Περιεχόμενα

Κεφάλαιο 1

Low-Code

Τα πετρογραφικά και οι ζωγραφιές στους τοίχους παλαιών σπηλαίων είναι ένα δείγμα από το πόσο ουσιώδης είναι η οπτική επικοινωνία για τον άνθρωπο. [5]

1.1 Ορισμός

Μια **Πλατφόρμα Ανάπτυξης Εφαρμογών σε Low-Code** (LCAP) είναι μια πλατφόρμα ανάπτυξης λογισμικού που υποστηρίζει την ταχεία ανάπτυξη και διαχείριση εφαρμογών. Συνήθως είναι Platform-as-a-service (PaaS) cloud μοντέλα, και χρησιμοποιείται ελάχιστος ή και μηδενικός δομημένος προγραμματισμός (structured programming).

Για τον προγραμματισμό παρέχεται γραφικό περιβάλλον με οπτικές αφαιρέσεις (visual abstractions), μάλιστα επιτρέποντας χρήστες χωρίς προγραμματιστική εμπειρία να συνεισφέρουν στην ανάπτυξη του λογισμικού χωρίς είναι αναγκαία η βοήθεια των προγραμματιστών. Έτσι οι προγραμματιστές εστιάζουν παραπάνω στη *σχεδίαση* της εφαρμογής, χωρίς να ξοδεύουν χρόνο άσκοπα σε λεπτομέρειες.

Με λίγα λόγια, σκοπός είναι η παραγωγική ανάπτυξη λογισμικού με τη λιγότερη δυνατή προσπάθεια και με χαμηλότερο κόστος, και η εύκολη προσαρμογή του λογισμικού στις ταχέως μεταβαλλόμενες συνθήκες των σημερινών λειτουργικών συστημάτων. Η χρήση των LCAP έχει τύχει θετικής αποδοχής από τη βιομηχανία και η υιοθέτησή τους αυξάνεται συνεχώς. [1, 2, 9]

> "When you can visually create new business applications with minimal hand-coding –when your developers can do more of greater value, faster– that's low-code." [10]

1.2 Πριν το Low-Code

Φυσικά η μηχανική λογισμικού[1] έχει περάσει πολλά στάδια στην ιστορία της μέχρι να αρχίσουμε να αναφερόμαστε και σε χρήση αφαιρέσεων. Μέχρι τη δεκαετία του 1970, η ανάπτυξη πληροφοριακών συστημάτων έμοιαζε παραπάνω ως μια τέχνη παρά ως επιστήμη, καθώς δεν ακολουθούσε κάποια δόμηση. Αντιθέτως, ο προγραμματιστής λάμβανε μια σειρά από απαιτήσεις και ανάγκες από την πλευρά του χρήστη, και μετά από ένα διάστημα παρέδιδε ένα σύστημα που συνήθως δεν κάλυπτε εξ' ολοκλήρου όλες τις απαιτήσεις του χρήστη αλλά ήταν σίγουρα καλύτερο από το τίποτα. Η συγκεκριμένη μέθοδος, αποκαλούμενη ως **κλασική μέθοδος**, χαρακτηρίζεται από ανεπίσημες οδηγίες, έλλειψη τυποποίησης και αναφορών (documentation).

Η ανάγκη για αύξηση της παραγωγικότητας στον κύκλο ζωής έκδοσης λογισμικού (software release life cycle)[2] οδήγησε στη δημιουργία **επίσημων μεθόδων** στα τέλη της δεκαετίας του 1970. Σκοπός τους ήταν η τυποποίηση του σχεδιασμού με στόχο τη βελτίωση της ποιότητάς του. Παράδειγμα αυτών των τεχνικών είναι η χρήση δομημένης ανάλυσης (structured analysis). Έτσι, οι μηχανικοί μπορούσαν να φτιάξουν διαγράμματα ροών δεδομένων (data flow diagrams)[3], μοντέλα οντοτήτων-συσχετίσεων (ER – entity-relationship models)[4], δημιουργώντας μια συστημική περιγραφή του λογικού και φυσικού μέρους του πληροφοριακού συστήματος που ανέπτυσσαν.

Με την περαιτέρω ανάπτυξη των προσωπικών υπολογιστών στα μέσα της δεκαετίας του 1980, η χρήση επίσημων μεθόδων ήταν μονόδρομος, και μάλιστα δημιούργησε μια νέα τάση, την ανάγκη για αυτοματοποίηση της ανάπτυξης κώδικα και την ελαχιστοποίηση της χειροκίνητης γραφής του.

Απόρροια αυτής της τάσης ήταν α) τη δεκαετία του 1980 οι **γλώσσες προγραμματισμού τέταρτης γενιάς** (4GLs)[5], τα **Computer-Aided Software Engineering** (CASE) περιβάλλοντα, β) η **Ταχεία Ανάπτυξη Εφαρμογών** (Rapid Application Development – RAD)[6] τη δεκαετία του 1990, γ) ο **Προγραμματισμός τελικού χρήστη** (End-User

[1] Θα ορίζαμε τη μηχανική λογισμικού ως μια πειθαρχημένη και αυστηρή εφαρμογή μεθόδων, διαδικασιών και εργαλείων στη διαχείριση και ανάπτυξη υπολογιστικών συστημάτων. Πρόκειται για ένα εννοιλογικό πλαίσιο που περιγράφει τη διαχείριση συστημάτων.

[2] Πρόκειται μια έννοια που αναφέρεται στις φάσεις ανάπτυξης και ύπαρξης ενός λογισμικού. Ξεκινάει από τη σύλληψη της ιδέας, τη μελέτη για τις απαιτήσεις του, την υλοποίηση του, τη διάθεση του προϊόντος στο πελάτη, τη υποστήριξή του με ενημερώσεις και τέλος την απόσυρσή του.

[3] Είναι μια οπτική αναπαράσταση της ροής των δεδομένων σε ένα σύστημα. Αναγράφονται οι διεργασίες (κύκλοι), οι είσοδοι και έξοδοι (τετράγωνα) και η αποθήκευση των δεδομένων (παράλληλες γραμμές).

[4] Περιγράφει ένα σύνολο αντικειμένων (οντότητες) και τις *σχέσεις* μεταξύ αυτών των αντικειμένων.

[5] Σε αντίθεση με τις γλώσσες προγραμματισμού τρίτης γενιάς (C, Java κτλ), οι γλώσσες προγραμματισμού τέταρτης γενιάς σχεδιάστηκαν με γνώμονα την απλοποίηση του προγραμματισμού ειδικά για χρήστες χωρίς προγραμματιστική εμπειρία. Χαρακτηρίζονται από υψηλού επιπέδου αφαιρέσεις, πλησιάζοντας στην ανθρώπινη γλώσσα, κάνοντάς τες πιο εύκολα κατανοητές.

[6] Πρόκειται για μια μεθοδολογία που δίνει βάση στη δημιουργία πρωτοτύπων. Έτσι οι προγραμματιστές δε χρειάζεται να ξεκινάνε από το μηδέν την ανάπτυξη κάθε νέου λογισμικού, ούτε να ξοδεύουν πολύτιμο χρόνο για να περιγράψουν λεπτομερώς όλες τις προδιαγραφές του. Χρησιμοποιώντας έτοι-

Development – EUD)[7] τη δεκαετία του 2000, και οι **αρχιτεκτονικές που βασίζονται σε μοντέλα** (model-driven architecture – MDA) τις τελευταίες δύο δεκαετίες. [3, 4, 8]

Στις επόμενες υποενότητες θα αναφερθούμε πιο εκτεταμένα στα CASE και MDA περιβάλλοντα, τα οποία υπήρξαν και τα κύρια πρωταίτια των Low-Code περιβαλλόντων.

1.2.1 Computer-Aided Software Enginnering (CASE)

Τα **Computer-Aided Software Engineering** (CASE – Μηχανική Λογισμικού Υποβοηθούμενη από Υπολογιστή) περιβάλλοντα επέτρεπαν τους μηχανικούς να καταγράφουν και να μοντελοποιούν με συστηματικό τρόπο ένα πληροφοριακό σύστημα από τις αρχικές περιγραφές του χρήστη ως τη σχεδίαση και την υλοποίηση και να εκτελούν δοκιμές για τη συνέπειά του.

Μέχρι πρότινος, τα εργαλεία που αφορούν την ανάπτυξη λογισμικού εστιάζουν κυρίως στην επεξεργασία πηγαίου κώδικα και την αποσφαλμάτωση του. Σε αντίθεση λοιπόν με τα υπάρχοντα εργαλεία, τα CASE περιβάλλοντα βοηθούν στη *μεθοδολογία* της ανάπτυξης λογισμικού: στην ανάλυση απαιτήσεων, στο λογικό σχεδιασμό, στον έλεγχο εγκυρότητας, την επαναχρησιμοποίηση και εξάλειψη πλεονασμών.

Είναι το δεξί χέρι ενός μηχανικού λογισμικού, βοηθώντας σε πολλές εργασίες που τον δυσκολεύουν, αυξάνοντας την παραγωγικότητα και την ποιότητα της δουλειάς του. Τα εργαλεία που περιλαμβάνουν ποικίλλουν: κάποια επιτρέπουν τη δημιουργία διαγραμμάτων ενώ άλλα μπορούν να αυτοματοποιούν όλα τα στάδια του κύκλου ζωής έκδοσης λογισμικού. Ένα ολοκληρωμένο CASE περιβάλλον περιλαμβάνει:

- ένα διαδραστικό, φιλικό για το χρήστη, οπτικό περιβάλλον διαχείρισης
- ένα σύνολο από εργαλεία ανάπτυξης (επεξεργαστές κειμένου, λεξικά, αναλυτές σχεδιασμού κ.α.)
- ένα σύνολο από εργαλεία για τον έλεγχο της διαδικασίας (για τον χρονοπρογραμματισμό, τη διασφάλιση ποιότητας κ.α.)
- ένα περιβάλλον βοήθειας με το documentation των εργαλείων
- ένα σύστημα διαχείρισης βάσεων δεδομένων

Τα συστήματα που δημιουργούνται από το CASE είναι εφαρμογές της πειθαρχημένης εφαρμογής μεθόδων της μηχανικής λογισμικού. Τα CASE περιβάλλοντα έθεσαν τα θεμέλια για τη δημιουργία νέων προτύπων, όπως ο οπτικός προγραμματισμός (visual programming) και ο προγραμματισμός που βασίζεται σε μοντέλα. [4, 3, 5, 7]

μα (μάλιστα και modular) πρωτότυπα, ο χρόνος ανάπτυξης μειώνεται. Παραδείγματα RAD εργαλείων είναι οι GUI builders· πρόκειται για WYSIWYG (What-You-See-Is-What-You-Get) επεξεργαστές για τη γρήγορη ανάπτυξη λογισμικών με διεπαφή χρήστη (user interface).

[7]Περιγράφει εργαλεία που επιτρέπουν τον προγραμματισμό από τους απλούς τελικούς χρήστες. Παραδείγματα είναι λογιστικά φύλλα όπως το Microsoft Excel ή εκπαιδευτικά εργαλεία όπως το Scratch.

1.2.2 Model-driven Architecture (MDA)

Οι **αρχιτεκτονικές που βασίζονται σε μοντέλα** (model-driven architecture – MDA) διαχωρίζουν τη λειτουργικότητα μιας εφαρμογής από την υλοποίησή της σε μια συγκεκριμένη πλατφόρμα, προσφέροντας ένα υψηλότερο επίπεδο αφαίρεσης. Στόχος είναι οι προγραμματιστές να εστιάζουν περισσότερο στον σχεδιασμό και λιγότερο στο να λύνουν θέματα που αφορούν την πλατφόρμα υλοποίησης.

Μέχρι πρότινος η ανάπτυξη λογισμικού αφορούσε καθαρά δομικό κώδικα. Η MDA άλλαξε τον τρόπο σκέψης των προγραμματιστών, καθώς πλέον επικεντρώνονταν παραπάνω στον σωστό διαχωρισμό των χαρακτηριστικών, στην αφαιρετικότητα και στην αυτοματοποίηση. [2, 6, 8]

Βιβλιογραφία

[1] Alexander C. Bock και Ulrich Frank. «Low-Code Platform». Στο: *Business and Information Systems Engineering* 63 (6 Δεκ. 2021), σσ. 733–740. ISSN: 18670202. DOI: 10.1007/s12599-021-00726-8.

[2] Alessio Bucaioni, Antonio Cicchetti και Federico Ciccozzi. «Modelling in low-code development: a multi-vocal systematic review». Στο: *Software and Systems Modeling* 21 (5 Οκτ. 2022), σσ. 1959–1981. ISSN: 16191374. DOI: 10.1007/s10270-021-00964-0.

[3] Albert E Case. *Computer-aided software engineering (case): technology for improving software development productivity*. 1985.

[4] E. J. Chikofsky. *Software Development — Computer-Aided Software Engineering (CASE)*.

[5] D. L. Kuhn. *Selecting and Effectively Using a Computer-Aided Software Engineering Tool*. 1989.

[6] *MDA FAQ | Object Management Group — omg.org*. https://www.omg.org/mda/faq_mda.htm. [Accessed 11-10-2024].

[7] G. Premkumar και Michael Potter. *Adoption of Computer Aided Software Engineering (CASE) Technology: An Innovation Adoption Perspective*.

[8] Davide Di Ruscio κ.ά. «Low-code development and model-driven engineering: Two sides of the same coin?» Στο: *Software and Systems Modeling* 21 (2 Απρ. 2022), σσ. 437–446. ISSN: 16191374. DOI: 10.1007/s10270-021-00970-2.

[9] Apurvanand Sahay κ.ά. «Supporting the understanding and comparison of low-code development platforms». Στο: *Proceedings - 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020*. Institute of Electrical και Electronics Engineers Inc., Αύγ. 2020, σσ. 171–178. ISBN: 9781728195322. DOI: 10.1109/SEAA51224.2020.00036.

[10] *What Is Low-Code? | IBM — ibm.com*. https://www.ibm.com/topics/low-code. [Accessed 11-10-2024].