχνολογικό και επιστημονικό υπόβαθρο

πίσω από την ανάπτυξη λογισμικού

σε αντίθεση με τον προγραμματισμό που είναι μόνο η γραφή κώδικα.

Με άλλα λόγια, η Τεχνολογία Λογισμικού **δεν είναι μόνο "γράφω κώδικα"**, αφορά όλο τον κύκλο ζωής του λογισμικού, από την ιδέα μέχρι τη συντήρηση και την εξέλιξή του.

Τομείς Τεχνολογίας Λογισμικού

- **Ανάλυση Απαιτήσεων** – καταγραφή και κατανόηση των αναγκών του χρήστη
- **Σχεδίαση Συστήματος** – αρχιτεκτονική λογισμικού, πρότυπα σχεδίασης (design patterns)
- **Προγραμματισμός & Υλοποίηση** – χρήση γλωσσών προγραμματισμού, frameworks, βιβλιοθηκών.
- **Δοκιμές & Ποιότητα** – testing, debugging, διασφάλιση ποιότητας (Quality Assurance - QA).
- **Συντήρηση & Εξέλιξη** – διορθώσεις, αναβαθμίσεις, βελτιώσεις.
- **Διαχείριση Έργων Λογισμικού** – Agile, Scrum, DevOps, CI/CD.
- **Ασφάλεια & Αξιοπιστία** – τεχνικές για ανθεκτικά και ασφαλή συστήματα.

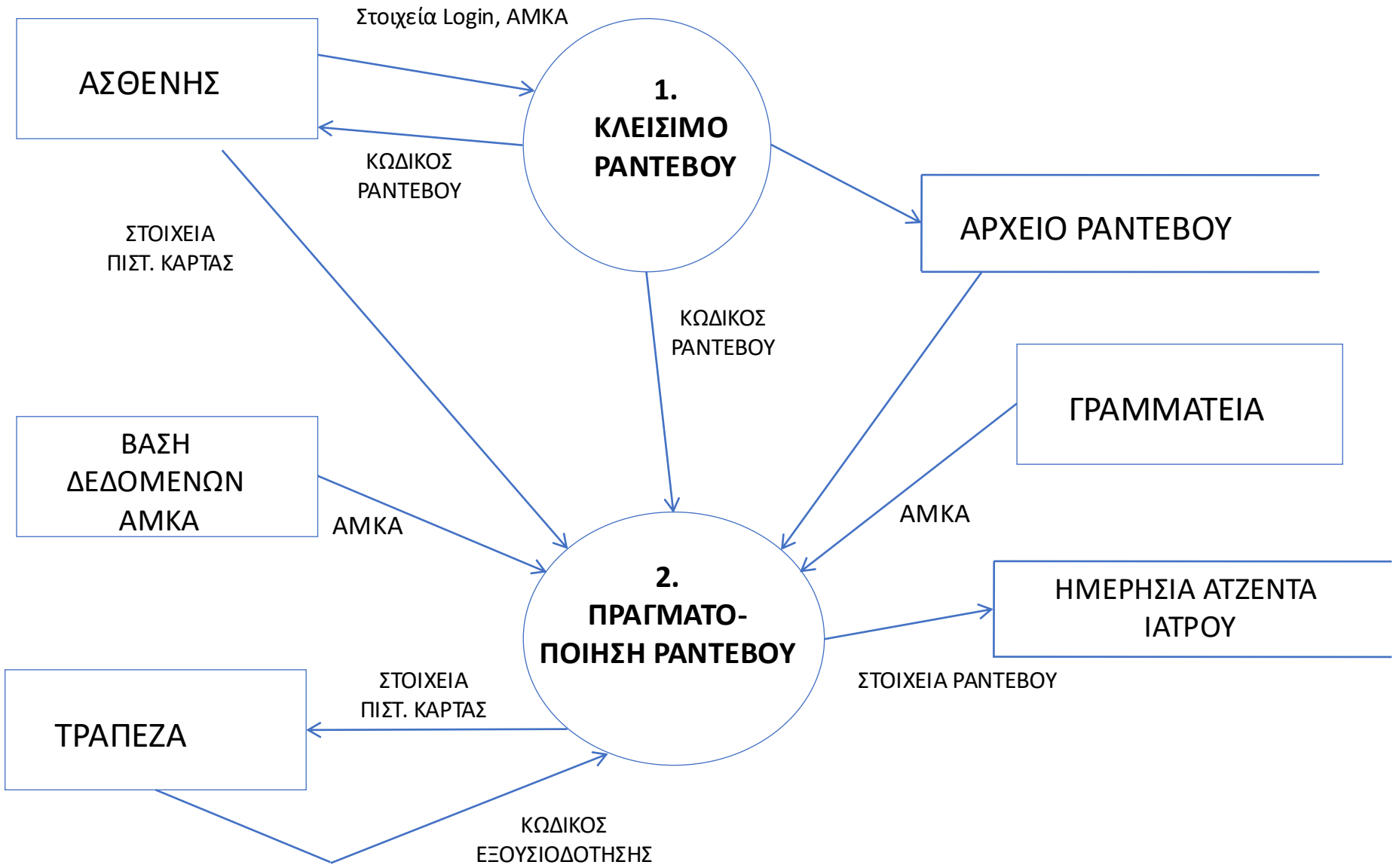
Ανάλυση Απαιτήσεων

Ακολουθούν **Παραδείγματα Διαγραμμάτων** που χρησιμοποιούνται για Μοντελοποίηση Απαιτήσεων

- **Διάγραμμα Ροής Δεδομένων** – Χρησιμοποιείται στη Δομημένη Ανάλυση
- Δύο από τα Διαγράμματα της UML (Unified Modeling Language)
  - **Διάγραμμα Περιπτώσεων Χρήσης**
  - **Διάγραμμα Κλάσεων**

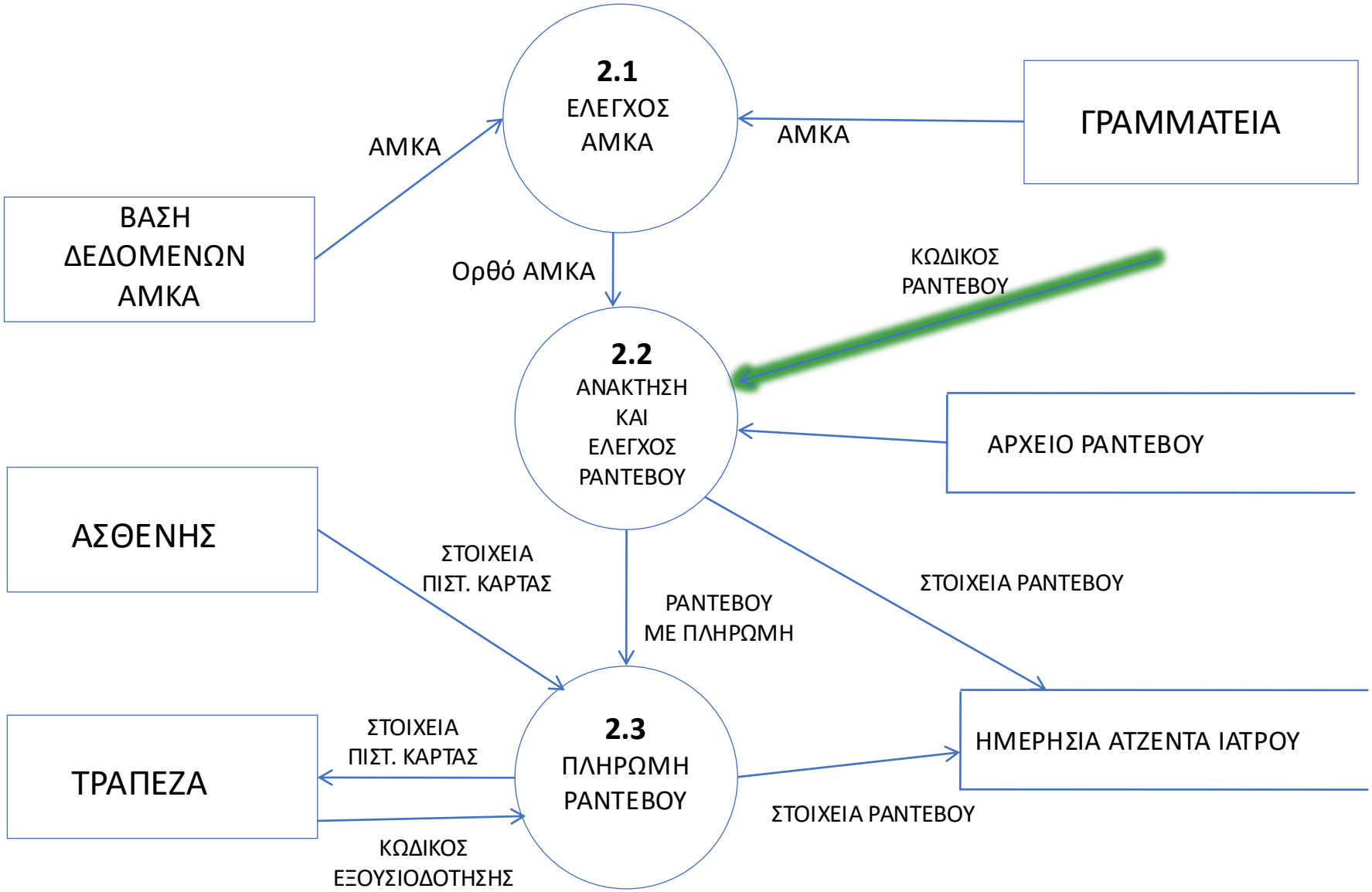
Ανάλυση Απαιτήσεων

Παράδειγμα Χρήσης Διαγράμματος Ροής Δεδομένων (ΔΡΔ)



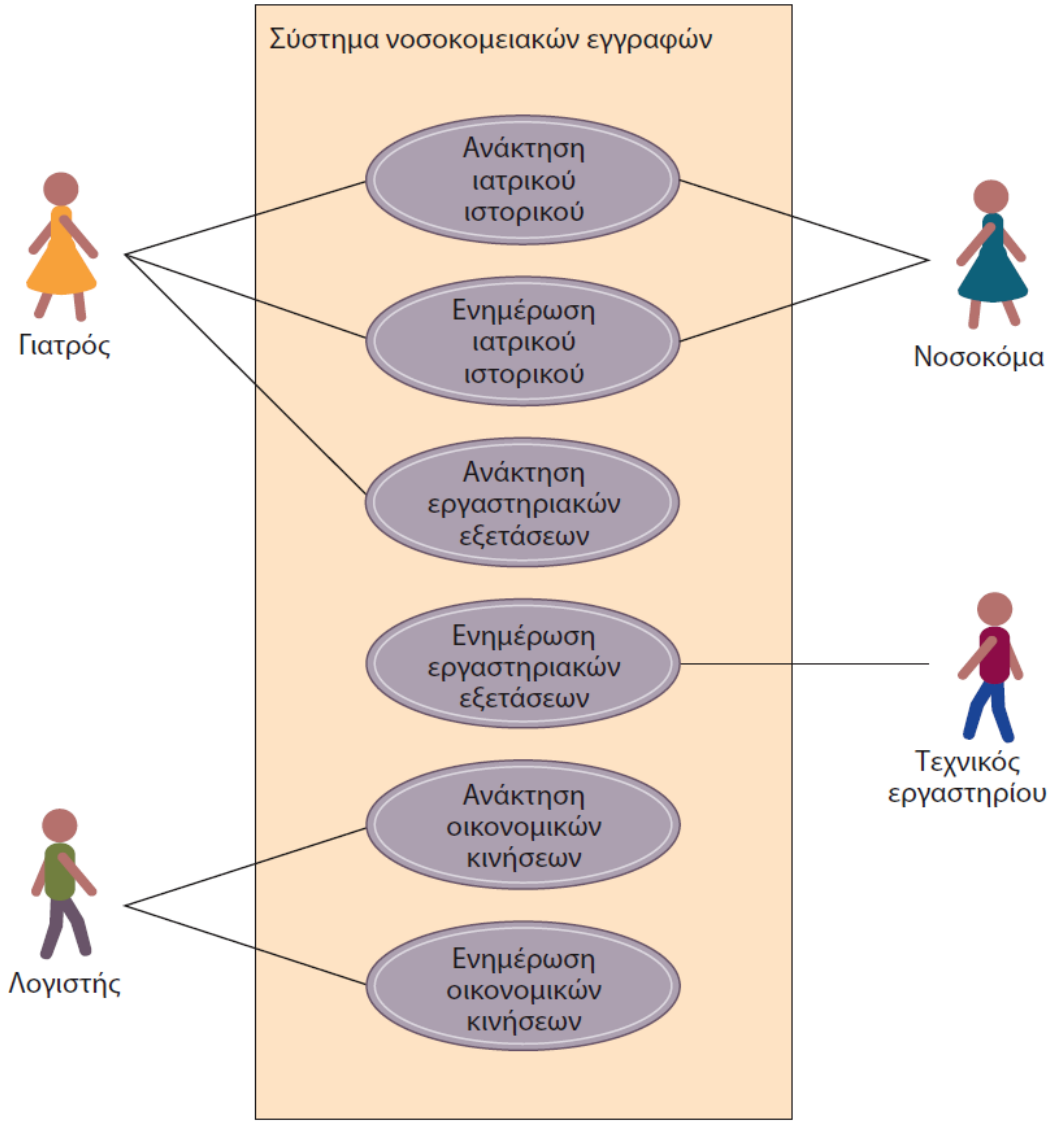
Ανάλυση Απαιτήσεων

Παράδειγμα Αποσύνθεσης της Διαδικασίας «2. Πραγματοποίηση Ραντεβού» του προηγούμενου ΔΡΔ



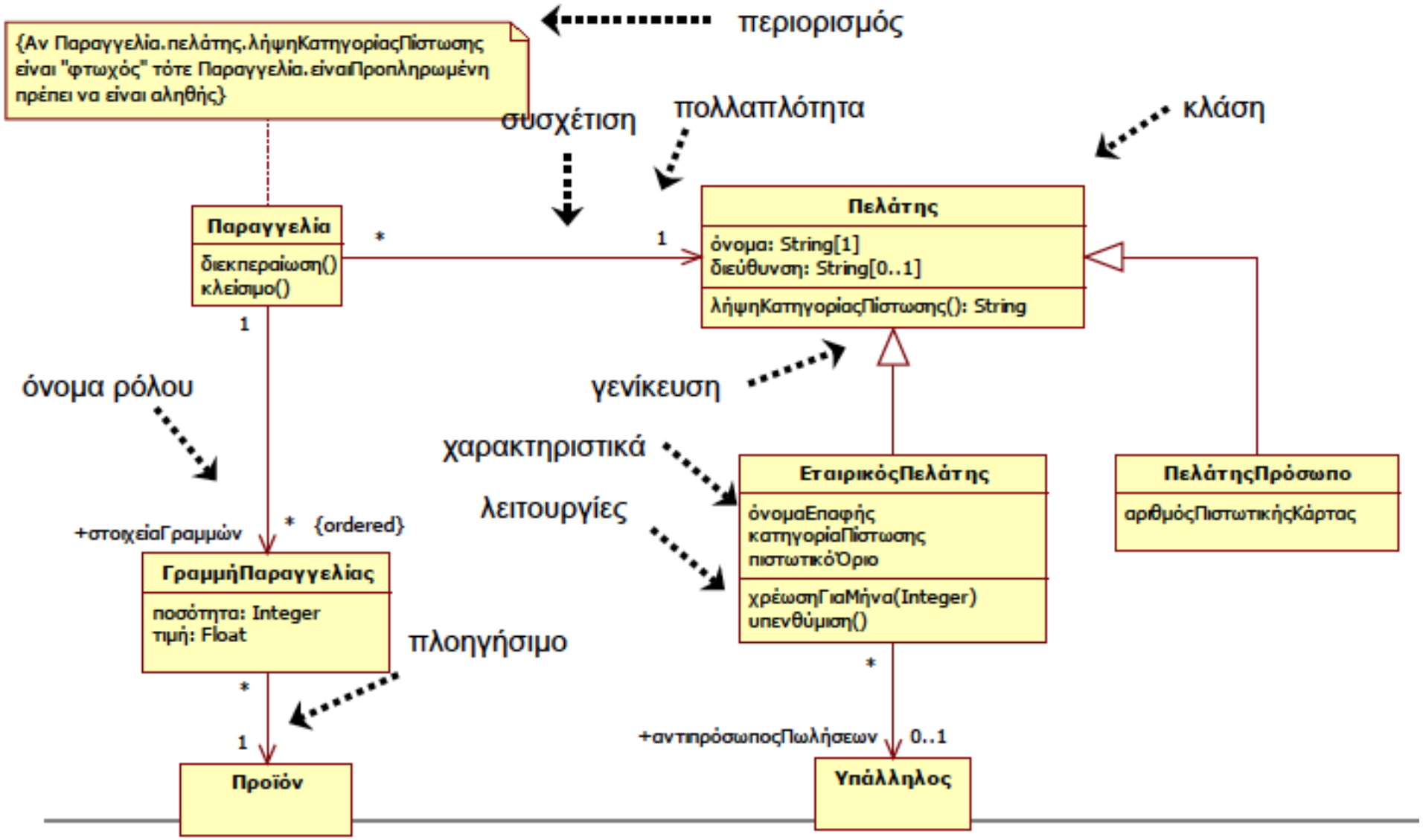
Ανάλυση Απαιτήσεων

UML – Διάγραμμα Περιπτώσεων Χρήσης



Ανάλυση Απαιτήσεων

UML – Διάγραμμα Κλάσεων



CASE Tools



www.educba.com

Εργαλεία CASE
(Computer-Aided Software Engineering)

Τι είναι

- Εφαρμογές λογισμικού για **αυτοματοποίηση και υποστήριξη των δραστηριοτήτων σε όλα τα στάδια του κύκλου ζωής ανάπτυξης λογισμικού (SDLC)**, από την αρχική ανάλυση απαιτήσεων μέχρι τη σχεδίαση, τον προγραμματισμό, τον έλεγχο και τη συντήρηση.
- Εμφανίστηκαν τη δεκαετία του 1980. Όμως, η κεντρική ιδέα παραμένει η ίδια μέχρι σήμερα: **η χρήση λογισμικού για τη δημιουργία καλύτερου λογισμικού.**
- Σήμερα, τα εργαλεία αυτά έχουν μετεξελιχθεί και ενσωματωθεί στα σύγχρονα IDEs, DevOps pipelines και AI εργαλεία.

Εργαλεία CASE
Forward Engineering

- Είναι η παραδοσιακή διαδικασία ανάπτυξης λογισμικού.
- Ξεκινά από ένα υψηλό επίπεδο αφαίρεσης (απαιτήσεις, λογικά σχέδια, αρχιτεκτονική) και κινείται προς τα κάτω, με σκοπό τη δημιουργία του τελικού κώδικα ή της βάσης δεδομένων.
- **Πώς λειτουργεί στα εργαλεία CASE:** Στην αρχή σχεδιάζεται ένα visual διάγραμμα κλάσεων (UML Class Diagram) ή ένα διάγραμμα οντοτήτων-συσχετίσεων (ERD) για μια βάση δεδομένων. Στη συνέχεια, το εργαλείο παράγει αυτόματα τον σκελετό του κώδικα (π.χ. σε Java ή C++) ή το SQL script για να στηθεί η βάση.
- **Το αποτέλεσμα:** Παραγωγή του boilerplate κώδικα και των δομών της εφαρμογής από τα σχέδια.

Εργαλεία CASE

Reverse Engineering

- Κάνει ακριβώς το αντίθετο από το Forward Engineering.
- Ξεκινά από τον ήδη υπάρχοντα πηγαίο κώδικα (source code) ή το εκτελέσιμο αρχείο (binary) και κινείται προς τα πάνω, για να ανακατασκευάσει το σχέδιο, την αρχιτεκτονική και τη λογική της εφαρμογής.
- **Πού χρησιμεύει:**
  - **Κατανόηση Legacy Συστημάτων:** Στα παλιά έργα με χιλιάδες γραμμές κώδικα και μηδενική τεκμηρίωση (documentation), το Reverse Engineering «διαβάζει» τον κώδικα και φτιάχνει αυτόματα τα διαγράμματα που δείχνουν πώς συνδέονται τα αρχεία.
  - **Ανάλυση Κυβερνοασφάλειας (Malware Analysis):** Οι AI Security Specialists παίρνουν έναν ιό (εκτελέσιμο αρχείο .exe) και του κάνουν Reverse Engineering (αποσυναρμολόγηση/decompilation) για να δουν τον κώδικα και να καταλάβουν τι ζημιά κάνει.
- **Πώς λειτουργεί στα εργαλεία CASE:** Παίρνει σαν είσοδο ένα φάκελο με αρχεία κώδικα, και δημιουργεί αυτόματα τα αντίστοιχα visual διαγράμματα.

Εργαλεία CASE
Πλεονεκτήματα/Μειονεκτήματα

Πλεονεκτήματα – Λόγοι δημιουργίας τους	Μειονεκτήματα - Προκλήσεις
Ταχύτητα & Αυτοματοποίηση: Μειώνουν το χρόνο ανάπτυξης, παράγοντας αυτόματα boilerplate κώδικα και σκελετούς εφαρμογών.	**Υψηλό κόστος:** Η αγορά και η συντήρησή τους είναι ακριβή.
Ανώτερη Ποιότητα: Λιγότερα ανθρώπινα λάθη στη σχεδίαση και τη μεταφορά της σε κώδικα.	**Καμπύλη εκμάθησης:** Απαιτούν μεγάλη εκπαίδευση από την ομάδα για να χρησιμοποιηθούν σωστά.
Εύκολη Συντήρηση: Η τεκμηρίωση και τα διαγράμματα είναι πάντα συγχρονισμένα με τον κώδικα	**Έλλειψη ευελιξίας**: Πολλά παλαιότερα εργαλεία ανάγκαζαν τους προγραμματιστές να ακολουθούν αυστηρά και δύσκαμπτα πρότυπα.

Η Εξέλιξη της Ανάπτυξης Λογισμικού

Η Εποχή της Ανθρώπινης Εργασίας (1970 - 2020):

- Από τον πλήρη ανθρώπινο έλεγχο και την πλήρως χειρογραφική συγγραφή κώδικα, σε ημι-αυτόματο έλεγχο και συγγραφή κώδικα

Η Εποχή της Τεχνητής Νοημοσύνης (2021 - σήμερα):

- Auto-complete (Tab-complete), Copilots, chatbots παραγωγής snippet κώδικα.
- Αυτόνομοι AI agents που σχεδιάζουν, εκτελούν και διορθώνουν ολόκληρα features ή repositories.

Η Ραχοκοκαλιά της Σύγχρονης Ανάπτυξης Λογισμικού

- Agile - η **φιλοσοφία**
- Scrum - το **πλαίσιο εφαρμογής**
- DevOps - η **κουλτούρα συνεργασίας** και
- CI/CD - τα **τεχνικά εργαλεία**.

Χρήσης Ευέλικτης Μεθοδολογίας Agile

- Είναι ένας **τρόπος σκέψης και προσέγγισης έργων**.
  - Αντί να σχεδιάζεται ένα τεράστιο λογισμικό για πολλούς μήνες και να παραδίδεται όλο μαζί στο τέλος (παραδοσιακό μοντέλο Waterfall), το Agile σπάει το έργο σε μικρά, διαχειρίσιμα κομμάτια.

- **Κεντρική ιδέα:** Συνεχής παράδοση μικρών κομματιών κώδικα, συνεχής ανατροφοδότηση (feedback) από τον πελάτη και άμεση προσαρμογή στις αλλαγές.

- **Βασικό πλεονέκτημα:** Τα λάθη βρίσκονται και διορθώνονται στην πορεία ανάπτυξης

Scrum – πλαίσιο για εφαρμογή της Agile

- Το Scrum είναι το **πιο δημοφιλές πρακτικό πλαίσιο (framework) για την εφαρμογή του Agile**. Καθορίζει συγκεκριμένους ρόλους, συναντήσεις και διαδικασίες.
- **Sprints:** Ο χρόνος χωρίζεται σε σταθερούς κύκλους (συνήθως 2-4 εβδομάδες) που ονομάζονται Sprints. Σε κάθε Sprint, η ομάδα δεσμεύεται να ολοκληρώσει συγκεκριμένες λειτουργίες.
- **Ρόλοι:** Product Owner: Αντιπροσωπεύει τον πελάτη και καθορίζει τι πρέπει να φτιαχτεί. Scrum Master: Βοηθά την ομάδα να λειτουργεί σωστά και λύνει προβλήματα. Developers/Team: Οι άνθρωποι που υλοποιούν το έργο.
- **Συναντήσεις:** Καθημερινά 15λεπτα meetings (Daily Scrum) για το τι έγινε χθες, τι θα γίνει σήμερα και τι εμπόδια υπάρχουν.

DevOps - Η Κουλτούρα και η Γέφυρα

- Στο ξεκίνημα της ανάπτυξης λογισμικού και για πολλά χρόνια μετά, οι Developers (Development - Dev) έγραφαν τον κώδικα και οι άνθρωποι των υποδομών/servers (Operations - Ops) αναλάμβαναν να τον τρέξουν. Αυτές οι δύο ομάδες συχνά δεν επικοινωνούσαν καλά.
- Το DevOps είναι μια **κουλτούρα που ενώνει το Dev και το Ops σε μία ενιαία ομάδα**.
- **Κεντρική ιδέα:** Αυτοματοποίηση των διαδικασιών και κοινή ευθύνη για τη σταθερότητα και την ταχύτητα του λογισμικού.
- **Στόχος:** Να μπορεί ο κώδικας που γράφει ένας προγραμματιστής να βγαίνει στην παραγωγή με ασφάλεια μέσα σε λίγα λεπτά ή ώρες, αντί για εβδομάδες.

CI/CD - Τα Εργαλεία και η Αυτοματοποίηση

- Το CI/CD είναι η **τεχνική καρδιά του DevOps**. Πρόκειται για μια αυτοματοποιημένη "γραμμή παραγωγής" (pipeline) για τον κώδικα.
- **Continuous Integration (CI - Συνεχής Ενσωμάτωση):** Κάθε φορά που ένας προγραμματιστής αλλάζει τον κώδικα, αυτός "ανεβαίνει" σε μια κεντρική βάση. Εκεί, αυτόματα συστήματα ελέγχουν αν ο νέος κώδικας δημιουργεί προβλήματα στον παλιό, τρέχοντας αυτοματοποιημένα τεστ.
- **Continuous Delivery / Deployment (CD - Συνεχής Παράδοση / Διάθεση):** Μόλις ο κώδικας περάσει τα τεστ, το σύστημα τον προετοιμάζει και τον στέλνει αυτόματα στους servers, ώστε οι χρήστες να δουν άμεσα τις νέες δυνατότητες ή τις διορθώσεις.

Έννοια	Τι αντιπροσωπεύει;	Που εστιάζει;
Agile	Φιλοσοφία & Νοοτροπία	Ευελιξία και ικανοποίηση του πελάτη.
Scrum	Διαδικασία & Ρόλοι	Πώς θα οργανωθεί η ομάδα στην καθημερινότητα.
DevOps	Κουλτούρα & Συνεργασία	Γέφυρα μεταξύ προγραμματιστών και διαχειριστών συστημάτων.
CI/CD	Τεχνική Πρακτική	Αυτοματοποίηση ελέγχων και μεταφοράς του κώδικα live.

Τι Αλλάζει στον Κύκλο Ζωής Λογισμικού (SDLC)

- **Από τη «Σύνταξη Κώδικα» στη «Σύνταξη Προδιαγραφών»**
  - Ο προγραμματιστής μετατρέπεται από συγγραφέα εντολών σε αρχιτέκτονα προθέσεων, δηλ. στο τι θέλει να πετύχει και γιατί (σχεδιασμός, επιχειρηματική λογική, αρχιτεκτονική)
- **Από την Κατασκευή Συστημάτων στη Δημιουργία/Παραγωγή τους (Generating Systems):**
  - Το boilerplate και ο επαναλαμβανόμενος κώδικας αυτοματοποιούνται σε ποσοστό άνω του 70%–95% σε ορισμένα workflows.
- **Η Νέα Προτεραιότητα:**
  - Μεγαλύτερο βάρος στην *επαλήθευση (verification)* και στον *έλεγχο ποιότητας* παρά στη συγγραφή κώδικα.

Από τα CASE Tools στους AI Agents

- **Δεκαετία '80 - '90 (Παραδοσιακά CASE):** Σχεδιάζεις ένα visual διάγραμμα (π.χ. UML) και το εργαλείο παράγει έναν «τυφλό», στατικό σκελετό κώδικα (Boilerplate).
- **Δεκαετία 2010 (DevOps & Σύγχρονα IDEs):** Τα CASE εργαλεία ενσωματώνονται σε πλατφόρμες όπως το GitHub, το Jira και το VS Code, αυτοματοποιώντας το testing και το deployment (CI/CD).
- **Σήμερα (AI Agents & Assistants):** Τα εργαλεία CASE δίνουν τη θέση τους σε εργαλεία όπως το Claude Code ή το Cursor.
  - Αντί για χειροκίνητη σχεδίαση visual διαγραμμάτων για να παραχθεί κώδικας, ο χρήστης περιγράφει αυτό που θέλει σε φυσική γλώσσα.
  - Το AI αναλαμβάνει τον ρόλο του Upper CASE (σχεδιασμός/αρχιτεκτονική) και του Lower CASE (συγγραφή κώδικα/testing/commit) ταυτόχρονα.

Πως αλλάζει στο AI το Forward & Reverse Engineering των CASE Tools

- Σήμερα, με εργαλεία όπως το Claude Code ή το Cursor έχουν εξελίξει το Forward και Reverse Engineering με χρήση φυσικής γλώσσας:
- **AI Reverse Engineering:** Ο χρήστης λέει *"Διάβασε αυτό το legacy repository και εξήγησέ μου την αρχιτεκτονική του"*. Το AI κάνει reverse engineering, κατανοεί το context και δίνει την απάντηση.
- **AI Forward Engineering:** Ο χρήστης λέει *"Με βάση την αρχιτεκτονική που μόλις διάβασες, φτιάξε ένα νέο microservice που να συνδέεται με την ίδια βάση"*. Το AI κάνει forward engineering και παράγει τον κώδικα.

Άλλα AI Εργαλεία

- **Αυτόνομοι Πράκτορες & Πράκτορες που τρέχουν στο τερματικό (Agent CLIs*):** προγράμματα AI που γράφουν κώδικα, το τεστάρουν, το διορθώνουν, το ξανατρέχουν κλπ, χωρίς ανθρώπινη παρέμβαση, π.χ. Aider, Mentat, Claude code
- **Βοηθοί Προγραμματισμού AI (AI Coding Assistants):** Εργαλεία λογισμικού που ενσωματώνονται στο περιβάλλον εργασίας ενός προγραμματιστή (IDE) για να τον **βοηθήσουν να γράψει, να διορθώσει, να τεκμηριώσει και να δοκιμάσει κώδικα πιο γρήγορα**, π.χ. GitHub Copilot, Cursor, Windsurf
  - Αντίθετα με τους *Autonomous Agents* (που προσπαθούν να λύσουν ένα πρόβλημα εντελώς μόνοι τους στο παρασκήνιο), οι AI Coding Assistants λειτουργούν ως **συν-οδηγοί (copilots)** σε πραγματικό χρόνο. Δουλεύουν δίπλα-δίπλα με τον άνθρωπο-προγραμματιστή, προσφέροντας άμεση υποστήριξη την ώρα που εκείνος πληκτρολογεί [Aider]
  - GitHub Copilot, Cursor, Windsurf

***CLI** = Command Line Interface*

Επαγγέλματα και Εξειδικεύσεις ΤΛ - 1

- **Αναλυτής Συστημάτων (Business Analyst)** – μεταφράζει τις ανάγκες του πελάτη σε απαιτήσεις λογισμικού.
- **Υπεύθυνος Προϊόντος (Product Manager)** – καθορίζει τι πρέπει να αναπτυχθεί και ποια χαρακτηριστικά έχουν προτεραιότητα.
- **Διευθυντής έργου ανάπτυξης λογισμικού (Project Manager)** – διαχειρίζεται ομάδες ανάπτυξης και εφαρμόζει μεθοδολογίες όπως Agile, Scrum, Kanban.
- **Μηχανικός Λογισμικού (Software Engineer)** – σχεδιάζει και υλοποιεί συστήματα λογισμικού.

Επαγγέλματα και Εξειδικεύσεις ΤΛ - 2

- **Backend / Frontend Developer** – υλοποιεί λειτουργίες είτε στο "παρασκήνιο" (server, βάσεις δεδομένων) είτε στο "προσκήνιο" (διεπαφή χρήστη).
- **Full-stack Developer** – καλύπτει και το backend και το frontend.
- **Mobile App Developer** – ανάπτυξη εφαρμογών για Android/iOS.
- **Embedded Systems Developer** – ανάπτυξη λογισμικού για "έξυπνες" συσκευές (IoT, αυτοκίνητα, ρομποτική).

Επαγγέλματα και Εξειδικεύσεις ΤΛ - 3

- **Μηχανικός Εξασφάλισης Ποιότητας (Quality Assurance Engineer)** – σχεδιάζει και εκτελεί δοκιμές για να διασφαλίσει ότι το λογισμικό λειτουργεί σωστά.
- **Μηχανικός Αυτοματοποιημένων Ελέγχων (Test Automation Engineer)** – δημιουργεί αυτοματοποιημένα tests
- **Μηχανικός Ασφάλειας (Security Engineer)** – εξειδικεύεται σε ασφαλή ανάπτυξη και προστασία από κυβερνοεπιθέσεις.
- **Μηχανικός Μηχανικής Μάθησης / Τεχνητής Νοημοσύνης** – αναπτύσσει συστήματα τεχνητής νοημοσύνης.

Επαγγέλματα και Εξειδικεύσεις ΤΛ - 4

- **Μηχανικός Υποδομών Τεχνητής Νοημοσύνης (AI Infrastructure Engineer) –** υπεύθυνος για hardware, servers, δίκτυα. Σχεδιάζει, χτίζει και συντηρεί τα πανίσχυρα υπολογιστικά συστήματα που απαιτούνται για να εκπαιδευτούν και να τρέξουν τα μοντέλα AI.
- **Μηχανικός Πλατφόρμας Μηχανικής Μάθησης (ML Platform Engineer)**: δημιουργεί και συντηρεί τα εσωτερικά εργαλεία και τις πλατφόρμες για να διευκολύνει τη δουλειά των Data Scientists/ML Engineerts.
  - Παίρνει την υποδομή που αναπτύσσει ο Μηχανικός Υποδομών AI και τη μετατρέπει σε μια εύχρηστη πλατφόρμα λογισμικού που δεν έχει πολυπλοκότητα, ώστε ένας Data Scientist να μπορεί να εκπαιδεύσει και να δημοσιεύσει ένα μοντέλο γρήγορα και απλά, χωρίς να χρειάζεται να ξέρει πώς να ρυθμίζει servers ή Kubernetes.

Επαγγέλματα και Εξειδικεύσεις ΤΛ - 5

- **Ειδικός Ασφάλειας Τεχνητής Νοημοσύνης (AI Security Specialist)**
  - Φροντίζει για την προστασία των συστημάτων AI από κυβερνοεπιθέσεις, τη διασφάλιση των ευαίσθητων δεδομένων και την αποτροπή κακόβουλης χρήσης των μοντέλων.
  - Ο Μηχανικός υποδομών AI και πλατφόρμας Μηχανικής Μάθησης χτίζει το «εργοστάσιο» και τις «γραμμές παραγωγής» του AI, ενώ ο AI Security Specialist είναι ο **υπεύθυνος ασφαλείας, ο επιθεωρητής και ο "ethical hacker"** του συστήματος.
  - Η δουλειά του είναι κρίσιμη επειδή τα μοντέλα AI (ιδίως τα LLMs) εισάγουν νέες ευπάθειες που δεν υπήρχαν στο παραδοσιακό λογισμικό.

ΠΑΝΤΑ
χρειάζεται
ΕΛΕΓΧΟΣ ΤΟΥ ΑΠΟΤΕΛΕΣΜΑΤΟΣ
από τον άνθρωπο

Ειδικά όταν πρόκειται για κρίσιμες εφαρμογές
και κρίσιμες αποφάσεις

Ευχαριστώ για την προσοχή σας!