Περιεχόμενα

Κεφάλαιο 1

Low-Code

"*The future of coding is no coding at all.*"
—Chris Wanstrath, Co-Founder, Github

Στο κεφάλαιο αυτό, περιγράφεται η έννοια του low-code (χαμηλός κώδικας). Αναφερόμαστε στον ορισμό του, στην ιστορική εξέλιξη της μηχανικής λογισμικού που επέτρεψε την ύπαρξη πλατφορμών ανάπτυξης λογισμικού σε low-code...

1.1 Τι είναι ο χαμηλός κώδικας

"*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.*" [14]

Ένας τρόπος για να αντιληφθούμε καλύτερα την έννοια του χαμηλού κώδικα είναι να κοιτάξουμε την εξέλιξη των γλωσσών προγραμματισμού. Για παράδειγμα, η Python, μια γλώσσα υψηλού επιπέδου, θα μπορούσε να χαρακτηριστεί ως χαμηλός κώδικας σε σύγκριση με τη C++. Αντίστοιχα η C θα μπορούσε να χαρακτηριστεί και αυτή χαμηλός κώδικας συγκριτικά με την Assembly, όπως και η Assembly είναι χαμηλός κώδικας αν τη συγκρίνουμε με το να αλλάζουμε χειροκίνητα μηδενικά και άσσους στους καταχωρητές. Πρακτικά, η εξέλιξη του προγραμματισμού, είναι και η εξέλιξη του χαμηλού κώδικα.

Επομένως, ο χαμηλός κώδικας, αν και ονομάστηκε πρόσφατα ως όρος, η *έννοιά* του δεν είναι καθόλου καινούρια, και είναι η άμεση εξέλιξη του υψηλού επιπέδου γλωσσών προγραμματισμού (4GLs) των τελευταίων δεκαετιών. Το υψηλότερο προγραμματιστικό επίπεδο με τις αφαιρέσεις που διαθέτει, επιτρέπει ευκολότερη και γρηγορότερη ανάπτυξη λογισμικού. Πέρα όμως από τους χρήστες που είναι ήδη προγραμματιστές, προσφέρει επιπλέον τη δυνατότητα σε χρήστες με λίγη ή και καθόλου προγραμματιστική εμπειρία[1] να τροποποιήσουν εφαρμογές ή και να φτιάξουν εξ' ο-

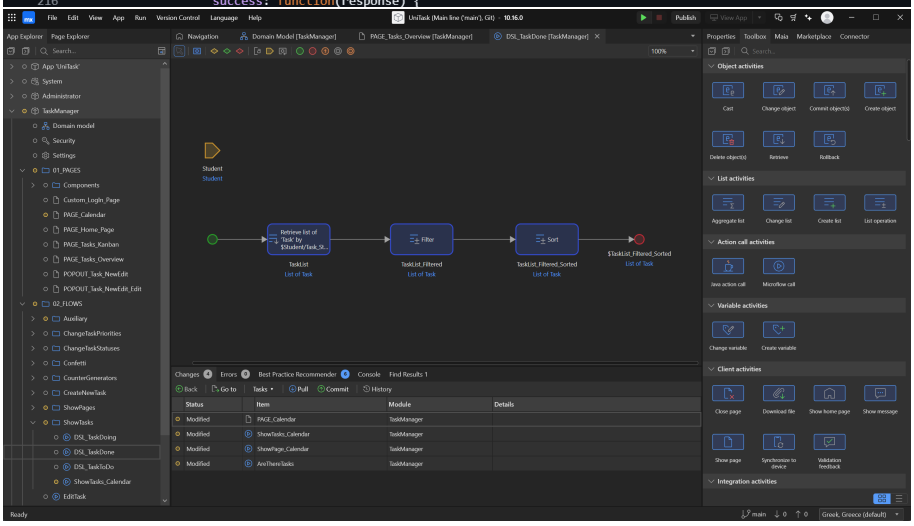
[1] Συχνά αποκαλούμενοι ως **citizen developers** (προγραμματιστές πολίτες), πρόκειται για χρήστες με λίγη ή μηδενική προγραμματιστική εμπειρία που χρησιμοποιούν προγραμματιστικά εργαλεία χαμηλού κώδικα για τον σχεδιασμό εφαρμογών.

λοκλήρου τις δικές τους, με την ίδια λογική όπως η Python επέτρεψε σε περισσότερο κόσμο να προγραμματίσει σε σχέση με την Assembly. Ο προγραμματισμός σε χαμηλό κώδικα πραγματοποιείται σε πλατφόρμες ονομάζονται **Πλατφόρμες Ανάπτυξης Λογισμικού σε Low-Code** (Low-Code Development Platforms – LCDPs), οι οποίες θα αναλυθούν εκτενέστερα στα επόμενα κεφάλαια. Οι πλατφόρμες περιλαμβάνουν γραφικό περιβάλλον με drag-and-drop και WYSIWYG (What-You-See-Is-What-You-Get) editors, επιτρέποντας την πιο γρήγορη και ενστικτώδη κατασκευή εφαρμογών. Αυτός ο οπτικός προγραμματισμός (visual programming) είναι σημαντικός παράγοντας στην προσβασιμότητα που προσφέρει ο χαμηλός κώδικας. [2] [12] [13]

1.1.1 Οπτικός προγραμματισμός (visual programming)

Παρακάτω παρατίθενται δύο παραδείγματα από την παραδοσιακή ανάπτυξη εφαρμογών και την ανάπτυξη εφαρμογών σε low-code στην πλατφόρμα Mendix.

```
192  document.getElementById('createRescuerForm').addEventListener('submit', function(e) {
193      e.preventDefault();
194
195      // Function to create rescuer account
196      function createRescuerAccount() {
197          return new Promise((resolve, reject) => {
198              // Get form data
199              var name = document.getElementById('rescuerName').value;
200              var username = document.getElementById('rescuerUsername').value;
201              var email = document.getElementById('rescuerEmail').value;
202              var password = document.getElementById('rescuerPassword').value;
203              var phone = document.getElementById('rescuerPhone').value;
204
205              // Make an AJAX POST request to create the rescuer account
206              $.ajax({
207                  url: './php/create_rescuer.php',
208                  type: 'POST',
209                  data: {
210                      'name': name,
211                      'username': username,
212                      'email': email,
213                      'password': password,
214                      'phone': phone
215                  },
216                  success: function(response) {
```



Σχήμα 1.1: Παραδοσιακός κώδικας και το γραφικό περιβάλλον του Mendix.

Στο γραφικό περιβάλλον ο προγραμματισμός γίνεται σε ένα διάγραμμα ροής με drag-and-drop επαναχρησιμοποιούμενα στοιχεία (εν προκειμένω στο σχήμα 1.1, ένα παράδειγμα επαναχρησιμοποιούμενων στοιχείων είναι τα μπλε τετράγωνα στο διάγραμμα ροής και δεξιά της οθόνης), σε αντίθεση με τον παραδοσιακό κώδικα όπου γράφουμε γραμμή-γραμμή, κάτι που καθιστά την ανάπτυξη εφαρμογών εξαιρετικά γρήγορη και προσβάσιμη από περισσοτέρους.

Η οπτικοποίηση του προγραμματισμού πρόκειται για μια εκ θεμελίων επαναστατική αλλαγή. Εξάλλου, τα πετρογραφικά και οι ζωγραφιές στους τοίχους παλαιών σπηλαίων είναι ένα δείγμα από το πόσο ουσιώδης ήταν πάντα η οπτική επικοινωνία για τον άνθρωπο. [7] Ο οπτικός προγραμματισμός τους επιτρέπει να προσθέτουν λειτουργικότητες χειριζόμενοι γραφικά στοιχεία αντί να τα προσδιορίζουν μέσω κειμένου. Πέρα από αυτό, οι απλοί χρήστες συνήθως θεωρούν το ποντίκι πολύ πιο προσβάσιμο από το πληκτρολόγιο.

Θα επεκταθούμε στα επόμενα κεφάλαια στα οφέλη του low-code και στα χαρακτηριστικά των πλατφορμών ανάπτυξης λογισμικού σε low-code. Παρόλα αυτά, πρωτύτερα αξίζει να κάνουμε μια αναφορά στις τεχνολογικές εξελίξεις που μας οδήγησαν σήμερα σε αυτές τις πλατφόρμες.

1.1.2 Καθόλου κώδικας (no-code)

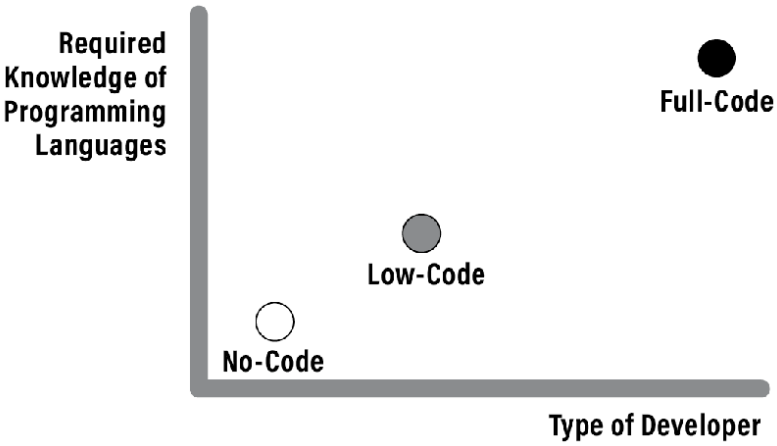
Ένας εύληπτος τρόπος για να καταλάβουμε τη διαφορά ανάμεσα στον χαμηλό κώδικα με τον καθόλου κώδικα είναι το γεγονός πως στα no-code εργαλεία οι χρήστες δε χρησιμοποιούν καθόλου το πληκτρολόγιό τους. Όλη η αλλεπιδράση με το λογισμικό πραγματοποιείται με το ποντίκι μέσω του γραφικού περιβάλλοντος. Ο περιοριστικός χαρακτήρας αυτών των εργαλείων έχει ως πλεονέκτημα το ότι οι χρήστες είναι δύσκολο να δημιουργήσουν σφάλματα, αλλά το μειονέκτημα είναι το ότι δεν υπάρχει η δυνατότητα εξατομικευμένης παραμετροποίησης από τον χρήστη.

Να σημειωθεί πως ο χαμηλός κώδικας περιλαμβάνει τα οφέλη και του κλασικού προγραμματισμού και του καθόλου κώδικα, αφού ο χρήστης διαθέτει την επιλογή να χρησιμοποιήσει τα έτοιμα εργαλεία της πλατφόρμας ή και να δημιουργήσει το δικό του κώδικα και να παραμετροποιήσει ό,τι ακριβώς θέλει αυτός.

1.2 Πώς φτάσαμε στον χαμηλό κώδικα

Η μηχανική λογισμικού[2] έχει περάσει πολλά στάδια στην ιστορία της μέχρι να αρχίσουμε να αναφερόμαστε και σε χρήση χαμηλού κώδικα. 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

[2]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.



Σχήμα 1.2: Σύγκριση ανάμεσα στην προγραμματιστική εμπειρία των χρηστών και την προγραμματιστική προσέγγιση που χρησιμοποιούν [12]

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)[3] 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)[4], μοντέλα οντοτήτων-συσχετίσεων (ER – entity-relationship models)[5], 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)[6], τα **Computer-Aided Software Engineering** (CASE)

[3]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.

[4]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).

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

[6]Σε αντίθεση με τις γλώσσες προγραμματισμού τρίτης γενιάς (C, Java κτλ) που σε αντίθεση με της

περιβάλλοντα, β) η **Ταχεία Ανάπτυξη Εφαρμογών** (Rapid Application Development – RAD)[7] τη δεκαετία του 1990, γ) ο **Προγραμματισμός τελικού χρήστη** (End-User Development – EUD)[8] τη δεκαετία του 2000, και οι **αρχιτεκτονικές που βασίζονται σε μοντέλα** (model-driven architecture – MDA) τις τελευταίες δύο δεκαετίες. [4, 5, 9]

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, γραφικό 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.)

δεύτερης (Assembly) επέτρεπαν τη δημιουργία προγραμμάτων ανεξάρτητα από το μηχάνημα που θα τις έτρεχε, οι γλώσσες προγραμματισμού τέταρτης γενιάς σχεδιάστηκαν με γνώμονα την απλοποίηση του προγραμματισμού. Χαρακτηρίζονται από υψηλού επιπέδου αφαιρέσεις, πλησιάζοντας στην ανθρώπινη γλώσσα, κάνοντάς τες πιο εύκολα κατανοητές. Παραδείγματα είναι η Python, SQL, Ruby.

[7]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 έτοιμα (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).

[8]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.

- 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. [5, 4, 7, 8]

1.2.2 Model-driven Architecture (MDA)

Τα μοντέλα προσφέρουν τη δυνατότητα αφαίρεσης σε ένα σύστημα, κάτι που επιτρέπει τους μηχανικούς να επικεντρώνονται μόνο στο πρόβλημα που προσπαθούν να λύσουν, αγνοώντας τις υπόλοιπες λεπτομέρειες. Η χρήση μοντέλων είναι ζωτικής σημασίας για την κατανόηση και επεξεργασία πολύπλοκων συστημάτων. Ένα παράδειγμα μοντελοποίησης, για παράδειγμα, θα μπορούσε να είναι τα διαφορετικά επίπεδα εμφάνισης ενός συστήματος (δομικό επίπεδο, επίπεδο συμπεριφοράς κα).

Η ιδέα των μοντέλων αποτέλεσε τη βάση για μια νέα προσέγγιση ανάπτυξης λογισμικού, τη *μηχανική που βασίζονται σε μοντέλα* (model-driven engineering – MDE). Μπορούμε να περιγράψουμε με σαφήνεια και με κανόνες τα μοντέλα (για παράδειγμα το πως θα μετατραπεί ένα μοντέλο σε ένα άλλο, την παρακολούθηση μεταξύ στοιχείων ενός μοντέλου κτλ), συντελώντας πλέον στην **αρχιτεκτονική που βασίζεται σε μοντέλα** (model-driven architecture – MDA).

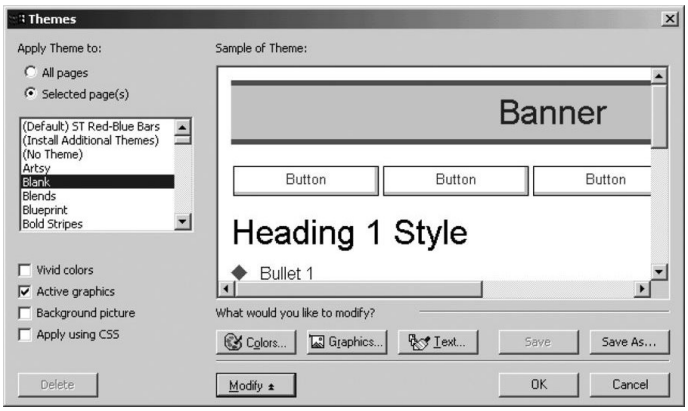
Η MDA υποστηρίζεται από την Object Management Group (OMG) [6], βασίζεται σε ένα σύνολο προτύπων για τον ορισμών μοντέλων, συμβολισμών και κανόνων μετασχηματισμού και προσφέρει μια βάση για τη λειτουργία μοντέλων όπως το UML. Τα μοντέλα χρησιμοποιούνται για τον προσδιορισμό, την προσομοίωση, την επαλήθευση, τον εκσυγχρονισμό, τη συντήρηση, την κατανόηση και τη δημιουργία κώδικα. Στόχος παραμένει η αυτοματοποίηση διαφόρων βημάτων στην ανάπτυξη λογισμικού, αυξάνοντας παράλληλα την ποιότητά του. Επιπλέον, χρησιμοποιώντας μοντέλα είναι εφικτός ο διαχωρισμός της λειτουργικότητας των εφαρμογών από την υλοποίησή της σε μια συγκεκριμένη πλατφόρμα. Ως αποτέλεσμα, οι προγραμματιστές μπορούν να εστιάζουν περισσότερο στον σχεδιασμό και λιγότερο στο να λύνουν θέματα που αφορούν την πλατφόρμα υλοποίησης.

Εν τέλει, η αρχιτεκτονική που βασίζεται σε μοντέλα άλλαξε τον τρόπο σκέψης των προγραμματιστών, καθώς πλέον επικεντρώνονταν παραπάνω στον σωστό διαχωρισμό των χαρακτηριστικών, στην αφαιρετικότητα και στην αυτοματοποίηση. [3, 9, 11]

1.2.3 Προγενέστερα λογισμικά οπτικού προγραμματισμού

Τα CASE περιβάλλοντα και η μηχανική που βασίζεται σε μοντέλα έφεραν ριζικές αλλαγές στη φιλοσοφία της μηχανικής λογισμικού. Στο κομμάτι του οπτικού προγραμματισμού όμως, δεν ήταν τα LCDP τα πρώτα που περιλάμβαναν ένα γραφικό

περιβάλλον εργασίας. Ένα από τα πρώτα λογισμικά με γραφικό περιβάλλον εργασίας ήταν το Microsoft Access. Οι χρήστες σέρνοντας και αφήνοντας στοιχεία μπορούσαν να κατασκευάζουν βάσεις δεδομένων, φόρμες κ.α. χωρίς την ανάγκη της SQL.



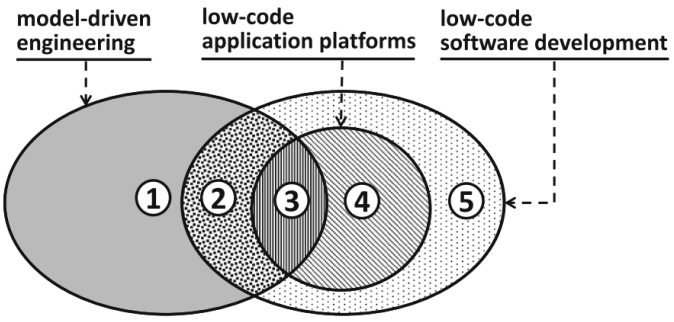
Σχήμα 1.3: Το γραφικό περιβάλλον του Microsoft FrontPage

Άλλα λογισμικά παρόμοιας λογικής ήταν το Microsoft FrontPage, ένας από τους πρώτους WYSIWYG επεξεργαστές και εργαλεία διαχείρισης ιστοσελίδων, το οποίο όμως γρήγορα έγινε παρωχημένο. Τα λογισμικά αυτά αντικαταστάθηκαν από ένα νέο τύπο λογισμικών, φιλικών προς το χρήστη, με προσβάσιμα και εύληπτα εργαλεία από όλους, τα LCDP. [12]

1.3 Πλατφόρμες Ανάπτυξης Εφαρμογών σε Low-Code (LCDP)

Όσο και αν η έννοια των μοντέλων άλλαξε τη μηχανική λογισμικού και έθεσε νέες προδιαγραφές στον σχεδιασμό του, η εξάρτησή της από δύσχρηστα πρότυπα όπως το UML περιόριζε την ευρεία υιοθέτησή της στη βιομηχανία. Τα τελευταία χρόνια έχουν εμφανιστεί πλατφόρμες χτισμένες πάνω στις αρχές της αφαίρεσης των μοντέλων, που απλοποιούν ακόμη παραπάνω τη διαδικασία της ανάπτυξης. Οι συγκεκριμένες πλατφόρμες ονομάζονται **Πλατφόρμες Ανάπτυξης Εφαρμογών σε Low-Code** (Low-Code Development Platforms – LCDPs)[9].[1]

[9]Εναλλακτικές ονομασίες είναι Low-Code Platforms (LCP), Low-Code Development Platforms (LCDP), ενώ η διαδικασία ανάπτυξης αναφέρεται ως Low-Code Software Development (LCSD). Η έννοια του Low-Code συχνά αναφέρεται και ως Χαμηλός Κώδικας.



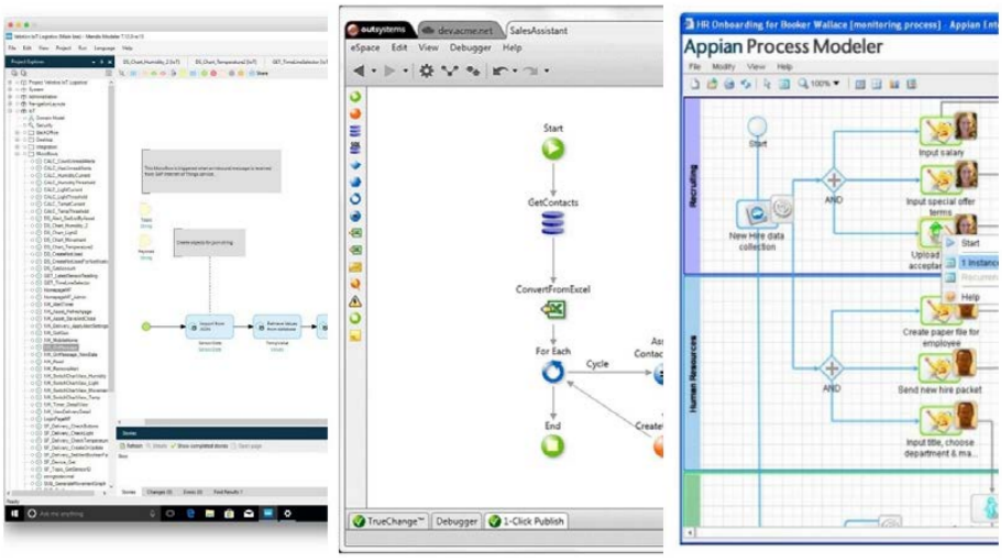
Σχήμα 1.4: Σύγκριση μεταξύ της μηχανικής που βασίζεται σε μοντέλα και των Low-Code πλατφορμών. [9]

Στην περιοχή 1 χρησιμοποιούνται μοντέλα χωρίς να γίνονται προσπάθειες για μείωση του κώδικα, σε αντίθεση με τις περιοχές 2 και 3 που στοχεύουν στην μείωση του κώδικα. Από την άλλη, οι πλατφόρμες ανάπτυξης κώδικα σε low-code δεν βασίζονται πάντα σε μοντέλα (περιοχές 4 και 5), αλλά για παράδειγμα χρησιμοποιούν δεδομένα σε σχεσιακές βάσεις ή σε XML αρχεία. Οι διαφορές μεταξύ low-code application platforms και low-code software development έγγειται στο ότι οι πλατφόρμες προσφέρουν επιπλέον την δυνατότητα διάθεσης του λογισμικού στο ευρύ κοινό, όπως επίσης και την διαχείρισή του καθ' όλο το κύκλο ζωής του.

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). Έτσι οι προγραμματιστές ή οι citizen developers 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, 3, 10]



Σχήμα 1.5: LCDPs από αριστερά προς τα δεξιά: Mendix, OutSystems, Appian [2]

1.3.1 Χαρακτηριστικά των LCDP

Παραθέτονται κάποια από τα χαρακτηριστικά που διακρίνουν τις πλατφόρμες ανάπτυξης σε Low-Code:

- **Γραφικό περιβάλλον χρήστη**: περιέρχονται εργαλεία και γραφικά στοιχεία, drag-and-drop ευελιξίες, μηχανές αποφάσεων για τη μοντελοποίηση σύνθετης λογικής, κατασκευαστές φορμών (form builder)
- **Συνεργατική ανάπτυξη**: επιτρέπει (απομακρυσμένους) χρήστες να εργαστούν στο πρότζεκτ
- **Επαναχρησιμοποίηση** προκατασκευασμένων στοιχείων, ύπαρξη marketplace,
- **Domain Model**: κατασκευή domain model για την αναπαράσταση εννοιών και σχέσεων μεταξύ τους. [9]

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] MICAH. KENNEWEG BRYAN KASAM IMRAN MCMULLEN. *Building Low-Code Applications with Mendix enterprise web and mobile app development made… easy with mendix and the power of no-code development*. PACKT PUBLISHING LIMITED, 2021. ISBN: 9781800201422.

[3] 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.

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

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

[6] https://themeforest.net/user/dan$_{f}isher$. *MDA FAQ | Object Management Group — omg.org*. https://www.omg.org/mda/faq_mda.htm. [Accessed 08-11-2024].

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

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

[9] 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.

[10] 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.

[11] Matthias Book Sami Beydeda και Volker Gruhn. *Model-Driven Software Development*.

[12] Phil Simon. *Low-Code/No-Code: Citizen Developers and the Surprising Future of Business Applications*. 2022.

[13] O'Reilly Editorial Team. «Low-Code and the Democratization of Programming». Στο: *O'Reilly Media* (2021).

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