

# Περιεχόμενα

<b>1 Διαχείριση Εργασιών</b>	<b>1</b>
1.1 Το πρόβλημα της διαχείρισης εργασιών . . . . .	1
1.1.1 Ορισμός της εργασίας . . . . .	2
1.1.2 Ορισμός της διαχείρισης εργασιών . . . . .	2
1.1.3 Πεδίο εφαρμογής διαχείρισης εργασιών . . . . .	2
1.1.4 Διαφορά με διαχείριση έργου . . . . .	3
1.2 Ιστορική αναδρομή . . . . .	4
1.2.1 Προφορικότητα και μνημονικές συσκευές . . . . .	4
1.2.2 Πρώτα ημερολόγια . . . . .	6
1.2.3 Σύγχρονη εποχή . . . . .	7
1.3 Η συνδρομή της τεχνολογίας . . . . .	8
1.3.1 Ψηφιακά εργαλεία . . . . .	9
1.3.1.1 Todoist . . . . .	9
1.3.1.2 Notion . . . . .	9
1.3.1.3 Microsoft Project . . . . .	10
1.3.1.4 Trello . . . . .	12
1.4 Μεθοδολογίες . . . . .	13
1.4.1 Διάγραμμα Γκαντ . . . . .	13
1.4.2 Program evaluation and review technique (PERT) . . . . .	14
1.4.3 Μέθοδος κρίσιμης διαδρομής (Critical Path Method – CPM) . . . . .	15
1.4.4 Ευέλικτη (Agile) μεθοδολογία . . . . .	15
1.4.5 Kanban . . . . .	16
1.5 Διαχείριση εργασιών στο πανεπιστήμιο . . . . .	16
1.5.1 Προβλήματα διαχείρισης που αντιμετωπίζουν οι φοιτητές . . . . .	16
1.5.2 Χαρακτηριστικά που οι φοιτητές θα επιθυμούσαν σε μια εφαρμογή . . . . .	17
<b>2 Low-Code</b>	<b>19</b>
2.1 Τι είναι ο χαμηλός κώδικας . . . . .	19
2.1.1 Οπτικός προγραμματισμός (visual programming) . . . . .	20
2.1.2 Καθόλου κώδικας (no-code) . . . . .	22
2.1.3 Η έννοια του προγραμματιστή πολίτη (citizen developer) . . . . .	22
2.1.3.1 Διαφορά με τους παραδοσιακούς προγραμματιστές . . . . .	23

2.1.3.2	Χαρακτηριστικά των προγραμματιστών πολιτών . . . . .	23
2.2	Πώς φτάσαμε στον χαμηλό κώδικα . . . . .	25
2.2.1	Computer-Aided Software Engineering (CASE) . . . . .	26
2.2.2	Model-driven Architecture (MDA) . . . . .	27
2.2.3	Προγενέστερα λογισμικά οπτικού προγραμματισμού . . . . .	27
2.3	Πλατφόρμες Ανάπτυξης Εφαρμογών σε Low-Code (LCDP) . . . . .	28
2.3.1	Χαρακτηριστικά των LCDP . . . . .	30

# Κεφάλαιο 1

## Διαχείριση Εργασιών

*“Plans are nothing; planning is everything”*

—Dwight D. Eisenhower

Όπως αναφέρθηκε στην εισαγωγή, η παρούσα διπλωματική εργασία επικεντρώνεται στην ανάπτυξη μιας εφαρμογής για τον προγραμματισμό και την παρακολούθηση εργασιών. Οι διαδικασίες σχεδιασμού, υλοποίησης και παρακολούθησης αυτών των εργασιών εντάσσονται στο ευρύτερο πλαίσιο της διαχείρισης εργασιών (task management). Για τον λόγο αυτό, στο παρόν κεφάλαιο κρίνεται απαραίτητο να παρουσιαστεί μια πιο αναλυτική ερμηνεία του όρου, καθώς και οι θεμελιώδεις αρχές και μέθοδοι που σχετίζονται με τη συγκεκριμένη έννοια.

Η διαχείριση εργασιών αποτελεί ουσιώδες στοιχείο της καθημερινότητας, τόσο σε προσωπικό όσο και σε επαγγελματικό επίπεδο. Με την αυξανόμενη πολυπλοκότητα των σύγχρονων υποχρεώσεων και την ανάγκη για αποτελεσματικό συντονισμό πολλαπλών δραστηριοτήτων, αναδύονται νέες προκλήσεις που καθιστούν τη διαδικασία αυτή ολοένα και πιο απαιτητική.

Στο πλαίσιο του κεφαλαίου αυτού, θα εξεταστεί αναλυτικά η διαδικασία διαχείρισης εργασιών, η ιστορική της εξέλιξη, οι μεθοδολογίες που βελτιώνουν την αποδοτικότητά της και ο ρόλος που διαδραματίζει στην πανεπιστημιακή κοινότητα. Ο στόχος είναι να αναδειχθεί η σημασία της έννοιας αυτής για την καθημερινότητα, με ιδιαίτερη έμφαση στους φοιτητές, καθώς αποτελεί τη βάση της λογικής για την ανάπτυξη της εφαρμογής που θα παρουσιαστεί στη συνέχεια.

### 1.1 Το πρόβλημα της διαχείρισης εργασιών

Στην παρούσα ενότητα θα οριστεί η έννοια της διαχείρισης εργασιών, εστιάζοντας στις διαφορές της από τη διαχείριση έργου. Παράλληλα, θα αναλυθούν τα κύρια πεδία εφαρμογής της, προκειμένου να αποκτηθεί μια ολοκληρωμένη κατανόηση του όρου, ο οποίος αποτελεί το επίκεντρο της ανάλυσης στο παρόν κεφάλαιο.

### 1.1.1 Ορισμός της εργασίας

Εργασία (task, project) στη διαχείριση εργασιών ονομάζεται μια προσωρινή δραστηριότητα που αναλαμβάνεται για τη δημιουργία ενός μοναδικού αποτελέσματος (προϊόντος, υπηρεσίας, αναφοράς ή κάποιου άλλου παραδοτέου). Η εργασία πραγματοποιείται σε μια προκαθορισμένη χρονική περίοδο και τερματίζεται όταν έχει γίνει η επίτευξη των στόχων, όταν δεν υπάρχει πλέον ανάγκη για την επίτευξη των στόχων ή όταν υπάρχει η σαφής εκτίμηση ότι οι στόχοι δεν μπορούν να επιτευχθούν. [8]

### 1.1.2 Ορισμός της διαχείρισης εργασιών

Η διαχείριση εργασιών (task management) ορίζεται ως η διαδικασία οργάνωσης, ιεράρχησης και παρακολούθησης των επιμέρους εργασιών καθ' όλη τη διάρκειά τους, από τον αρχικό σχεδιασμό έως την ολοκλήρωσή τους, με στόχο τη διασφάλιση της αποδοτικής και αποτελεσματικής εκτέλεσής τους.

Η διαδικασία αυτή αποτελεί θεμελιώδες στοιχείο για τη βελτίωση της παραγωγικότητας, τόσο σε ατομικό όσο και σε συλλογικό επίπεδο. Αν και παραδοσιακά η διαχείριση εργασιών πραγματοποιούνται χειροκίνητα, η ραγδαία ανάπτυξη της τεχνολογίας έχει καταστήσει τα ψηφιακά εργαλεία τον κύριο τρόπο υλοποίησής της, προσφέροντας αυξημένη ευελιξία και ακρίβεια.

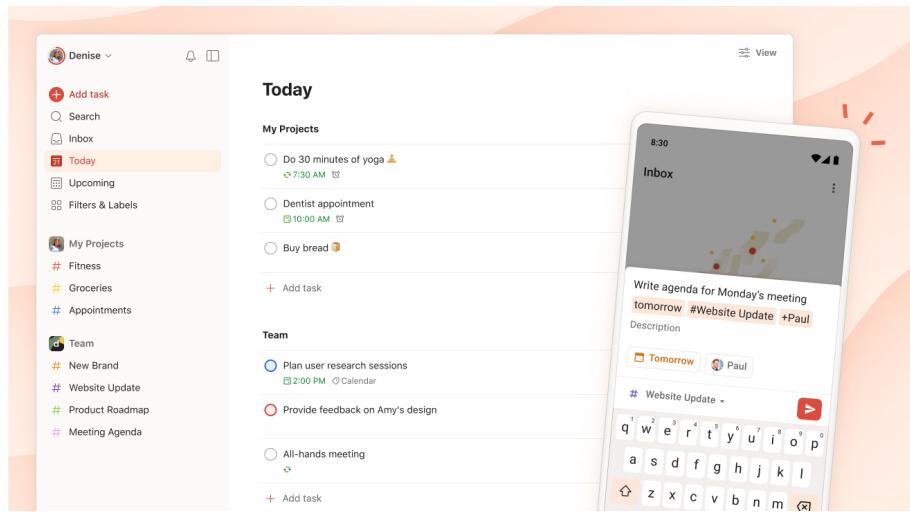
### 1.1.3 Πεδίο εφαρμογής διαχείρισης εργασιών

Η διαχείριση εργασιών διαθέτει ένα ευρύ πεδίο εφαρμογής, καλύπτοντας δραστηριότητες που κυμαίνονται από απλές καθημερινές υποχρεώσεις έως τη διαχείριση σύνθετων και απαιτητικών εργασιών.

Μια από τις πιο άμεσες και προσιτές εφαρμογές της συναντάται στον καθημερινό προσωπικό προγραμματισμό. Σε αυτό το πλαίσιο, περιλαμβάνει τη χρήση εργαλείων όπως λίστες υποχρεώσεων (to-do lists), ημερολόγια ή ψηφιακές εφαρμογές (π.χ. Todoist [28], Notion [33], Google Keep), τα οποία διευκολύνουν την οργάνωση προσωπικών υποχρεώσεων, τον προγραμματισμό ραντεβού ή δραστηριοτήτων αναψυχής, καθώς και τη διαχείριση στόχων. Μέσα από τέτοιες πλατφόρμες, δίνεται η δυνατότητα τόσο για μακροπρόθεσμο όσο και για μακροπρόθεσμο σχεδιασμό, ενώ παρέχεται και η ευκολία παρακολούθησης της προόδου.<sup>1</sup>

Στα επαγγελματικά περιβάλλοντα, η διαχείριση εργασιών διαδραματίζει καθοριστικό ρόλο, συμβάλλοντας στη βελτίωση της συνεργασίας και της συνολικής αποδοτικότητας. Κεντρικός στόχος της είναι η ορθολογική κατανομή του φόρτου εργασίας, η διασφάλιση του συντονισμού μεταξύ των μελών μιας ομάδας και η οργάνωση των καθημερινών δραστηριοτήτων. Μέσω της διαχείρισης εργασιών καθίσταται δυνατή η διευθέτηση παράλληλων εργασιών, η ιεράρχησή τους με βάση την προτεραιότητα (επείγουσες ή μη), καθώς και η δίκαιη και στοχευμένη ανάθεση καθηκόντων στους

<sup>1</sup>Θα αναφερθούμε πιο εκτεταμένα σε ψηφιακά εργαλεία στο κεφάλαιο 1.3.1



Σχήμα 1.1: Η εφαρμογή Todoist.

εργαζομένους. Ένα από τα πλέον δημοφιλή εργαλεία που χρησιμοποιούνται σε ομαδικά επαγγελματικά περιβάλλοντα για τον σκοπό αυτό είναι το Trello [14], το οποίο παρέχει ευελιξία και οπτική οργάνωση των εργασιών.

Τέλος, η ραγδαία εξέλιξη της τεχνολογίας έχει οδηγήσει στη δημιουργία προγραμμάτων πλατφορμών που αξιοποιούν σύγχρονες τεχνολογίες, όπως η τεχνητή νοημοσύνη και η ανάλυση δεδομένων, για τη βελτιστοποίηση της διαχείρισης εργασιών. Αυτές οι πλατφόρμες υπερβαίνουν τις παραδοσιακές μεθόδους οργάνωσης καθώς είναι σε θέση να αναγνωρίζουν πρότυπα, να προβλέπουν χρονικές απαιτήσεις, να ανιχνεύουν πιθανές συγκρούσεις στον χρονοπρογραμματισμό και να προτείνουν την ιδανική σειρά εκτέλεσης των εργασιών.

#### 1.1.4 Διαφορά με διαχείριση έργου

Συχνά η έννοια της διαχείρισης εργασιών (task management) συγχέεται με αυτή της διαχείρισης έργου (project management). Η αλήθεια είναι πως πρόκειται για έννοιες που όντως συσχετίζονται αλλά στην πραγματικότητα η καθεμία εστιάζει σε διαφορετικά αντικείμενα.

Η διαχείριση εργασιών, όπως έχει αναφέρθηκε, αφορά την παρακολούθηση διαφορετικών, μεμονομένων δραστηριοτήτων οι οποίες χρειάζεται να ολοκληρωθούν. Η διαχείριση εργασιών επικεντρώνεται στο μικροεπίπεδο, στη διαχείριση καθημερινών υποχρεώσεων, στα διαφορετικά deadlines που μπορεί να υπάρχουν, την εξέλιξή τους ανά το χρόνο κ.α. Τα εργαλεία που αφορούν τη διαχείριση εργασιών περιλαμβάνουν ημερολόγια, υπενθυμίσεις ή χρονοδιαγράμματα.

Αντίθετα, η διαχείριση έργου περιγράφει τον σχεδιασμό, την εκτέλεση και την ολοκλήρωση ενός ολόκληρου έργου. Ένα έργο αποτελείται και αυτό από διαφορετικές εργασίες, οι οποίες όμως είναι οργανωμένες προς έναν ευρύτερο στόχο. Επομένως,

η έννοια της διαχείρισης έργου συμπεριλαμβάνει τη διαχείριση εργασιών, αλλά επίσης προϋποθέτει επιπλέον απαιτήσεις όπως τη σωστή κατανομή πόρων (resource allocation) ή την αξιολόγηση κινδύνου (risk assessment). Τα λογισμικά διαχείρισης έργου έχουν λειτουργικότητες όπως διαγράμματα Γκαντ ή παρακολούθηση εξαρτήσεων.

Στην παρούσα διπλωματική εργασία για λόγους πληρότητας θα αναλυθούν και κάποιες έννοιες που αφορούν τη διαχείριση έργου, έχοντας όμως υπόψην ότι η υλοποίηση της εφαρμογής αφορά τη διαχείριση εργασιών.

## 1.2 Ιστορική αναδρομή

Η διαχείριση και ο προγραμματισμός εργασιών αποτελούν έννοιες που υπήρχαν ήδη από την αρχαιότητα, πολύ πριν την ανάπτυξη φηφιακών εργαλείων και αυτοματισμών, με τις πρώτες μορφές οργάνωσης να βασίζονται κυρίως στον προφορικό λόγο και την ανθρώπινη μνήμη. Με την ανάγκη για καταγραφή και παρακολούθηση, αναπτύχθηκαν και χρησιμοποιήθηκαν διάφορες μνημονικές συσκευές, οι οποίες προσέφεραν στους πολιτισμούς της εποχής τρόπους οργάνωσης και αποθήκευσης κρίσιμων πληροφοριών.

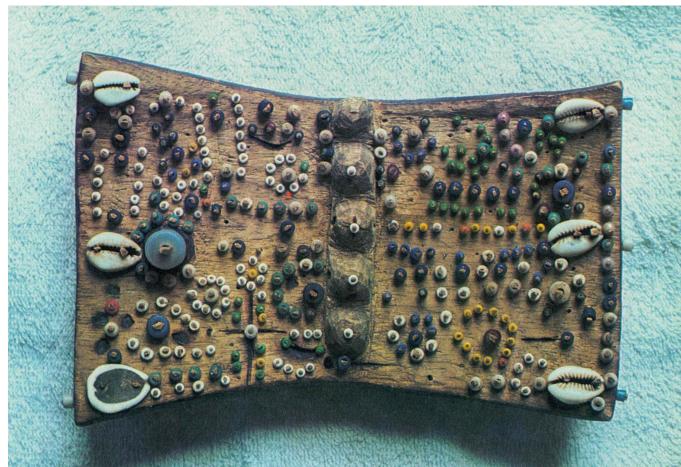
Στην ενότητα αυτή, θα ασχοληθούμε με τις πρώτες καταγεγγραμμένες μορφές διαχείρισης εργασιών, την εξέλιξή τους μέχρι σήμερα όπως επίσης και τις σημαντικές καινοτομίες και μεθοδολογίες που αναπτύχθηκαν στην πορεία της ιστορίας, όπως το διάγραμμα Gantt και άλλες οργανωτικές τεχνικές, οι οποίες εξέλιξαν τη διαχείριση και τον προγραμματισμό των εργασιών.

### 1.2.1 Προφορικότητα και μνημονικές συσκευές

Στην αρχαιότητα, η διαχείριση των εργασιών βασιζόταν κυρίως στον προφορικό λόγο, ο οποίος αποτελούσε το κύριο μέσο μετάδοσης πληροφοριών και οδηγιών. Έτσι οι εργασίες αναθέτονταν μέσω προφορικών οδηγιών, ενώ η ακρίβεια της εκτέλεσης εξαρτιόταν σε μεγάλο βαθμό από την αξιοπιστία της ανθρώπινης μνήμης. Αυτό το σύστημα, αν και ήταν η μόνη επιλογή που υπήρχε εκείνη την εποχή, παρουσίαζε σοβαρά μειονεκτήματα, καθώς η μνήμη είναι επιρρεπής σε λάθη, ειδικά σε καταστάσεις που απαιτούν τη διαχείριση πολλαπλών ή σύνθετων εργασιών. Με λίγα λόγια, η εξάρτηση από την ανθρώπινη μνήμη περιορίζε την ακρίβεια και τη δυνατότητα αποτελεσματικής οργάνωσης, ιδίως σε περιπτώσεις που οι εργασίες ήταν περίπλοκες, έπρεπε να εκτελεστούν σε μεγάλα χρονικά διαστήματα ή αφορούσαν πολλά άτομα. [7]

Αυτή η αναγκαιότητα οδήγησε στην ανάπτυξη τεχνικών απομνημόνευσης και συστημάτων καταγραφής, που είχαν ως στόχο τη βελτίωση της διαχείρισης πληροφοριών και την αύξηση της αξιοπιστίας. Τέτοιες τεχνικές περιλάμβαναν τη χρήση επαναλαμβανόμενων φράσεων και ρυθμικών μοτίβων για την ευκολότερη απομνημόνευση οδηγιών. Επιπλέον, η δημιουργία χειροκίνητων συσκευών, όπως το λουκάσα (lukasa) από τους Λούμπα του Κονγκό και το κουίπου (quipu) από τους Ίνκα, εισήγαγε συστήματα

που υποκαθιστούσαν εν μέρει την ανθρώπινη μνήμη, παρέχοντας μια οπτικοποιημένη αναπαράσταση πληροφοριών. Αυτές οι συσκευές δεν ήταν απλώς εργαλεία καταγραφής, αλλά αποτελούσαν καινοτόμες λύσεις διαχείρισης εργασιών, ενισχύοντας την ικανότητα ανάλησης και οργάνωσης πληροφοριών.



Σχήμα 1.2: Η συσκευή λουκάσα



Σχήμα 1.3: Η συσκευή κουίπου

Το λουκάσα αποτελούνταν από πολύχρωμες χάντρες τοποθετημένες σε συγκεκριμένες θέσεις πάνω σε ξύλινες ή δερμάτινες επιφάνειες, προσφέροντας στους χειριστές έναν τρόπο να αποθηκεύουν, να οργανώνουν και να ανακαλούν πληροφορίες. [20] Το κουίπου ήταν μια κατασκευή με χορδές από βαμβάκι ή μαλλί. Οι χορδές ήταν πολύχρωμες με κόμπους, επιτρέποντας έτσι την κατηγοριοποίηση και αποθήκευση πληροφοριών βάσει χρώματος, διάταξης και αριθμού. Οι Ίνκα δημιουργούσαν κόμπους στις χορδές και τις χρησιμοποιούσαν για τη συλλογή και παρακολούθηση των υποχρεώσεών τους ή και για την αποθήκευση άλλων πληροφοριών όπως δεδομένα

απογραφής, φορολογικών υποχρεώσεων και άλλα. [19]

Η δημιουργία τέτοιων τεχνικών και συστημάτων καταγραφής αναδεικνύει την ανθρώπινη ικανότητα να προσαρμόζεται σε πρακτικές ανάγκες και να δημιουργεί καινοτόμες λύσεις. Αυτά τα πρώιμα μέσα διαχείρισης εργασιών όχι μόνο κάλυψαν τις απαιτήσεις της εποχής, αλλά έθεσαν τα θεμέλια για τη μεταγενέστερη ανάπτυξη γραπτών και, τελικά, ψηφιακών συστημάτων, που επανακαθόρισαν τον τρόπο με τον οποίο οργανώνουμε και εκτελούμε εργασίες στις μέρες μας.

### 1.2.2 Πρώτα ημερολόγια

Κατασκευές όπως τα ηλιακά ρολόγια αποτέλεσαν ένα από τα πρώτα εργαλεία που έδωσαν στους ανθρώπους τη δυνατότητα να διαιρέσουν την ημέρα σε διαχριτά τμήματα. Η ανακάλυψη αυτών των εργαλείων αποτέλεσε καθοριστικό βήμα στην κατανόηση του χρόνου ως δομημένου και μετρήσιμου πόρου, επιτρέποντας στους πληθυσμούς να οργανώσουν καλύτερα τις καθημερινές τους δραστηριότητες. Η δυνατότητα αυτή οδήγησε σε μια σαφή διάκριση μεταξύ υποχρεώσεων και ελεύθερου χρόνου, καθώς οι άνθρωποι άρχισαν να προγραμματίζουν τις ώρες της ημέρας με μεγαλύτερη ακρίβεια. Αυτός ο διαχωρισμός ήταν θεμελιώδης για την ανάπτυξη πιο σύνθετων συστημάτων διαχείρισης του χρόνου, καθώς οι κοινότητες αντιλήφθηκαν τη σημασία του χρονοπρογραμματισμού για τη βελτιστοποίηση των συλλογικών τους δραστηριοτήτων.



Σχήμα 1.4: Το παλαιότερο γνωστό ηλιακό ρολόι από τους Αιγυπτίους· χρησιμοποιούνταν για να μετράει τις ώρες εργασίας τους

Παράλληλα με τα ηλιακά ρολόγια, οι πρώιμες προσπάθειες δημιουργίας ημερολογίων συνέβαλαν καθοριστικά στην εξέλιξη της διαχείρισης εργασιών και χρόνου. Πολιτισμοί όπως οι Αιγύπτιοι, οι Ρωμαίοι και οι Μάγια ανέπτυξαν πολύπλοκα συστήματα ημερολογίων που βασίζονταν στις κινήσεις του ήλιου, της σελήνης και των άστρων. Αυτά τα ημερολόγια όχι μόνο προσδιόριζαν τον χρόνο για γεωργικές δρα-

στηριότητες, όπως η σπορά και η συγκομιδή, αλλά χρησίμευαν και ως οδηγός για θρησκευτικές τελετές, κοινωνικές εκδηλώσεις και άλλες τελετουργίες. Μέσω αυτών των εργαλείων, έγινε εφικτός ο διαχωρισμός του χρόνου προσφέροντας μια πρώιμη μορφή συστηματικής οργάνωσης που συνέβαλε στην ανάπτυξη των κοινωνιών. [21]

Η τεχνολογική πρόοδος έφερε σημαντικές εξελίξεις στον τρόπο μέτρησης και διαχείρισης του χρόνου. Τα ηλιακά ρολόγια, τα οποία ήταν εξαρτώμενα από την παρουσία του ήλιου, σταδιακά εξελίχθηκαν σε μηχανικά ρολόγια, τα οποία μπορούσαν να λειτουργούν ανεξάρτητα από τις καιρικές συνθήκες ή την ώρα της ημέρας. Αυτή η μετάβαση στα μηχανικά ρολόγια σηματοδότησε μια νέα εποχή για τον χρονοπρογραμματισμό. Τα μηχανικά ρολόγια προσέφεραν μεγαλύτερη ακρίβεια και έθεσαν τα θεμέλια για την ανάπτυξη πιο σύνθετων εργαλείων διαχείρισης εργασιών, που θα μπορούσαν να εξυπηρετήσουν τις αυξανόμενες απαιτήσεις των κοινωνιών.

### 1.2.3 Σύγχρονη εποχή

Η οργανωμένη διαχείριση εργασιών έχει τις ρίζες της βαθιά μέσα στην ιστορία, καθώς οι άνθρωποι πάντα αναζητούσαν τρόπους να οργανώσουν καλύτερα τις δραστηριότητές τους. Ωστόσο, οι πρώτες προσπάθειες τυποποίησης αυτής της διαδικασίας εντοπίζονται στον 18ο αιώνα, όταν η ανάγκη για μια συστηματική προσέγγιση έγινε πιο έντονη λόγω της ανάπτυξης των κοινωνιών και της αυξανόμενης πολυπλοκότητας των έργων. Στα τέλη του 19ου αιώνα, η βιομηχανική επανάσταση επέφερε τεράστιες αλλαγές στον τρόπο παραγωγής και κατασκευής. Τα έργα μεγάλης κλίμακας, όπως σιδηροδρομικά δίκτυα, γέφυρες και εργοστάσια, απαιτούσαν πιο οργανωμένες προσεγγίσεις στη διαχείριση ανθρώπινου δυναμικού και πόρων. Αυτές οι νέες απαιτήσεις οδήγησαν στην ανάγκη για πιο λεπτομερή και αποτελεσματική διαχείριση των εργασιών. Όμως η οργάνωση χιλιάδων εργατών, η διαχείριση μεγάλων ποσοτήτων πρώτων υλών και η τήρηση αυστηρών χρονοδιαγραμμάτων ήταν προκλήσεις που δε θα μπορούσαν να αντιμετωπιστούν με τις παραδοσιακές τεχνικές.

Μηχανικοί όπως ο Henry Gantt εισήγαγαν πρωτοποριακές μεθόδους οργάνωσης, όπως το **διάγραμμα Γκαντ**. Το διάγραμμα Γκαντ είναι ένα εργαλείο που παρέχει οπτικοποίηση, αναπαριστώντας όλες τις επιμέρους εργασίες ενός έργου κατά μήκος ενός χρονικού άξονα, παρέχοντας μια καθαρή εικόνα των φάσεων υλοποίησής του. Με αυτόν τον τρόπο, οι υπεύθυνοι έργων μπορούσαν να παρακολουθούν την πρόοδο κάθε φάσης, να εντοπίζουν πιθανές καθυστερήσεις και να αναπροσαρμόζουν τον προγραμματισμό όπου ήταν απαραίτητο. Ένα από τα μεγαλύτερα πλεονεκτήματα του διαγράμματος Γκαντ ήταν η δυνατότητα προσδιορισμού της κρίσιμης διαδρομής του έργου, δηλαδή της αλληλουχίας των εργασιών που πρέπει να ολοκληρωθούν εντός συγκεκριμένων χρονικών ορίων για να διασφαλιστεί η έγκαιρη ολοκλήρωση του έργου. Θα αναφερθούμε με μεγαλύτερη λεπτομέρεια στο διάγραμμα Γκαντ στην ενότητα 1.4. Πάντως, το εργαλείο αυτό ήταν καθοριστικό σε μεγάλα έργα υποδομών, όπως η κατασκευή της Διώρυγας του Παναμά, που αποτέλεσε ένα από τα πιο φιλόδοξα

και απαιτητικά έργα της εποχής, καθώς και το φράγμα Χούβερ, το οποίο απαίτησε σχολαστικό σχεδιασμό και συντονισμό πόρων σε πρωτόγνωρη κλίμακα. [9]



Σχήμα 1.5: Το φράγμα Χούβερ [10]

Η εισαγωγή μεθοδολογικών εργαλείων όπως το διάγραμμα Γκαντ δεν περιορίστηκε μόνο στη βιομηχανία και τα έργα υποδομών· αποτέλεσε την έμπνευση για νέες έρευνες και πρακτικές που επεκτάθηκαν σε διάφορους τομείς. Ένα από τα πιο χαρακτηριστικά παραδείγματα είναι το **Πρότζεκτ Μανχάταν** (Manhattan Project), το οποίο σχεδιάστηκε κατά τη διάρκεια του Β' Παγκοσμίου Πολέμου για το σχεδιασμό πυρηνικών όπλων. Αυτό το ιδιαίτερα απαιτητικό έργο προκάλεσε την ανάπτυξη δύο νέων μοντέλων διαχείρισης, του PERT (Program Evaluation and Review Technique) και του CPM (Critical Path Method). Το PERT σχεδιάστηκε για να αντιμετωπίσει την αβεβαιότητα στις εκτιμήσεις του χρόνου υλοποίησης των εργασιών, ενώ το CPM επικεντρώθηκε στην ανάλυση και τη βελτιστοποίηση της κρίσιμης διαδρομής του έργου. [3]. Θα αναφερθούμε και σε αυτά τα μοντέλα στην ενότητα 1.4.

### 1.3 Η συνδρομή της τεχνολογίας

Από τη δεκαετία του '60 και έπειτα, οι επιχειρήσεις άρχισαν να αναγνωρίζουν την αξία της συστηματικής και μεθοδικής οργάνωσης της εργασίας. Η φημιακή επανάσταση που ακολούθησε δε θα μπορούσε παρά να γιγαντώσει αυτή τη νέα πραγματικότητα. Η είσοδο των υπολογιστών, επέφερε και νέες δυνατότητες αποθήκευσης και ανάλυσης δεδομένων, δυνατότητες που άλλαξαν ριζικά τη διαχείριση των εργασιών. Έτσι, διαδικασίες που προηγουμένως απαιτούσαν χρονοβόρα χειρωνακτική εργασία και εκτεταμένη χρήση χαρτιού, έγιναν πλέον πιο γρήγορες και πιο ακριβείς.

Η τεχνολογική πρόοδος έφερε νέα εργαλεία και λογισμικά που ενίσχυσαν τη συνεργασία μεταξύ ομάδων και τμημάτων, όπως το Microsoft Project και αργότερα οι πλατφόρμες συνεργασίας τύπου Trello και Asana, επέτρεψαν σε ομάδες διαφορετι-

κών γεωγραφικών περιοχών να συνεργάζονται απρόσκοπτα, μειώνοντας τα εμπόδια επικοινωνίας, ενώ πρόσθεσε και αυτοματισμούς στην κατανομή εργασιών και στη δημιουργία χρονοδιαγραμμάτων. Η τεχνολογία όχι μόνο βελτίωσε τη λειτουργικότητα των εργαλείων διαχείρισης αλλά τα έκανε επίσης πιο προσιτά σε μικρότερες επιχειρήσεις, που προηγουμένως δεν είχαν τη δυνατότητα να επενδύσουν σε τέτοιες λύσεις.

### 1.3.1 Ψηφιακά εργαλεία

Με την εξέλιξη της τεχνολογίας, οι σημειώσεις και η οργάνωση μεταφέρθηκε από τις χειρόγραφες σημειώσεις, τα ημερολόγια και τα έγγραφα των γραφομηχανών σε ψηφιακά εργαλεία. Παραδείγματα αυτών σε ατομικό επίπεδο είναι το Todoist ή το Notion και σε επαγγελματικό επίπεδο προγράμματα σαν το Microsoft Project.

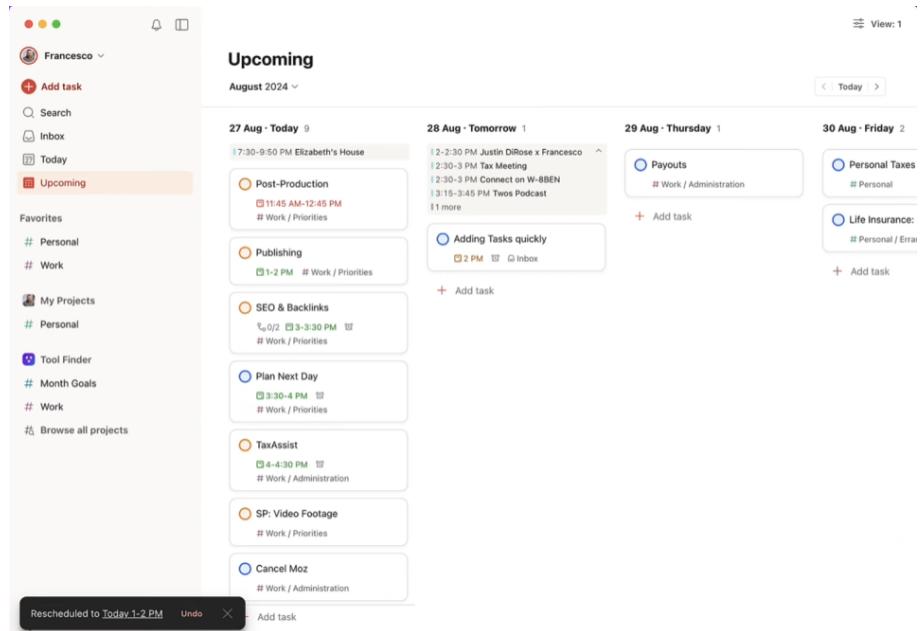
#### 1.3.1.1 Todoist

Αν και έχει χάσει πλέον την πρωτοκαθεδρία του, το Todoist [28] παραμένει μια από τις πιο δημοφιλείς εφαρμογές διαχείρισης εργασιών, σχεδιασμένη για να βοηθάει τόσο απλούς χρήστες όσο και επαγγελματίες να οργανώνουν τις υποχρεώσεις τους και να βελτιώνουν την παραγωγικότητά τους. Η εφαρμογή επιτρέπει τη δημιουργία λιστών εργασιών με δυνατότητα ομαδοποίησης σε έργα, υποέργα ή ετικέτες (tags). Οι χρήστες μπορούν να καθορίσουν ημερομηνίες και προθεσμίες καθώς και να ρυθμίσουν υπενθυμίσεις, καταφέροντας έτσι την αποδοτικότερη οργάνωση του χρόνου τους, ενώ οι ειδοποιήσεις τους διασφαλίζουν ότι δε θα παραλείψουν καμία κρίσιμη εργασία. Επιπλέον, περιλαμβάνει σύστημα επιβράβευσης (“Karma”) ενθαρρύνοντας τη συνέπεια και την ολοκλήρωση των εργασιών, ενώ υπάρχει η δυνατότητα ενσωμάτωσης με εξωτερικές εφαρμογές όπως το Google Calendar.

#### 1.3.1.2 Notion

Το Notion [33] είναι αυτή τη στιγμή ίσως η πιο δημοφιλής πλατφόρμα προσωπικής οργάνωσης. Συνδυάζει στοιχεία για task-management, note-taking, βάσεις δεδομένων και συνεργατικότητας σε μία ενιαία εφαρμογή, καθιστώντας το ιδανικό για χρήστες που επιζητούν έναν κεντρικό χώρο για τη διαχείριση της εργασιακής ή προσωπικής τους ζωής. Το κύριο χαρακτηριστικό του Notion είναι η δυνατότητα δημιουργίας προσαρμοσμένων σελίδων χρησιμοποιώντας Markdown γλώσσα, στις οποίες οι χρήστες μπορούν να μορφοποιήσουν το κείμενο, να προσθέσουν πίνακες ή να ενσωματώσουν διάφορα στοιχεία όπως λίστες εργασιών, πίνακες Kanban, ημερολόγια, checklists, ή ακόμη και embedded αρχεία. Αυτή η ευελιξία επιτρέπει τη δημιουργία εξατομικευμένων συστημάτων διαχείρισης, προσαρμοσμένων στις μοναδικές ανάγκες του κάθε χρήστη, λειτουργώντας ως μια προσωπική εγκυλοπαίδεια.

Επίσης, υπάρχει η δυνατότητα δημιουργίας βάσεων δεδομένων (οι οποίες μπορούν να λειτουργήσουν ως λίστες εργασιών, παρακολούθησης προόδου για έργα κ.α.) τις οποίες μπορούν να φιλτράρουν, να ταξινομούν και να χειρίζονται όπως θέλουν. Τέλος,



Σχήμα 1.6: Στιγμιότυπο του Todoist

Σχήμα 1.7: Στιγμιότυπο του Notion

υπάρχει δυνατότητα για συνεργατική κοινή χρήση σελίδων, την ενσωμάτωση με άλλα εργαλεία όπως το Google Drive ή το Slack.

### 1.3.1.3 Microsoft Project

Πρόκειται για ένα από τα πρώτα λογισμικά διαχείρισης έργων, σχεδιασμένα για το κοινό. Η ιδέα για τη δημιουργία του προήλθε από μια φάρσα του Ron Bredehoeft,

ο οποίος, θέλοντας να αναπαραστήσει τη διαδικασία παρασκευής αυγών μπένεντικτ σε όρους διαχείρισης έργων, ανέπτυξε την ιδέα για ένα εργαλείο που θα μπορούσε να χρησιμοποιηθεί για την οργάνωση και τον προγραμματισμό οποιουδήποτε έργου. Το Microsoft Project παρουσιάστηκε για πρώτη φορά το 1984 ως εφαρμογή για DOS, κερδίζοντας αμέσως την προσοχή των επαγγελματιών. Σήμερα, αποτελεί ένα από τα πιο καθιερωμένα εργαλεία για την οργάνωση, τον χρονοπρογραμματισμό και την παρακολούθηση έργων, με εφαρμογές σε πλήθος βιομηχανιών, από την κατασκευή μέχρι την πληροφορική και τη διαχείριση ανθρώπινων πόρων.

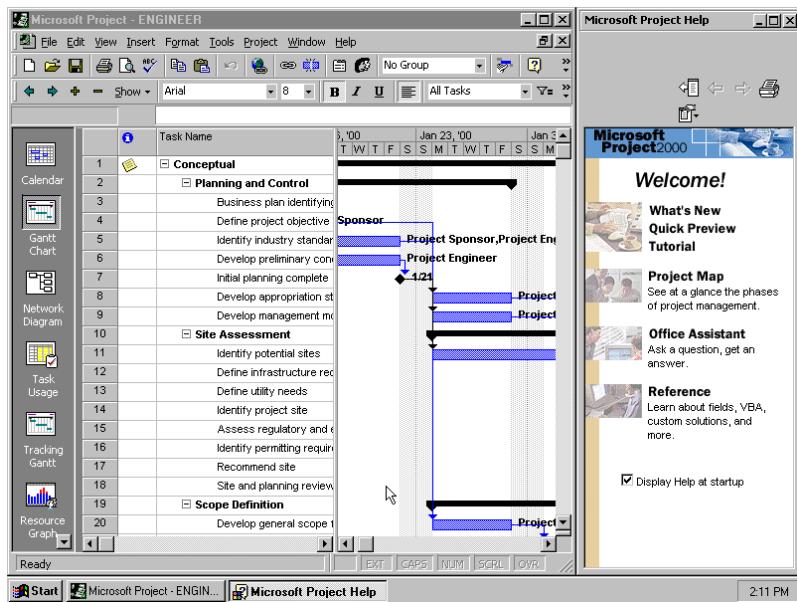


Σχήμα 1.8: Στιγμιότυπο από το Microsoft Project 3.0 (σε DOS) [32]

Η κεντρική οθόνη του λογισμικού χωρίζεται σε δύο βασικές περιοχές: το διάγραμμα Γκαντ (Gantt chart) και τον πίνακα εισαγωγής εργασιών (input table). Ο πίνακας εισαγωγής επιτρέπει την εισαγωγή λεπτομερών πληροφοριών σχετικά με κάθε εργασία, όπως η διάρκεια, οι εξαρτήσεις και οι πόροι.

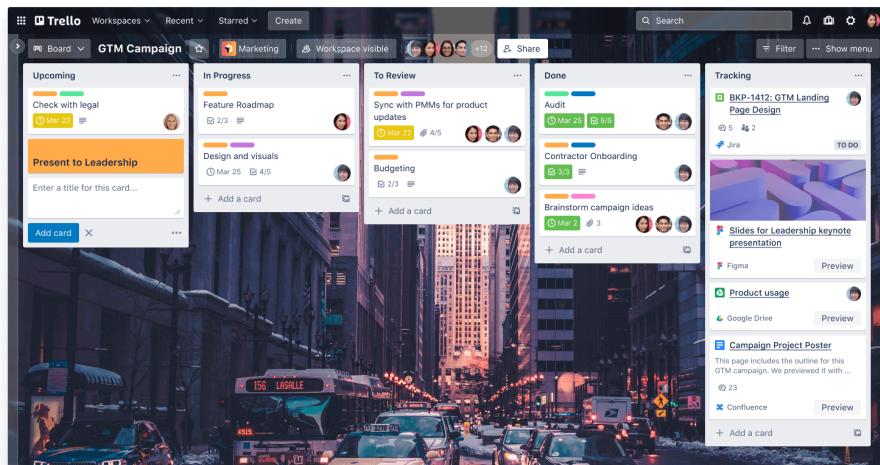
Παρέχονται λειτουργικότητες όπως η δυνατότητα ιεράρχησης των εργασιών μέσω της τοποθέτησης εσοχών (indents) (δημιουργώντας μιας σαφούς δομής του έργου, διευκολύνοντας τη διαχείριση πολύπλοκων έργων με πολλές υποκατηγορίες), η δημιουργία εξαρτήσεων μεταξύ των εργασιών (predecessors) (για παράδειγμα, μια εργασία μπορεί να προγραμματιστεί να ξεκινήσει μόνο όταν ολοκληρωθεί μια άλλη), η δυνατότητα αυτόματου προγραμματισμού των εργασιών (auto-scheduling), η δυνατότητα δημιουργίας αλυσιδωτών εργασιών (linked tasks), στις οποίες μια εργασία εκτελείται αμέσως μετά την ολοκλήρωση μιας άλλης, διευκολύνοντας τον προγραμματισμό μεγάλων και σύνθετων έργων. Επιπλέον, το λογισμικό προσφέρει ευελιξία στην προβολή των δεδομένων, επιτρέποντας την αποτύπωση των εργασιών πέρα από το διάγραμμα Γκαντ σε μορφή ημερολογίου, φύλλου εργασίας (task sheet) ή ακόμη και η δημιουργία στατιστικών αναφορών. Έτσι, για παράδειγμα μια ομάδα μπορεί να επιλέξει το ημερολόγιο για να βλέπει τις ημερήσιες υποχρεώσεις της, ενώ ένας διευθυντής μπορεί να επιλέξει τις στατιστικές αναφορές για να αξιολογήσει τη συνολική

πρόοδο.



Σχήμα 1.9: Στιγμιότυπο από το Microsoft Project 2000 [32]

#### 1.3.1.4 Trello



Σχήμα 1.10: Στιγμιότυπο του Trello

To Trello [14] είναι μια πλατφόρμα διαχείρισης εργασιών που βασίζεται στους πίνακες Kanban (θα αναλυθούν στην ενότητα 1.4), επιτρέποντας μια εύληπτη οργάνωση της καθημερινότητας. Με τη χρήση πινάκων, λιστών και καρτών, οι χρήστες μπορούν συνεργατικά να παρακολουθούν την πρόοδο των εργασιών τους, ιδιαίτερα για εργασίες που χρειάζονται ομαδική συνεργασία, καθιστώντας το ιδιαίτερα δημοφιλές σε

ομάδες όλων των μεγεθών και σε διάφορους τομείς. Κάθε πίνακας αντιπροσωπεύει ένα πρότζεκτ, ενώ οι λίστες μπορούν να χρησιμοποιηθούν για την κατηγοριοποίηση των εργασιών σε στάδια (“To Do”, “In Progress”, “Done”). Οι κάρτες, που τοποθετούνται μέσα στις λίστες, αντιπροσωπεύουν συγκεκριμένες εργασίες που πρέπει να ολοκληρωθούν. Οι χρήστες μπορούν να προσθέτουν περιγραφές, checklists, προθεσμίες, συνημμένα αρχεία, και ετικέτες (labels) στις κάρτες, επιτρέποντας την εξατομίκευση και την οργάνωση κάθε εργασίας σύμφωνα με τις ανάγκες τους. Προσφέρονται εργαλεία αυτοματοποίησης (λειτουργία “Butler”) και υπάρχουν δυνατότητες ενσωμάτωσης με άλλες εφαρμογές.

## 1.4 Μεθοδολογίες

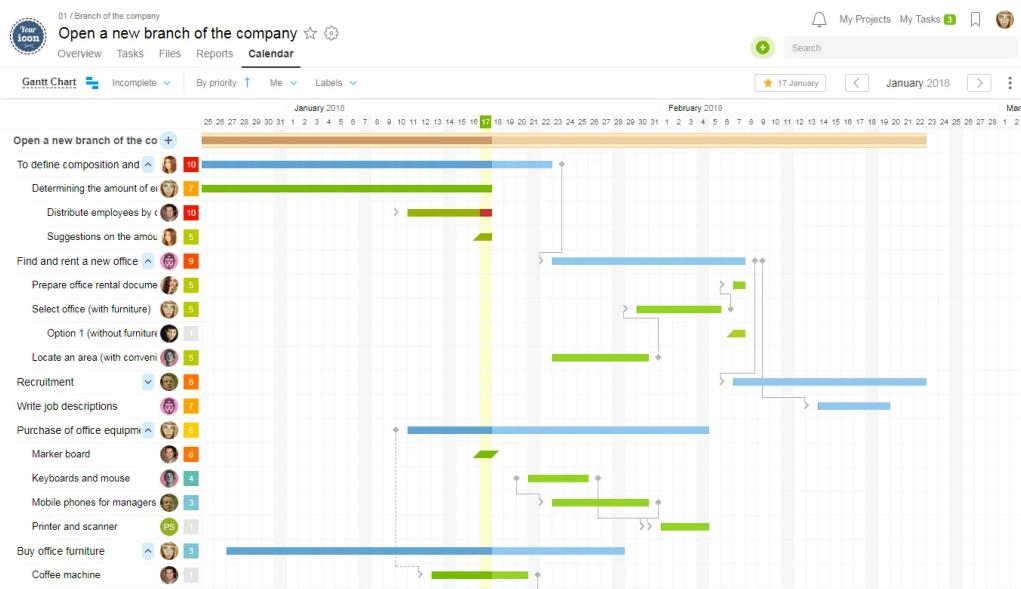
### 1.4.1 Διάγραμμα Γκαντ

Το διάγραμμα Γκαντ (Gantt chart) έχει καθιερωθεί ως ένα από τα πιο χρήσιμα εργαλεία στη διαχείριση έργων, καθώς παρέχει μια εύληπτη γραφική αναπαράσταση των επιμέρους εργασιών ενός έργου. Στον οριζόντιο άξονα απεικονίζεται ο χρόνος, ενώ στον κατακόρυφο άξονα παρατίθενται οι διαφορετικές εργασίες που συνθέτουν το έργο. Για τη δημιουργία ενός διαγράμματος Γκαντ, είναι απαραίτητος ο αρχικός καταμερισμός του έργου σε επιμέρους εργασίες. Αυτό περιλαμβάνει την αναλυτική καταγραφή κάθε δραστηριότητας που πρέπει να ολοκληρωθεί και την εκτίμηση της χρονικής διάρκειας που θα απαιτηθεί για την ολοκλήρωσή της. Αφού γίνει ο καταμερισμός, οι εργασίες τοποθετούνται στο διάγραμμα με τέτοιο τρόπο ώστε αυτές που ολοκληρώνονται νωρίτερα να βρίσκονται ψηλότερα, διατηρώντας μια σαφή δομή που διευκολύνει την ανάγνωση και την κατανόηση του χρονοδιαγράμματος.

Η ευκολία και η ταχύτητα με την οποία μπορεί να κατασκευαστεί ένα διάγραμμα Γκαντ αποτελούν έναν από τους κύριους λόγους για τη δημοτικότητά του. Παρέχει μια σαφή απεικόνιση της χρονικής διάρκειας και της αλληλουχίας των εργασιών, κάνοντας τη χρήση του προσιτή ακόμη και για άτομα που δεν έχουν εξειδικευμένες γνώσεις στη διαχείριση έργων.

Παρόλα αυτά, το διάγραμμα Γκαντ έχει και ορισμένους περιορισμούς. Ένας από αυτούς είναι η αδυναμία του να αποτυπώσει τις εξαρτήσεις μεταξύ των εργασιών και την επίδραση της καθυστέρισης μιας εργασίας στο συνολικό έργο. Για παράδειγμα, δεν είναι εμφανές ποιες εργασίες πρέπει να ολοκληρωθούν πριν ξεκινήσει μια άλλη, κάτι που μπορεί να οδηγήσει σε παρανοήσεις ή καθυστερήσεις αν δεν υπάρχει κατάλληλος συντονισμός. Επιπλέον, η στατική του φύση περιορίζει τη δυνατότητα αναπροσαρμογής όταν οι συνθήκες αλλάξουν, όπως σε περιπτώσεις μεταβολής της διάρκειας μιας εργασίας ή προσθήκης νέων δραστηριοτήτων. Αυτό σημαίνει ότι, ενώ είναι εξαιρετικό για την αρχική φάση σχεδιασμού και την παρακολούθηση, μπορεί να μην επαρκεί σε δυναμικά περιβάλλοντα όπου απαιτείται συνεχής αναπροσαρμογή. Παρόλα αυτά, παραμένει ένα από τα πιο χρήσιμα εργαλεία για την κατανόηση της

χρονικής διάστασης ενός έργου και τη συνολική εποπτεία της προόδου του. [34]



Σχήμα 1.11: Παράδειγμα διαγράμματος Γκαντ

#### 1.4.2 Program evaluation and review technique (PERT)

To **Program Evaluation and Review Technique (PERT)** συνδυαστικά και με τη μέθοδο κρίσιμης διαδρομής (Critical Path Method – CPM) που θα αναλυθεί αμέσως μετά, είναι μια μεθοδολογία προγραμματισμού και ελέγχου έργων που επικεντρώνεται στον υπολογισμό και την αξιολόγηση του χρόνου ολοκλήρωσης ενός έργου, ενώ παράλληλα παρέχει σαφή εικόνα των σχέσεων και των εξαρτήσεων μεταξύ των δραστηριοτήτων του. Αναπτύχθηκε τη δεκαετία του 1950 για την υποστήριξη σύνθετων έργων με υψηλή αβεβαιότητα, όπως ο προγραμματισμός στρατιωτικών προγραμμάτων.

Στο PERT, το έργο αναλύεται σε επιμέρους δραστηριότητες, καθεμία από τις οποίες απεικονίζεται ως κλάδος σε ένα διάγραμμα δικτύου. Οι κόμβοι του διαγράμματος αντιπροσωπεύουν γεγονότα που σηματοδοτούν την αρχή ή τη λήξη αυτών των δραστηριοτήτων. Αυτή η δομή επιτρέπει την οπτικοποίηση των σχέσεων και των εξαρτήσεων μεταξύ των δραστηριοτήτων, καθιστώντας σαφές ποιες πρέπει να ολοκληρωθούν πρώτα και ποιες μπορούν να εκτελούνται παράλληλα.

Το PERT βασίζεται στην εκτίμηση του χρόνου για κάθε δραστηριότητα χρησιμοποιώντας τρεις παραμέτρους: τον αισιόδοξο χρόνο (ο ελάχιστος χρόνος ολοκλήρωσης), τον πιο πιθανό χρόνο και τον απαισιόδοξο χρόνο (ο μέγιστος χρόνος ολοκλήρωσης). Με βάση αυτές τις εκτιμήσεις, υπολογίζεται ο αναμενόμενος χρόνος για κάθε δραστηριότητα, λαμβάνοντας υπόψη την αβεβαιότητα.

Μια σημαντική έννοια στο PERT είναι ο υπολογισμός της κρίσιμης διαδρομής,

δηλαδή της αλληλουχίας δραστηριοτήτων που καθορίζει τη συνολική διάρκεια του έργου. Οι δραστηριότητες που βρίσκονται στην κρίσιμη διαδρομή δεν επιτρέπουν καθυστερήσεις, καθώς οποιαδήποτε καθυστέρηση σε αυτές θα παρατείνει το συνολικό χρονοδιάγραμμα του έργου. Αντίθετα, οι μη κρίσιμες δραστηριότητες έχουν ένα χρονικό περιθώριο (slack), το οποίο τους επιτρέπει κάποιες καθυστερήσεις χωρίς να επηρεαστεί το συνολικό έργο.

Η μεθοδολογία PERT παρέχει σημαντικά εργαλεία για την ανάλυση του χρόνου ολοκλήρωσης του έργου, όπως ο υπολογισμός του νωρίτερου και του αργότερου χρόνου για κάθε γεγονός και δραστηριότητα. Με τη βοήθεια αυτών των υπολογισμών, οι διαχειριστές έργων μπορούν να προβλέψουν την ελάχιστη και μέγιστη διάρκεια του έργου, να αναγνωρίσουν κρίσιμα σημεία και να σχεδιάσουν κατάλληλα τις ενέργειες τους για την αντιμετώπιση πιθανών καθυστερήσεων.

Το PERT, παρόλο που είναι ιδιαίτερα χρήσιμο σε έργα με πολλές αβεβαιότητες, παρουσιάζει ορισμένους περιορισμούς. Οι εκτιμήσεις χρόνου συχνά βασίζονται σε υποκειμενικά δεδομένα, γεγονός που μπορεί να οδηγήσει σε αποκλίσεις. Παρόλα αυτά, η εφαρμογή του PERT συμβάλλει σημαντικά στη διαχείριση έργων, διευκολύνοντας τη λήψη αποφάσεων και την κατανομή πόρων με τρόπο αποτελεσματικό. [13]

#### 1.4.3 Μέθοδος κρίσιμης διαδρομής (Critical Path Method – CPM)

#### 1.4.4 Ευέλικτη (Agile) μεθοδολογία

Το Agile αποτελεί μια φιλοσοφία διαχείρισης έργων που στοχεύει στη βελτιστοποίηση της συνεργασίας, της προσαρμοστικότητας και της αποτελεσματικότητας. Η ευέλικτη αυτή προσέγγιση πρωτοεμφανίστηκε το 2001 με τη δημοσίευση του Agile Manifesto, το οποίο θέτει τέσσερις κεντρικές αξίες που εστιάζουν στους ανθρώπους και τις αλληλεπιδράσεις, τη λειτουργικότητα του προϊόντος, τη συνεργασία με τους πελάτες και την προσαρμογή στις αλλαγές. Αυτές οι αξίες αντικατοπτρίζουν τη φιλοσοφία ότι η ανάπτυξη λογισμικού και η διαχείριση έργων απαιτούν ευελιξία και συνεχή ανατροφοδότηση για να ανταποκρίνονται στις ανάγκες της αγοράς και των πελατών.

Η βασική αρχή του Agile είναι η διαίρεση της εργασίας σε μικρότερες, διαχειρίσιμες επαναλήψεις, γνωστές ως sprints. Κάθε sprint έχει συγκεκριμένη διάρκεια, συνήθως δύο έως τέσσερις εβδομάδες, και στο τέλος του παραδίδεται λειτουργικό λογισμικό ή προϊόν, το οποίο μπορεί να αξιολογηθεί από τον πελάτη. Αυτός ο τρόπος εργασίας επιτρέπει την ενσωμάτωση των σχολίων και τη γρήγορη προσαρμογή στις νέες απαιτήσεις. Η συνεργασία είναι κεντρικό στοιχείο του Agile, με τις ομάδες να εργάζονται στενά και διαρκώς με τους πελάτες και τα ενδιαφερόμενα μέρη για την επίτευξη του καλύτερου δυνατού αποτελέσματος.

Μια από τις πιο δημοφιλείς πρακτικές που υιοθετούνται στο πλαίσιο του Agile είναι το Scrum, το οποίο οργανώνει την εργασία μέσω συγκεκριμένων ρόλων, συναντήσεων και εργαλείων, διατηρώντας μια σαφή εικόνα της προόδου. Επίσης, το

Extreme Programming (XP) και το Lean Software Development είναι άλλες μέθοδοι που βασίζονται στις αρχές του Agile, εστιάζοντας στην ποιότητα του κώδικα και την αποδοτικότητα αντίστοιχα.

### 1.4.5 Kanban

Το Kanban είναι μια μέθοδος διαχείρισης εργασιών που σχεδιάστηκε για να βελτιώνει τη ροή εργασιών και να ελαχιστοποιεί τις καθυστερήσεις. Η προσέγγιση αυτή αναπτύχθηκε αρχικά από την Toyota στον τομέα της παραγωγής, αλλά έχει πλέον υιοθετηθεί ευρέως στον τομέα της ανάπτυξης λογισμικού και άλλων έργων. Κεντρική ιδέα του Kanban είναι η οπτικοποίηση των εργασιών μέσω ενός πίνακα, όπου οι εργασίες αναπαρίστανται ως κάρτες που προχωρούν μέσα από διαφορετικά στάδια, όπως “To Do”, “In Progress” και “Done”.

Η βασική αρχή του Kanban είναι ο περιορισμός της εργασίας που βρίσκεται σε εξέλιξη, γνωστός ως Work In Progress (WIP). Αυτή η πρακτική αποτρέπει τις ομάδες από το να αναλαμβάνουν περισσότερες εργασίες από όσες μπορούν να διαχειριστούν, εξασφαλίζοντας την ομαλή ροή και την αποφυγή μποτιλιαρίσματος. Παράλληλα, η ανάλυση των δεδομένων της ροής, όπως ο χρόνος κύκλου (cycle time), δίνει τη δυνατότητα στις ομάδες να εντοπίζουν σημεία βελτίωσης και να αυξάνουν την αποδοτικότητά τους.

Το Kanban δε βασίζεται σε αυστηρά χρονικά πλαίσια ή επαναλαμβανόμενα στάδια, όπως το Agile, αλλά προσφέρει συνεχή ευελιξία και βελτίωση. Η σταδιακή εξέλιξη και η διαρκής παράδοση αξίας είναι στο επίκεντρο αυτής της μεθόδου, καθιστώντας την ιδανική για έργα που απαιτούν δυναμικές προσαρμογές χωρίς προκαθορισμένα sprints.

## 1.5 Διαχείριση εργασιών στο πανεπιστήμιο

### 1.5.1 Προβλήματα διαχείρισης που αντιμετωπίζουν οι φοιτητές

Σε πολλές περιπτώσεις, η αποτελεσματικότητα των φοιτητών καθορίζεται κυρίως από την οργάνωσή τους και τον σωστό προγραμματισμό τους, τόσο στις ακαδημαϊκές όσο και στις προσωπικές τους υποχρεώσεις. Η σωστή διαχείριση χρόνου και προτεραιοτήτων είναι καθοριστική για την αποφυγή άγχους, την αύξηση της παραγωγικότητας και την επίτευξη των στόχων τους. Παρόλα αυτά, οι φοιτητές συχνά έρχονται αντιμέτωποι με προκλήσεις, όπως η αναβλητικότητα και η έλλειψη σωστής οργάνωσης για την ολοκλήρωση των καθημερινών τους καθηκόντων. Οι ερευνητικές προσπάθειες στον τομέα αυτό αναδεικνύουν τα εμπόδια που αντιμετωπίζουν οι φοιτητές και προσφέρουν πολύτιμες πληροφορίες για τη βελτίωση των δεξιοτήτων διαχείρισης εργασιών.

Σε έρευνα [6] που διεξήχθη στο Πανεπιστήμιο του Τσουκούμπα της Ιαπωνίας, η οποία διερευνούσε τη διαχείριση του προγραμματισμού των εργασιών από την πλευρά

των φοιτητών, παρατηρήθηκε πως η πλειοψηφία τους αντιμετωπίζει δυσκολίες στην εκκίνηση μιας νέας εργασίας. Οι βασικοί λόγοι που καταγράφηκαν περιλαμβάνουν: α) την έλλειψη χρόνου (26,9%), β) την αγνόηση της εργασίας επειδή τη θεωρούσαν ελάσσονος σημασίας (15,7%), γ) επειδή τη ξέχασαν (12,3%), δ) λόγω κακής συνεργασίας (11,2%) και ε) επειδή ήταν κουραστική (8,9%). Οι τρεις πρώτοι λόγοι, που καλύπτουν το μεγαλύτερο ποσοστό (54,9%), αφορούν σε θέματα κακής οργάνωσης από την πλευρά των φοιτητών, υποδεικνύοντας την ανάγκη για αποτελεσματικότερα εργαλεία χρονοπρογραμματισμού.

Παράλληλα, μια διαφορετική έρευνα [30] καταδεικνύει πάλι πως το κυριότερο πρόβλημα που αντιμετωπίζουν οι φοιτητές είναι η σωστή δόμηση του προγράμματός τους. Παρατηρήθηκε ότι ο τρόπος με τον οποίο οργανώνουν το διάβασμά τους καθοδηγείται κυρίως από τις καταληκτικές ημερομηνίες των εργασιών τους, με αποτέλεσμα να παραμελούν άλλες σημαντικές ακαδημαϊκές δραστηριότητες, όπως η παρακολούθηση διαλέξεων. Αυτό οδηγεί σε ανισορροπία μεταξύ των ακαδημαϊκών τους υποχρεώσεων, επηρεάζοντας την απόδοσή τους συνολικά.

Από τις παραπάνω έρευνες προκύπτουν ορισμένα σημαντικά συμπεράσματα. Πρώτον, η έλλειψη οργανωτικών δεξιοτήτων παραμένει ένας από τους κύριους παράγοντες που εμποδίζουν την αποτελεσματική διαχείριση των εργασιών από τους φοιτητές. Οι λόγοι αυτοί συνδέονται συχνά με την αναβλητικότητα και την έλλειψη εργαλείων που θα μπορούσαν να βοηθήσουν στην αποδοτικότερη οργάνωση των καθημερινών τους υποχρεώσεων. Δεύτερον, η ανάγκη για ένα δομημένο σύστημα προγραμματισμού είναι εμφανής, καθώς θα μπορούσε να παρέχει σαφείς προτεραιότητες και να συμβάλει στη μείωση του άγχους που προκαλείται από τις καταληκτικές ημερομηνίες. Συνεπώς, ένα αποτελεσματικό σύστημα διαχείρισης και προγραμματισμού των εργασιών, προσαρμοσμένο στις ανάγκες των φοιτητών, μπορεί να λειτουργήσει ως βασικό εργαλείο για την ενίσχυση της παραγωγικότητας και της επιτυχίας τους.

### 1.5.2 Χαρακτηριστικά που οι φοιτητές θα επιθυμούσαν σε μια εφαρμογή

Σε έρευνα [30] που πραγματοποιήθηκε στο τμήμα Πληροφορικής του Πανεπιστημίου του Εδιμβούργου, αναδείχθηκαν κάποιες σημαντικές προτιμήσεις και απαιτήσεις της ακαδημαϊκής κοινότητας σχετικά με τη λειτουργικότητα των εφαρμογών διαχείρισης εργασιών. Η πλειοψηφία των συμμετεχόντων τόνισε τη σημασία της ύπαρξης ενός ενσωματωμένου ημερολογίου, το οποίο θα παρέχει τη δυνατότητα καταγραφής των ημερομηνιών έναρξης και λήξης για κάθε εργασία. Αυτή η λειτουργία θεωρήθηκε κρίσιμη για τη σωστή οργάνωση και τον προγραμματισμό των υποχρεώσεων, καθώς επιτρέπει στους χρήστες να έχουν σαφή εικόνα των προθεσμιών τους. Παράλληλα, υπογραμμίστηκε η αξία της χρωματικής ταξινόμησης (color-coding), η οποία διευκολύνει τη διάκριση μεταξύ διαφορετικών κατηγοριών ή τύπων εργασιών, ενισχύοντας τη διαφάνεια και την ευκολία χρήσης της εφαρμογής.

Επιπλέον, οι συμμετέχοντες επισήμαναν την ανάγκη για ειδοποιήσεις/γνωστοποιή-

σεις, οι οποίες θα λειτουργούν ως υπενθυμίσεις για τις επερχόμενες προθεσμίες ή τις εκκρεμότητες που απαιτούν άμεση προσοχή. Εξίσου σημαντική θεωρήθηκε η ύπαρξη το-δο λιστών, οι οποίες θα διαθέτουν λειτουργίες όπως ιεράρχηση των εργασιών, ομαδοποίηση σε κατηγορίες, και δυνατότητα εμφάνισης μπάρας προόδου. Αυτές οι δυνατότητες συμβάλλουν στην αποτελεσματικότερη παρακολούθηση της προόδου των εργασιών και στη δημιουργία μιας αίσθησης ολοκλήρωσης και επίτευξης στόχων.

Ιδιαίτερο ενδιαφέρον παρουσίασε η πρόταση για την ενσωμάτωση ενός συστήματος ανταμοιβής, το οποίο στοχεύει στην ενθάρρυνση των φοιτητών να ολοκληρώνουν τις εργασίες τους εγκαίρως. Ένα τέτοιο σύστημα θα μπορούσε να περιλαμβάνει την εμφάνιση γραφικών στοιχείων όπως κομφετί ή εικονικά μετάλλια, ή την ανταμοιβή πόντων, συμβάλλοντας στη δημιουργία κινήτρων. Η εισαγωγή αυτού του συστήματος έχει σκοπό να ενισχύσει τη δέσμευση και την παραγωγικότητα των χρηστών, προσφέροντας έναν πιο διαδραστικό και ελκυστικό τρόπο διαχείρισης εργασιών. Με την ενσωμάτωση χαρακτηριστικών όπως τα προαναφερθέντα, τέτοιες εφαρμογές μπορούν να βελτιώσουν ουσιαστικά την παραγωγικότητα και την αποτελεσματικότητα, προσφέροντας παράλληλα μια ευχάριστη εμπειρία χρήσης.

## Κεφάλαιο 2

# Low-Code

“*The future of coding is no coding at all.*”

—Chris Wanstrath, Co-Founder, Github

Σε αυτό το κεφάλαιο, 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...

### 2.1 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.*” [31]

Για να κατανοήσουμε καλύτερα την έννοια του χαμηλού κώδικα, μπορούμε να εξετάσουμε 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 συγχριτικά 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 ταυτίζεται με την εξέλιξη του χαμηλού κώδικα.

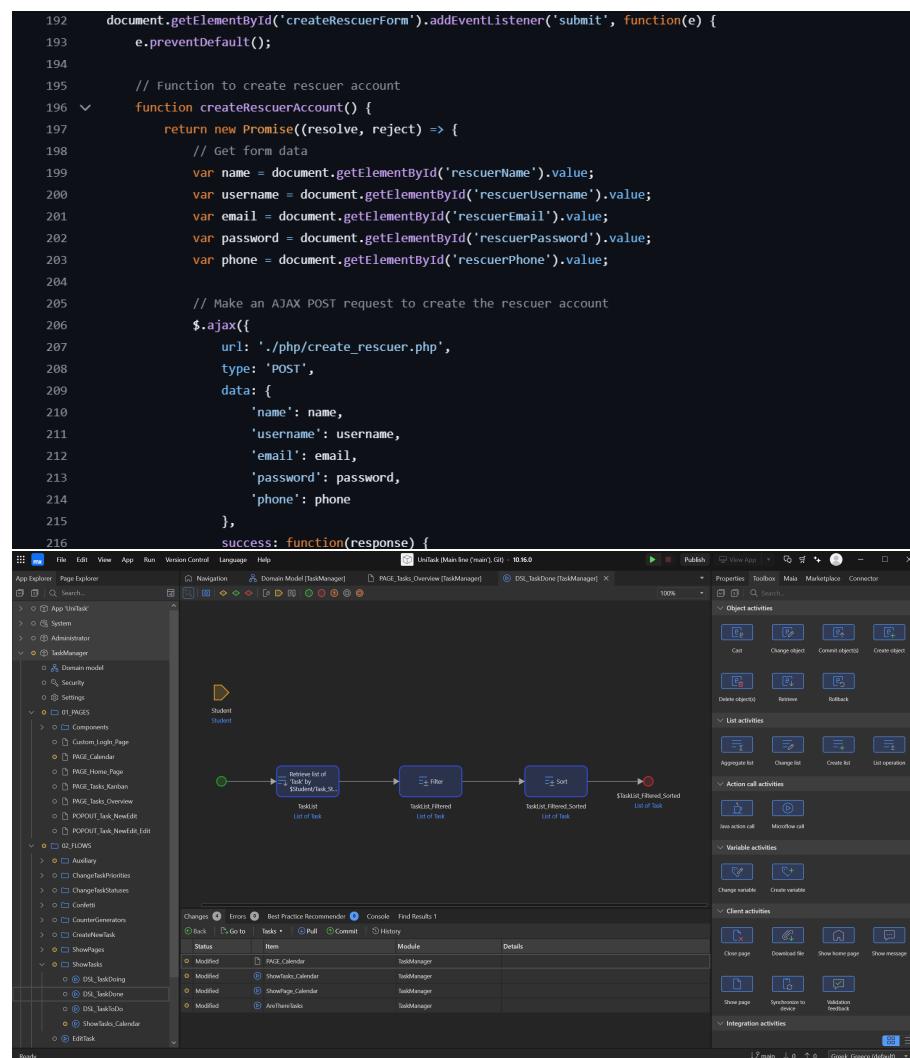
Επομένως, ο χαμηλός κώδικας, αν και ονομάστηκε πρόσφατα ως όρος, η έννοιά του δεν είναι καθόλου καινούρια, και είναι η άμεση εξέλιξη του υψηλού επιπέδου γλωσσών προγραμματισμού (4GLs) των τελευταίων δεκαετιών. Το υψηλότερο προγραμματιστικό επίπεδο με τις αφαιρέσεις που διαθέτει, επιτρέπει ευκολότερη 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<sup>1</sup> να τροποποιήσουν εφαρμογές ή και να φτιάξουν εξ’ ο-

<sup>1</sup> 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 (περιγράφονται αναλυτικότερα στο 2.1.3).

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 που 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.

### 2.1.1 Οπτικός προγραμματισμός (visual programming)

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



Σχήμα 2.1: Παραδοσιακός κώδικας και το γραφικό περιβάλλον του Mendix.

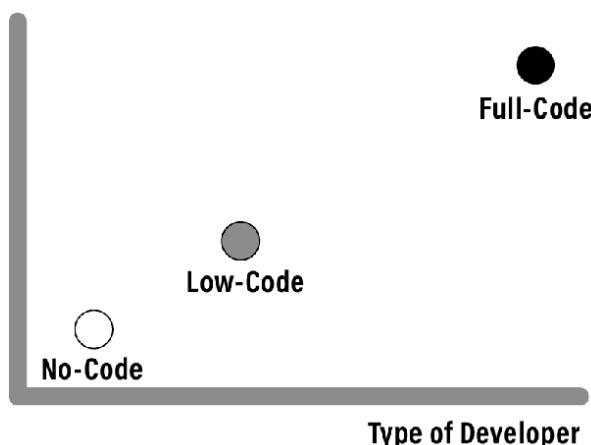
Στο γραφικό περιβάλλον ο προγραμματισμός γίνεται σε ένα διάγραμμα ροής με drag-and-drop επαναχρησιμοποιούμενα στοιχεία (εν προκειμένω στο σχήμα 2.1, ένα παράδειγμα επαναχρησιμοποιούμενων στοιχείων είναι τα μπλε τετράγωνα στο διάγραμμα ροής και δεξιά της οθόνης), σε αντίθεση με τον παραδοσιακό κώδικα 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. [12] 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.

### 2.1.2 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 πληκτρολόγιο τους. Όλη η αλληλεπίδραση 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.



Σχήμα 2.2: Σύγκριση ανάμεσα στην προγραμματιστική εμπειρία των χρηστών και την προγραμματιστική προσέγγιση που χρησιμοποιούν [26]

### 2.1.3 Η έννοια του προγραμματιστή πολίτη (citizen developer)

Ο προγραμματιστής πολίτης είναι ένας χρήστης που αναπτύσσει εφαρμογές σε συνεργασία με τους προγραμματιστές, χωρίς να είναι απαραίτητο να διαθέτει προγραμματιστικές γνώσεις. Η έλλειψη προηγούμενων γνώσεων και εμπειρίας στον προγραμματισμό δεν τον εμποδίζει από το να συνεισφέρει στην ανάπτυξη με ισότιμο τρόπο όπως οι παραδοσιακοί προγραμματιστές.

Μια περίληψη των βασικών διαφορών που παρατηρούνται ανάμεσα στους προγραμματιστές πολίτες και τους μηχανικούς λογισμικού συνοφίζεται στον παρακάτω πίνακα [26]:

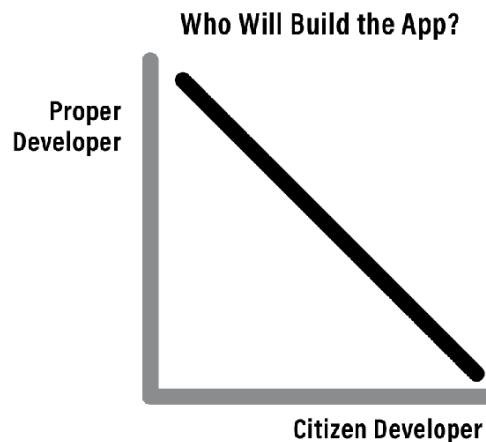
Χαρακτηριστικό	Παραδοσιακός μηχανικός λογισμικού	Προγραμματιστής πολίτης
Γνωστικό υπόβαθρο	Μηχανική λογισμικού ή Επιστήμη των Υπολογιστών	Ποικίλει
Θέση στην επιχείρηση	Τεχνολογία πληροφοριών (IT), DevOps	Μάρκετινγκ, πωλήσεις, HR, λογιστικά
Γνώσεις που αφορούν την επιχείρηση	Λίγες	Πολλές

Ο ρόλος των προγραμματιστών πολιτών προβλέπεται να εκτιναχθεί στα επόμενα χρόνια, όχι τόσο ως μια καινούρια θέση εργασίας αλλά παραπάνω ως ένα επιπλέον χαρακτηριστικό στις υπάρχουσες θέσεις. Η Gartner<sup>2</sup> αναφέρει πως “ως το 2023, ο αριθμός των ενεργών προγραμματιστών πολιτών σε μεγάλες επιχειρήσεις θα είναι τουλάχιστον τετραπλάσιος από τον αριθμό των παραδοσιακών προγραμματιστών” [18] ενώ η Microsoft αναφέρει πως “οι προγραμματιστές πολίτες θα αναπτύξουν 450 εκατομμύρια εφαρμογές το 2025 χρησιμοποιώντας χαμηλό ή καθόλου κώδικα” [22].

#### 2.1.3.1 Διαφορά με τους παραδοσιακούς προγραμματιστές

Οι παραδοσιακοί προγραμματιστές, παραδόξως, δεν αντιμετωπίζουν τους προγραμματιστές πολίτες με καχυποφία αλλά με ενθουσιασμό. Ο λόγος είναι απλός και συνοφίζεται στο σχήμα 2.3. Όσο εντυπωσιακές και αν είναι οι ικανότητες των προγραμματιστών πολιτών, παραμένει ανέφικτο για αυτούς να σχεδιάσουν πολύπλοκα συστήματα για επιχειρήσεις. Οι παραδοσιακοί προγραμματιστές και μηχανικοί λογισμικού είναι αυτοί που μπορούν, και με το βάρος των αδιάφορων λεπτομερειών και του troubleshooting να έχει φύγει από πάνω τους και να έχει μετακινηθεί στους προγραμματιστές πολίτες, μπορούν και εστιάζουν περισσότερο στον σχεδιασμό και την αρχιτεκτονική των εφαρμογών.

<sup>2</sup>Η Gartner είναι μια διεθνώς αναγνωρισμένη εταιρία ερευνών και συμβουλευτικών υπηρεσιών με κύριους πελάτες της οργανισμούς στους τομείς της τεχνολογίας, της επιχειρηματικής στρατηγικής και της καινοτομίας. Πρόκειται για έναν από τους πιο έγκυρους φορείς για αναλύσεις σε τάσεις στο τομέα της πληροφορικής και της τεχνολογίας.



Σχήμα 2.3: Ποιος θα φτιάξει την εφαρμογή; [26]

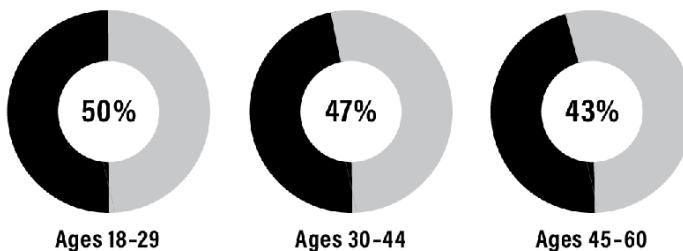
#### 2.1.3.2 Χαρακτηριστικά των προγραμματιστών πολιτών

Σε έρευνα [17] που πραγματοποίησε το QuickBase ανάμεσα σε 148 αυτοαποκαλούμενους προγραμματιστές πολίτες, το 97% κατείχε βασικές ικανότητες επεξεργασίας κειμένου και υπολογιστικών φύλλων, το 36% κατείχε βασικές γνώσεις σε front-end web development (HTML, CSS και Javascript), ενώ το 8% είχε επαφή με Java, .NET, Python, Ruby, PHP και άλλες γλώσσες προγραμματισμού.



Σχήμα 2.4: Επαγγελματικό υπόβαθρο προγραμματιστών πολιτών. [26]

Έχει ενδιαφέρον ότι το 72% των συμμετεχόντων δεν είναι προγραμματιστές αλλά έχουν διαφορετικό επαγγελματικό αντικείμενο. Από την άλλη το 28% των συμμετεχόντων οι οποίοι είναι προγραμματιστές φανερώνει μια τάση ότι και οι ίδιοι οι προγραμματιστές χρησιμοποιούν εργαλεία χαμηλού κώδικα.



Σχήμα 2.5: Ηλικιακή κατανομή προγραμματιστών πολιτών. [29]

Γιάρχει τάση οι νεότεροι ηλικιακά να έχουν μεγαλύτερες πιθανότητες να είναι προγραμματιστές πολίτες.

## 2.2 Πώς φτάσαμε στον χαμηλό κώδικα

Θα επεκταθούμε στα επόμενα κεφάλαια στα οφέλη του low-code και στα χαρακτηριστικά των πλατφορμών ανάπτυξης λογισμικού σε low-code. Πριν προχωρήσουμε, αξίζει να εξετάσουμε πρώτα τις τεχνολογικές εξελίξεις που μας οδήγησαν σήμερα σε αυτές τις πλατφόρμες.

Η μηχανική λογισμικού<sup>3</sup> 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 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 **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)<sup>4</sup> 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)<sup>5</sup>, μοντέλα οντοτήτων-συσχετίσεων (ER –

<sup>3</sup>Θα ορίζαμε τη μηχανική λογισμικού ως μια πειθαρχημένη και αυστηρή εφαρμογή μεθόδων, διαδικασιών και εργαλείων στη διαχείριση και ανάπτυξη υπολογιστικών συστημάτων. Πρόκειται για ένα εννοιολογικό πλαίσιο που περιγράφει τη διαχείριση συστημάτων.

<sup>4</sup>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.

<sup>5</sup>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).

entity-relationship models)<sup>6</sup>, δημιουργώντας μια συστηματική περιγραφή του λογικού και φυσικού μέρους του πληροφοριακού συστήματος που ανέπτυσσαν.

Με την περατέρω 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)<sup>7</sup>, τα Computer-Aided Software Engineering (CASE) περιβάλλοντα, β) η Ταχεία Ανάπτυξη Εφαρμογών (Rapid Application Development – RAD)<sup>8</sup> τη δεκαετία του 1990, γ) ο Προγραμματισμός τελικού χρήστη (End-User Development – EUD)<sup>9</sup> τη δεκαετία του 2000, και οι αρχιτεκτονικές που βασίζονται σε μοντέλα (model-driven architecture – MDA) τις τελευταίες δύο δεκαετίες. [4, 5, 23]

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

### 2.2.1 Computer-Aided Software Engineering (CASE)

Τα Computer-Aided Software Engineering (CASE – Μηχανική Λογισμικού Ύποβοηθούμενη 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 επικεντρώνονται 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

<sup>6</sup>Περιγράφει ένα σύνολο αντικειμένων (οντότητες) και τις σχέσεις μεταξύ αυτών των αντικειμένων.

<sup>7</sup>Σε αντίθεση με τις γλώσσες προγραμματισμού τέταρτης γενιάς (C, Java κτλ) που σε αντίθεση με της 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.

<sup>8</sup>Πρόκειται για μια μεθοδολογία που δίνει βάση στη δημιουργία πρωτότυπων. 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).

<sup>9</sup>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.

τον δυσκολεύουν, αυξάνοντας την παραγωγικότητα και την ποιότητα της δουλειάς του. Τα εργαλεία που περιλαμβάνουν ποικίλουν: 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 και 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.)
- 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, 12, 16]

### 2.2.2 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 απαραίτητα 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) [15], βασίζεται σε ένα σύνολο προτύπων για τον ορισμόν 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. [2, 23, 25]

### 2.2.3 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. Το Microsoft Access ήταν ένα από τα πρώτα λογισμικά που πρόσφεραν γραφικό περιβάλλον εργασίας. Οι χρήστες σέρνοντας και αφήνοντας στοιχεία μπορούσαν να κατασκευάζουν βάσεις δεδομένων, φόρμες κ.α. χωρίς την ανάγκη της SQL.



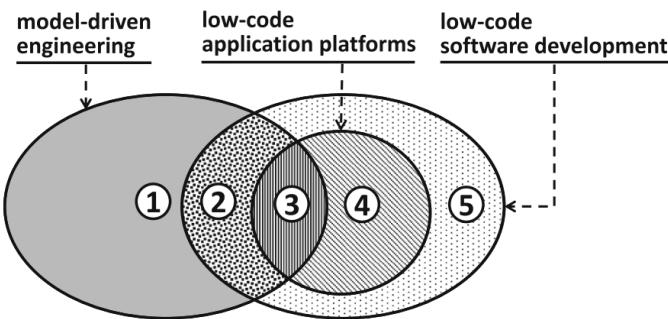
Σχήμα 2.6: Το γραφικό περιβάλλον του Microsoft FrontPage

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. [26]

## 2.3 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 αφαιρεσης 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)<sup>10</sup>.[1]



Σχήμα 2.7: Σύγκριση μεταξύ της μηχανικής που βασίζεται σε μοντέλα και των Low-Code πλατφορμών. [23]

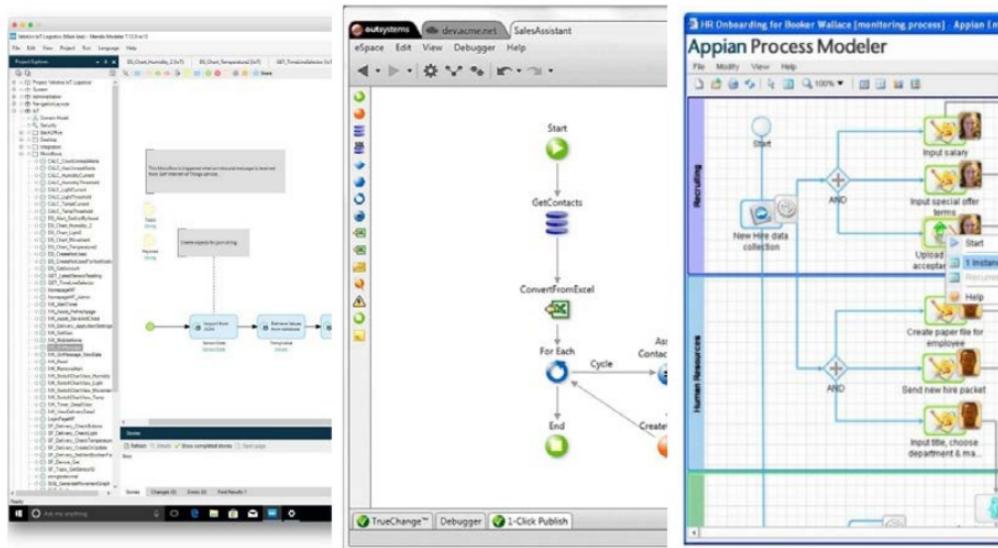
Στην περιοχή 1 χρησιμοποιούνται μοντέλα χωρίς να γίνονται προσπάθειες για μείωση του κώδικα, σε αντίθεση με τις περιοχές 2 και 3 που στοχεύουν στην μείωση του κώδικα. Από την άλλη, οι πλατφόρμες ανάπτυξης κώδικα σε low-code δεν βασίζονται πάντα σε μοντέλα (περιοχές 4 και 5), αλλά για παράδειγμα χρησιμοποιούν δεδομένα σε σχεσιακές βάσεις ή σε XML αρχεία. Οι διαφορές μεταξύ low-code application platforms και low-code software development έγκειται 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, στόχος είναι η αποδοτική ανάπτυξη λογισμικού με τη λιγότερη δυνατή προσπάθεια και με μειωμένο 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, 24]

<sup>10</sup> 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 2.8: LCDPs από αριστερά προς τα δεξιά: Mendix, OutSystems, Appian [11]

### 2.3.1 Χαρακτηριστικά των LCDP

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

- **Γραφικό περιβάλλον χρήστη:** περιέχονται εργαλεία και γραφικά στοιχεία, drag-and-drop χαρακτηριστικά, μηχανές αποφάσεων για τη μοντελοποίηση σύνθετης λογικής, κατασκευαστές φορμών (form builder)
- **Φορητότητα και συνεργασία:** επιτρέπουν στους χρήστες να εργάζονται από οποιαδήποτε τοποθεσία και οποιαδήποτε συσκευή ή λειτουργικό σύστημα.
- **Επεκτασιμότητα:** επαναχρησιμοποίηση προκατασκευασμένων στοιχείων, ύπαρξη marketplace και εύκολη ενσωμάτωση modules τρίτων.
- **Έλεγχος έκδοσης:** DevOps χαρακτηριστικά (για παράδειγμα σύστημα για version control όπως Git) και δυνατότητα για άμεσο deploy της εφαρμογής για το κοινό.

# 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] *BUS402: History of Project Management* | Saylor Academy — learn.saylor.org. <https://learn.saylor.org/mod/page/view.php?id=65663>. [Accessed 26-10-2024].
- [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] Ryoko Fukuzawa, Hideo Joho και Tetsuya Maeshiro. «Practice and experience of task management of university students: Case of University of Tsukuba, Japan». Στο: *Education for Information* 31 (3 Ιούλ. 2015), σσ. 109–124. issn: 01678329. doi: 10.3233/EFI-150953.
- [7] Jack Goody. «Memory in Oral Tradition». Στο: Cambridge University Press, Μαρ. 2013, σσ. 73–94. doi: 10.1017/cbo9781139171137.005.
- [8] *Guide to the Project Management Body of Knowledge*. Project Management Institute, 2021. isbn: 1628256648.
- [9] *Hoover Dam – the Greatest Project in Times of the Great Depression. What Can Be Done to Achieve Success? - Strefa PMI* — strefapmi.pl. <https://strefapmi.pl/strefa-studenta/hoover-dam-the-greatest-project-in-times-of-the-great-depression/>. [Accessed 30-10-2024].
- [10] *Hoover Dam | Description, Location, Construction, Facts, History, & Pictures* | Britannica — britannica.com. <https://www.britannica.com/topic/Hoover-Dam>. [Accessed 25-12-2024].
- [11] Bryan Kasam, Imran McMullen και Micah Kenneweg. *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.

- [12] D. L. Kuhn. *Selecting and Effectively Using a Computer-Aided Software Engineering Tool*. 1989.
- [13] Benne Lientz και Kathryn Rea. *Project Management for the 21st Century*. 2007.
- [14] *Manage Your Team's Projects From Anywhere | Trello* — trello.com. <https://trello.com/>. [Accessed 16-10-2024].
- [15] *MDA FAQ* | Object Management Group — omg.org. [https://www.omg.org/mda/faq\\_mda.htm](https://www.omg.org/mda/faq_mda.htm). [Accessed 08-11-2024].
- [16] G. Premkumar και Michael Potter. *Adoption of Computer Aided Software Engineering (CASE) Technology: An Innovation Adoption Perspective*.
- [17] QuickBase. *The State Of Citizen Development Report – September 2015*. [https://cdn2.hubspot.net/hubfs/172645/QuickBase\\_Citizen\\_Developer\\_Report.pdf](https://cdn2.hubspot.net/hubfs/172645/QuickBase_Citizen_Developer_Report.pdf). [Accessed 25-11-2024].
- [18] Quickbase. *Gartner® Report: Future of Work Trends* — quickbase.com. <https://www.quickbase.com/gartner-future-of-work>. [Accessed 25-11-2024].
- [19] Quipu - Wikipedia — en.wikipedia.org. <https://en.wikipedia.org/wiki/Quipu>. [Accessed 22-10-2024].
- [20] Thomas Q Reefe. *Lukasa: A Luba Memory Device*. doi: doi:10.2307/3335144.
- [21] E. G. Richards. *Mapping time: The calendar and its history*. Oxford University Press, 2000.
- [22] Eric Rosenbaum. *Next frontier in Microsoft, Google, Amazon cloud battle is over a world without code* — cnbc.com. <https://www.cnbc.com/2020/04/01/new-microsoft-google-amazon-cloud-battle-over-world-without-code.html>. [Accessed 25-11-2024].
- [23] 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.
- [24] 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.
- [25] Matthias Book Sami Beydeda και Volker Gruhn. *Model-Driven Software Development*.
- [26] Phil Simon. *Low-Code/No-Code: Citizen Developers and the Surprising Future of Business Applications*. 2022.
- [27] O'Reilly Editorial Team. «Low-Code and the Democratization of Programming». Στο: *O'Reilly Media* (2021).

- [28] *Todoist | A To-Do List to Organize Your Work & Life* — [todoist.com.](https://todoist.com/) <https://todoist.com/>. [Accessed 16-10-2024].
- [29] TrackVia. *The next generation worker: The Citizen Developer – Insights on the behaviors and characteristics of an emerging class of technology users within the enterprise*. [https://lumenmarketing.com/wp-content/uploads/2017/11/TV\\_Citizen\\_Dev.pdf](https://lumenmarketing.com/wp-content/uploads/2017/11/TV_Citizen_Dev.pdf). [Accessed 25-11-2024]. 2014.
- [30] Julia Castillo Trujillo. *Designing A Time Management App For And With Informatics Students*. 2020.
- [31] *What Is Low-Code? | IBM* — [ibm.com](https://www.ibm.com/topics/low-code). <https://www.ibm.com/topics/low-code>. [Accessed 11-10-2024].
- [32] *WinWorld: Welcome* — [winworldpc.com](https://winworldpc.com/home). <https://winworldpc.com/home>. [Accessed 31-10-2024].
- [33] *Your connected workspace for wiki, docs & projects | Notion* — [notion.so](https://www.notion.so/). <https://www.notion.so/>. [Accessed 16-10-2024].
- [34] Μιχαήλ Ξένος. *Ποιότητα Λογισμικού*. GOTSIS, 2021. ISBN: 9786185560102.