



ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ
ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΑ ΠΛΗΡΟΦΟΡΙΑΚΑ
ΣΥΣΤΗΜΑΤΑ

Διπλωματική Εργασία

Προσέγγιση της σχεδίασης λογισμικού με τη χρήση διαγραμμάτων
της ενοποιημένης γλώσσας μοντελοποίησης UML

Διονύσιος Αντίοχος

Επιβλέπων καθηγητής: Αθανάσιος Σύψας

Πάτρα, Ιούνιος 2025

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του φοιτητή («συγγραφέας/δημιουργός») που την εκπόνησε. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης ο συγγραφέας/δημιουργός εκχωρεί στο ΕΑΠ, μη αποκλειστική άδεια χρήσης του δικαιώματος αναπαραγωγής, προσαρμογής, δημόσιου δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσής τους διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος και για όλο το χρόνο διάρκειας των δικαιωμάτων πνευματικής ιδιοκτησίας. Η ανοικτή πρόσβαση στο πλήρες κείμενο για μελέτη και ανάγνωση δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, αποθήκευση, πώληση, εμπορική χρήση, μετάδοση, διανομή, έκδοση, εκτέλεση, «μεταφόρτωση» (downloading), «ανάρτηση» (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού. Ο συγγραφέας/δημιουργός διατηρεί το σύνολο των ηθικών και περιουσιακών του δικαιωμάτων.



Προσέγγιση της σχεδίασης λογισμικού με τη χρήση διαγραμμάτων της ενοποιημένης γλώσσας μοντελοποίησης UML

Διονύσιος Αντίοχος

Επιτροπή Επίβλεψης Διπλωματικής Εργασίας

Επιβλέπων Καθηγητής:

Αθανάσιος Σύψας

Συν-Επιβλέπων Καθηγητής:

Ιωάννης Μοσχολιός

Πάτρα, Ιούνιος 2025

«Στην οικογένειά μου»

Περίληψη

Σκοπός της διπλωματικής εργασίας είναι η θεωρητική διερεύνηση του ρόλου και της συμβολής των διαγραμμάτων της Ενοποιημένης Γλώσσας Μοντελοποίησης (Unified Modeling Language – UML) στη διαδικασία σχεδίασης συστημάτων λογισμικού. Επιδιώκεται η κατανόηση του τρόπου με τον οποίο η UML υποστηρίζει τη μετάβαση από την αφηρημένη σύλληψη ενός συστήματος στη δομημένη και κατανοητή απεικόνισή του μέσω διαγραμμάτων, τα οποία διευκολύνουν την επικοινωνία μεταξύ των αναλυτών, προγραμματιστών και λοιπών εμπλεκομένων.

Η UML, λειτουργεί ως εργαλείο αναπαράστασης για την μοντελοποίηση της αρχιτεκτονικής, της δομής, της συμπεριφοράς και των αλληλεπιδράσεων συστημάτων λογισμικού. Η UML, ως τυποποιημένη γλώσσα οπτικής μοντελοποίησης, επιτρέπει την περιγραφή της αρχιτεκτονικής, της δομής, της συμπεριφοράς και των αλληλεπιδράσεων ενός συστήματος, προσφέροντας ένα κοινό πλαίσιο επικοινωνίας μεταξύ των μελών της ομάδας ανάπτυξης.

Η εργασία εστιάζει στη θεωρητική ανάλυση των διαγραμμάτων UML (όπως Use Case, Class, Sequence, Activity, State και Component Diagrams), στη μελέτη των μεταξύ τους σχέσεων και της συμβολής τους στη συνολική διαδικασία σχεδίασης καθώς και στην αποτύπωση των πλεονεκτημάτων και περιορισμών που προκύπτουν από τη χρήση της UML σε θεωρητικό επίπεδο.

Επιπλέον, εξετάζονται παραδείγματα εφαρμογών των διαγραμμάτων UML σε σενάρια σχεδίασης λογισμικού, με στόχο την ανάδειξη της θεωρητικής τους χρησιμότητας στη φάση ανάλυσης και μοντελοποίησης. Μέσω αυτής της μελέτης επιχειρείται η ανάδειξη της UML ως βασικού γνωστικού εργαλείου στη μεθοδολογία σχεδίασης λογισμικού, συμβάλλοντας στη θεωρητική εμβάθυνση και κατανόηση των διαδικασιών μοντελοποίησης συστημάτων.

Λέξεις – Κλειδιά

Ενοποιημένη Γλώσσα Μοντελοποίησης(UML), διάγραμμα UML, μοντέλο, σχεδίαση λογισμικού

Approaching software design using Unified Modeling Language diagrams

Dionissios Antiochos

Abstract

The purpose of this thesis is to theoretically investigate the role and contribution of Unified Modeling Language (UML) diagrams in the software systems design process. The aim is to understand how UML supports the transition from the abstract conception of a system to its structured and understandable representation through diagrams, which facilitate communication between analysts, programmers and other stakeholders.

UML functions as a representation tool for modeling the architecture, structure, behavior and interactions of software systems. UML, as a standardized visual modeling language, allows the description of the architecture, structure, behavior and interactions of a system, offering a common communication framework among the members of the development team.

The work focuses on the theoretical analysis of UML diagrams (such as Use Case, Class, Sequence, Activity, State and Component Diagrams), on the study of their relationships and their contribution to the overall design process as well as on the depiction of the advantages and limitations that arise from the use of UML at a theoretical level.

In addition, examples of applications of UML diagrams in software design scenarios are examined, with the aim of highlighting their theoretical usefulness in the analysis and modeling phase. Through this study, an attempt is made to highlight UML as a basic cognitive tool in software design methodology, contributing to the theoretical deepening and understanding of systems modeling processes.

Keywords

Unified Modeling Language(UML), UML diagram, model, software design

Περιεχόμενα

Περίληψη.....	v
Abstract	vi
Περιεχόμενα	vii
Κατάλογος Εικόνων / Σχημάτων	x
Κατάλογος Πινάκων	xi
Συντομογραφίες & Ακρωνύμια.....	xii
1. Εισαγωγή.....	1
1.1 Αντικείμενο της εργασίας	1
1.2 Σκοπός και στόχοι.....	1
1.3 Συνεισφορά της εργασίας	2
1.4 Μεθοδολογία και προσέγγιση.....	2
1.5 Διάρθρωση κεφαλαίων	2
2. Η σχεδίαση λογισμικού ως διαδικασία	4
2.1 Ο ρόλος της ανάλυσης και της μοντελοποίησης	4
2.1.1 Η ανάλυση	4
2.1.2 Η μοντελοποίηση.....	6
2.2 Από την αφηρημένη σύλληψη σε σχεδιαστικά μοντέλα	7
2.3 Προσεγγίσεις στη σχεδίαση συστημάτων λογισμικού	8
2.4 Προβλήματα επικοινωνίας χωρίς τυποποιημένα μοντέλα.....	11
3. Η Ενοποιημένη Γλώσσα Μοντελοποίησης (UML)	13
3.1 Ιστορική εξέλιξη και τυποποίηση της UML.....	13
3.1.1 Το παρελθόν	13
3.1.2 Το παρόν.....	15
3.1.3 Το μέλλον	16
3.2 Σκοπός και φιλοσοφία της UML	18
3.3 Πλεονεκτήματα και περιορισμοί της UML στη θεωρητική σχεδίαση	19
3.4 UML και σύγχρονες τάσεις στη σχεδίαση λογισμικού	20
4. Τα Διαγράμματα της UML: θεωρία και χρήση.....	23
4.1 Διαγράμματα Περιπτώσεων Χρήσης.....	24
4.1.1 Βασικά στοιχεία των διαγραμμάτων περιπτώσεων χρήσης.....	24
4.1.2 Συνεισφορά στη φάση ανάλυσης απαιτήσεων	26
4.1.3 Σκοπός και οφέλη των διαγραμμάτων περιπτώσεων χρήσης	27
4.2 Διαγράμματα Κλάσεων.....	28
4.2.1 Κλάση.....	29
4.2.2 Συσχέτιση κλάσεων.....	30
4.2.3 Κλάση συσχέτισης.....	31
4.2.4 Αυτοσυσχέτιση.....	31
4.2.5 Σχέση εξάρτησης κλάσεων.....	32
4.2.6 Σχέση γενίκευσης κλάσεων.....	32
4.2.7 Σχέση συνάθροισης κλάσεων.....	33
4.2.8 Σχέση σύνθεσης κλάσεων	34
4.3 Διαγράμματα Ακολουθίας	34
4.3.1 Μοντελοποίηση αλληλεπιδράσεων	34
4.3.2 Σενάριο εκτέλεσης.....	36
4.4 Διαγράμματα Δραστηριοτήτων	37

4.4.1 Ροή εργασιών	38
4.4.2 Απόφαση και διακλάδωση	41
4.5 Διαγράμματα Καταστάσεων	43
4.6 Διαγράμματα Συστατικών Στοιχείων	46
4.7 Συγκριτική θεώρηση διαγραμμάτων.....	47
5. Σχέσεις και συνέργειες μεταξύ των διαγραμμάτων UML	50
5.1 Η λογική ροή από τις απαιτήσεις στη σχεδίαση	50
5.2 Μετάβαση από Use Case σε Class και Sequence Diagrams	53
5.2.1 Για τα διαγράμματα κλάσεων.....	53
5.2.2 Για τα διαγράμματα ακολουθίας	54
5.3 Συνδυαστική χρήση των Activity και State Diagrams	54
5.4 Πώς τα Component Diagrams συνθέτουν την αρχιτεκτονική	56
5.5 Καλές πρακτικές στην ενοποιημένη χρήση UML διαγραμμάτων.....	60
5.6 Παράδειγμα μετάβασης μεταξύ διαγραμμάτων UML.....	64
6. Συμβολή των UML διαγραμμάτων στη σχεδίαση λογισμικού	70
6.1 Η UML ως εργαλείο θεωρητικής σκέψης	70
6.2 Ενίσχυση της κατανόησης σύνθετων συστημάτων	72
6.3 Υποστήριξη επικοινωνίας και τεκμηρίωσης.....	73
6.4 Θεωρητικά πλεονεκτήματα στη φάση ανάλυσης & αρχιτεκτονικής.....	75
6.5 Περιορισμοί και κριτική της UML από θεωρητική σκοπιά	77
6.6 Χρήση της UML στην πράξη	80
7. Παραδείγματα εφαρμογής UML στη σχεδίαση συστημάτων.....	82
7.1 Διάγραμμα περιπτώσεων χρήσης	82
7.2 Διάγραμμα κλάσεων	83
7.3 Διάγραμμα ακολουθίας.....	84
7.4 Διάγραμμα δραστηριοτήτων.....	85
7.5 Διάγραμμα καταστάσεων	86
7.6 Διάγραμμα συστατικών	87
7.7 Συμπέρασμα.....	87
8. Αξιολόγηση της συμβολής των UML διαγραμμάτων	89
8.1 Κριτήρια αξιολόγησης μοντελοποίησης.....	89
8.1.1 Φυσική Ποιότητα	90
8.1.2 Εμπειρική Ποιότητα	91
8.1.3 Συντακτική Ποιότητα	91
8.1.4 Σημασιολογική Ποιότητα.....	92
8.1.5 Πραγματολογική Ποιότητα	92
8.1.6 Κοινωνική Ποιότητα	93
8.1.7 Δεοντολογική Ποιότητα	94
8.1.8 Εφαρμογή κριτηρίων αξιολόγησης	95
8.2 Πώς τα διαγράμματα βελτιώνουν την κατανόηση του συστήματος.....	97
8.3 Αποτελεσματικότητα στη θεωρητική σχεδίαση λογισμικού	98
8.4 Περιορισμοί στη χρήση των UML διαγραμμάτων	100
8.5 Σύγκριση με άλλες προσεγγίσεις μοντελοποίησης.....	101
8.5.1 Δομημένη μοντελοποίηση	101
8.5.2 Ευέλικτες μέθοδοι	102
8.5.3 BPMN.....	103
8.5.4 SysML και DSL.....	104
9. Συζήτηση – Συμπεράσματα	105

9.1 Συνοπτική σύνθεση ευρημάτων.....	105
9.2 Θεωρητική συμβολή της UML στη σχεδίαση συστημάτων.....	107
9.3 Σημεία προς περαιτέρω διερεύνηση	108
Βιβλιογραφία.....	110

Κατάλογος Εικόνων / Σχημάτων

Σχήμα 1: Διάγραμμα δραστηριοτήτων για την επεξεργασία παραγγελίας αγοράς προϊόντος.....	19
Σχήμα 2: Ταξινόμηση των βασικών UML διαγραμμάτων	24
Σχήμα 3: Διάγραμμα περιπτώσεων χρήσης για σύστημα on line αγορών.....	27
Σχήμα 4: Διάγραμμα κλάσεων.....	29
Σχήμα 5: Βασικός συμβολισμός κλάσης.....	30
Σχήμα 6: Συσχέτιση δύο κλάσεων με εμφάνιση πολλαπλότητας.....	30
Σχήμα 7: Κλάση συσχέτισης.....	31
Σχήμα 8: Αυτοσυσχέτιση.....	32
Σχήμα 9: Σχέση εξάρτησης κλάσεων.....	32
Σχήμα 10: Σχέση γενίκευσης κλάσεων.....	33
Σχήμα 11: Σχέση συνάθροισης κλάσεων.....	33
Σχήμα 12: Σχέση σύνθεσης κλάσεων.....	34
Σχήμα 13: Διάγραμμα ακολουθίας (επεξεργασία μέσω Visual Paradigm).....	36
Σχήμα 14: Διάγραμμα δραστηριοτήτων για εικονικό εργαστήριο.....	40
Σχήμα 15: Διάγραμμα δραστηριοτήτων με μικροσκόπιο για εικονικό εργαστήριο.....	41
Σχήμα 16: Διάγραμμα μηχανής καταστάσεων.....	44
Σχήμα 17: Μετάβαση από απαιτήσεις σε αρχιτεκτονικό σχεδιασμό.....	52
Σχήμα 18: Διαφορετικοί τρόποι απεικόνισης διαμερίσματος ονόματος συστατικού στοιχείου.....	56
Σχήμα 19: Πρόσθετο διαμέρισμα ονόματος συστατικού στοιχείου που δείχνει ποιες διεπαφές παρέχονται/απαιτούνται.....	57
Σχήμα 20: Εναλλακτική προσέγγιση εμφάνισης συστατικού στοιχείου που δείχνει ποιες διεπαφές παρέχονται/απαιτούνται.....	57
Σχήμα 21: Απεικόνιση εξαρτήσεων στοιχείου.....	58
Σχήμα 22: Απεικόνιση στοιχείου υποσυστήματος.....	59
Σχήμα 23: Απεικόνιση δομής στοιχείου που αποτελείται από άλλα στοιχεία.....	59
Σχήμα 24: Ροή διαδικασίας μοντελοποίησης της ανάλυσης και σχεδίασης λογισμικού με χρήση UML.....	64
Σχήμα 25: Διάγραμμα περιπτώσεων χρήσης.....	65
Σχήμα 26: Διάγραμμα κλάσεων.....	66
Σχήμα 27: Διάγραμμα ακολουθίας.....	67
Σχήμα 28: Διάγραμμα δραστηριοτήτων.....	68
Σχήμα 29: Διάγραμμα καταστάσεων.....	69
Σχήμα 30: Διάγραμμα περιπτώσεων χρήσης.....	82
Σχήμα 31: Διάγραμμα κλάσεων.....	83
Σχήμα 32: Διάγραμμα ακολουθίας.....	84
Σχήμα 33: Διάγραμμα δραστηριοτήτων.....	85
Σχήμα 34: Διάγραμμα καταστάσεων.....	86
Σχήμα 35: Απλό διάγραμμα συστατικών στοιχείων.....	87
Σχήμα 36: Πλαίσιο SEQUAL για την ποιότητα μοντέλων.....	90
Σχήμα 37: Διάγραμμα κλάσεων για σύστημα παραγγελιών.....	96

Κατάλογος Πινάκων

Πίνακας 4.1: Συμβολισμός βασικών στοιχείων διαγραμμάτων περιπτώσεων χρήσης.....	26
Πίνακας 4.2: Συμβολισμός βασικών στοιχείων διαγραμμάτων δραστηριοτήτων.	38
Πίνακας 4.3: Συμβολισμός βασικών στοιχείων διαγραμμάτων συμπεριφορικών μηχανών καταστάσεων.	44

Συντομογραφίες & Ακρωνύμια

API	Application Programming Interface
BPMN	Business Process Model and Notation
DSL	Domain Specific Language
IBM	International Business Machines
MVP	Minimum Viable Product
OCL	Object Constraint Language
OMG	Object Management Group
OMT	Object Modeling Technique
OOD	Object Oriented Design
OOPSLA	Object Oriented Programming Systems Languages & Applications
OOSE	Object Oriented Software Engineering
SySML	Systems Modeling Language
UML	Unified Modeling Language

1. Εισαγωγή

1.1 Αντικείμενο της εργασίας

Η Ενοποιημένη Γλώσσα Μοντελοποίησης (UML) είναι μια γενικής χρήσης γλώσσα οπτικής αναπαράστασης που χρησιμοποιείται κατά κύριο λόγο στη μοντελοποίηση συστημάτων λογισμικού που αναπτύσσονται με χρήση αντικειμενοστρεφών τεχνικών.

Η εργασία εστιάζει στη θεωρητική ανάλυση των διαγραμμάτων UML (όπως είναι τα διαγράμματα περιπτώσεων χρήσης, κλάσεων, ακολουθίας, δραστηριοτήτων, μηχανών καταστάσεων και συστατικών), στη μελέτη των μεταξύ τους σχέσεων και της συμβολής τους στη συνολική διαδικασία σχεδίασης καθώς και στην αποτύπωση των πλεονεκτημάτων και περιορισμών που προκύπτουν από τη χρήση της UML σε θεωρητικό επίπεδο. Επιπλέον, παρατίθενται παραδείγματα εφαρμογών των διαγραμμάτων UML σε σενάρια σχεδίασης λογισμικού.

1.2 Σκοπός και στόχοι

Σκοπός της διπλωματικής εργασίας είναι η διερεύνηση σε θεωρητικό επίπεδο του σκοπού και της συμβολής των διαγραμμάτων της UML κατά τη διαδικασία σχεδίασης συστημάτων λογισμικού. Επιδίωξη είναι να γίνει κατανοητός ο τρόπος με τον οποίο η UML, μέσω των διαγραμμάτων της, υποστηρίζει τη μετάβαση από την αφηρημένη σύλληψη ενός συστήματος στη δομημένη και κατανοητή απεικόνισή του.

Μέσω της παράθεσης παραδειγμάτων εφαρμογών των διαγραμμάτων UML σε σενάρια σχεδίασης λογισμικού, στόχος είναι η ανάδειξη της θεωρητικής τους χρησιμότητας στη φάση της ανάλυσης και της μοντελοποίησης. Μέσω αυτής της μελέτης επιχειρείται η ανάδειξη της UML ως βασικού γνωστικού εργαλείου στη μεθοδολογία σχεδίασης λογισμικού.

1.3 Συνεισφορά της εργασίας

Η παρούσα εργασία προσπαθεί να αναδείξει τη σπουδαιότητα και τη συμβολή της UML ως εργαλείο για το σχεδιασμό λογισμικού. Επιδιώκει τη διερεύνηση της σημασιολογικής διάστασης της UML και την ανάλυση του τρόπου με τον οποίο η σαφής σημασιολογική ερμηνεία των μοντέλων συμβάλλει στη βελτίωση της κατανόησης, της συνέπειας και της αυτοματοποίησης κατά τον σχεδιασμό συστημάτων λογισμικού.

Πιο συγκεκριμένα, η συνεισφορά της έγκειται στη συστηματική σύγκριση των διαγραμμάτων της UML, όπως και στην αναπαράσταση των οντοτήτων ενός συστήματος καθώς και των σχέσεών τους με στόχο να κατανοηθεί η δομή και τα σημασιολογικά τους χαρακτηριστικά.

1.4 Μεθοδολογία και προσέγγιση

Η μεθοδολογία έρευνας που χρησιμοποιήθηκε είναι:

1. Βιβλιογραφική έρευνα.
2. Ανάλυση UML μοντέλων.
3. Μελέτη περίπτωσης.

1.5 Διάρθρωση κεφαλαίων

Η παρούσα εργασία δομείται ως εξής:

Το κεφάλαιο 1 αναφέρεται στο αντικείμενο, στους σκοπούς και τους στόχους της εργασίας. Επίσης παρουσιάζεται η συνεισφορά και η μεθοδολογία της εργασίας και ολοκληρώνεται με την διάρθρωση των κεφαλαίων.

Το κεφάλαιο 2 ασχολείται με τη διαδικασία της ανάλυσης και της μοντελοποίησης ενώ γίνεται και αναφορά σε κάποιες μεθοδολογίες σχεδίασης λογισμικού. Ολοκληρώνεται με την ανάλυση των προβλημάτων επικοινωνίας όταν δεν υπάρχουν τυποποιημένα μοντέλα.

Το κεφάλαιο 3 ξεκινά με μια ιστορική αναδρομή στην εξέλιξη και τυποποίηση της UML, έπειτα αναφέρεται στην σκοπιμότητα και τη φιλοσοφία της UML, στα πλεονεκτήματα και τα μειονεκτήματα της όσον αφορά τη θεωρητική σχεδίαση και ολοκληρώνεται με το πώς τοποθετείται η UML στο σύγχρονο περιβάλλον σχεδίασης λογισμικού.

Στο κεφάλαιο 4 παρουσιάζονται βασικά διαγράμματα της UML, με παρουσίαση των βασικών τους στοιχείων και συμβόλων μέσω διαγραμμάτων και πινάκων και καταλήγει με αναφορά στην καταλληλότητα κάθε διαγράμματος για κάποιο συγκεκριμένο σχεδιαστικό ερώτημα.

Το κεφάλαιο 5 παρουσιάζει το πώς συνεργάζονται, σχετίζονται και συνδέονται κάποια συγκεκριμένα είδη διαγραμμάτων και καταλήγει με το πώς μπορεί να γίνει σωστά αυτή η συνδυαστική χρήση αυτών των διαγραμμάτων.

Το κεφάλαιο 6 ασχολείται πιο αναλυτικά με τη συμβολή των διαγραμμάτων στη σχεδίαση λογισμικού. Παρουσιάζει το πώς βοηθά η UML τη θεωρητική σκέψη, την κατανόηση σύνθετων συστημάτων, την επικοινωνία και ολοκληρώνεται με τα πλεονεκτήματα αλλά και τους περιορισμούς της σε θεωρητικό επίπεδο.

Στο κεφάλαιο 7 παρουσιάζονται παραδείγματα εφαρμογής της UML μέσω συγκεκριμένων διαγραμμάτων.

Στο κεφάλαιο 8 αξιολογείται η συμβολή της UML, αναφέρονται οι περιορισμοί της και συγκρίνεται με άλλες προσεγγίσεις μοντελοποίησης.

Στο κεφάλαιο 9 γίνεται μια συνοπτική παράθεση των διαπιστώσεων γενικά για την UML και ειδικότερα για την θεωρητική συμβολή της στη σχεδίαση συστημάτων.

Η εργασία οργανώνεται σε εννέα κεφάλαια. Στο πρώτο κεφάλαιο παρουσιάζεται η εισαγωγή της εργασίας. Στο δεύτερο κεφάλαιο παρουσιάζεται η σχεδίαση λογισμικού ως διαδικασία. Στο τρίτο κεφάλαιο αναλύεται η Ενοποιημένη Γλώσσα Μοντελοποίησης (UML). Στο τέταρτο κεφάλαιο παρουσιάζονται τα βασικά διαγράμματα UML. Στο πέμπτο κεφάλαιο εξετάζονται οι σχέσεις μεταξύ των διαγραμμάτων. Στο έκτο κεφάλαιο αναλύεται η συμβολή των UML διαγραμμάτων στη σχεδίαση λογισμικού. Στο έβδομο κεφάλαιο παρουσιάζονται παραδείγματα εφαρμογής της UML. Στο όγδοο κεφάλαιο γίνεται αξιολόγηση της UML. Στο ένατο κεφάλαιο υπάρχει μια σύνοψη της εργασίας.

2. Η σχεδίαση λογισμικού ως διαδικασία

Η σχεδίαση κατέχει κεντρικό ρόλο στον κύκλο ζωής συστημάτων λογισμικού. Στόχος της είναι ο μετασχηματισμός των απαιτήσεων, οι οποίες έχουν καταγραφεί στο στάδιο της ανάλυσης, σε πλήρεις, λεπτομερείς, ακριβείς και σαφείς προδιαγραφές υλοποίησης του συστήματος (Ghezzi, Jazayeri & Mandrioli, 2003).

2.1 Ο ρόλος της ανάλυσης και της μοντελοποίησης

Η ανάλυση και η μοντελοποίηση στο σχεδιασμό λογισμικού βοηθούν στην αποσαφήνιση των απαιτήσεων των χρηστών, στη βελτίωση της επικοινωνίας μεταξύ των ενδιαφερόμενων μερών και στη δημιουργία μιας δομημένης αναπαράστασης του συστήματος. Αυτές οι διαδικασίες διασφαλίζουν ότι το λογισμικό ανταποκρίνεται στις ανάγκες των χρηστών και μειώνει τα σφάλματα κατά την ανάπτυξη.

2.1.1 Η ανάλυση

Στην φάση της ανάλυσης συλλέγονται οι απαιτήσεις των χρηστών και δημιουργείται σειρά από διαγράμματα με στόχο την αποτύπωση της βασικής λειτουργικότητας και της δομής του νέου συστήματος. Η φάση αυτή απαντά στα ερωτήματα του ποιος θα χρησιμοποιήσει το σύστημα και τι θα κάνει με αυτό. Επίσης γίνεται τεκμηρίωση των απαιτήσεων μέσω του καθορισμού των λειτουργικών και μη λειτουργικών απαιτήσεων.

Οι λειτουργικές απαιτήσεις (functional requirements) περιγράφουν τι πρέπει να κάνει το σύστημα από την άποψη της σκοπιάς του χρήστη. Αναφέρονται και ως δυνατότητες (capabilities) και εκφράζονται με προτάσεις του τύπου «το σύστημα θα πρέπει να κάνει...». Για παράδειγμα: «Το σύστημα θα χρεώνει την πιστωτική κάρτα του πελάτη κατά την αποδοχή της αγοράς» (Λαζαρίνης & Μαυρομμάτης, 2015).

Οι μη λειτουργικές απαιτήσεις (nonfunctional requirements) δεν αφορούν συγκεκριμένη λειτουργικότητα του συστήματος, αλλά γενικότερα χαρακτηριστικά που αυτό πρέπει να έχει και τα οποία, στην ουσία, αποτελούν περιορισμούς του συστήματος που υλοποιείται.

Αναφέρονται και ως απαιτήσεις ποιότητας και εκφράζονται με προτάσεις του τύπου «το σύστημα πρέπει να είναι... ή να έχει...». Για παράδειγμα: «Το σύστημα πρέπει να αποκρίνεται σε χρόνο μικρότερο των 3msec» ή «Το σύστημα πρέπει να λειτουργεί σε περιβάλλον Linux» ή «Το σύστημα θα πρέπει να είναι υλοποιημένο σε γλώσσα Java» (Λαζαρίνης & Μαυρομάτης, 2015).

Οι μη λειτουργικές απαιτήσεις μπορούν να κατηγοριοποιηθούν με διάφορους τρόπους, π.χ. σε επιχειρησιακές (operational), απόδοσης (performance), ασφάλειας (security), κουλτούρας και πολιτικής (cultural and political) (Dennis et al., 2010). Ακόμα, έχει προταθεί ως εξίσου σημαντικό να καταγράφονται και χαρακτηριστικά που δεν πρέπει να έχει ένα σύστημα (Somerville, 2010).

Η συγγραφή των απαιτήσεων χρειάζεται ιδιαίτερη προσοχή, καθώς οι απαιτήσεις πρέπει να είναι απλές, σαφείς, περιεκτικές και πλήρεις. Ο αναλυτής οφείλει να κατέχει δεξιότητες πολύ καλού χειρισμού του γραπτού λόγου. Καθώς πολλές φορές οι απαιτήσεις είναι δεσμευτικές για την υλοποίηση και αποτελούν συμβατική υποχρέωση, οι επιπτώσεις ελλιπώς καταγεγραμμένων απαιτήσεων ενδέχεται να είναι καθοριστικές.

Ο εντοπισμός και η διατύπωση των λειτουργιών που θα επιτελεί το σύστημα είναι μια δύσκολη και ίσως μια από τις πλέον κρίσιμες φάσεις σε μεγάλα και πολύπλοκα έργα ανάπτυξης λογισμικού. Η όλη αυτή διαδικασία ονομάζεται μηχανική απαιτήσεων (requirements engineering) (Somerville, 2011 - Pressman, 2010) και συνίσταται στα ακόλουθα βήματα:

1. Εντοπισμός: γίνεται συνήθως μέσω διεξαγωγής συνεντεύξεων ή χρήσης ερωτηματολογίων και καταγραφής σεναρίων. Ο ρόλος του αναλυτή είναι σημαντικός, καθώς πρέπει να επιλέξει σε ποιους θα απευθυνθεί (διευθυντές, άλλοι εργαζόμενοι, πελάτες, προμηθευτές κ.λ.π.)
2. Ταξινόμηση και ομαδοποίηση: οι απαιτήσεις που συλλέγονται ομαδοποιούνται σε συναφείς λειτουργίες και δημιουργούν συνεκτικές συλλογές που περιγράφουν διαδικασίες.
3. Καθορισμός προτεραιότητας: σε περίπτωση αντικρουόμενων απαιτήσεων, η διαπραγμάτευση/συνεννόηση ανάμεσα στα μέρη θα οδηγήσει στην επίλυση τέτοιων προβλημάτων.

4. Προδιαγραφή: αφού συγκεντρωθούν οι απαιτήσεις, πρέπει να καταγραφούν με κάποιο συγκεκριμένο τρόπο.

Τα βήματα αυτά δε γίνονται σειριακά, καθώς συχνά χρειάζεται επιστροφή σε προηγούμενο βήμα, ώστε μέσα από την ανατροφοδότηση να καταλήξουμε στο επιθυμητό αποτέλεσμα.

2.1.2 Η μοντελοποίηση

Η μοντελοποίηση παίζει κεντρικό ρόλο στη σχεδίαση λογισμικού, διότι λειτουργεί ως γέφυρα ανάμεσα στις απαιτήσεις και στην υλοποίηση. Η μοντελοποίηση παρέχει μια οπτική αναπαράσταση του συστήματος λογισμικού, διευκολύνοντας την κατανόηση σύνθετων δομών και συμπεριφορών. Αυτή η οπτικοποίηση βοηθά στην επικοινωνία ιδεών και σχεδίων μεταξύ των μελών της ομάδας.

Ακόμα, επιτρέπει την εξέταση διαφορετικών οπτικών γωνιών για το λογισμικό, επιτρέποντας στα ενδιαφερόμενα μέρη να δουν το σύστημα από διάφορες οπτικές γωνίες. Πιο συγκεκριμένα, οι βασικοί ρόλοι της μοντελοποίησης είναι οι εξής:

- Κατανόηση και αποσαφήνιση απαιτήσεων: μετατρέπει τις αφηρημένες ή ασαφείς απαιτήσεις σε οπτικές και δομημένες αναπαραστάσεις και βοηθά όλους τους εμπλεκόμενους (αναλυτές, προγραμματιστές, πελάτες) να έχουν κοινή αντίληψη του συστήματος.
- Μείωση πολυπλοκότητας: διασπά το σύστημα σε υποσυστήματα, κλάσεις, διεργασίες και σχέσεις και επιτρέπει τη μελέτη του τμηματικά, χωρίς να χρειάζεται να δούμε όλη την πολυπλοκότητα μαζί.
- Υποστήριξη του σχεδιασμού: βοηθά στον καθορισμό της αρχιτεκτονικής του συστήματος, επιτρέπει τη σύγκλιση εναλλακτικών σχεδιαστικών λύσεων πριν την υλοποίηση και συνεισφέρει σε καλύτερες αποφάσεις για επεκτασιμότητα, συντηρησιμότητα και απόδοση.
- Επικοινωνία και τεκμηρίωση: τα μοντέλα (π.χ. διαγράμματα UML) λειτουργούν ως κοινή γλώσσα επικοινωνίας και αποτελούν επίσημη τεκμηρίωση του συστήματος, χρήσιμη και μετά την ολοκλήρωση της ανάπτυξης.

- Έγκαιρος εντοπισμός σφαλμάτων: επιτυγχάνεται ο εντοπισμός λογικών λαθών, ελλείψεων ή ασυνεπειών πριν γραφτεί ο κώδικας οπότε μειώνεται έτσι το κόστος διορθώσεων, καθώς τα λάθη εντοπίζονται σε πρώιμο στάδιο.
- Υποστήριξη αυτοματοποίησης: με τις σύγχρονες προσεγγίσεις ανάπτυξης, τα μοντέλα μπορούν να χρησιμοποιηθούν για αυτόματη παραγωγή κώδικα ή δοκιμών.

Τα μοντέλα χρησιμεύουν ως τεκμηρίωση για την αρχιτεκτονική, τον σχεδιασμό και τις απαιτήσεις του λογισμικού. Παρέχουν ένα σημείο αναφοράς που μπορεί να χρησιμοποιηθεί καθ' όλη τη διάρκεια του κύκλου ζωής ανάπτυξης λογισμικού. Υποστηρίζουν διάφορες διαδικασίες ανάπτυξης, όπως ανάλυση, σχεδιασμό και δοκιμές. Βοηθούν στη δομή των δραστηριοτήτων ανάπτυξης και στη διασφάλιση ότι λαμβάνονται υπόψη όλες οι πτυχές του λογισμικού.

Επίσης εστιάζουν στα βασικά χαρακτηριστικά του λογισμικού, παραλείποντας περιττές λεπτομέρειες. Τα μοντέλα πρέπει να είναι συνεπή και πλήρη για να είναι αποτελεσματικά. Θα πρέπει να αντιπροσωπεύουν με ακρίβεια τις απαιτήσεις και τον σχεδιασμό του λογισμικού, διασφαλίζοντας ότι καλύπτονται όλες οι απαραίτητες πτυχές.

2.2 Από την αφηρημένη σύλληψη σε σχεδιαστικά μοντέλα

Η διαδικασία μετάβασης από αφηρημένες έννοιες σε μοντέλα σχεδιασμού στο λογισμικό περιλαμβάνει τη μετατροπή ιδεών και απαιτήσεων υψηλού επιπέδου σε δομημένα σχέδια που μπορούν να υλοποιηθούν. Αυτό συνήθως περιλαμβάνει φάσεις όπως ο εννοιολογικός σχεδιασμός, ο αρχιτεκτονικός σχεδιασμός και ο λεπτομερής σχεδιασμός, όπου οι προγραμματιστές περιγράφουν πώς θα λειτουργεί το σύστημα και πώς θα αλληλεπιδρά με τους χρήστες και άλλα συστήματα.

Ο εννοιολογικός σχεδιασμός είναι η αρχική φάση στην ανάπτυξη λογισμικού όπου οι ιδέες μετασχηματίζονται σε ένα δομημένο μοντέλο. Αυτό περιλαμβάνει τη συλλογή και ανάλυση των στόχων που πρέπει να επιτύχει το λογισμικό, καθώς και την ανάπτυξη αφηρημένων αναπαραστάσεων της λειτουργικότητας και των αλληλεπιδράσεων του

συστήματος. Αυτή η φάση είναι κρίσιμη, καθώς θέτει τα θεμέλια για τα επόμενα στάδια σχεδιασμού και υλοποίησης.

Μόλις καθοριστεί ο εννοιολογικός σχεδιασμός, η διαδικασία μεταβαίνει σε πιο απτά σχεδιαστικά μοντέλα. Αυτή η διαδικασία περιλαμβάνει τον σχεδιασμό της διεπαφής (interface), δηλαδή τον καθορισμό του τρόπου με τον οποίο οι χρήστες θα αλληλεπιδρούν με το σύστημα, εστιάζοντας στην εμπειρία του χρήστη και γενικότερα τη χρηστικότητα. Επίσης περιλαμβάνει τον αρχιτεκτονικό σχεδιασμό, δηλαδή την περιγραφή των κύριων στοιχείων του συστήματος, των ευθυνών τους και του τρόπου με τον οποίο αλληλεπιδρούν. Εδώ πρέπει να ληφθούν αποφάσεις σχετικά με τη συνολική δομή και οργάνωση του λογισμικού.

2.3 Προσεγγίσεις στη σχεδίαση συστημάτων λογισμικού

Οι προσεγγίσεις στη σχεδίαση συστημάτων λογισμικού είναι δομημένες μεθοδολογίες για τον προγραμματισμό, τη σχεδίαση και την ανάπτυξη συστημάτων λογισμικού. Παρέχουν ένα πλαίσιο για την οργάνωση της διαδικασίας της ανάπτυξης, διασφαλίζοντας ότι τα έργα παραμένουν στο σωστό δρόμο, ανταποκρίνονται στις προσδοκίες των χρηστών και παρέχουν υψηλής ποιότητας, κλιμακούμενες λύσεις. Η επιλογή της σωστής προσέγγισης είναι κρίσιμη, καθώς μπορεί να επηρεάσει σημαντικά τα χρονοδιαγράμματα, το κόστος και τη συνολική επιτυχία του έργου. Διαφορετικές μεθοδολογίες είναι κατάλληλες για διαφορετικούς τύπους έργων, ανάλογα με παράγοντες όπως το μέγεθος της ομάδας εργασίας, η πολυπλοκότητα του έργου και οι απαιτήσεις ευελιξίας. Οι σημαντικότερες προσεγγίσεις σχεδίασης λογισμικού είναι οι ακόλουθες:

- Η μεθοδολογία Waterfall: είναι μια γραμμική, σειριακή προσέγγιση στο σχεδιασμό λογισμικού, όπου κάθε φάση (π.χ., συλλογή απαιτήσεων, σχεδιασμός, κωδικοποίηση, δοκιμή, ανάπτυξη) πρέπει να ολοκληρωθεί πριν προχωρήσουμε στην επόμενη. Είναι ιδανική για έργα με σαφώς καθορισμένες απαιτήσεις και σαφή τελικό στόχο, όπως ιατρικές συσκευές ή αεροδιαστημικά συστήματα, όπου οι αλλαγές στη μέση του έργου μπορεί να είναι δαπανηρές ή επικίνδυνες.

- Η μεθοδολογία Agile: είναι μια ευέλικτη προσέγγιση που δίνει έμφαση στη συνεχή βελτίωση και στα σχόλια των πελατών. Αντί για μια άκαμπτη ακολουθία, χωρίζει τα έργα σε μικρότερα, διαχειρίσιμα κομμάτια που ονομάζονται σπριντ(sprint). Κάθε σπριντ διαρκεί συνήθως 1-4 εβδομάδες και έχει ως αποτέλεσμα μια λειτουργική προσθήκη στο προϊόν. Αυτή η προσέγγιση ταιριάζει σε δυναμικώς μεταβαλλόμενα έργα όπου οι απαιτήσεις ενδέχεται να εξελιχθούν, όπως πλατφόρμες ηλεκτρονικού εμπορίου ή εφαρμογές νεοσύστατων επιχειρήσεων. Άρα οι βασικές διαφορές μεταξύ των δύο αυτών προσεγγίσεων που προαναφέρθηκαν είναι ότι η Waterfall είναι πιο άκαμπτη, καθυστερεί τις δοκιμές μέχρι τα τελευταία στάδια και εμπλέκει ενεργά τον πελάτη μόνο στην αρχή και το τέλος, ενώ η Agile είναι αντίστοιχα πιο ευέλικτη, ενσωματώνει τις δοκιμές σε όλη τη διαδικασία και εμπλέκει τον πελάτη σε κάθε στάδιο.
- Ο επαναληπτικός σχεδιασμός είναι μια επαναλαμβανόμενη, κυκλική προσέγγιση που εστιάζει στη συνεχή βελτίωση. Αντί να κατασκευάζουν ένα πλήρες σύστημα μονομιάς, οι ομάδες δημιουργούν ένα ελάχιστο βιώσιμο προϊόν (MVP) ή πρωτότυπο. (Ένα ελάχιστο βιώσιμο προϊόν είναι η απλούστερη έκδοση ενός προϊόντος που μπορεί να δημιουργηθεί, η οποία ωστόσο λύνει ένα βασικό πρόβλημα για τους χρήστες. Έχει ακριβώς όσα χαρακτηριστικά χρειάζεται για να είναι χρήσιμο και βοηθά να διαπιστωθεί αν η ιδέα λειτουργεί στον πραγματικό κόσμο. Δημιουργώντας ένα MVP, αποφεύγεται η σπατάλη υπερβολικού χρόνου και χρημάτων σε χαρακτηριστικά που οι άνθρωποι μπορεί να μην χρειάζονται ή να μην επιθυμούν.) Στη συνέχεια οι ομάδες συλλέγουν σχόλια χρηστών και βελτιώνουν τον σχεδιασμό σε πολλαπλούς κύκλους. Αυτή η προσέγγιση μειώνει σημαντικά τον κίνδυνο αποτυχίας, επιτρέποντας στους προγραμματιστές να εντοπίζουν τα προβλήματα έγκαιρα και να προσαρμόζονται ανάλογα με τις ανάγκες. Είναι ιδιαίτερα αποτελεσματικό για έργα με αβέβαιες απαιτήσεις ή ταχέως μεταβαλλόμενες αγορές, καθώς επιτρέπει τη συνεχή προσαρμογή και βελτίωση.
- Ο σχεδιασμός με επίκεντρο τον χρήστη τοποθετεί τον τελικό χρήστη στην καρδιά της διαδικασίας της ανάπτυξης. Δίνει προτεραιότητα στη χρηστικότητα, την προσβασιμότητα και τη συνολική ικανοποίηση του χρήστη, διασφαλίζοντας ότι το

τελικό προϊόν λύνει πραγματικά τα προβλήματα των χρηστών. Περιλαμβάνει εκτεταμένη έρευνα χρηστών και δοκιμές χρηστικότητας, οδηγώντας σε λογισμικό που ανταποκρίνεται καλύτερα στις ανάγκες του πραγματικού κόσμου. Για παράδειγμα, η Airbnb βασίζεται σε μεγάλο βαθμό στο σχεδιασμό με επίκεντρο τον χρήστη, βελτιώνοντας συνεχώς την πλατφόρμα της με βάση τα σχόλια των χρηστών για να βελτιώσει την εμπειρία κράτησης (Drobac-Ristovic & Kavoliunas, 2025).

Καθώς η τεχνολογία συνεχίζει να εξελίσσεται, ο σχεδιασμός λογισμικού πρόκειται να υποστεί σημαντικές αλλαγές. Κάποιες βασικές τάσεις που αναμένονται σύντομα είναι οι εξής:

- Εργαλεία σχεδιασμού που βασίζονται στην τεχνητή νοημοσύνη: Η τεχνητή νοημοσύνη μεταμορφώνει τον σχεδιασμό λογισμικού αυτοματοποιώντας εργασίες ρουτίνας όπως η δημιουργία κώδικα, η ανίχνευση σφαλμάτων και η βελτιστοποίηση της απόδοσης.
- Πλατφόρμες low-code / no-code: Αυτές οι πλατφόρμες δίνουν τη δυνατότητα σε μη τεχνικούς χρήστες να δημιουργούν πλήρως λειτουργικές διαδικτυακές εφαρμογές χωρίς να γράφουν ούτε μία γραμμή κώδικα.
- Ενσωμάτωση DevOps (Development Operations): Βασίζεται σε μεγάλο βαθμό στον αυτοματισμό, χρησιμοποιώντας διάφορα εργαλεία για την αυτοματοποίηση των δοκιμών, της ανάπτυξης και του ελέγχου. Αυτή η προσέγγιση επιταχύνει τον κύκλο ανάπτυξης και μειώνει την πιθανότητα διακοπών λειτουργίας και εμφάνισης προβλημάτων απόδοσης.

Η υιοθέτηση αυτών των σύγχρονων τάσεων σχεδιασμού είναι πιο κρίσιμη από ποτέ. Τέτοιες προσεγγίσεις αναμένεται να βελτιστοποιήσουν την ανάπτυξη, αλλά και να βοηθήσουν επίσης τις ομάδες εργασίας να παρέχουν προϊόντα υψηλότερης ποιότητας ταχύτερα, με λιγότερα σφάλματα και λιγότερο τεχνικό χρέος. (Το τεχνικό χρέος αναφέρεται στο μελλοντικό κόστος που σχετίζεται με την επιλογή συντομεύσεων ή τη λήψη μη βέλτιστων αποφάσεων κατά την ανάπτυξη λογισμικού, κάτι που μπορεί να οδηγήσει σε αυξημένη συντήρηση και επαναλειτουργία αργότερα.)

Καταλήγοντας, να αναφερθεί ότι ο σωστός σχεδιασμός έχει μεγάλη επίδραση στο λογισμικό, το οποίο μπορεί έτσι να αντέξει στη δοκιμασία του χρόνου. Το καλά σχεδιασμένο λογισμικό είναι πιο αποτελεσματικό, ασφαλές και επεκτάσιμο, μειώνοντας το μακροπρόθεσμο κόστος συντήρησης και βελτιώνοντας τη συνολική ικανοποίηση των χρηστών.

2.4 Προβλήματα επικοινωνίας χωρίς τυποποιημένα μοντέλα

Τα προβλήματα επικοινωνίας στην ανάπτυξη λογισμικού μπορούν να προκύψουν από διάφορους παράγοντες, όπως αντικρουόμενες ιδέες, πολιτισμικά ή γλωσσικά εμπόδια και περίπλοκες δομές ομάδων. Χωρίς τυποποιημένα μοντέλα, αυτά τα ζητήματα μπορούν να οδηγήσουν σε παρεξηγήσεις, καθυστερήσεις και αποτυχίες έργων, καθιστώντας την αποτελεσματική επικοινωνία απαραίτητη για την επιτυχία της ομάδας.

Οι αντικρουόμενες ιδέες και απόψεις μπορούν να οδηγήσουν σε ασαφή επικοινωνία εντός των ομάδων ανάπτυξης λογισμικού. Όταν τα μέλη της ομάδας έχουν διαφορετικές απόψεις και δυσκολεύονται να βρουν κοινό έδαφος, αυτό μπορεί να προκαλέσει ένταση, καθυστερήσεις και μειωμένη συνεργασία. Οι ανεπίλυτες συγκρούσεις μπορούν επίσης να οδηγήσουν σε σύγχυση σχετικά με τους στόχους, τις απαιτήσεις και τις προτεραιότητες του έργου, εμποδίζοντας τελικά την πρόοδο και την ποιότητα του τελικού προϊόντος.

Τα πολιτισμικά και γλωσσικά εμπόδια μπορούν να δημιουργήσουν σημαντικές προκλήσεις στην επικοινωνία εντός των ομάδων. Οι παρεξηγήσεις λόγω διαφορών στη γλωσσική επάρκεια, στα πολιτισμικά πρότυπα ή στους τρόπους επικοινωνίας μπορούν να οδηγήσουν σε σύγχυση, λάθη και μειωμένη παραγωγικότητα. Επιπλέον, οι πολιτισμικές διαφορές μπορούν να προκαλέσουν ένταση ή συγκρούσεις μεταξύ των μελών της ομάδας, γεγονός που δυσχεραίνει την οικοδόμηση εμπιστοσύνης, την αποτελεσματική συνεργασία και την επίτευξη των στόχων του έργου.

Μια περίπλοκη δομή ομάδας μπορεί να εμποδίσει την αποτελεσματική επικοινωνία στην ανάπτυξη λογισμικού δημιουργώντας πολλαπλά επίπεδα ιεραρχίας και γραφειοκρατίας. Αυτό μπορεί να οδηγήσει σε απομόνωση πληροφοριών, καθυστερημένη λήψη αποφάσεων

και μειωμένη διαφάνεια σε ολόκληρη την ομάδα. Μια πολύπλοκη δομή ομάδας μπορεί τελικά να επιβραδύνει την πρόοδο, να προκαλέσει αναντιστοιχίες και να αυξήσει τον κίνδυνο κακής επικοινωνίας και σφαλμάτων στη διαδικασία ανάπτυξης.

Τα κακά ή ακατάλληλα συστήματα και πλατφόρμες επικοινωνίας μπορούν να εμποδίσουν σε μεγάλο βαθμό την αποτελεσματική επικοινωνία στις ομάδες. Όταν τα εργαλεία είναι δύσχρηστα, δεν διαθέτουν τα απαραίτητα χαρακτηριστικά ή δεν ενσωματώνονται καλά με άλλα συστήματα, αυτό μπορεί να οδηγήσει σε χαμένα μηνύματα, καθυστερήσεις και απογοήτευση μεταξύ των μελών της ομάδας. Οι αναποτελεσματικές διαδικασίες επικοινωνίας, όπως οι ασαφείς οδηγίες για το πότε και πώς να επικοινωνείτε, μπορούν να προκαλέσουν σύγχυση, επικάλυψη προσπαθειών και μειωμένη παραγωγικότητα.

Οι απαιτήσεις του έργου που περιγράφονται ή επικοινωνούνται ελλιπώς μπορούν να αποτελέσουν σημαντική πηγή σύγχυσης και παρεξηγήσεων στις ομάδες. Όταν οι απαιτήσεις είναι ασαφείς, ελλιπείς ή δεν κοινοποιούνται αποτελεσματικά σε όλα τα μέλη της ομάδας, αυτό μπορεί να οδηγήσει σε υποθέσεις, παρερμηνείες και σφάλματα στη διαδικασία ανάπτυξης. Αυτή η έλλειψη σαφήνειας μπορεί να οδηγήσει σε προβλήματα και τελικά, θα καταλήξουμε με ένα προϊόν που δεν ανταποκρίνεται στους επιδιωκόμενους στόχους ή στις ανάγκες των χρηστών.

Μπορεί να υπάρχουν προβλήματα επικοινωνίας εντός της ομάδας, ειδικά σε συζητήσεις που χρησιμοποιούν τεχνική ορολογία. Όταν δεν υπάρχει κατάλληλη τεκμηρίωση, μπορεί να οδηγήσει σε παρερμηνείες του κώδικα ή οι αναθεωρήσεις κώδικα μπορεί να είναι ελλιπείς ή να μην έχουν γίνει διεξοδικά. Η βάση κώδικα ενδέχεται να περιέχει πολλά σφάλματα και η ομάδα θα αφιερώσει περισσότερο χρόνο στην αποσφαλμάτωση.

Μερικές φορές, τα στελέχη της ομάδας δεν ακολουθούν το επαγγελματικό πρωτόκολλο επικοινωνίας. Ορισμένα μέλη μπορεί να μην ακούν καλά ή να τους αρέσει να διακόπτουν τους άλλους κατά τη διάρκεια συζητήσεων. Άλλα άτομα μπορεί να κάνουν υποθέσεις για ένα συγκεκριμένο θέμα χωρίς να το συζητούν με τα υπόλοιπα μέλη της ομάδας. Όταν τα μέλη της ομάδας δεν ξέρουν πώς να επικοινωνούν σωστά, θα χάνονται πολλές πληροφορίες. Αυτό θα επηρεάσει επίσης το συνολικό ηθικό της ομάδας, με τα μέλη να αισθάνονται ότι δεν ακούγεται η άποψή τους και ότι υποτιμώνται (“Top 7 Communication Problems in Software Development and How to Solve Them”, 2024).

3. Η Ενοποιημένη Γλώσσα Μοντελοποίησης (UML)

Η Ενοποιημένη Γλώσσα Μοντελοποίησης (UML) αποτελεί τυποποιημένη γλώσσα οπτικής μοντελοποίησης, η οποία χρησιμοποιείται για την αναπαράσταση της δομής και της συμπεριφοράς συστημάτων λογισμικού, διευκολύνοντας την κατανόηση της αρχιτεκτονικής τους από τα μέλη της ομάδας ανάπτυξης (Sommerville, 2016).

3.1 Ιστορική εξέλιξη και τυποποίηση της UML

3.1.1 Το παρελθόν

Στα τέλη της δεκαετίας του 1950, παρουσιάστηκε η πρώτη αντικειμενοστρεφής γλώσσα προγραμματισμού, η Simula. Για να συνοδεύσουν αυτή τη νέα ιδέα των αντικειμενοστρεφών γλωσσών, άρχισαν επίσης να εμφανίζονται μέθοδοι για τον σχεδιασμό λογισμικού με αυτόν τον αντικειμενοστρεφή τρόπο, οι οποίες με την πάροδο του χρόνου αναφέρθηκαν ως γλώσσες μοντελοποίησης.

Μέχρι τα τέλη της δεκαετίας του 1980 υπήρχαν περισσότερες από πενήντα ξεχωριστές γλώσσες μοντελοποίησης - καθεμία με τη δική της σύνταξη, δομή και συμβολισμό. Υπήρχαν πολλά προβλήματα με αυτή την συντριπτική ποικιλία γλωσσών και έχει σημειωθεί ότι «τέτοιες ανοιχτές προσεγγίσεις (θα μπορούσαν) να επηρεάσουν και να περιορίσουν το σύστημα με απροσδόκητους τρόπους ή ακόμη και να οδηγήσουν σε αποτυχία.

Το 1989 ιδρύεται η Ομάδα Διαχείρισης Αντικειμένων (Object Management Group - OMG), με αρχικό στόχο «να δημιουργήσει ένα πρότυπο για την επικοινωνία μεταξύ κατανεμημένων αντικειμένων» (Cook, 2012, σ. 472). Αυτός ο στόχος είχε ως στόχο να προωθήσει την πρόοδο προς ένα κοινό μοντέλο αντικειμένων που θα λειτουργούσε σε όλες τις πλατφόρμες και σε όλα τα είδη έργων ανάπτυξης. Προκειμένου να προωθήσει αυτόν τον στόχο, ειδικά στον τομέα των γλωσσών μοντελοποίησης, η OMG ξεκίνησε την Ομάδα Ειδικού Ενδιαφέροντος Ανάλυσης και Σχεδιασμού Αντικειμένων για τη μελέτη μεθόδων σχεδιασμού.

Την εποχή που ιδρύθηκε η OMG, μια ξεχωριστή εταιρεία που ονομαζόταν Rational προσπαθούσε επίσης να εφαρμόσει μια λύση στον υπερκορεσμό των γλωσσών μοντελοποίησης που χρησιμοποιούνταν. Για το σκοπό αυτό, προσέλαβαν τους Grady Booch και James Rumbaugh το 1996. Αυτοί οι άνδρες ήταν οι δημιουργοί δύο από των κυρίαρχων γλωσσών μοντελοποίησης της εποχής.

Η μέθοδος του Booch ονομαζόταν Αντικειμενοστρεφής Σχεδίαση (OOD) (Booch, 1991) και η μέθοδος του Rumbaugh ήταν γνωστή ως Τεχνική Αντικειμενοστρεφούς Μοντελοποίησης (OMT). (Rumbaugh, Blaha, Lorensen, Eddy, & Premerlani, 1990). Σύντομα προστέθηκε και ο Ivar Jacobson, του οποίου η μέθοδος Αντικειμενοστρεφούς Μηχανικής Λογισμικού (OOSE) (Jacobson, 1992) ήταν επίσης μια εξέχουσα γλώσσα μοντελοποίησης εκείνη την εποχή.

Οι «Τρεις Φίλοι», όπως έγιναν αργότερα γνωστοί, άρχισαν στη συνέχεια να εργάζονται για την ανάπτυξη της Ενοποιημένης Γλώσσας Μοντελοποίησης. Μια δυνητικά καθολική τυποποιημένη μορφή σημειογραφίας με σκοπό να δημιουργούν ευκολία στην επικοινωνία και να μειώνουν τον κίνδυνο αποτυχίας των έργων, με τον ανθρώπινο παράγοντα να λαμβάνεται πάνω απ' όλα υπόψη, καθώς αυτό είχε αναγνωριστεί ως η κύρια αιτία αποτυχίας προηγούμενων έργων (Erickson & Siau, 2013).

Η προδιαγραφή UML 0.91 ήταν το αρχικό αποτέλεσμα της ενοποίησης των OOD, OMT και OOSE, μια κάπως επιτυχημένη προσπάθεια, καθώς κάθε βασική γλώσσα μοντελοποίησης είχε μοναδικά πλεονεκτήματα. Η OOD του Booch ήταν καλή για σχεδιασμό χαμηλού επιπέδου, η OMT του Rumbaugh ήταν αποτελεσματική για ανάλυση ΟΟ και η OOSE του Jacobson ήταν καλή για σχεδιασμό υψηλού επιπέδου, ενώ επέτρεπε την υλοποίηση περιπτώσεων χρήσης.

Με τους "Τρεις Φίλους" συνεργάστηκε μια ομάδα ανάπτυξης λογισμικού που εκπροσωπούσε μια σειρά από διαφορετικούς προμηθευτές και ολοκληρωτές συστημάτων, οι οποίοι θα συνεργάζονταν για να προτείνουν την UML ως την τυπική γλώσσα μοντελοποίησης για την OMG (Kobryn, 1999). Εκπρόσωποι από άλλες εταιρείες (όπως η IBM, η Microsoft και η Oracle) συσκέπηταν κατά τη διάρκεια του συνεδρίου Αντικειμενοστρεφούς Προγραμματισμού, Συστημάτων, Γλωσσών και Εφαρμογών (OOPSLA) που πραγματοποιήθηκε εκείνο το έτος, με το αποτέλεσμα αυτών των συσκέψεων να οδηγεί στο προσχέδιο UML 1.0, το οποίο στη συνέχεια υποβλήθηκε στην

OMG σε απάντηση στην Αίτηση Υποβολής Προσφοράς. Η UML 1.0 έγινε δεκτή από την OMG τον Νοέμβριο του 1997.

Η αρχική ανταπόκριση μετά την κυκλοφορία της προδιαγραφής έδειξε ότι η Ενοποιημένη Γλώσσα Μοντελοποίησης ήταν πολύ αποτελεσματική, αφού το εμπλεκόμενο προσωπικό είχε ξεπεράσει τη δύσκολη καμπύλη εκμάθησης μιας νέας γλώσσας μοντελοποίησης.

Στην πραγματικότητα, υπάρχουν εικασίες ότι η ανταπόκριση στην UML ήταν στην πραγματικότητα πολύ μεγάλη και ενώ αποδείχθηκε πολύ πιο αποτελεσματική από τους προκατόχους της, εξακολουθούσε να παρουσιάζει προβλήματα. Η ταχεία υιοθέτηση και η θετική ανταπόκριση σήμαιναν ότι η υιοθέτηση της UML κατέληξε να αυξάνεται με ανησυχητικό ρυθμό πριν ολοκληρωθεί η σωστή τυποποίησή της.

3.1.2 Το παρόν

Όταν αρχικά έγινε δεκτή ως πρότυπο, η UML 1.0 φάνηκε να πληροί όλες τις δηλωμένες απαιτήσεις και να είναι μια αποτελεσματική γλώσσα μοντελοποίησης. Έκτοτε, ωστόσο, έχουν πραγματοποιηθεί αρκετές αναθεωρήσεις για να τροποποιηθεί ο συμβολισμός, προκειμένου να διορθωθούν διάφορες αδυναμίες και να γίνει πιο αποτελεσματική. Υπήρχε επίσης πρόβλημα με την αρχιτεκτονική ευθυγράμμιση.

Μέσω του συστήματος OMG, έχουν γίνει ορισμένες σημαντικές αλλαγές ως απάντηση σε ορισμένες από τις ελλείψεις που εντοπίστηκαν στην UML. Μεταξύ της UML 1.1 και της UML 1.2, η προδιαγραφή αναδιαμορφώθηκε προκειμένου να ευθυγραμμιστεί καλύτερα με άλλες προδιαγραφές OMG. Σε αυτήν την αναθεώρηση, εντοπίστηκαν επίσης τυπογραφικά και γραμματικά λάθη.

Μεταξύ των εκδόσεων UML 1.2 και UML 1.3, διορθώθηκαν προβλήματα που είχαν προκύψει κατά τις τροποποιήσεις της τελευταίας αναθεώρησης, ολοκληρώθηκε ο συμβολισμός του γραφήματος δραστηριότητας και τα τυπικά στοιχεία οργανώθηκαν πιο επίσημα.

Οι αναθεωρήσεις της προδιαγραφής ήταν όλες μάλλον μικρές στη συνέχεια, μέχρι την κυκλοφορία της UML 2.0 το 2005. Η προδιαγραφή UML 2.0, σε αντίθεση με την UML 1.x, οργανώθηκε σε τέσσερις ενότητες:

- Η Υπερδομή η οποία ορίζει τα σύμβολα και τη σημασιολογία των διαγραμμάτων και των στοιχείων των μοντέλων.
- Η Υποδομή η οποία ορίζει το βασικό μεταμοντέλο στο οποίο βασίζεται η Υπερδομή.
- Η OCL για τον ορισμό κανόνων για στοιχεία μοντέλου.
- Η ανταλλαγή διαγραμμάτων UML, η οποία καθορίζει τον τρόπο ανταλλαγής των διατάξεων διαγραμμάτων UML 2.

Από την έκδοση 2.5 (Ιούνιος 2015) και έπειτα έχουν αφαιρεθεί η Υπερδομή και η Υποδομή και η πιο πρόσφατη έκδοση είναι η UML 2.5.1, η οποία εκδόθηκε τον Δεκέμβριο του 2017. Η UML συνεχίζει να ενημερώνεται, να βελτιώνεται και να επιλύονται τυχόντα προβλήματα.

Μελέτες έχουν δείξει ότι η συνεχιζόμενη εφαρμογή αναθεωρήσεων έχει επιτύχει με επιτυχία τον στόχο της τυποποίησης. Καθώς η UML γίνεται πιο εκλεπτυσμένη, γίνεται επίσης πιο προσβάσιμη και αποδεκτή παγκοσμίως. Μια μελέτη διαπίστωσε ότι το 21% των μελών της Αυστραλιανής Εταιρείας Υπολογιστών χρησιμοποιούσε συχνά την UML (Davies, 2006).

Περαιτέρω στοιχεία αυτής της αυξανόμενης τυποποίησης καταδείχθηκαν από τους Dobing και Parsons (2006), οι οποίοι σημείωσαν ότι τα διαγράμματα κλάσεων ήταν η πιο συχνά χρησιμοποιούμενη πτυχή της UML όπως αναφέρθηκε από το 73% των συμμετεχόντων. Από τη διεξαγωγή αυτών των μελετών έχει αποδειχθεί ότι οι επαγγελματίες εφαρμόζουν με επιτυχία την UML πιο αποτελεσματικά και συχνά, σε τέτοιο σημείο που πλέον αποτελεί μέρος πολλών προπτυχιακών πανεπιστημιακών προγραμμάτων σπουδών σε τομείς της Πληροφορικής.

3.1.3 Το μέλλον

Παρά τις αναθεωρήσεις, όλα αυτά τα χρόνια, που έχουν φέρει με επιτυχία σταδιακές βελτιώσεις στην προδιαγραφή, εξακολουθούν να υπάρχουν προβλήματα που πρέπει να αντιμετωπιστούν. Το επίπεδο πολυπλοκότητας και κάποιες ασυνέπειες στην ονομασία και την οργάνωση που εισήχθησαν κατά τη διάρκεια των αναθεωρήσεων, ήταν επιζήμιες για τη συνολική αναγνωσιμότητα της προδιαγραφής. Υπάρχει ο προβληματισμός ότι ο

σχεδιασμός των συμβόλων δεν είναι φιλικός προς τον χρήστη, γεγονός που θα αποθάρρυνε τους πιθανούς χρήστες από την υιοθέτηση και χρήση της UML υπέρ άλλων απλούστερων εναλλακτικών λύσεων.

Ένα άλλο πιθανό ζήτημα που πρέπει να αντιμετωπιστεί αφορά τη διαδικασία ενημέρωσης της UML, για την αντιμετώπιση των παραπάνω ζητημάτων και προβλημάτων, όπου η υπερβολική προσθήκη, αφαίρεση και τροποποίηση σημαντικών εννοιών θα μπορούσε να επηρεάσει τη βασική δομή της UML. Η τρέχουσα μέθοδος αναθεώρησης αφήνει την βασική δομή ευάλωτη.

Όπως αναφέρθηκε προηγουμένως, η λειτουργικότητα της UML επικεντρώθηκε κυρίως στη μηχανική λογισμικού. Ως αποτέλεσμα, τα εργαλεία λογισμικού προσφέρουν εκτεταμένη υποστήριξη για την UML όταν χρησιμοποιούνται με γνώμονα αυτή την εστίαση. Ωστόσο, υπάρχει πολύ μικρή υποστήριξη στα εργαλεία λογισμικού για οποιαδήποτε άλλη εφαρμογή, παρά τον στόχο της γλώσσας να αποτελεί μια τυποποιημένη αναπαράσταση μοντέλων γενικής χρήσης.

Έχει αποδειχθεί ότι η UML έχει εφαρμοστεί στον τομέα της μηχανικής λογισμικού όλο και περισσότερο με την πάροδο των ετών, μεταβαίνοντας από τη σχετικά χαμηλή χρήση στη βιομηχανία (π.χ. Davies, 2006) στην παρούσα κατάσταση όπου η αύξηση της χρήσης της UML έχει οδηγήσει σε μια πληθώρα εργαλείων και λογισμικού για την καλύτερη υποστήριξη της γλώσσας.

Ωστόσο, η έρευνα για την UML είναι περιορισμένη τα τελευταία χρόνια. Έχουν διεξαχθεί μερικές έρευνες με βάση τη χρήση της UML όσον αφορά τη μηχανική και την ανάπτυξη λογισμικού, αλλά πολύ λίγες όσον αφορά τις άλλες εφαρμογές της ως γλώσσα μοντελοποίησης γενικής χρήσης.

Η τρέχουσα μελέτη φαίνεται να επικεντρώνεται στη συμμόρφωση των εργαλείων με την UML, παρά στη συμμόρφωση της UML με τον επιδιωκόμενο σκοπό της. Μελλοντικές έρευνες για την υιοθέτηση της UML (τόσο εντός όσο και εκτός του τομέα της μηχανικής λογισμικού) θα ήταν καλό να συμπεριλάβουν στοιχεία σχετικά με την αντιληπτή αποτελεσματικότητα της γλώσσας μοντελοποίησης από τους χρήστες σε πραγματικές καταστάσεις.

Μια άλλη πιθανή ερευνητική κατεύθυνση θα μπορούσε να εξετάσει πώς η γλώσσα μοντελοποίησης έχει επηρεάσει τις πρακτικές ανάπτυξης και τη χρήση τεχνικών, και αν το συνολικό ποσοστό επιτυχίας του έργου έχει αυξηθεί ως αποτέλεσμα αυτού.

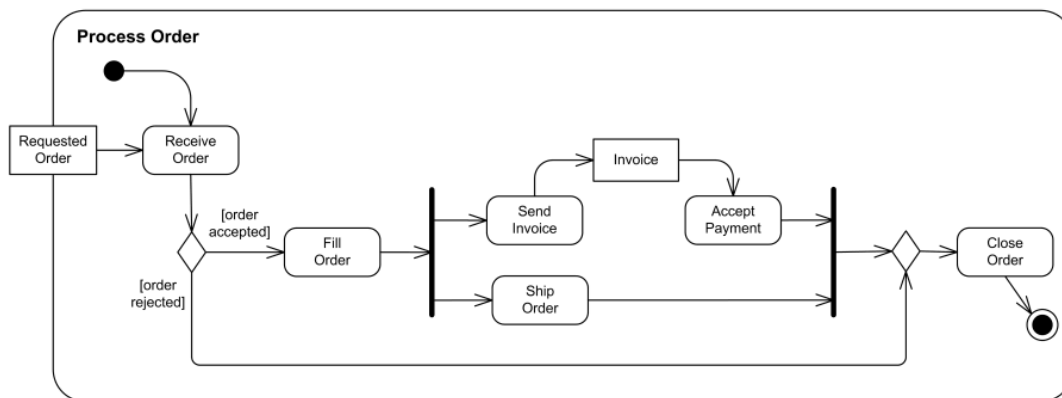
3.2 Σκοπός και φιλοσοφία της UML

Ο σκοπός της UML είναι να παρέχει έναν τυποποιημένο τρόπο απεικόνισης, καθορισμού, κατασκευής και τεκμηρίωσης των συστημάτων λογισμικού, διευκολύνοντας τη διαχείριση πολύπλοκων συστημάτων και βελτιώνοντας την επικοινωνία μεταξύ των ομάδων εργασίας. Στοχεύει στη δημιουργία μιας κοινής γλώσσας για τη μοντελοποίηση που είναι ανεξάρτητη από συγκεκριμένες γλώσσες προγραμματισμού και διαδικασίες ανάπτυξης. Αναπτύχθηκε για να αντιμετωπίσει διάφορους στόχους στο σχεδιασμό και τη μοντελοποίηση λογισμικού. Οι κύριοι στόχοι είναι:

- Ως κοινή γλώσσα οπτικής μοντελοποίησης να μπορεί να χρησιμοποιηθεί σε διάφορες μεθόδους ανάπτυξης λογισμικού ώστε να μπορούν να τυποποιηθούν ανάμοια συστήματα συμβολισμού στη σχεδίαση λογισμικού.
- Να παρέχει ένα οπτικό μέσο για τον καθορισμό, την απεικόνιση, την κατασκευή και την τεκμηρίωση αντικειμένων των συστημάτων λογισμικού.
- Να χρησιμεύει ως σχέδιο τόσο για συστήματα λογισμικού όσο και μη λογισμικά συστήματα, ενισχύοντας την επικοινωνία μεταξύ των ενδιαφερόμενων ομάδων έργου, διασφαλίζοντας την αρχιτεκτονική αρτιότητα και τη σαφήνεια στο σχεδιασμό του συστήματος.
- Να είναι εφαρμόσιμη σε διαφορετικές γλώσσες προγραμματισμού και διαδικασίες ανάπτυξης.
- Να επιτρέπει την επέκταση και την εξειδίκευση βασικών εννοιών για την κάλυψη συγκεκριμένων αναγκών.
- Να μοντελοποιεί αποτελεσματικά μεγάλα και σύνθετα συστήματα.
- Να αναπαριστά μια συλλογή βέλτιστων πρακτικών που έχουν αποδειχθεί επιτυχημένες στη μοντελοποίηση, βελτιώνοντας έτσι την ποιότητα του λογισμικού και μειώνοντας το κόστος.

Όλοι αυτοί οι στόχοι καθιστούν την UML ένα ζωτικό εργαλείο στη μηχανική λογισμικού, βοηθώντας στην διαχείριση της πολυπλοκότητας και στην ενίσχυση της συνεργασίας σε όλη τη διαδικασία ανάπτυξης, γι' αυτό και χρησιμοποιείται πλέον ευρέως τις τελευταίες δύο δεκαετίες από διάφορους ερευνητές.

Για παράδειγμα τα διαγράμματα δραστηριότητας της UML, λόγω του υψηλότερου επιπέδου αφάιρησης που διαθέτουν, είναι ιδιαίτερα κατάλληλα για τη δημιουργία δοκιμών που βασίζονται σε μοντέλα και οπτικοποιούν ιδανικά τη ροή εργασίας επιχειρηματικών διαδικασιών και λειτουργιών. Στο Σχήμα 1 απεικονίζεται παράδειγμα διαγράμματος δραστηριοτήτων UML για την επεξεργασία παραγγελίας αγοράς ενός προϊόντος.



Σχήμα 1: Διάγραμμα δραστηριοτήτων για την επεξεργασία παραγγελίας αγοράς προϊόντος.

3.3 Πλεονεκτήματα και περιορισμοί της UML στη θεωρητική σχεδίαση

Ειδικά για την επικοινωνία με τους προγραμματιστές, υπάρχουν πολλά πλεονεκτήματα των γραφικών συμβόλων σε σύγκριση με μορφές έκφρασης μέσω κειμένου. Επιτρέπουν στον θεατή να έχει μια γρήγορη επισκόπηση και να απλοποιεί την αντίληψη του για τα μέρη του συστήματος και τις σχέσεις τους.

Λόγω όμως του δισδιάστατου χαρακτήρα τους, οι μέθοδοι γραφικής περιγραφής έχουν και μειονεκτήματα. Χρειάζονται, για παράδειγμα, σημαντικά περισσότερο χώρο όταν παρουσιάζονται ή εκτυπώνονται. Αυτή η χαμηλότερη πυκνότητα πληροφοριών οδηγεί εύκολα σε έλλειψη συνολικής εικόνας, ειδικά όσον αφορά μεγάλα μοντέλα. Επιπλέον, ο

ακριβής ορισμός της σύνταξης και της σημασιολογίας μιας γραφικής γλώσσας θεωρείται γενικά πιο δύσκολος.

Η UML σημείωσε επιτυχία στην ενσωμάτωση διαφορετικών συμβόλων και, ως εκ τούτου, έγινε μια τυπική γλώσσα μοντελοποίησης για τη μηχανική λογισμικού και μπορεί να μεταφράσει χιλιάδες γραμμές κώδικα σε προσβάσιμα οπτικά διαγράμματα, επιταχύνοντας τις δυνατότητες ανάλυσης.

Έχει δείξει στην πράξη την ανωτερότητά της στα αρχικά στάδια εξέλιξης ενός συστήματος. Π.χ. τα διαγράμματα περιπτώσεων χρήσης και ακολουθίας παρέχουν ό,τι χρειάζεται για τη σύλληψη των χαρακτηριστικών του συστήματος, ενώ διατηρούν την απαιτούμενη απλότητα που απαιτείται για τον πελάτη ώστε αυτός να κατανοήσει τη συμπεριφορά του σχεδιασμού του συστήματος.

Επίσης η παρανόηση μεταξύ σχεδιαστή και πελάτη αποκλείεται με τη χρήση της UML και είναι γνωστό ότι η παρανόηση αυτή καθ' αυτή συνήθως οδηγεί σε αποτυχία τέτοια πρότζεκτ. Ειδικότερα τα διαγράμματα περιπτώσεων χρήσης είναι πολύτιμη πηγή για τη δημιουργία δοκιμαστικών σεναρίων και επαληθεύσεων σε μερικούς ή και πλήρως υλοποιημένα συστήματα.

Με την UML, έχει επιτευχθεί η ενσωμάτωση αρκετών υπάρχουσών γλωσσών μοντελοποίησης. Οι συντακτικές διαφορές έχουν εναρμονιστεί και έννοιες από διαφορετικά πεδία έχουν ενσωματωθεί στην βασική γλώσσα. Αν και αυτό δημιούργησε μια αρκετά μεγάλη και εν μέρει υπερφορτωμένη γλώσσα, μπορούμε να αναμένουμε ότι η UML θα διεκδικήσει τον τίτλο του βασικού προτύπου γλώσσας για τουλάχιστον μία ακόμη δεκαετία.

3.4 UML και σύγχρονες τάσεις στη σχεδίαση λογισμικού

Σήμερα η κυριαρχία της UML στην ανάπτυξη λογισμικού έχει μειωθεί. Με την άνοδο των ευέλικτων μεθοδολογιών, των συστημάτων cloud-native και των εργαλείων συνεργασίας σε πραγματικό χρόνο, πολλοί στην κοινότητα ανάπτυξης αμφισβητούν εάν η UML εξακολουθεί να είναι κατάλληλη.

Ένας από τους μεγαλύτερους παράγοντες στην παρακμή της UML είναι η ευρεία υιοθέτηση πρακτικών ανάπτυξης λογισμικού Agile και Lean. Αυτές οι μεθοδολογίες δίνουν έμφαση στην επαναληπτική ανάπτυξη, τη συνεχή παράδοση και το λειτουργικό λογισμικό έναντι της ολοκληρωμένης τεκμηρίωσης. Η UML, όντας πιο τυπική και με μεγάλο όγκο τεκμηρίωσης, συχνά φαίνεται να μην είναι συμβατή με αυτές τις σύγχρονες ροές εργασίας.

Με την άνοδο των μικροϋπηρεσιών, της υπολογιστικής χωρίς διακομιστή και των αρχιτεκτονικών που βασίζονται στο cloud, η ανάγκη για μοντελοποίηση μεγάλων, μονολιθικών συστημάτων έχει μειωθεί. Οι μικροϋπηρεσίες σχεδιάζονται ως μικρά, αποσυνδεδεμένα στοιχεία που επικοινωνούν μέσω API, πράγμα που σημαίνει ότι η πολυπλοκότητα του συστήματος είναι κατανεμημένη και όχι συγκεντρωτική. Σε αυτό το περιβάλλον, τα λεπτομερή διαγράμματα σε ολόκληρο το σύστημα που παρέχει η UML είναι συχνά περιττά.

Τα παραδοσιακά διαγράμματα UML είναι στατικά, απαιτώντας χειροκίνητες ενημερώσεις και προσπάθεια για να διατηρούνται συγχρονισμένα με τις εξελισσόμενες βάσεις κώδικα. Αντίθετα, τα σύγχρονα εργαλεία ανάπτυξης επικεντρώνονται στη συνεργασία και την ευελιξία σε πραγματικό χρόνο. Εργαλεία όπως τα Lucidchart, Miro και Figma επιτρέπουν στις ομάδες να δημιουργούν και να τροποποιούν διαγράμματα αρχιτεκτονικής σε πραγματικό χρόνο, καθιστώντας τη συνεργασία πιο απρόσκοπτη.

Ενώ η UML δεν είναι νεκρή, η πολυπλοκότητά της συχνά υπερτερεί της χρησιμότητάς της σε σύγχρονα, ταχέως εξελισσόμενα περιβάλλοντα ανάπτυξης. Ευτυχώς, αρκετές εναλλακτικές λύσεις προσφέρουν απλούστερους, πιο προσαρμόσιμους τρόπους μοντελοποίησης συστημάτων.

Η UML δεν είναι εντελώς ξεπερασμένη, αλλά η θέση της στο σύγχρονο τοπίο ανάπτυξης έχει αλλάξει. Για ορισμένες περιπτώσεις χρήσης, όπως μεγάλα εταιρικά συστήματα ή περιβάλλοντα όπου απαιτείται εκτεταμένη τεκμηρίωση, η UML εξακολουθεί να προσφέρει αξία. Ωστόσο, για την πλειονότητα των ομάδων ανάπτυξης που εργάζονται σε ευέλικτα περιβάλλοντα, μικροϋπηρεσιών ή cloud-native, οι εναλλακτικές λύσεις που υπάρχουν είναι πιο πρακτικές και σχετικές (McBride, 2024).

Επίσης η UML εφαρμόζεται σε πολυάριθμες πλατφόρμες για να μοντελοποιήσει τη δυναμική συμπεριφορά των συστημάτων (Rahmoune et al., 2015). Σε μια πρόσφατη

μελέτη (Barral et al., 2020), τα προφίλ UML χρησιμοποιούνται για να επιτρέπουν στους σχεδιαστές να διαχωρίζουν τις ανησυχίες των συστημάτων και να μοντελοποιούν την ιχνηλασιμότητα της διαδικασίας λήψης αποφάσεων σε σοβαρά παιχνίδια, ενώ σε μια μελέτη των (Rehman, Latif, & Zafar, 2019), τα διαγράμματα δραστηριοτήτων χρησιμοποιούνται για τη μοντελοποίηση της λειτουργικότητας ενός έξυπνου γραφείου.

Τα διαγράμματα UML μπορούν να μοντελοποιήσουν τόσο στατικές όσο και δυναμικές πτυχές (Hettab et al., 2015) και ειδικότερα τα διαγράμματα δραστηριοτήτων μπορούν να μοντελοποιήσουν σύνθετες ροές (Ahmad et al., 2019). Στη συνέχεια, τα διαγράμματα δραστηριότητας UML που περιγράφουν εικονικά πειράματα μπορούν να αντιστοιχιστούν σε οποιαδήποτε συγκεκριμένη γλώσσα υλοποίησης, όπως C ή Java, έτσι ώστε να μπορούν να ενσωματωθούν σε οποιοδήποτε εικονικό εργαστηριακό περιβάλλον.

Διάφορες έρευνες (Helms et al., 2017; Peng et al., 2016; Wilsdorf et al., 2019, 2020) έδειξαν την ανάγκη για μια επίσημη γλώσσα/σημειογραφία που να περιγράφει πειράματα σε διαφορετικά εκπαιδευτικά επίπεδα. Η επιλογή των διαγραμμάτων δραστηριοτήτων για τη μοντελοποίηση πειραμάτων εικονικού εργαστηρίου υποκινήθηκε από την ανάγκη συμμετοχής επιστημόνων από διάφορους κλάδους, συμπεριλαμβανομένης της Χημείας, της Βιολογίας και της Επιστήμης Υπολογιστών. Επειδή η UML διαθέτει μια τυποποιημένη σημειογραφία που είναι αναγνωρίσιμη και ευρέως κατανοητή από ερευνητές διάφορων κλάδων, καθίσταται ιδανική για διεπιστημονική συνεργασία.

Όσον αφορά τα εικονικά πειράματα, οι πειραματικές διαδικασίες που εκτελούνται σε ένα εικονικό εργαστήριο μπορούν να απεικονιστούν με τη χρήση διαγραμμάτων δραστηριοτήτων, τα οποία αποτελούν μια ευρέως χρησιμοποιούμενη και εύκολα κατανοητή γλώσσα μοντελοποίησης. Αυτά τα διαγράμματα προσφέρουν μια ολοκληρωμένη και ακριβή αναπαράσταση της πειραματικής διαδικασίας, διευκολύνοντας την ανάλυση από τα υπόλοιπα ενδιαφερόμενα μέρη.

Επιπλέον, η UML παρέχει ένα πλούσιο σύνολο γραφικών σημειώσεων και κατασκευών που μπορούν να χρησιμοποιηθούν για τη μοντελοποίηση σύνθετων συστημάτων και διαδικασιών, συμπεριλαμβανομένων των δυναμικών πτυχών των πειραμάτων εικονικού εργαστηρίου. Συνολικά, η χρήση των διαγραμμάτων δραστηριοτήτων διευκολύνει τη διεπιστημονική συνεργασία και υποστηρίζει την αυστηρή ανάλυση και σχεδιασμό πειραμάτων εικονικού εργαστηρίου (Σύψας, 2023).

4. Τα Διαγράμματα της UML: θεωρία και χρήση

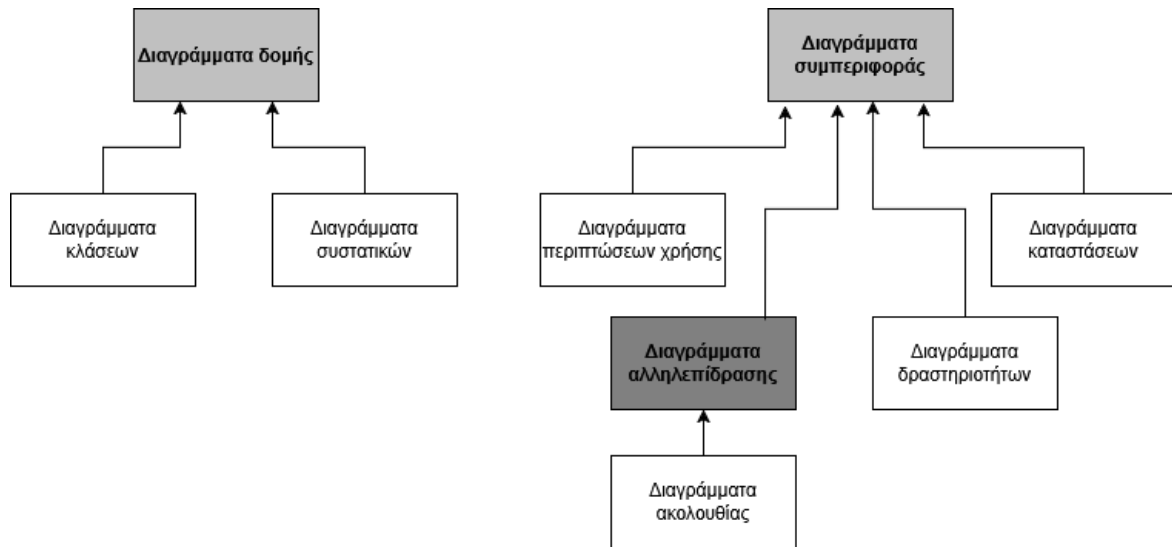
Η γλώσσα UML παρέχει μια σειρά από διαγράμματα τα οποία χρησιμοποιούνται στην ανάπτυξη συστημάτων λογισμικού αλλά και σε άλλους τομείς. Το διάγραμμα είναι μια οπτική αναπαράσταση ενός μοντέλου συστήματος. Η χρήση διαγραμμάτων διευκολύνει την επικοινωνία, την αποφυγή λαθών, τη δημιουργία νέων ιδεών, τον έλεγχο ιδεών και την κατανόηση δομών, συμπεριφορών και συσχετίσεων. Ένα διάγραμμα περιέχει συγκεκριμένα σύμβολα και ακολουθεί συγκεκριμένους κανόνες σύνταξης.

Στην έκδοση 2.5.1, η οποία είναι η πιο πρόσφατη έκδοση, η UML υποστηρίζει συνολικά 14 διαγράμματα. Τα διαγράμματα αυτά χωρίζονται σε δύο μεγάλες κατηγορίες: στα διαγράμματα δομής (structure diagrams) και στα διαγράμματα συμπεριφοράς (behavior diagrams).

Τα διαγράμματα δομής δείχνουν τη στατική δομή του συστήματος και των μερών του σε διαφορετικά επίπεδα αφαίρεσης και υλοποίησης και πώς αυτά τα μέρη σχετίζονται μεταξύ τους. Τα στοιχεία σε ένα διάγραμμα δομής αντιπροσωπεύουν τις ουσιαστικές έννοιες ενός συστήματος και μπορεί να περιλαμβάνουν αφηρημένες έννοιες, έννοιες του πραγματικού κόσμου και έννοιες υλοποίησης. Δεν χρησιμοποιούν έννοιες που σχετίζονται με τον χρόνο, δεν δείχνουν τις λεπτομέρειες της δυναμικής συμπεριφοράς. Σε αυτά ανήκουν τα διαγράμματα κλάσεων, παράταξης, αντικειμένων, συστατικών, πακέτων, προφίλ και σύνθετης δομής.

Τα διαγράμματα συμπεριφοράς δείχνουν τη δυναμική συμπεριφορά των αντικειμένων σε ένα σύστημα, η οποία μπορεί να περιγραφεί ως μια σειρά αλλαγών στο σύστημα με την πάροδο του χρόνου. Σε αυτά ανήκουν τα διαγράμματα δραστηριοτήτων, (μηχανών) καταστάσεων, περιπτώσεων χρήσης και υπάρχει μια υποκατηγορία με τα διαγράμματα αλληλεπίδρασης τα οποία είναι: τα διαγράμματα ακολουθίας, επικοινωνίας, συγχρονισμού και επισκόπησης αλληλεπίδρασης.

Στο Σχήμα 2 απεικονίζονται τα βασικά διαγράμματα UML στα οποία θα γίνει αναφορά στα επόμενες ενότητες του κεφαλαίου.



Σχήμα 2: Ταξινόμηση των βασικών UML διαγραμμάτων

4.1 Διαγράμματα Περιπτώσεων Χρήσης

Ένα διάγραμμα περιπτώσεων χρήσης (use case diagram) αναπαριστά οπτικά την αλληλεπίδραση μεταξύ χρηστών και του πληροφοριακού συστήματος. Ουσιαστικά οπτικοποιεί τις λειτουργίες του συστήματος από την μεριά ενός εξωτερικού παρατηρητή. Η έμφαση δίνεται στο τι κάνει ένα σύστημα και όχι στο πώς.

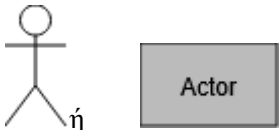
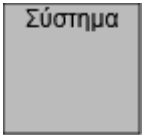
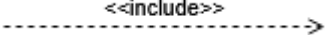
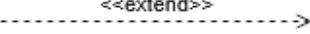
4.1.1 Βασικά στοιχεία των διαγραμμάτων περιπτώσεων χρήσης

Τα βασικά στοιχεία ενός διαγράμματος περίπτωσης χρήσης είναι:

- Ο χειριστής (actor), ο οποίος είναι μια εξωτερική οντότητα που αλληλεπιδρά με το σύστημα. Ο χειριστής μπορεί να είναι χρήστης ή κάποιο άλλο σύστημα και είναι αυτός που ξεκινάει την περίπτωση χρήσης.
- Η περίπτωση χρήσης (use case), η οποία σε ένα διάγραμμα περίπτωσης χρήσης αποτελεί οπτική αναπαράσταση μιας διακριτής επιχειρηματικής λειτουργίας σε ένα σύστημα.

- Τα όρια του συστήματος, εντός του οποίου περικλείονται οι περιπτώσεις χρήσης. Αυτά προσδιορίζουν έναν έμμεσο διαχωρισμό των χειριστών(εξωτερικοί του συστήματος) και των περιπτώσεων χρήσης(εσωτερικά του συστήματος).
- Η σχέση συσχέτισης, η οποία αναπαριστά την αλληλεπίδραση ανάμεσα στον χειριστή και την περίπτωση χρήσης. Απεικονίζεται με μια γραμμή που συνδέει τον δράστη με την περίπτωση χρήσης. Αυτή η σχέση υποδηλώνει ότι ο δράστης εμπλέκεται στη λειτουργικότητα που περιγράφεται από την περίπτωση χρήσης.
- Η σχέση γενίκευσης, η οποία υποδεικνύει ότι ένας χειριστής είναι μια εξειδικευμένη έκδοση ενός άλλου ή ότι μια περίπτωση χρήσης είναι μια εξειδικευμένη έκδοση μιας άλλης. Αναπαρίσταται από ένα βέλος που δείχνει από το εξειδικευμένο στο γενικό.
- Η σχέση συμπερίληψης, η οποία υποδεικνύει ότι μια περίπτωση χρήσης περιλαμβάνει τη λειτουργικότητα μιας άλλης περίπτωσης χρήσης. Υποδηλώνεται με ένα διακεκομμένο βέλος που δείχνει από την περίπτωση χρήσης που περιλαμβάνει την περίπτωση χρήσης στην περίπτωση χρήσης που περιλαμβάνεται.
- Η σχέση επέκτασης, η οποία δείχνει ότι μια περίπτωση χρήσης μπορεί να επεκταθεί από μια άλλη περίπτωση χρήσης υπό συγκεκριμένες συνθήκες. Αναπαρίσταται με ένα διακεκομμένο βέλος. Αυτή η σχέση είναι χρήσιμη για τον χειρισμό προαιρετικής ή εξαιρετικής συμπεριφοράς.

Ο Πίνακας 4.1 περιέχει τα σύμβολα των βασικών στοιχείων των διαγραμμάτων περιπτώσεων χρήσης.

Σύμβολο	Εξήγηση
	Χειριστής (actor)
	Περίπτωση χρήσης (use case)
	Όρια συστήματος (system boundary)
	Σχέση συσχέτισης μεταξύ χειριστή και περίπτωσης χρήσης
	Σχέση γενίκευσης είτε μεταξύ δύο χειριστών είτε μεταξύ δύο περιπτώσεων χρήσης
	Σχέση συμπερίληψης μεταξύ δύο περιπτώσεων χρήσης
	Σχέση επέκτασης μεταξύ δύο περιπτώσεων χρήσης

Πίνακας 4.1: Συμβολισμός βασικών στοιχείων διαγραμμάτων περιπτώσεων χρήσης.

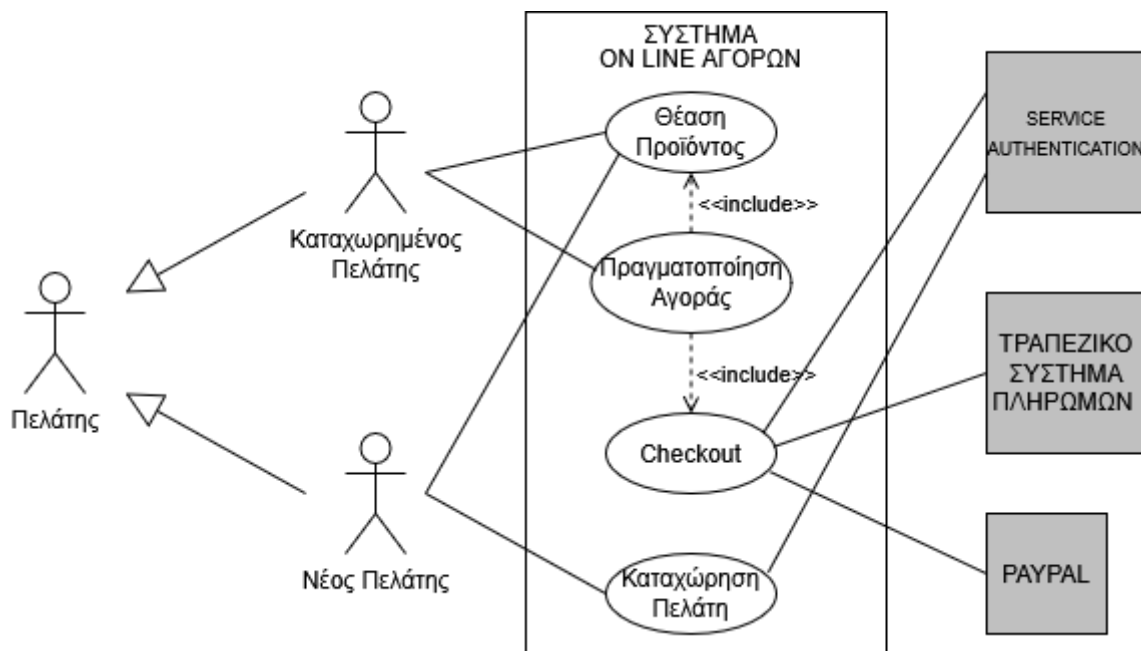
4.1.2 Συνεισφορά στη φάση ανάλυσης απαιτήσεων

Στη φάση της ανάλυσης υπάρχει η ανάγκη αναγνώρισης των απαιτήσεων των χρηστών του συστήματος, δηλαδή ποιες θα είναι οι λειτουργίες του. Η σαφής καταγραφή των

επιθυμητών λειτουργιών και η επικύρωσή τους μέσα από τους ίδιους τους χρήστες είναι ζωτικής σημασίας στην επιτυχή ανάπτυξη του έργου.

Τα διαγράμματα περιπτώσεων χρήσης παίζουν σημαντικό ρόλο στο σημείο αυτό, διότι βοηθούν στη γραφική απεικόνιση των λειτουργιών του συστήματος με τρόπο που είναι κατανοητός και στους πελάτες και στους προγραμματιστές. Τα διαγράμματα περιπτώσεων χρήσης απεικονίζουν γραφικά τις περιπτώσεις χρήσης. Οι βασικοί συμβολισμοί είναι οι ελλείψεις που απεικονίζουν τις περιπτώσεις χρήσης και τα ανθρωπάκια που αναπαριστούν τους χειριστές.

Βασικός σκοπός των διαγραμμάτων αυτών είναι να περιγραφούν οι λειτουργικές απαιτήσεις του συστήματος με ένα σαφή, εύληπτο και συνεπές τρόπο. Τα διαγράμματα αυτά θα βοηθήσουν στο να επαληθευτεί αν το σύστημα υποστηρίζει όλες τις επιθυμητές λειτουργίες. Στο Σχήμα 3 απεικονίζεται ένα διάγραμμα περιπτώσεων χρήσης για ένα σύστημα on line αγορών.



Σχήμα 3: Διάγραμμα περιπτώσεων χρήσης για σύστημα on line αγορών.

4.1.3 Σκοπός και οφέλη των διαγραμμάτων περιπτώσεων χρήσης

Τα διαγράμματα περιπτώσεων χρήσης προσφέρουν μια σαφή οπτική αναπαράσταση των λειτουργιών ενός συστήματος και απεικονίζουν τους διαφορετικούς τρόπους με τους οποίους οι χρήστες αλληλεπιδρούν με το σύστημα. Αυτή η αναπαράσταση βοηθά τα

ενδιαφερόμενα μέρη, συμπεριλαμβανομένων εκείνων που δεν έχουν τεχνική εμπειρία, να κατανοήσουν τις λειτουργίες και τη συνολική συμπεριφορά του συστήματος.

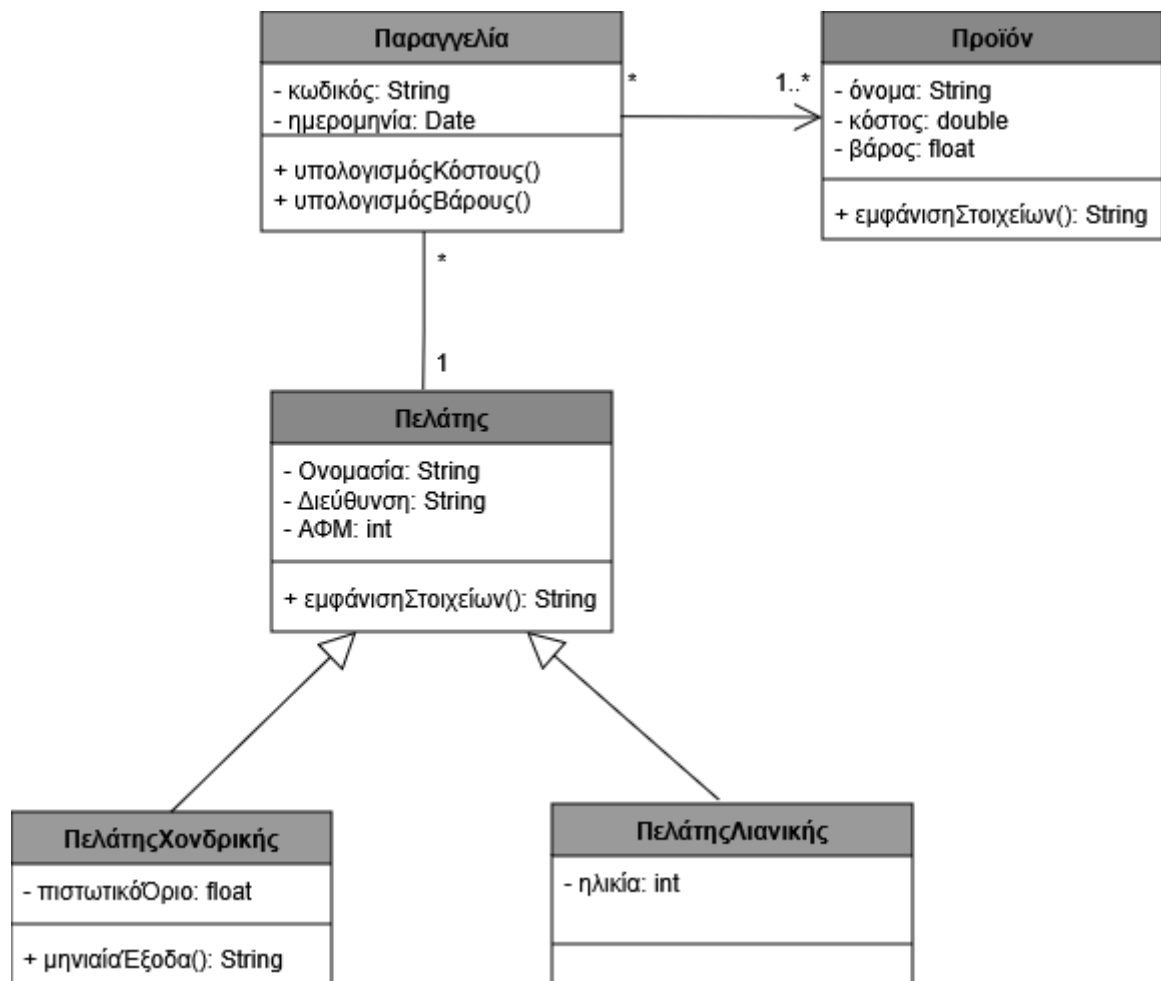
Δημιουργούν μια κοινή γλώσσα για τη διατύπωση των απαιτήσεων του συστήματος, διασφαλίζοντας ότι όλα τα μέλη της ομάδας έχουν κοινή κατανόηση και έτσι μπορούν να αποφευχθούν οι παρεξηγήσεις και οι παρανοήσεις μεταξύ των μελών.

Στη φάση του σχεδιασμού, τα διαγράμματα περιπτώσεων χρήσης βοηθούν στην περιγραφή του τρόπου με τον οποίο οι χρήστες θα αλληλεπιδράσουν με το σύστημα. Υποστηρίζουν τον σχεδιασμό των διεπαφών χρήστη και βοηθούν στη δομή των λειτουργιών του συστήματος.

4.2 Διαγράμματα Κλάσεων

Ο στόχος των διαγραμμάτων κλάσεων (class diagrams) είναι η μοντελοποίηση της στατικής δομής του λογισμικού και τελικά του αρχιτεκτονικού σχεδιασμού του. Είναι τα μόνα διαγράμματα που μπορούν να αντιστοιχιστούν απευθείας στις γλώσσες προγραμματισμού και έτσι χρησιμοποιούνται πάντα στην ανάλυση και σχεδίαση αντικειμενοστρεφών συστημάτων. Δεν απεικονίζουν ενέργειες, αλλά τη στατική πλευρά του συστήματος, δηλαδή τις κλάσεις που συνθέτουν το σύστημα που θα αναπτυχθεί. Πιο συγκεκριμένα περιγράφουν λεπτομερώς το πώς οι κλάσεις αλληλεπιδρούν και σχετίζονται μεταξύ τους χωρίς να εστιάζουν στη συμπεριφορά.

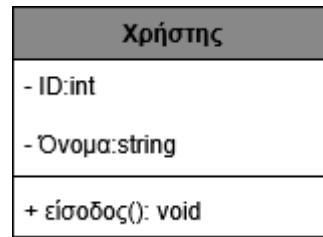
Τα βασικά δομικά στοιχεία των διαγραμμάτων κλάσεων είναι οι κλάσεις (με τις ιδιότητες και τις μεθόδους τους) και οι σχέσεις με τις οποίες συνδέονται οι κλάσεις. Οι σχέσεις αποτελούνται από σχέσεις συσχέτισης, συνάθροισης, σύνθεσης και γενίκευσης. Στο Σχήμα 4 απεικονίζεται ένα διάγραμμα κλάσεων.



Σχήμα 4: Διάγραμμα κλάσεων.

4.2.1 Κλάση

Κλάση είναι ένα σύνολο αντικειμένων που έχουν την ίδια δομή, την ίδια συμπεριφορά και μοιράζονται ίδια χαρακτηριστικά. Οι κλάσεις απεικονίζονται ως ένα ορθογώνιο σχήμα με τρία τμήματα. Στο πρώτο τμήμα εμφανίζεται το όνομα της κλάσης, στο δεύτερο εμφανίζονται οι ιδιότητες (ή γνωρίσματα) της κλάσης και στο τελευταίο τμήμα αναγράφονται οι μέθοδοι (ή λειτουργίες) της κλάσης. Οι μέθοδοι αναπαριστούν τη συμπεριφορά ή την λειτουργικότητα της κλάσης. Στο Σχήμα 5 απεικονίζεται ο βασικός συμβολισμός για την αναπαράσταση μιας κλάσης.



Σχήμα 5: Βασικός συμβολισμός κλάσης.

Η πρόσβαση σε κάθε ιδιότητα και μέθοδο μπορεί να είναι ιδιωτική (private) ή δημόσια (public) ή προστατευμένη (protected). Για τη σηματοδότηση της ορατότητας (visibility) χρησιμοποιούνται τα σύμβολα “ - “ (ιδιωτική), “ + ” (δημόσια), “ # ” (προστατευμένη). Πάντως να σημειωθεί ότι η σημασία των συμβόλων δεν είναι ίδια σε όλες τις αντικειμενοστρεφείς γλώσσες προγραμματισμού.

4.2.2 Συσχέτιση κλάσεων

Η συσχέτιση των κλάσεων απεικονίζει μια σχέση συνεργασίας μεταξύ δύο κλάσεων, όπως φαίνεται στο Σχήμα 6. Με απλά λόγια, η συσχέτιση δείχνει ότι για την εκτέλεση μιας λειτουργίας μιας κλάσης απαιτείται η συνεργασία με μία ή περισσότερες άλλες κλάσεις.



Σχήμα 6: Συσχέτιση δύο κλάσεων με εμφάνιση πολλαπλότητας.

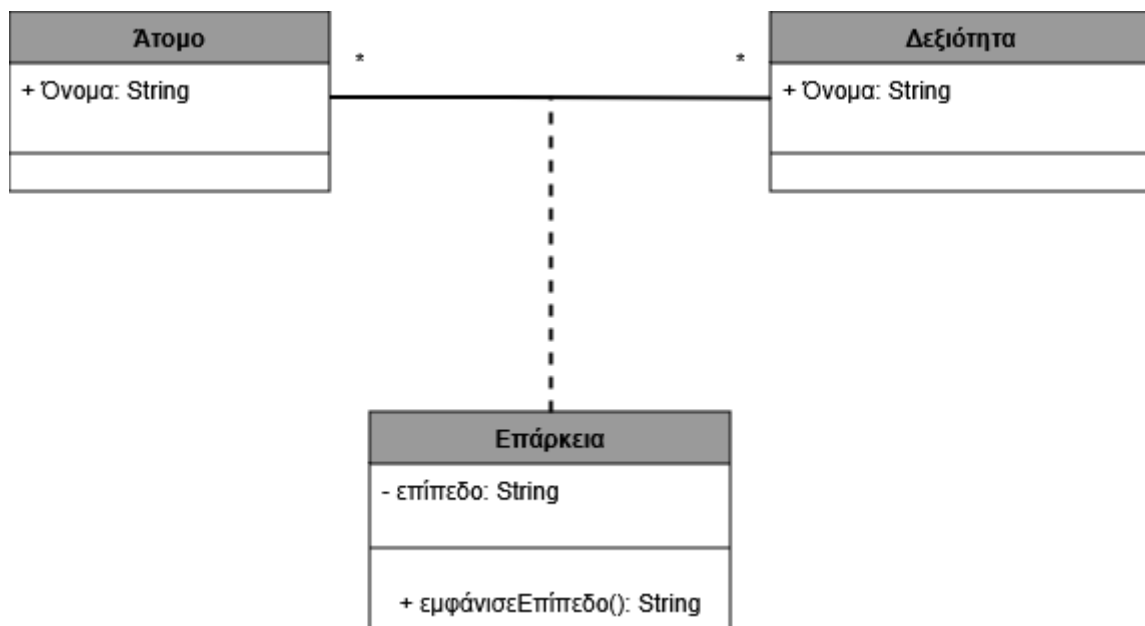
Η συσχέτιση δύο κλάσεων χρησιμοποιείται όταν:

- Η ιδιότητα μιας κλάσης είναι αντικείμενο μιας συσχετιζόμενης κλάσης.
- Μια μέθοδος μιας κλάσης βασίζεται στη συσχετιζόμενη κλάση.
- Ένα στιγμιότυπο μιας κλάσης πρέπει να γνωρίζει τη συσχετιζόμενη κλάση για να μπορεί να εκτελέσει τη λειτουργία του.
- Υπάρχει ανταλλαγή μηνυμάτων μεταξύ αντικειμένων των κλάσεων.

Οι συσχετιζόμενες κλάσεις συνδέονται με μία γραμμή, πάνω στην οποία μπορούν να προστεθούν πληροφορίες, όπως είναι το όνομα της συσχέτισης, η πολλαπλότητα, τα ονόματα άκρων συσχέτισης και η πλοηγησιμότητα.

4.2.3 Κλάση συσχέτισης

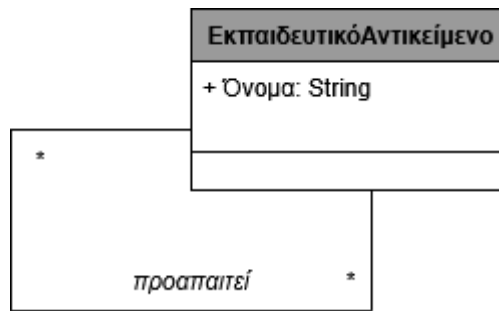
Σε κάποιες περιπτώσεις, μια ιδιότητα που αφορά δύο συσχετιζόμενες κλάσεις δεν μπορεί να τοποθετηθεί σε καμία από τις δύο κλάσεις και η συσχέτιση μεταξύ δύο κλάσεων μπορεί να διαθέτει και η ίδια ιδιότητες ή μεθόδους. Σε μια τέτοια περίπτωση δημιουργείται μια νέα κλάση που ονομάζεται κλάση συσχέτισης. Αυτή η κλάση εισάγεται ανάμεσα από τις δύο κλάσεις, όπως φαίνεται στο Σχήμα 7, και συνδέεται με διακεκομμένη γραμμή με τη συσχέτιση.



Σχήμα 7: Κλάση συσχέτισης.

4.2.4 Αυτοσυσχέτιση

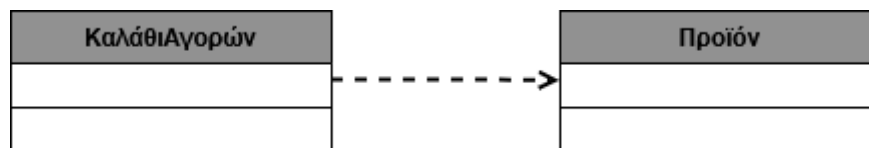
Μια κλάση είναι δυνατόν να συσχετίζεται με τον εαυτό της. Αυτό επιτυγχάνεται με την ύπαρξη μιας κυκλικής συσχέτισης η οποία καλείται αυτοσυσχέτιση ή ανακλαστική συσχέτιση. Ο συμβολισμός μιας κλάσης που αυτοσυσχετίζεται φαίνεται στο Σχήμα 8.



Σχήμα 8: Αυτοσυσχέτιση.

4.2.5 Σχέση εξάρτησης κλάσεων

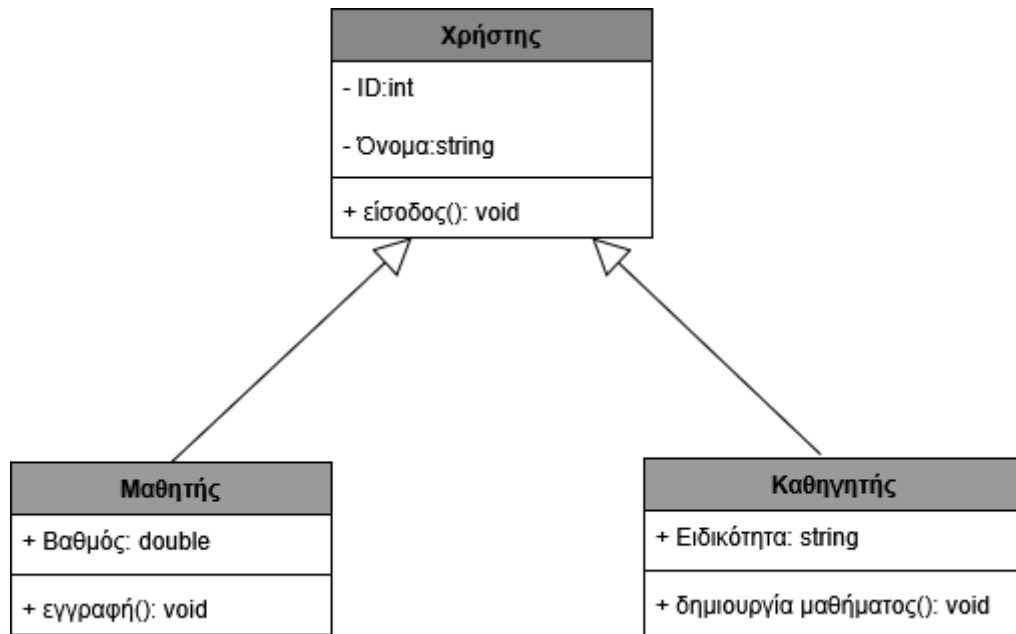
Η σχέση εξάρτησης μεταξύ δύο κλάσεων δημιουργεί μια εξάρτηση μεταξύ των δύο αυτών κλάσεων, με την έννοια ότι μια αλλαγή στη μία κλάση επηρεάζει την άλλη. Συμβολίζεται με διακεκομμένη γραμμή και ανοικτό βέλος στη μία άκρη της γραμμής, όπως απεικονίζεται στο Σχήμα 9.



Σχήμα 9: Σχέση εξάρτησης κλάσεων.

4.2.6 Σχέση γενίκευσης κλάσεων

Η γενίκευση των κλάσεων χρησιμοποιείται για να αποδώσει την αρχή της κληρονομικότητας των αντικειμενοστρεφών συστημάτων. Στο Σχήμα 10 απεικονίζεται μια σχέση γενίκευσης, όπου οι κλάσεις «Μαθητής» και «Καθηγητής» κληρονομούν από την γενική κλάση «Χρήστης».

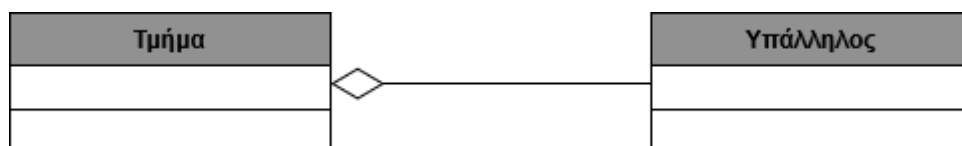


Σχήμα 10: Σχέση γενίκευσης κλάσεων.

4.2.7 Σχέση συνάθροισης κλάσεων

Οι σχέσεις περιεκτικότητας συνάθροισης και σύνθεσης, που παρουσιάζονται στις δύο επόμενες υποενότητες, αποτελούν πιο εξειδικευμένες εκφράσεις της συσχέτισης των κλάσεων ή με άλλα λόγια, πιο ισχυρές συσχετίσεις μεταξύ των κλάσεων.

Η σχέση συνάθροισης, ή συναρμολόγησης ή συσσωμάτωσης, δηλώνει μεν ότι κάποιες κλάσεις συσχετίζονται, αλλά επιπρόσθετα και ότι μία κλάση είναι μέρος της άλλης. Η κλάση από την οποία ξεκινά το βέλος είναι ουσιαστικά το μέρος του συνόλου που είναι από την πλευρά του άδειου ρόμβου. Στο παράδειγμα που απεικονίζεται στο Σχήμα 11, αν αφαιρεθεί η κλάση «Υπάλληλος», η κλάση «Τμήμα» θα εξακολουθεί να έχει νόημα, εννοώντας ένα μη στελεχωμένο τμήμα, το οποίο ως φυσική οντότητα έχει νόημα να υπάρχει, άρα θα πρέπει να απεικονιστεί και προγραμματιστικά.



Σχήμα 11: Σχέση συνάθροισης κλάσεων.

4.2.8 Σχέση σύνθεσης κλάσεων

Η σχέση σύνθεσης κλάσεων αναφέρεται και αυτή στη συσχέτιση μέρους-όλου μεταξύ κλάσεων, αλλά το μέρος είναι απαραίτητο για την ύπαρξη του όλου. Συμβολίζεται με βέλος με ένα γεμάτο ρόμβο στην πλευρά του όλου. Στο παράδειγμα που απεικονίζεται στο Σχήμα 12, στη συσχέτιση «Τμήμα» - «Υπάλληλος», η ύπαρξη ενός αντικειμένου της κλάσης «Τμήμα» απαιτεί και την ύπαρξη ενός αντικειμένου της κλάσης «Υπάλληλος».



Σχήμα 12: Σχέση σύνθεσης κλάσεων.

4.3 Διαγράμματα Ακολουθίας

Τα διαγράμματα ακολουθίας (sequence diagrams) χρησιμοποιούνται για την απεικόνιση της αλληλεπίδρασης των αντικειμένων μέσω της ανταλλαγής μηνυμάτων, στο πλαίσιο ενός σεναρίου χρήσης ενός συστήματος λογισμικού. Αποτελούν τον έναν από τους τέσσερις τύπους διαγραμμάτων αλληλεπίδρασης (interaction diagrams) και παρέχουν την ίδια πληροφορία με τα διαγράμματα επικοινωνίας (communication diagrams) της UML. Στα διαγράμματα ακολουθίας δίνεται έμφαση στη χρονική ακολουθία των ενεργειών που πραγματοποιούνται για την επίτευξη μιας λειτουργίας, ενώ στα διαγράμματα επικοινωνίας δίνεται μεγαλύτερη έμφαση στα εμπλεκόμενα αντικείμενα που αλληλεπιδρούν.

4.3.1 Μοντελοποίηση αλληλεπιδράσεων

Ένα διάγραμμα ακολουθίας αποτυπώνει τον τρόπο με τον οποίο τα αντικείμενα ενός συστήματος αλληλεπιδρούν μεταξύ τους για την επίτευξη ενός σεναρίου χρήσης. Σε ένα τέτοιο διάγραμμα προσδιορίζονται τα μηνύματα που αποστέλλονται μεταξύ αντικειμένων, αλλά και η χρονική ακολουθία των μηνυμάτων αυτών. Τα διαγράμματα αυτά σχεδιάζονται σε δύο διαστάσεις: η κάθετη διάσταση αντιστοιχεί στην κλίμακα του χρόνου, και ένα μήνυμα, που απεικονίζεται ως ακμή κάτω από ένα άλλο μήνυμα,

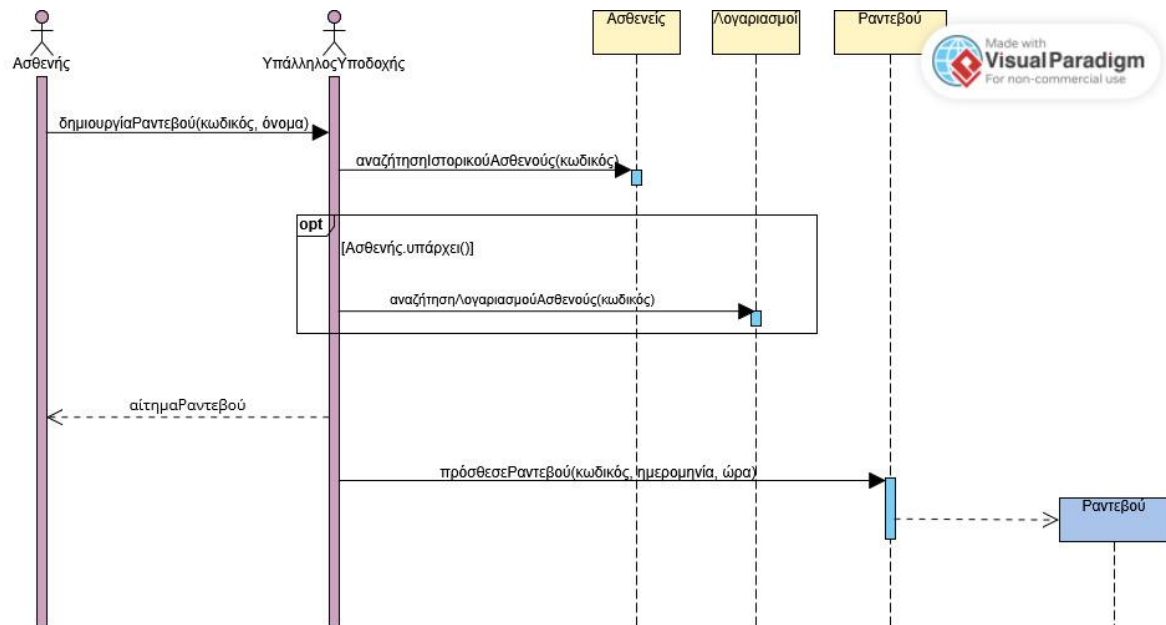
υπονοείται ότι αποστέλλεται σε μεταγενέστερη χρονική στιγμή. Στην οριζόντια διάσταση αναπαριστώνται τα εμπλεκόμενα αντικείμενα του εξεταζόμενου σεναρίου ή, σε ορισμένες περιπτώσεις, οι εξωτερικοί χειριστές του συστήματος (π.χ. ανθρώπινοι χρήστες που ενεργοποιούν μια περίπτωση χρήσης).

Κάτω από κάθε αντικείμενο εκτείνεται η (διακεκομμένη) γραμμή ζωής του, που υποδηλώνει το χρονικό διάστημα κατά το οποίο το αντικείμενο υφίσταται στη μνήμη. Όταν το αντικείμενο γίνει αποδέκτης ενός μηνύματος, η ροή του ελέγχου μεταβιβάζεται στο αντικείμενο αυτό, γεγονός που υποδηλώνεται από ένα ορθογώνιο πλαίσιο ενεργοποίησης πάνω στη γραμμή ζωής του αντικειμένου. Το πλαίσιο ενεργοποίησης παύει να υπάρχει όταν ολοκληρωθεί η μέθοδος που αποκρίθηκε στη λήψη του μηνύματος.

Συνίσταται από πολλούς να μη χρησιμοποιούνται πλαίσια ενεργοποίησης, καθώς καθιστούν τη μελέτη του διαγράμματος πιο περίπλοκη, χωρίς να παρέχουν ιδιαίτερα σημαντική πληροφορία στους προγραμματιστές οι οποίοι με βάση το διάγραμμα ακολουθίας θα υλοποιήσουν το αντίστοιχο σύστημα λογισμικού.

Οι ακμές που συμβολίζουν την αποστολή μηνυμάτων από ένα αντικείμενο σε κάποιο άλλο, αντιστοιχούν σε κλήσεις μεθόδων. Για αυτόν το λόγο, σε κάθε μήνυμα σημειώνεται το όνομα της αντίστοιχης μεθόδου που καλείται. Σε περίπτωση που απαιτείται η μεταβίβαση παραμέτρων για την εκτέλεση της καλούμενης μεθόδου, οι παράμετροι συμβολίζονται μέσα σε παρενθέσεις μαζί με το όνομα του μηνύματος. Η ολοκλήρωση μιας λειτουργίας από ένα αντικείμενο (που ισοδυναμεί με επιστροφή της καλούμενης μεθόδου) μπορεί να συμβολιστεί με ένα μήνυμα επιστροφής. Ωστόσο, μηνύματα επιστροφής δεν απεικονίζονται πάντοτε στα διαγράμματα ακολουθίας για λόγους ανάγνωσης. Η εισαγωγή τους συνίσταται όταν είναι επιθυμητό να συμβολιστεί ρητά η επιστροφή μιας τιμής.

4.3.2 Σενάριο εκτέλεσης



Σχήμα 13: Διάγραμμα ακολουθίας (επεξεργασία μέσω Visual Paradigm).

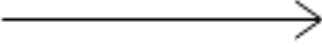
Στο Σχήμα 13 παρουσιάζεται ένα διάγραμμα ακολουθίας που εξετάζει την δημιουργία ιατρικού ραντεβού. Στο σενάριο ο ασθενής αιτείται ραντεβού μέσω της εφαρμογής του ιατρικού κέντρου. Ο υπάλληλος υποδοχής αναζητά το ιστορικό του ασθενούς, δηλαδή αν ο ασθενής είναι καταχωρημένος στη βάση δεδομένων και εφ' όσον υπάρχει αναζητά αν ο ασθενής έχει κάποιους εκκρεμείς λογαριασμούς με την επιχείρηση. Στη συνέχεια ο ασθενής ειδοποιείται για την επικύρωση του ραντεβού, ενημερώνεται η βάση δεδομένων με τα καταχωρημένα ραντεβού.

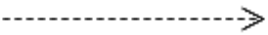
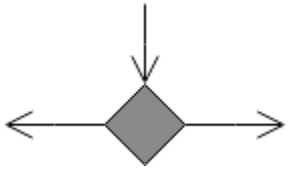
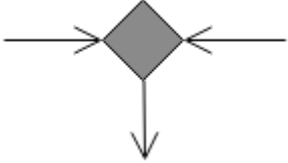
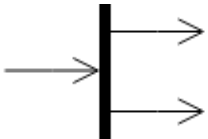
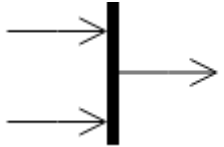
Στο συγκεκριμένο σενάριο έχουμε δύο χειριστές (Ασθενής, Υπάλληλος Υποδοχής) και τρεις κλάσεις (Ασθενείς, Λογαριασμοί, Ραντεβού). Υπάρχουν τέσσερις μέθοδοι που συμβολίζονται από τέσσερις ακμές με φορά προς τα δεξιά, φαίνεται ακόμα ένα μήνυμα επιστροφής (αίτημα Ραντεβού) και η δημιουργία ενός νέου αντικειμένου για την κλάση Ραντεβού (μπλε ορθογώνιο κουτάκι), το οποίο τοποθετείται χαμηλότερα καθώς δεν υφίσταται πριν από τη χρονική στιγμή δημιουργίας του. Επίσης υπάρχει και ένα πλαίσιο επιλογής (με την επισήμανση opt στο πάνω αριστερά τμήμα του), στο οποίο ελέγχεται αν ο ασθενής είναι καταχωρημένος.

4.4 Διαγράμματα Δραστηριοτήτων

Τα διαγράμματα περιπτώσεων χρήσης απεικονίζουν μόνο τις λειτουργίες και τους χειριστές, αλλά δεν απεικονίζουν την ακολουθία των ενεργειών σε κάθε περίπτωση, κάτι που προφανώς είναι απαραίτητο για τη μετέπειτα υλοποίηση του συστήματος. Τα διαγράμματα δραστηριοτήτων (activity diagrams) χρησιμοποιούνται για να καλύψουν αυτήν ακριβώς την ανάγκη, δηλαδή να απεικονίσουν τις ροές εργασίας (workflows) μεταξύ των επιμέρους βημάτων μιας περίπτωσης χρήσης.

Γενικά, χρησιμοποιούνται όταν θέλουμε να μοντελοποιήσουμε δραστηριότητες, διαδικασίες, αλγόριθμους και ενέργειες που αποτελούνται από επιμέρους βήματα. Η μοντελοποίηση δραστηριοτήτων δίνει έμφαση στην απεικόνιση της σειράς των ενεργειών που πρέπει να εκτελεστούν για να πραγματοποιηθεί η δραστηριότητα. Ο Πίνακας 4.2 παρουσιάζει τα βασικά σύμβολα των διαγραμμάτων δραστηριοτήτων.

Σύμβολο	Εξήγηση
	Αρχικός κόμβος (initial node)
	Τελικός κόμβος (final node)
	Κόμβος τελικής ροής (flow final node)
	Δραστηριότητα (activity) ή ενέργεια (action). Η δραστηριότητα μπορεί να περιέχει ενέργειες, ενώ μια ενέργεια δεν κατατέμνεται περαιτέρω.
	Ροή ελέγχου (control flow)

	Κόμβος αντικειμένου (object node)
	Ροή αντικειμένου (object flow)
	Κόμβος απόφασης (decision node)
	Κόμβος συγχώνευσης (merge node)
	Κόμβος διακλάδωσης (fork node)
	Κόμβος συνένωσης (join node)
	Διάδρομος (swimlane). Αποτελεί ειδική περίπτωση διαμερίσματος

Πίνακας 4.2: Συμβολισμός βασικών στοιχείων διαγραμμάτων δραστηριοτήτων.

4.4.1 Ροή εργασιών

Ένα διάγραμμα δραστηριοτήτων συνήθως αναπαριστά μια διαδικασία. Μια δραστηριότητα (activity) αποτελείται από ενέργειες (actions). Κάθε ενέργεια αναπαριστά ένα βήμα της δραστηριότητας που δεν μπορεί να καταταμηθεί περαιτέρω. Ωστόσο, πρέπει να σημειωθεί πως μια ενέργεια μπορεί να είναι η κλήση μιας άλλης δραστηριότητας.

Ένα διάγραμμα έχει συνήθως έναν αρχικό κόμβο (initial node). Στη UML επιτρέπονται και πολλαπλοί αρχικοί κόμβοι, ενώ η έναρξη της δραστηριότητας μπορεί να γίνει και χωρίς να υπάρχει κανένας αρχικός κόμβος. Ωστόσο, στις περισσότερες περιπτώσεις έχουμε μόνο έναν αρχικό κόμβο. Συνεπώς, ο αρχικός κόμβος σηματοδοτεί την έναρξη της δραστηριότητας. Από τον αρχικό κόμβο ξεκινά μία μόνο ροή ελέγχου (control flow), η οποία δείχνει τη σειρά εκτέλεσης των ενεργειών.

Ένα διάγραμμα μπορεί να έχει έναν ή περισσότερους τελικούς κόμβους (final nodes), π.χ. ένα για επιτυχή έκβαση της δραστηριότητας και ένα για αποτυχημένη έκβαση. Μόλις ο έλεγχος φτάσει σε έναν τελικό κόμβο, σταματούν όλες οι τυχόν ροές (flows) και ενέργειες στη δραστηριότητα.

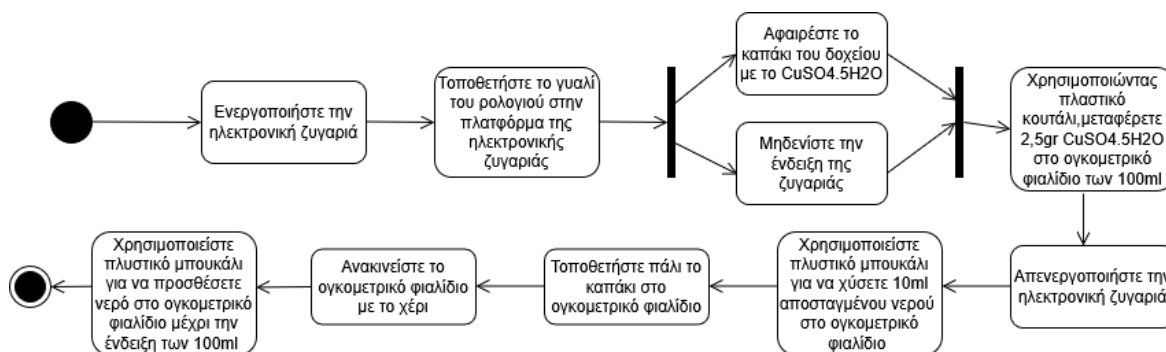
Για να γίνει κατανοητή η σειρά εκτέλεσης των εργασιών σε πολύπλοκα διαγράμματα, η UML έχει εισαγάγει την έννοια του συμβόλου ή νομίσματος ή σκανδάλης (token). Το νόμισμα δείχνει στην ουσία πού βρίσκεται κάθε φορά η εκτέλεση δραστηριότητας. Συνεπώς, όποιο σημείο της δραστηριότητας έχει ένα νόμισμα, αυτό είναι το τρέχον σημείο της δραστηριότητας.

Ο αρχικός κόμβος κάθε δραστηριότητας δημιουργεί ένα νόμισμα το οποίο κινείται μέσω της ροής ελέγχου. Όταν το νόμισμα φτάσει σε κάποια ενέργεια που έχει μια ροή εισόδου, η ενέργεια αυτή μπορεί να ξεκινήσει την εκτέλεση της. Όταν η ενέργεια ολοκληρωθεί, τότε το νόμισμα συνεχίζει στη ροή εξόδου της ενέργειας. Όταν ένα νόμισμα φτάσει στον τελικό κόμβο, τότε καταστρέφεται, και μαζί με αυτό και όλα τα υπόλοιπα νομίσματα που τυχόν κυκλοφορούν στο διάγραμμα. Θεωρούμε πως τα νομίσματα κινούνται με άπειρη ταχύτητα στις ροές ελέγχου, συνεπώς δεν απαιτείται χρόνος για να ταξιδέψει ένα νόμισμα διαμέσου μιας ροής έλεγχου.

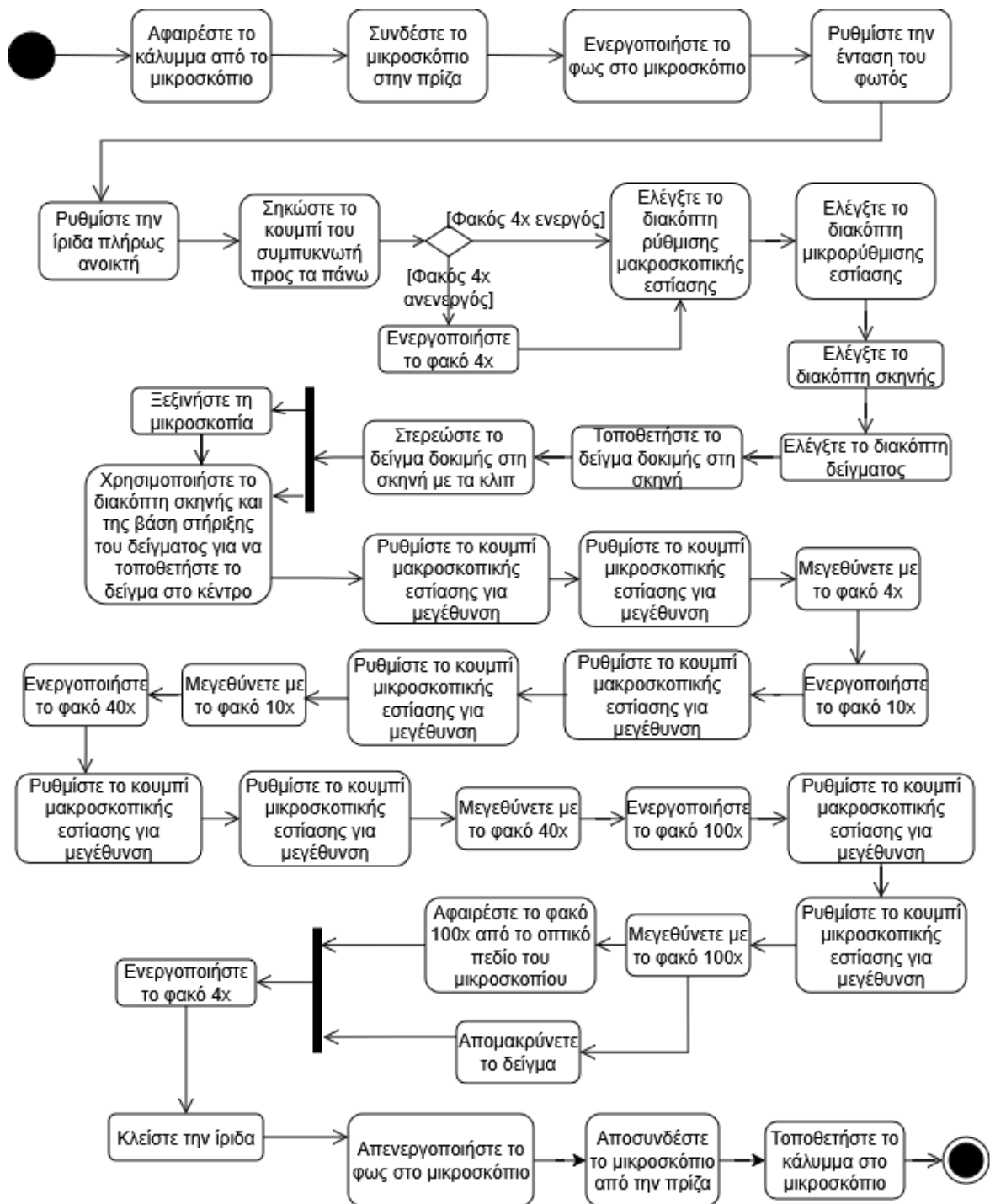
Ένα διαμέρισμα (partition) ομαδοποιεί ενέργειες που έχουν κάποιο κοινό χαρακτηριστικό. Η πιο συνηθισμένη χρήση των διαμερισμάτων είναι οι διάδρομοι (swimlanes). Με τη χρήση των διαδρόμων δείχνουμε ποιος ή τι εκτελεί τις ενέργειες. Αυτοί αναφέρονται συχνά ως ρόλοι (roles).

Στα διαγράμματα δραστηριοτήτων συνήθως πρώτα βρίσκονται οι ρόλοι που εκτελούν τις ενέργειες και σχεδιάζονται ως διάδρομοι. Στη συνέχεια, σχεδιάζουμε τον αρχικό κόμβο (συνήθως πάνω και αριστερά και σε έναν διάδρομο) και, κατόπιν σχεδιάζουμε κάθε ενέργεια μέσα στον κατάλληλο διάδρομο. Ακολούθως σχεδιάζουμε τις ροές ελέγχου που

δείχνουν τη σειρά των ενεργειών και τον τελικό κόμβο (σε ένα διάδρομο πάλι). Στο παρακάτω Σχήμα 14 απεικονίζεται ένα διάγραμμα δραστηριοτήτων που αναπαριστά ένα πείραμα παραγωγής υδατικού διαλύματος για εικονικό εργαστήριο και στο Σχήμα 15 απεικονίζεται πείραμα με χρήση μικροσκοπίου σε εικονικό εργαστήριο.



Σχήμα 14: Διάγραμμα δραστηριοτήτων για εικονικό εργαστήριο.



Σχήμα 15: Διάγραμμα δραστηριοτήτων με μικροσκόπιο για εικονικό εργαστήριο.

4.4.2 Απόφαση και διακλάδωση

Όταν σε κάποια δραστηριότητα χρειάζεται να παρθεί μια απόφαση από την οποία εξαρτώνται οι ενέργειες που θα ακολουθήσουν τότε χρησιμοποιείται ο κόμβος απόφασης

(decision node), στον οποίο σημειώνεται η απόφαση που πρέπει να παρθεί. (π.χ. αν το PIN της κάρτας είναι έγκυρο ή όχι, όπως φαίνεται στο Σχήμα 13). Ο κόμβος απόφασης έχει συνήθως δύο ή περισσότερες εξερχόμενες ροές. Κάθε εξερχόμενη ροή συνοδεύεται από ένα φρουρό (guard), μια λογική έκφραση η οποία δείχνει τη συνθήκη που πρέπει να ισχύει για να επιλεγεί η συγκεκριμένη ροή. Οι τιμές των φρουρών πρέπει να καλύπτουν όλο το φάσμα των πιθανών τιμών και ταυτόχρονα να μην έχουν επικαλύψεις.

Έπειτα από κάθε κόμβο απόφασης, από τις πιθανές εξερχόμενες ροές μόνο μία θα ακολουθηθεί τελικά. Συνεπώς στην έξοδο ενός τέτοιου κόμβου δημιουργείται ένα και μόνο νόμισμα.

Σε κάποιες περιπτώσεις, έπειτα από έναν κόμβο απόφασης και κάποιες ενέργειες που διαφέρουν ανάλογα με τη διακλάδωση που ακολουθεί η ροή ελέγχου, οι ενέργειες γίνονται και πάλι κοινές για όλες τις ροές. Σε αυτή την περίπτωση χρησιμοποιείται ο κόμβος συγχώνευσης (merge mode), ο οποίος έχει πολλές εισόδους και μία μόνο έξοδο.

Κάποιες φορές, έπειτα από μια ενέργεια ξεκινούν δύο ή περισσότερες ενέργειες ανεξάρτητες μεταξύ τους. Στην περίπτωση αυτή χρησιμοποιούμε τον κόμβο διακλάδωσης (fork node). Έπειτα από αυτόν τον κόμβο, η ροή συνεχίζεται σε όλες τις εξερχόμενες ροές. Συνεπώς, σε έναν κόμβο διακλάδωσης δημιουργούνται πολλά νομίσματα και η ροή συνεχίζεται σε πολλές κατευθύνσεις. Επομένως σε ένα διάγραμμα δραστηριοτήτων μπορούμε να έχουμε πάνω από μια ενεργές ροές ανά πάσα στιγμή. Ουσιαστικά, ο κόμβος διακλάδωσης αυξάνει τις ροές του διαγράμματος δημιουργώντας νομίσματα.

Αν θέλουμε να σταματήσουμε μόνο μια ροή σε ένα διάγραμμα, χωρίς να επηρεαστούν οι υπόλοιπες, χρησιμοποιούμε τον κόμβο τελικής ροής (flow final node). Με τη χρήση αυτού του κόμβου καταστρέφονται μόνο τα νομίσματα που εισρέουν στον κόμβο.

Αν έπειτα από έναν κόμβο διακλάδωσης θέλουμε όλες ή κάποιες διακλαδώσεις να συνενωθούν και πάλι, χρησιμοποιούμε έναν κόμβο συνένωσης (join node). Ο κόμβος συνένωσης στην ουσία αποτελεί ένα «σημείο συγχρονισμού», δηλαδή σημείο στο οποίο πρέπει να έχει φτάσει η ροή από όλες τις διακλαδώσεις, ώστε να συνεχιστεί η διαδικασία.

4.5 Διαγράμματα Καταστάσεων

Τα διαγράμματα καταστάσεων ή διαγράμματα μηχανών καταστάσεων (state machine diagrams) δείχνουν τη συμπεριφορά μέρους ενός συστήματος μέσα από τις μεταβάσεις πεπερασμένων καταστάσεων. Χωρίζονται στα διαγράμματα συμπεριφορικών μηχανών καταστάσεων (behavioral state machine diagrams) και στα διαγράμματα μηχανών καταστάσεων πρωτοκόλλου (protocol state machine diagrams). Τα πρώτα χρησιμοποιούνται συνηθέστερα.

Τα διαγράμματα συμπεριφορικών μηχανών καταστάσεων χρησιμοποιούνται συνήθως όταν θέλουμε να κατανοήσουμε πολύπλοκες κλάσεις, δηλαδή κλάσεις των οποίων τα αντικείμενα μπορεί να βρίσκονται σε διαφορετικές καταστάσεις κατά τη διάρκεια της ζωής τους.

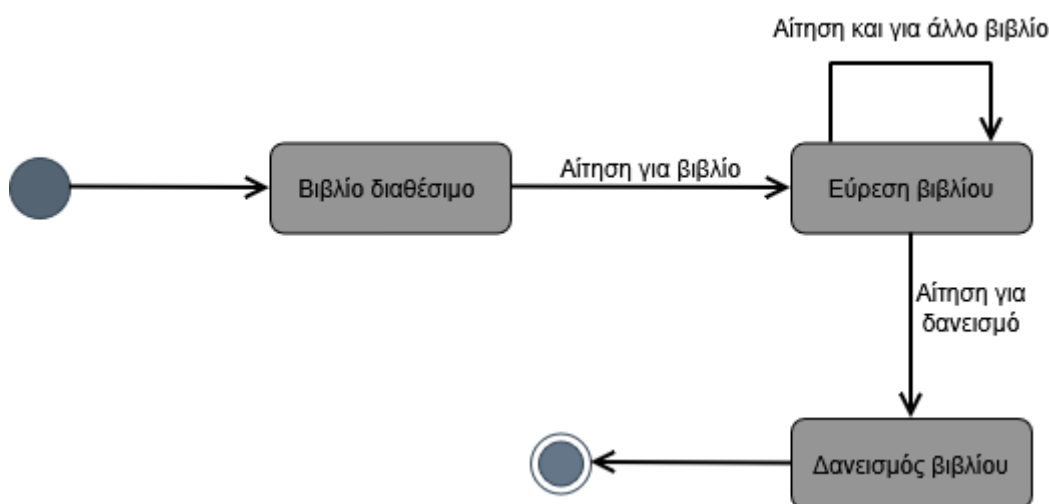
Για παράδειγμα, σε ένα εμπορικό σύστημα, τα αντικείμενα της κλάσης Προϊόν μπορεί να βρίσκονται σε καταστάσεις, όπως υπό παραγγελία, υπό κατασκευή, έτοιμο για παράδοση κ.λ.π. Η μελέτη των καταστάσεων (states) αυτών, καθώς και των συμβάντων (events) που επιτρέπουν τη μετάβαση (transition) από τη μια κατάσταση στην άλλη, μας βοηθά να κατανοήσουμε καλύτερα τη συμπεριφορά του συστήματος. Ο Πίνακας 4.3 παρουσιάζει τα βασικά σύμβολα των διαγραμμάτων συμπεριφορικών μηχανών καταστάσεων.

Σύμβολο	Εξήγηση
	Κατάσταση (state)
	Αρχική κατάσταση (initial state)
	Τελική κατάσταση (final state)



Πίνακας 4.3: Συμβολισμός βασικών στοιχείων διαγραμμάτων συμπεριφορικών μηχανών καταστάσεων.

Ένα διάγραμμα ξεκινάει από μια αρχική κατάσταση. Η μετάβαση μεταξύ καταστάσεων αναπαρίσταται με ένα βέλος. Κάθε μετάβαση συμβαίνει λόγω ενός συμβάντος και/ή κάποιων συνθηκών που μπορεί να ισχύουν. Ένα συμβάν είναι κάτι που συμβαίνει σε ένα αντικείμενο, όπως για παράδειγμα, μια συνθήκη που γίνεται αληθής, η λήψη ενός μηνύματος ή η παρέλευση κάποιου χρονικού διαστήματος. Τα συμβάντα και οι τυχόν συνθήκες φαίνονται πάνω στο βέλος. Στο παρακάτω Σχήμα 16 απεικονίζεται ένα διάγραμμα μηχανής καταστάσεων.



Σχήμα 16: Διάγραμμα μηχανής καταστάσεων.

Στο Σχήμα 16 απεικονίζονται οι καταστάσεις στις οποίες μεταπίπτει το αντικείμενο «Βιβλίο» και παρουσιάζεται πώς τελικά από την κατάσταση «διαθέσιμο» μεταβαίνει στην κατάσταση «δανεισμένο». Βλέπουμε λοιπόν μέσω αυτού του απλού διαγράμματος τα μηνύματα που ουσιαστικά λαμβάνουν χώρα και βοηθούν στην καταγραφή της συμπεριφοράς του αντικειμένου. Προφανώς αυτά τα διαγράμματα βοηθούν τους πελάτες να καταλάβουν ποια είναι η ακολουθία γεγονότων που λαμβάνει χώρα, αλλά κατά κύριο

λόγο βοηθούν τους προγραμματιστές στην κατανόηση των λειτουργιών που θα υλοποιηθούν.

Η μηχανή καταστάσεων συμπεριφοράς είναι η εξειδίκευση της συμπεριφοράς και χρησιμοποιείται για τον καθορισμό διακριτής συμπεριφοράς ενός μέρους του σχεδιασμένου συστήματος μέσω πεπερασμένων μεταβάσεων κατάστασης.

Η συμπεριφορά μοντελοποιείται ως διάσχιση ενός γραφήματος κόμβων κατάστασης που συνδέονται με μεταβάσεις. Οι μεταβάσεις ενεργοποιούνται από την αποστολή μιας σειράς συμβάντων. Κατά τη διάρκεια της διάσχισης, η μηχανή καταστάσεων θα μπορούσε επίσης να εκτελέσει ορισμένες δραστηριότητες.

Η μηχανή καταστάσεων συμπεριφοράς θα μπορούσε να ανήκει σε έναν ταξινομητή συμπεριφοράς που ονομάζεται περιβάλλον της. Το περιβάλλον καθορίζει ποια σήματα και ενεργοποιητές κλήσης ορίζονται για αυτήν τη μηχανή κατάστασης και ποια χαρακτηριστικά και λειτουργίες είναι διαθέσιμα στις δραστηριότητες της μηχανής κατάστασης. Οι ενεργοποιητές σήματος και οι ενεργοποιητές κλήσης για τη μηχανή κατάστασης ορίζονται σύμφωνα με τις λήψεις και τις λειτουργίες αυτού του ταξινομητή.

Η μηχανή κατάστασης μπορεί να έχει ένα συσχετισμένο χαρακτηριστικό συμπεριφοράς (προδιαγραφή) και να είναι η μέθοδος αυτού του χαρακτηριστικού συμπεριφοράς. Σε αυτήν την περίπτωση, η μηχανή κατάστασης καθορίζει τη συμπεριφορά αυτού του χαρακτηριστικού συμπεριφοράς. Οι παράμετροι της μηχανής κατάστασης ταιριάζουν με τις παραμέτρους του χαρακτηριστικού συμπεριφοράς και παρέχουν τα μέσα για την πρόσβαση στις παραμέτρους χαρακτηριστικών συμπεριφοράς εντός της μηχανής κατάστασης.

Η ομάδα συμβάντων για τη μηχανή κατάστασης είναι η ομάδα συμβάντων της παρουσίας σύμφωνα με τον ταξινομητή περιβάλλοντος με συμπεριφορά ή τον ταξινομητή που κατέχει το χαρακτηριστικό συμπεριφοράς για το οποίο η μηχανή κατάστασης είναι μια μέθοδος.

Ο ταξινομητής περιβάλλοντος της μηχανής κατάστασης μεθόδου ενός χαρακτηριστικού συμπεριφοράς πρέπει να είναι ο ταξινομητής που κατέχει το χαρακτηριστικό συμπεριφοράς. Μια μηχανή κατάστασης χωρίς ταξινομητή περιβάλλοντος μπορεί να χρησιμοποιεί εναύσματα που είναι ανεξάρτητα από τις λήψεις ή τις λειτουργίες ενός

ταξινομητή, δηλαδή, είτε απλώς εναύσματα σήματος είτε εναύσματα κλήσης με βάση τις παραμέτρους του προτύπου λειτουργίας της (παραμετροποιημένης) μηχανής κατάστασης.

4.6 Διαγράμματα Συστατικών Στοιχείων

Ο κύριος σκοπός του διαγράμματος συστατικών στοιχείων (component diagram) είναι να απεικονίσει τις φυσικές (δομικές) σχέσεις μεταξύ των στοιχείων του λογισμικού. Στη UML 2.x τα συστατικά στοιχεία θεωρούνται αυτόνομες, ενθυλακωμένες μονάδες μέσα σε ένα σύστημα ή υποσύστημα τα οποία παρέχουν μία ή περισσότερες διεπαφές (interfaces). Κάθε συστατικό πρέπει να είναι ένα επαναχρησιμοποιήσιμο τμήμα λογισμικού και ανεξάρτητο ώστε να μπορεί να αντικατασταθεί, αν χρειαστεί, σε μελλοντική έκδοση, διότι ένα συστατικό ενθυλακώνει τη συμπεριφορά και υλοποιεί συγκεκριμένες διεπαφές. Τα συστατικά μπορεί να είναι και ήδη έτοιμα πακέτα (δηλαδή, εκτελέσιμα και αυτοτελή τμήματα κώδικα) που θα ενσωματωθούν στο σύστημα που αναπτύσσουμε, π.χ., χρήση ενός πακέτου java που παρέχει λειτουργίες σύνδεσης με βάση δεδομένων μέσω διαδικτύου.

Τα διαγράμματα συστατικών στοιχείων μας βοηθούν να αναπαραστήσουμε τα συστατικά λογισμικού και τις εξαρτήσεις τους, αποκρύπτοντας ουσιαστικά τις λεπτομέρειες υλοποίησης. Μπορούμε να απεικονίσουμε γραφικά τα αρχεία υλοποίησης του συστήματος, δηλαδή τα εκτελέσιμα αρχεία που υλοποιούν τις κλάσεις που έχουν μοντελοποιηθεί στο σύστημα, τα αρχεία πηγαίου κώδικα, έτοιμες βιβλιοθήκες που χρησιμοποιούμε κ.λ.π.

Τα διαγράμματα συστατικών στοιχείων προσφέρουν στους σχεδιαστές λογισμικού μια φυσική μορφή για να ξεκινήσουν τη μοντελοποίηση μιας λύσης. Επιτρέπουν στον σχεδιαστή να επαληθεύσει ότι η απαιτούμενη λειτουργικότητα ενός συστήματος υλοποιείται από τα στοιχεία, διασφαλίζοντας έτσι το τελικό σύστημα.

Επιπλέον, τα διαγράμματα συστατικών στοιχείων αποτελούν χρήσιμα εργαλεία επικοινωνίας για διάφορες ομάδες. Τα διαγράμματα μπορούν να παρουσιαστούν σε

βασικούς ενδιαφερόμενους φορείς του έργου και στο προσωπικό υλοποίησης και βοηθούν σε μια πρόωπη κατανόηση του υπό κατασκευή συστήματος.

Οι προγραμματιστές βρίσκουν τα διαγράμματα αυτά χρήσιμα επειδή τους παρέχουν μια υψηλού επιπέδου αρχιτεκτονική εικόνα του συστήματος που θα δημιουργήσουν, η οποία τους βοηθά να ξεκινήσουν την τυποποίηση ενός οδικού χάρτη για την υλοποίηση και να λαμβάνουν αποφάσεις σχετικά με τις αναθέσεις εργασιών ή/και τις απαραίτητες βελτιώσεις δεξιοτήτων.

Οι διαχειριστές συστημάτων βρίσκουν τα διαγράμματα αυτά χρήσιμα επειδή λαμβάνουν μια αρχική εικόνα των λογικών στοιχείων λογισμικού που θα εκτελούνται στα συστήματά τους. Παρόλο που οι διαχειριστές αυτοί δεν θα είναι σε θέση να αναγνωρίσουν τους φυσικούς υπολογιστές ή τα φυσικά εκτελέσιμα αρχεία από το διάγραμμα, ένα τέτοιο διάγραμμα θα είναι ευπρόσδεκτο επειδή παρέχει πρώιμες πληροφορίες σχετικά με τα στοιχεία και τις σχέσεις τους.

Το διάγραμμα στοιχείων είναι ένα πολύ σημαντικό διάγραμμα που οι αρχιτέκτονες συχνά δημιουργούν νωρίς σε ένα έργο. Ωστόσο, η χρησιμότητα του διαγράμματος στοιχείων εκτείνεται σε όλη τη διάρκεια ζωής του συστήματος. Είναι ανεκτίμητα επειδή μοντελοποιούν και τεκμηριώνουν την αρχιτεκτονική ενός συστήματος. Συνεπώς, οι προγραμματιστές και οι τελικοί διαχειριστές συστήματος του συστήματος θεωρούν ότι η υλοποίηση αυτών των διαγραμμάτων είναι μια εργασία κρίσιμη για το προϊόν, καθώς τους βοηθά να κατανοήσουν το σύστημα.

4.7 Συγκριτική θεώρηση διαγραμμάτων

Είδαμε ότι η UML περιλαμβάνει μια ποικιλία διαγραμμάτων που αναπαριστούν διαφορετικές πτυχές ενός συστήματος λογισμικού. Κάθε τύπος διαγράμματος εξυπηρετεί έναν μοναδικό σκοπό, αντιμετωπίζοντας διαφορετικές πτυχές του σχεδιασμού, της λειτουργικότητας και της συμπεριφοράς του συστήματος.

Τα διαγράμματα κλάσης είναι απαραίτητα για τον ορισμό της υποκείμενης δομής και των μοντέλων δεδομένων ενός συστήματος. Χρησιμεύουν ως βάση για τον αντικειμενοστρεφή

σχεδιασμό και την κωδικοποίηση, παρέχοντας μια σαφή δομή για το πώς τα αντικείμενα και οι σχέσεις τους θα αναπαρασταθούν στη βάση κώδικα του συστήματος. Είναι πολύτιμα για τους προγραμματιστές βοηθώντας τη διατήρηση της συνέπειας στο σχεδιασμό του συστήματος, διασφαλίζοντας την ακεραιότητα και τη συνοχή των μοντέλων δεδομένων σε ολόκληρη την εφαρμογή.

Τα διαγράμματα περιπτώσεων χρήσης είναι ιδανικά για την καταγραφή απαιτήσεων υψηλού επιπέδου και την παροχή μιας εικόνας του συστήματος με επίκεντρο τον χρήστη. Βοηθούν τα ενδιαφερόμενα μέρη, συμπεριλαμβανομένων των πελατών και των επιχειρηματικών αναλυτών, να κατανοήσουν τις λειτουργίες, τις αλληλεπιδράσεις και τους στόχους του συστήματος. Αυτά τα διαγράμματα είναι απαραίτητα στα αρχικά στάδια του σχεδιασμού του συστήματος, για να διασφαλιστεί ότι το σύστημα θα καλύπτει τις ανάγκες των χρηστών του και για να καθοριστεί το εύρος του έργου.

Τα διαγράμματα ακολουθίας εστιάζουν στη δυναμική συμπεριφορά ενός συστήματος, δείχνοντας την ακολουθία αλληλεπιδράσεων μεταξύ αντικειμένων. Είναι ιδιαίτερα χρήσιμα για τους προγραμματιστές για να μοντελοποιήσουν τη ροή δεδομένων και τον έλεγχο μεταξύ των στοιχείων του συστήματος κατά τη διάρκεια συγκεκριμένων περιπτώσεων χρήσης ή σεναρίων. Επίσης βοηθούν στην αποσαφήνιση του χρονισμού των λειτουργιών, καθιστώντας τα κρίσιμα για την κατανόηση του τρόπου με τον οποίο επικοινωνούν τα διαφορετικά μέρη του συστήματος και διασφαλίζοντας ότι η λογική πίσω από αυτές τις αλληλεπιδράσεις είναι σωστή και αποτελεσματική.

Τα διαγράμματα δραστηριοτήτων βοηθούν στην αποσαφήνιση της συμπεριφοράς του συστήματος όσον αφορά τις ροές εργασίας, βοηθώντας στον εντοπισμό ανεπαρκειών ή πιθανών βελτιώσεων στις διαδικασίες. Χρησιμοποιούνται συχνά για την αναπαράσταση επιχειρηματικών διαδικασιών, λειτουργιών συστήματος ή αλληλεπιδράσεων πολλαπλών βημάτων, παρέχοντας πληροφορίες τόσο για τη λογική του συστήματος όσο και για την εμπειρία χρήστη.

Τα διαγράμματα μηχανών καταστάσεων απεικονίζουν διαφορετικές καταστάσεις ενός αντικειμένου και τις μεταβάσεις μεταξύ αυτών των καταστάσεων, χρησιμεύοντας στην κατανόηση της δυναμικής συμπεριφοράς.

Τα διαγράμματα συστατικών στοιχείων εξηγούν τα στοιχεία που είναι απαραίτητα για την εκτέλεση λειτουργιών λογισμικού. Εμφανίζουν την οργάνωση και τις εξαρτήσεις μεταξύ

ενός συνόλου στοιχείων, διευκολύνοντας την κατανόηση της αρχιτεκτονικής του συστήματος.

Συνοψίζοντας:

- **Διάγραμμα κλάσεων:** δείχνει κλάσεις, ιδιότητες, λειτουργίες και σχέσεις. Χρησιμοποιείται για τη λεπτομερή περιγραφή της δομής ενός συστήματος. Δεν χρησιμοποιείται όταν θέλουμε να δείξουμε ροή ελέγχου ή ακολουθία αλληλεπιδράσεων.
- **Διάγραμμα περιπτώσεων χρήσης:** απεικονίζει τις αλληλεπιδράσεις των χρηστών και τη λειτουργικότητα του συστήματος μέσω των actors και των περιπτώσεων χρήσης, καθιστώντας σαφές τι κάνει το σύστημα από την οπτική γωνία του χρήστη. Δεν χρησιμοποιείται για ροές εργασίας διεπαφής χρήστη, για λεπτομερείς λειτουργικές απαιτήσεις και τεχνικές λεπτομέρειες σχεδιασμού.
- **Διάγραμμα ακολουθίας:** χρησιμοποιείται για καταγραφή βήμα προς βήμα της αλληλεπίδρασης των αντικειμένων μέσω της λεπτομερούς καταγραφής της ακολουθίας ανταλλαγής μηνυμάτων. Δεν χρησιμοποιείται για στατική δομική προβολή, για περιγραφή ταυτόχρονων εσωτερικών αλλαγών κατάστασης.
- **Διάγραμμα δραστηριοτήτων:** αντιπροσωπεύει ροές εργασίας ή διαδικασίες, περιγράφοντας λεπτομερώς τη ροή ελέγχου ή δεδομένων. Χρήσιμο για τη μοντελοποίηση αλγορίθμων και ροών διεργασιών. Δεν χρησιμοποιείται για στατικές σχέσεις μεταξύ στοιχείων και για λεπτομερή σειρά ανταλλαγής μηνυμάτων μεταξύ αντικειμένων.
- **Διάγραμμα μηχανών καταστάσεων:** απεικονίζει τις διαφορετικές καταστάσεις ενός αντικειμένου και τις μεταβάσεις μεταξύ αυτών των καταστάσεων, κάτι χρήσιμο για την κατανόηση της δυναμικής συμπεριφοράς. Δεν χρησιμοποιείται για την αναπαράσταση της συνολικής αρχιτεκτονικής και για ροές διεργασιών υψηλού επιπέδου.
- **Διάγραμμα συστατικών στοιχείων:** απεικονίζει την οργάνωση και τις εξαρτήσεις μεταξύ ενός συνόλου στοιχείων, διευκολύνοντας την κατανόηση της αρχιτεκτονικής του συστήματος. Δεν χρησιμοποιείται όταν θέλουμε λεπτομέρεια σε επίπεδο κλάσης ή αλληλεπίδρασης αντικειμένων και για χρονική συμπεριφορά ή ακολουθία μηνυμάτων.

5. Σχέσεις και συνέργειες μεταξύ των διαγραμμάτων UML

Τα διαγράμματα UML δεν είναι αυτόνομα αλλά λειτουργούν συνεργατικά για να περιγράψουν ένα σύστημα από διαφορετικές αλλά συνδεδεμένες απόψεις.

5.1 Η λογική ροή από τις απαιτήσεις στη σχεδίαση

Η διαδικασία μετατροπής των απαιτήσεων σε σχεδιασμό είναι δομημένη και διασφαλίζει ότι το τελικό προϊόν ανταποκρίνεται στις ανάγκες των χρηστών και στους στόχους του έργου. Αυτή η ροή μπορεί να αναλυθεί σε διάφορα βασικά στάδια.

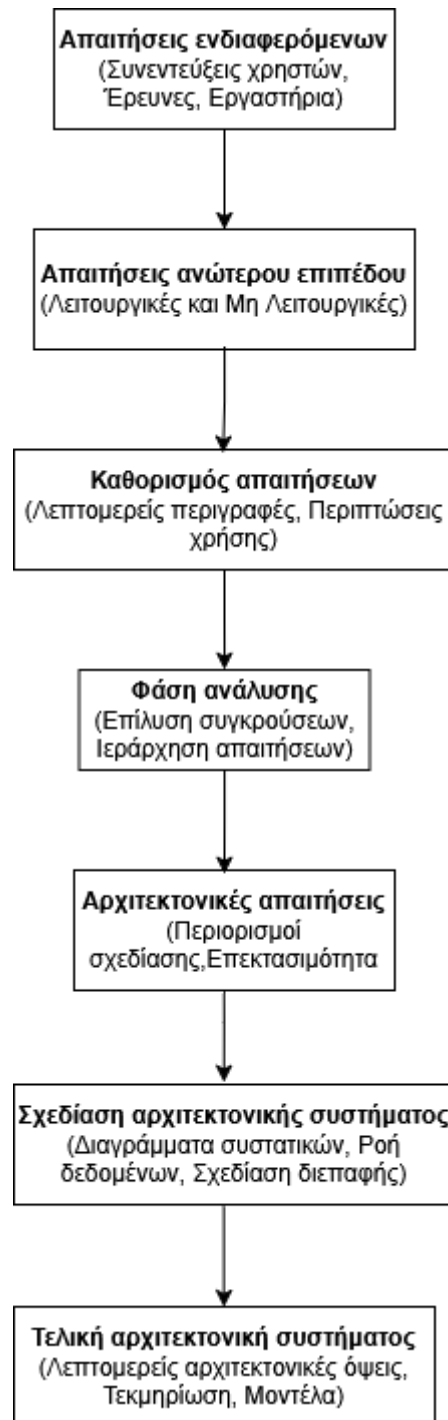
1. Συγκέντρωση απαιτήσεων: τα ενδιαφερόμενα μέρη συνεργάζονται για την κατανόηση των αναγκών και τεκμηριώνουν με σαφήνεια τις λειτουργικές και μη λειτουργικές απαιτήσεις. Αυτές κατατάσσονται με βάση τη σημασία και τη σκοπιμότητα.
2. Ανάλυση απαιτήσεων: γίνεται αξιολόγηση της τεχνικής και οικονομικής σκοπιμότητας, διευκρίνιση των απαιτήσεων επανεξετάζοντας ασαφείς ή αντικρουόμενες απαιτήσεις και δημιουργία περιπτώσεων χρήσης για να δειχθεί πώς οι χρήστες θα αλληλεπιδρούν με το σύστημα.
3. Προδιαγραφές: μέσω του εγγράφου προδιαγραφών απαιτήσεων διατυπώνονται λεπτομερώς οι προδιαγραφές (συμπεριλαμβανομένων διαγραμμάτων) και επικυρώνονται οι απαιτήσεις σε συμφωνία με τα ενδιαφερόμενα μέρη για διασφάλιση της ακρίβειας.
4. Σχεδιασμός υψηλού επιπέδου: ορίζεται η συνολική δομή και οι αλληλεπιδράσεις εντός του συστήματος, το σύστημα αναλύεται σε διαχειρίσιμα στοιχεία/μονάδες και αποφασίζεται ποιες τεχνολογίες και πλατφόρμες θα χρησιμοποιηθούν για την υλοποίηση.
5. Λεπτομερής σχεδιασμός: δημιουργούνται λεπτομερή σχέδια για κάθε στοιχείο, συμπεριλαμβανομένων των δομών δεδομένων και των αλγορίθμων που θα χρησιμοποιηθούν. Επίσης γίνεται ο ορισμός διεπαφών χρήστη και προγραμματισμού εφαρμογών (Application Program Interface - API)

διασφαλίζοντας την χρηστικότητα και την προσβασιμότητα και ενσωματώνονται οι αρχές σχεδιασμού που αφορούν την ασφάλεια, την προστασία και τη συμμόρφωση με τα πρότυπα.

6. Επικύρωση σχεδιασμού: διεξάγονται αξιολογήσεις σχεδιασμού με τα ενδιαφερόμενα μέρη για τη συλλογή σχολίων, γίνεται ανάπτυξη πρωτοτύπων για την οπτικοποίηση στοιχείων του σχεδιασμού και τέλος πραγματοποιούνται οι απαραίτητες αναθεωρήσεις με βάση τα σχόλια και τις δοκιμές επικύρωσης.

Συνοψίζοντας, αυτή η διαδικασία τονίζει τη σημασία της αλληλεπίδρασης με τα ενδιαφερόμενα μέρη, της επαναληπτικής επικύρωσης και της σχολαστικής τεκμηρίωσης σε κάθε στάδιο. Κάθε βήμα βασίζεται στο προηγούμενο, καταλήγοντας τελικά σε ένα σχεδιασμό που ευθυγραμμίζεται αποτελεσματικά με τις αρχικά καθορισμένες απαιτήσεις.

Η διατήρηση μιας προσέγγισης με επίκεντρο το χρήστη σε όλη τη διάρκεια διασφαλίζει ότι ο τελικός σχεδιασμός όχι μόνο πληροί τις τεχνικές προδιαγραφές αλλά και βελτιώνει την εμπειρία του χρήστη. Στο σχήμα 17 απεικονίζεται ένα διάγραμμα που απεικονίζει τη διαδικασία μετάβασης από τις απαιτήσεις σε αρχιτεκτονικό σχεδιασμό συστήματος.



Σχήμα 17: Μετάβαση από απαιτήσεις σε αρχιτεκτονικό σχεδιασμό.

5.2 Μετάβαση από Use Case σε Class και Sequence Diagrams

Σε ένα αντικειμενοστρεφές σύστημα η λειτουργικότητα επιτυγχάνεται μέσω της αλληλεπίδρασης των αντικειμένων κατά τη διάρκεια της εκτέλεσης. Τα αντικείμενα αλληλεπιδρούν μέσω αποστολής μηνυμάτων, δηλαδή μέσω της κλήσης μεθόδων, είτε για την εκτέλεση κάποιας υπηρεσίας που παρέχει ένα αντικείμενο, είτε για τη λήψη δεδομένων.

Ένας από τους βασικότερους στόχους στη διαδικασία της σχεδίασης είναι η μετατροπή της περιγραφής των λειτουργικών απαιτήσεων σε ακολουθίες αλληλεπιδράσεων μεταξύ των εμπλεκόμενων οντοτήτων/αντικειμένων.

Το διάγραμμα κλάσεων δείχνει τη στατική δομή ενός συστήματος, ενώ το διάγραμμα ακολουθίας δείχνει τη δυναμική συμπεριφορά του συστήματος. Ένα διάγραμμα ακολουθίας μπορεί να είναι τόσο αναλυτικό και λεπτομερές, ώστε σε συνδυασμό με το αντίστοιχο διάγραμμα κλάσεων να παρέχουν όλη την απαραίτητη πληροφορία που χρειάζεται ένας προγραμματιστής για να προβεί στην πλήρη υλοποίηση του κώδικα.

Αρχικά γίνεται η εξαγωγή των βασικών στοιχείων που είναι κοινή διαδικασία και για τα δύο είδη διαγραμμάτων (κλάσης και ακολουθίας). Οι χειριστές (actors) θα είναι πιθανά αντικείμενα ή εξωτερικές οντότητες. Τα ουσιαστικά στις περιγραφές περιπτώσεων χρήσης είναι υποψήφιες κλάσεις (π.χ. Παραγγελία, Προϊόν). Για κάθε δράση (action) που περιγράφει μια περίπτωση χρήσης, αναθέτουμε μια μέθοδο στην κλάση που κατέχει το ουσιαστικό («Πρόσθεσε Παραγγελία», σημαίνει τη μέθοδο «Δημιουργία Παραγγελίας»). Τέλος, εντοπίζουμε τα όρια του συστήματος: οτιδήποτε βρίσκεται εκτός συστήματος (εξωτερικές υπηρεσίες, εφαρμογές τρίτων).

5.2.1 Για τα διαγράμματα κλάσεων

Αφού έχει γίνει η εξαγωγή των βασικών στοιχείων, μετά εντοπίζονται τα χαρακτηριστικά (attributes) των κλάσεων (π.χ. ο «κωδικός» του προϊόντος, η «ημερομηνία» της παραγγελίας) και καθορίζεται η ορατότητα (visibility) των χαρακτηριστικών και των μεθόδων, εφόσον η περίπτωση χρήσης ορίζει δημόσια (public) ή ιδιωτική (private) λειτουργία.

Στη συνέχεια καθορίζονται οι σχέσεις (π.χ. εξάρτησης, σύνθεσης, συνάθροισης, γενίκευσης) μεταξύ των κλάσεων με τις κατάλληλες πολλαπλότητες (π.χ. ένα προϊόν μπορεί να συνδέεται με καμία έως πολλές παραγγελίες). Τέλος γίνεται συγχώνευση των διπλότυπων κλάσεων που εμφανίζονται σε αρκετές περιπτώσεις χρήσης και δημιουργούνται υπερκλάσεις όταν διάφορες κλάσεις μοιράζονται ίδια χαρακτηριστικά και ίδιες μεθόδους (π.χ. ο «χρήστης» γίνεται υπερκλάση για τις κλάσεις «πελάτης» και «διαχειριστής συστήματος»).

5.2.2 Για τα διαγράμματα ακολουθίας

Επιλέγουμε μια συγκεκριμένη περίπτωση χρήσης από το διάγραμμα περιπτώσεων χρήσης (π.χ. «Τοποθέτηση Παραγγελίας»). Εντοπίζουμε τους χειριστές (π.χ. «Πελάτης») που εμπλέκονται στη συγκεκριμένη περίπτωση χρήσης, τα αντικείμενα που συμμετέχουν στην αλληλεπίδραση (π.χ. «Παραγγελία», «Καλάθι Αγορών») και κάθε χειριστής και αντικείμενο θα έχει τη δική του (κατακόρυφη) γραμμή ζωής.

Για κάθε βήμα στην αφήγηση της περίπτωσης χρήσης φτιάχνουμε ένα μήνυμα που ανταλλάσσεται μεταξύ των χειριστών και των συστατικών του συστήματος, με τη σειρά που αυτά συμβαίνουν. Αυτό το μήνυμα θα είναι ένα (οριζόντιο) βέλος μηνύματος από τη γραμμή ζωής του χειριστή προς τη γραμμή ζωής του αντικειμένου (ή και αντίστροφα). Αυτό το μήνυμα φέρει την περιγραφή της λειτουργίας (μέθοδος) που επιτελείται.

Ολοκληρώνοντας, να σημειωθεί ότι πρέπει να υπάρχει συνέπεια στην ονοματοδοσία (συνταύτιση στα ονόματα των λειτουργιών και στα ονόματα των κλάσεων) ανάμεσα στα δύο διαγράμματα και επαλήθευση ότι κάθε βήμα στην επιλεγμένη περίπτωση χρήσης εμφανίζεται ως μήνυμα στο διάγραμμα ακολουθίας και κάθε μήνυμα αντιστοιχεί σε μία μέθοδο στο διάγραμμα κλάσης.

5.3 Συνδυαστική χρήση των Activity και State Diagrams

Τα διαγράμματα δραστηριοτήτων και κατάστασης φαίνονται παρόμοια εκ πρώτης όψεως, αλλά ξεχωρίζουν σε διαφορετικά σημεία. Στη UML, ένας συνδυασμός διαγραμμάτων

κατάστασης και δραστηριοτήτων δεν είναι ένας ενιαίος επίσημος τύπος διαγράμματος, αλλά μια πολύ κοινή πρακτική μοντελοποίησης όπου χρησιμοποιούνται μαζί για να περιγραφούν διαφορετικές πτυχές του ίδιου συστήματος ή στοιχείου.

Συνήθως χρησιμοποιούνται συνδυαστικά όταν:

- το σύστημα έχει διακριτές καταστάσεις και πολύπλοκες ροές διαδικασίας.
- Ένα ή περισσότερα αντικείμενα έχουν μεγάλο κύκλο ζωής.
- Θέλουμε να καταγράψουμε τη συμπεριφορά του συστήματος για τα διάφορα ενδιαφερόμενα μέρη (προγραμματιστές, αναλυτές, εκπαιδευτές).

Ένα διάγραμμα καταστάσεων απεικονίζει τις καταστάσεις στις οποίες βρίσκεται το σύστημα και κάθε κατάσταση μπορεί να επεκταθεί μέσω του διαγράμματος δραστηριοτήτων το οποίο θα απεικονίζει τι συμβαίνει μέσα σε αυτήν την κατάσταση.

Για παράδειγμα ένα διάγραμμα καταστάσεων μπορεί να μας δείξει τις καταστάσεις του αντικειμένου «Παραγγελία» στη διαδικασία μιας online παραγγελίας:

1. «Δημιουργία Παραγγελίας»
2. «Επεξεργασία Παραγγελίας»
3. «Παράδοση Παραγγελίας»
4. «Κλείσιμο Παραγγελίας».

Ένα διάγραμμα δραστηριοτήτων μπορεί να μας δείξει τι συμβαίνει μέσα στην κατάσταση π.χ. «Επεξεργασία Παραγγελίας» με τα εξής βήματα:

1. «Έλεγχος Αποθέματος»
2. «Κράτηση Προϊόντων»
3. «Δημιουργία Τιμολογίου»
4. «Ειδοποίηση Πελάτη».

Το ερώτημα για το ποιο διάγραμμα κατασκευάζουμε πρώτα έχει να κάνει με το στόχο μας και σε ποιο στάδιο της σχεδίασης βρισκόμαστε. Ξεκινάμε πρώτα με το διάγραμμα δραστηριοτήτων στην περίπτωση που θέλουμε πρώτα να κατανοηθεί η συνολική ροή λειτουργίας ή η λογική της διαδικασίας. Μόλις η ροή λειτουργίας γίνει σαφής, μεταβαίνουμε στο διάγραμμα καταστάσεων για να μοντελοποιήσουμε τον τρόπο με τον οποίο συμπεριφέρονται τα μεμονωμένα αντικείμενα μέσα σε αυτήν τη διαδικασία,

δείχνοντας πώς αυτά αλλάζουν καταστάσεις ανταποκρινόμενα σε συγκεκριμένα συμβάντα ή συνθήκες.

Τις περισσότερες φορές οι σχεδιαστές ξεκινούν με το διάγραμμα δραστηριοτήτων για να συλλάβουν τη μεγάλη εικόνα του συστήματος και έπειτα φιλτράρουν τη συμπεριφορά του μέσω του διαγράμματος καταστάσεων για βαθύτερη λογική, βασισμένη σε γεγονότα.

5.4 Πώς τα Component Diagrams συνθέτουν την αρχιτεκτονική

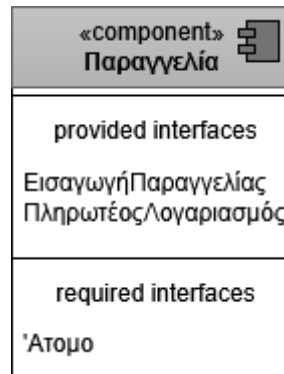
Η σχεδίαση ενός στοιχείου στην UML 2.x είναι πλέον πολύ παρόμοια με τη σχεδίαση μιας κλάσης σε ένα διάγραμμα κλάσης. Στην πραγματικότητα, στην UML 2.x ένα στοιχείο είναι απλώς μια εξειδικευμένη έκδοση της έννοιας της κλάσης.

Ένα στοιχείο σχεδιάζεται ως ορθογώνιο με προαιρετικά διαμερίσματα στοιβαγμένα κάθετα. Μια υψηλού επιπέδου, αφηρημένη προβολή ενός στοιχείου μπορεί να μοντελοποιηθεί απλώς ως ένα ορθογώνιο με το όνομα του στοιχείου και το κείμενο ή/και το εικονίδιο του στερεότυπου του στοιχείου. Το Σχήμα 18 δείχνει δύο διαφορετικούς τρόπους απεικόνισης για ένα στοιχείο χρησιμοποιώντας την προδιαγραφή UML 2.x.



Σχήμα 18: Διαφορετικοί τρόποι απεικόνισης διαμερίσματος ονόματος συστατικού στοιχείου.

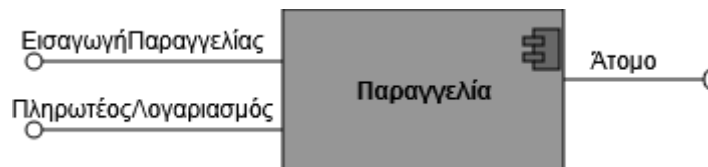
Ένα στοιχείο είναι μια αυτόνομη μονάδα που παρέχει μία ή περισσότερες δημόσιες διεπαφές. Το Σχήμα 19 δείχνει το στοιχείο «Παραγγελία» να έχει ένα δεύτερο διαμέρισμα που υποδηλώνει ποιες διεπαφές παρέχει (provided interfaces) και απαιτεί (required interfaces) το στοιχείο «Παραγγελία».



Σχήμα 19: Πρόσθετο διαμέρισμα ονόματος συστατικού στοιχείου που δείχνει ποιες διεπαφές παρέχονται/απαιτούνται.

Όπως βλέπουμε στο Σχήμα 19, το στοιχείο παρέχει τις διεπαφές «ΕισαγωγήΠαραγγελίας» και «ΠληρωτέοςΛογαριασμός». Επιπλέον απαιτεί ένα άλλο στοιχείο που παρέχει τη διεπαφή «Άτομο».

Η UML 2.x έχει επίσης εισαγάγει έναν άλλο τρόπο για την εμφάνιση των παρεχόμενων και απαιτούμενων διεπαφών ενός στοιχείου. Αυτός ο δεύτερος τρόπος βασίζεται στο ενιαίο ορθογώνιο, με το όνομα του στοιχείου σε αυτό, και τοποθετεί αυτά που η προδιαγραφή UML 2.x αποκαλεί «σύμβολα διεπαφής» συνδεδεμένα με το εξωτερικό του ορθογώνιου. Αυτή η δεύτερη προσέγγιση απεικονίζεται στο Σχήμα 20.

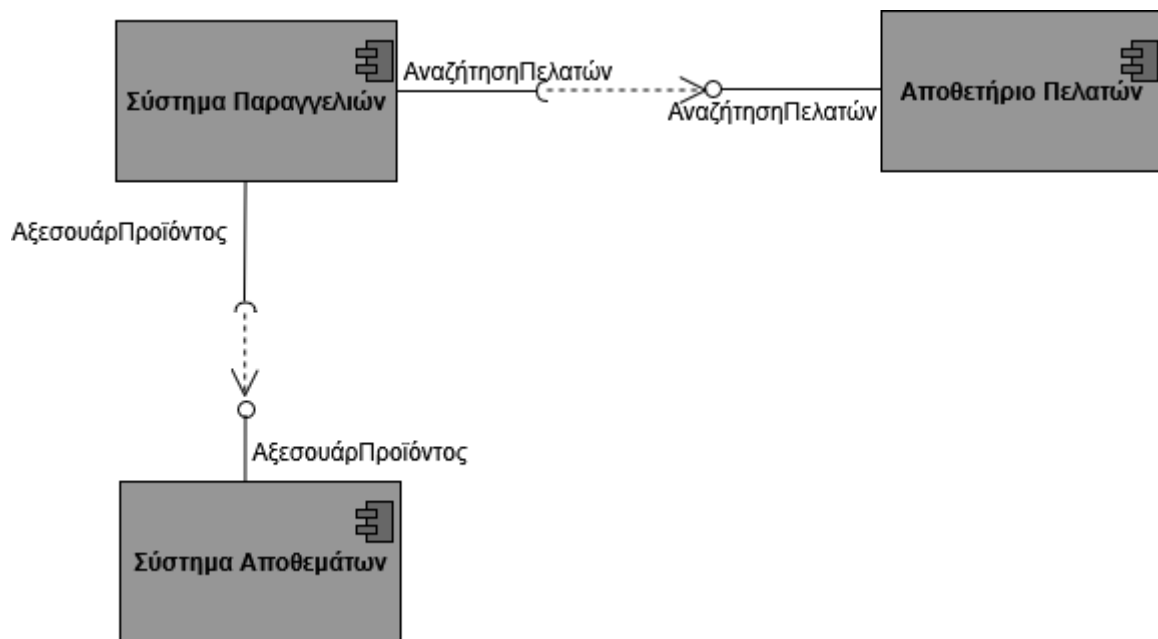


Σχήμα 20: Εναλλακτική προσέγγιση εμφάνισης συστατικού στοιχείου που δείχνει ποιες διεπαφές παρέχονται/απαιτούνται.

Σε αυτήν τη δεύτερη προσέγγιση, τα σύμβολα διεπαφής με έναν πλήρη κύκλο στο άκρο τους αντιπροσωπεύουν μια διεπαφή που παρέχει το στοιχείο. Αυτό το σύμβολο "γλειφιτζούρι" είναι η συντομογραφία μιας σχέσης υλοποίησης ενός ταξινομητή διεπαφής. Τα σύμβολα διεπαφής με μόνο ένα ημικύκλιο στο άκρο τους (γνωστά και ως sockets) αντιπροσωπεύουν μια διεπαφή που απαιτεί το στοιχείο. Παρόλο που το Σχήμα 20 φαίνεται πολύ διαφορετικό από το Σχήμα 19, και τα δύο σχήματα παρέχουν τις ίδιες πληροφορίες,

δηλαδή το στοιχείο «Παραγγελία» παρέχει δύο διεπαφές, τις «ΕισαγωγήΠαραγγελίας» και «ΠληρωτέοςΛογαριασμός», και το ίδιο στοιχείο απαιτεί τη διεπαφή «Άτομο».

Όταν εμφανίζεται η σχέση ενός στοιχείου με άλλα στοιχεία, η σημειογραφία γλειφιτζουριού και υποδοχής περιλαμβάνει ένα βέλος εξάρτησης. Σε ένα τέτοιο διάγραμμα στοιχείων, το βέλος εξάρτησης προέρχεται από την υποδοχή που απαιτεί και η αιχμή του βέλους του συνδέεται με το γλειφιτζούρι του παρόχου, όπως φαίνεται στο Σχήμα 21.

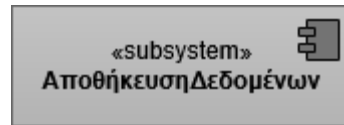


Σχήμα 21: Απεικόνιση εξαρτήσεων στοιχείου.

Το Σχήμα 21 δείχνει ότι το στοιχείο «Σύστημα Παραγγελιών» εξαρτάται από τα στοιχεία «Αποθετήριο Πελατών» και «Σύστημα Αποθεμάτων». Παρατηρούμε στο σχήμα τα διπλότυπα ονόματα των διεπαφών "ΑναζήτησηΠελατών" και "ΑξεσουάρΠροϊόντος". Ενώ αυτό μπορεί να φαίνεται άσκοπα επαναλαμβανόμενο σε αυτό το παράδειγμα, η σημειογραφία επιτρέπει στην πραγματικότητα διαφορετικές διεπαφές (και διαφορετικά ονόματα) σε κάθε στοιχείο ανάλογα με τις διαφορές υλοποίησης (π.χ., ένα στοιχείο παρέχει μια διεπαφή που είναι μια υποκλάση μιας μικρότερης απαιτούμενης διεπαφής).

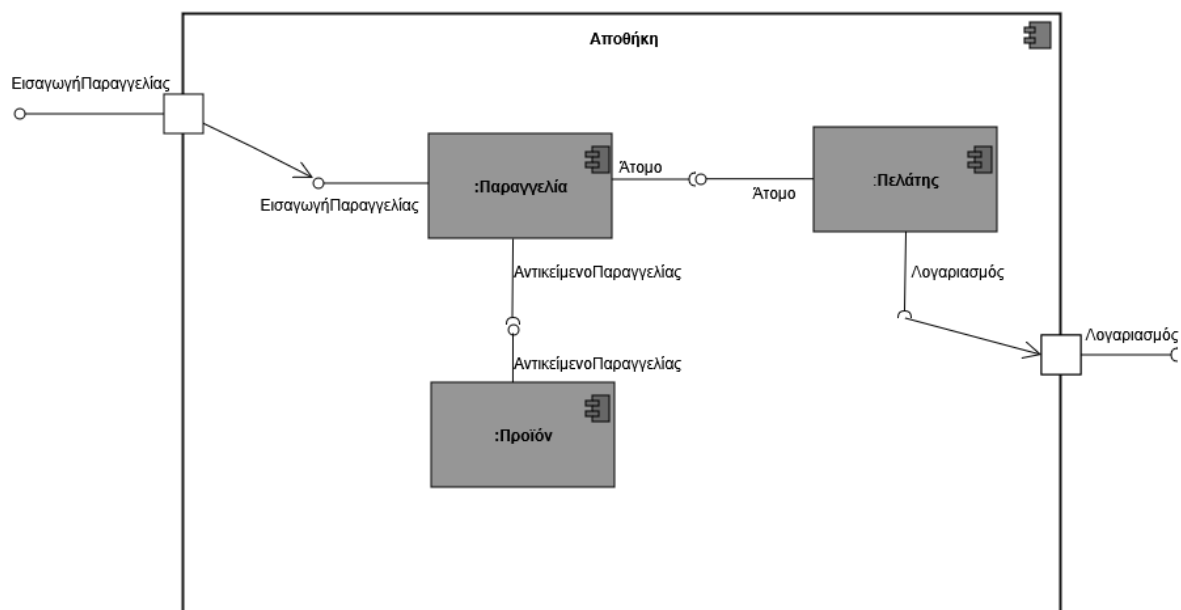
Στην UML 2.x, ο ταξινομητής υποσυστήματος είναι μια εξειδικευμένη έκδοση ενός ταξινομητή στοιχείων. Εξαιτίας αυτού, το στοιχείο συμβολισμού υποσυστήματος κληρονομεί όλους τους ίδιους κανόνες με το στοιχείο συμβολισμού του συστατικού

στοιχείου. Η μόνη διαφορά είναι ότι ένα στοιχείο συμβολισμού υποσυστήματος έχει τη λέξη-κλειδί "subsystem" αντί για "component", όπως φαίνεται στο Σχήμα 22.



Σχήμα 22: Απεικόνιση στοιχείου υποσυστήματος.

Κάποιες φορές χρειάζεται να εμφανίζεται η εσωτερική δομή ενός στοιχείου. Για να αναπαραστήσουμε την εσωτερική δομή ενός στοιχείου, απλώς σχεδιάζουμε το στοιχείο μεγαλύτερο από το κανονικό και τοποθετήστε τα εσωτερικά μέρη μέσα στο διαμέρισμα ονόματος του περιβάλλοντος στοιχείου. Το Σχήμα 23 δείχνει την εσωτερική δομή του στοιχείου «Αποθήκη».



Σχήμα 23: Απεικόνιση δομής στοιχείου που αποτελείται από άλλα στοιχεία.

Στο παράδειγμα που φαίνεται στο Σχήμα 23, το στοιχείο «Αποθήκη» παρέχει τη διεπαφή του «Εισαγωγή Παραγγελίας» και απαιτεί τη διεπαφή «Λογαριασμός». Το στοιχείο «Αποθήκη» αποτελείται από τρία στοιχεία: τα στοιχεία «Παραγγελία», «Πελάτης» και «Προϊόν». Παρατηρήστε πώς τα σύμβολα διεπαφής «Εισαγωγή Παραγγελίας» και «Λογαριασμός» του «Αποθήκη» έχουν ένα τετράγωνο στην άκρη του στοιχείου. Αυτό το τετράγωνο ονομάζεται θύρα (port). Με απλά λόγια, οι θύρες παρέχουν μία μοντελοποίηση

του τρόπου με τον οποίο οι παρεχόμενες/απαιτούμενες διεπαφές ενός στοιχείου σχετίζονται με τα εσωτερικά του μέρη.

Χρησιμοποιώντας μια θύρα, το διάγραμμά μας είναι σε θέση να αποσυνδέσει τα εσωτερικά στοιχεία του στοιχείου «Αποθήκη» από εξωτερικές οντότητες. Στο Σχήμα 23, η θύρα «ΕισαγωγήΠαραγγελίας» αναθέτει την επεξεργασία στη διεπαφή «ΕισαγωγήΠαραγγελίας» του στοιχείου «Παραγγελία». Επίσης, η απαιτούμενη διεπαφή «Λογαριασμός» του εσωτερικού στοιχείου «Πελάτης» ανατίθεται στην απαιτούμενη θύρα διεπαφής «Λογαριασμός» του στοιχείου «Αποθήκη». Συνδεόμενοι στη θύρα «Λογαριασμός», τα εσωτερικά στοιχεία του στοιχείου «Αποθήκη» (π.χ. το στοιχείο «Πελάτης») μπορούν να έχουν έναν τοπικό εκπρόσωπο κάποιας άγνωστης εξωτερικής οντότητας που υλοποιεί τη διεπαφή της θύρας. Η απαιτούμενη διεπαφή «Λογαριασμός» θα υλοποιηθεί από ένα στοιχείο εκτός του στοιχείου «Αποθήκη».

Θα παρατηρήσετε επίσης στο Σχήμα 23 ότι οι διασυνδέσεις μεταξύ των εσωτερικών στοιχείων είναι διαφορετικές από αυτές που φαίνονται στο Σχήμα 21. Αυτό συμβαίνει επειδή αυτές οι απεικονίσεις εσωτερικών δομών είναι στην πραγματικότητα διαγράμματα συνεργασίας που είναι τοποθετημένα μέσα στον ταξινομητή (ένα στοιχείο, στην περίπτωση μας), αφού τα διαγράμματα συνεργασίας δείχνουν στιγμιότυπα ή ρόλους ταξινομητών. Η σχέση που μοντελοποιείται μεταξύ των εσωτερικών στοιχείων σχεδιάζεται με αυτό που η UML αποκαλεί σύνδεσμο συναρμολόγησης. Ένας σύνδεσμος συναρμολόγησης συνδέει την παρεχόμενη διεπαφή ενός στοιχείου με την απαιτούμενη διεπαφή ενός άλλου στοιχείου. Οι σύνδεσμοι συναρμολόγησης σχεδιάζονται ως σύμβολα γλειφιτζουριού και υποδοχής το ένα δίπλα στο άλλο. Η σχεδίαση αυτών των συνδέσμων συναρμολόγησης με αυτόν τον τρόπο καθιστά τα σύμβολα γλειφιτζουριού και υποδοχής πολύ εύκολα στην ανάγνωση.

5.5 Καλές πρακτικές στην ενοποιημένη χρήση UML διαγραμμάτων

Στη διαδικασία ανάπτυξης λογισμικού, το πρώτο βήμα είναι η συλλογή όλων των απαιτήσεων από τα ενδιαφερόμενα μέρη που ενδιαφέρονται για το λογισμικό. Το

λογισμικό είναι καλό όταν ικανοποιούνται όλες οι ανάγκες όλων των ενδιαφερόμενων μερών.

Η εξαγωγή απαιτήσεων είναι μια διαδικασία αναζήτησης, περιήγησης και κατανόησης με περισσότερες λεπτομέρειες. Έτσι, η ίδια η εξαγωγή απαιτήσεων είναι μια διαδικασία κατά την οποία αναζητούνται πληροφορίες σε μεγαλύτερο βάθος και διεξοδικά από όλα τα ενδιαφερόμενα μέρη που έχουν σχέση με το λογισμικό.

Τα αποτελέσματα της εξαγωγής απαιτήσεων έχουν τη μορφή έκθεσης σε οποιαδήποτε φυσική γλώσσα που σχετίζεται με τις επιθυμίες και τα ενδιαφέροντα του λογισμικού. Στη συνέχεια, η φυσική γλώσσα από την εξαγωγή απαιτήσεων αντιστοιχίζεται σε απαίτηση λογισμικού που μπορεί να είναι μια δήλωση με φυσική πρόταση μέχρι ένα μαθηματικό σύμβολο.

Είδαμε ότι οι απαιτήσεις λογισμικού μπορούν να κατηγοριοποιηθούν σε λειτουργικές και μη λειτουργικές. Για την εύκολη ανίχνευση της συνέπειας της μοντελοποίησης λογισμικού με διαγράμματα UML, οι απαιτήσεις μπορούν να κωδικοποιηθούν και να καταγραφούν σε έναν πίνακα, όπου στη μια στήλη θα καταγράφονται οι κωδικοί και στην άλλη στήλη οι περιγραφές των αντίστοιχων απαιτήσεων. Π.χ. η απαίτηση «Το σύστημα είναι σε θέση να λαμβάνει δεδομένα εισόδου από έγγραφα κειμένου», θα μπορούσε να αναφέρεται με τον κωδικό «Λ.Α. 01» (δηλαδή λειτουργική απαίτηση με κωδικό 01).

Το διάγραμμα περίπτωσης χρήσης χρησιμοποιείται για τη μοντελοποίηση ή την περιγραφή της αλληλεπίδρασης μεταξύ δρώντος και συστήματος από την οπτική γωνία του δράστη, τι κάνουν ή δίνουν οι δρώντες στο σύστημα και τι χρειάζονται ή λαμβάνουν οι δρώντες από το σύστημα.

Το διάγραμμα περίπτωσης χρήσης χρησιμοποιείται συνήθως για την περιγραφή της επιχειρηματικής διαδικασίας του συστήματος και τη μοντελοποίηση της ανάλυσης του συστήματος. Η μοντελοποίηση της ανάλυσης του συστήματος συνδέεται αυτόματα με τη λειτουργική απαίτηση.

Κάθε περίπτωση χρήσης στο διάγραμμα περίπτωσης χρήσης πρέπει να αντιπροσωπεύει τουλάχιστον μία απαίτηση. Επίσης σε κάθε περίπτωση χρήσης θα είναι καλύτερα εάν της δοθεί ειδικός κώδικας για εύκολη διαδικασία ανίχνευσης, π.χ. «Π.Χ. 01» (δηλαδή περίπτωση χρήσης με κωδικό 01).

Το διάγραμμα κλάσης είναι το κύριο διάγραμμα που χαρακτηρίζει την αντικειμενοστρεφή ανάλυση και σχεδιασμό λογισμικού, επειδή το διάγραμμα κλάσης αναπαριστά αντικείμενα και περιγράφει τη σχέση μεταξύ αντικειμένων του συστήματος. Στην αντικειμενοστρεφή ανάλυση και σχεδιασμό λογισμικού, η αναγνώριση αντικειμένων είναι ένα σημαντικό βήμα. Η αναγνώριση αντικειμένων ή κλάσεων συστήματος γίνεται εύκολα από το αποτέλεσμα της διαδικασίας εξαγωγής απαιτήσεων. Κάθε απαίτηση φυσικά έχει ένα αντικείμενο που σχετίζεται με τις λειτουργίες του συστήματος.

Επιπλέον, από την περίπτωση χρήσης μπορούν να αναγνωριστούν αντικείμενα του συστήματος. Όπως συμβαίνει και με το διάγραμμα περιπτώσεων χρήσης, για την εύκολη ανίχνευση της σύνδεσης των διαγραμμάτων, είναι καλύτερο να δίνεται μοναδικός κώδικας για κάθε κλάση στο διάγραμμα κλάσης.

Ένα διάγραμμα ακολουθίας είναι ένα διάγραμμα αλληλεπίδρασης που περιγράφει την αλληλεπίδραση μεταξύ αντικειμένων σε μια χρονική ακολουθία με μια σειρά μηνυμάτων που ρέουν και τα οποία ζητούνται ή παρέχονται από ένα αντικείμενο σε ένα άλλο αντικείμενο. Κάθε περίπτωση χρήσης σε ένα διάγραμμα περίπτωσης χρήσης έχει ένα σενάριο χρήσης που περιγράφει λεπτομερώς τη διαδικασία μέσα στην περίπτωση χρήσης. Η ροή του μηνύματος στο διάγραμμα ακολουθίας είναι σύμφωνα με τη ροή της διαδικασίας από την περίπτωση χρήσης σεναρίου. Ιδανικά, ένα διάγραμμα ακολουθίας μοντελοποιεί κάθε κανονική και εναλλακτική ροή της διαδικασίας από την περίπτωση χρήσης σεναρίου.

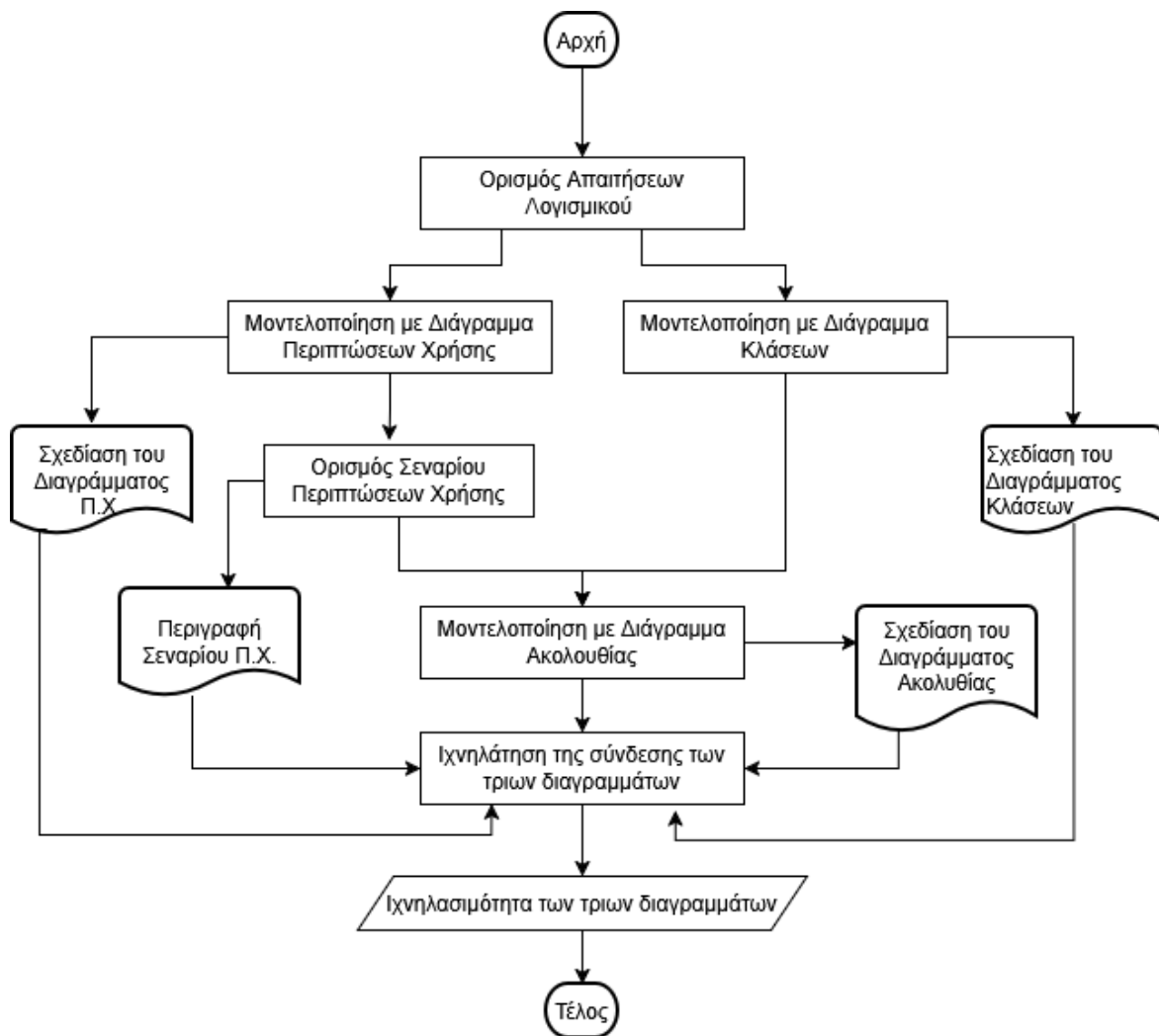
Σε αντίθεση με τη ροή μηνύματος σε ένα διάγραμμα ακολουθίας που σχετίζεται με την περίπτωση χρήσης σεναρίου, ένα αντικείμενο σε ένα διάγραμμα ακολουθίας σχετίζεται με ένα αντικείμενο ή μια κλάση σε ένα διάγραμμα κλάσης. Στη συνέχεια, το μήνυμα στη ροή μηνύματος είναι μια λειτουργία από το αντικείμενο ή την κλάση που χρησιμοποιείται.

Η σχέση μεταξύ αντικειμένων στο διάγραμμα ακολουθίας πρέπει να είναι συνεπής με τη σχέση μεταξύ αντικειμένων στο διάγραμμα κλάσεων. Είναι καλή πρακτική κάθε διάγραμμα ακολουθίας να έχει έναν μοναδικό κωδικό για να είναι πιο εύκολη η ανίχνευση των διαγραμμάτων.

Η ανίχνευση της σχέσης ή της σύνδεσης μεταξύ των διαγραμμάτων στην UML είναι μια σημαντική διαδικασία, διότι η ιχνηλασιμότητα των διαγραμμάτων μπορεί να μετρήσει το πόσο καλά έχει δημιουργηθεί το λογισμικό μοντελοποίησης και αν πληρούνται όλες οι

απαιτήσεις του χρήστη και του λογισμικού. Για παράδειγμα η ιχνηλασιμότητα για τις λειτουργικές απαιτήσεις, τις περιπτώσεις χρήσης, και τις κλάσεις μπορεί να απεικονιστεί σε έναν πίνακα τριών στηλών, όπου στην πρώτη στήλη θα είναι οι κωδικοί των λειτουργικών απαιτήσεων, στη δεύτερη στήλη οι κωδικοί των περιπτώσεων χρήσης και στην τρίτη στήλη οι κλάσεις με τους κωδικούς τους. Σε κάθε γραμμή του πίνακα θα υπάρχει αντιστοιχία λειτουργικής απαίτησης με τις αντίστοιχες περιπτώσεις χρήσης και κλάσεις.

Στο Σχήμα 24 υπάρχει περιγραφή σχετικά με τη διαδικασία ροής της μοντελοποίησης, ανάλυσης και σχεδιασμού λογισμικού με UML, ειδικά χρησιμοποιώντας διαγράμματα περιπτώσεων χρήσης, κλάσης και ακολουθίας, μέχρι να προκύψει η ιχνηλασιμότητα αυτών των διαγραμμάτων.

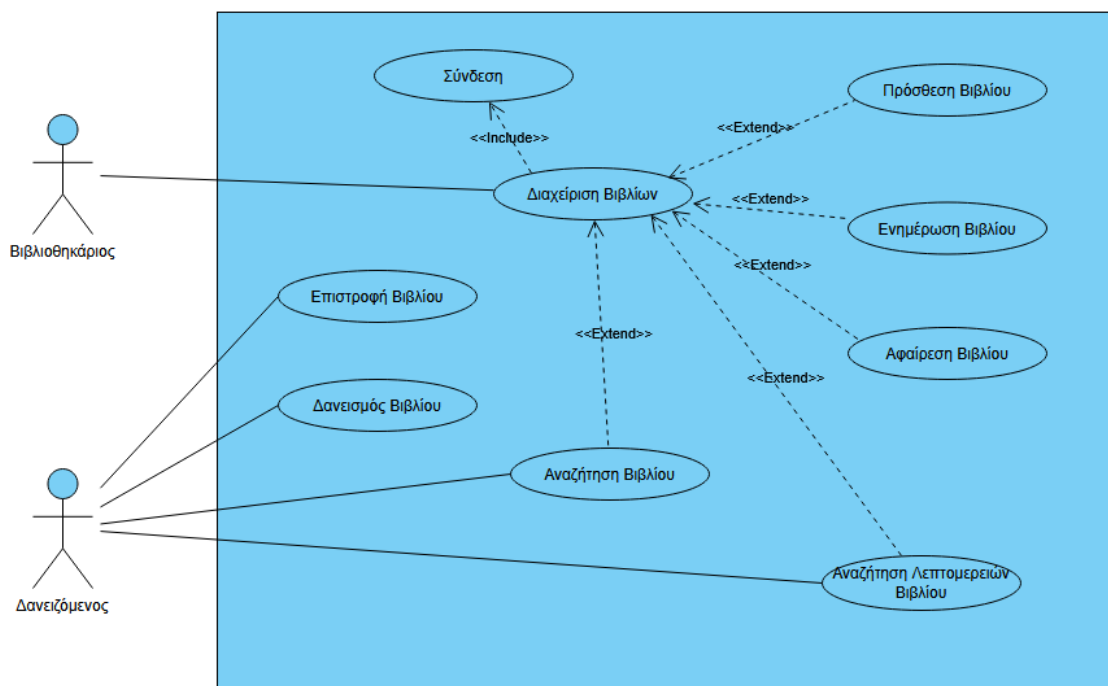


Σχήμα 24: Ροή διαδικασίας μοντελοποίησης της ανάλυσης και σχεδίασης λογισμικού με χρήση UML.

5.6 Παράδειγμα μετάβασης μεταξύ διαγραμμάτων UML

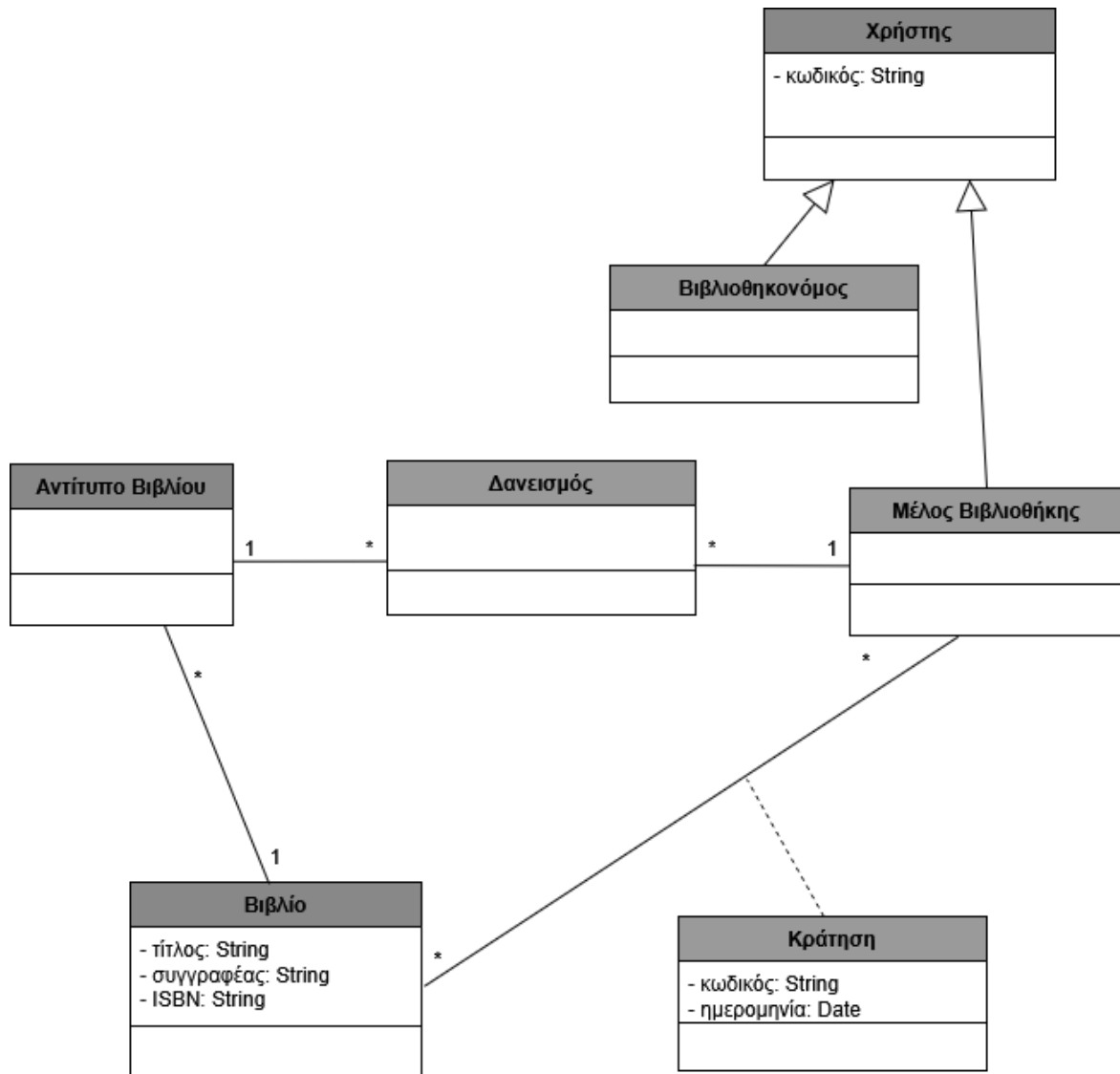
Σε αυτή την ενότητα θα γίνει παρουσίαση ενός ενιαίου παραδείγματος όπου θα φανεί η σύνδεση μεταξύ των διαγραμμάτων, δηλαδή η μετάβαση από τον ένα τύπο διαγράμματος στον άλλο, μέσω ενός συστήματος διαχείρισης βιβλιοθήκης.

Από το διάγραμμα περιπτώσεων χρήσης προκύπτει η βασική λειτουργία «Δανεισμός βιβλίου», όπως φαίνεται στο σχήμα 25.



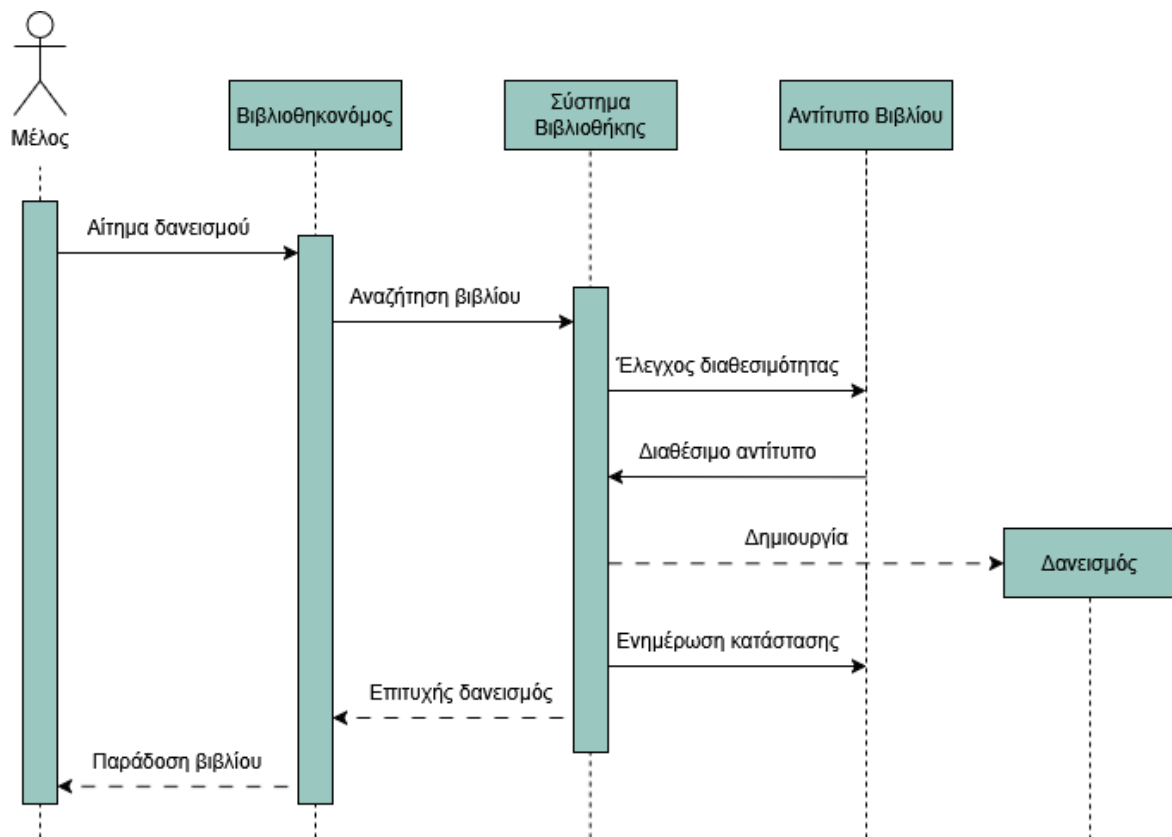
Σχήμα 25: Διάγραμμα περιπτώσεων χρήσης.

Με βάση την περίπτωση χρήσης «Δανεισμός βιβλίου», προκύπτουν οι βασικές κλάσεις: Μέλος Βιβλιοθήκης, Βιβλίο, Αντίτυπο, Δανεισμός, για την κατασκευή του διαγράμματος κλάσεων, όπως αυτό φαίνεται στο σχήμα 26.



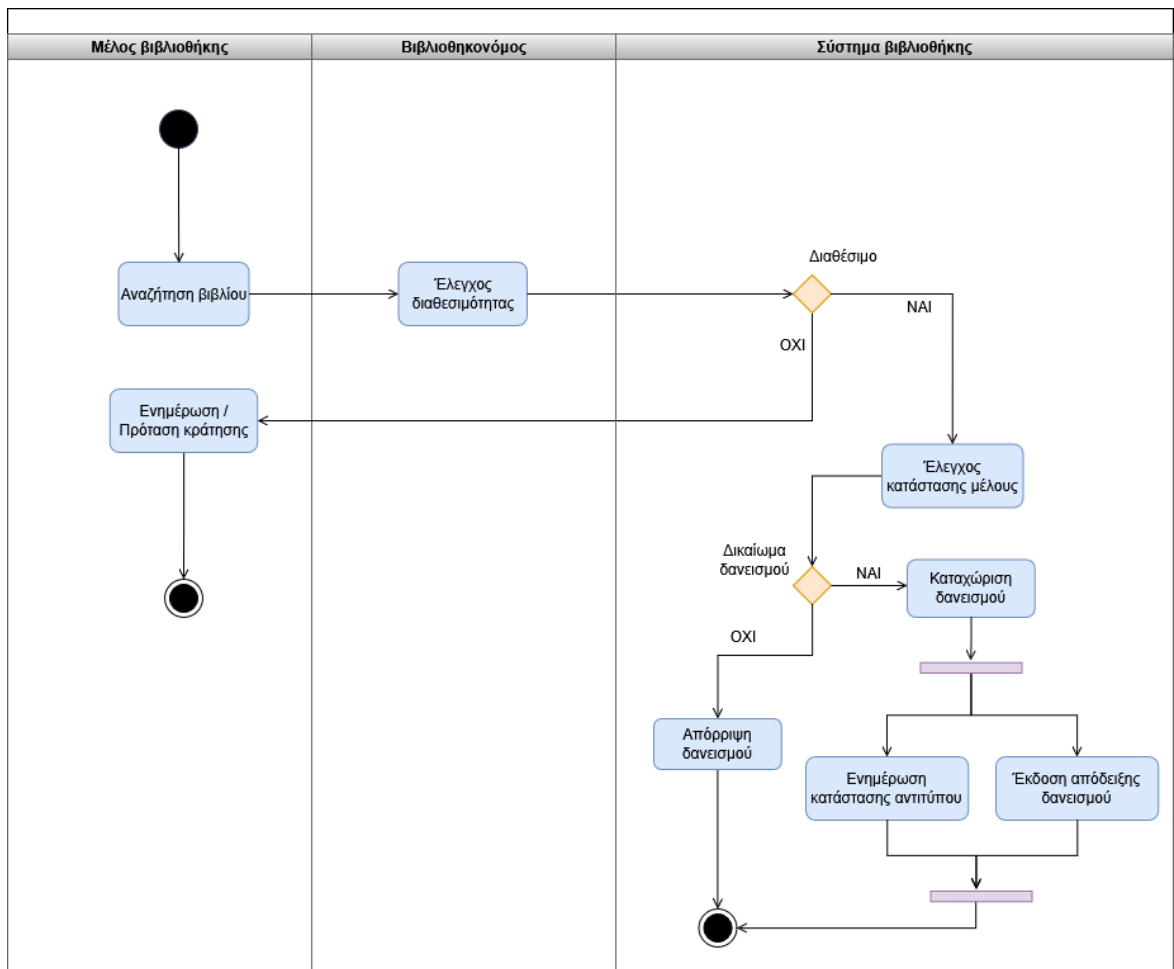
Σχήμα 26: Διάγραμμα κλάσεων.

Η διαδικασία δανεισμού απεικονίζεται και δυναμικά μέσω διαγράμματος ακολουθίας όπου παρουσιάζεται η αλληλεπίδραση μεταξύ Μέλους, Βιβλιοθηκονόμου και Συστήματος, όπως φαίνεται στο σχήμα 27.



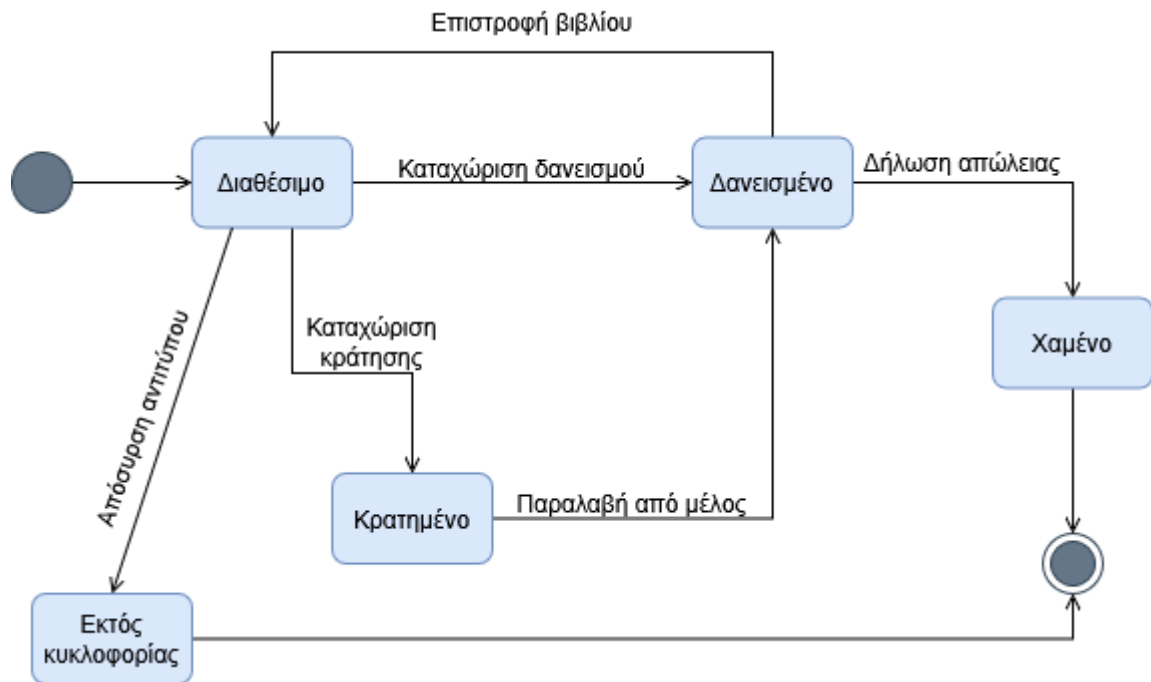
Σχήμα 27: Διάγραμμα ακολουθίας.

Η συνολική ροή της διαδικασίας δανεισμού απεικονίζεται με διάγραμμα δραστηριοτήτων, το οποίο αποτυπώνει τα βήματα και τις αποφάσεις της διαδικασίας, όπως φαίνεται στο σχήμα 28.



Σχήμα 28: Διάγραμμα δραστηριοτήτων.

Η κατάσταση ενός αντιτύπου βιβλίου μεταβάλλεται από «διαθέσιμο» σε «δανεισμένο» και αντίστροφα, όπως αποτυπώνεται στο διάγραμμα καταστάσεων, στο σχήμα 29.



Σχήμα 29: Διάγραμμα καταστάσεων.

Γίνεται έτσι κατανοητό ότι η σχεδίαση του συστήματος ακολουθεί μια λογική μετάβαση από τις απαιτήσεις προς τη δομή και τη συμπεριφορά. Αρχικά, οι λειτουργικές απαιτήσεις αποτυπώνονται μέσω διαγραμμάτων περιπτώσεων χρήσης. Στη συνέχεια, οι βασικές έννοιες του συστήματος μοντελοποιούνται με διαγράμματα κλάσεων. Οι αλληλεπιδράσεις μεταξύ των οντοτήτων περιγράφονται με διαγράμματα ακολουθίας, ενώ η συνολική ροή των διαδικασιών αποτυπώνεται μέσω διαγραμμάτων δραστηριοτήτων. Τέλος, τα διαγράμματα καταστάσεων παρουσιάζουν τον κύκλο ζωής των βασικών οντοτήτων του συστήματος.

6. Συμβολή των UML διαγραμμάτων στη σχεδίαση λογισμικού

6.1 Η UML ως εργαλείο θεωρητικής σκέψης

Η UML δεν είναι απλώς ένα τεχνικό εργαλείο σχεδίασης λογισμικού, αλλά μπορεί να λειτουργήσει ουσιαστικά ως εργαλείο θεωρητικής σκέψης, δηλαδή ως μέσο οργάνωσης, ανάλυσης και αναπαράστασης σύνθετων και πολύπλοκων εννοιών και σχέσεων. Συνεισφέρει σημαντικά στη θεωρητική σκέψη παρέχοντας ένα δομημένο τρόπο αναπαράστασης αφηρημένων εννοιών, σχέσεων και συμπεριφορών. Μέσα από την οπτικοποίηση, την τυποποίηση και την αφαίρεση, τα διαγράμματά της βοηθούν ερευνητές και επαγγελματίες να αναπτύξουν βαθύτερη κατανόηση εννοιών και αναλυτική σκέψη.

Ένας από τους κύριους τρόπους με τους οποίους τα διαγράμματα UML υποστηρίζουν τη θεωρητική σκέψη είναι μέσω της αφαίρεσης. Τα διαγράμματα κλάσεων, περιπτώσεων χρήσης και ακολουθίας, επιτρέπουν στους σχεδιαστές να αναπαριστούν μόνο τις σχετικές πτυχές ενός συστήματος, παραλείποντας περιττές λεπτομέρειες. Αυτή η αφαίρεση ενθαρρύνει τους στοχαστές να επικεντρώνονται σε εννοιολογικές δομές και όχι σε λεπτομέρειες υλοποίησης.

Μια άλλη σημαντική συμβολή είναι η οπτική αναπαράσταση σχέσεων και δομών. Η ανθρώπινη νόηση επεξεργάζεται αποτελεσματικά τις οπτικές πληροφορίες και τα διαγράμματα UML μετατρέπουν τις θεωρητικές ιδέες σε οπτικά μοντέλα που μπορούν εύκολα να ερμηνευθούν. Για παράδειγμα, τα διαγράμματα κλάσεων απεικονίζουν σχέσεις όπως η κληρονομικότητα, η συσσωμάτωση και η συσχέτιση μεταξύ των στοιχείων του συστήματος. Αυτές οι γραφικές αναπαραστάσεις επιτρέπουν στους θεωρητικούς να εξερευνήσουν τις δομές του συστήματος, να εντοπίσουν εξαρτήσεις και να συλλογιστούν σχετικά με τις αλληλεπιδράσεις μεταξύ οντοτήτων.

Η τυποποιημένη σημειογραφία της UML παρέχει μια κοινή γλώσσα που επιτρέπει στους μελετητές και τους προγραμματιστές να εκφράζουν θεωρητικές ιδέες με ακρίβεια. Όταν οι έννοιες αναπαρίστανται μέσω τυποποιημένων συμβόλων και σχέσεων, γίνεται ευκολότερη η ανάλυση της λογικής του συστήματος και η επαλήθευση υποθέσεων. Για παράδειγμα,

τα διαγράμματα ακολουθίας μοντελοποιούν τις αλληλεπιδράσεις με την πάροδο του χρόνου, επιτρέποντας στους θεωρητικούς να εξετάσουν τη λογική ροή των γεγονότων και να εντοπίσουν ασυνέπειες ή ανεπάρκειες στα εννοιολογικά σχέδια. Αυτή η δομημένη αναπαράσταση ενισχύει την αναλυτική σκέψη και υποστηρίζει την ανάπτυξη υποθέσεων στη μοντελοποίηση συστημάτων.

Επιπλέον, τα διαγράμματα UML υποστηρίζουν τη συνεργατική θεωρητική ανάπτυξη. Η θεωρητική σκέψη σε τομείς όπως η μηχανική λογισμικού, τα συστήματα πληροφοριών και η αρχιτεκτονική συστημάτων συχνά απαιτεί επικοινωνία μεταξύ πολλαπλών ερευνητών ή προγραμματιστών. Παρέχουν μια κοινή οπτική γλώσσα που μειώνει την ασάφεια και βελτιώνει την κατανόηση μεταξύ των μελών της ομάδας. Αναπαριστώντας οπτικά τις θεωρητικές ιδέες, τα διαγράμματα UML επιτρέπουν στα ενδιαφερόμενα μέρη να συζητούν, να ασκούν κριτική και να βελτιώνουν τα εννοιολογικά μοντέλα πιο αποτελεσματικά.

Τέλος, η UML ενθαρρύνει τη συστηματική ανάλυση προβλημάτων, μια βασική πτυχή της θεωρητικής συλλογιστικής. Τα σύνθετα συστήματα μπορούν να αναλυθούν σε μικρότερα στοιχεία χρησιμοποιώντας διαγράμματα όπως διαγράμματα στοιχείων, δραστηριοτήτων και καταστάσεων. Αυτή η ανάλυση επιτρέπει στους στοχαστές να αναλύουν μεμονωμένα μέρη ενός συστήματος διατηρώντας παράλληλα μια κατανόηση της συνολικής δομής. Μέσω αυτής της διαδικασίας, τα διαγράμματα UML βοηθούν στη γεφύρωση του χάσματος μεταξύ των θεωρητικών μοντέλων και του πρακτικού σχεδιασμού συστημάτων.

Συμπερασματικά, τα διαγράμματα UML παίζουν σημαντικό ρόλο στη θεωρητική σκέψη, επιτρέποντας την αφαίρεση, την οπτικοποίηση, την τυποποίηση, τη συνεργασία και τη συστηματική ανάλυση. Χρησιμοποιούν όχι μόνο ως πρακτικά εργαλεία σχεδιασμού, αλλά και ως εννοιολογικά πλαίσια που υποστηρίζουν την αναλυτική συλλογιστική και την ανάπτυξη θεωρητικών μοντέλων στη μηχανική λογισμικού και σε συναφείς κλάδους. Μετασχηματίζοντας σύνθετες ιδέες σε δομημένες οπτικές αναπαραστάσεις, τα διαγράμματα UML ενισχύουν τόσο την κατανόηση όσο και την επικοινωνία των θεωρητικών εννοιών.

6.2 Ενίσχυση της κατανόησης σύνθετων συστημάτων

Η UML αποτελεί ένα από τα σημαντικότερα εργαλεία οπτικής μοντελοποίησης, συμβάλλοντας ουσιαστικά στην κατανόηση σύνθετων συστημάτων. Σε ένα περιβάλλον όπου τα πληροφοριακά ή τεχνολογικά συστήματα χαρακτηρίζονται από πολλαπλότητα στοιχείων και αλληλεπιδράσεων, η ανάγκη για σαφή και δομημένη αναπαράσταση καθίσταται κρίσιμη. Η UML προσφέρει ένα τυποποιημένο πλαίσιο μέσα από το οποίο τα πολύπλοκα αυτά συστήματα αποδομούνται, οργανώνονται και αναλύονται με τρόπο συστηματικό και κατανοητό.

Η UML διευκολύνει τη διαχείριση της πολυπλοκότητας μέσω της αφαιρετικής αναπαράστασης. Αντί να εξετάζεται ένα σύστημα στο σύνολό του με όλες τις λεπτομέρειες ταυτόχρονα, τα διαγράμματα της UML επιτρέπουν την εστίαση σε συγκεκριμένες πτυχές του. Για παράδειγμα, τα διαγράμματα κλάσεων αναδεικνύουν τη δομή και τις σχέσεις μεταξύ των βασικών οντοτήτων, ενώ τα διαγράμματα ακολουθίας παρουσιάζουν τη δυναμική αλληλεπίδραση μεταξύ τους. Με αυτόν τον τρόπο, ο αναλυτής μπορεί να μελετήσει σταδιακά το σύστημα, κατανοώντας πρώτα τα βασικά δομικά του στοιχεία και στη συνέχεια τις λειτουργικές του συμπεριφορές.

Επίσης, η UML προάγει τη συστημική σκέψη. Τα σύνθετα συστήματα δεν αποτελούνται απλώς από επιμέρους μέρη, αλλά από σχέσεις, εξαρτήσεις και ροές πληροφορίας. Μέσα από τα διαφορετικά είδη διαγραμμάτων (δομικά και συμπεριφορικά), η UML επιτρέπει την πολυεπίπεδη προσέγγιση ενός συστήματος. Ο μελετητής μπορεί να εξετάσει πώς οργανώνεται η αρχιτεκτονική, πώς επικοινωνούν τα υποσυστήματα και πώς εξελίσσονται οι διαδικασίες στον χρόνο. Η δυνατότητα αυτή ενισχύει την ολιστική κατανόηση, καθώς καθίσταται σαφές όχι μόνο «τι» αποτελεί το σύστημα, αλλά και «πώς» λειτουργεί.

Παράλληλα, η UML λειτουργεί ως εργαλείο εννοιολογικής αποσαφήνισης. Σε σύνθετα έργα, συχνά διαφορετικοί εμπλεκόμενοι (αναλυτές, προγραμματιστές, διαχειριστές έργου) ενδέχεται να έχουν διαφορετικές ερμηνείες για τις ίδιες έννοιες. Η τυποποίηση της UML μειώνει την ασάφεια, προσφέροντας κοινή γλώσσα επικοινωνίας. Μέσω σαφώς ορισμένων συμβόλων και κανόνων, οι ιδέες αποδίδονται με ακρίβεια, διευκολύνοντας τη συνεργασία και περιορίζοντας τα σφάλματα κατανόησης.

Συμπερασματικά, η UML ενισχύει την κατανόηση σύνθετων συστημάτων παρέχοντας δομημένη, αφαιρετική και πολυεπίπεδη αναπαράσταση της πραγματικότητας. Μέσω της οπτικοποίησης και της τυποποίησης που προσφέρει, μετατρέπει την πολυπλοκότητα σε διαχειρίσιμη γνώση και συμβάλλει ουσιαστικά στη βαθύτερη κατανόηση και στον αποτελεσματικό σχεδιασμό πολύπλοκων συστημάτων.

6.3 Υποστήριξη επικοινωνίας και τεκμηρίωσης

Πέρα από τη σχεδιαστική της διάσταση, η UML διαδραματίζει καθοριστικό ρόλο στην ενίσχυση της επικοινωνίας μεταξύ των εμπλεκόμενων μερών ενός έργου και στη συστηματική τεκμηρίωσή του. Η αξία της δεν περιορίζεται στη δημιουργία διαγραμμάτων, αλλά επεκτείνεται στη διαμόρφωση ενός κοινού πλαισίου κατανόησης και αναφοράς.

Πρωταρχικώς, η UML υποστηρίζει την επικοινωνία μέσω της οπτικοποίησης σύνθετων εννοιών. Τα διαγράμματα, όπως τα διαγράμματα κλάσεων, περιπτώσεων χρήσης, ακολουθίας και δραστηριοτήτων, επιτρέπουν την απεικόνιση της δομής και της συμπεριφοράς ενός συστήματος με τρόπο σαφή και οργανωμένο. Με τον τρόπο αυτό, μειώνονται οι παρερμηνείες και ενισχύεται η συνεργασία μεταξύ αναλυτών, προγραμματιστών, διαχειριστών έργου και χρηστών.

Πιο συγκεκριμένα, η οπτική αναπαράσταση διευκολύνει την κατανόηση ακόμη και από άτομα που δεν διαθέτουν βαθιά τεχνική κατάρτιση, όπως στελέχη διοίκησης ή πελάτες. Ειδικά σε αυτήν την περίπτωση η επικοινωνία είναι κάποιες φορές πιο δύσκολη και η UML βοηθά στην καθιέρωση ενός τυποποιημένου μέσου εξήγησης του μεγέθους και της πολυπλοκότητας του έργου στα ενδιαφερόμενα μέρη.

Επίσης η UML μπορεί να χρησιμοποιηθεί για την επικοινωνία της τρέχουσας κατάστασης στους νεοεισερχόμενους στο έργο. Αυτό είναι πολύ σημαντικό στην προσπάθεια αποφυγής της «εξάτμισης της γνώσης». Αυτή η κατάσταση συμβαίνει όταν ένας ειδικός σε ένα σύστημα εγκαταλείπει ένα έργο, παίρνοντας μαζί του κάποιες γνώσεις που δεν είναι τεκμηριωμένες.

Παράλληλα, η UML λειτουργεί ως κοινή γλώσσα μεταξύ διαφορετικών ειδικοτήτων. Η τυποποίηση των συμβόλων και των κανόνων της εξασφαλίζει ότι ένα διάγραμμα έχει την ίδια σημασία ανεξάρτητα από το ποιος το δημιούργησε. Αυτό είναι ιδιαίτερα σημαντικό σε μεγάλα ή διεθνή έργα, όπου συμμετέχουν πολλαπλές ομάδες ανάπτυξης. Η χρήση ενός ενιαίου πλαισίου μοντελοποίησης ενισχύει τη συνέπεια, επιταχύνει τη λήψη αποφάσεων και διευκολύνει τη διαχείριση αλλαγών.

Ειδικότερα μελέτες έχουν διαπιστώσει ότι για άτομα με πιο εκτενή εκπαίδευση, τα συστήματα που μοντελοποιούνται με την UML είναι σε καλύτερη θέση να μεταδίδουν πληροφορίες σχετικά με τα δεδομένα τους σε σχέση με παραδοσιακές γλώσσες (Fernandez-Saez, Chaudron & Genero, 2018).

Όσον αφορά την τεκμηρίωση, η UML προσφέρει έναν δομημένο και διαχρονικό τρόπο καταγραφής των απαιτήσεων, της αρχιτεκτονικής και της λειτουργίας ενός συστήματος. Τα διαγράμματα αποτελούν μέρος της επίσημης τεκμηρίωσης και μπορούν να χρησιμοποιηθούν τόσο κατά τη φάση της ανάπτυξης όσο και κατά τη συντήρηση.

Μέσα από τα διαγράμματα περιπτώσεων χρήσης αποτυπώνονται οι λειτουργικές απαιτήσεις, ενώ τα διαγράμματα κλάσεων και ακολουθίας τεκμηριώνουν τη λογική και τη ροή των διαδικασιών. Έτσι, η γνώση για το σύστημα δεν παραμένει αποκλειστικά στα μέλη της ομάδας, αλλά αποθηκεύεται σε οργανωμένη μορφή.

Για τους σχεδιαστές προγραμμάτων, τους αρχιτέκτονες και τους προγραμματιστές, τα διαγράμματα UML αποτελούν ένα σημαντικό εργαλείο στη διαδικασία δημιουργίας καλοφτιαγμένου, χρήσιμου λογισμικού. Όταν χρησιμοποιούνται σωστά, τα διαγράμματα UML μπορούν να προσθέσουν αξία στην τεχνική τεκμηρίωση, βοηθώντας τον χρήστη να οργανώσει, να οπτικοποιήσει και να κατανοήσει πολύπλοκες προγραμματιστικές έννοιες και επιχειρηματικές διαδικασίες.

Επίσης, η UML διευκολύνει τη μελλοντική επέκταση και αναβάθμιση ενός συστήματος. Η ύπαρξη ενημερωμένων μοντέλων επιτρέπει σε νέους προγραμματιστές να κατανοήσουν γρήγορα τη δομή και τη λειτουργία του έργου. Αυτό μειώνει τον χρόνο προσαρμογής, περιορίζει τα σφάλματα και συμβάλλει στη βιωσιμότητα του λογισμικού.

Επομένως, στο σύνολό της, η UML δεν αποτελεί απλώς ένα τεχνικό εργαλείο σχεδίασης, αλλά ένα μέσο επικοινωνίας και οργανωμένης τεκμηρίωσης. Μέσα από την

τυποποιημένη, οπτική και συστηματική αναπαράσταση των στοιχείων ενός συστήματος, ενισχύει τη συνεργασία, μειώνει τις ασάφειες και διασφαλίζει τη διατήρηση της γνώσης σε όλο τον κύκλο ζωής του έργου.

6.4 Θεωρητικά πλεονεκτήματα στη φάση ανάλυσης & αρχιτεκτονικής

Τα πλεονεκτήματά της UML δεν περιορίζονται στην απλή απεικόνιση δομών, αλλά σχετίζονται με τη συστηματοποίηση της σκέψης, τη μείωση της πολυπλοκότητας και τη διαμόρφωση ενός συνεκτικού πλαισίου κατανόησης του συστήματος. Αποτελεί λοιπόν ένα από τα σημαντικότερα εργαλεία θεωρητικής και πρακτικής υποστήριξης κατά τη φάση της ανάλυσης και της αρχιτεκτονικής σχεδίασης ενός πληροφοριακού συστήματος.

Κατά τη φάση της ανάλυσης, συμβάλλει στη σαφή αποτύπωση των απαιτήσεων και της λειτουργικής συμπεριφοράς του συστήματος. Μέσα από διαγράμματα περιπτώσεων χρήσης, επιτυγχάνεται η αναπαράσταση της αλληλεπίδρασης μεταξύ χρηστών και συστήματος, γεγονός που διευκολύνει την κατανόηση των λειτουργικών αναγκών και τον εντοπισμό πιθανών ασαφειών.

Η οπτική μοντελοποίηση ενισχύει τη μετάβαση από τη λεκτική περιγραφή απαιτήσεων σε μια δομημένη μορφή αναπαράστασης, μειώνοντας τον κίνδυνο παρερμηνειών. Παράλληλα, τα διαγράμματα δραστηριοτήτων και ακολουθίας επιτρέπουν την ανάλυση της ροής διαδικασιών και της δυναμικής συμπεριφοράς, προσφέροντας μια συστηματική κατανόηση των λειτουργικών σεναρίων.

Σε θεωρητικό επίπεδο, υποστηρίζει την αρχή της αφαίρεσης, η οποία είναι θεμελιώδης στην ανάλυση συστημάτων. Μέσα από την επιλογή των κατάλληλων επιπέδων μοντελοποίησης, οι αναλυτές μπορούν να εστιάσουν στις βασικές έννοιες και σχέσεις, αποφεύγοντας την πρόωρη ενασχόληση με τεχνικές λεπτομέρειες υλοποίησης. Αυτό επιτρέπει την ανάπτυξη ενός εννοιολογικού μοντέλου που αντικατοπτρίζει την επιχειρησιακή λογική του οργανισμού, πριν από τη μετάβαση σε πιο τεχνικές αποφάσεις.

Στη φάση της αρχιτεκτονικής σχεδίασης, προσφέρει εργαλεία για τη δομική και συμπεριφορική αναπαράσταση του συστήματος. Τα διαγράμματα κλάσεων συμβάλλουν

στη διαμόρφωση της στατικής δομής, αναδεικνύοντας σχέσεις όπως η συσχέτιση, η κληρονομικότητα και η σύνθεση.

Παράλληλα, τα διαγράμματα συστατικών και ανάπτυξης (deployment diagrams) υποστηρίζουν τη μοντελοποίηση της φυσικής και λογικής αρχιτεκτονικής, επιτρέποντας τον διαχωρισμό των επιπέδων (layers) και την αποτύπωση της κατανομής των πόρων.

Ένα σημαντικό θεωρητικό πλεονέκτημα της UML στην αρχιτεκτονική είναι η ενίσχυση της αρθρωσιμότητας (modularity) και της επαναχρησιμοποίησης. Μέσα από τη σαφή οριοθέτηση των συστατικών και των διεπαφών τους, διευκολύνεται η ανάπτυξη χαλαρά ταιριασμένων (loosely coupled)¹ μονάδων, οι οποίες μπορούν να εξελιχθούν ανεξάρτητα. Επιπλέον, η μοντελοποίηση πολλαπλών όψεων του συστήματος (δομική, δυναμική, φυσική) επιτρέπει την πολυδιάστατη προσέγγιση της αρχιτεκτονικής, ενισχύοντας την πληρότητα και τη συνέπεια του σχεδιασμού.

Τέλος, η UML υποστηρίζει τη θεωρητική αρχή της ιχνηλασιμότητας (traceability) μεταξύ απαιτήσεων, ανάλυσης και αρχιτεκτονικής. Τα μοντέλα που δημιουργούνται στην ανάλυση μπορούν να συνδεθούν με τα αρχιτεκτονικά στοιχεία, εξασφαλίζοντας ότι οι τεχνικές αποφάσεις ευθυγραμμίζονται με τις αρχικές ανάγκες. Με αυτόν τον τρόπο, η UML λειτουργεί ως γέφυρα μεταξύ επιχειρησιακής κατανόησης και τεχνικής υλοποίησης.

Καταλήγοντας, τα θεωρητικά πλεονεκτήματα της UML στη φάση της ανάλυσης και της αρχιτεκτονικής έγκεινται στην προώθηση της αφαίρεσης, της δομικής οργάνωσης, της πολυδιάστατης προσέγγισης και της ιχνηλασιμότητας. Αποτελεί ένα ενιαίο εννοιολογικό πλαίσιο που μειώνει την πολυπλοκότητα, ενισχύει τη σαφήνεια και θεμελιώνει έναν συστηματικό και τεκμηριωμένο σχεδιασμό πληροφοριακών συστημάτων.

¹ Με τον όρο loosely coupled εννοούμε μια σχεδιαστική προσέγγιση στην οποία τα στοιχεία ενός συστήματος είναι ασθενώς συνδεδεμένα, επιτρέποντάς τους να λειτουργούν ανεξάρτητα. Αυτό σημαίνει ότι οι αλλαγές σε ένα στοιχείο έχουν ελάχιστη επίδραση στα άλλα, ενισχύοντας την ευελιξία, την επεκτασιμότητα και τη συντηρησιμότητα του συστήματος.

6.5 Περιορισμοί και κριτική της UML από θεωρητική σκοπιά

Αναμφισβήτητα η UML έχει καθιερωθεί ως η κυρίαρχη γλώσσα οπτικής μοντελοποίησης συστημάτων λογισμικού, ιδίως στο πλαίσιο της αντικειμενοστρεφούς ανάλυσης και σχεδίασης. Ωστόσο, από θεωρητική σκοπιά έχουν διατυπωθεί σημαντικές επιφυλάξεις που αφορούν τη φιλοσοφική της θεμελίωση, τη σημασιολογική της πληρότητα και τη μεθοδολογική της συνοχή.

Ένας από τους συχνότερα αναφερόμενους θεωρητικούς περιορισμούς της UML αφορά τη σημασιολογική της ασάφεια. Παρότι η γλώσσα διαθέτει τυπικές προδιαγραφές που εκδίδονται από τον οργανισμό OMG, μεγάλο μέρος της σημασιολογίας της δεν είναι πλήρως τυποποιημένο ή μάλλον είναι ημι-τυποποιημένο (Daw & Cleaveland, 2016).

Εκτός από την ασυνέπεια στη σημασιολογία της, η UML έχει ακόμα επικριθεί για την πολυπλοκότητά της και τις διφορούμενες δομές της. Οι μετρήσεις πολυπλοκότητας δείχνουν ότι η UML είναι θεωρητικά πιο πολύπλοκη από πολλές τεχνικές μοντελοποίησης (Siau, Erickson & Lee, 2005).

Επίσης η UML δεν έχει την ικανότητα να καθορίζει με ακρίβεια τις απαιτήσεις, όπως είναι οι συμβάσεις λειτουργίας του συστήματος, και να επαληθεύει τη συνέπεια και τη βελτίωση των απαιτήσεων. Αυτά τα μειονεκτήματα έχουν ως αποτέλεσμα τα πιθανά σφάλματα του λογισμικού να είναι δύσκολο να εντοπιστούν στο αρχικό στάδιο της διαδικασίας ανάπτυξης λογισμικού και στη συνέχεια να απαιτείται να καταβληθεί μεγάλη προσπάθεια στις δοκιμές λογισμικού για να εντοπιστούν τα σφάλματα (Yang, Yang & Li, 2018).

Η UML δεν συγκροτεί ένα πλήρως αξιωματικό ή μαθηματικά θεμελιωμένο σύστημα με αυστηρά ορισμένη σημασιολογία, αντίστοιχη με εκείνη που συναντάται σε τυπικές γλώσσες προδιαγραφών (π.χ. βασισμένες σε λογική ή σε θεωρία συνόλων). Αυτό έχει ως αποτέλεσμα η ερμηνεία των διαγραμμάτων να διαφέρει μεταξύ αναλυτών, η δυνατότητα τυπικής επαλήθευσης² να είναι περιορισμένη και να δημιουργείται χάσμα μεταξύ μοντέλου και υλοποίησης. Η θεωρητική κριτική υποστηρίζει ότι η UML λειτουργεί

² Τυπική επαλήθευση είναι μια διαδικασία που χρησιμοποιείται για να αποδείξει ή να απορρίψει την ορθότητα ενός συστήματος, όπως υλικού ή λογισμικού, σε σχέση με μια επίσημη προδιαγραφή, χρησιμοποιώντας μαθηματικές τεχνικές. Διασφαλίζει ότι το σύστημα συμπεριφέρεται όπως προβλέπεται σε όλα τα πιθανά σενάρια, παρέχοντας υψηλότερο επίπεδο εμπιστοσύνης στην αξιοπιστία του.

περισσότερο ως επικοινωνιακό εργαλείο παρά ως αυστηρή γλώσσα προδιαγραφών (Cabot, 2023).

Ακόμα, η εκτεταμένη τυπολογία της UML, η οποία περιλαμβάνει μεγάλο αριθμό διαγραμμάτων (δομικών και συμπεριφορικών), στερεοτύπων και μηχανισμών επέκτασης, δημιουργεί εννοιολογική υπερφόρτωση και ασυνέπειες ή επικαλύψεις μεταξύ διαφορετικών τύπων διαγραμμάτων. Η κριτική αυτή σχετίζεται με το ζήτημα της εσωτερικής συνοχής: μια γλώσσα μοντελοποίησης οφείλει να διατηρεί ισορροπία μεταξύ εκφραστικότητας και οικονομίας εννοιών. Η UML, ιδίως στις μεταγενέστερες εκδόσεις της, έχει κατηγορηθεί για γλωσσική φλυαρία (language bloat), η οποία δυσχεραίνει τη θεωρητική της καθαρότητα (Thomas, 2003).

Επίσης, η UML δεν προέκυψε από ενιαία θεωρητική σύλληψη, αλλά από ενοποίηση επιμέρους αντικειμενοστρεφών μεθοδολογιών της δεκαετίας του 1990 (Booch, OMT, OOSE). Αυτό έχει ως αποτέλεσμα να μη βασίζεται σε μια συνεκτική γλωσσολογική ή οντολογική θεωρία και να συνδυάζει έννοιες που έχουν διαφορετικές φιλοσοφικές αφετηρίες.

Ταυτόχρονα δεν επιλύει ρητά ζητήματα όπως η φύση της αφαίρεσης, η αντιστοίχιση μοντέλου – πραγματικότητας ή η έννοια του αντικειμένου σε οντολογικό επίπεδο. Μάλιστα, από φιλοσοφική σκοπιά, η UML έχει χαρακτηριστεί περισσότερο ως τεχνικό συμβιβαστικό πλαίσιο παρά ως θεωρητικά ενοποιημένο σύστημα αναπαράστασης.

Σε θεωρητικό επίπεδο, η UML έχει δεχθεί κριτική για το πώς προσεγγίζει τη μοντελοποίηση της πραγματικότητας. Η γλώσσα υιοθετεί αντικειμενοστρεφή οντολογία, η οποία ενδέχεται να μην ανταποκρίνεται σε όλα τα είδη συστημάτων (π.χ. κοινωνικά), ενθαρρύνει την αναγωγή σύνθετων φαινομένων σε στατικές δομές κλάσεων και δυσκολεύεται να αποτυπώσει έννοιες όπως η ρευστότητα, η αβεβαιότητα ή η εξελικτική μεταβολή. Οπότε, από επιστημολογική σκοπιά, τίθεται το ερώτημα κατά πόσο η UML αποτελεί ουδέτερο εργαλείο αναπαράστασης ή ενσωματώνει συγκεκριμένες παραδοχές για τη φύση των συστημάτων.

Παρότι τα διαγράμματα ακολουθίας, καταστάσεων και δραστηριοτήτων παρέχουν μηχανισμούς αναπαράστασης δυναμικής συμπεριφοράς, η UML δεν αποτυπώνει με ακρίβεια πολύπλοκες ή μη-ντετερμινιστικές διεργασίες, δεν παρέχει ενιαίο μαθηματικό υπόβαθρο για τη σύνθεση συμπεριφορών και απαιτεί συχνά συμπληρωματικές γλώσσες

(π.χ. OCL) για αυστηρότερους περιορισμούς (Al-Fedaghi, χ.χ.). Η ανάγκη εξωτερικών φορμαλισμών υποδηλώνει ότι η UML, ως αυτόνομο σύστημα, δεν επαρκεί για πλήρη τυπική ανάλυση.

Επειδή η UML υποστηρίζει πολλαπλούς τύπους διαγραμμάτων η διατήρηση της συνέπειας μεταξύ αυτών των προβολών είναι θεωρητικά δύσκολη. Αυτό έχει ως αποτέλεσμα οι αλλαγές σε ένα διάγραμμα να απαιτούν ενημερώσεις και σε άλλα οπότε η διατήρηση της συνοχής μεταξύ των μοντέλων καθίσταται υπολογιστικά και γνωστικά δύσκολη. Συνεπώς η διατήρηση της ακεραιότητας σε πολλαπλά μοντέλα UML μπορεί να καταστεί δυσεπίλυτη λόγω των αλληλοεπιδράσεων των αλλαγών μεταξύ των διαγραμμάτων (Dori, 2002).

Επίσης η UML έχει κατηγορηθεί ότι δημιουργεί κενό μεταξύ μοντέλων και υλοποίησης, διότι ως ανεξάρτητη που είναι από συγκεκριμένη πλατφόρμα και αφηρημένη, συχνά υπάρχει αποσύνδεση μεταξύ μοντέλων και εκτελέσιμου κώδικα (ο όρος "ανεξάρτητη από πλατφόρμα" υποδηλώνει ότι η UML δεν συνδέεται με καμία συγκεκριμένη πλατφόρμα υλικού ή λογισμικού, επιτρέποντας ευελιξία στην αναπαράστασή τους(AlRababah, 2024)). Αυτό έχει επιπτώσεις στα μοντέλα, τα οποία έτσι μπορούν να αποκλίνουν από την υλοποίηση, αλλά και η ανάπτυξη που βασίζεται σε αυτά τα μοντέλα μπορεί να γίνει πιο δύσκολη στην πράξη (Cabot, 2023).

Η UML παρουσιάζεται ως μεθοδολογικά ουδέτερη γλώσσα, δηλαδή δεν επιβάλλει συγκεκριμένη διαδικασία ανάπτυξης λογισμικού. Ωστόσο, αυτή η ουδετερότητα έχει διττή όψη καθώς παρέχει μεν ευελιξία, αλλά ταυτόχρονα στερείται κανονιστικής καθοδήγησης για το πότε και πώς πρέπει να χρησιμοποιείται κάθε διάγραμμα. Από θεωρητική σκοπιά, μια γλώσσα χωρίς σαφές μεθοδολογικό πλαίσιο ενδέχεται να οδηγεί σε αποσπασματική ή επιφανειακή χρήση.

Συγκριτικά με άλλες σύγχρονες μεθόδους μοντελοποίησης, η UML παραμένει ισχυρή. Η δομημένη μοντελοποίηση είναι χρήσιμη για απλές και διαισθητικές αναπαραστάσεις, οι ευέλικτες μέθοδοι προσφέρουν μια πιο ευέλικτη, συνεργατική προσέγγιση, το BPMN είναι κατάλληλο για επιχειρηματικές διαδικασίες με πιο συμπαγή και εξειδικευμένη σημειογραφία, ενώ η SysML επεκτείνει την UML για συστήματα μηχανικής και οι DSLs προσφέρουν εξειδικευμένη εκφραστικότητα για συγκεκριμένα πεδία.

Τέλος, πρέπει να σημειωθεί ότι καμία μέθοδος μοντελοποίησης δεν είναι από μόνη της επαρκής για όλες τις ανάγκες ανάπτυξης λογισμικού και η αποτελεσματική μοντελοποίηση συχνά απαιτεί συνδυασμό εργαλείων και προσεγγίσεων, ανάλογα με τις απαιτήσεις και το πεδίο εφαρμογής.

6.6 Χρήση της UML στην πράξη

Η UML είναι ένα αποτελεσματικό εργαλείο μοντελοποίησης διεπαφών χρηστών για διαδικτυακές εφαρμογές, προσφέροντας σαφήνεια στο ρόλο της ιστοσελίδας. Ποικίλα εργαλεία της UML μπορούν να απλοποιήσουν την πολυπλοκότητα του διαδικτυακού λογισμικού. Παρέχεται έτσι ένα προφίλ προσαρμοσμένο στις συγκεκριμένες ανάγκες της ανάπτυξης ιστού, το οποίο περιλαμβάνει μια συλλογή μοντέλων για διαδικτυακές εφαρμογές όπως: μοντέλο περίπτωσης χρήσης, μοντέλο υλοποίησης, μοντέλο ανάπτυξης, μοντέλο ασφάλειας και χάρτη τοποθεσίας.

Η UML χρησιμοποιείται επίσης για τη μοντελοποίηση ιστοσελίδων, υπερσυνδέσμων και δυναμικού περιεχομένου τόσο στην πλευρά του πελάτη όσο και στον διακομιστή, διακρίνοντας μεταξύ των δύο χρησιμοποιώντας μηχανισμούς επέκτασης, όπως στερεότυπα και εικονίδια. Η UML επιτρέπει τη μοντελοποίηση φορμών HTML και παρέχει τα μέσα για την έκφραση της εκτέλεσης επιχειρηματικής λογικής εντός διαδικτυακών τεχνολογιών και στοιχείων.

Σε συστήματα ηλεκτρονικού εμπορίου, η λογική παραγγελιών, οι ροές πληρωμών και ο συντονισμός αποθεμάτων γίνονται πολύπλοκα χωρίς σωστή μοντελοποίηση. Χρησιμοποιώντας την UML νωρίς, ειδικά τα διαγράμματα κλάσης, ακολουθίας και ανάπτυξης, επιτυγχάνεται η απόκτηση μια κοινής γλώσσας που αποτρέπει την κακή ευθυγράμμιση μεταξύ προϊόντος, μηχανικής και λειτουργιών.

Σε κλινικά περιβάλλοντα όπου η παρερμηνεία μπορεί να επηρεάσει την ασφάλεια των ασθενών, η χρήση της UML είναι καθοριστική. Τα νοσοκομεία δομούν τις ψηφιακές τους λειτουργίες χρησιμοποιώντας UML για να μοντελοποιήσουν τρεις βασικές ενότητες: εγγραφή ασθενών, προγραμματισμό και χρέωση. Ο σχεδιασμός νοσοκομειακών

συστημάτων UML αφορά την κατασκευή μοντέλων που καθοδηγούν την ανάπτυξη, αποτρέπουν σφάλματα και υποστηρίζουν τη συμμόρφωση με τους κανονισμούς.

Κρίσιμη είναι επίσης η χρήση της UML στα συστήματα τραπεζικών συναλλαγών. Κατά το σχεδιασμό μιας τραπεζικής πλατφόρμας, η γραμμή μεταξύ «λειτουργικού» και «ασφαλούς» συχνά εξαρτάται από το πόσο καθαρή είναι η μοντελοποίηση στις μεταβάσεις κινδύνου και κατάστασης. Πολλοί προγραμματιστές ξεκινούν με ένα διάγραμμα κλάσεων γεμάτο λογαριασμούς και συναλλαγές, μόνο και μόνο για να συνειδητοποιήσουν αργότερα ότι η ακολουθία των ελέγχων ελέγχου ταυτότητας, της επικύρωσης συναλλαγών και της λογικής ανίχνευσης απάτης δεν καταγράφηκε καθόλου.

Επίσης σε βιομηχανικά συστήματα υπάρχουν εφαρμογές της UML για τον έλεγχο σε πραγματικό χρόνο. Ένας αισθητήρας σε ένα εργοστάσιο που ελέγχει τη θερμοκρασία, μπορεί να ενεργοποιεί συναγερμούς, να καταγράφει τάσεις και να προσαρμόζει τη συχνότητα αναφοράς του με βάση τις συνθήκες. Η UML παρέχει τα εργαλεία για τη μοντελοποίηση αυτής της συμπεριφοράς με σαφήνεια. Τα διαγράμματα μηχανών καταστάσεων καταγράφουν πώς ένας αισθητήρας μεταβαίνει από την κατάσταση αδράνειας στην κατάσταση συναγερμού.

Η σύνδεση της UML με την έννοια του πραγματικού χρόνου έχει πεδίο δράσης και στις μεταφορές, για τη μοντελοποίηση συστημάτων που πρέπει να λαμβάνουν αποφάσεις σε πραγματικό χρόνο με βάση την τοποθεσία του οχήματος, τη διαθεσιμότητα του οδηγού και την επείγουσα ανάγκη παράδοσης. Τα διαγράμματα ακολουθίας αποτυπώνουν τη λογική του πραγματικού χρόνου πίσω από ένα σύστημα διαχείρισης στόλου.

Η έννοια του πραγματικού χρόνου έχει ακόμα άμεση σχέση και με τα συστήματα διαχείρισης της οδικής κυκλοφορίας, όπου τα «έξυπνα» φανάρια πρέπει να προσαρμόζονται ανάλογα με τις συνθήκες της κυκλοφορίας. Τα διαγράμματα καταστάσεων μοντελοποιούν τις καταστάσεις (χρώματα) των φαναριών και τα διαγράμματα δραστηριοτήτων μοντελοποιούν τη ροή της κυκλοφορίας.

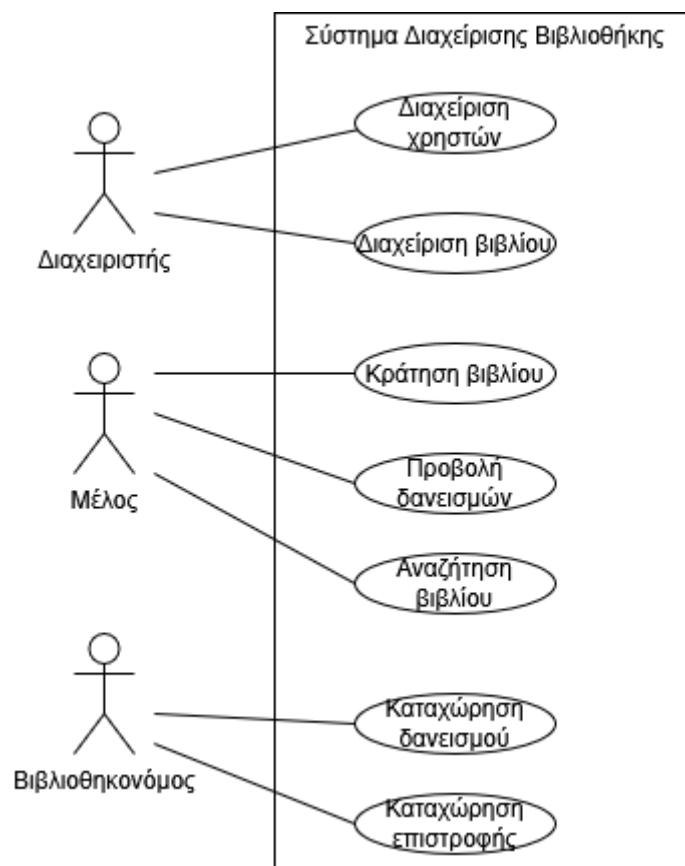
Στα ανώτερα εκπαιδευτικά ιδρύματα τα διαγράμματα της UML μοντελοποιούν συστήματα που διαχειρίζονται διάφορους γραμματειακούς τομείς όπως η εγγραφή των φοιτητών, τις δηλώσεις μαθημάτων και την καταγραφή βαθμολογιών (“Visual Paradigm Skills”, 2026).

7. Παραδείγματα εφαρμογής UML στη σχεδίαση συστημάτων

Στο παρόν κεφάλαιο θα παρουσιαστεί ως παράδειγμα ένα σύστημα διαχείρισης βιβλιοθήκης, μέσα από τα διαγράμματα UML. Σκοπός του παραδείγματος είναι να παρουσιαστεί το πώς διαφορετικά διαγράμματα UML μπορούν να χρησιμοποιηθούν συνδυαστικά για την ανάλυση και τη σχεδίαση ενός απλού αλλά ρεαλιστικού πληροφοριακού συστήματος.

Το σύστημα διαχείρισης βιβλιοθήκης υποστηρίζει βασικές λειτουργίες όπως αναζήτηση βιβλίων, δανεισμό, επιστροφή, κράτηση βιβλίου και διαχείριση χρηστών. Οι βασικοί χρήστες του συστήματος είναι ο αναγνώστης/μέλος, ο βιβλιοθηκονόμος και ο διαχειριστής.

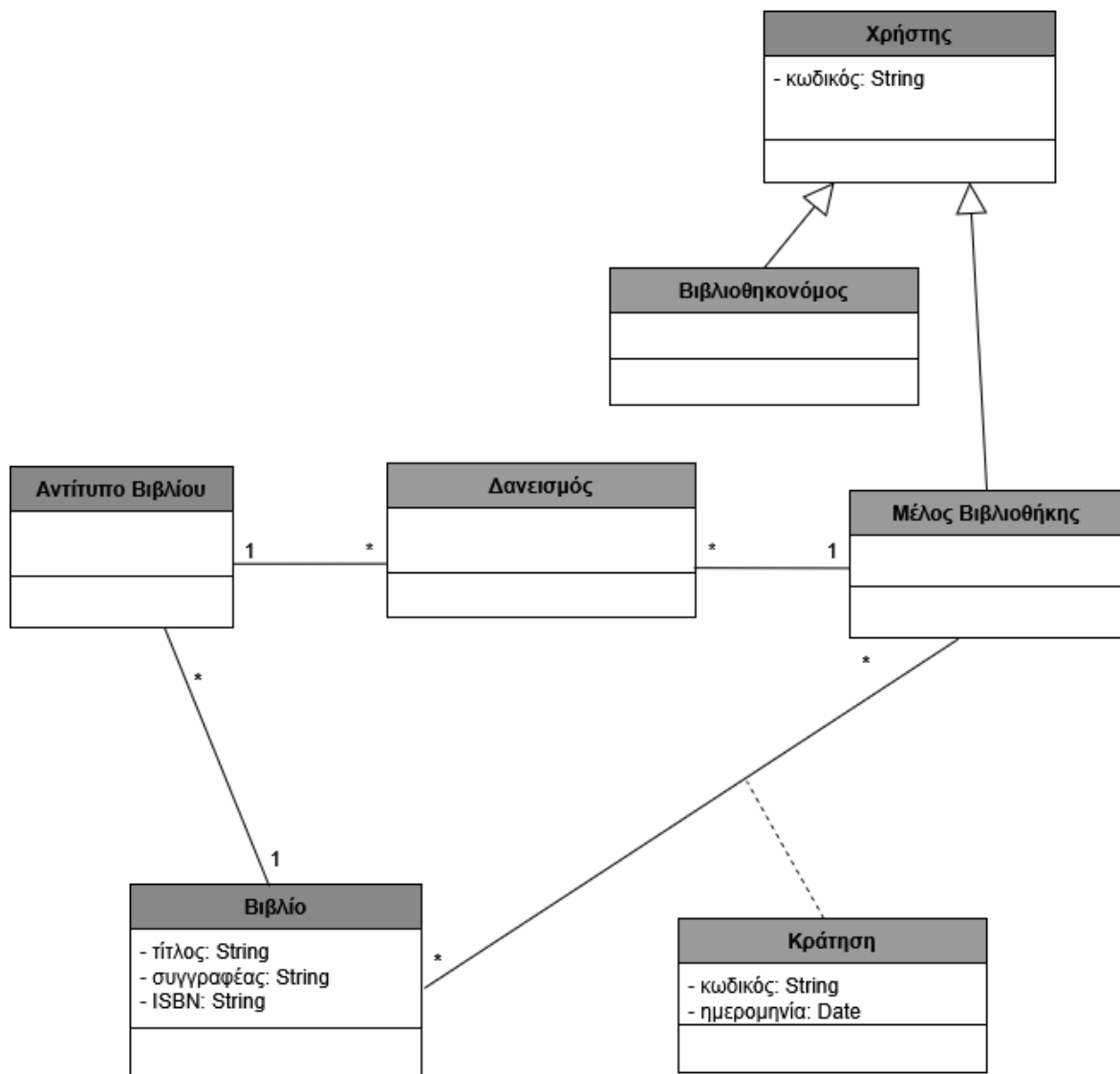
7.1 Διάγραμμα περιπτώσεων χρήσης



Σχήμα 30: Διάγραμμα περιπτώσεων χρήσης.

Στο σχήμα 30, που παριστάνει το διάγραμμα περιπτώσεων χρήσης για το σύστημα διαχείρισης βιβλιοθήκης, οι χειριστές (δρώντες) είναι ο δανειζόμενος, ο βιβλιοθηκονόμος και ο διαχειριστής συστήματος. Το μέλος μπορεί να αναζητήσει βιβλία, να κάνει κράτηση και να δει τους ενεργούς δανεισμούς του. Ο βιβλιοθηκονόμος μπορεί να καταχωρίσει δανεισμό, να καταχωρίσει επιστροφή και να ενημερώσει τη διαθεσιμότητα βιβλίων. Ο διαχειριστής μπορεί να διαχειρίζεται χρήστες και βιβλιογραφικά δεδομένα.

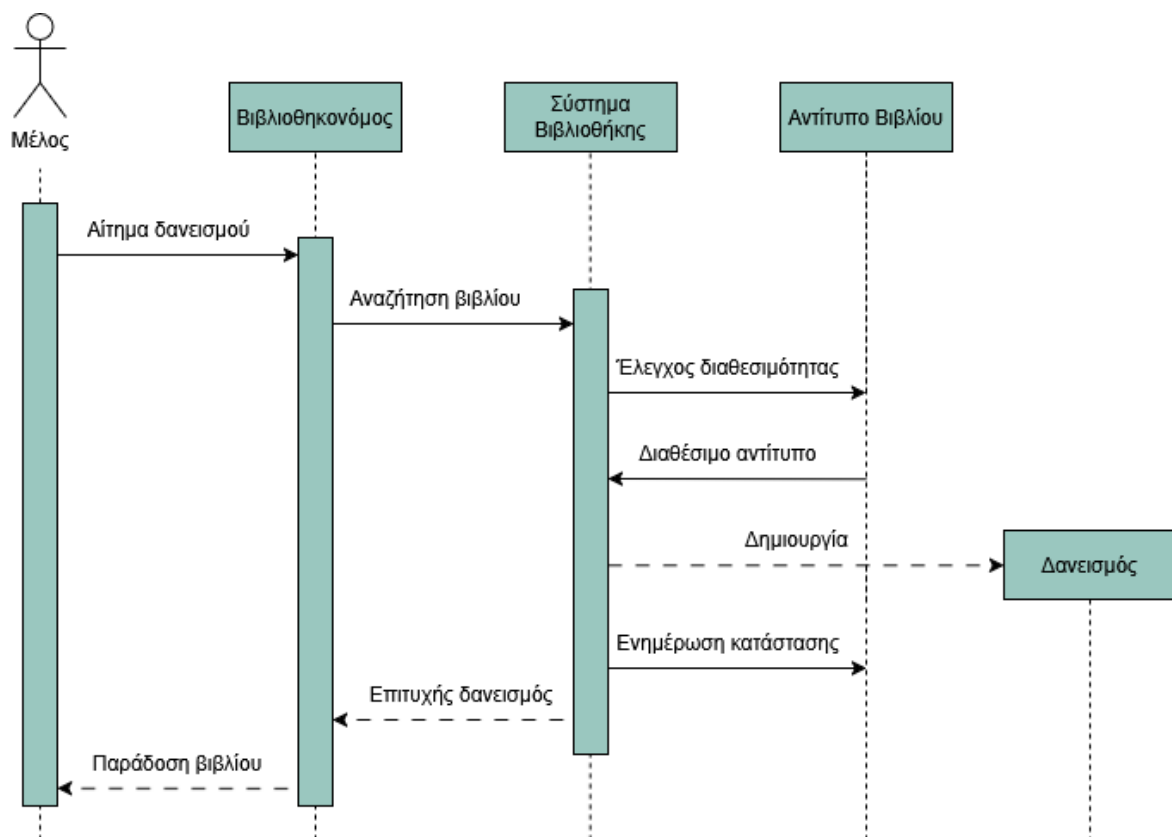
7.2 Διάγραμμα κλάσεων



Σχήμα 31: Διάγραμμα κλάσεων.

Στο σχήμα 31 απεικονίζεται ένα διάγραμμα κλάσεων για το σύστημα. Η κλάση Βιβλίο περιγράφει τα βιβλιογραφικά στοιχεία ενός βιβλίου, όπως τίτλο, συγγραφέα και ISBN. Η κλάση Αντίτυπο Βιβλίου αναπαριστά ένα φυσικό αντίτυπο του βιβλίου, καθώς ένα βιβλίο μπορεί να υπάρχει σε περισσότερα από ένα αντίτυπα. Η κλάση Δανεισμός καταγράφει έναν δανεισμό, συνδέοντας ένα μέλος με ένα αντίτυπο βιβλίου. Η κλάση Κράτηση καταγράφει την κράτηση ενός βιβλίου από μέλος.

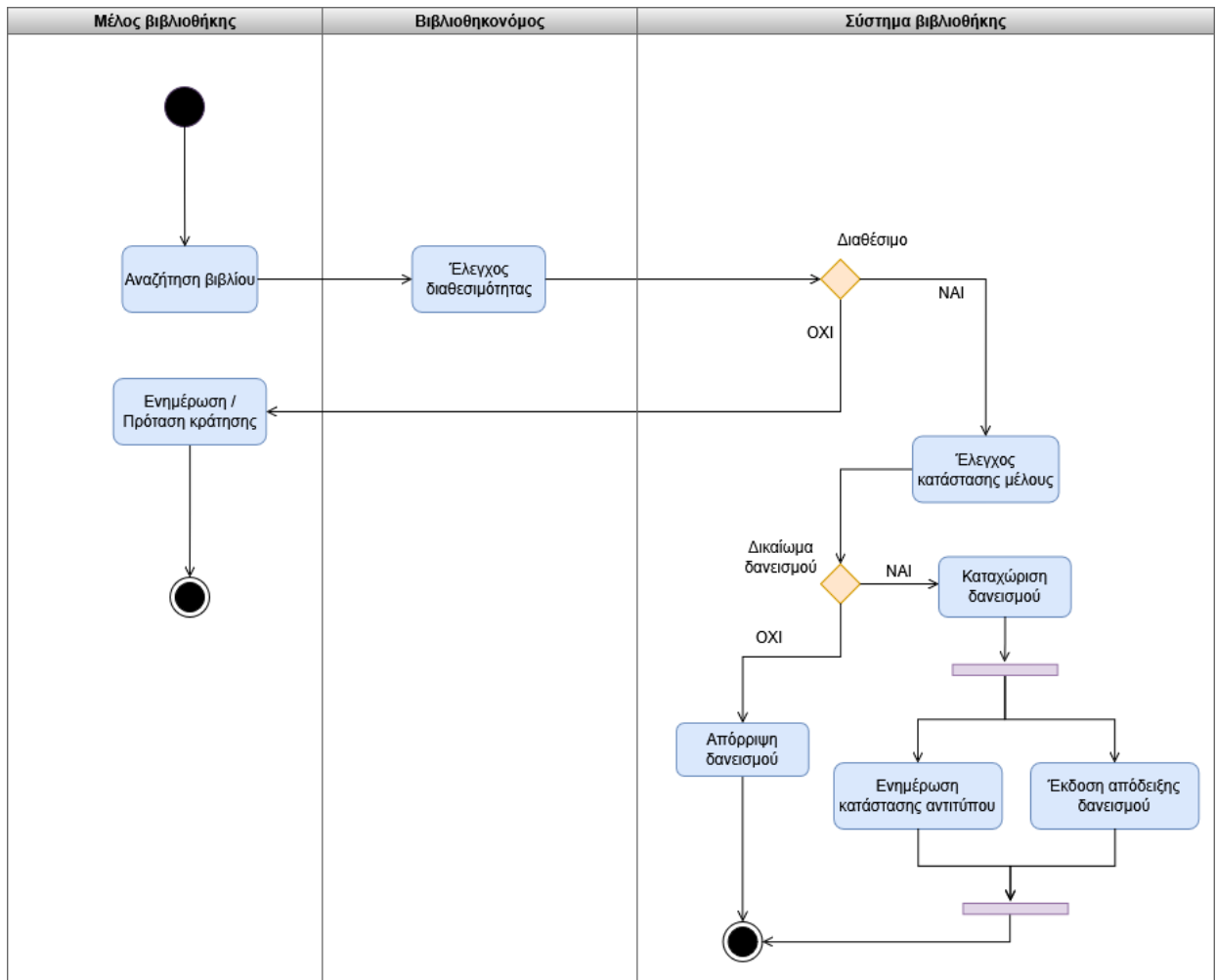
7.3 Διάγραμμα ακολουθίας



Σχήμα 32: Διάγραμμα ακολουθίας.

Στο σχήμα 32 απεικονίζεται ένα διάγραμμα ακολουθίας για τη διαδικασία δανεισμού βιβλίου. Το μέλος ζητά να δανειστεί ένα βιβλίο. Ο βιβλιοθηκονόμος αναζητά το βιβλίο στο σύστημα, ελέγχει αν υπάρχει διαθέσιμο αντίτυπο και καταχωρίζει τον δανεισμό.

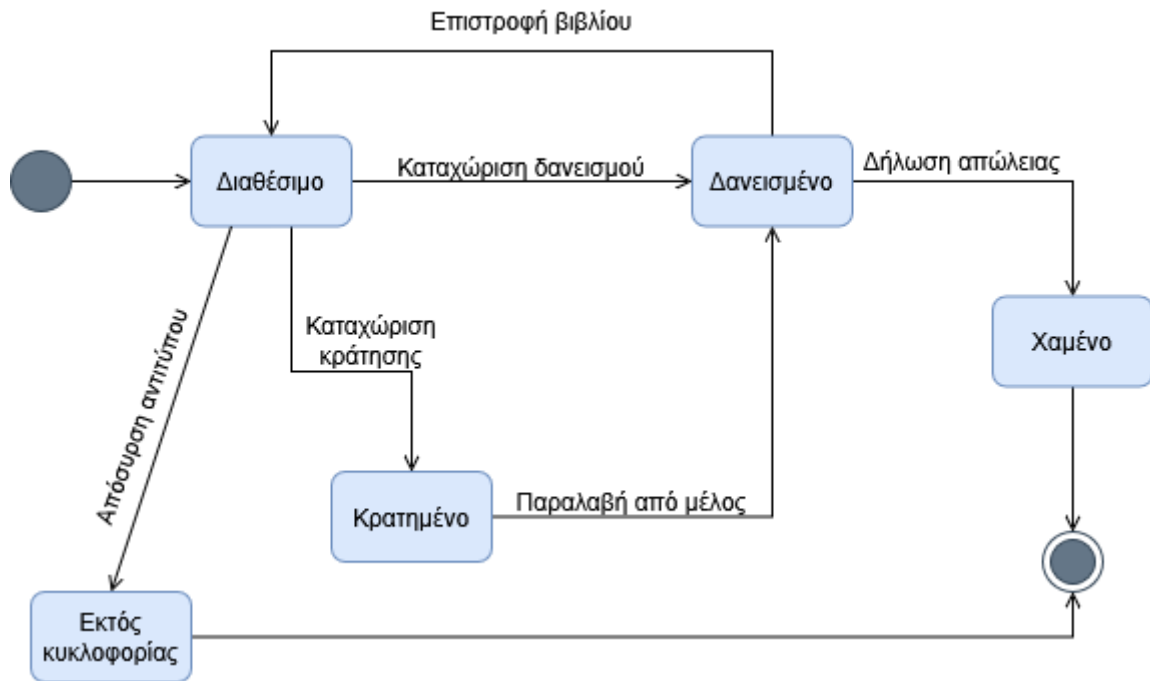
7.4 Διάγραμμα δραστηριοτήτων



Σχήμα 33: Διάγραμμα δραστηριοτήτων.

Στο σχήμα 33 απεικονίζεται ένα διάγραμμα δραστηριοτήτων για τη διαδικασία δανεισμού βιβλίου. Το μέλος αναζητά να δανειστεί ένα βιβλίο. Ο βιβλιοθηκονόμος ελέγχει τη διαθεσιμότητα του βιβλίου στο σύστημα, και ανάλογα με τη διαθεσιμότητα ελέγχει και το δικαίωμα δανεισμού του μέλους και καταχωρίζει τον δανεισμό, ενημερώνοντας την κατάσταση του αντιτύπου και εκδίδοντας απόδειξη δανεισμού.

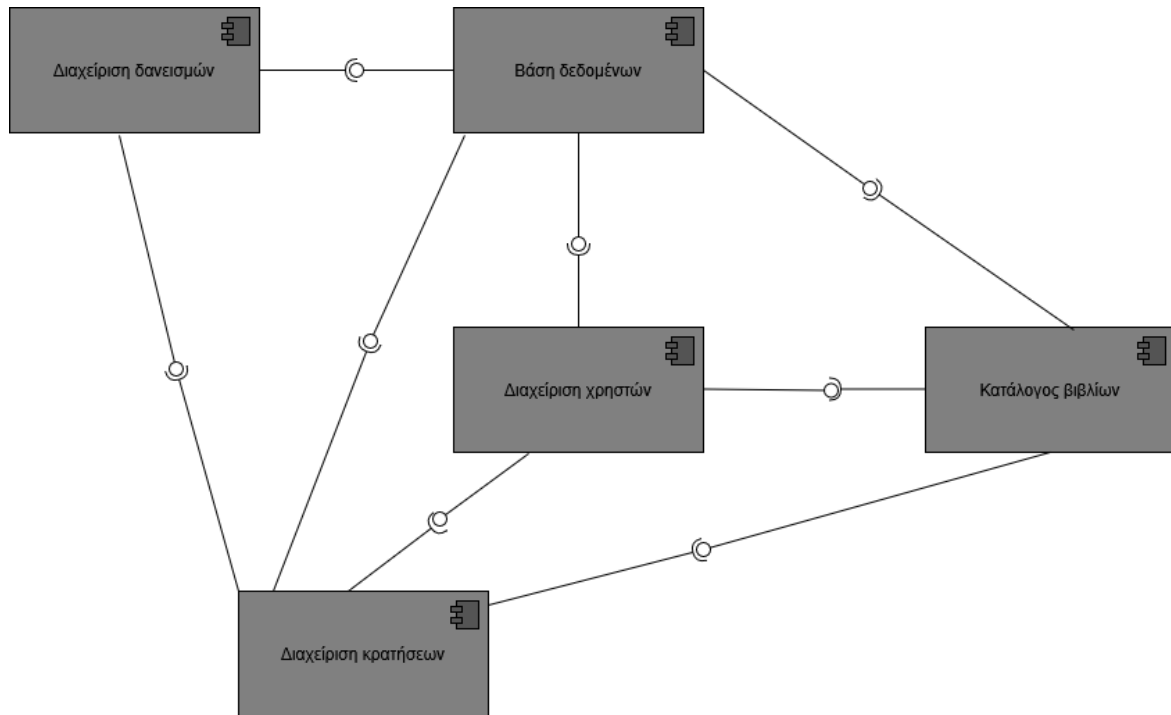
7.5 Διάγραμμα καταστάσεων



Σχήμα 34: Διάγραμμα καταστάσεων.

Το διάγραμμα καταστάσεων του σχήματος 34 απεικονίζει τις διαφορετικές καταστάσεις στις οποίες μπορεί να βρίσκεται ένα αντίτυπο βιβλίου κατά τη διάρκεια του κύκλου ζωής του.

7.6 Διάγραμμα συστατικών



Σχήμα 35: Απλό διάγραμμα συστατικών στοιχείων.

Το σύστημα μπορεί να οργανωθεί σε βασικά συστατικά μέρη, όπως φαίνεται στο σχήμα 35. Το συστατικό «Διαχείριση χρηστών» αναλαμβάνει τη διαχείριση των χρηστών του συστήματος, το συστατικό «Κατάλογος βιβλίων» αναλαμβάνει τη διαχείριση των βιβλίων και των αντιτύπων τους, το συστατικό «Διαχείριση δανεισμών» αναλαμβάνει τη διαχείριση δανεισμών και αντιτύπων των βιβλίων, το συστατικό «Διαχείριση κρατήσεων» διαχειρίζεται τις κρατήσεις και το συστατικό «Βάση δεδομένων» αποθηκεύει τα δεδομένα από όλες τις διαδικασίες δηλαδή ουσιαστικά σχετίζεται με όλα τα άλλα συστατικά.

7.7 Συμπέρασμα

Το παράδειγμα του συστήματος βιβλιοθήκης δείχνει ότι τα διαγράμματα UML μπορούν να λειτουργήσουν συμπληρωματικά. Το διάγραμμα περιπτώσεων χρήσης αποτυπώνει τις λειτουργικές απαιτήσεις, το διάγραμμα κλάσεων περιγράφει τη στατική δομή, ενώ τα

διαγράμματα ακολουθίας και δραστηριοτήτων παρουσιάζουν τη δυναμική συμπεριφορά του συστήματος. Επιπλέον, το διάγραμμα καταστάσεων βοηθά στην κατανόηση του κύκλου ζωής ενός αντιτύπου βιβλίου, ενώ το διάγραμμα συστατικών αποτυπώνει την αρχιτεκτονική διάρθρωση του συστήματος. Έτσι, η UML συμβάλλει στη μετάβαση από τις απαιτήσεις στη σχεδίαση, παρέχοντας μια κοινή γλώσσα κατανόησης και τεκμηρίωσης για όλους τους εμπλεκόμενους.

8. Αξιολόγηση της συμβολής των UML διαγραμμάτων

8.1 Κριτήρια αξιολόγησης μοντελοποίησης

Τα υψηλής ποιότητας εννοιολογικά μοντέλα είναι κρίσιμης σημασίας για την επιτυχή εξέλιξη ενός συστήματος. Παρ' όλη αυτή τη σπουδαιότητα, πρακτικά δεν υπάρχουν ποσοτικές μέθοδοι για την αξιολόγηση της ποιότητας τους. Ακόμα και οι ορισμοί της ποιότητας - όταν αυτοί δίνονται - είναι ασαφείς και περίπλοκοι και δεν υπάρχει βασική δομή που να βοηθάει το χρήστη να κατανοήσει το πώς σχετίζονται οι ιδιότητες μεταξύ τους.

Η αξιολόγηση της συμβολής των διαγραμμάτων της UML προϋποθέτει την υιοθέτηση ρητών κριτηρίων ποιότητας μοντελοποίησης. Στη διεθνή βιβλιογραφία έχουν προταθεί διάφορα πλαίσια αξιολόγησης, όπως το εννοιολογικό πλαίσιο ποιότητας μοντέλων των Lindland, Sindre και Sølvberg (1994), αλλά σε αυτή την εργασία θα επικεντρώσουμε στο SEQUAL (SEmiotic QUALity), ένα πλαίσιο βασισμένο στη σημειωτική, για την κατανόηση των διαφορετικών πτυχών της ποιότητας των εννοιολογικών μοντέλων και των γλωσσών μοντελοποίησης.

Εδώ ο όρος «ποιότητα» αντιμετωπίζεται και χρησιμοποιείται σε διάφορα επίπεδα. Έτσι λοιπόν έχουμε ποιότητα. Η συμπερίληψη της δεοντολογικής ποιότητας (παλαιότερα ονομαζόταν «οργανωσιακή ποιότητα») έχει ως στόχο να ληφθούν καλύτερα υπόψη οι κυρίαρχοι στόχοι της μοντελοποίησης σε μια εργασία μοντελοποίησης, που συχνά αποφασίζονται σε ένα ευρύτερο οργανωσιακό πλαίσιο.

Στην συντακτική, σημασιολογική και πραγματολογική ποιότητα υπάρχουν δύο κύριες έννοιες: η πληρότητα και η εγκυρότητα. Ένα μοντέλο είναι πλήρες όταν περιέχει όλα τα στοιχεία ενός τομέα και είναι έγκυρο όταν δεν περιέχει στοιχεία που δεν υπάρχουν στον τομέα.

Οι κύριες έννοιες και οι σχέσεις τους παρουσιάζονται στο Σχήμα 36. Οι τύποι ποιότητας για ένα εννοιολογικό μοντέλο υποδεικνύονται στο σχήμα με συνεχείς γραμμές, που υποδεικνύουν τις σχέσεις μεταξύ συνόλων, τα οποία απεικονίζονται ως πλαίσια.



Αν και τα μοντέλα συστημάτων πληροφοριών δεν είναι συνήθως φυσικού (τρισδιάστατου, απτικού) είδους, οποιοδήποτε μοντέλο πρέπει να αναπαρίσταται φυσικά με κάποιο τρόπο, π.χ. σε δίσκο, σε χαρτί ή σε μαυροπίνακα. Τα βασικά χαρακτηριστικά ποιότητας σε φυσικό επίπεδο είναι ότι το εξωτερικευμένο μοντέλο πρέπει να είναι επίμονο, ενημερωμένο και διαθέσιμο, επιτρέποντας στο κοινό να το κατανοήσει (και ενδεχομένως στους δημιουργούς μοντέλων να το αλλάξουν όταν είναι απαραίτητο).

Διπλωματική Εργασία

Με τον όρο ενημερωμένο, εννοείται πόσο καιρό πριν συμπεριλήφθηκαν οι δηλώσεις μοντέλου σε αυτό (υποθέτοντας ότι οι δηλώσεις ήταν ενημερωμένες κατά την εισαγωγή). Ανάλογα με τον τύπο του μοντέλου, η ηλικία των δηλώσεων του μοντέλου έχει διαφορετική σημασία. Όταν το πεδίο αλλάζει ραγδαία (έχει υψηλή μεταβλητότητα), η αξία της επικαιρότητας (του κύρους) του αποθηκευμένου μοντέλου έχει μεγαλύτερη σημασία για το μοντέλο. Αυτό είναι ιδιαίτερα επείγον για μοντέλα σε επίπεδο στιγμιοτύπου (για παράδειγμα, δεδομένα σε μια βάση δεδομένων, αλλά επίσης πολλά επιχειρηματικά μοντέλα βρίσκονται εν μέρει σε επίπεδο στιγμιοτύπου).

Με τον όρο διαθέσιμο, εννοείται πόσο εύκολα διαθέσιμο είναι το μοντέλο στο κοινό. Σαφώς, αυτό εξαρτάται από το αν το μοντέλο εξωτερικεύεται και γίνεται επίμονο εξαρχής. Η διαθεσιμότητα εξαρτάται επίσης από τη δυνατότητα διανομής, ειδικά όταν τα μέλη του κοινού είναι γεωγραφικά διασκορπισμένα. Ένα μοντέλο που είναι σε ηλεκτρονικά διανεμήσιμη μορφή θα διανεμηθεί πιο εύκολα από ένα που πρέπει να εκτυπωθεί σε χαρτί (ή είναι μόνο σε χαρτί εξαρχής) και να αποσταλεί μέσω ταχυδρομείου ή φαξ. Μπορεί επίσης να έχει σημασία τι ακριβώς διανέμεται, π.χ. το μοντέλο σε επεξεργάσιμη μορφή ή απλώς σε μορφή εξόδου, ή αλλιώς σε μια μορφή στην οποία μπορούν να προστεθούν σχόλια, αλλά δεν μπορεί να αλλάξει το πραγματικό μοντέλο.

8.1.2 Εμπειρική Ποιότητα

Η εμπειρική ποιότητα ασχολείται με την ποικιλία των στοιχείων που διακρίνονται, τις συχνότητες σφαλμάτων κατά τη σύνταξη ή την ανάγνωση, την κωδικοποίηση (συμπεριλαμβανομένων των σχημάτων των πλαισίων) και την εργονομία τόσο για τα εργαλεία τεκμηρίωσης όσο και για τα εργαλεία μοντελοποίησης. Ουσιαστικά είναι η σχέση μεταξύ του μοντέλου και ενός άλλου μοντέλου που περιέχει τις ίδιες δηλώσεις και θεωρείται κατά κάποιο τρόπο καλύτερο μέσω διαφορετικής διάταξης ή εμφάνισης.

8.1.3 Συντακτική Ποιότητα

Υπάρχει μόνο ένα χαρακτηριστικό συντακτικής ποιότητας, η συντακτική ορθότητα, που σημαίνει ότι όλες οι δηλώσεις στο μοντέλο είναι σύμφωνες με τη σύνταξη και το λεξιλόγιο της γλώσσας μοντελοποίησης. Τα συντακτικά σφάλματα είναι δύο ειδών:

- Η συντακτική ακυρότητα, στην οποία χρησιμοποιούνται λέξεις ή γραφήματα που δεν αποτελούν μέρος της γλώσσας.
- Συντακτική ατέλεια, στην οποία το μοντέλο δεν διαθέτει δομές ή πληροφορίες που να υπακούουν στη γραμματική της γλώσσας.

8.1.4 Σημασιολογική Ποιότητα

Η σημασιολογική ποιότητα ορίστηκε αρχικά ως η αντιστοιχία μεταξύ του μοντέλου και του τομέα μοντελοποίησης (Lindland et al. 1994). Το πλαίσιο περιέχει δύο κύρια χαρακτηριστικά σημασιολογικής ποιότητας: την εγκυρότητα και την πληρότητα. Εγκυρότητα σημαίνει ότι όλες οι δηλώσεις που γίνονται στο μοντέλο θεωρούνται σωστές και σχετικές με το πρόβλημα και πληρότητα σημαίνει ότι το μοντέλο περιέχει όλες τις δηλώσεις που θα ήταν σωστές και σχετικές με το πεδίο μοντελοποίησης.

Ο πρωταρχικός στόχος για τη σημασιολογική ποιότητα είναι η αντιστοιχία μεταξύ του εξωτερικευμένου μοντέλου και του πεδίου, αλλά αυτή η αντιστοιχία δεν μπορεί ούτε να διαπιστωθεί ούτε να ελεγχθεί άμεσα. Επομένως, αυτό που παρατηρείται στον έλεγχο ποιότητας δεν είναι η πραγματική σημασιολογική ποιότητα του μοντέλου, αλλά αυτό που ονομάζεται αντιληπτή (perceived) σημασιολογική ποιότητα με βάση τις συγκρίσεις της τρέχουσας γνώσης με την τρέχουσα ερμηνεία του μοντέλου.

Συνοψίζοντας, σημασιολογική ποιότητα είναι η σχέση μεταξύ του μοντέλου και του πεδίου μοντελοποίησης και αντιληπτή σημασιολογική ποιότητα είναι η παράλληλη σχέση μεταξύ της γνώσης των συμμετεχόντων και της ερμηνείας του μοντέλου από αυτούς.

8.1.5 Πραγματολογική Ποιότητα

Η πραγματολογική ποιότητα σχετίζεται με την κατανόηση του μοντέλου από τους συμμετέχοντες. Μπορούν να διακριθούν δύο πτυχές:

- Η ερμηνεία του μοντέλου από τους ανθρώπους που εμπλέκονται είναι σωστή σε σχέση με αυτό που προορίζεται να εκφραστεί στο μοντέλο. Να σημειωθεί ότι ένα μοντέλο μπορεί να θεωρηθεί ότι σημαίνει τυπικά οτιδήποτε μόνο εάν η σύνταξη και η σημασιολογία της γλώσσας που χρησιμοποιείται είναι διωποκειμενικά

συμφωνημένες και είναι τουλάχιστον ημιτυπικές, αλλά κατά προτίμηση τυπικές. Έτσι, μπορεί κανείς να εντοπίσει ή να εκτελέσει το μοντέλο και να βιώσει τη δυναμική συμπεριφορά του.

- Η ερμηνεία του εργαλείου είναι σωστή σε σχέση με αυτό που προορίζεται να εκφραστεί στο μοντέλο.

Ξεκινώντας από το κομμάτι της ανθρώπινης κατανόησης, η πραγματολογική ποιότητα σε αυτό το επίπεδο είναι η αντιστοιχία μεταξύ του μοντέλου και της ερμηνείας του από το κοινό. Ούτε καν το πιο λαμπρό μοντέλο δεν θα είχε καμία χρησιμότητα αν κανείς δεν ήταν σε θέση να το καταλάβει. Επιπλέον, δεν είναι μόνο σημαντικό να έχει γίνει κατανοητό το μοντέλο, αλλά και ποιος έχει κατανοήσει (τα σχετικά μέρη) του.

Ένα εννοιολογικό μοντέλο μπορεί να είναι δύσκολο στην κατανόηση λόγω της τυπικότητας ή της έλλειψης εξοικείωσης με τη γλώσσα μοντελοποίησης που χρησιμοποιείται, της πολυπλοκότητας ή του μεγέθους του μοντέλου ή της προσπάθειας που απαιτείται για την εξαγωγή σημαντικών ιδιοτήτων του. Ένα περιβάλλον εννοιολογικής μοντελοποίησης μπορεί να χρησιμοποιεί ορισμένες τεχνικές για να βελτιώσει την κατανόηση του χρήστη.

Συνοψίζοντας, η πραγματολογική ποιότητα αφορά τη σχέση μεταξύ του μοντέλου και της ερμηνείας του μοντέλου από τα ενδιαφερόμενα μέρη. Σχετίζεται με το κατά πόσο το κοινό κατανοεί τις επιπτώσεις του μέρους του μοντέλου που το αφορά.

8.1.6 Κοινωνική Ποιότητα

Ο στόχος που ορίζεται για την κοινωνική ποιότητα είναι η «συμφωνία», σύμφωνα με τις ακόλουθες διαστάσεις:

- Συμφωνία στη γνώση έναντι συμφωνίας στην ερμηνεία του μοντέλου. Στην περίπτωση όπου δύο μοντέλα κατασκευάζονται με βάση την άποψη δύο διαφορετικών παραγόντων, μπορούμε επίσης να μιλήσουμε για συμφωνία στο μοντέλο.
- Σχετική συμφωνία έναντι απόλυτης συμφωνίας. Σχετική συμφωνία σημαίνει ότι τα διάφορα σύνολα που πρόκειται να συγκριθούν είναι συνεπή, επομένως, μπορεί να υπάρχουν πολλές δηλώσεις στο μοντέλο ενός χρήστη, που δεν υπάρχουν σε αυτό

ενός άλλου, εφόσον δεν έρχονται σε αντίθεση μεταξύ τους. Απόλυτη συμφωνία, από την άλλη πλευρά, σημαίνει ότι όλα τα μοντέλα είναι τα ίδια.

Δεδομένου ότι διαφορετικοί συμμετέχοντες θα έχουν την εμπειρία τους σε διαφορετικούς τομείς, η σχετική συμφωνία θεωρείται πιο χρήσιμη από την απόλυτη συμφωνία. Από την άλλη πλευρά, οι διαφορετικοί χρήστες πρέπει να έχουν την ευκαιρία να συμφωνήσουν και να διαφωνήσουν σε κάτι, δηλαδή τα μέρη του μοντέλου που τους αφορούν θα πρέπει να επικαλύπτονται σε κάποιο βαθμό.

Ο πραγματολογικός στόχος της κατανόησης θεωρείται ως κοινωνικό μέσο. Αυτό συμβαίνει επειδή η συμφωνία χωρίς κατανόηση δεν είναι πολύ χρήσιμη, τουλάχιστον όχι όταν υπάρχουν δημοκρατικά ιδανικά. Προφανώς, αν κάποιος προσπαθεί να χειραγωγήσει μια κατάσταση, η συμφωνία χωρίς κατανόηση είναι «χρήσιμη».

Συνοψίζοντας, κοινωνική ποιότητα είναι η σχέση μεταξύ διαφορετικών ερμηνειών μοντέλων. Δείχνει κατά πόσο συμφωνούν οι διαφορετικοί συμμετέχοντες στη μοντελοποίηση, ως προς τη σημασιολογική ποιότητα του μοντέλου.

8.1.7 Δεοντολογική Ποιότητα

Η μοντελοποίηση (συνήθως) δεν γίνεται για διασκέδαση, αλλά για την επίτευξη κάποιου στόχου, ο οποίος συνήθως συνδέεται με ορισμένους επιχειρηματικούς και οργανωτικούς στόχους. Αυτό εισάγει την ανάγκη να εξεταστεί τόσο το κόστος όσο και το όφελος της μοντελοποίησης. Τα μέσα εδώ σχετίζονται με τη μοντελοποίηση αυτών των στόχων και τον έλεγχο της εκπλήρωσής τους (δηλαδή, ότι όλοι οι στόχοι επιτυγχάνονται και ότι δεν υπάρχουν στόχοι που δεν επιτυγχάνονται).

Για οτιδήποτε άλλο εκτός από εξαιρετικά απλούς και σε μεγάλο βαθμό διωποκειμενικά συμφωνημένους τομείς, η πλήρης εγκυρότητα, πληρότητα, κατανόηση και συμφωνία όπως περιγράφονται παραπάνω δεν μπορούν να επιτευχθούν. Επομένως, για να είναι ρεαλιστικοί οι στόχοι σε αυτούς τους τομείς, πρέπει να είναι κάπως χαλαροί, εισάγοντας την ιδέα της «εφικτότητας».

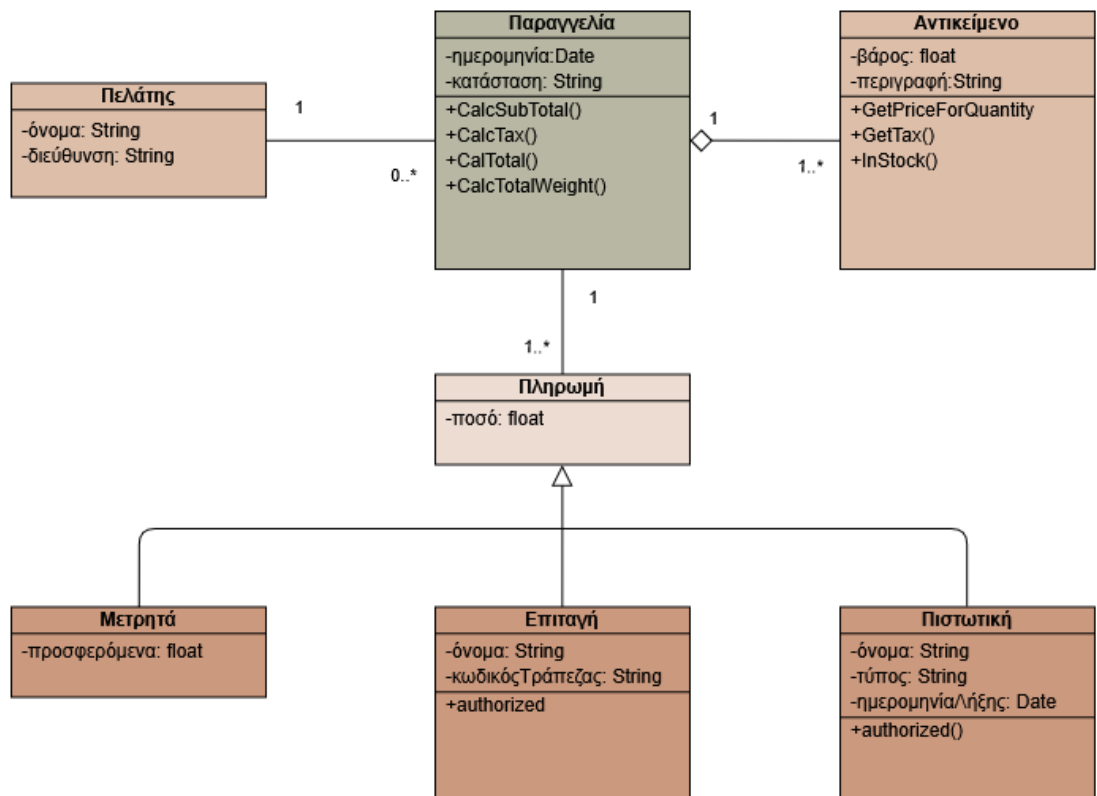
Η εφικτότητα εισάγει έτσι μια αντιστάθμιση μεταξύ της αξίας και των μειονεκτημάτων για την επίτευξη μιας δεδομένης ποιότητας μοντέλου. Χρησιμοποιήσαμε τον όρο

«μειονέκτημα» εδώ αντί για τον πιο συνηθισμένο «κόστος» για να υποδείξουμε ότι η συζήτηση δεν περιορίζεται απαραίτητα σε καθαρά οικονομικά ζητήματα.

Συνοψίζοντας, η δεοντολογική ποιότητα αφορά το πώς τα μοντέλα συμβάλλουν στην εκπλήρωση των συνολικών στόχων της μοντελοποίησης.

8.1.8 Εφαρμογή κριτηρίων αξιολόγησης

Θα δούμε πώς εφαρμόζονται τα κριτήρια αξιολόγησης σε ένα τυπικό διάγραμμα κλάσεων, όπως αυτό φαίνεται στο σχήμα 37, για ένα σύστημα παραγγελιών / πελατών.



Σχήμα 37:Διάγραμμα κλάσεων για σύστημα παραγγελιών.

Όσον αφορά τη συντακτική ορθότητα, αυτή εξετάζει αν το διάγραμμα ακολουθεί σωστά τη σύνταξη της UML. Οι συμβολισμοί είναι σωστοί, οι κλάσεις χρησιμοποιούνται σωστά, η σχέση γενίκευσης εμφανίζεται με το σωστό βελάκι, η σχέση συνάθροισης με το λευκό ρόμβο, οι μέθοδοι βρίσκονται στο κάτω τμήμα του ορθογωνίου και οι πολλαπλότητες είναι σωστά τοποθετημένες.

Για τη σημασιολογική ποιότητα, σύμφωνα με το SEQUAL, το μοντέλο πρέπει να αναπαριστά σωστά το πρόβλημα. Εδώ το διάγραμμα περιγράφει λογικά ότι οι πελάτες κάνουν παραγγελίες, οι παραγγελίες περιέχουν αντικείμενα και οι πληρωμές εξειδικεύονται σε διαφορετικούς τύπους.

Ένα πρόβλημα που εντοπίζεται είναι η σχέση συνάθροισης μεταξύ παραγγελίας και αντικειμένου καθώς δεν μπορεί να υπάρχει αντικείμενο παραγγελίας ανεξάρτητα από την παραγγελία, οπότε χρειάζεται σχέση σύνθεσης αντί για συνάθροιση, οπότε ο ρόμβος θα πρέπει να γίνει μαύρος. Επίσης δεν είναι ξεκάθαρο αν μια παραγγελία έχει ακριβώς μία πληρωμή ή επιτρέπονται πολλές πληρωμές.

Για την πραγματολογική ποιότητα το μοντέλο θα πρέπει να είναι κατανοητό από τους διάφορους χρήστες. Το διάγραμμα έχει απλά ονόματα για τις κλάσεις, περιορισμένο αριθμό κλάσεων, σαφή κληρονομική ιεράρχηση και γενικά έχει σωστή οπτική διάταξη. Η υπερκλάση Πληρωμή βοηθά για να καταλάβει ο αναγνώστης ότι υπάρχουν πολλοί τρόποι πληρωμής που έχουν όμως κοινή συμπεριφορά. Από την άλλη η σχέση συνάθροισης ενδεχομένως να μπερδέψει μη έμπειρους χρήστες.

8.2 Πώς τα διαγράμματα βελτιώνουν την κατανόηση του συστήματος

Τα διαγράμματα UML παρέχουν μια οπτική αναπαράσταση της δομής και συμπεριφοράς του συστήματος. Μεταφράζουν αφηρημένες ιδέες σε οπτικές δομές που είναι ευκολότερο να κατανοηθούν από διαφορετικούς ενδιαφερόμενους (σχεδιαστές, αναλυτές, πελάτες). Μέσω των διαγραμμάτων κλάσεων παρουσιάζονται οι δομές δεδομένων, οι σχέσεις και οι ιδιότητες. Μέσω των διαγραμμάτων ακολουθίας απεικονίζονται οι ανταλλαγές μηνυμάτων μεταξύ αντικειμένων με την πάροδο του χρόνου. Με τα διαγράμματα δραστηριοτήτων φαίνονται οι ροές εργασίας και οι λογικές αποφάσεις.

Αποσαφηνίζουν τις απαιτήσεις και τον σχεδιασμό, αφού πριν ακόμα ξεκινήσει η ανάπτυξη του λογισμικού, βοηθούν στη διατύπωση των απαιτήσεων και την αποφυγή της ασάφειας. Με αυτόν τον τρόπο οι ομάδες μπορούν να συμφωνήσουν σχετικά με τη λειτουργικότητα του συστήματος και οι αλλαγές μπορούν να εντοπιστούν και να

αξιολογηθούν πιο νωρίς. Αυτό έχει ως συνέπεια να μειώνονται οι εσφαλμένες ερμηνείες και τα επακόλουθα λάθη κατά την υλοποίηση. Ειδικότερα τα διαγράμματα περιπτώσεων χρήσης καθορίζουν τα όρια του συστήματος, προσδιορίζουν τους ρόλους (actors) και διευκρινίζουν τις λειτουργικές απαιτήσεις.

Λειτουργούν ως κοινή γλώσσα επικοινωνίας μεταξύ αναλυτών, προγραμματιστών, διευθυντών έργου και πελατών / χρηστών, διότι η οπτική απεικόνιση που παρέχουν μειώνει τις παρερμηνείες και διευκολύνει τη συνεργασία. Ακόμα και μη τεχνικοί χρήστες μπορούν να κατανοήσουν βασικές πτυχές των διαγραμμάτων αυτών, οπότε διευκολύνεται η συνεργασία.

Δίνουν τη δυνατότητα για ευκολότερη τεκμηρίωση και συντήρηση αφού τεκμηριώνουν τον σχεδιασμό και την αρχιτεκτονική ενός συστήματος. Αυτό είναι πολύτιμο για την περίπτωση μελλοντικής συντήρησης, για την εξέταση των εξαρτήσεων του συστήματος, για το ενδεχόμενο μελλοντικής επέκτασης και την διευκόλυνση νέων προγραμματιστών που μπορεί να ενταχθούν μελλοντικά στην ομάδα εργασίας.

Αναλύουν πολύπλοκες διεργασίες, εντοπίζουν ελλείψεις στη σχεδίαση και αναδεικνύουν λανθασμένες συσχετίσεις. Ειδικότερα μέσω των διαγραμμάτων ακολουθίας και δραστηριοτήτων, μπορεί κάποιος να κατανοήσει τις αλληλεπιδράσεις μεταξύ στοιχείων, να διαχωρίσει χρονικές και λογικές εξαρτήσεις και να εντοπίσει ενεργά σημεία συμφόρησης. Αυτό είναι κρίσιμο για τον εντοπισμό σφαλμάτων πριν την υλοποίηση. Έτσι μπορεί να χρειαστούν λιγότερες ή και μηδαμινές διορθώσεις, οπότε μειώνεται και το κόστος.

8.3 Αποτελεσματικότητα στη θεωρητική σχεδίαση λογισμικού

Η αποτελεσματικότητα των διαγραμμάτων UML στη θεωρητική σχεδίαση λογισμικού σχετίζεται με την ικανότητά τους να λειτουργούν ως τυποποιημένο, αφηρημένο και αναλυτικό εργαλείο μοντελοποίησης πριν την υλοποίηση. Στο θεωρητικό επίπεδο, η UML δεν αποτελεί απλώς ένα σχεδιαστικό εργαλείο, αλλά μια μεθοδολογική προσέγγιση αναπαράστασης δομής, συμπεριφοράς και αλληλεπιδράσεων συστημάτων.

Δύο βασικά χαρακτηριστικά της UML που είναι η κοινή γλώσσα μοντελοποίησης και η τυποποίηση μειώνουν τις ασάφειες στη θεωρητική περιγραφή συστημάτων και έτσι επιτρέπεται η ακριβής αναπαράσταση δομών και σχέσεων και διευκολύνεται η επιστημονική και ακαδημαϊκή τεκμηρίωση. Σε θεωρητικό επίπεδο, αυτό σημαίνει ότι τα μοντέλα μπορούν να αναλυθούν, να συγκριθούν και να επαναχρησιμοποιηθούν.

Επειδή η θεωρητική σχεδίαση απαιτεί υψηλό επίπεδο αφαίρεσης, η UML με την αφαιρετικότητα της και την εννοιολογική της σαφήνεια, μέσω των διαγραμμάτων της απομονώνει τις βασικές έννοιες (κλάσεις, συσχετίσεις, καταστάσεις, διεργασίες), αποτυπώνει αρχιτεκτονικές επιλογές πριν την υλοποίηση και επιτρέπει τον έλεγχο συνέπειας και πληρότητας του μοντέλου. Ειδικότερα η χρήση διαγραμμάτων κλάσεων, δραστηριοτήτων και καταστάσεων υποστηρίζει την αναλυτική θεώρηση του συστήματος ως σύνολο αλληλοεξαρτώμενων οντοτήτων.

Επίσης η UML υποστηρίζει τη θεωρητική αρχιτεκτονική σχεδίαση μέσω των διαγραμμάτων συστατικών στοιχείων και των διαγραμμάτων ανάπτυξης, τα οποία επιτρέπουν την αφηρημένη αναπαράσταση της κατανομής ευθυνών και των εξαρτήσεων.

Επιπρόσθετα η UML είναι χρήσιμη στη θεωρητική αξιολόγηση και την επαλήθευση, διότι επιτρέπει την ανάλυση της συνέπειας μεταξύ των διαγραμμάτων, τον έλεγχο πληρότητας των απαιτήσεων και τον εντοπισμό λογικών αντιφάσεων. Στη θεωρητική σχεδίαση, αυτό λειτουργεί ως μορφή πρόωρης επαλήθευσης πριν την κωδικοποίηση, μειώνοντας το γνωστικό και υπολογιστικό κόστος σφαλμάτων.

Επειδή η UML χρησιμοποιείται ευρέως στην έρευνα λογισμικού ως εργαλείο μοντελοποίησης συστημάτων, ως μέσο ανάλυσης απαιτήσεων και ως βάση για μηχανική που βασίζεται σε μοντέλα (Model Driven Engineering - MDE), αυτό της χαρίζει επιστημονική και μεθοδολογική αξία. Στο πλαίσιο της αρχιτεκτονικής που βασίζεται σε μοντέλα (Model Driven Architecture - MDA), τα μοντέλα της UML αποκτούν κεντρικό ρόλο στη θεωρητική περιγραφή του συστήματος.

8.4 Περιορισμοί στη χρήση των UML διαγραμμάτων

Ο Sommerville (2016) υποστηρίζει ότι η UML είναι λιγότερο χρήσιμη για την ανάπτυξη συστημάτων με τη χρήση μιας δυναμικά τυποποιημένης γλώσσας προγραμματισμού, όπως η Python και ότι δεν έχει νόημα η χρήση της αν το σύστημα υλοποιείται διαμορφώνοντας ένα έτοιμο πακέτο.

Επίσης σχολιάζει ότι η UML δεν περιλαμβάνει κάποιο τύπο διαγράμματος για μοντελοποίηση βάσεων δεδομένων, καθώς μοντελοποιεί δεδομένα χρησιμοποιώντας αντικείμενα και τις σχέσεις τους. Επιπλέον αναφέρει ότι σε μεγάλα συστήματα υπάρχει πρόβλημα με τη διαγράμματα καταστάσεων επειδή ο αριθμός των πιθανών καταστάσεων αυξάνεται ραγδαία. Συνεπώς, σε μοντέλα μεγάλων συστημάτων, πρέπει να αποκρυφτούν οι λεπτομέρειες.

Ακόμα θεωρεί ότι η UML δεν είναι τόσο χρήσιμη κατά τη διάρκεια της διαδικασίας αρχιτεκτονικού σχεδιασμού και προτιμά άτυπες σημειώσεις που είναι πιο γρήγορες στη σύνταξη και που μπορούν εύκολα να σχεδιαστούν σε έναν πίνακα.

Ένας άλλος περιορισμός της UML αφορά τα έξοδα συντήρησης. Η διαδικασία διατήρησης ενημερωμένων διαγραμμάτων απαιτεί συνεχή προσπάθεια. Αυτό έχει ως αποτέλεσμα τα διαγράμματα συχνά να παραμελούνται σε ευέλικτα ή γρήγορου ρυθμού περιβάλλοντα, οδηγώντας έτσι σε ανακριβή ή ξεπερασμένη τεκμηρίωση (Ambler, 2002).

Επίσης, η UML έχει πρακτικά περιορισμένη εκτελεσιμότητα και αυτό περιορίζει τα οφέλη της. Χωρίς εξειδικευμένα εργαλεία ή επεκτάσεις, τα διαγράμματα UML παραμένουν σε μεγάλο βαθμό περιγραφικά και δεν μπορούν να επικυρώσουν άμεσα τη συμπεριφορά του συστήματος, περιορίζοντας τον πρακτικό τους αντίκτυπο σε επαναληπτικούς κύκλους ανάπτυξης (OMG, 2017).

Δεν υπάρχει επίσημος τρόπος που να αντιστοιχίζει την UML σε οποιαδήποτε συγκεκριμένη γλώσσα προγραμματισμού. Δεν υπάρχει η δυνατότητα από ένα συγκεκριμένο διάγραμμα UML να εξαχθεί ο ισοδύναμος κώδικας, παρά μόνο μπορεί να σχηματιστεί μια γενική ιδέα για το πώς θα μοιάζει ο κώδικας (Fowler, 2004).

Η UML παρουσιάζει επίσης περιορισμένη δυνατότητα στην έκφραση μη λειτουργικών απαιτήσεων. Πτυχές όπως η απόδοση, η ασφάλεια, η αξιοπιστία και η επεκτασιμότητα είναι δύσκολο να αποτυπωθούν αποτελεσματικά μέσα από τα διαγράμματά της. Η UML επικεντρώνεται κυρίως στη δομική και συμπεριφορική μοντελοποίηση, αφήνοντας κενά στις προδιαγραφές του συστήματος.

Χρησιμοποιώντας την UML οι χρήστες πρέπει να κατανοήσουν πολλαπλούς τύπους διαγραμμάτων, γεγονός που συμβάλλει σε μια απότομη καμπύλη εκμάθησης. Η κακή χρήση ή η υπερβολική χρήση διαγραμμάτων μπορεί να μειώσει την αποτελεσματικότητά τους και να οδηγήσει σε περιττή πολυπλοκότητα (Fowler, 2004).

Τέλος, κάποια διαφορετικά εργαλεία μοντελοποίησης ενδέχεται να εφαρμόζουν την UML με ασυνέπεια ή να εισάγουν ιδιόκτητες επεκτάσεις, κάτι που μπορεί να εμποδίσει τη συνεργασία και την ανταλλαγή μοντέλων. Αυτό αποδυναμώνει έναν από τους βασικούς στόχους της UML, που είναι η τυποποίηση σε πλατφόρμες και οργανισμούς (OMG, 2017).

8.5 Σύγκριση με άλλες προσεγγίσεις μοντελοποίησης

Κάθε μοντέλο έχει τα δυνατά και τα αδύνατα σημεία του. Επομένως, κανένα μεμονωμένο μοντέλο δεν επαρκεί για όλες τις ανάγκες ανάπτυξης λογισμικού. Αν και η UML είναι στην πραγματικότητα αρκετά ισχυρή, η πραγματικότητα είναι ότι δεν επαρκεί για τις ανάγκες μοντελοποίησης.

8.5.1 Δομημένη μοντελοποίηση

Τη δεκαετία του 1970, οι μέθοδοι δομημένου σχεδιασμού χρησιμοποιούσαν διαγράμματα ροής δεδομένων (data flow diagrams - DFD) ως τρόπο απεικόνισης των βημάτων επεξεργασίας σε ένα σύστημα. Τα διαγράμματα ροής δεδομένων εστιάζουν στη ροή των δεδομένων και των διαδικασιών, και τα διαγράμματα οντοτήτων-συσχετίσεων (entity relationship diagrams – ERD) μοντελοποιούν μόνο τις σχέσεις των δεδομένων.

Τα διαγράμματα ροής δεδομένων και οντοτήτων-συσχετίσεων είναι χρήσιμα επειδή βοηθούν τους αναλυτές και τους σχεδιαστές να κατανοήσουν τι συμβαίνει στη διαδικασία. Τα διαγράμματα ροής δεδομένων είναι απλά και διαισθητικά και επομένως είναι πιο προσιτά στους ενδιαφερόμενους από ορισμένους άλλους τύπους μοντέλων.

Ένας εναλλακτικός τρόπος εμφάνισης της ακολουθίας διαδικασιών σε ένα σύστημα είναι η χρήση διαγραμμάτων ακολουθίας της UML. Γενικά, τα διαγράμματα ακολουθίας επισημαίνουν αντικείμενα σε ένα σύστημα, ενώ τα διαγράμματα ροής δεδομένων επισημαίνουν τις λειτουργίες ή τις δραστηριότητες. Στην πράξη, οι μη ειδικοί φαίνεται να βρίσκουν τα διαγράμματα ροής δεδομένων πιο διαισθητικά, αλλά οι μηχανικοί προτιμούν τα διαγράμματα ακολουθίας (Sommerville, 2016).

8.5.2 Ευέλικτες μέθοδοι

Οι ευέλικτες μέθοδοι πρωτοεμφανίστηκαν ανάμεσα στα τέλη του '90 και στις αρχές του 2000. Πρόκειται για μεθόδους που είναι επαναληπτικές και επαυξητικές και στοχεύουν στην ταχεία ανάπτυξη λογισμικού (Λαζαρίνης & Μαυρομμάτης, 2015). Περιλαμβάνουν κυρίως τον Ακραίο Προγραμματισμό (eXtreme Programming - XP) και τη μεθοδολογία Scrum.

Οι ευέλικτες μέθοδοι δίνουν έμφαση στην επικοινωνία εις βάρος της τεκμηρίωσης και ενθαρρύνουν την ταχεία, συνεργατική, άτυπη μοντελοποίηση, ενώ η UML χρησιμοποιείται για λεπτομερή τεκμηρίωση και είναι πιο τυπική. Ωστόσο η χρήση των ευέλικτων μεθόδων δεν αποκλείει τη χρήση UML διαγραμμάτων, καθώς είναι συμβατά μεταξύ τους.

Ο Ambler (2002) υποστηρίζει ότι οι ευέλικτες μέθοδοι δεν αντικαθιστούν τη UML. Αντιθέτως, οι ευέλικτες μέθοδοι είναι ο οδηγός για το πώς και πότε πρέπει να χρησιμοποιήσουμε τα σύμβολα της UML. Επίσης οι μέθοδοι αυτές αποτρέπουν τις ομάδες εργασίας να γίνουν γραφειοκρατικές και δυσκίνητες στη λειτουργία τους.

Τα διαγράμματα της UML γίνονται πιο αποτελεσματικά όταν χρησιμοποιούνται σε συνδυασμό με τις αρχές των ευέλικτων μεθόδων, όπως είναι η απλότητα, η συνεργασία και η επανάληψη.

Ανακεφαλαιώνοντας, η ευέλικτη μοντελοποίηση είναι μια ελαφριά και ανθρωποκεντρική προσέγγιση μοντελοποίησης σε σχέση με τη UML, που είναι μια επίσημη σημειογραφία μοντελοποίησης, και παρέχει την πειθαρχία για την αποτελεσματική χρήση της UML χωρίς να την αφήνει να γίνει βαριά ή σπάταλη.

8.5.3 BPMN

Το μοντέλο και σημειογραφία επιχειρηματικής διαδικασίας (Business Process Model and Notation – BPMN) έχει αρκετές ομοιότητες με τα διαγράμματα δραστηριοτήτων της UML, ωστόσο έχει ευρύτερη χρήση στον τομέα διαχείρισης διαδικασιών (Λαζαρίνης & Μαυρομμάτης, 2015).

Από ένα πείραμα που διεξήγαγαν οι Peixoto et al. (2008) με φοιτητές πληροφορικής, που δεν είναι εξοικειωμένοι με τις γλώσσες και με τον μοντελοποιημένο τομέα, έδειξε ότι τόσο η BPMN όσο και τα διαγράμματα δραστηριοτήτων της UML (UML AD) είναι εξίσου εύκολα κατανοητές για επιχειρηματικές διαδικασίες από τα ενδιαφερόμενα μέρη.

Σε πολλές περιπτώσεις, το BPMN και τα διαγράμματα δραστηριοτήτων της UML χρησιμοποιούν παρόμοια σύμβολα για να περιγράψουν επιχειρηματικές διαδικασίες. Ωστόσο, υπάρχουν πτυχές των επιχειρηματικών διαδικασιών που μπορούν να μοντελοποιηθούν στο BPMN χρησιμοποιώντας μόνο ένα σύμβολο, αλλά για τις οποίες η αναπαράσταση στο διάγραμμα δραστηριοτήτων απαιτεί τη χρήση μιας ομάδας συμβόλων.

Αξιολόγηση του BPMN και των UML AD έδειξε ότι και οι δύο γλώσσες μοντελοποίησης επιχειρηματικών διαδικασιών παρέχουν παρόμοιες λύσεις για τα περισσότερα μοτίβα. Επίσης, έρευνες που διεξήχθησαν από μια σειρά συγγραφέων σχετικά με την ικανότητα των προηγούμενων εκδόσεων του BPMN και των UML AD να καταγράψουν τα μοτίβα ροής εργασίας έδειξαν ότι και οι δύο συμβολισμοί προσφέρουν ολοκληρωμένη υποστήριξη για τις προοπτικές ροής ελέγχου και δεδομένων, αλλά προσφέρουν περιορισμένο αριθμό λύσεων για μοτίβα πόρων ροής εργασίας και μοτίβα χειρισμού εξαιρέσεων.

Όσον αφορά την πολυπλοκότητα των γραφικών συμβόλων που χρησιμοποιούνται για την αναπαράσταση των πραγματικών επιχειρηματικών διαδικασιών ενός οργανισμού τα αποτελέσματα ανάλυσης έδειξαν ότι, στις περισσότερες περιπτώσεις, το BPMN και τα

UML AD χρησιμοποιούν παρόμοια σύμβολα για να περιγράψουν επιχειρηματικές διαδικασίες. Υπάρχουν όμως και περιπτώσεις όπου τα στοιχεία των επιχειρηματικών διαδικασιών μοντελοποιούνται χρησιμοποιώντας μόνο ένα σύμβολο στο BPMN και χρησιμοποιώντας μια ομάδα συμβόλων στα UML AD (Geambasu, 2012).

8.5.4 SysML και DSL

Η SysML (Systems Modeling Language) επαναχρησιμοποιεί ένα υποσύνολο της UML 2.0 και παρέχει πρόσθετες επεκτάσεις που απαιτούνται για την αντιμετώπιση των απαιτήσεων στη UML για αίτηση υποβολής προσφοράς στη Μηχανική Συστημάτων (Roques, 2011).

Η SysML παρέχει έναν νέο τύπο διαγράμματος, το διάγραμμα απαιτήσεων (requirement diagram), το οποίο τοποθετείται ανάμεσα στα συμπεριφορικά διαγράμματα και τα διαγράμματα δομής.

Όσον αφορά τα συμπεριφορικά διαγράμματα, περιλαμβάνει τα διαγράμματα ακολουθίας, μηχανών καταστάσεων και περιπτώσεων χρήσης και είναι ίδια με της UML. Επίσης χρησιμοποιεί και τα διαγράμματα δραστηριοτήτων, αλλά είναι τροποποιημένα σε σχέση με τα αντίστοιχα της UML.

Όσον αφορά τα διαγράμματα δομής περιλαμβάνει τα διαγράμματα πακέτων, που είναι ίδια με της UML. Επίσης χρησιμοποιεί τα διαγράμματα ορισμού μπλοκ, που είναι ουσιαστικά τροποποιημένα διαγράμματα κλάσεων της UML και τα διαγράμματα εσωτερικών μπλοκ. Τα τελευταία περιλαμβάνουν και τα παραμετρικά διαγράμματα.

Γενικά η SysML αποτελεί μια επέκταση της UML, κατάλληλη για μηχανική συστημάτων, ενώ η UML περιορίζεται κυρίως σε συστήματα λογισμικού ([SysML® v1 Specification | Object Management Group](#)).

Ακόμα υπάρχουν οι DSL (Domain-Specific Languages). Είναι γλώσσες που εξειδικεύονται σε ένα συγκεκριμένο πεδίο εφαρμογών σε αντίθεση με τις γλώσσες γενικής χρήσης, όπως είναι η UML. Ένα πλεονέκτημα της προσέγγισης DSL είναι η αυξημένη εκφραστικότητα που παρέχει εντός ενός συγκεκριμένου πεδίου σε σχέση με τη UML που είναι πιο ευέλικτη αλλά μπορεί να στερείται ακρίβειας για το πεδίο αυτό.

9. Συζήτηση – Συμπεράσματα

9.1 Συνοπτική σύνθεση ευρημάτων

Η σωστή ανάλυση απαιτήσεων αποτελεί θεμελιώδη παράγοντα για την επιτυχή ανάπτυξη λογισμικού, με τη διακριτή κατηγοριοποίηση σε λειτουργικές και μη λειτουργικές απαιτήσεις να συμβάλλει στη σαφήνεια και πληρότητα του έργου. Η μοντελοποίηση γεφυρώνει το χάσμα μεταξύ απαιτήσεων και υλοποίησης, μειώνοντας την πολυπλοκότητα και υποστηρίζοντας την ποιότητα του τελικού προϊόντος.

Η μετάβαση από αφηρημένες έννοιες σε υλοποιήσιμα μοντέλα γίνεται μέσω μιας δομημένης και διαδοχικής διαδικασίας, όπου ο εννοιολογικός σχεδιασμός διασφαλίζει τη σαφήνεια και αποφεύγει μελλοντικά προβλήματα. Η επιτυχία εξαρτάται όχι μόνο από την τεχνική γνώση αλλά και από την αποτελεσματική επικοινωνία, καθιστώντας τη διαχείριση προβλημάτων επικοινωνίας κρίσιμη για την ποιότητα του αποτελέσματος.

Η UML αποτελεί το κύριο πρότυπο γλώσσας μοντελοποίησης αντικειμενοστρεφούς σχεδιασμού με σημαντική επίδραση στη βιομηχανία και την εκπαίδευση. Παρά τη σημαντική τυποποίηση και αποδοχή της, παραμένει πολύπλοκη και όχι πάντα φιλική προς τον χρήστη, με ανάγκη για συνεχή βελτίωση και έρευνα ώστε να αξιοποιηθεί ευρύτερα εκτός μηχανικής λογισμικού.

Η UML προσφέρει ανεξαρτησία από γλώσσες προγραμματισμού και δυνατότητα επέκτασης, καθιστώντας την ευέλικτη και ευρέως εφαρμόσιμη. Βελτιώνει την επικοινωνία των ομάδων εργασίας, παρέχει αρχιτεκτονική αρτιότητα και διαχείριση της πολυπλοκότητας. Ακόμα προσφέρει σαφήνεια και απλότητα στα πρώτα στάδια σχεδιασμού, ιδανική για περιβάλλοντα με απαιτητική τεκμηρίωση και διεπιστημονική συνεργασία, όπως εικονικά εργαστήρια και μεγάλα εταιρικά συστήματα.

Ωστόσο, η μετατόπιση προς ευέλικτες μεθοδολογίες και καταναεμημένες αρχιτεκτονικές καθιστά την UML λιγότερο κατάλληλη για πολλές σύγχρονες πρακτικές ανάπτυξης, με άλλες απλούστερες και πιο συνεργατικές εναλλακτικές να κερδίζουν έδαφος.

Προσφέρει διαγράμματα που καλύπτουν τόσο τη στατική δομή όσο και τη δυναμική συμπεριφορά, επιτρέποντας την αποδοτική και σαφή αναπαράσταση πολύπλοκων συστημάτων.

Τα διαγράμματα περιπτώσεων χρήσης εστιάζουν στο τι κάνει το σύστημα, βοηθούν στην επαλήθευση των απαιτήσεων, στην αποφυγή λαθών και στη σωστή διαμόρφωση των διεπαφών.

Τα διαγράμματα κλάσεων επιτρέπουν την αποτύπωση της στατικής δομής του συστήματος με σαφήνεια και ακρίβεια, εστιάζοντας στις κλάσεις και τις σχέσεις μεταξύ τους, χωρίς να καλύπτουν τη συμπεριφορά ή τις λειτουργίες.

Τα διαγράμματα ακολουθίας καταγράφουν τη δυναμική συμπεριφορά των αντικειμένων με χρονική σειρά, υποστηρίζοντας τη σχεδίαση υλοποίησης.

Τα διαγράμματα δραστηριοτήτων συμπληρώνουν τα διαγράμματα περιπτώσεων χρήσης, καθώς παρέχουν την απαιτούμενη λεπτομέρεια στη ροή και τη σειρά των ενεργειών.

Τα διαγράμματα μηχανών καταστάσεων εξειδικεύουν τη δυναμική συμπεριφορά μέσω καταστάσεων και μεταβάσεων, ενώ τα διαγράμματα συμπεριφορικών μηχανών καταστάσεων είναι το βασικό εργαλείο για πολύπλοκες κλάσεις.

Τα διαγράμματα συστατικών στοιχείων είναι απαραίτητα για αρχιτεκτονική λογισμικού, προσφέρουν φυσική, επαναχρησιμοποιήσιμη και διαχειρίσιμη δομή με λειτουργικότητα και διεπαφές.

Η μετατροπή απαιτήσεων σε σχεδιασμό ακολουθεί δομημένη μεθοδολογία, διασφαλίζοντας ότι το τελικό προϊόν ανταποκρίνεται στις ανάγκες χρηστών και στόχους έργου.

Η μετατροπή των λειτουργικών απαιτήσεων σε μοντέλα αλληλεπίδρασης αποτελεί κρίσιμο βήμα, με συνδυαστική χρήση διαγραμμάτων κλάσεων και ακολουθίας για ολοκληρωμένη απεικόνιση δομής και συμπεριφοράς.

Η συνδυαστική χρήση διαγραμμάτων καταστάσεων και δραστηριοτήτων επιτρέπει ολιστική και πολύπλευρη μοντελοποίηση πολύπλοκων διαδικασιών και μακροχρόνιων αντικειμένων.

Η κωδικοποίηση απαιτήσεων και η διασφάλιση ιχνηλασιμότητας σε συνδυασμό με UML συνιστούν οργανωμένη μεθοδολογία ανάπτυξης αξιόπιστου λογισμικού.

Παρουσιάζει σημαντικές θεωρητικές αδυναμίες όσον αφορά τη φιλοσοφική θεμελίωση, τη σημασιολογική πληρότητα και τη μεθοδολογική συνοχή, αλλά παραμένει εργαλείο κρίσιμης σημασίας, ενώ η ανάγκη για βελτίωση και προσαρμογή στις σύγχρονες μεθοδολογίες είναι εμφανής.

9.2 Θεωρητική συμβολή της UML στη σχεδίαση συστημάτων

Η UML δεν αποτελεί απλώς ένα τεχνικό εργαλείο σχεδίασης λογισμικού, αλλά λειτουργεί και ως σημαντικό εργαλείο θεωρητικής σκέψης, οργάνωσης και ανάλυσης σύνθετων εννοιών και σχέσεων.

Τα διαγράμματα της UML βοηθούν στην ανάπτυξη βαθύτερης κατανόησης και αναλυτικής σκέψης, υποστηρίζοντας τη διαμόρφωση θεωρητικών μοντέλων στη μηχανική λογισμικού και σε συναφείς τομείς.

Υποστηρίζει τη σαφή αποτύπωση απαιτήσεων μέσω διαγραμμάτων περιπτώσεων χρήσης. Ενισχύει τη μετάβαση από λεκτική περιγραφή σε δομημένη αναπαράσταση, μειώνοντας τις παρερμηνείες, εστιάζοντας σε βασικές έννοιες πριν από τεχνικές λεπτομέρειες.

Τα διαγράμματα της μετατρέπουν τις θεωρητικές ιδέες σε γραφικά μοντέλα, απλοποιώντας την κατανόηση σχέσεων όπως κληρονομικότητα, συσχέτιση και αλληλεπίδραση οντοτήτων.

Η κοινή γλώσσα συμβόλων και σχέσεων διευκολύνει την ακριβή έκφραση θεωρητικών ιδεών και ενισχύει την ανάλυση και επαλήθευση υποθέσεων.

Προσφέρει ένα κοινό πλαίσιο επικοινωνίας μεταξύ ερευνητών, προγραμματιστών και άλλων εμπλεκόμενων, μειώνοντας ασάφειες και βελτιώνοντας τη συλλογική ανάπτυξη θεωρητικών μοντέλων.

Μέσω των διαγραμμάτων στοιχείων, δραστηριοτήτων και καταστάσεων, επιτρέπει την αποσύνθεση σύνθετων συστημάτων σε διαχειρίσιμα μέρη, γεφυρώνοντας θεωρητικά μοντέλα και πρακτικό σχεδιασμό.

Αποτελεί βασικό εργαλείο για την οπτική μοντελοποίηση και κατανόηση σύνθετων συστημάτων, ειδικά εκείνων με πολλαπλές αλληλεπιδράσεις και δομές.

Μέσω της αφαιρετικής αναπαράστασης, επιτρέπει τον σταδιακό και εστιασμένο χειρισμό συστημάτων.

Προωθεί τη συστημική σκέψη, αναδεικνύοντας όχι μόνο τα μέρη αλλά και τις σχέσεις και τις ροές πληροφορίας.

Μετατρέπει την πολυπλοκότητα σε διαχειρίσιμη γνώση, βελτιώνοντας το σχεδιασμό πολύπλοκων συστημάτων.

Επιτρέπει την κατανόηση από μη τεχνικά άτομα, όπως στελέχη διοίκησης και πελάτες και βοηθά στην αποφυγή «εξάτμισης της γνώσης» μέσω τεκμηρίωσης, διευκολύνοντας τη μετάδοση γνώσης σε νέους προγραμματιστές.

Στη φάση αρχιτεκτονικής, υποστηρίζει τη μοντελοποίηση στατικών και δυναμικών στοιχείων (διαγράμματα κλάσεων, συστατικών, ανάπτυξης).

Υποστηρίζει την ιχνηλασιμότητα μεταξύ απαιτήσεων, ανάλυσης και αρχιτεκτονικής, εξασφαλίζοντας συνέπεια μεταξύ επιχειρησιακών αναγκών και τεχνικών αποφάσεων.

9.3 Σημεία προς περαιτέρω διερεύνηση

Όπως προαναφέρθηκε στην παρούσα εργασία η UML εφαρμόζεται στην ανάπτυξη εμπορικών συστημάτων λογισμικού σε διάφορους κλάδους, όπως οι επιχειρηματικές διαδικασίες, ο τομέας της υγείας και η σχεδίαση hardware.

Θα μπορούσε να γίνει μια περαιτέρω διερεύνηση στην παρουσίαση της σχεδίασης ενός ολοκληρωμένου συστήματος από τη φάση της σχεδίασης μέχρι τη φάση της υλοποίησης. Πιο συγκεκριμένα θα μπορούσε να παρουσιαστεί η χρήση της UML κατά τη διαδικασία

σχεδίασης ενός εμπορικού συστήματος λογισμικού που χρησιμοποιείται π.χ. σε ιατρικό μηχάνημα.

Ωστόσο μια τέτοια έρευνα θα ήταν πολύ περιορισμένη λόγω αδυναμίας πρόσβασης στη διαδικασία σχεδίασης μιας τέτοιας εμπορικής εφαρμογής.

Επίσης θα μπορούσε να συμπληρωθεί η έρευνα στον τρόπο που συνεργάζεται η UML με άλλες μεθόδους μοντελοποίησης κατά τη διαδικασία σχεδίασης ενός τέτοιου συστήματος.

Η ραγδαία εξάπλωση της τεχνητής νοημοσύνης σε διάφορους τομείς, δεν έχει αφήσει ανεπηρέαστη και την ανάπτυξη συστημάτων λογισμικού. Συνεπώς θα μπορούσε μελλοντικά να γίνει μια έρευνα για το πώς συνεργάζεται η UML με τη μηχανική μάθηση, καθώς έχει αποδειχτεί ότι ο συνδυασμός τους μπορεί να απλοποιήσει σημαντικά τη διαδικασία ανάπτυξης ενός συστήματος. Η μηχανική μάθηση μπορεί να αυτοματοποιήσει ορισμένες πτυχές της διαδικασίας ανάπτυξης, με αποτέλεσμα ακόμη μεγαλύτερη εξοικονόμηση χρόνου (Gadhi, Gondu, Bandaru, Reddy & Abiona, 2023).

Η UML διευκολύνει την καλύτερη κατανόηση του συστήματος, ενώ η μηχανική μάθηση εξασφαλίζει μεγαλύτερη ακρίβεια στην ανάλυση των παραμέτρων του συστήματος. Αυτή η συγχώνευση της UML και της μηχανικής μάθησης έχει καταδείξει την ικανότητά της να μειώνει τα σφάλματα, να βελτιώνει την απόδοση, ακόμη και να αυτοματοποιεί τη διαδικασία ανάπτυξης (Kant & Das, 2017).

Βιβλιογραφία

Ακολουθούν οι βιβλιογραφικές αναφορές (πηγές) της Εργασίας.

Λαζαρίνης, Φ & Μαυρομμάτης, Γ. (2015). *Τεχνολογία Λογισμικού*. Πάτρα: Ελληνικό Ανοικτό Πανεπιστήμιο

Ταμπούρης, Ε. & Χατζηγεωργίου, Α. (2015). *Σχεδίαση Λογισμικού*. Πάτρα: Ελληνικό Ανοικτό Πανεπιστήμιο

Ahmad, T., Iqbal, J., Ashraf, A., Truscan, D., & Porres, I. (2019). Model-based testing using UML activity diagrams: A systematic mapping study. *Computer Science Review*, 33(July), 98–112. <https://doi.org/10.1016/j.cosrev.2019.07.001>

Al-Fedaghi, S. (n.d.). *Conceptual Modeling with Constraints*.

AlRababah, A. A. (2024). Assessing the Effectiveness of UML Models in Software System Development. *International Journal of Applied Science and Research*, 07(01), 13–24. <https://doi.org/10.56293/ijasr.2024.5703>

Barral, L. V., Pinet, F., Tacnet, J. M., & Joussetme, A. L. (2020). Combining UML profiles to design serious games dedicated to trace information in decision processes. *International Journal of Information System Modeling and Design*, 11(2), 1–27. <https://doi.org/10.4018/IJISMD.2020040101>

Bell, D. (2004). Uml basics: The component diagram. *IBM Global Services*, 1-10.

Booch, G. (1990). *Object oriented design with applications*. Benjamin-Cummings Publishing Co., Inc..

Booch, G., Rumbaugh, J., & Jacobson, I. (2005). *The Unified Modeling Language user guide* (2nd ed.). Addison-Wesley.

Cook, S. (2012). Looking back at UML. *Software & Systems Modeling*, 11(4), 471–480. <https://doi.org/10.1007/s10270-012-0256-x>

Davies, I., Green, P., Rosemann, M., Indulska, M., & Gallo, S. (2006). How do practitioners use conceptual modeling in practice? *Data & Knowledge Engineering*, 58(3), 358–380. <https://doi.org/10.1016/j.datak.2005.07.007>

- Daw, Z., & Cleaveland, R. (2016). *An extensible formal semantics for UML activity diagrams*. <http://arxiv.org/abs/1604.02386>
- Dobing, B., & Parsons, J. (2006). How UML is used. *Communications of the ACM*, 49(5), 109–113. <https://doi.org/10.1145/1125944.1125949>
- Dori, D. (2002). Why significant UML change is unlikely. *Communications of the ACM*, 45(11), 82–85. <https://doi.org/10.1145/581571.581599>
- Erickson, J., & Siau, K. (2013). Unified Modeling Language: The Teen Years and Growing Pains. In S. Yamamoto (Ed.), *Human Interface and the Management of Information. Information and Interaction Design* (pp. 295–304). Springer Berlin Heidelberg.
- Evans, A., Kent, S., & Selic, B. (1998). The UML as a formal modeling notation. *UML '98: Beyond the Notation*. Springer.
- Fernández-Sáez, A. M., Chaudron, M. R., & Genero, M. (2018). An industrial case study on the use of UML in software maintenance and its perceived benefits and hurdles. *Empirical Software Engineering*, 23(6), 3281-3345.
- Fowler, M. (2004). *UML Distilled* (3rd ed.). Addison-Wesley.
- France, R., & Rumpe, B. (2007). Model-driven development of complex software: A research roadmap. *Future of Software Engineering (FOSE 2007)*, IEEE.
- Gadhi, A., Gondu, R. M., Bandaru, C. M., Reddy, K. C., & Abiona, O. (2023). Applying UML and Machine Learning to Enhance System Analysis and Design. *International Journal of Communications, Network and System Sciences*, 16(05), 67–76. <https://doi.org/10.4236/ijcns.2023.165005>
- Geambașu, C. V. (2012). BPMN VS. UML ACTIVITY DIAGRAM FOR BUSINESS PROCESS MODELING Accounting and Management Information Systems. In *Accounting and Management Information Systems* (Vol. 11, Issue 4).
- Guizzardi, G., Herre, H., & Wagner, G. (2005). On the general ontological foundations of conceptual modeling. *ER 2005*. Springer.

- Harel, D., & Rumpe, B. (2004). Meaningful modeling: What's the semantics of "semantics"? *IEEE Computer*, 37(10), 64–72.
- Helms, T., Warnke, T., Maus, C., & Uhrmacher, A. M. (2017). Semantics and Efficient Simulation Algorithms of an Expressive Multilevel Modeling Language. *ACM Trans. Model. Comput. Simul.*, 27(2), 8:1--8:25. <https://doi.org/10.1145/2998499>
- Hettab, A., Kerkouche, E., & Chaoui, A. (2015). A graph transformation approach for automatic test cases generation from UML Activity Diagrams. *ACM International Conference Proceeding* <https://doi.org/10.1145/2790798.2790801> Series, 13-17-July, 88–97. <https://doi.org/10.1145/2790798.2790801>
- John, B. (2025). An Analysis of the Significance of UML Diagrams Focus on Medium-Sized Projects.
- Krogstie, J. (2012). *Model-Based Development and Evolution of Information Systems: A quality approach*. <https://doi.org/10.1007/978-1-4471-2936-3>
- Lazarova-Molnar, Z. Zheng, T. Roeder, & R. Thiesing (Eds.), *Proceedings of the 2020 Winter Simulation Conference* (Vol. 1, Issue 1, pp. 2353–2364).
- Lindland, O.I., Sindre, G., Sølvsberg, A.: *Understanding quality in conceptual modeling*. *IEEE Softw.* 11 (2), 42–49 (1994)
- Maylawati, D. S. A., Ramdhani, M. A., & Amin, A. S. (2018). Tracing the linkage of several Unified Modelling Language diagrams in software modelling based on best practices. *International Journal of Engineering & Technology (UEA)*, 7(2.19), 776-780.
- Omg. (2009). *An OMG ® Unified Modeling Language ® Publication* *OMG ® Unified Modeling Language ® (OMG UML ®)* *OMG Document Number: Date*. <https://www.omg.org/spec/UML/20161101/PrimitiveTypes.xmi>
- Peixoto, D. C. C., Batista, V. A., Atayde, A. P., Borges, E. P., Resende, R. F., Isaías, C., & Pádua, P. S. (n.d.). *A Comparison of BPMN and UML 2.0 Activity Diagrams*.
- Peng, D., Warnke, T., Haack, F., & Uhrmacher, A. M. (2016). Reusing simulation experiment specifications to support developing models by successive extension. *Simulation Modelling Practice and Theory*, 68, 33–53. <https://doi.org/10.1016/j.simpat.2016.07.006>

- Platt, R. & Thompson, N. (2014). The Evolution of UML. In Mehdi Khosrow-Pour(Ed.), *Encyclopedia of Information Science and Technology, Third Edition* (pp. 1931-1936). USA: IGI Global.
- Rahmoune, Y., Chaoui, A., & Kerkouche, E. (2015). A framework for modeling and analysis UML activity diagram using graph transformation. *Procedia Computer Science*, 56(1), 612–617. <https://doi.org/10.1016/j.procs.2015.07.261>
- Rehman, A., Latif, S., & Zafar, N. A. (2019). Formal modeling of smart office using activity diagram and non deterministic finite automata. *2019 International Conference on Information Science and Communication Technology, ICISCT 2019*, 1–5. <https://doi.org/10.1109/CISCT.2019.8777444>
- Roques, P. (2011). *SysML par l'exemple-Un langage de modélisation pour systèmes complexes*. Editions Eyrolles.
- Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., & Lorensen, W. E. (1991). *Object-oriented modeling and design* (Vol. 199, No. 1). Englewood Cliffs, NJ: Prentice-hall.
- Rumpe, B. (2016). Modeling with UML: Language, Concepts, Methods. In *Modeling with UML: Language, Concepts, Methods*. Springer International Publishing. <https://doi.org/10.1007/978-3-319-33933-7>
- Saeed, S., Jhanjhi, N. Z., Naqvi, M., & Humayun, M. (2019). Analysis of software development methodologies. *International Journal of Computing and Digital Systems*, 8(5), 445–460. <https://doi.org/10.12785/ijcds/080502>
- Siau, K., Erickson, J., & Lee, L. (2005). Theoretical vs. Practical Complexity. *Journal of Database Management*, 16(3), 40–57. <https://doi.org/10.4018/jdm.2005070103>
- Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson.
- Sypsas, A. (2023). *Assessing the educational suitability of laboratory simulators with conceptual modeling and process similarity metrics* (Ph.D. Thesis). Hellenic Open University, Patras. <https://doi.org/10.12681/eadd/54494>
- Thomas, D. (2003). UML - Unified or universal modeling language? UML2, OCL, MOF, EDOC - The emperor has too many clothes. In *Journal of Object Technology* (Vol. 2, Issue 1). <https://doi.org/10.5381/jot.2003.2.1.c1>

Wiley, J., & Ambler, S. (2002). *Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process*

Wilsdorf, P., Dombrowsky, M., Uhrmacher, A. M., Zimmermann, J., & van Rienen, U. (2019). Simulation Experiment Schemas –Beyond Tools and Simulation Approaches. In N. Mustafee, K.-H. G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, & Y. J. Son (Eds.), *2019 Winter Simulation Conference (WSC)* (pp. 2783–2794). <https://doi.org/10.1109/WSC40007.2019.9004710>

Wilsdorf, P., Haack, F., & Uhrmacher, A. M. (2020). CONCEPTUAL MODELS IN SIMULATION STUDIES: MAKING IT EXPLICIT. In K.-H. Bae, B. Feng, S. Kim, S.

Ιστοσελίδες:

Cabot, J. (2026). *Debunking the top 10 myths and misconceptions about the Unified Modeling Language*. Ανακτήθηκε Φεβρουάριος 6, 2026, από <https://modeling-languages.com/debunking-top-10-uml-myths/>

Drobac-Ristovic, K. & Kavoliunas, D. (2025). *Understanding software design: Key principles and practices*. Ανακτήθηκε Δεκέμβριος 5, 2025, από https://www.hostinger.com/tutorials/software-design#Software_design_methodologies

McBride, J. (2025). *Is UML Obsolete in 2024? Exploring its Decline and Modern Alternatives*. Ανακτήθηκε Δεκέμβριος 4, 2025, από <https://jaystechbites.com/posts/2024/uml-alternatives-in-2024/>

Perera, N. (2025). *State Diagram vs. Activity Diagram Comparison*. Ανακτήθηκε Φεβρουάριος 6, 2026, από <https://creately.com/guides/difference-between-activity-diagram-and-state-diagram/>

Top 7 Communication Problems in Software Development and How to Solve Them. (2024, April 25). Ανακτήθηκε από <https://fullscale.io/blog/communication-problems-in-software-development/>

Visual Paradigm Skills. (2026, February 25). UML case Studies: Real-World modeling Examples. Ανακτήθηκε από <https://skills.visual-paradigm.com/docs/real-world-uml-case-studies-software-design/uml-case-studies/>

Watson, M. (2025). *Communication Problems in Software Development*. Ανακτήθηκε Δεκέμβριος 8, 2025, από <https://fullscale.io/blog/communication-problems-in-software-development/>

Υπεύθυνη Δήλωση Συγγραφέα:

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν.1599/1986, η παρούσα εργασία αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης.