



Μεταπτυχιακή Εξειδίκευση στα Πληροφοριακά Συστήματα

Διπλωματική Εργασία

Ο ρόλος της σημασιολογίας (semantics) στην ενοποιημένη γλώσσα μοντελοποίησης UML κατά το σχεδιασμό συστημάτων λογισμικού

Αλεξάνδρα Άννα Γασπαρινάτου

Επιβλέπων καθηγητής: Αθανάσιος Σύψας

Πάτρα, «Ιούνιος» «2026»

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.

Ο ρόλος της σημασιολογίας (semantics) στην ενοποιημένη γλώσσα
μοντελοποίησης UML κατά το σχεδιασμό συστημάτων λογισμικού

Αλεξάνδρα Άννα Γασπαρινάτου

Επιτροπή Επίβλεψης Διπλωματικής Εργασίας

Επιβλέπων Καθηγητής:

Αθανάσιος Σύψας

Συνεργαζόμενο Εκπαιδευτικό Προσωπικό
(ΣΕΠ) Ελληνικού Ανοικτού
Πανεπιστημίου

Συν-Επιβλέπων Καθηγητής:

Ιωάννης Μοσχολιός

Συνεργαζόμενο Εκπαιδευτικό Προσωπικό
(ΣΕΠ) Ελληνικού Ανοικτού
Πανεπιστημίου

Πάτρα, «Ιούνιος» «2026»

Ευχαριστίες

Ευχαριστώ θερμά τον επιβλέποντα καθηγητή μου κ. Αθανάσιο Σύψα τόσο για την ανάθεση του θέματος της εργασίας όσο και για την καθοδήγηση και τη συνεργασία καθόλη τη διάρκεια εκπόνησης της παρούσας εργασίας.

Θερμές ευχαριστίες και στον συν-επιβλέποντα καθηγητή μου κ. Ιωάννη Μοσχολιό για το ενδιαφέρον και την υποστήριξη αυτής της προσπάθειας.

Θα ήθελα επίσης, να εκφράσω την ευγνωμοσύνη μου στους Καθηγητές – Συμβούλους με τους οποίους συνεργάστηκα στο πλαίσιο των Θεματικών Ενοτήτων του Μεταπτυχιακού Προγράμματος.

*«Αλεξάνδρα – Άννα Γασπαρινάτου», «Ο ρόλος της σημασιολογίας
(semantics) στην ενοποιημένη γλώσσα μοντελοποίησης UML κατά
το σχεδιασμό συστημάτων λογισμικού»*

*«Αφιερώνεται με πολλή αγάπη στη μάνα μου
που πάντα με ωθούσε στη μάθηση»*

Περίληψη

Η παρούσα διπλωματική εργασία διερευνά τον ρόλο της σημασιολογίας (semantics) στην Ενοποιημένη Γλώσσα Μοντελοποίησης (Unified Modeling Language – UML) και τη συμβολή της στη σχεδίαση και ανάπτυξη συστημάτων λογισμικού. Παρότι η UML αποτελεί το πλέον διαδεδομένο πρότυπο μοντελοποίησης, η αποτελεσματικότητα των μοντέλων δεν εξαρτάται μόνο από τη σωστή χρήση της σύνταξης και των συμβολισμών της, αλλά και από την ακρίβεια με την οποία αποδίδεται το νόημα των εννοιών, των διαδικασιών και των αλληλεπιδράσεων που αυτά αναπαριστούν. Η σημασιολογική πληρότητα αποτελεί κρίσιμο παράγοντα για τη διασφάλιση της ποιότητας, της συνέπειας και της αξιοπιστίας των μοντέλων λογισμικού.

Στο θεωρητικό μέρος της εργασίας παρουσιάζονται η ιστορική εξέλιξη της UML, οι βασικές κατηγορίες διαγραμμάτων, οι έννοιες και τα επίπεδα της σημασιολογίας, καθώς και οι κυριότερες τυπικές προσεγγίσεις σημασιολογικής τεκμηρίωσης. Παράλληλα, αναλύεται η σημασιολογία των βασικών διαγραμμάτων UML, με ιδιαίτερη έμφαση στα Use Case Diagrams, Activity Diagrams, Class Diagrams και State Machine Diagrams, ενώ εξετάζονται συνηθισμένα σημασιολογικά σφάλματα και τρόποι αντιμετώπισής τους. Επιπλέον, διερευνάται η σχέση της σημασιολογίας με τη Model-Driven Engineering (MDE) και η σημασία της για τη μετατροπή μοντέλων σε εκτελέσιμο κώδικα.

Η θεωρητική ανάλυση υποστηρίζεται από δύο μελέτες περίπτωσης σε εκπαιδευτικά και εικονικά εργαστηριακά περιβάλλοντα: ένα σύστημα ελέγχου LED με Arduino Uno στο περιβάλλον Tinkercad και το πείραμα της Άωσης στο Klett Book-App και Publior Virtual Lab. Για κάθε περίπτωση αναπτύχθηκαν Use Case Models και Activity Diagrams και αξιολογήθηκε η σημασιολογική τους πληρότητα.

Τα αποτελέσματα της έρευνας δείχνουν ότι η σημασιολογική τεκμηρίωση βελτιώνει σημαντικά την κατανόηση, τη συνέπεια και την ποιότητα των UML μοντέλων, μειώνει τις πιθανότητες παρερμηνείας και ενισχύει τη δυνατότητα αξιοποίησής τους σε διαδικασίες αυτοματοποίησης και ανάπτυξης λογισμικού. Επιπλέον, αναδεικνύεται ότι η συνδυαστική χρήση διαφορετικών τύπων UML διαγραμμάτων συμβάλλει στην πληρέστερη αναπαράσταση σύνθετων συστημάτων και εκπαιδευτικών διαδικασιών, επιβεβαιώνοντας τον καθοριστικό ρόλο της σημασιολογίας στη σύγχρονη μοντελοποίηση λογισμικού.

Λέξεις – Κλειδιά

Ενοποιημένη Γλώσσα Μοντελοποίησης (UML)

Η σημασιολογία στη UML

UML Activity Diagrams

Model-Driven Engineering (MDE)

Εικονικά Εργαστήρια

The role of semantics in the Unified Modeling Language (UML) when designing software systems

Alexandra Anna Gasparinatu

Abstract

This thesis investigates the role of semantics in the Unified Modeling Language (UML) and its contribution to the design and development of software systems. Although UML is the most widely used modeling standard in software engineering, the effectiveness of UML models depends not only on the correct use of syntax and notation but also on the accurate representation of the meaning of concepts, processes, and interactions. Semantic completeness is therefore considered a critical factor in ensuring the quality, consistency, and reliability of software models.

The theoretical part of the study presents the historical evolution of UML, the main categories of UML diagrams, the fundamental concepts and levels of semantics, and the most important formal approaches to semantic specification. Furthermore, the semantics of the main UML diagram types are analyzed, with particular emphasis on Use Case Diagrams, Activity Diagrams, Class Diagrams, and State Machine Diagrams. Common semantic errors and inconsistencies are also examined, along with methods for addressing them. In addition, the study explores the relationship between semantics and Model-Driven Engineering (MDE), highlighting the importance of semantic consistency in model transformation and automated code generation.

The theoretical framework is supported by two case studies conducted in educational and virtual laboratory environments. The first case study focuses on a LED control system implemented with Arduino Uno in the Tinkercad environment, while the second examines the Buoyancy Experiment based on Archimedes' Principle in the Klett Book-App and Publior Virtual Lab environment. For each case study, UML Use Case Models and Activity Diagrams were developed and evaluated with respect to their semantic completeness.

The findings indicate that semantic documentation significantly improves the clarity, consistency, and overall quality of UML models, reducing the risk of misinterpretation and enhancing their suitability for software development and automation processes. Furthermore, the study demonstrates that the combined use of different UML diagram types contributes to a more comprehensive representation of complex systems and educational procedures. Overall, the results confirm the crucial role of semantics in modern software modeling and emphasize its importance for achieving reliable and meaningful model-driven development.

Keywords

Unified Modeling Language (UML)

Semantics in UML

UML Activity Diagrams

Model-Driven Engineering (MDE)

Virtual Labs

Περιεχόμενα

Περίληψη.....	vii
Abstract	viii
Περιεχόμενα	xii
Κατάλογος Εικόνων / Σχημάτων	xii
Κατάλογος Πινάκων	xiv
Συντομογραφίες & Ακρωνύμια.....	xv
1. Εισαγωγή.....	1
1.1 Αντικείμενο της εργασίας.....	1
1.2 Σκοπός και στόχοι της μελέτης	2
1.3 Ερευνητικά Ερωτήματα.....	3
1.4 Μεθοδολογία και Δομή της Εργασίας.....	4
1.5 Συμβολή της Εργασίας στη Διεθνή Βιβλιογραφία.....	5
2. Η UML ως γλώσσα μοντελοποίησης	7
2.1 Ιστορική εξέλιξη της UML	7
2.2 Ο Ρόλος της UML στον σχεδιασμό λογισμικού.....	10
2.3 Βασικές κατηγορίες διαγραμμάτων UML.....	12
2.4 Προκλήσεις στην ερμηνεία και κατανόηση UML μοντέλων.....	23
3. Σημασιολογία στη μοντελοποίηση λογισμικού.....	26
3.1 Ορισμός και βασικές έννοιες σημασιολογίας.....	26
3.2 Είδη σημασιολογίας.....	27
3.3 Η σημασιολογία στην UML.....	32
3.3.1 Πώς ορίζεται η σημασιολογία στην UML.....	32
3.3.2 Τυπικές προσεγγίσεις ορισμού σημασιολογίας (με βάση τη βιβλιογραφία).....	33
3.3.3 Semantics-Driven Modeling: αρχές και ρόλος στη διαδικασία ανάπτυξης.....	41
3.3.4 Καλές πρακτικές για σημασιολογικά σωστή χρήση της UML.....	43
4. Ανάλυση σημασιολογίας στα βασικά διαγράμματα UML.....	47
4.1 Σημασιολογία Διαγραμμάτων Περιπτώσεων Χρήσης (Use Case).....	47
4.2 Σημασιολογία Διαγραμμάτων Κλάσεων.....	58
4.3 Σημασιολογία Διαγραμμάτων Δραστηριοτήτων (Activity Diagrams).....	64
4.4 Σημασιολογία Διαγραμμάτων Κατάστασης (State Machine Diagrams).....	68
4.5 Συνηθισμένα λάθη και ελλείψεις στη σημασιολογική ερμηνεία.....	73
5. UML Activity Diagrams: Σημασιολογία και εφαρμογές.....	77
5.1 Ο ρόλος των Activity Diagrams στη μοντελοποίηση συστημάτων.....	77
5.2 Σημασιολογικές δομές στα Activity Diagrams.....	81
5.3 Χρήση των Activity Diagrams για μοντελοποίηση πειραματικών διαδικασιών σε εικονικά εργαστήρια.....	83
6. Από τη μοντελοποίηση στην υλοποίηση: Μετατροπή UML σε εκτελέσιμο κώδικα.....	90
6.1 Model - Driven Engineering (MDE) και σημασιολογία.....	90
6.2 Τεχνικές αυτόματης παραγωγής κώδικα.....	92
6.3 Ρόλος της σημασιολογικής συνέπειας στη μετατροπή μοντέλων.....	97

6.4 Μελέτες Περίπτωσης από τη βιβλιογραφία.....	99
6.5 Εργαλεία UML και δυνατότητες παραγωγής κώδικα.....	104
7. Μελέτη Περίπτωσης 1: Μοντελοποίηση πειραματικών διαδικασιών σε εικονικό εργαστήριο.....	108
7.1 Εικονικά Εργαστήρια.....	108
7.2 Περιγραφή σεναρίου: 1 ^η Μελέτη Περίπτωσης: Σύστημα Ελέγχου LED με Arduino Uno στο Tinkercad.....	114
7.3 Δημιουργία Use Case Models	118
7.4 Δημιουργία Activity Diagrams με σημασιολογική τεκμηρίωση.....	124
7.5 Αξιολόγηση των μοντέλων ως προς τη σημασιολογική πληρότητα.....	131
7.6 Συμπεράσματα της μελέτης περίπτωσης.....	137
8. Μελέτη Περίπτωσης 2: Μοντελοποίηση πειραματικών διαδικασιών σε εικονικό εργαστήριο.....	143
8.1 Περιγραφή του σεναρίου: 2 ^η Μελέτη Περίπτωσης: Πείραμα της Άνωσης στο Εικονικό Εργαστήριο Klett/Publior Virtual Lab.....	143
8.2 Δημιουργία Use Case Models.....	147
8.3 Δημιουργία Activity Diagrams με σημασιολογική τεκμηρίωση.....	157
8.4 Αξιολόγηση των μοντέλων ως προς τη σημασιολογική πληρότητα.....	167
8.5 Συμπεράσματα της μελέτης περίπτωσης.....	169
9. Συζήτηση - Αποτελέσματα.....	171
9.1 Επισκόπηση ευρημάτων.....	171
9.2 Συσχέτιση με τη βιβλιογραφία.....	172
9.3 Σημασία της σημασιολογίας για την ανάπτυξη λογισμικού.....	173
10. Συμπεράσματα και μελλοντικές κατευθύνσεις.....	176
10.1 Απάντηση στα ερευνητικά ερωτήματα.....	176
10.2 Συμπεράσματα της εργασίας.....	177
10.3 Περιορισμοί της μελέτης.....	178
10.4 Προτάσεις για περαιτέρω έρευνα.....	179
Βιβλιογραφία.....	180
Παράρτημα Α: «1 ^η μελέτη περίπτωσης: Σύστημα Ελέγχου LED με Arduino Uno στο Tinkercad».....	188
Παράρτημα Β: «2 ^η μελέτη περίπτωσης: Περιγραφή του Πειράματος της Άνωσης στο Εικονικό Εργαστήριο Klett/Pulior Virtual Lab.....	191

Κατάλογος Εικόνων / Σχημάτων

Εικόνα 2-1	Η Ιστορική εξέλιξη της UML.....	9
Εικόνα 2-2	Παράδειγμα Διάγραμμα Δομής.....	13
Εικόνα 2-3	Παράδειγμα Κλάσεων (Class Diagram).....	14
Εικόνα 2-4	Παράδειγμα Διάγραμμα Συστατικών.....	15
Εικόνα 2-5	Παράδειγμα Διάγραμμα Ανάπτυξης.....	16
Εικόνα 2-6	Παράδειγμα Διάγραμμα Πακέτων.....	17
Εικόνα 2-7	Παράδειγμα Διάγραμμα Συμπεριφοράς.....	18
Εικόνα 2-8	Παράδειγμα Περίπτωση Χρήσης (Use Case).....	19
Εικόνα 2-9	Παράδειγμα Διάγραμμα Ακολουθίας (Sequence Diagram).....	20
Εικόνα 2-10	Παράδειγμα Διάγραμμα Δραστηριοτήτων (Activity Diagram).....	21
Εικόνα 2-11	Παράδειγμα Διάγραμμα Καταστάσεων (State Machine Diagram).....	22
Εικόνα 2-12	Παράδειγμα Διάγραμμα Επικοινωνίας (Communication Diagram).....	22
Εικόνα 4-1	Ένα απλό μοντέλο περίπτωσης χρήσης.....	50
Εικόνα 4-2	Τύποι ηθοποιών ως μέρος μιας ιεραρχίας.....	52
Εικόνα 4-3	Διαχείριση Κλινικού Συστήματος.....	53
Εικόνα 4-4	Παράδειγμα σχέσης «include».....	55
Εικόνα 4-5	Η Περίπτωση Χρήσης «Σύνταξη Κειμένου».....	56
Εικόνα 4-6	Σχέση Γενίκευσης.....	57
Εικόνα 4-7	Κλάση Μάθημα: Ιδιότητες και λειτουργίες.....	59
Εικόνα 4-8	Ασφαλιστική Εταιρεία.....	60
Εικόνα 4-9	Γενίκευση.....	61
Εικόνα 4-10	Διάγραμμα κλάσεων – Παράδειγμα 1.....	62
Εικόνα 4-11	Διάγραμμα κλάσεων – Παράδειγμα 2.....	62
Εικόνα 4-12	Παράδειγμα Συσσώρευσης.....	63
Εικόνα 4-13	Παράδειγμα Σύνθεσης.....	64
Εικόνα 4-14	Στοιχεία Διαγράμματος Δραστηριοτήτων.....	66
Εικόνα 4-15	Διαχείριση Ραντεβού νοσοκομείου.....	66
Εικόνα 4-16	Εγγραφή φοιτητή σε σεμινάριο.....	67
Εικόνα 4-17	Σύστημα Διαχείρισης μαθημάτων για καθηγητή.....	68
Εικόνα 4-18	Διάγραμμα με ταυτόχρονες καταστάσεις για έναν υποψήφιο Πανελληνίων Εξετάσεων.....	70
Εικόνα 5-1	Διαδικασία Παραγγελίας Προϊόντος.....	78
Εικόνα 5-2	Παράλληλη Εκτέλεση Διεργασιών.....	79
Εικόνα 5-3	Activity Diagram με Swimlanes.....	79
Εικόνα 6-1	Μετατροπή κώδικα μέσω ενδιάμεσου μοντέλου.....	101
Εικόνα 6-2	Μετατροπή κώδικα μέσω ενδιάμεσου μοντέλου.....	101
Εικόνα 6-3	Οδηγία LLM στην παραγωγή κώδικα μέσω UML Activity Diagrams.....	102
Εικόνα 6-4	Οδηγία LLM στην παραγωγή κώδικα μέσω UML Activity Diagrams.....	102
Εικόνα 7-1	Προσομοίωση έξυπνου φωτισμού.....	115
Εικόνα 7-2	Κώδικας δραστηριότητας.....	116
Εικόνα 7-3	Δημιουργία Use Case Models για το σύστημα Ελέγχου LED με Arduino Uno στο Tinkercad.....	122

Εικόνα 7-4 Activity Diagram για το σύστημα Ελέγχου LED με Arduino Uno στο Tinkercad.....	126
Εικόνα 7-5 Χρήση Swimlanes για κατανομή ευθυνών για σύστημα Ελέγχου LED με Arduino Uno στο Tinkercad.....	128
Εικόνα 8-1 Πείραμα: Άνωση – Αρχή του Αρχιμήδη- Εικονικό Εργαστήριο Klett Physics – B’ Γυμνασίου.....	143
Εικόνα 8-2 Δημιουργία Use Case Models για την περιγραφή του πειράματος της Άνωσης στο Εικονικό Εργαστήριο Klett/Publitor Virtual Lab.....	151
Εικόνα 8-3 Δημιουργία Activity Diagrams.....	158

Κατάλογος Πινάκων

Πίνακας 3-1 Αντιστοίχιση τύπων UML διαγραμμάτων με επίπεδα και τύπους σημασιολογίας.....	29
Πίνακας 3-2 Συγκριτική αξιολόγηση τυπικών προσεγγίσεων σημασιολογίας UML.....	37
Πίνακας 4-1 Σύγκριση με Activity Diagrams.....	72
Πίνακας 4-2 Συσχέτιση Σημασιολογικών Σφαλμάτων και Τυπικών Προσεγγίσεων.....	75
Πίνακας 5-1 Σύγκριση Activity Diagrams και BPMN.....	87
Πίνακας 6-1 Activity Diagram σε Java.....	95
Πίνακας 7-1 Use Cases.....	121
Πίνακας 7-2 Σημασιολογική τεκμηρίωση του Activity Diagram.....	130
Πίνακας 7-3 Βαθμός κάλυψης των κριτηρίων για κάθε μοντέλο.....	136
Πίνακας 8-1 Actors για το Πείραμα της Άνωσης.....	148
Πίνακας 8-2 Use Cases για το Πείραμα της άνωσης.....	148
Πίνακας 8-3 Σύνδεση του Use Case Model με τους Μαθησιακούς Στόχους.....	156

Συντομογραφίες & Ακρωνύμια

ΔΕ	Διπλωματική Εργασία
ΕΑΠ	Ελληνικό Ανοικτό Πανεπιστήμιο
ΘΕ	Θεματική Ενότητα

1. Εισαγωγή

1.1 Αντικείμενο της εργασίας

Η ενοποιημένη γλώσσα μοντελοποίησης UML, λειτουργεί ως εργαλείο αναπαράστασης για την μοντελοποίηση της αρχιτεκτονικής, της δομής, της συμπεριφοράς και των αλληλεπιδράσεων εντός μιας εφαρμογής λογισμικού. Η σημασιολογία στη UML αναφέρεται στη σημασία ή την ερμηνεία που αποδίδεται στα διάφορα στοιχεία και σχέσεις που απεικονίζονται στα διαγράμματα της UML. Επιπρόσθετα, παρέχει ένα κοινό πεδίο για τους αναλυτές, προγραμματιστές και άλλα μέλη της ομάδας εργασίας για να κατανοήσουν και να ερμηνεύσουν τις προδιαγραφές σχεδιασμού έργων λογισμικού. Με την ενσωμάτωση της σημασιολογίας, η UML υπερβαίνει τον απλό περιγραφικό ρόλο των στατικών δομών και δυναμικών συμπεριφορών, αποδίδοντας τις αλληλεπιδράσεις των οντοτήτων ενός συστήματος λογισμικού. Η δημοτικότητα των UML Activity Diagrams τα καθιστά ένα ισχυρό εργαλείο επειδή μπορούν να μοντελοποιήσουν σύνθετες ροές τόσο στατικές όσο και δυναμικές, όπως οι πειραματικές διαδικασίες που λαμβάνουν χώρα σε εικονικά εργαστήρια φυσικών επιστημών.

Η μοντελοποίηση αποτελεί μία από τις σημαντικότερες δραστηριότητες στη σύγχρονη ανάπτυξη λογισμικού, καθώς επιτρέπει την αναπαράσταση της δομής, της συμπεριφοράς και των λειτουργικών απαιτήσεων ενός συστήματος πριν από την υλοποίησή του. Στο πλαίσιο αυτό, η Unified Modeling Language (UML) έχει καθιερωθεί ως το κυρίαρχο πρότυπο μοντελοποίησης λογισμικού, παρέχοντας ένα σύνολο διαγραμμάτων που υποστηρίζουν την περιγραφή διαφορετικών όψεων ενός συστήματος (OMG, 2023).

Παρά την ευρεία χρήση της UML, η αποτελεσματικότητα των μοντέλων δεν εξαρτάται αποκλειστικά από τη σωστή χρήση των συμβολισμών της γλώσσας αλλά και από τη δυνατότητα των μοντέλων να αποδίδουν με ακρίβεια το νόημα των εννοιών, των διαδικασιών και των αλληλεπιδράσεων που αναπαριστούν. Η διάσταση αυτή αφορά στη σημασιολογία των μοντέλων και αποτελεί κρίσιμο παράγοντα για τη διασφάλιση της ποιότητας και της αξιοπιστίας των πληροφοριακών συστημάτων (Seidewitz, 2003).

Αντικείμενο της παρούσας διπλωματικής εργασίας είναι η μελέτη της σημασιολογίας των UML μοντέλων και η διερεύνηση του ρόλου της σημασιολογικής πληρότητας στην

ανάπτυξη λογισμικού και στη Model-Driven Engineering (MDE). Η έρευνα επικεντρώνεται ιδιαίτερα στα Use Case Diagrams, Class Diagrams, Activity Diagrams και State Machine Diagrams, καθώς και στη σημασιολογική αξιολόγηση μοντέλων που χρησιμοποιούνται σε εκπαιδευτικά περιβάλλοντα και εικονικά εργαστήρια.

Η μελέτη υποστηρίζεται από δύο διαφορετικές μελέτες περίπτωσης: την ανάπτυξη και προσομοίωση ενός κυκλώματος Arduino Uno σε περιβάλλον Tinkercad και την εκτέλεση του πειράματος της Άνωσης σε εικονικό εργαστήριο φυσικής στο περιβάλλον Klett Book-App και Publior Virtual Lab.

1.2 Σκοπός και στόχοι της μελέτης

Σκοπός της διπλωματικής είναι η διερεύνηση του ρόλου της σημασιολογίας (semantics) στην ενοποιημένη γλώσσα μοντελοποίησης UML κατά το σχεδιασμό συστημάτων λογισμικού, ώστε να διασφαλίζεται κατά το δυνατόν η σαφήνεια και η ακρίβεια στην επικοινωνία μεταξύ των εμπλεκόμενων φορέων στη διαδικασία ανάπτυξης λογισμικού.

Σύμφωνα με τους Broy and Rumpe (2023), οι γλώσσες μοντελοποίησης (και προγραμματισμού) ορίζονται από τη σύνταξη (αυτό που βλέπουμε) και τη σημασιολογία τους (τι σημαίνει). Η σύνταξη διαμορφώνει την οπτική αναπαράσταση και η σημασιολογία ορίζει την έννοια των συντακτικών δομών. Οι χρήσιμες σημασιολογικές περιγραφές των νοημάτων συμβάλλουν στην κατανόηση και τη χρήση των γλωσσών. Ένας σημασιολογικός ορισμός ενσωματώνεται σε μια χρήσιμη θεωρία και ένα μεθοδολογικό πλαίσιο που ιδανικά προσφέρει ένα πλούσιο σύνολο αντιλήψεων που συμβάλλουν στην κατανόηση του τι μοντελοποιεί η γλώσσα, καθώς και τεχνικές (βοηθούμενες από εργαλεία) για την εξαγωγή ιδιοτήτων. Αυτό περιλαμβάνει ζητήματα προδιαγραφών και επαλήθευσης, για τη σαφήνεια των πληροφοριών και των σχέσεων ή ακόμα και για τη σύνθεση εκτελέσιμων αρχείων.

Σκοπός της παρούσας εργασίας είναι η διερεύνηση της συμβολής της σημασιολογικής ανάλυσης στη βελτίωση της ποιότητας των UML μοντέλων και η αξιολόγηση της σημασιολογικής πληρότητας σε πραγματικά εκπαιδευτικά σενάρια.

Για την επίτευξη του σκοπού αυτού τέθηκαν οι ακόλουθοι επιμέρους στόχοι:

- Η μελέτη της UML και της ιστορικής εξέλιξής της ως γλώσσας μοντελοποίησης.
- Η ανάλυση των διαφορετικών επιπέδων σημασιολογίας και των τυπικών σημασιολογικών προσεγγίσεων.
- Η διερεύνηση της σημασιολογικής διάστασης των βασικών UML διαγραμμάτων.
- Η μελέτη συνηθισμένων σημασιολογικών σφαλμάτων και ελλείψεων στη μοντελοποίηση.
- Η ανάλυση των Use Case Diagrams και Activity Diagrams ως εργαλείων μοντελοποίησης εκπαιδευτικών και πειραματικών διαδικασιών.
- Η διερεύνηση της σχέσης μεταξύ σημασιολογίας και Model-Driven Engineering.
- Η εφαρμογή των θεωρητικών εννοιών σε δύο εκπαιδευτικές μελέτες περίπτωσης.
- Η αξιολόγηση της σημασιολογικής πληρότητας των παραγόμενων UML μοντέλων.

Η επίτευξη των παραπάνω στόχων επιτρέπει την εξαγωγή συμπερασμάτων σχετικά με τον ρόλο της σημασιολογίας στη μοντελοποίηση και την αυτοματοποίηση της ανάπτυξης λογισμικού

1.3 Ερευνητικά ερωτήματα

Η παρούσα έρευνα οργανώνεται γύρω από τα ακόλουθα ερευνητικά ερωτήματα:

EE1: Ποιος είναι ο ρόλος της σημασιολογίας στη σωστή ερμηνεία και κατανόηση των UML μοντέλων;

EE2: Πώς επηρεάζει η σημασιολογική πληρότητα την ποιότητα και την αξιοπιστία των UML διαγραμμάτων;

EE3: Ποια είναι τα συνηθέστερα σημασιολογικά σφάλματα που εμφανίζονται στα Use Case, Class, Activity και State Diagrams;

EE4: Πώς μπορούν οι τυπικές σημασιολογικές προσεγγίσεις (OCL, Petri Nets, Formal Semantics) να συμβάλουν στη μείωση των σημασιολογικών ασυνεπειών;

EE5: Ποιος είναι ο ρόλος της σημασιολογικής πληρότητας στη Model-Driven Engineering και στους μετασχηματισμούς μοντέλων σε κώδικα;

EE6: Σε ποιο βαθμό τα UML μοντέλα που αναπτύσσονται για εκπαιδευτικά περιβάλλοντα και εικονικά εργαστήρια μπορούν να θεωρηθούν σημασιολογικά πλήρη;

1.4 Μεθοδολογία και δομή της εργασίας

Η παρούσα εργασία ακολουθεί ποιοτική ερευνητική προσέγγιση, βασισμένη σε βιβλιογραφική ανασκόπηση, θεωρητική ανάλυση και μελέτες περίπτωσης.

Αρχικά πραγματοποιείται ανασκόπηση της διεθνούς βιβλιογραφίας σχετικά με τη UML, τη σημασιολογία μοντέλων και τη Model-Driven Engineering. Στη συνέχεια αναλύονται οι βασικές σημασιολογικές προσεγγίσεις και εξετάζεται η εφαρμογή τους στα κυριότερα UML διαγράμματα.

Η θεωρητική ανάλυση συμπληρώνεται από δύο μελέτες περίπτωσης, οι οποίες επιλέχθηκαν ώστε να καλύπτουν διαφορετικά είδη εκπαιδευτικών δραστηριοτήτων:

- **Κύκλωμα Arduino Uno – LED Circuit στο Tinkercad**, το οποίο αντιπροσωπεύει ένα παράδειγμα μοντελοποίησης κυβερνοφυσικού και προγραμματιστικού συστήματος.
- **Πείραμα της Άνωσης στο Klett Book-App / Publior Virtual Lab**, το οποίο αντιπροσωπεύει ένα παράδειγμα εικονικού εργαστηρίου φυσικών επιστημών.

Για κάθε μελέτη περίπτωσης αναπτύσσονται UML μοντέλα και αξιολογείται η σημασιολογική τους πληρότητα βάσει κριτηρίων που προκύπτουν από τη βιβλιογραφία.

Η δομή της εργασίας οργανώνεται ως εξής:

- **Κεφάλαιο 2:** Θεωρητικό υπόβαθρο της UML. Το κεφάλαιο 2 συζητά θέματα όπως, ιστορική εξέλιξη της UML, το ρόλο της UML στο σχεδιασμό λογισμικού, τις βασικές κατηγορίες διαγραμμάτων UML, προκλήσεις στην ερμηνεία και στην κατανόηση των μοντέλων UML.
- **Κεφάλαιο 3:** Σημασιολογία UML μοντέλων. Συζητά θέματα όπως, ορισμός και βασικές έννοιες σημασιολογίας και τη σημασιολογία στην UML.
- **Κεφάλαιο 4:** Σημασιολογική ανάλυση βασικών UML διαγραμμάτων. Παρουσιάζονται η σημασιολογία Διαγραμμάτων Περιπτώσεων Χρήσης (Use Case), η σημασιολογία Διαγραμμάτων Κλάσεων, η σημασιολογία Διαγραμμάτων Δραστηριοτήτων (Activity Diagrams), η σημασιολογία Διαγραμμάτων Κατάστασης

(State Machine Diagrams), καθώς και τα συνηθισμένα λάθη και ελλείψεις που εμφανίζονται στη σημασιολογική ερμηνεία.

- **Κεφάλαιο 5:** Activity Diagrams και εφαρμογές σε εικονικά εργαστήρια. Παρουσιάζονται ο ρόλος των Activity Diagrams στη μοντελοποίηση συστημάτων, οι σημασιολογικές δομές στα Activity Diagrams, η χρήση των Activity Diagrams για μοντελοποίηση πειραματικών διαδικασιών σε εικονικά εργαστήρια.
- **Κεφάλαιο 6:** Model-Driven Engineering και σημασιολογικοί μετασχηματισμοί. Περιλαμβάνει θέματα όπως: Model-Driven Engineering (MDE) και σημασιολογία, Τεχνικές αυτόματης παραγωγής κώδικα, ο ρόλος της σημασιολογικής συνέπειας στη μετατροπή μοντέλων, μελέτες περίπτωσης από τη βιβλιογραφία, εργαλεία UML και δυνατότητες παραγωγής κώδικα.
- **Κεφάλαιο 7:** Μελέτη περίπτωσης 1 και αξιολόγηση σημασιολογικής πληρότητας. Περιλαμβάνει μοντελοποίηση πειραματικής διαδικασίας για το **Κύκλωμα Arduino Uno – LED Circuit στο Tinkercad**, το οποίο αντιπροσωπεύει ένα παράδειγμα μοντελοποίησης κυβερνοφυσικού και προγραμματιστικού συστήματος.
- **Κεφάλαιο 8:** Μελέτη περίπτωσης 2 και αξιολόγηση σημασιολογικής πληρότητας. Περιλαμβάνει μοντελοποίηση πειραματικής διαδικασίας για το **Πείραμα της Άνωσης στο Klett Book-App / Publior Virtual Lab**, το οποίο αντιπροσωπεύει ένα παράδειγμα εικονικού εργαστηρίου φυσικών επιστημών.
- **Κεφάλαιο 9:** Συζήτηση – Συμπεράσματα: Συζητούνται τα αποτελέσματα της εργασίας.
- **Κεφάλαιο 10:** Συμπεράσματα και μελλοντικές κατευθύνσεις: Συζητούνται τα συμπεράσματα της εργασίας, οι περιορισμοί της μελέτης και οι προτάσεις για περαιτέρω έρευνα.

1.5 Συμβολή της Εργασίας στη Διεθνή Βιβλιογραφία

Η συμβολή της παρούσας εργασίας εντοπίζεται σε τρία βασικά επίπεδα:

Πρώτον, επιχειρείται η συστηματική σύνδεση της θεωρίας της σημασιολογίας με την πρακτική χρήση της UML, γεφυρώνοντας το χάσμα μεταξύ θεωρητικών σημασιολογικών

προσεγγίσεων και εφαρμοσμένης μοντελοποίησης. Παρότι η διεθνής βιβλιογραφία περιλαμβάνει εκτεταμένες αναφορές στη UML και στη Model-Driven Engineering, η σημασιολογική διάσταση των μοντέλων συχνά αντιμετωπίζεται αποσπασματικά (Brambilla, Cabot & Wimmer, 2017).

Δεύτερον, η εργασία προτείνει ένα πλαίσιο αξιολόγησης της σημασιολογικής πληρότητας UML μοντέλων μέσω της ανάλυσης διαφορετικών τύπων διαγραμμάτων και της εφαρμογής τους σε πραγματικές εκπαιδευτικές διαδικασίες.

Τρίτον, η αξιοποίηση δύο εκπαιδευτικών μελετών περίπτωσης παρέχει ένα πρακτικό πλαίσιο διερεύνησης της σημασιολογικής ποιότητας των μοντέλων σε διαφορετικά μαθησιακά περιβάλλοντα. Η προσέγγιση αυτή συνεισφέρει στη βιβλιογραφία που αφορά τη μοντελοποίηση εκπαιδευτικών συστημάτων και εικονικών εργαστηρίων, έναν τομέα που παρουσιάζει αυξανόμενο ερευνητικό ενδιαφέρον τα τελευταία χρόνια (Ma & Nickerson, 2006; Di Ruscio et al., 2021).

Συνολικά, η εργασία αναδεικνύει ότι η σημασιολογική πληρότητα δεν αποτελεί απλώς θεωρητική ιδιότητα των UML μοντέλων, αλλά βασική προϋπόθεση για την αξιόπιστη ανάπτυξη λογισμικού, τη Model-Driven Engineering και την επιτυχή εφαρμογή μοντέλων σε εκπαιδευτικά και πληροφοριακά συστήματα.

2. Η UML ως γλώσσα μοντελοποίησης

2.1 Ιστορική εξέλιξη της UML

Η Ενοποιημένη Γλώσσα Μοντελοποίησης (Unified Modeling Language – UML) αποτελεί το πλέον διαδεδομένο πρότυπο οπτικής μοντελοποίησης συστημάτων λογισμικού. Η ανάπτυξή της συνδέεται άμεσα με την εξέλιξη της αντικειμενοστραφούς τεχνολογίας και με την ανάγκη δημιουργίας ενός κοινού πλαισίου περιγραφής σύνθετων συστημάτων λογισμικού. Πριν από την εμφάνιση της UML, η κοινότητα της μηχανικής λογισμικού χρησιμοποιούσε πληθώρα διαφορετικών μεθοδολογιών ανάλυσης και σχεδίασης, γεγονός που δημιουργούσε προβλήματα επικοινωνίας, επαναχρησιμοποίησης γνώσης και διαλειτουργικότητας μεταξύ ομάδων ανάπτυξης (Booch et al., 2005).

Οι ρίζες της UML (1980–1994)

Κατά τις δεκαετίες του 1980 και των αρχών του 1990 αναπτύχθηκαν πολυάριθμες μέθοδοι και αναπαραστάσεις για αντικειμενοστραφή ανάλυση και σχεδίαση. Ανάμεσα σε αυτές ξεχώριζαν:

- Η **Booch Method**, από τον Grady Booch, η οποία έδινε έμφαση στη λεπτομερή σχεδίαση αντικειμένων.
- Η **Object-Modeling Technique (OMT)**, που αναπτύχθηκε από τον James Rumbaugh, και ήταν ιδιαίτερα δημοφιλής σε μεγάλα έργα λογισμικού.
- Η **Objectory Method**, από τον Ivar Jacobson, η οποία εισήγαγε για πρώτη φορά τη θεμελιώδη έννοια των *use cases*.

Η πληθώρα αυτών των μεθόδων και η έλλειψη κοινών προτύπων τελικά οδήγησαν σε προβλήματα επικοινωνίας μεταξύ ομάδων ανάπτυξης, με αποτέλεσμα να προκύψει η ανάγκη για ενοποίηση.

Η ενοποίηση των μεθόδων και η γέννηση της UML (1994–1997)

Το 1994 ο Grady Booch και ο James Rumbaugh ξεκίνησαν να συνεργάζονται στην προσπάθεια δημιουργίας μιας ενιαίας γλώσσας, η οποία θα συνδύαζε τα πλεονεκτήματα των Booch και OMT. Το έργο αυτό ονομάστηκε *Unified Method*.

Το 1995 προσχώρησε στο εγχείρημα και ο Ivar Jacobson, φέρνοντας την εμπειρία του από τη μέθοδο Objectory. Έτσι γεννήθηκαν οι πρώτες εκδόσεις της **Unified Modeling Language**, οι οποίες άρχισαν να διαδίδονται γρήγορα στη βιομηχανία λογισμικού.

Το 1997 η UML υποβλήθηκε επίσημα στον οργανισμό Object Management Group, ο οποίος την αποδέχτηκε ως πρότυπο και ξεκίνησε την επίσημη διαδικασία τυποποίησης.

Τυποποίηση και εξέλιξη της UML (1997–σήμερα)

Το 1997 δημοσιεύθηκε η επίσημη έκδοση **UML 1.0** από τον OMG. Έκτοτε, η UML πέρασε από αρκετές σημαντικές εκδόσεις:

- **UML 1.x (1997–2003):** Βελτιώσεις στα διαγράμματα, καθορισμός πιο αυστηρής σύνταξης και πρώτες προσπάθειες ορισμού σημασιολογικών κανόνων.
- **UML 2.0 (2005):** Μεγάλος ανασχεδιασμός της γλώσσας, εισαγωγή νέων διαγραμμάτων (όπως τα *Interaction Overview* και *Timing Diagrams*) και βελτίωση της συνθετικής ισχύος της.
- **UML 2.x (2005–σήμερα):** Σταδιακές αναθεωρήσεις, διευκρινίσεις σημασιολογίας και προσαρμογές στις νέες ανάγκες της βιομηχανίας, συμπεριλαμβανομένων σύνθετων καταναεμημένων συστημάτων και μοντέλων λογισμικού βασισμένων σε πλατφόρμες.

Πλέον, η UML αποτελεί βασικό εργαλείο τόσο στην αντικειμενοστραφή ανάλυση όσο και στη μοντελοποίηση συστημάτων ευρείας κλίμακας.

Χρονογραμμή της ιστορικής εξέλιξης της UML

1980–1990: Ανάπτυξη μεθόδων Booch, OMT και Objectory

1994: Έναρξη συνεργασίας Booch–Rumbaugh (Unified Method)

1995: Συμμετοχή Jacobson και διαμόρφωση των πρώτων εκδόσεων UML

1997: Υποβολή και έγκριση της UML από τον OMG (Έκδοση 1.0)

1997–2003: Διαδοχικές εκδόσεις UML 1.x

2005: Κυκλοφορία UML 2.0

2005–σήμερα: Εκδόσεις UML 2.x και συνεχής τυποποίηση από τον OMG

Η Ιστορική Εξέλιξη της UML



Εικόνα 2-1 Η Ιστορική Εξέλιξη της UML

Κριτική αποτίμηση της ιστορικής εξέλιξης

Η ιστορική πορεία της UML θεωρείται γενικά επιτυχημένη, καθώς κατάφερε να ενοποιήσει διαφορετικές σχολές σκέψης και να καθιερωθεί ως διεθνές πρότυπο μοντελοποίησης. Παράλληλα, όμως, η συνεχής επέκταση της γλώσσας είχε ως αποτέλεσμα την αύξηση της πολυπλοκότητάς της.

Σύμφωνα με τους France και Rumpfe (2007), η UML εξελίχθηκε από ένα σχετικά απλό εργαλείο επικοινωνίας σε μια ιδιαίτερα πλούσια γλώσσα μοντελοποίησης, γεγονός που ενίσχυσε τις δυνατότητές της αλλά αύξησε τις απαιτήσεις εκμάθησης και εφαρμογής. Η εξέλιξη αυτή ανέδειξε ένα κρίσιμο ζήτημα: ενώ η σύνταξη της UML τυποποιήθηκε σε μεγάλο βαθμό, η σημασιολογική ερμηνεία πολλών μοντέλων εξακολουθεί να εξαρτάται από το πλαίσιο χρήσης και από την εμπειρία των χρηστών.

Η παρατήρηση αυτή αποτελεί τη βάση για τη σύγχρονη έρευνα γύρω από τη σημασιολογία της μοντελοποίησης, η οποία επιχειρεί να εξασφαλίσει ότι τα μοντέλα δεν είναι απλώς συντακτικά ορθά αλλά και σημασιολογικά σαφή και συνεπή.

2.2 Ο ρόλος της UML στον σχεδιασμό Λογισμικού

Ο σχεδιασμός λογισμικού αποτελεί μία από τις σημαντικότερες δραστηριότητες του κύκλου ζωής ανάπτυξης συστημάτων. Στο πλαίσιο αυτό, η UML λειτουργεί ως κοινή γλώσσα επικοινωνίας μεταξύ αναλυτών, σχεδιαστών, προγραμματιστών και ενδιαφερόμενων μερών, επιτρέποντας την οπτική αναπαράσταση πολύπλοκων δομών και διαδικασιών.

Η αξία της UML δεν περιορίζεται στην τεκμηρίωση. Αντιθέτως, συμβάλλει στη συστηματική κατανόηση των απαιτήσεων, στον σχεδιασμό της αρχιτεκτονικής, στην περιγραφή της συμπεριφοράς του συστήματος και στη διατήρηση της γνώσης καθ' όλη τη διάρκεια ζωής ενός έργου λογισμικού (OMG, 2017).

Ο ρόλος της UML στην ανάλυση και κατανόηση απαιτήσεων

Κατά τη φάση της ανάλυσης απαιτήσεων, η UML επιτρέπει την αναπαράσταση των λειτουργικών απαιτήσεων μέσω των διαγραμμάτων περιπτώσεων χρήσης (Use Case Diagrams). Τα διαγράμματα αυτά συμβάλλουν στη γεφύρωση του χάσματος μεταξύ τεχνικών και μη τεχνικών εμπλεκομένων, καθώς περιγράφουν τις αλληλεπιδράσεις χρηστών και συστήματος με τρόπο κατανοητό και οπτικά σαφή (Jacobson et al., 1992).

Η χρήση περιπτώσεων χρήσης διευκολύνει τον εντοπισμό ελλείψεων, ασαφειών και αντικρουόμενων απαιτήσεων ήδη από τα πρώτα στάδια ανάπτυξης. Ως αποτέλεσμα, μειώνεται το κόστος διορθώσεων που θα απαιτούνταν σε μεταγενέστερες φάσεις του έργου.

Η UML ως εργαλείο αρχιτεκτονικού σχεδιασμού

Σε επίπεδο αρχιτεκτονικής, η UML προσφέρει διαγράμματα που επιτρέπουν την απεικόνιση της συνολικής δομής ενός συστήματος. Τα διαγράμματα συστατικών (Component Diagrams) και τα διαγράμματα ανάπτυξης (Deployment Diagrams) χρησιμοποιούνται για την περιγραφή υποσυστημάτων, εξαρτήσεων και υποδομών εκτέλεσης.

Η δυνατότητα αυτή αποκτά ιδιαίτερη σημασία σε κατανεμημένα συστήματα και αρχιτεκτονικές μικροϋπηρεσιών (microservices), όπου η κατανόηση των αλληλεπιδράσεων μεταξύ επιμέρους στοιχείων είναι κρίσιμη για τη σωστή λειτουργία του συστήματος (Brambilla et al., 2017).

Η UML στη λεπτομερή σχεδίαση και στην υλοποίηση λογισμικού

Η UML υποστηρίζει επίσης τη λεπτομερή σχεδίαση μέσω διαγραμμάτων κλάσεων, αντικειμένων, ακολουθίας και καταστάσεων. Τα διαγράμματα αυτά επιτρέπουν την περιγραφή της εσωτερικής δομής και της δυναμικής συμπεριφοράς ενός συστήματος με υψηλό βαθμό λεπτομέρειας.

Ιδιαίτερα τα διαγράμματα κλάσεων θεωρούνται θεμελιώδες εργαλείο της αντικειμενοστραφούς σχεδίασης, καθώς περιγράφουν τις έννοιες του πεδίου εφαρμογής και τις σχέσεις μεταξύ τους. Παράλληλα, τα διαγράμματα ακολουθίας διευκολύνουν την κατανόηση της συνεργασίας μεταξύ αντικειμένων κατά την εκτέλεση συγκεκριμένων σεναρίων (Rumbaugh et al., 1991).

Επικοινωνία και τεκμηρίωση

Ένα από τα σημαντικότερα πλεονεκτήματα της UML είναι η λειτουργία της ως εργαλείου επικοινωνίας. Η χρήση ενός τυποποιημένου συμβολισμού επιτρέπει τη δημιουργία κοινής αντίληψης για τη δομή και τη συμπεριφορά του συστήματος μεταξύ όλων των εμπλεκομένων.

Παράλληλα, η UML λειτουργεί ως μέσο τεκμηρίωσης που διατηρεί τη γνώση του συστήματος πέρα από την αρχική ομάδα ανάπτυξης. Η τεκμηρίωση αυτή διευκολύνει τη συντήρηση, την επέκταση και τη μεταφορά του συστήματος σε νέα περιβάλλοντα ανάπτυξης (Booch et al., 2005).

UML και Model-Driven Engineering

Η εξάπλωση της UML συνδέθηκε στενά με την ανάπτυξη των προσεγγίσεων Model-Driven Engineering (MDE) και Model-Driven Architecture (MDA). Στο πλαίσιο αυτό, τα μοντέλα δεν αντιμετωπίζονται απλώς ως τεκμηρίωση αλλά ως πρωτεύοντα τεχνουργήματα ανάπτυξης από τα οποία μπορούν να παραχθούν αυτόματα τμήματα κώδικα, δοκιμών και τεκμηρίωσης (Brambilla et al., 2017).

Η προσέγγιση αυτή ενισχύει την παραγωγικότητα και συμβάλλει στη μείωση λαθών που προκύπτουν από ασυνέπειες μεταξύ μοντέλου και υλοποίησης. Παράλληλα, καθιστά τη

σημασιολογική ακρίβεια των μοντέλων ακόμη πιο σημαντική, καθώς τα μοντέλα χρησιμοποιούνται πλέον ως βάση για αυτοματοποιημένες διαδικασίες.

Περιορισμοί της UML ως μέσου αναπαράστασης γνώσης

Παρά τα σημαντικά πλεονεκτήματά της, η UML δεν αποτελεί πανάκεια για όλα τα προβλήματα σχεδίασης λογισμικού. Η μεγάλη ποικιλία διαγραμμάτων και συμβολισμών αυξάνει την εκφραστική δύναμη της γλώσσας, αλλά παράλληλα επιβαρύνει τη διαδικασία εκμάθησης και συντήρησης των μοντέλων (Whittle et al., 2014).

Επιπλέον, η ύπαρξη τυποποιημένου συμβολισμού δεν διασφαλίζει απαραίτητα κοινή κατανόηση. Δύο διαφορετικοί αναλυτές μπορεί να αποδώσουν διαφορετικές ερμηνείες στο ίδιο μοντέλο, ιδιαίτερα όταν αυτό περιγράφει σύνθετες έννοιες ή επιχειρησιακούς κανόνες που δεν αποτυπώνονται πλήρως στα διαγράμματα (France & Rumpe, 2007).

Το πρόβλημα αυτό αναδεικνύει έναν θεμελιώδη περιορισμό της UML: η γλώσσα περιγράφει αποτελεσματικά τη δομή και τη συμπεριφορά ενός συστήματος, αλλά δεν διασφαλίζει πάντοτε την ακριβή αναπαράσταση του νοήματος των εννοιών που περιέχει. Ως αποτέλεσμα, η κατανόηση των μοντέλων συχνά εξαρτάται από εξωγενείς παράγοντες, όπως η εμπειρία των χρηστών, η τεκμηρίωση και το πλαίσιο εφαρμογής.

Η παρατήρηση αυτή οδηγεί σε ένα ευρύτερο ερευνητικό ερώτημα: «πώς μπορεί να διασφαλιστεί ότι διαφορετικοί χρήστες θα αποδίδουν το ίδιο νόημα στα στοιχεία ενός μοντέλου»; Η απάντηση στο ερώτημα αυτό σχετίζεται άμεσα με τη σημασιολογία της μοντελοποίησης και αποτελεί αντικείμενο του επόμενου κεφαλαίου.

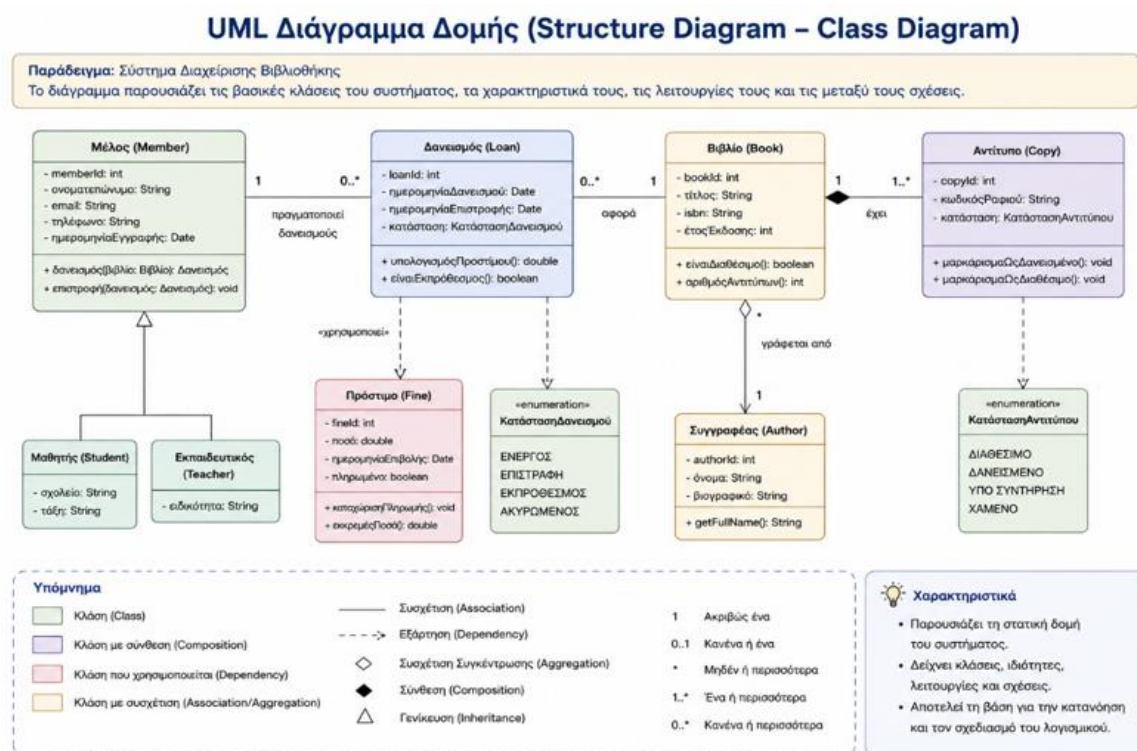
2.3 Βασικές Κατηγορίες Διαγραμμάτων UML

Η UML παρέχει ένα ολοκληρωμένο σύνολο διαγραμμάτων που επιτρέπουν την αναπαράσταση διαφορετικών όψεων ενός συστήματος λογισμικού. Η φιλοσοφία της βασίζεται στην παραδοχή ότι κανένα μεμονωμένο διάγραμμα δεν είναι επαρκές για να περιγράψει πλήρως ένα πολύπλοκο σύστημα. Αντίθετα, απαιτείται ένας συνδυασμός διαφορετικών αναπαραστάσεων, καθεμία από τις οποίες εστιάζει σε συγκεκριμένες πτυχές του συστήματος (OMG, 2017).

Τα διαγράμματα της UML διακρίνονται σε δύο βασικές κατηγορίες: τα διαγράμματα δομής (Structural Diagrams) και τα διαγράμματα συμπεριφοράς (Behavioral Diagrams). Η πρώτη κατηγορία επικεντρώνεται στη στατική οργάνωση του συστήματος, ενώ η δεύτερη περιγράφει τη δυναμική συμπεριφορά και τις αλληλεπιδράσεις μεταξύ των στοιχείων του (Booch et al., 2005).

Διαγράμματα δομής (Structural Diagrams)

Τα διαγράμματα δομής χρησιμοποιούνται για την περιγραφή της αρχιτεκτονικής και της στατικής οργάνωσης ενός συστήματος. Μέσω αυτών αποτυπώνονται οι βασικές οντότητες, οι σχέσεις μεταξύ τους και οι εξαρτήσεις που αναπτύσσονται κατά τη λειτουργία του συστήματος.



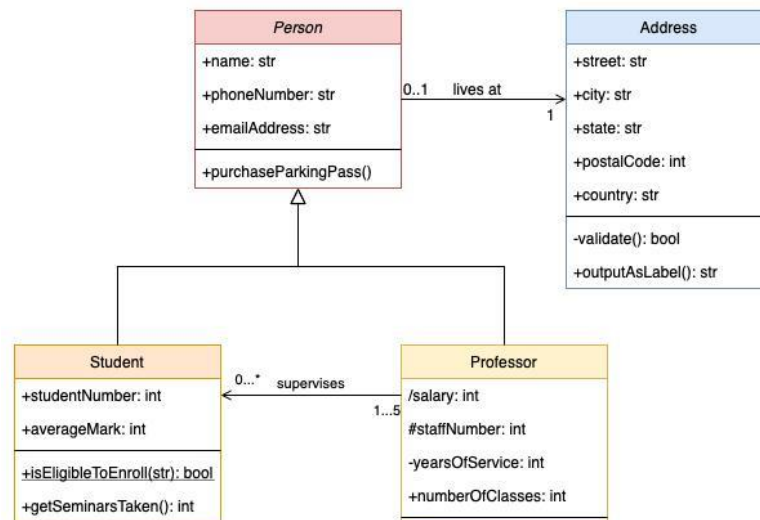
Εικ 2-2 Παράδειγμα Διάγραμμα Δομής

Διάγραμμα κλάσεων (Class Diagram)

Το διάγραμμα κλάσεων αποτελεί το πλέον αναγνωρίσιμο διάγραμμα της UML και θεωρείται το θεμέλιο της αντικειμενοστραφούς μοντελοποίησης. Απεικονίζει τις κλάσεις

ενός συστήματος, τα χαρακτηριστικά τους, τις λειτουργίες τους και τις σχέσεις που αναπτύσσονται μεταξύ τους, όπως συσχετίσεις, γενικεύσεις, συνθέσεις και εξαρτήσεις (Rumbaugh et al., 1991).

Το διάγραμμα κλάσεων είναι ιδιαίτερα χρήσιμο κατά τη φάση της ανάλυσης και της σχεδίασης, καθώς παρέχει μια αφαιρετική αλλά σαφή εικόνα της δομής του συστήματος.

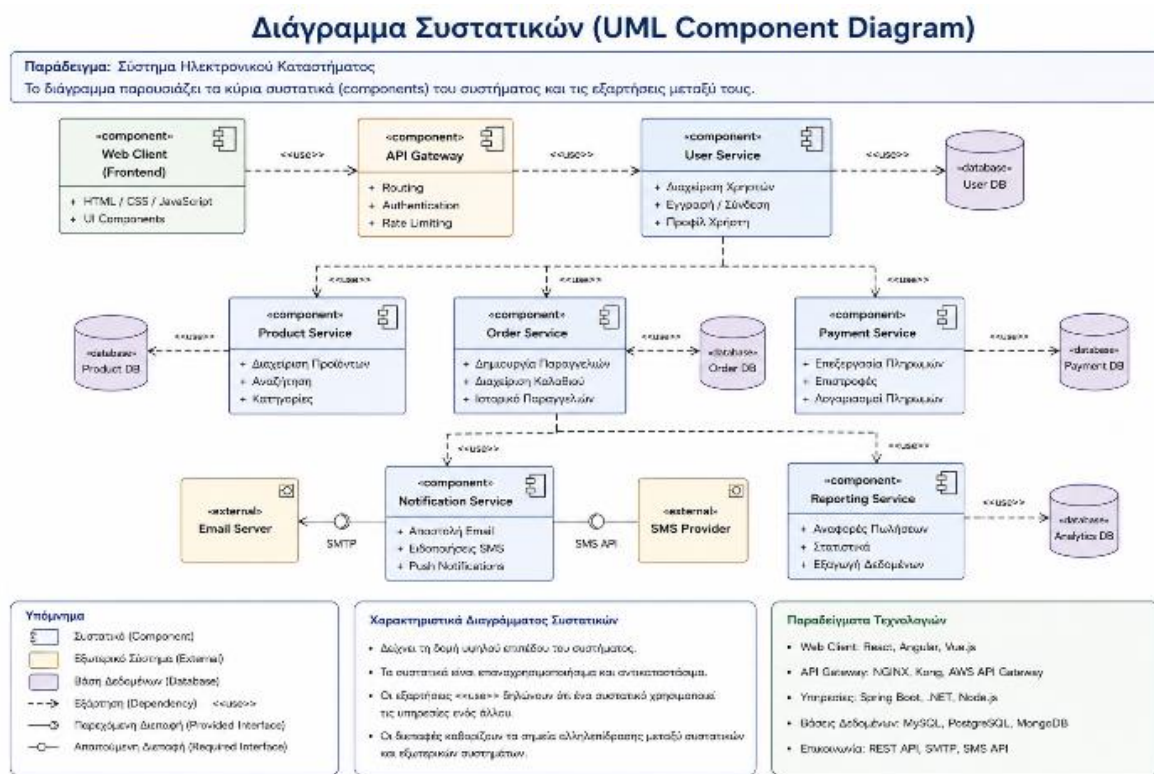


Εικόνα 2-3 Διάγραμμα κλάσεων

Διάγραμμα συστατικών (Component Diagram)

Τα διαγράμματα συστατικών χρησιμοποιούνται για την περιγραφή των λογικών μονάδων ενός συστήματος και των εξαρτήσεων μεταξύ τους. Τα συστατικά μπορεί να αντιστοιχούν σε βιβλιοθήκες λογισμικού, υπηρεσίες, υποσυστήματα ή ανεξάρτητα modules.

Η σημασία τους έχει αυξηθεί ιδιαίτερα τα τελευταία χρόνια λόγω της ευρείας χρήσης αρχιτεκτονικών μικροϋπηρεσιών, όπου η κατανόηση των σχέσεων μεταξύ υπηρεσιών είναι κρίσιμη για την αξιοπιστία και τη συντήρηση του συστήματος (Brambilla et al., 2017).

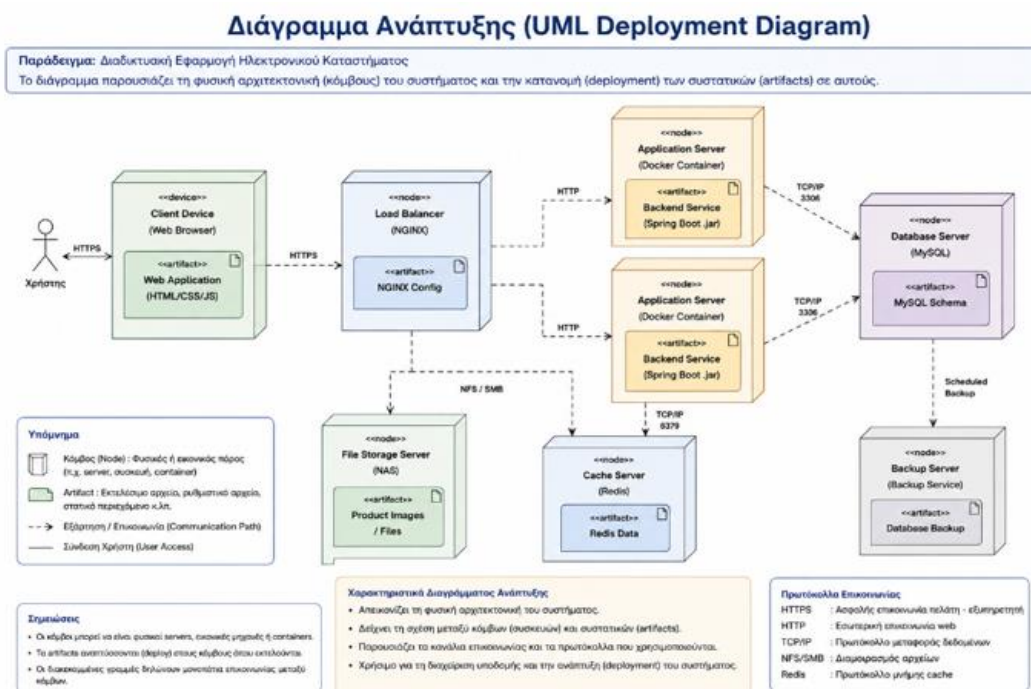


Εικ 2-4 Παράδειγμα Διάγραμμα Συστατικών

Διάγραμμα ανάπτυξης (Deployment Diagram)

Τα διαγράμματα ανάπτυξης περιγράφουν τη φυσική διάσταση του συστήματος, δηλαδή την κατανομή λογισμικού και υλικού σε κόμβους εκτέλεσης. Χρησιμοποιούνται για την απεικόνιση εξυπηρετητών, βάσεων δεδομένων, δικτυακών υποδομών και συσκευών που συμμετέχουν στην τελική λειτουργία του συστήματος.

Η χρησιμότητά τους είναι ιδιαίτερα εμφανής σε κατανεμημένα συστήματα και περιβάλλοντα υπολογιστικού νέφους (cloud computing), όπου η κατανόηση της φυσικής αρχιτεκτονικής αποτελεί βασική προϋπόθεση για την αποτελεσματική ανάπτυξη και διαχείριση εφαρμογών.



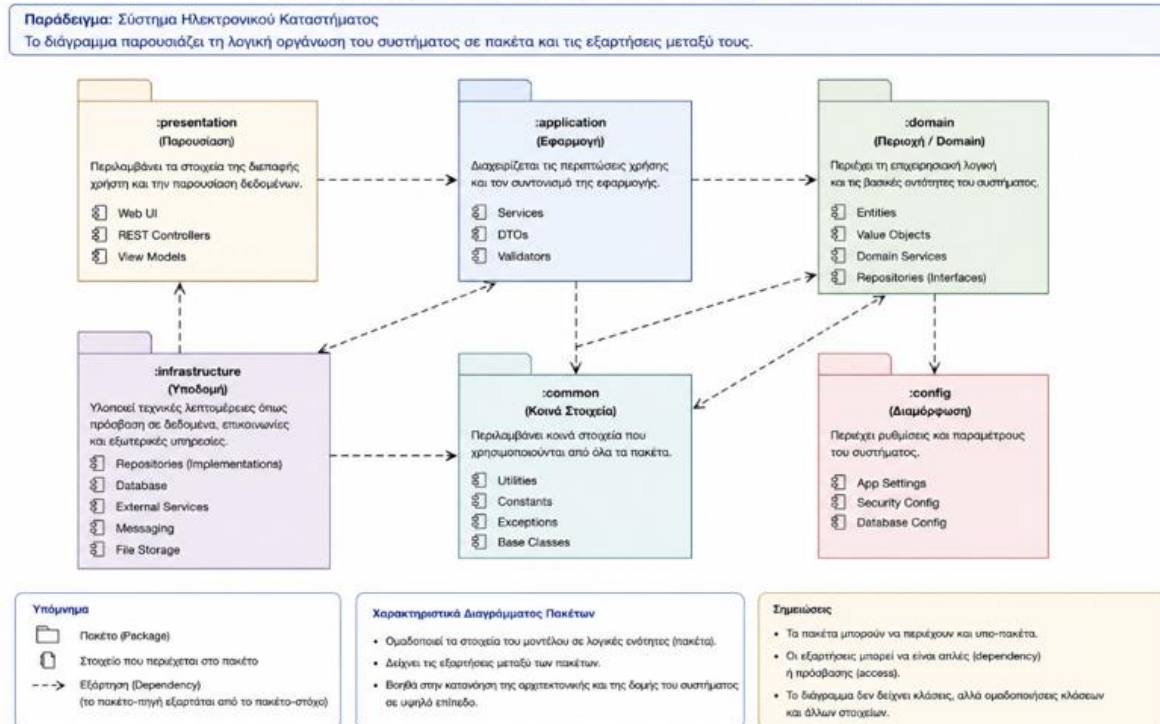
Εικ 2-5 Παράδειγμα Διάγραμμα Ανάπτυξης

Διάγραμμα πακέτων (Package Diagram)

Τα διαγράμματα πακέτων επιτρέπουν την οργάνωση μεγάλων συστημάτων σε λογικές ενότητες. Μέσω αυτών διευκολύνεται η διαχείριση της πολυπλοκότητας, καθώς οι σχετικές κλάσεις και λειτουργίες ομαδοποιούνται σε πακέτα με σαφώς ορισμένες ευθύνες.

Η προσέγγιση αυτή συμβάλλει στη βελτίωση της επεκτασιμότητας και της συντηρησιμότητας του λογισμικού.

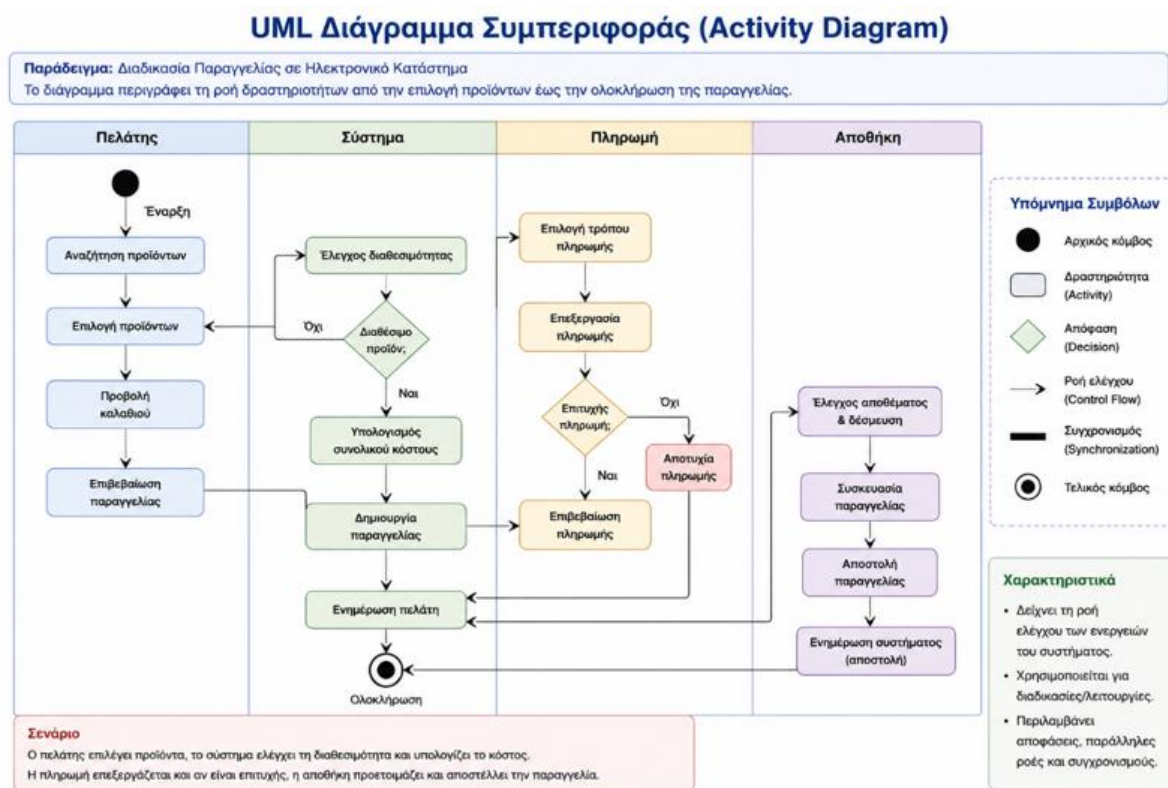
Διάγραμμα Πακέτων (UML Package Diagram)



Εικ 2-6 Παράδειγμα Διάγραμμα Πακέτων

Διαγράμματα συμπεριφοράς (Behavioral Diagrams)

Σε αντίθεση με τα διαγράμματα δομής, τα διαγράμματα συμπεριφοράς περιγράφουν τον τρόπο με τον οποίο λειτουργεί ένα σύστημα κατά την εκτέλεσή του. Εστιάζουν στις αλληλεπιδράσεις, στις διαδικασίες και στις αλλαγές κατάστασης που συμβαίνουν κατά τη διάρκεια της λειτουργίας του.

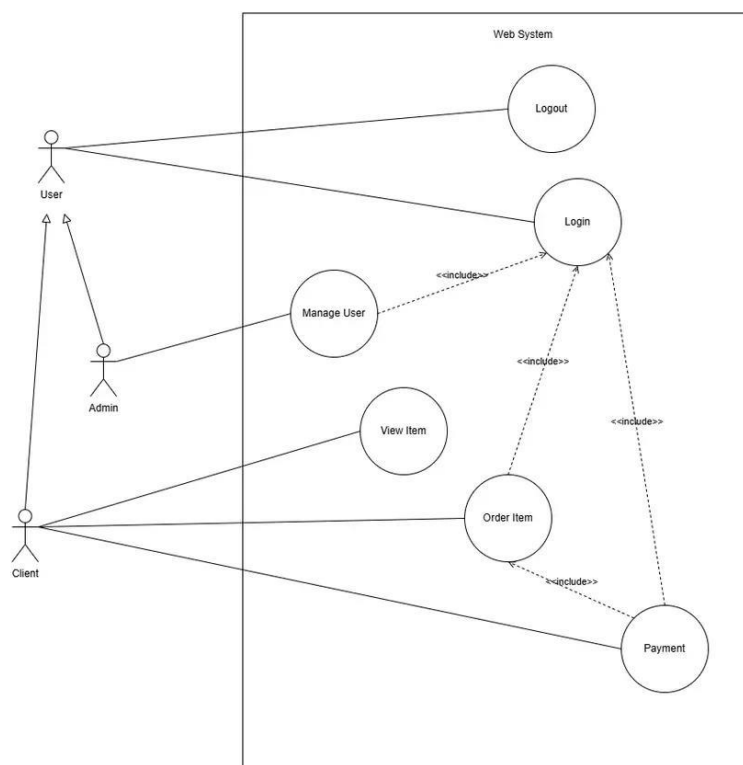


Εικ 2-7 Παράδειγμα Διάγραμμα Συμπεριφοράς

Διάγραμμα περιπτώσεων χρήσης (Use Case Diagram)

Τα διαγράμματα περιπτώσεων χρήσης αποτελούν ένα από τα σημαντικότερα εργαλεία ανάλυσης απαιτήσεων. Περιγράφουν τις αλληλεπιδράσεις μεταξύ χρηστών (actors) και συστήματος και αποτυπώνουν τις βασικές λειτουργίες που παρέχει το λογισμικό (Jacobson et al., 1992).

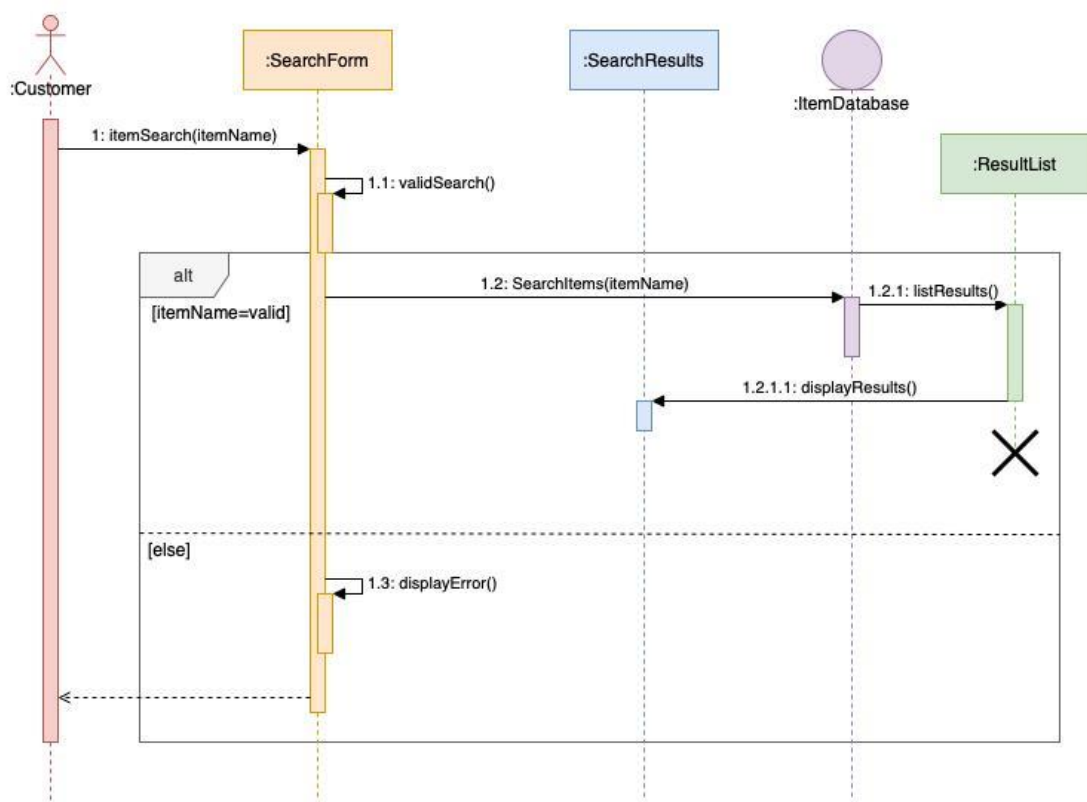
Η ιδιαίτερη αξία τους έγκειται στο γεγονός ότι μπορούν να γίνουν κατανοητά όχι μόνο από τεχνικούς αλλά και από επιχειρησιακούς χρήστες, διευκολύνοντας την επικοινωνία μεταξύ διαφορετικών ομάδων ενδιαφερομένων.



Εικόνα 2-8 Παράδειγμα Use Case Diagram

Διάγραμμα ακολουθίας (Sequence Diagram)

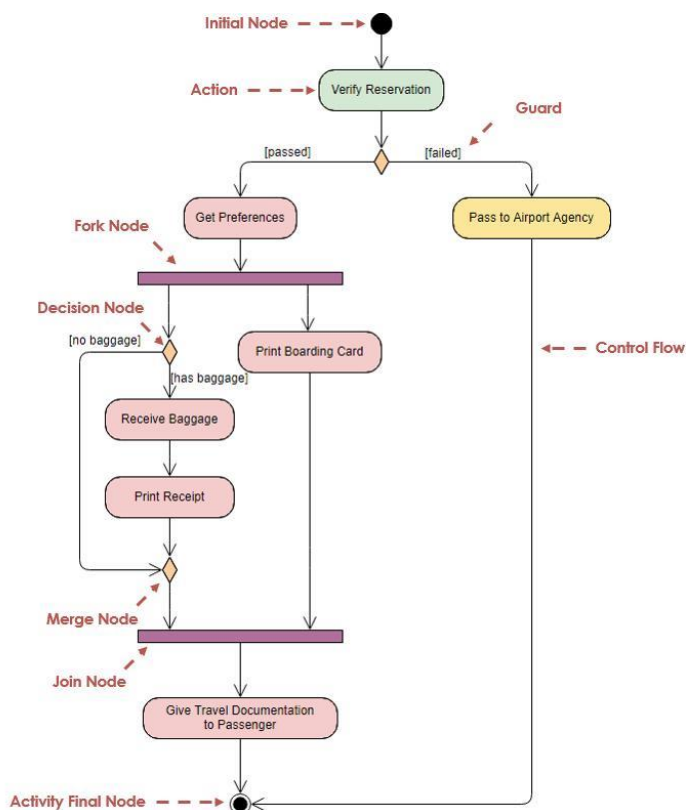
Τα διαγράμματα ακολουθίας παρουσιάζουν τη χρονική σειρά ανταλλαγής μηνυμάτων μεταξύ αντικειμένων. Μέσω αυτών αναδεικνύεται η δυναμική συμπεριφορά του συστήματος κατά την εκτέλεση ενός συγκεκριμένου σεναρίου. Η χρήση τους είναι ιδιαίτερα χρήσιμη κατά την ανάλυση σύνθετων επιχειρησιακών διαδικασιών και τη σχεδίαση αλληλεπιδράσεων μεταξύ διαφορετικών υπηρεσιών ή υποσυστημάτων.



Εικόνα 2- 9 Παράδειγμα Διάγραμμα Ακολουθίας (Sequence Diagram)

Διάγραμμα δραστηριοτήτων (Activity Diagram)

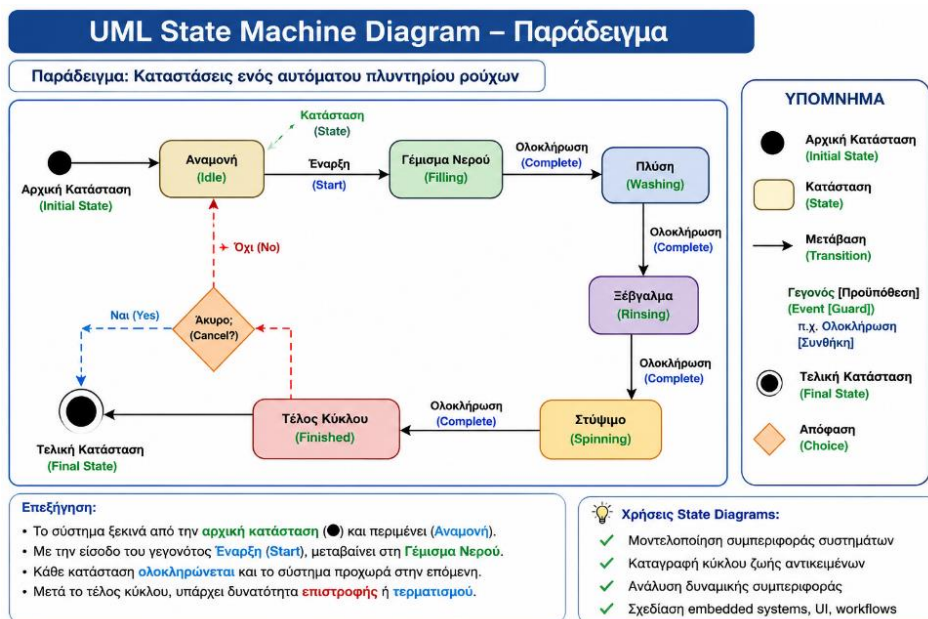
Τα διαγράμματα δραστηριοτήτων χρησιμοποιούνται για την περιγραφή ροών εργασίας και επιχειρησιακών διαδικασιών. Παρουσιάζουν τη σειρά εκτέλεσης ενεργειών, τις αποφάσεις και τις παράλληλες δραστηριότητες που μπορεί να πραγματοποιηθούν κατά την εκτέλεση μιας διαδικασίας. Χρησιμοποιούνται ευρέως τόσο στη μηχανική λογισμικού όσο και στη μοντελοποίηση επιχειρησιακών διαδικασιών.



Εικόνα 2-10 Παράδειγμα Διάγραμμα Δραστηριοτήτων (Activity Diagram)

Διάγραμμα καταστάσεων (State Machine Diagram)

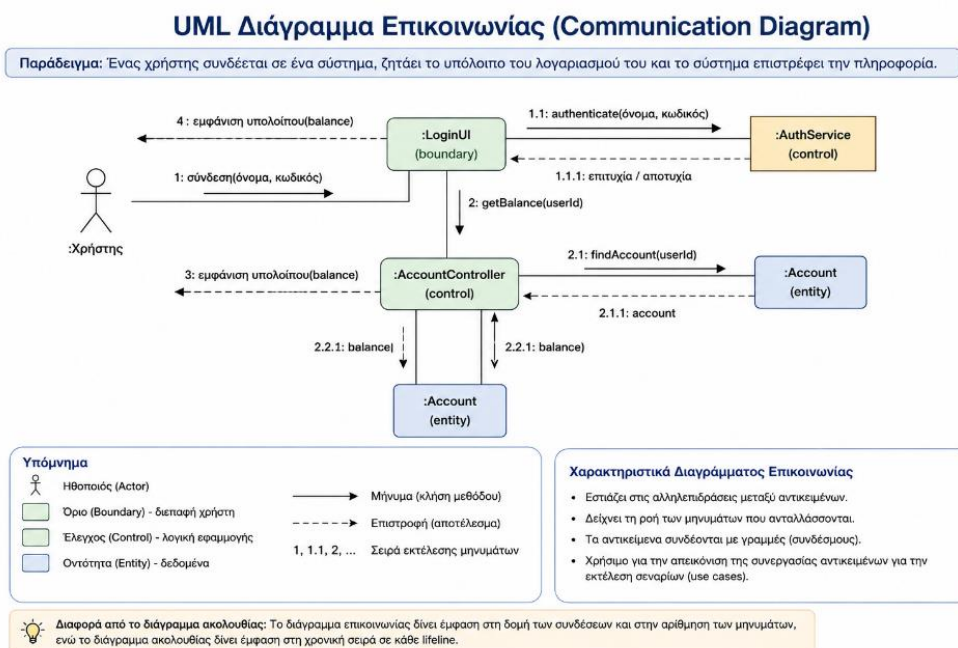
Τα διαγράμματα καταστάσεων περιγράφουν τις καταστάσεις ενός αντικειμένου και τις μεταβάσεις μεταξύ αυτών. Είναι ιδιαίτερα χρήσιμα σε συστήματα πραγματικού χρόνου, ενσωματωμένα συστήματα και εφαρμογές όπου η συμπεριφορά εξαρτάται από γεγονότα ή συνθήκες.



Εικόνα 2-11 Παράδειγμα Διάγραμμα Καταστάσεων (State Machine Diagram)

Διάγραμμα επικοινωνίας (Communication Diagram)

Τα διαγράμματα επικοινωνίας εστιάζουν στη δομή των αλληλεπιδράσεων μεταξύ αντικειμένων. Σε αντίθεση με τα διαγράμματα ακολουθίας, δίνουν μεγαλύτερη έμφαση στις σχέσεις μεταξύ των συμμετεχόντων παρά στη χρονική σειρά των μηνυμάτων.



Εικ 2-12 Παράδειγμα Διάγραμμα επικοινωνίας (Communication Diagram)

Συγκριτική ανάλυση των βασικών τύπων διαγραμμάτων

Η συνύπαρξη πολλών τύπων διαγραμμάτων αποτελεί ένα από τα σημαντικότερα πλεονεκτήματα της UML, καθώς επιτρέπει την εξέταση ενός συστήματος από διαφορετικές οπτικές γωνίες. Ένα διάγραμμα κλάσεων περιγράφει τη δομή, ένα διάγραμμα ακολουθίας τη συμπεριφορά, ενώ ένα διάγραμμα περιπτώσεων χρήσης τις λειτουργικές απαιτήσεις.

Η πολυδιάστατη αυτή προσέγγιση αυξάνει σημαντικά την εκφραστικότητα της μοντελοποίησης. Ωστόσο, δημιουργεί και νέες προκλήσεις, καθώς τα διαφορετικά διαγράμματα πρέπει να παραμένουν συνεπή μεταξύ τους. Ένα στοιχείο που εμφανίζεται σε διαφορετικά διαγράμματα οφείλει να διατηρεί την ίδια σημασία και να εντάσσεται στο ίδιο εννοιολογικό πλαίσιο (Brambilla et al., 2017).

Η απαίτηση αυτή εισάγει το ζήτημα της σημασιολογικής συνέπειας των μοντέλων. Δεν αρκεί τα διαγράμματα να είναι συντακτικά ορθά· πρέπει επίσης να αποδίδουν συμβατές ερμηνείες της πραγματικότητας που μοντελοποιούν. Το ζήτημα αυτό αποτελεί μία από τις σημαντικότερες προκλήσεις της σύγχρονης μοντελοποίησης λογισμικού και εξετάζεται στην επόμενη ενότητα.

2.4 Προκλήσεις στην ερμηνεία UML μοντέλων και η ανάγκη σημασιολογικής θεμελίωσης

Η ευρεία αποδοχή της UML στη βιομηχανία και στην ακαδημαϊκή κοινότητα δεν συνεπάγεται ότι η κατανόηση των μοντέλων της είναι πάντοτε εύκολη ή αδιαμφισβήτητη. Παρά την ύπαρξη τυποποιημένου συμβολισμού, η ερμηνεία των UML μοντέλων εξακολουθεί να επηρεάζεται από το γνωστικό υπόβαθρο των χρηστών, το επίπεδο λεπτομέρειας των διαγραμμάτων και το πλαίσιο μέσα στο οποίο αυτά χρησιμοποιούνται (France & Rumpe, 2007).

Η πρόκληση αυτή αναδεικνύει μια θεμελιώδη διάκριση μεταξύ σύνταξης και σημασιολογίας. Η σύνταξη καθορίζει τον τρόπο με τον οποίο αναπαρίστανται τα στοιχεία ενός μοντέλου, ενώ η σημασιολογία αφορά το νόημα που αποδίδεται σε αυτά. Ένα UML διάγραμμα μπορεί να είναι πλήρως συντακτικά ορθό, χωρίς ωστόσο να διασφαλίζεται ότι όλοι οι χρήστες θα το ερμηνεύσουν με τον ίδιο τρόπο (Harel & Rumpe, 2004).

Στο πλαίσιο αυτό, οι δυσκολίες κατανόησης των UML μοντέλων δεν αποτελούν απλώς τεχνικά προβλήματα σχεδίασης, αλλά συνδέονται με βαθύτερα ζητήματα αναπαράστασης γνώσης και σημασιολογικής συνέπειας. Η κατανόηση των προκλήσεων αυτών είναι απαραίτητη για την αποτελεσματική αξιοποίηση της UML σε σύνθετα πληροφοριακά συστήματα.

Η Καθιέρωση της UML ως Πρότυπο Μοντελοποίησης

Η καθιέρωση της UML ως διεθνούς προτύπου αποτέλεσε ένα από τα σημαντικότερα γεγονότα στην εξέλιξη της Μηχανικής Λογισμικού. Μέχρι τα μέσα της δεκαετίας του 1990, η ανάπτυξη αντικειμενοστραφών συστημάτων βασιζόταν σε διαφορετικές μεθοδολογίες και συμβολισμούς, γεγονός που δυσχέραινε την επικοινωνία μεταξύ αναλυτών, σχεδιαστών και προγραμματιστών. Η ενοποίηση των προσεγγίσεων των Booch, Rumbaugh και Jacobson οδήγησε στη δημιουργία μιας κοινής γλώσσας μοντελοποίησης, η οποία υιοθετήθηκε επίσημα από τον Object Management Group (OMG) και εξελίχθηκε σταδιακά στο κυρίαρχο πρότυπο της βιομηχανίας λογισμικού (OMG, 2023).

Σύμφωνα με τον Sommerville (2016), η UML δεν πρέπει να αντιμετωπίζεται ως μία πλήρης μεθοδολογία ανάπτυξης λογισμικού αλλά ως μία γλώσσα μοντελοποίησης που υποστηρίζει την περιγραφή, ανάλυση και σχεδίαση πολύπλοκων συστημάτων. Ο συγγραφέας επισημαίνει ότι η UML παρέχει διαφορετικές οπτικές αναπαράστασης ενός συστήματος μέσω ποικίλων διαγραμμάτων, επιτρέποντας στους μηχανικούς λογισμικού να προσεγγίζουν τόσο τη δομή όσο και τη συμπεριφορά ενός συστήματος.

Αντίστοιχα, ο Fowler (2018) υποστηρίζει ότι η αξία της UML δεν έγκειται αποκλειστικά στο πλήθος των διαθέσιμων διαγραμμάτων αλλά κυρίως στη δυνατότητα αποτελεσματικής επικοινωνίας μεταξύ των εμπλεκόμενων μερών ενός έργου ανάπτυξης λογισμικού. Ο Fowler θεωρεί ότι η UML λειτουργεί ως ένα κοινό λεξιλόγιο μοντελοποίησης, το οποίο επιτρέπει την απεικόνιση σύνθετων εννοιών με τρόπο κατανοητό και επαναχρησιμοποιήσιμο. Παράλληλα, επισημαίνει ότι η υπερβολική χρήση συμβολισμών και λεπτομερειών μπορεί να μειώσει την αναγνωσιμότητα των μοντέλων και να περιορίσει την πρακτική τους χρησιμότητα.

Η σύγχρονη χρήση της UML έχει επεκταθεί πέρα από τα παραδοσιακά πληροφοριακά συστήματα και εφαρμόζεται σε πεδία όπως τα κυβερνοφυσικά συστήματα, τα εκπαιδευτικά περιβάλλοντα, η μηχανική απαιτήσεων και το Model-Based Systems Engineering (MBSE). Η εξέλιξη αυτή αποδεικνύει ότι η UML εξακολουθεί να αποτελεί βασικό εργαλείο ανάλυσης και σχεδίασης, παρά την εμφάνιση νεότερων προσεγγίσεων ανάπτυξης λογισμικού.

Κριτική Αποτίμηση της Ιστορικής Εξέλιξης

Παρότι η UML καθιερώθηκε ως το επικρατέστερο πρότυπο μοντελοποίησης λογισμικού, η ιστορική της εξέλιξη συνοδεύτηκε από μία σταδιακή αύξηση της πολυπλοκότητάς της. Οι νεότερες εκδόσεις εισήγαγαν περισσότερους τύπους διαγραμμάτων, μηχανισμούς επέκτασης και εξειδικευμένες δυνατότητες, γεγονός που ενίσχυσε την εκφραστική ισχύ της γλώσσας αλλά παράλληλα αύξησε το γνωστικό φορτίο των χρηστών. Ο Sommerville (2016) αναγνωρίζει ότι στην πράξη μόνο ένα υποσύνολο των διαγραμμάτων χρησιμοποιείται συστηματικά από τους μηχανικούς λογισμικού, ενώ ο Fowler (2018) υποστηρίζει ότι η επιτυχία της UML εξαρτάται περισσότερο από την αποτελεσματική επικοινωνία που επιτυγχάνει ένα μοντέλο παρά από την πλήρη αξιοποίηση όλων των διαθέσιμων συμβολισμών.

Η παρατήρηση αυτή αποκτά ιδιαίτερη σημασία για την παρούσα εργασία, καθώς αναδεικνύει ότι η αξία ενός UML μοντέλου δεν εξαρτάται μόνο από τη συντακτική του ορθότητα αλλά και από την ικανότητά του να αποδίδει με σαφήνεια το νόημα των εννοιών και των διαδικασιών που αναπαριστά. Η διάσταση αυτή συνδέεται άμεσα με τη σημασιολογία των μοντέλων και προετοιμάζει το έδαφος για την ανάλυση που ακολουθεί στο επόμενο κεφάλαιο.

3. Η Σημασιολογία στη μοντελοποίηση λογισμικού

3.1 Ορισμός και βασικές έννοιες σημασιολογίας

Η έννοια της **σημασιολογίας (semantics)** αποτελεί θεμελιώδες στοιχείο πολλών επιστημονικών πεδίων, όπως η γλωσσολογία, η φιλοσοφία, η λογική και η επιστήμη υπολογιστών. Στο πλαίσιο της μοντελοποίησης και της ανάπτυξης λογισμικού, η σημασιολογία αναφέρεται στον τρόπο με τον οποίο τα μοντέλα — είτε γραφικά είτε τυπικά — αποκτούν νόημα και ερμηνεύονται με μη αμφίσημο τρόπο. Σύμφωνα με τον John Sowa, η σημασιολογία αφορά «τη σχέση μεταξύ των συμβολικών αναπαραστάσεων και της πραγματικότητας την οποία περιγράφουν» (Sowa, 2000). Η ανάγκη για αυστηρή και συνεκτική σημασιολογία καθίσταται ιδιαίτερα σημαντική σε περιβάλλοντα όπου τα μοντέλα αποτελούν βάση για ανάλυση, επικοινωνία ή αυτοματοποίηση διαδικασιών (Harel & Rumpe, 2004). Η έλλειψη καλά ορισμένης σημασιολογίας οδηγεί σε παρερμηνείες, ασυνέπειες και χαμηλή ποιότητα στην ανάπτυξη συστημάτων.

Ορισμός Σημασιολογίας

Η σημασιολογία αποτελεί τον κλάδο που μελετά το νόημα, δηλαδή τη σχέση ανάμεσα στα σύμβολα και στις έννοιες ή στα αντικείμενα που αυτά αναπαριστούν. Στο πεδίο των γλωσσών μοντελοποίησης — όπως η UML — ο όρος αναφέρεται στον τρόπο ερμηνείας των συμβολισμών, κανόνων και δομών με τρόπο που παράγει σαφές νόημα (Object Management Group, 2017).

Η σημασιολογία συνδέεται άμεσα με:

- την ερμηνεία (interpretation),
- τη λογική συνέπεια,
- τη συμπεριφορά που προκύπτει από τις μοντελοποιημένες δομές,
- τους περιορισμούς που διέπουν ένα σύστημα.

Σύμφωνα με τον Broy, η σημασιολογία είναι εκείνο το στρώμα που γεφυρώνει τη συντακτική μορφή μιας γλώσσας με την πραγματική λειτουργία των συστημάτων που περιγράφει (Broy & Stølen, 2001).

Διάκριση Συντακτικού και Σημασιολογίας

Για να γίνει κατανοητή η σημασία της σημασιολογίας, είναι αναγκαίο να διακριθεί από το συντακτικό (syntax). Το συντακτικό περιγράφει τη μορφή ενός μοντέλου: τα σύμβολα, τους κανόνες σύνδεσης και τη δομή. Αντιθέτως, η σημασιολογία περιγράφει το νόημα που μεταφέρει η μορφή αυτή. Ο David Harel επισημαίνει ότι τα μοντέλα γίνονται πραγματικά χρήσιμα μόνο όταν διαθέτουν «σημασιολογικό περιεχόμενο που επιτρέπει ουσιαστική ερμηνεία και αξιολόγηση» (Harel & Rumpfe, 2004). Η διάκριση αυτή συνιστά θεμέλιο για οποιαδήποτε προσπάθεια τυπικής ανάλυσης ή αυτοματοποίησης της μοντελοποίησης.

3.2 Είδη Σημασιολογίας

Στο πλαίσιο των γλωσσών μοντελοποίησης και προγραμματισμού, η σημασιολογία μπορεί να πάρει διαφορετικές μορφές:

- **Περιγραφική (denotational) σημασιολογία**

Η περιγραφική σημασιολογία αποδίδει στα σύμβολα μαθηματικά αντικείμενα τα οποία λειτουργούν ως «ερμηνευτές» του νοήματος. Στόχος είναι η δημιουργία ενός τυπικού πλαισίου που επιτρέπει την ακριβή ανάλυση της συμπεριφοράς (Stølen, 2001).

- **Επιχειρησιακή (operational) σημασιολογία**

Στην επιχειρησιακή σημασιολογία, το νόημα ορίζεται μέσω της περιγραφής των βημάτων ή των ενεργειών που εκτελεί ένα σύστημα. Είναι ιδιαίτερα χρήσιμη στην τυπική επαλήθευση και την ανάπτυξη εκτελέσιμων μοντέλων.

- **Αξιοματική (axiomatic) σημασιολογία**

Η αξιωματική σημασιολογία περιγράφει τις ιδιότητες που πρέπει να ισχύουν πριν και μετά την εκτέλεση ενός μοντέλου ή μιας λειτουργίας. Χρησιμοποιείται συχνά για την επαλήθευση ορθότητας.

Σημασιολογία στις Γλώσσες Μοντελοποίησης

Η σημασιολογία παίζει κρίσιμο ρόλο στον τρόπο με τον οποίο ερμηνεύονται τα στοιχεία των μοντέλων.

Στην UML, για παράδειγμα, υπάρχουν τρία επίπεδα σημασιολογίας (OMG, 2017):

1. **Τυπική σημασιολογία** (formal semantics): βασισμένη σε λογικά ή μαθηματικά μοντέλα.
2. **Ημι-τυπική σημασιολογία** (semi-formal): υποστηριζόμενη από διαγραμματικούς κανόνες και constraints.
3. **Ανεπίσημη σημασιολογία** (informal): περιγραφές σε φυσική γλώσσα.

Η ύπαρξη πολλαπλών στρωμάτων σημασιολογίας καθιστά τη UML ευέλικτη αλλά ταυτόχρονα επιτρέπει διαφορετικές ερμηνείες — γεγονός που αναδεικνύει τη σημασία της σημασιολογικής ακρίβειας κατά τη χρήση της (France & Rumpe, 2007).

Αντιστοίχιση επιπέδων σημασιολογίας με τύπους UML διαγραμμάτων

Η παραπάνω διάκριση μεταξύ τυπικής, ημι-τυπικής και ανεπίσημης σημασιολογίας μπορεί να γίνει περισσότερο συγκεκριμένη όταν συσχετιστεί με τους διαφορετικούς τύπους UML διαγραμμάτων. Η UML δεν αξιοποιεί το ίδιο επίπεδο σημασιολογικής αυστηρότητας σε όλα τα διαγράμματά της. Ορισμένα διαγράμματα, όπως τα Use Case Diagrams, βασίζονται σε μεγάλο βαθμό σε φυσική γλώσσα και περιγραφικά σενάρια, ενώ άλλα, όπως τα State Machine Diagrams ή τα Activity Diagrams, μπορούν να συνδεθούν ευκολότερα με πιο αυστηρά επιχειρησιακά ή τυπικά μοντέλα εκτέλεσης (Harel & Rumpe, 2004· OMG, 2017).

Ο Πίνακας 3-1 παρουσιάζει μία ενδεικτική αντιστοίχιση μεταξύ τύπων UML διαγραμμάτων και επιπέδων σημασιολογίας. Η αντιστοίχιση αυτή δεν είναι απόλυτη, καθώς η ίδια κατηγορία διαγράμματος μπορεί να χρησιμοποιηθεί με διαφορετικό βαθμό αυστηρότητας ανάλογα με το πλαίσιο χρήσης, την τεκμηρίωση και τα εργαλεία που τη συνοδεύουν.

Πίνακας 3-1. Αντιστοίχιση τύπων UML διαγραμμάτων με επίπεδα και τύπους σημασιολογίας

Τύπος UML διαγράμματος	Κύριος τύπος σημασιολογίας	Επίπεδο τυπικότητας	Σημασιολογική ερμηνεία / πρακτική συνέπεια
Use Case Diagram	Ανεπίσημη και ημι-τυπική σημασιολογία στόχων και ρόλων	Χαμηλή έως μέτρια	Περιγράφει τι αναμένουν οι actors από το σύστημα. Απαιτεί συνοδευτική λεκτική τεκμηρίωση, επειδή οι ελλείψεις ή οι ασάφειες στα σενάρια οδηγούν εύκολα σε διαφορετικές ερμηνείες.
Activity Diagram	Επιχειρησιακή / συμπεριφορική σημασιολογία ροής	Μέτρια έως υψηλή	Αποτυπώνει ροές ελέγχου, αποφάσεις, παράλληλες διεργασίες και κατανομή ευθυνών. Μπορεί να συνδεθεί με Petri Nets ή fUML για πιο αυστηρή ανάλυση.
Sequence Diagram	Επιχειρησιακή σημασιολογία αλληλεπιδράσεων	Μέτρια	Περιγράφει τη χρονική σειρά ανταλλαγής μηνυμάτων. Είναι χρήσιμο για σενάρια εκτέλεσης, αλλά μπορεί να γίνει ασαφές όταν δεν δηλώνονται ρητά συνθήκες, εναλλακτικές ροές και χρονικοί περιορισμοί.
State Machine Diagram	Τυπική ή ημι-τυπική σημασιολογία καταστάσεων	Υψηλή	Αναπαριστά καταστάσεις, γεγονότα και μεταβάσεις. Είναι από τα UML διαγράμματα που προσφέρονται περισσότερο για τυπική επαλήθευση μέσω automata ή statecharts.
Class Diagram	Δομική / περιγραφική σημασιολογία	Μέτρια	Περιγράφει κλάσεις, ιδιότητες, συσχετίσεις και πολλαπλότητες. Η σημασιολογική ακρίβεια ενισχύεται σημαντικά όταν χρησιμοποιείται OCL για invariants και περιορισμούς.

Τύπος UML διαγράμματος	Κύριος τύπος σημασιολογίας	Επίπεδο τυπικότητας	Σημασιολογική ερμηνεία / πρακτική συνέπεια
Component / Deployment Diagram	Αρχιτεκτονική και κατανεμημένη σημασιολογία	Χαμηλή έως μέτρια	Αποτυπώνει την οργάνωση υποσυστημάτων, κόμβων και εξαρτήσεων. Είναι χρήσιμο για επικοινωνία αρχιτεκτονικής, αλλά συνήθως δεν επαρκεί για τυπική ανάλυση συμπεριφοράς.
OCL Constraints	Αξιοματική σημασιολογία περιορισμών	Υψηλή	Εκφράζει προϋποθέσεις, μετα-συνθήκες και αναλλοίωτες ιδιότητες. Συμπληρώνει τα γραφικά UML διαγράμματα και μειώνει τις ασάφειες.

Η συσχέτιση αυτή δείχνει ότι η UML λειτουργεί κυρίως ως ημι-τυπική γλώσσα: διαθέτει τυποποιημένο συντακτικό και ορισμένους αυστηρούς σημασιολογικούς κανόνες, αλλά μεγάλο μέρος της ερμηνείας εξαρτάται από τη συνοδευτική τεκμηρίωση και από το πλαίσιο εφαρμογής. Επομένως, η επιλογή διαγράμματος πρέπει να γίνεται με βάση το είδος της σημασιολογίας που απαιτείται: για απαιτήσεις και ρόλους προτιμώνται use cases, για διαδικασίες activity diagrams, ενώ για αυστηρή ανάλυση συμπεριφοράς απαιτούνται state machines, OCL ή άλλες τυπικές τεχνικές.

Συσχέτιση των Επιπέδων Σημασιολογίας με τα Διαγράμματα UML

Η διάκριση μεταξύ τυπικής (formal), ημι-τυπικής (semi-formal) και ανεπίσημης (informal) σημασιολογίας αποκτά ιδιαίτερη σημασία όταν εξετάζεται σε σχέση με τα διαφορετικά είδη διαγραμμάτων της UML. Παρότι η προδιαγραφή της UML ορίζει και τα τρία επίπεδα σημασιολογίας, η εφαρμογή τους δεν είναι ομοιόμορφη σε όλα τα διαγράμματα (OMG, 2017).

Τα Use Case Diagrams βασίζονται κυρίως σε ανεπίσημη και ημι-τυπική σημασιολογία, καθώς η ερμηνεία των λειτουργικών απαιτήσεων εξαρτάται σε μεγάλο βαθμό από περιγραφές σε φυσική γλώσσα και από την κατανόηση του πεδίου εφαρμογής (Cockburn,

2001). Αντίθετα, τα Class Diagrams διαθέτουν ισχυρότερη δομική σημασιολογία, η οποία μπορεί να υποστηριχθεί από τυπικούς περιορισμούς μέσω της OCL (Warmer & Kleppe, 2003).

Στην περίπτωση των Activity Diagrams και State Machine Diagrams, η σημασιολογία μετακινείται προς περισσότερο τυπικές μορφές αναπαράστασης, καθώς οι έννοιες της ροής ελέγχου, των καταστάσεων και των μεταβάσεων μπορούν να συσχετιστούν με μαθηματικά μοντέλα όπως τα Petri Nets και operational semantics (Harel & Rumpe, 2004). Αντίστοιχα, τα Sequence Diagrams διαθέτουν ημι-τυπική έως τυπική συμπεριφορική σημασιολογία, ιδιαίτερα όταν χρησιμοποιούνται για ανάλυση αλληλεπιδράσεων και επαλήθευση σεναρίων εκτέλεσης (France & Rumpe, 2007).

Η παρατήρηση αυτή αναδεικνύει ότι η UML δεν αποτελεί μία ενιαία σημασιολογική οντότητα αλλά ένα σύνολο διαφορετικών αναπαραστάσεων, οι οποίες απαιτούν διαφορετικό βαθμό σημασιολογικής αυστηρότητας ανάλογα με τον σκοπό για τον οποίο χρησιμοποιούνται.

Ο Ρόλος της Σημασιολογίας στην Ανάπτυξη Λογισμικού

Η σημασιολογία επιτελεί κρίσιμες λειτουργίες στη μοντελοποίηση και την ανάπτυξη συστημάτων:

- **Εξασφάλιση συνέπειας**

Η καλά ορισμένη σημασιολογία εξασφαλίζει ότι τα μοντέλα διαφορετικών τύπων (π.χ. διαγράμματα δομής και συμπεριφοράς) παραμένουν συνεπή μεταξύ τους.

- **Υποστήριξη εργαλείων αυτόματης ανάλυσης**

Η ύπαρξη τυπικής σημασιολογίας καθιστά δυνατή την αυτοματοποίηση διαδικασιών όπως:

- έλεγχος συνέπειας,
- προσομοίωση συμπεριφοράς,
- παραγωγή κώδικα,
- τυπική επαλήθευση.

- **Βελτίωση της επικοινωνίας μεταξύ των μελών της ομάδας**

Τα μοντέλα με καλά καθορισμένη σημασιολογία μειώνουν τις παρερμηνείες και ενισχύουν την κατανόηση μεταξύ αναλυτών, σχεδιαστών, προγραμματιστών και τελικών χρηστών (Fowler, 2003).

Συμπεράσματα

Η σημασιολογία αποτελεί θεμελιώδη συνιστώσα της μοντελοποίησης και της επιστήμης υπολογιστών γενικότερα. Η ακριβής κατανόηση και εφαρμογή των σημασιολογικών αρχών εξασφαλίζουν μοντέλα υψηλής ποιότητας και αποτελεσματική επικοινωνία μεταξύ εμπλεκομένων. Επιπλέον, συμβάλλουν στην αυτοματοποίηση, την επαλήθευση και τη βελτίωση της συνολικής διαδικασίας ανάπτυξης. Συνεπώς, η σημασιολογία δεν αποτελεί απλώς θεωρητικό υπόβαθρο, αλλά έναν πρακτικό και αναπόσπαστο πυλώνα της σύγχρονης μηχανικής λογισμικού.

3.3. Η σημασιολογία στη UML

Η Unified Modeling Language (UML) αποτελεί πρότυπο για την οπτική αναπαράσταση και τεκμηρίωση πολύπλοκων συστημάτων λογισμικού. Παρόλο που η σύνταξη (syntax) των διαγραμμάτων UML είναι τυποποιημένη και σαφώς ορισμένη, η σημασιολογία (semantics) της —δηλαδή το τι ακριβώς “σημαίνουν” τα στοιχεία και οι σχέσεις που αποτυπώνονται στα μοντέλα— παραμένει ένα από τα πιο απαιτητικά και συζητήσιμα ζητήματα στη σχεδιαστική πρακτική και στη θεωρία των μοντέλων. Η σημασιολογία αποτελεί κρίσιμο παράγοντα για τη σωστή κατανόηση, ανάλυση και αυτόματη επεξεργασία των UML μοντέλων (Booch et al., 2005· Siever, 2015).

3.3.1 Πώς ορίζεται η σημασιολογία στη UML

Σημασιολογία είναι το σύνολο των κανόνων που καθορίζουν τη συσχέτιση μεταξύ των συμβολικών στοιχείων ενός μοντέλου και του πραγματικού ή υποθετικού συστήματος που αυτά αναπαριστούν. Στο πλαίσιο της UML, αφορά στην ερμηνεία των αντικειμένων, των σχέσεων, των συμπεριφορικών διαγραμμάτων και των περιορισμών που περιγράφουν την εσωτερική λογική ενός συστήματος (Evans et al., 1999).

Η UML διαθέτει πλούσιο σημασιολογικό υπόβαθρο, βασισμένο σε τυπικές έννοιες όπως συμπεριφορική σημασιολογία, δομική σημασιολογία, και σημασιολογία περιορισμών. Ωστόσο, η ευελιξία και η εκφραστικότητα της γλώσσας συχνά επιτρέπουν πολλαπλές ερμηνείες, οδηγώντας σε φαινόμενα ασάφειας (Cengarle et al., 2014).

3.3.2 Τυπικές προσεγγίσεις ορισμού σημασιολογίας

Δομική (Structural) σημασιολογία

Η δομική σημασιολογία αφορά σε στοιχεία όπως κλάσεις, ιδιότητες, συσχετίσεις και πακέτα. Παρέχει τους κανόνες που καθορίζουν πώς τα δομικά στοιχεία ενός συστήματος αντιστοιχούν σε πραγματικές οντότητες και σχέσεις στον χώρο εφαρμογής (Evans et al., 1999). Για παράδειγμα, η πολλαπλότητα (multiplicity) μιας συσχέτισης δεν αποτελεί απλώς σημειογραφία, αλλά σημασιολογική δέσμευση που καθορίζει πόσα στιγμιότυπα μπορούν ή πρέπει να συμμετέχουν σε μια σχέση. Οι σημασιολογικές αυτές ιδιότητες καθίστανται ακόμη πιο σημαντικές όταν το μοντέλο χρησιμοποιείται για ανάλυση συνέπειας ή για αυτόματη παραγωγή κώδικα.

Οι προσεγγίσεις που βασίζονται σε μοντέλα για τον σχεδιασμό συστημάτων είναι πλέον ευρέως διαδεδομένες και επιτυχημένες. Αυτές οι προσεγγίσεις κάνουν εκτεταμένη χρήση της δομής του μοντέλου για να περιγράψουν συστήματα χρησιμοποιώντας αφαιρέσεις ανά τομέα, να καθορίσουν και να εφαρμόσουν μετασχηματισμούς μοντέλων και να αναλύσουν δομικές ιδιότητες των μοντέλων. Σύμφωνα με την έρευνα των Jackson and Sztipanovits, (2008), παρά τη γενική της σημασία, η δομική σημασιολογία των γλωσσών μοντελοποίησης δεν είναι καλά κατανοητή. Στην έρευνά τους ανέπτυξαν τα τυπικά θεμέλια για τη δομική σημασιολογία των γλωσσών μοντελοποίησης ανά τομέα (Data Science & Machine Learning-DSML), συμπεριλαμβανομένων των μηχανισμών με τους οποίους τα μεταμοντέλα καθορίζουν τη δομική σημασιολογία των DSML. Επιπλέον, έδειξαν πώς η τυποποίησή τους μπορεί να συμπληρώσει τα υπάρχοντα εργαλεία και πώς αποδίδει αλγόριθμους για την ανάλυση των DSML και των μετασχηματισμών μοντέλων.

Η Συμπεριφορική (Behavioral) σημασιολογία

Η συμπεριφορική σημασιολογία αποτελεί μία από τις πιο πολύπλοκες πτυχές της UML, καθώς καλύπτει την εκτέλεση διεργασιών, τη μετάβαση καταστάσεων, την αλληλεπίδραση αντικειμένων και τη μοντελοποίηση γεγονότων. Διαγράμματα όπως τα State Machines, Activity Diagrams και Sequence Diagrams διαθέτουν συντακτικούς κανόνες, ωστόσο η πραγματική σημασία τους εξαρτάται από την ακριβή εκτέλεση των ενεργειών, των μηνυμάτων και των χρονικών περιορισμών (Harel & Rump, 2004). Σημαντική συμβολή στη διαμόρφωση της σημασιολογίας αποτελεί η υιοθέτηση της τυπικής εκτελέσιμης σημασιολογίας (Executable UML), η οποία επιχειρεί να συνδέσει τα UML μοντέλα με ένα εκτελέσιμο υπόδειγμα λειτουργίας (Mellor & Balcer, 2002). Ενώ τα διαγράμματα δομής (structural) δείχνουν τι περιέχει το σύστημα, τα διαγράμματα συμπεριφοράς δείχνουν πώς λειτουργεί το σύστημα "εν κινήσει".

Σύμφωνα με τους Li Dan and Li Danning, (2010), η σημασία της Συμπεριφορικής Σημασιολογίας συνίσταται στα παρακάτω:

- Επικύρωση Απαιτήσεων: Βοηθά στην επιβεβαίωση ότι το σύστημα ανταποκρίνεται σωστά στα σενάρια χρήσης.
- Ανίχνευση Σφαλμάτων: Επιτρέπει τον εντοπισμό προβλημάτων στη λογική ροής πριν από την υλοποίηση (κώδικα).
- Παραγωγή Κώδικα: Τα μοντέλα συμπεριφοράς (ειδικά τα state machines) μπορούν να μετατραπούν σε εκτελέσιμο κώδικα.

Σημασιολογία περιορισμών και OCL

Η Object Constraint Language (OCL) αποτελεί το κύριο μέσο για την έκφραση τυπικών περιορισμών εντός UML μοντέλων. Η σημασιολογία της OCL είναι πλήρως τυποποιημένη και επιτρέπει την αυστηρή περιγραφή κανόνων που δεν μπορούν να αποδοθούν γραφικά (Warmer & Kleppe, 2003).

Οι περιορισμοί OCL εμπλουτίζουν το σημασιολογικό περιεχόμενο, εξασφαλίζοντας ότι τα μοντέλα είναι όχι μόνο δομικά ορθά, αλλά και λογικά συνεπή. Η ύπαρξη σαφούς

σημασιολογίας είναι κρίσιμη για την αυτόματη επαλήθευση μοντέλων και την ανίχνευση σφαλμάτων σε πρώιμα στάδια. Ενώ η UML παρέχει γραφικές αναπαραστάσεις, η OCL προσθέτει «σημασιολογία» (semantics) - δηλαδή ακριβές νόημα - σε αυτά τα διαγράμματα επιτρέποντας την περιγραφή συνθηκών που δεν μπορούν να εκφραστούν μόνο με γραφικά (Richters & Gogolla, 1998).

Η OCL χρησιμοποιείται για να διασφαλίσει την ακεραιότητα των δεδομένων και τη σωστή συμπεριφορά του συστήματος. Οι κυριότεροι τύποι περιορισμών που εκφράζονται σύμφωνα με τους (Steimann, Clariso & Goggolla, 2025) είναι:

- Αναλλοίωτες (Invariants): Συνθήκες που πρέπει πάντα να ισχύουν για όλα τα στιγμιότυπα (instances) μιας κλάσης. Αν μια αναλλοίωτη παραβιαστεί το μοντέλο θεωρείται ασταθές.
- Προ-συνθήκες (Pre-conditions): Συνθήκες που πρέπει να είναι αληθείς πριν από την εκτέλεση μιας λειτουργίας (operation/method).
- Μετα-συνθήκες (Post-conditions): Συνθήκες που πρέπει να είναι αληθείς μετά την ολοκλήρωση μιας λειτουργίας (περιγράφουν την αλλαγή κατάστασης).
- Κανόνες Παραγωγής (Derivation Rules): Ορισμός του πώς υπολογίζεται η τιμή ενός παραγόμενου (derived) χαρακτηριστικού.

Τυπικές (Formal) σημασιολογικές προσεγγίσεις

Παρά τις προσπάθειες τυποποίησης, η UML παραμένει εν μέρει ακριβής ως προς τη σημασιολογία της. Για τον λόγο αυτό έχουν αναπτυχθεί πληθώρα τυπικών προσεγγίσεων, όπως:

- σημασιολογία βασισμένη σε λογισμό καταστάσεων,
- σημασιολογία βασισμένη σε θεωρία κατηγοριών,
- σημασιολογία με χρήση Petri Nets,
- σημασιολογία βασισμένη σε operational semantics.

Οι προσεγγίσεις αυτές επιχειρούν να μετατρέψουν την UML σε μια πιο αυστηρή και λιγότερο αμφίσημη γλώσσα, ιδιαίτερα για εφαρμογές που απαιτούν υψηλό βαθμό αξιοπιστίας (Cengarle et al., 2014· Harel & Rumpe, 2004).

Η σημασιολογία βασισμένη στον λογισμό καταστάσεων (Situation Calculus) είναι ένα τυπικό λογικό πλαίσιο (formal logic framework) που χρησιμοποιείται στην Τεχνητή Νοημοσύνη (AI) για την αναπαράσταση και τον συλλογισμό σχετικά με δυναμικούς κόσμους όπου οι αλλαγές προκαλούνται από ενέργειες. Αναπτύχθηκε αρχικά από τον John McCarthy (1963) ως μια λογική πρώτης τάξης (με ορισμένα χαρακτηριστικά δεύτερης τάξης) για να μοντελοποιεί πώς οι πράξεις αλλάζουν την κατάσταση του κόσμου (Levesque, Plwousakis, Belle, & Bure, 2019)

“Η σημασιολογία βασισμένη στη θεωρία κατηγοριών (categorical semantics) είναι ένας κλάδος των μαθηματικών και της θεωρητικής πληροφορικής που μελετά τη σημασία (semantics) τυπικών γλωσσών λογικών συστημάτων και προγραμμάτων υπολογιστών χρησιμοποιώντας τις δομές της θεωρίας κατηγοριών. Αντί να εστιάζει μόνο στα αντικείμενα (όπως η κλασική θεωρία συνόλων), δίνει έμφαση στις σχέσεις και τους μορφισμούς μεταξύ τους (Academia.edu)”.

Η σημασιολογία των Δικτύων Petri (Petri Nets Semantics) αναφέρεται στον αυστηρό μαθηματικό ορισμό της συμπεριφοράς της λειτουργίας και της ερμηνείας ενός δικτύου Petri. Καθορίζει πώς το δίκτυο αλλάζει κατάσταση (marking) μέσω της πυροδότησης μεταβάσεων (transition firing) μοντελοποιώντας τον συγχρονισμό, τον ανταγωνισμό και την παραλληλία σε κατανεμημένα συστήματα (Juhas, Lehocki & Lorenz, 2007).

Η λειτουργική σημασιολογία (operational semantics) είναι ένα πλαίσιο τυπικής σημασιολογίας γλωσσών προγραμματισμού το οποίο ορίζει το νόημα ενός προγράμματος περιγράφοντας πώς εκτελείται σε μια αφηρημένη μηχανή. Αντί να αντιστοιχίζει συντακτικά στοιχεία σε μαθηματικές συναρτήσεις (όπως η δηλωτική/denotational σημασιολογία), η λειτουργική σημασιολογία καθορίζει κανόνες μετάβασης που δείχνουν πώς αλλάζει η κατάσταση (state) του προγράμματος κατά την εκτέλεση (Marriott, Stuckey & Wallace, 2006).

Συγκριτική αξιολόγηση τυπικών σημασιολογικών προσεγγίσεων

Οι τυπικές προσεγγίσεις που αναφέρθηκαν δεν είναι ισοδύναμες ως προς την πρακτική τους χρησιμότητα. Κάθε προσέγγιση προσφέρει διαφορετικό βαθμό αυστηρότητας, διαφορετικές δυνατότητες ανάλυσης και διαφορετικό κόστος υιοθέτησης. Για τον λόγο αυτό, η επιλογή τους πρέπει να συνδέεται με τον τύπο του UML διαγράμματος, τον στόχο της ανάλυσης και το επίπεδο εμπειρίας των χρηστών.

Πίνακας 3-2. Συγκριτική αξιολόγηση τυπικών προσεγγίσεων σημασιολογίας UML

Τυπική προσέγγιση	Καταλληλότερη χρήση σε UML	Πρακτικότητα	Κύρια εμπόδια / περιορισμοί
Petri Nets	Activity Diagrams, workflow, παράλληλες διεργασίες, συγχρονισμός	Υψηλή για ανάλυση ροών και concurrency. Υποστηρίζονται από ώριμα εργαλεία προσομοίωσης και model checking.	Η αντιστοίχιση από UML σε Petri Nets δεν είναι πάντα άμεση. Μπορεί να χαθεί πληροφορία σχετική με actors, αντικείμενα ή επιχειρησιακό πλαίσιο.
Finite State Machines / Statecharts	State Machine Diagrams, reactive systems, embedded behavior	Πολύ υψηλή για συστήματα με σαφείς καταστάσεις και γεγονότα. Κατάλληλη για verification και testing.	Δυσκολεύεται σε συστήματα με έντονη παραλληλία ή μεγάλο αριθμό καταστάσεων, λόγω state explosion.
Operational Semantics / fUML	Activity Diagrams και εκτελέσιμα υποσύνολα UML	Υψηλή όταν απαιτείται εκτέλεση ή προσομοίωση μοντέλων. Συνδέει το UML μοντέλο με συγκεκριμένο τρόπο εκτέλεσης.	Περιορίζεται σε συγκεκριμένα υποσύνολα της UML και απαιτεί εξειδικευμένα εργαλεία. Δεν καλύπτει πάντα πλήρως όλα τα στοιχεία ενός διαγράμματος.

Τυπική προσέγγιση	Καταλληλότερη χρήση σε UML	Πρακτικότητα	Κύρια εμπόδια / περιορισμοί
OCL / Axiomatic Semantics	Class Diagrams, constraints, invariants, pre/post conditions	Πολύ πρακτική για έλεγχο συνέπειας και έκφραση περιορισμών που δεν φαίνονται γραφικά.	Απαιτεί εξοικείωση με τυπική λογική. Δεν περιγράφει από μόνη της δυναμική συμπεριφορά ή αλληλουχίες ενεργειών.
Situation Calculus	Συστήματα ενεργειών, reasoning για αλλαγές κατάστασης	Χρήσιμο σε ερευνητικά ή AI-oriented περιβάλλοντα όπου απαιτείται λογικός συλλογισμός για ενέργειες.	Χαμηλότερη πρακτική υιοθέτηση στη συνηθισμένη UML μοντελοποίηση. Η μαθηματική διατύπωση αυξάνει την πολυπλοκότητα.
Category Theory	Θεμελίωση μεταμοντέλων, μετασχηματισμοί μοντέλων, θεωρητική ανάλυση	Ισχυρή ως θεωρητικό θεμέλιο για MDE και μετασχηματισμούς μοντέλων.	Περιορισμένη άμεση χρήση από αναλυτές και μηχανικούς. Υψηλό επίπεδο αφαίρεσης και δυσκολία εκπαιδευτικής εφαρμογής.

Από τον Πίνακα 3-2 προκύπτει ότι οι Petri Nets και οι State Machines είναι πιο πρακτικές για συμπεριφορικά UML διαγράμματα, επειδή διαθέτουν άμεση αντιστοίχιση με έννοιες όπως ροή, κατάσταση, μετάβαση και συγχρονισμός. Αντίθετα, η θεωρία κατηγοριών προσφέρει ισχυρό θεωρητικό υπόβαθρο, αλλά έχει μικρότερη άμεση χρησιμότητα για τον μέσο αναλυτή. Η OCL καταλαμβάνει μία ενδιάμεση αλλά ιδιαίτερα σημαντική θέση, καθώς δεν αντικαθιστά τα διαγράμματα UML, αλλά τα συμπληρώνει με αυστηρούς περιορισμούς. Συνεπώς, οι τυπικές προσεγγίσεις πρέπει να αξιολογούνται όχι μόνο με βάση τη μαθηματική τους αυστηρότητα, αλλά και με βάση την πρακτικότητα, τη μαθησιακή καμπύλη και τη διαθεσιμότητα εργαλείων.

Η σύγκριση αυτή καλύπτει ένα σημαντικό κενό της απλής παράθεσης τυπικών τεχνικών: δείχνει ότι η τυπικότητα δεν είναι αυτοσκοπός. Για εκπαιδευτικά ή αναλυτικά UML

μοντέλα, όπως τα use case και activity diagrams της παρούσας εργασίας, συχνά επαρκεί η ημι-τυπική σημασιολογία με σαφή τεκμηρίωση. Όταν όμως το μοντέλο πρόκειται να χρησιμοποιηθεί για προσομοίωση, αυτόματη επαλήθευση ή παραγωγή κώδικα, απαιτείται ισχυρότερη τυπική βάση.

Συγκριτική Αξιολόγηση Τυπικών Σημασιολογικών Προσεγγίσεων

Παρά την ύπαρξη πολλών τυπικών προσεγγίσεων για τον ορισμό της σημασιολογίας των UML μοντέλων, καμία δεν έχει επικρατήσει ως καθολικό πρότυπο. Κάθε προσέγγιση παρουσιάζει διαφορετικά πλεονεκτήματα και περιορισμούς, γεγονός που επηρεάζει την πρακτική εφαρμογή της στη βιομηχανία λογισμικού.

Η χρήση των Petri Nets θεωρείται ιδιαίτερα αποτελεσματική για τη μοντελοποίηση παράλληλων και κατανεμημένων συστημάτων, καθώς παρέχει σαφή αναπαράσταση συγχρονισμού, αλληλεξάρτησης και ανταγωνισμού μεταξύ διεργασιών (Juhás, Lehocki & Lorenz, 2007). Ωστόσο, η μαθηματική τους πολυπλοκότητα δυσχεραίνει τη χρήση τους από μη ειδικούς χρήστες.

Αντίστοιχα, η operational semantics προσφέρει υψηλό βαθμό κατανόησης της συμπεριφοράς ενός συστήματος μέσω κανόνων εκτέλεσης και μεταβάσεων καταστάσεων. Για τον λόγο αυτό χρησιμοποιείται συχνά σε εκτελέσιμα μοντέλα UML και σε περιβάλλοντα προσομοίωσης (Marriott, Stuckey & Wallace, 2006). Παρά τα πλεονεκτήματα αυτά, η δημιουργία πλήρως τυπικών κανόνων εκτέλεσης για μεγάλα μοντέλα παραμένει ιδιαίτερα απαιτητική.

Η σημασιολογία που βασίζεται στον λογισμό καταστάσεων (Situation Calculus) παρέχει ισχυρές δυνατότητες συλλογισμού σχετικά με δυναμικά συστήματα και μεταβολές κατάστασης, γεγονός που την καθιστά ιδιαίτερα χρήσιμη σε εφαρμογές τεχνητής νοημοσύνης (Levesque et al., 2019). Ωστόσο, η αυξημένη θεωρητική πολυπλοκότητα περιορίζει τη διάδοσή της σε πρακτικά έργα ανάπτυξης λογισμικού.

Τέλος, η θεωρία κατηγοριών προσφέρει υψηλό βαθμό μαθηματικής αυστηρότητας και υποστηρίζει την τυπική περιγραφή μετασχηματισμών μοντέλων και μεταμοντέλων. Παρά τα σημαντικά θεωρητικά πλεονεκτήματα, η δυσκολία κατανόησης των αφηρημένων

εννοιών της θεωρίας κατηγοριών αποτελεί σημαντικό εμπόδιο για την ευρεία βιομηχανική αξιοποίησή της (Cengarle et al., 2014).

Συνεπώς, η επιλογή μιας τυπικής σημασιολογικής προσέγγισης δεν εξαρτάται μόνο από τη θεωρητική της πληρότητα αλλά και από παράγοντες όπως η πολυπλοκότητα, η διαθεσιμότητα εργαλείων, η καμπύλη εκμάθησης και οι ανάγκες του συγκεκριμένου έργου. Η διαπίστωση αυτή εξηγεί γιατί η UML εξακολουθεί να συνδυάζει τυπικά, ημι-τυπικά και ανεπίσημα σημασιολογικά στοιχεία αντί να βασίζεται αποκλειστικά σε μία ενιαία τυπική σημασιολογία (OMG, 2017; France & Rumpe, 2007).

Προβλήματα και προκλήσεις στη σημασιολογία της UML

Παρά τις εκτεταμένες έρευνες, αρκετές προκλήσεις παραμένουν ανοιχτές:

1. **Ασάφειες στη φυσική γλώσσα** των περιγραφών.
2. **Πολυσημία** συμβολισμών σε διαφορετικά περιβάλλοντα ανάπτυξης.
3. **Δυσκολία στον πλήρη τυπικό ορισμό** της συμπεριφορικής σημασιολογίας.
4. **Προβλήματα ευθυγράμμισης σημασιολογίας** μεταξύ διαφορετικών εργαλείων μοντελοποίησης.
5. **Περιορισμένη υποστήριξη τυπικής επαλήθευσης** στα περισσότερα εμπορικά εργαλεία.

Η αντιμετώπιση των παραπάνω αποτελεί κρίσιμο σημείο για την εξέλιξη της UML και τη βελτίωση της ποιότητας των μοντέλων (Siever, 2015).

Συμπεράσματα

Η σημασιολογία στην UML αποτελεί θεμέλιο για την αξιόπιστη και συνεπή μοντελοποίηση λογισμικού. Αν και έχουν επιτευχθεί σημαντικές πρόοδοι στην τυποποίηση και τυπικοποίηση των σημασιολογικών στοιχείων, οι ασάφειες που παραμένουν υπογραμμίζουν την ανάγκη για περαιτέρω έρευνα και εναρμόνιση μεταξύ θεωρητικών και πρακτικών προσεγγίσεων.

3.3.3 Semantics – Driven Modeling : αρχές και ρόλος στη διαδικασία ανάπτυξης

Η αυξανόμενη πολυπλοκότητα των συστημάτων λογισμικού έχει αναδείξει την ανάγκη για μοντελοποίηση που υπερβαίνει τη συντακτική περιγραφή και επιτρέπει την ακριβή και μη διφορούμενη ερμηνεία μοντέλων. Στο πλαίσιο αυτό, το **Semantics-Driven Modeling (SDM)** αποτελεί μια προσέγγιση η οποία τοποθετεί τη **σημασιολογία (semantics)** στο επίκεντρο της διαδικασίας μοντελοποίησης, ώστε τα μοντέλα να εκφράζουν με σαφήνεια τη συμπεριφορά, τους περιορισμούς και τη λογική του υπό ανάπτυξη συστήματος (Broy & Rumpe, 2023). Σε αντίθεση με τις παραδοσιακές προσεγγίσεις που επικεντρώνονται κυρίως σε διαγραμματικές αναπαραστάσεις, το SDM δίνει έμφαση στη σημασιολογική συνέπεια και τυπικότητα, καθιστώντας τα μοντέλα υπολογίσιμα και ελέγξιμα.

Θεμελιώδεις Αρχές του Semantics-Driven Modeling

Η προσέγγιση SDM θεμελιώνεται σε ένα σύνολο αρχών που επιτρέπουν την ακριβή και αξιόπιστη κατασκευή μοντέλων:

- **Σαφής και τυπική σημασιολογία**

Ένα από τα κεντρικά χαρακτηριστικά του SDM είναι η υιοθέτηση **τυπικής και μη αμφίσημης** σημασιολογίας. Η χρήση μαθηματικών ή λογικών τυπώσεων μειώνει την πιθανότητα παρερμηνειών, διευκολύνοντας την αναλυτική επαλήθευση και προσομοίωση μοντέλων (Clark et al., 2014). Αυτό είναι ιδιαίτερα κρίσιμο στη βιομηχανία, όπου η ανακριβής σημασιολογία μπορεί να οδηγήσει σε σημαντικά σφάλματα στον σχεδιασμό συστημάτων.

- **Προσανατολισμός στη συμπεριφορά**

Το SDM εστιάζει στη μοντελοποίηση της **δυναμικής συμπεριφοράς** του συστήματος και όχι μόνο στη δομή του. Έτσι, τα μοντέλα δεν αποτελούν απλή τεκμηρίωση, αλλά γίνονται εκτελέσιμα και επιτρέπουν την προσομοίωση και αξιολόγηση της λειτουργικότητας πριν ξεκινήσει η υλοποίηση (Harel & Rumpe, 2004).

- **Ενοποιημένη σημασιολογία σε πολλαπλές απόψεις**

Στα σύγχρονα μοντέλα, το σύστημα αποτυπώνεται μέσα από πολλαπλές απόψεις (views), όπως στατικά και δυναμικά διαγράμματα ή διαγράμματα απαιτήσεων. Το SDM επιδιώκει τη σημασιολογική **ενοποίηση αυτών των απόψεων**, ώστε να αποφεύγονται ασυνέχειες και αντιφάσεις μεταξύ τους (Bjørner, 2010).

- **Υποστήριξη αυτοματοποίησης και ανάλυσης**

Η σημασιολογικά εμπλουτισμένη μοντελοποίηση επιτρέπει την αυτόματη παραγωγή κώδικα, τη στατική ανάλυση και την επαλήθευση ιδιοτήτων του συστήματος. Έτσι, το SDM προωθεί την ανάπτυξη λογισμικού υψηλής αξιοπιστίας (France & Rumpe, 2007).

Ο Ρόλος του Semantics-Driven Modeling στη Διαδικασία Ανάπτυξης

- **Ενίσχυση της επικοινωνίας μεταξύ των εμπλεκομένων**

Με σαφή σημασιολογία, τα μοντέλα γίνονται κατανοητά από αναλυτές, προγραμματιστές, μελετητές και πελάτες. Η τυπική σημασιολογία λειτουργεί ως κοινή γλώσσα και περιορίζει τις διαφωνίες και τις ερμηνευτικές αποκλίσεις (Object Management Group, 2017).

- **Υποστήριξη πρώιμου εντοπισμού σφαλμάτων**

Η δυνατότητα εκτέλεσης, προσομοίωσης ή τυπικής επαλήθευσης σημασιολογικών μοντέλων καθιστά εφικτό τον εντοπισμό λαθών σε πρώιμα στάδια. Αυτό μειώνει το κόστος ανάπτυξης και βελτιώνει την ποιότητα του τελικού συστήματος (Whittle et al., 2011).

- **Βελτίωση της συνέπειας σε μεγάλης κλίμακας έργα**

Σε μεγάλα έργα, όπου πολλαπλές ομάδες εργάζονται σε διαφορετικές πτυχές του συστήματος, η σημασιολογικά καθοδηγούμενη μοντελοποίηση εξασφαλίζει ότι τα επιμέρους μοντέλα παραμένουν συνεπή και ευθυγραμμισμένα με τις συνολικές απαιτήσεις (Broy & Stølen, 2001).

- **Υποστήριξη μοντέλων ως πρώτης κατηγορίας τεχνημάτων**

Στο SDM, τα μοντέλα δεν είναι απλές στατικές αναπαραστάσεις, αλλά βασικά τεχνήματα της διαδικασίας ανάπτυξης. Χρησιμοποιούνται για παραγωγή κώδικα, έλεγχο, τεκμηρίωση

και επικαιροποίηση του συστήματος, καθιστώντας την ανάπτυξη περισσότερο μοντελοκεντρική και λιγότερο κωδικοκεντρική (Kent, 2002).

Προκλήσεις του Semantics-Driven Modeling

Παρά τα πλεονεκτήματα, η υιοθέτηση του SDM δεν είναι χωρίς δυσκολίες. Μία από τις πιο σημαντικές προκλήσεις σχετίζεται με την πολυπλοκότητα της τυπικής σημασιολογίας, που συχνά απαιτεί ειδικευμένες γνώσεις από τους μηχανικούς (France & Rumpe, 2007 · Brooy & Rumpe, 2023). Επίσης, η ενσωμάτωση εργαλείων που υποστηρίζουν την πλήρη σημασιολογική ανάλυση μπορεί να απαιτεί σημαντική επένδυση χρόνου και πόρων.

Συμπεράσματα

Το Semantics-Driven Modeling αποτελεί μια στρατηγική προσέγγιση για την ανάπτυξη λογισμικού υψηλής ποιότητας, εστιάζοντας στη σημασιολογική ακρίβεια των μοντέλων. Με την ενίσχυση της ανάλυσης, της επαλήθευσης και της αυτοματοποίησης, το SDM υποστηρίζει έναν μοντελοκεντρικό τρόπο ανάπτυξης που ανταποκρίνεται στις ανάγκες των σύγχρονων σύνθετων συστημάτων. Παρά τις προκλήσεις που σχετίζονται με την υιοθέτηση της τυπικής σημασιολογίας, τα οφέλη του αποτελούν ισχυρό κίνητρο για την ενσωμάτωσή του στη μηχανική λογισμικού.

3.3.4 Καλές Πρακτικές για Σημασιολογικά Σωστή Χρήση της UML

Η **Unified Modeling Language (UML)** αποτελεί το πιο διαδεδομένο πρότυπο για την οπτική αναπαράσταση συστημάτων λογισμικού. Παρότι η UML έχει θεσπιστεί από τον OMG ως μια ημι-τυπική γλώσσα, η ορθή σημασιολογική χρήση της αποτελεί πρόκληση, ιδιαίτερα σε έργα μεγάλης κλίμακας (Booch, Rumbaugh, & Jacobson, 2005). Η σημασιολογία των μοντέλων καθορίζει τον τρόπο με τον οποίο πρέπει να ερμηνεύονται οι έννοιες, οι σχέσεις και οι συμπεριφορές που εκφράζονται στα διαγράμματα. Η μη συνεπής ή ελλιπής σημασιολογική εφαρμογή μπορεί να οδηγήσει σε παρερμηνείες, σφάλματα σχεδιασμού και αυξημένο κόστος ανάπτυξης (Harel & Rumpe, 2004).

Παρακάτω, παρουσιάζονται οι βασικές καλές πρακτικές που ενισχύουν τη σημασιολογικά ορθή χρήση της UML σε όλο το φάσμα της μοντελοποίησης.

- **Χρήση των στοιχείων σύμφωνα με τον επίσημο ορισμό**

Η UML περιλαμβάνει λεπτομερείς προδιαγραφές για κάθε δομικό και συμπεριφορικό στοιχείο. Η ορθή σημασιολογική χρήση απαιτεί συμμόρφωση με τις επίσημες περιγραφές, όπως αναλύονται στις προδιαγραφές του OMG (OMG, 2017). Η αυθαίρετη αναπροσαρμογή των στοιχείων οδηγεί σε ασυνέπειες και δυσκολία κατανόησης.

- **Αποφυγή ασαφών ή υπερφορτωμένων συμβολισμών**

Η χρήση στοιχείων για σκοπούς διαφορετικούς από αυτούς για τους οποίους σχεδιάστηκαν αποτελεί συχνή πηγή σημασιολογικών σφαλμάτων. Για παράδειγμα, η χρήση των association ή aggregation για την έκφραση εξαρτήσεων ή ιδιοτήτων είναι συχνά παραπλανητική (France & Rumpe, 2007).

Διατήρηση Σημασιολογικής Συνέπειας μεταξύ διαγραμμάτων

Η ύπαρξη πολλαπλών διαγραμμάτων που περιγράφουν διαφορετικές όψεις του συστήματος αυξάνει τον κίνδυνο ασυνεπειών, ειδικά όταν τα διαγράμματα δημιουργούνται σε διαφορετικές χρονικές στιγμές ή από διαφορετικές ομάδες (Broy & Stølen, 2001).

- **Συνέπεια μεταξύ στατικών και δυναμικών διαγραμμάτων**

Η συμπεριφορά που περιγράφεται στα sequence ή state machine diagrams πρέπει να συμβαδίζει με τη δομή που παρουσιάζεται στα class diagrams. Ασυνέπειες μεταξύ τους μειώνουν τη σημασιολογική ακρίβεια των μοντέλων.

- **Μοντελοποίηση με ενιαίο λεξιλόγιο**

Η συνέπεια στην ονοματολογία των κλάσεων, ιδιοτήτων, ρόλων και λειτουργιών είναι κρίσιμη για την αποφυγή σημασιολογικής σύγχυσης (Fowler, 2003).

Χρήση περιορισμών και Τυπικής Σημασιολογίας

- **Ενσωμάτωση OCL για ενίσχυση της σημασιολογικής ακρίβειας**

Η Object Constraint Language (OCL) αποτελεί βασικό εργαλείο για τη διατύπωση ακριβών σημασιολογικών περιορισμών. Μέσω της OCL μπορούν να εκφραστούν προϋποθέσεις, ακεραιότητες και συμπεριφορικοί κανόνες που δεν αποτυπώνονται καθαρά σε διαγράμματα (Cabot & Teniente, 2006). Η χρήση OCL ενισχύει την ανεπίσημη σημασιολογία των διαγραμμάτων με τυπικά ορισμένες συνθήκες.

- **Συσχέτιση μοντέλων με εκτελέσιμη σημασιολογία**

Η υιοθέτηση μοντέλων με εκτελέσιμη σημασιολογία – όπως το fUML (Foundational UML) – επιτρέπει την προσομοίωση και τον έλεγχο συμπεριφορών πριν από την υλοποίηση (Mellor, Clark, & Futagami, 2003). Με τον τρόπο αυτό μειώνονται τα σημασιολογικά σφάλματα.

Καλές Πρακτικές στη Διαγραμματική Μοντελοποίηση

- **Διατήρηση απλότητας και αποφυγή υπερβολικής λεπτομέρειας**

Η υπερφόρτωση των διαγραμμάτων με λεπτομέρειες οδηγεί σε δυσκολία ερμηνείας. Η καλή πρακτική απαιτεί δημιουργία διαγραμμάτων εστιασμένων στον σκοπό τους, με σαφή διαχωρισμό επιπέδων αφαίρεσης (Larman, 2004).

- **Σωστή χρήση stereotypes και profiles**

Τα UML profiles και stereotypes επιτρέπουν την επέκταση της γλώσσας, αλλά η ακατάλληλη χρήση τους μπορεί να δημιουργήσει σημασιολογική σύγχυση (Selic, 2012). Η χρήση τους πρέπει να γίνεται μόνο όταν υπάρχει πραγματική ανάγκη εξειδίκευσης.

- **Επικέντρωση στις σχέσεις και όχι μόνο στα σχήματα**

Οι σημασιολογικές σχέσεις – όπως γενίκευση, υλοποίηση, συσχετισμός και εξάρτηση – πρέπει να χρησιμοποιούνται με ακρίβεια, καθώς καθορίζουν τον τρόπο με τον οποίο τα στοιχεία του συστήματος αλληλεπιδρούν (Fowler, 2003).

Υποστήριξη με Εργαλεία και Αυτοματοποιημένη Ανάλυση

- Έλεγχος συνέπειας και επικύρωση μοντέλων

Σύγχρονα CASE εργαλεία υποστηρίζουν την αυτόματη ανίχνευση συντακτικών και σημασιολογικών σφαλμάτων. Η συστηματική χρήση τους ενισχύει την ποιότητα και αξιοπιστία των μοντέλων (France & Rumpé, 2007).

- Σύνδεση μοντέλων με διαδικασίες MDD/MDA

Η σημασιολογικά ορθή μοντελοποίηση αποτελεί προϋπόθεση για μοντελοκεντρικές προσεγγίσεις όπως Model-Driven Engineering (MDE) και Model-Driven Architecture (MDA). Η καλή σημασιολογία διασφαλίζει την ακρίβεια της αυτόματης παραγωγής κώδικα (Kent, 2002).

Συμπεράσματα

Η σημασιολογικά σωστή χρήση της UML αποτελεί καθοριστικό παράγοντα για την επιτυχία των έργων ανάπτυξης λογισμικού. Οι καλές πρακτικές που παρουσιάστηκαν στο κεφάλαιο καλύπτουν την ανάγκη για συνέπεια, ακρίβεια, σωστή χρήση των συμβολισμών και αξιοποίηση των τυπικών μηχανισμών σημασιολογίας. Η υιοθέτηση αυτών των πρακτικών ενισχύει την κατανόηση των μοντέλων, μειώνει τα σφάλματα και συμβάλλει στη δημιουργία υψηλής ποιότητας συστημάτων.

4. Ανάλυση σημασιολογίας στα βασικά διαγράμματα UML

4.1 Σημασιολογία Διαγραμμάτων Περιπτώσεων Χρήσης (Use Case)

Εισαγωγή

Η μοντελοποίηση περιπτώσεων χρήσης (Use Case Modeling) αποτελεί μία από τις πιο διαδεδομένες τεχνικές καταγραφής και ανάλυσης απαιτήσεων λογισμικού. Η τεχνική αυτή εισήχθη αρχικά από τον Ivar Jacobson στο πλαίσιο της αντικειμενοστραφούς μηχανικής λογισμικού και ενσωματώθηκε αργότερα στη γλώσσα μοντελοποίησης UML (Unified Modeling Language) (Jacobson et al., 1992).

Μια περίπτωση χρήσης (use case) περιγράφει τον τρόπο με τον οποίο ένας χρήστης ή ένα εξωτερικό σύστημα (actor) αλληλεπιδρά με ένα σύστημα ώστε να επιτευχθεί ένας συγκεκριμένος στόχος. Στην UML, μια περίπτωση χρήσης ορίζεται ως η προδιαγραφή μιας ακολουθίας ενεργειών που εκτελεί το σύστημα και παράγει ένα παρατηρήσιμο αποτέλεσμα με αξία για έναν ή περισσότερους δρώντες (actors). Οι περιπτώσεις χρήσης χρησιμοποιούνται κυρίως στη φάση της ανάλυσης απαιτήσεων, καθώς επιτρέπουν την καταγραφή της λειτουργικότητας ενός συστήματος από την οπτική των χρηστών και των ενδιαφερόμενων μερών

Τα διαγράμματα περιπτώσεων χρήσης (UCDs), περιγράφουν τις λειτουργίες ενός συστήματος με βάση τον τρόπο με τον οποίον οι χρήστες επιτυγχάνουν τους στόχους τους χρησιμοποιώντας το σύστημα (Friedenthal, S., Moore, A., & Steiner, R, 2014 · Kautz, Rumpe, & Wachmeister, 2022). Επειδή τα διαγράμματα αυτά είναι απλά και σαφή, χρησιμοποιούνται τόσο στην ανάπτυξη πολύπλοκων συστημάτων λογισμικού όσο και για την ανάπτυξη συστημάτων γενικά.

Σύμφωνα με τους (France & Rumpe, 2007· Pretschner, Broy, Kruger & Stauner, 2007) επειδή η πολυπλοκότητα των συστημάτων λογισμικού αυξάνεται, είναι πρόκληση για τους μηχανικούς λογισμικού να έχουν μια συνολική εικόνα για τη λειτουργικότητα των συστημάτων λογισμικού που βρίσκονται υπό ανάπτυξη. Η δυνατότητα αυτή, είναι ιδιαίτερα σημαντική όταν συγκρίνονται δύο διαφορετικές εκδόσεις ή δύο διαφορετικές παραλλαγές του ίδιου συστήματος (Kautz et al., 2022). Επιπλέον, είναι σημαντικό να προκύψουν συμπεράσματα σχετικά με τη συμβατότητα και την καταλληλότητα των προδιαγραφών του

συστήματος. Ένα παράδειγμα θα ήταν η περίπτωση ενός προμηθευτή και ενός αγοραστή: Ο προμηθευτής θα μπορούσε να χρησιμοποιήσει ένα διάγραμμα περιπτώσεων για να καθορίσει τη λειτουργικότητα του συστήματος που προσφέρει. Από την άλλη ο αγοραστής, θα μπορούσε να αξιοποιήσει ένα διάγραμμα περιπτώσεων για να καθορίσει τις ιδιότητες του συστήματος που επιθυμεί να αγοράσει. Και οι δύο θα μπορούσαν να χρησιμοποιήσουν μια προσέγγιση σημασιολογικής διαφοροποίησης για να διαπιστώσουν με ακρίβεια τις διαφορές ανάμεσα στο προσφερόμενο και το απαιτούμενο από τον αγοραστή σύστημα ώστε να χρησιμοποιήσουν το αποτέλεσμα ως βάση για τις περαιτέρω διαπραγματεύσεις τους.

Μια άλλη πρόκληση που προκύπτει στην αξιοποίηση των περιπτώσεων χρήσης είναι ότι μικρές αλλαγές σε μια περιγραφή υψηλού επιπέδου, μπορεί να έχουν μεγάλες επιπτώσεις στη συνολική συμπεριφορά του συστήματος. Η γνώση της σημασιολογικής διαφοράς ανάμεσα σε 2 διαγράμματα περιπτώσεων χρήσης υποστηρίζει τους μηχανικούς λογισμικού στο να κατανοήσουν τις διαφορές στις λειτουργικότητες των συστημάτων. Άρα, τους επιτρέπει να εντοπίσουν τους συνδυασμούς περιπτώσεων χρήσης και δρώντων (actors) που είναι δυνατοί στη μία έκδοση αλλά όχι στην άλλη (Maoz, Ringert & Rumpe, 2011c,d ·Maoz & Ringert, 2015, 2018· Kautz & Rumpe 2018a ·Kautz 2021·Kautz et al., 2022).

Όταν αναπτύσσονται κρίσιμα συστήματα με διαγράμματα περιπτώσεων χρήσης ως σημείο εκκίνησης, είναι απαραίτητη μια επακριβώς καθορισμένη σημασιολογία για την επιτυχή ανάπτυξη και ασφαλή λειτουργία του υπό ανάπτυξη συστήματος. Διαφορετικά η έννοια των διαγραμμάτων περίπτωσης μπορεί να παρερμηνευτεί ή να ερμηνευτεί διαφορετικά από διαφορετικούς προγραμματιστές με αποτέλεσμα να δημιουργηθούν κρίσιμα σφάλματα στο αναπτυγμένο προϊόν. Επιπρόσθετα, μια επακριβώς καθορισμένη σημασιολογία επιτρέπει την ανάπτυξη ενός τελεστή σημασιολογικής διαφοροποίησης ο οποίος υποστηρίζει τους προγραμματιστές στη σύγκριση διαφορετικών εκδόσεων διαγραμμάτων περίπτωσης και αποκαλύπτει τις διαφορές όσον αφορά τις διαφορές στις σημασίες των διαγραμμάτων περίπτωσης (Maoz et al. 2011c, 2012· Kautz & Rumpe 2018a· Kautz, 2021·Kautz, 2022).

Βασικές έννοιες των Use Cases

Η μοντελοποίηση περιπτώσεων χρήσης βασίζεται σε ορισμένες βασικές έννοιες.

Actor (Δρών)

Ο actor είναι μια οντότητα που αλληλεπιδρά με το σύστημα. Μπορεί να είναι:

- ανθρώπινος χρήστης
- εξωτερικό πληροφοριακό σύστημα
- συσκευή

Σημαντικό είναι ότι ο actor **αναπαριστά έναν ρόλο** και όχι απαραίτητα ένα συγκεκριμένο άτομο. Αντιπροσωπεύει συνήθως κάποιο χρήστη του συστήματος, αλλά μπορεί να είναι και ένας οργανισμός ή κάποιο άλλο εξωτερικό σύστημα.

Παραδείγματα actors σε ένα σύστημα βιβλιοθήκης είναι:

- Χρήστης
- Βιβλιοθηκονόμος
- Σύστημα πληρωμών

Use Case

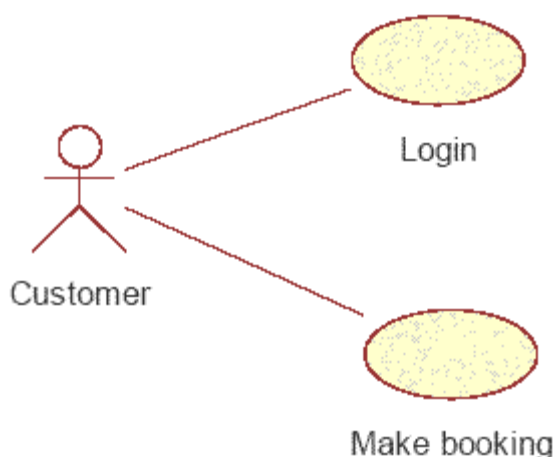
Μια περίπτωση χρήσης περιγράφει ένα **σύνολο αλληλεπιδράσεων μεταξύ actor και συστήματος** που οδηγούν στην επίτευξη ενός στόχου του χρήστη. Μια περίπτωση χρήσης είναι μια αναπαράσταση ενός στόχου του χρήστη που πρέπει να επιτευχθεί. Ένα σύστημα μπορεί να θεωρηθεί ως μια συλλογή περιπτώσεων χρήσης που αναπαρίστανται μαζί σε ένα μοντέλο περίπτωσης χρήσης. Το μοντέλο περίπτωσης χρήσης είναι μια εικόνα που προορίζεται να είναι εύκολα «εξερευνήσιμη» και μεταβλητή τόσο από πελάτες όσο και από προγραμματιστές (Jacobson, Jonsson et al.'92). Ένα μοντέλο περίπτωσης χρήσης (UCM) έχει ηθοποιούς, οβάλ εργασιών, συσχετίσεις και ένα όριο συστήματος. Τα UCM ξεκινούν απλά και γίνονται πιο σύνθετα με την πάροδο του χρόνου (Pooley και Stevens'99). Κάθε περίπτωση χρήσης έχει δύο μέρη: μια γραφική αναπαράσταση και μια κειμενική. Το μέρος κειμένου προσθέτει λεπτομέρειες στη γραφική αναπαράσταση. Είναι βολικό να θεωρήσουμε το γραφικό στοιχείο μιας περίπτωσης χρήσης ως ένα είδος πίνακα περιεχομένων που κατευθύνει τον αναγνώστη (Merric and Barrow, 2018).

Παραδείγματα:

- Δανεισμός βιβλίου
- Επιστροφή βιβλίου
- Αναζήτηση βιβλίου

Κάθε περίπτωση χρήσης συνήθως περιλαμβάνει:

- βασική ροή γεγονότων (main flow)
- εναλλακτικές ροές (alternative flows)
- εξαιρέσεις (exception flows)

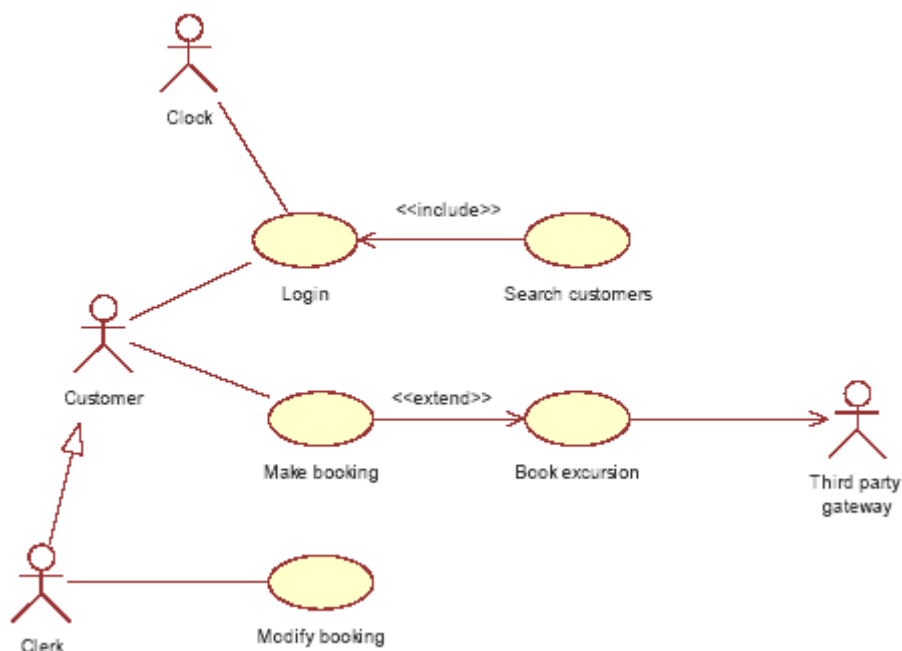


Εικ 4-1 Ένα απλό μοντέλο περίπτωσης χρήσης, που περιλαμβάνει έναν ηθοποιό, δύο ελλείψεις, με συσχετίσεις.

Στο Σχήμα 4.1, παρουσιάζεται η πιο βασική μορφή μιας περίπτωσης χρήσης αναπαρίσταται γραφικά από ένα οβάλ εργασίας με όνομα που αντιπροσωπεύει έναν στόχο χρήστη. Το παραπάνω διάγραμμα απεικονίζει την ικανότητα ενός πελάτη να συνδέεται και να κάνει κράτηση. Καθώς περισσότερες λεπτομέρειες γίνονται διαθέσιμες, προστίθενται στην περίπτωση χρήσης. Με την πάροδο του χρόνου, η περίπτωση χρήσης γίνεται κυρίως μια

κειμενική κατασκευή που περιγράφει τη συμπεριφορά του συστήματος σε ημι-τυποποιημένη φυσική γλώσσα.

Οι περιπτώσεις χρήσης ενεργοποιούνται από δράστες. Ο Jacobson ορίζει έναν δράστη ως μια κατασκευή που αντιπροσωπεύει έναν ρόλο που μπορεί να διαδραματίσει ένας χρήστης (Jacobson, Jonsson et al., 92). Ένας δράστης που ενεργοποιεί μια περίπτωση χρήσης είναι ένας άνθρωπος χρήστης που παίζει έναν σαφώς καθορισμένο ρόλο. Οι ανθρώπινοι δράστες είναι επίσης γνωστοί ως πρωτεύοντες δράστες. Υπάρχουν, επιπλέον, δευτερεύοντες δράστες που αντιπροσωπεύουν άλλα συστήματα λογισμικού με τα οποία πρέπει να δημιουργηθεί επικοινωνία. Τέλος, υπάρχουν δευτερεύοντες δράστες που αντιπροσωπεύουν ενθυλακωμένη συμπεριφορά. Σκεφτείτε την έννοια ενός δράστη ρολογιού που ενεργοποιεί αυτόματα περιπτώσεις χρήσης ή μια μηχανή ηλεκτρονικού ταχυδρομείου που προσφέρει κοινή συμπεριφορά σε πολλές μεμονωμένες περιπτώσεις χρήσης για να βοηθήσει την επικοινωνία. Σύμφωνα με τον Jacobson, οι δράστες είναι εύκολο να ανακαλυφθούν, καθώς προέρχονται από μια ανάλυση πελατών, συνεργατών, προμηθευτών, αρχών και θυγατρικών (Jacobson, 95). Οι δράστες θεωρούνται εκτός των ορίων του συστήματος, αν και οι πρωτεύοντες δράστες μπορεί να έχουν μια εσωτερική αναπαράσταση ακόμη και αν βρίσκεται στο ασήμαντο επίπεδο δικαιωμάτων, όπως στην περίπτωση ενός πλασματικού πελάτη που θα χρειαζόταν να έχει δικαιώματα για τον δικό του λογαριασμό. Είναι χρήσιμο να διακρίνουμε τον τελικό πρωτεύοντα δράστη. Ο δράστης που «ενδιαφέρεται πραγματικά» για την επιτυχή ολοκλήρωση μιας περίπτωσης χρήσης, και όχι ένας πληρεξούσιος δράστης που ενεργεί στη θέση του. Έτσι, ένας υπάλληλος μπορεί να θεωρηθεί ως πληρεξούσιος δράστης εκ μέρους ενός πελάτη. Αυτό οδηγεί στην ανάπτυξη ιεραρχιών πρωταρχικών δρώντων που σχετίζονται μέσω εξειδίκευσης (Merric and Barrow, 2018).



Εικ. 4-2 Τύποι ηθοποιών ως μέρος μιας ιεραρχίας. Ο Υπάλληλος και ο Πελάτης είναι και οι δύο πρωτεύοντες (ανθρώπινοι) ηθοποιοί. Η third party gateway είναι ένα παλαιό σύστημα που δέχεται κρατήσεις εκδρομών. Το Ρολόι είναι μια αυτοματοποιημένη ευκολία που αποσυνδέει τους χρήστες από το σύστημα όταν είναι ανενεργοί για ένα χρονικό διάστημα.

Οι ηθοποιοί μπορούν να υλοποιηθούν από συσκευές, άλλα συστήματα ή από ανθρώπους. Και οι τρεις απεικονίζονται στο Σχήμα 4.2. Οι ηθοποιοί συνδέονται με περιπτώσεις χρήσης μόνο μέσω συσχέτισης, που σημαίνει ότι επικοινωνούν τόσο στέλνοντας όσο και λαμβάνοντας πληροφορίες. Οι πρωτεύοντες ηθοποιοί έχουν έναν στόχο που επιθυμούν να δουν να εκπληρώνεται. Οι δευτερεύοντες ηθοποιοί βοηθούν στην επίτευξη των στόχων των πρωτευόντων ηθοποιών. Ένα οβάλ εργασίας περίπτωσης χρήσης σχετίζεται με μια περιγραφή κειμένου. Ένας ηθοποιός ενεργοποιεί μια περίπτωσης χρήσης, που ονομάζεται σενάριο, η οποία εκτελεί μια συναλλαγή εκ μέρους του ηθοποιού. Ο Cockburn θεωρεί ότι η βασική πορεία δράσης μπορεί να θεωρηθεί ως το «αρχέτυπο σενάριο» (Cockburn'01). Περίεργως, αν και η UML δέχεται ότι οι ηθοποιοί μπορούν να συσχετιστούν μέσω

εξειδίκευσης, δεν ορίζει τις περιπτώσεις χρήσης ως ικανές για συσχέτιση μέσω εξειδίκευσης. Αυτός είναι ένας ατυχής περιορισμός.

Παραδείγματα περιπτώσεων χρήσης

Διαχείριση του ιατρικού φακέλου των ασθενών

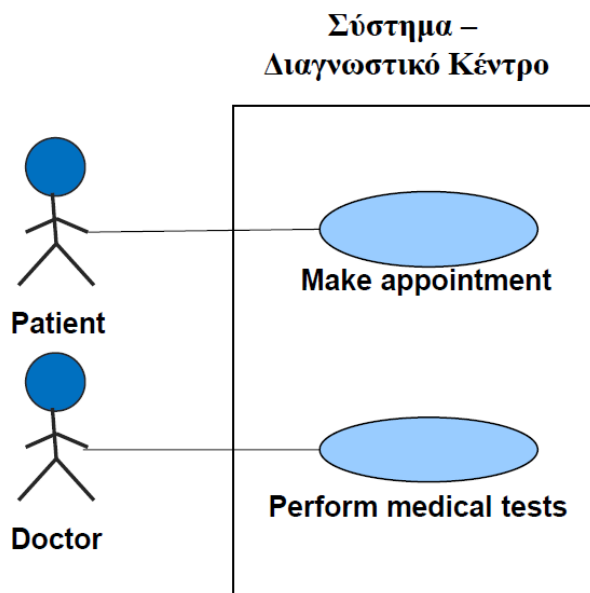
Διακρίνουμε τις ακόλουθες περιπτώσεις χρήσης:

- «Διαχείριση προσωπικών στοιχείων ασθενών»
- «Διαχείριση ιατρικής πληροφορίας ασθενών»

Διαχείριση κλινικού συστήματος

Διακρίνουμε τις ακόλουθες περιπτώσεις χρήσης:

- «Η περίπτωση χρήσης «Καθορισμός ραντεβού» (Make appointment)
- «Διαχείριση ιατρικής πληροφορίας ασθενών»



Εικ. 4-3 Διαχείριση κλινικού συστήματος

Σημασιολογία Σχέσεων στα Διαγράμματα Use Case

Στα διαγράμματα περιπτώσεων χρήσης εμφανίζονται διάφορες σχέσεις μεταξύ των στοιχείων, οι οποίες αποδίδουν τη δομή και τη συμπεριφορά του συστήματος.

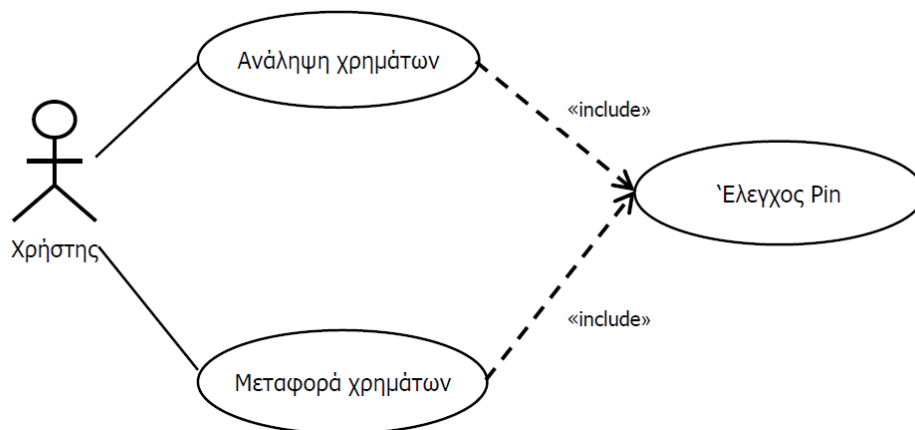
Σχέσεις μεταξύ περιπτώσεων χρήσης

Οι σχέσεις που αναπτύσσονται στα σε ένα διάγραμμα περιπτώσεων χρήσης είναι οι παρακάτω:

- «Περιλαμβάνει» (include)
- «Επεκτείνει» (extend)
- «Γενικεύει» (/generalization)

Η σχέση «include» χρησιμοποιείται όταν μια περίπτωση χρήσης επαναχρησιμοποιεί υποχρεωτικά τη λειτουργικότητα μιας άλλης περίπτωσης χρήσης. Με αυτόν τον τρόπο αποφεύγεται η επανάληψη περιγραφών και επιτυγχάνεται καλύτερη οργάνωση των λειτουργικών απαιτήσεων (Pressman & Maxim, 2015). Από σημασιολογική άποψη, η σχέση αυτή υποδηλώνει ότι η συμπεριφορά της περιλαμβανόμενης περίπτωσης χρήσης αποτελεί αναπόσπαστο μέρος της βασικής λειτουργίας.

Παράδειγμα της σχέσης “include” φαίνεται παρακάτω. Μια περίπτωση χρήσης περιλαμβάνει τη συμπεριφορά μιας άλλης. Για παράδειγμα, οι περιπτώσεις χρήσης «Ανάληψη χρημάτων» και «Μεταφορά χρημάτων» περιλαμβάνουν την περίπτωση «Έλεγχος Pin».

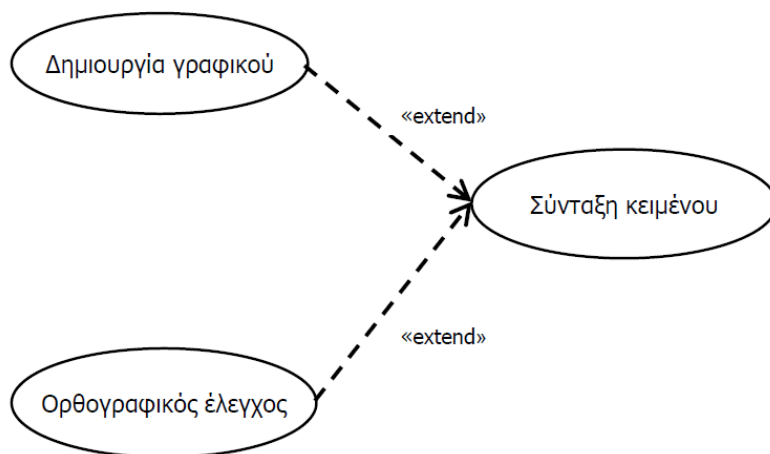


Εικ. 4-4 Παράδειγμα σχέσης «include»

Η περίπτωση χρήσης «Έλεγχος Pin», εντάσσεται στην ενεργοποίηση της περίπτωσης χρήσης «Μεταφορά χρημάτων» και της περίπτωσης χρήσης «Ανάληψη Χρημάτων».

Η σχέση «extend» χρησιμοποιείται για την αναπαράσταση προαιρετικής ή υπό συνθήκη συμπεριφοράς. Δηλαδή, μια βασική περίπτωση χρήσης μπορεί να επεκταθεί με επιπλέον λειτουργίες όταν ικανοποιούνται συγκεκριμένες συνθήκες (Fowler, 2004). Σημασιολογικά, η σχέση αυτή επιτρέπει τη μοντελοποίηση παραλλαγών συμπεριφοράς χωρίς να τροποποιείται η βασική ροή της περίπτωσης χρήσης.

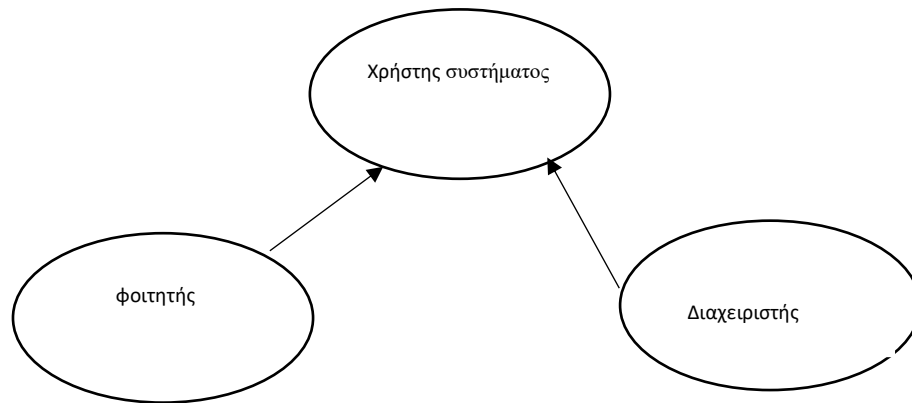
Παράδειγμα της σχέσης “extend” φαίνεται στην εικ. 4.5. Μια περίπτωση χρήσης επεκτείνει κάποια άλλη προσθέτοντας επιπλέον ενέργειες στη γενική περίπτωση χρήσης.



Εικ. 4-5 Η περίπτωση χρήσης «Σύνταξη κειμένου» επεκτείνει τις περιπτώσεις χρήσης «Δημιουργία γραφικού» και «Ορθογραφικός Έλεγχος»

Η γενίκευση (generalization) χρησιμοποιείται για την αναπαράσταση ιεραρχικών σχέσεων μεταξύ δρώντων ή μεταξύ περιπτώσεων χρήσης. Για παράδειγμα, ένας «Χρήστης Συστήματος» μπορεί να αποτελεί γενικό τύπο δρώντα, ενώ ειδικότεροι τύποι μπορεί να είναι ο «Φοιτητής» και ο «Διαχειριστής». Η σημασιολογία της γενίκευσης υποδηλώνει ότι οι εξειδικευμένοι δρώντες κληρονομούν τις ιδιότητες και τις σχέσεις του γενικού δρώντα (Booch et al., 2005).

Η έννοια αυτή είναι ιδιαίτερα χρήσιμη για την οργάνωση πολύπλοκων συστημάτων με πολλούς τύπους χρηστών. Παράδειγμα της σχέσης «Γενικεύει», φαίνεται στην εικόνα 4.6.



Εικ. 4-6 Σχέση Γενίκευσης

Ο Ρόλος της Σημασιολογίας στην Ανάλυση Απαιτήσεων

Η σημασιολογία των διαγραμμάτων περιπτώσεων χρήσης παίζει σημαντικό ρόλο στη διαδικασία της ανάλυσης απαιτήσεων στη Μηχανική Λογισμικού. Μέσω της σωστής ερμηνείας των στοιχείων του διαγράμματος, οι αναλυτές μπορούν να κατανοήσουν καλύτερα τις ανάγκες των χρηστών και να μεταφράσουν αυτές τις ανάγκες σε λειτουργικές προδιαγραφές συστήματος. Επιπλέον, τα διαγράμματα περιπτώσεων χρήσης λειτουργούν ως μέσο επικοινωνίας μεταξύ προγραμματιστών, αναλυτών και ενδιαφερόμενων μερών, διευκολύνοντας τη συνεργασία κατά την ανάπτυξη λογισμικού (Cockburn, 2001).

Συμπεράσματα

Η σημασιολογία των διαγραμμάτων περιπτώσεων χρήσης αποτελεί βασικό στοιχείο για την κατανόηση της λειτουργικότητας ενός συστήματος. Μέσω της σωστής ερμηνείας των δρώντων, των περιπτώσεων χρήσης και των σχέσεων μεταξύ τους, είναι δυνατό να αποτυπωθούν με σαφήνεια οι λειτουργικές απαιτήσεις ενός πληροφοριακού συστήματος. Η χρήση των διαγραμμάτων αυτών στο πλαίσιο της UML παρέχει ένα τυποποιημένο και κατανοητό τρόπο μοντελοποίησης συστημάτων, συμβάλλοντας σημαντικά στη βελτίωση της διαδικασίας ανάπτυξης λογισμικού.

4.2 Σημασιολογία Διαγραμμάτων Κλάσεων

Εισαγωγή

Τα διαγράμματα κλάσεων καταγράφουν τις κλάσεις που ανήκουν σε ένα σύστημα, δηλαδή τους τύπους των αντικειμένων και τις σχέσεις που υπάρχουν μεταξύ τους. Μια κλάση περιγράφει ένα σύνολο αντικειμένων με όμοιες ιδιότητες, κοινή συμπεριφορά και κοινές συσχετίσεις με άλλα αντικείμενα.

Το διάγραμμα κλάσεων αναπαριστά τη στατική δομή του συστήματος σε επίπεδο κλάσεων (class structure) και τα περιεχόμενά τους (contents). Επιπλέον, αναπαριστά τις σχέσεις μεταξύ των κλάσεων όπως είναι η συσχέτιση (association), η εξειδίκευση (specialization), η γενίκευση (generalization) και άλλες. Ένα παράδειγμα είναι το σχέδιο με τα εξαρτήματα ενός αυτοκινήτου το οποίο δείχνει πώς συναρμολογούνται αλλά δεν δείχνει πώς αυτά τα εξαρτήματα συμπεριφέρονται όταν η μηχανή λειτουργήσει.

Διάγραμμα Κλάσεων - Αναπαράσταση

Σε ένα διάγραμμα κλάσεων τα ονόματα των κλάσεων πρέπει να είναι περιγραφικά και να ονομάζονται σύμφωνα με τις επιχειρησιακές οντότητες (χρησιμοποιείται ουσιαστικό ή πρόταση που περιέχει ουσιαστικό, σε ενικό αριθμό). Αρχικά περιλαμβάνει τις βασικές κλάσεις του συστήματος και στη συνέχεια όσο προχωρά η ανάλυση του συστήματος, εμπλουτίζεται με νέες κλάσεις. Στη συνέχεια προστίθενται κλάσεις και διεπαφές ανάλογα με τον τρόπο υλοποίησης του συστήματος (π.χ. κλάσεις για τη σύνδεση με μια διεπαφή ή με ένα άλλο σύστημα).

Έστω για παράδειγμα, η κλάση Μάθημα που φαίνεται στην παρακάτω εικόνα:

Μια κλάση «Μάθημα» αναπαριστά την οντότητα του μαθήματος σε ένα σύστημα. Ενσωματώνει πληροφορίες για το μάθημα όπως τον κωδικό του μαθήματος, τη θεματική ενότητα του μαθήματος, τον τίτλο του μαθήματος και το όνομα του διδάσκοντα. Επιπλέον, η κλάση «Μάθημα» έχει κάποια λειτουργικότητα, η οποία αναπαρίσταται με τις λειτουργίες Προσθήκη Μαθήματος (), Διαγραφή Μαθήματος (), Μεταβολή Στοιχείων Μαθήματος (), Ανάθεση Μαθήματος ().

Μάθημα
Κωδικός μαθήματος
Θεματική ενότητα
Τίτλος
Διδάσκων
Προσθήκη Μαθήματος ()
Διαγραφή Μαθήματος ()
Μεταβολή στοιχείων Μαθήματος ()
Ανάθεση Μαθήματος ()

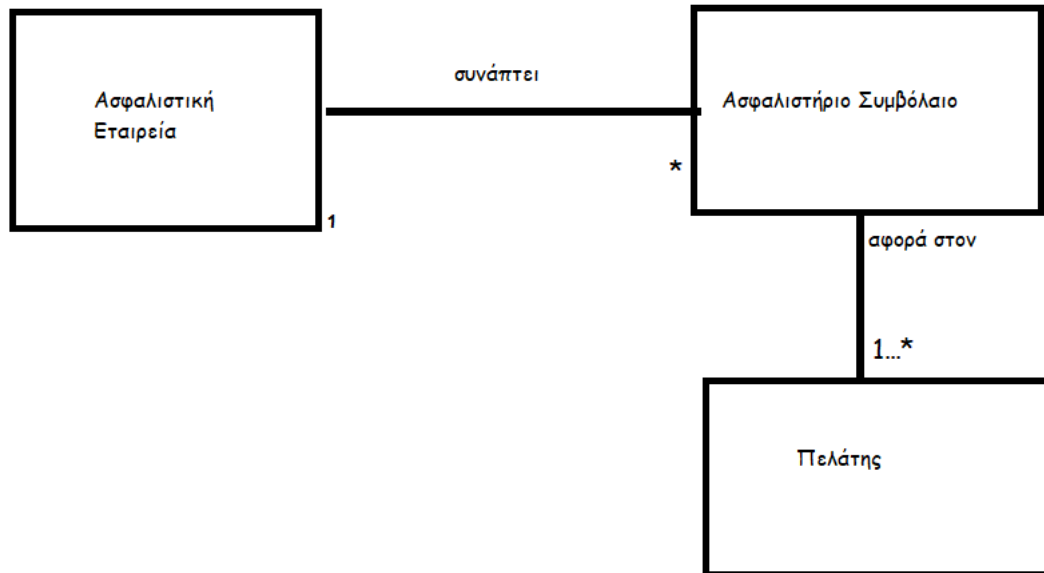
Εικ. 4-7 Κλάση Μάθημα: ιδιότητες και λειτουργίες

Συσχετίσεις κλάσεων

Η συσχέτιση μεταξύ δύο κλάσεων χρειάζεται όταν για να λειτουργήσει μια κλάση θα πρέπει να συνεργαστεί με μία ή περισσότερες άλλες κλάσεις. Στην εικ.....παρακάτω, φαίνεται το παράδειγμα μιας ασφαλιστικής εταιρείας. Σύμφωνα με το διάγραμμα:

- Μια ασφαλιστική εταιρεία μπορεί να έχει από 0 έως πολλά ασφαλιστικά συμβόλαια.
- Ένας πελάτης μπορεί να έχει από 0 έως πολλά ασφαλιστικά συμβόλαια.
- Ένα συμβόλαιο σχετίζεται με μια ασφαλιστική εταιρεία.
- Ένα συμβόλαιο αφορά έναν ή περισσότερους πελάτες.

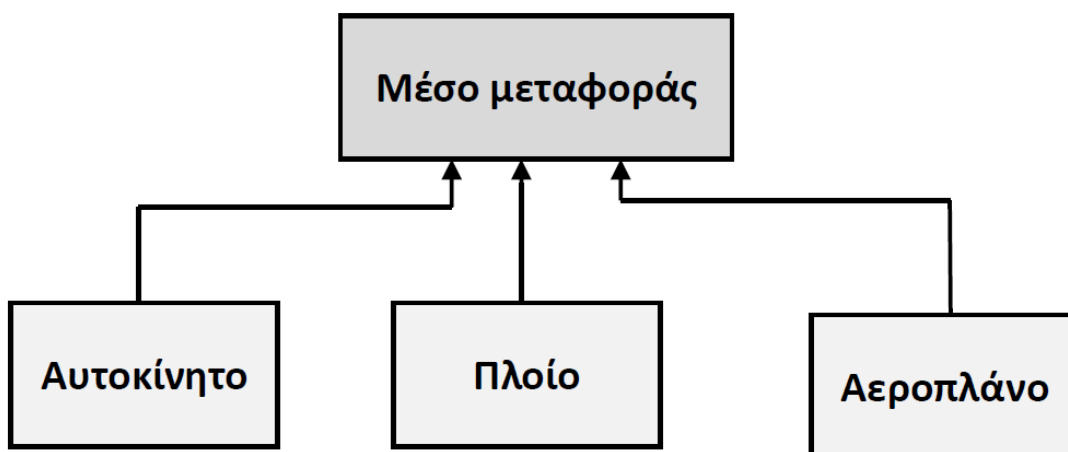
Παράδειγμα 1. Ασφαλιστική εταιρεία



Εικ. 4-8 Ασφαλιστική Εταιρεία

Γενίκευση (Generalization)

Η γενίκευση είναι ειδική μορφή συσχέτισης στην οποία μια γενική κλάση αποτελεί τη βάση για τη δήλωση μίας ή περισσότερων ειδικότερων κλάσεων. Η γενική κλάση ονομάζεται υπερκλάση και οι ειδικές κλάσεις ονομάζονται υποκλάσεις. Η γενίκευση στις περισσότερες γλώσσες προγραμματισμού υλοποιείται με το μηχανισμό της κληρονομικότητας ή της επέκτασης. Η γενική κλάση παρέχει λειτουργίες, ιδιότητες, συσχετίσεις κ.λ.π, χρήσιμες σε όλες τις υποκλάσεις. Οι υποκλάσεις παρέχουν επιπλέον λειτουργίες από την υπερκλάση ή εξειδικεύουν τη συμπεριφορά τους. Όπως φαίνεται στην εικ. το αυτοκίνητο, το πλοίο και το αεροπλάνο είναι ένα είδος Μέσου Μεταφοράς.

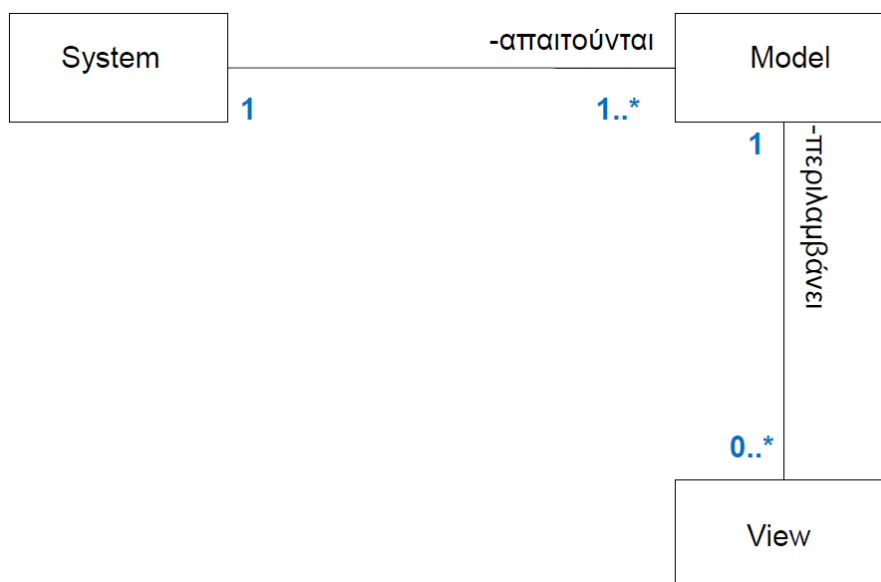


Εικ. 4-9 Γενίκευση

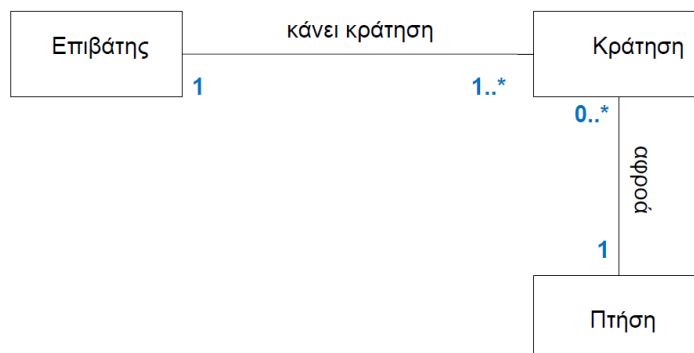
Σχεδίαση διαγράμματος κλάσεων

Για τη σχεδίαση διαγράμματος κλάσεων για ένα πρόβλημα, θα πρέπει:

- Να περιγραφεί το πρόβλημα: Ένα μοντέλο, μπορεί να περιλαμβάνει διαφορετικές όψεις (views). Για την περιγραφή ενός συστήματος (systems), απαιτούνται μια σειρά από διαφορετικά μοντέλα.
- Να προσδιοριστούν οι κλάσεις και το πλήθος τους: (model view, system).



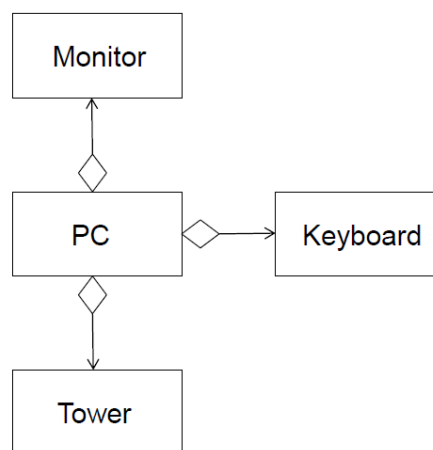
Εικ. 4-10 Διάγραμμα κλάσεων-παράδειγμα 1



Εικ. 4-11 Διάγραμμα κλάσεων-παράδειγμα 2:Συσχέτιση κράτηση-πτήση

Συσσώρευση (aggregation)

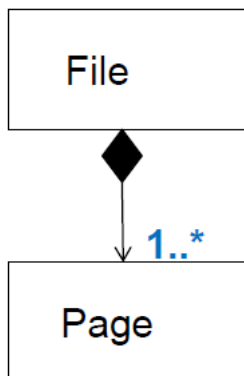
Μία κλάση αποτελεί συλλογή από άλλες κλάσεις: Περιλαμβάνονται τα παρακάτω: «Αποτελείται από», «περιέχει», «περιλαμβάνει», «έχει». Συμβολίζεται με μια συσχέτιση από το όλο προς το μέρος στην οποία τοποθετείται ένας άσπρος ρόμβος στην πλευρά του όλου (aggregating entity).



Εικόνα 4-12 Παράδειγμα συσσώρευσης: PC-monitor, keyboard, tower

Σύνθεση (composition)

Η συσσώρευση που αναφέραμε παραπάνω, χρησιμοποιείται όταν οι επιμέρους κλάσεις έχουν διάρκεια ζωής ανεξάρτητη από τη σύνθετη κλάση. Για παράδειγμα μια ρόδα μπορεί να υπάρξει ανεξάρτητα από ένα ποδήλατο. Η σύνθεση χρησιμοποιείται όταν οι επιμέρους κλάσεις έχουν διάρκεια ζωής που εξαρτάται και συμπίπτει με αυτή της σύνθετης κλάσης. Για παράδειγμα, αν σβήσουμε ένα ηλεκτρονικό αρχείο, τότε σβήνονται ταυτόχρονα και οι σελίδες του. Δεν μπορούν να υπάρξουν ανεξάρτητα από το αρχείο. Δηλαδή, όταν διαγράφεται το όλον, διαγράφονται τα μέρη του.



Εικ. 4-13 Παράδειγμα σύνθεσης

4.3 Σημασιολογία Διαγραμμάτων Δραστηριοτήτων (Activity Diagrams)

Εισαγωγή

Τα Διαγράμματα Δραστηριοτήτων (Activity Diagrams) αποτελούν βασικό εργαλείο της UML για τη μοντελοποίηση της δυναμικής συμπεριφοράς ενός συστήματος, επιτρέποντας την αναπαράσταση της ροής εργασιών και της αλληλουχίας ενεργειών (Πρέντζα, 2025).

Η σημασιολογία τους αναφέρεται στον τρόπο με τον οποίο αποδίδεται νόημα στα επιμέρους στοιχεία τους, όπως δραστηριότητες, μεταβάσεις, συνθήκες και παράλληλες ροές. Η σωστή κατανόηση της σημασιολογίας είναι απαραίτητη για την αποφυγή ασάφειας και την ακριβή περιγραφή της συμπεριφοράς ενός συστήματος (Booch, Rumbaugh and Jacobson, 2005).

Ρόλος στη Μοντελοποίηση Συστημάτων

Στη UML, τα Activity Diagrams χρησιμοποιούνται κυρίως για τη μοντελοποίηση της συμπεριφοράς (behavior modeling), καλύπτοντας:

- Ανάλυση απαιτήσεων
- Περιγραφή περιπτώσεων χρήσης
- Επιχειρησιακές διαδικασίες
- Αλγοριθμικές ροές

Σε αντίθεση με τα Use Case Diagrams, που περιγράφουν την εξωτερική αλληλεπίδραση, τα Activity Diagrams επικεντρώνονται στην εσωτερική λειτουργία του συστήματος (Πρέντζα, 2025).

Βασικά Στοιχεία και Σημασιολογία

Δραστηριότητες (Activities)

Οι δραστηριότητες αναπαριστούν μονάδες εργασίας και αποτελούν τα βασικά δομικά στοιχεία των διαγραμμάτων. Μπορούν να είναι είτε χειροκίνητες είτε αυτοματοποιημένες διαδικασίες (Πρέντζα, 2025).

Ροή Ελέγχου (Control Flow)

Η ροή ελέγχου καθορίζει τη σειρά εκτέλεσης των δραστηριοτήτων. Η μετάβαση πραγματοποιείται με την ολοκλήρωση μιας δραστηριότητας, χωρίς απαραίτητα εξωτερικό γεγονός (Sommerville, 2007).

Κόμβοι Απόφασης

Οι κόμβοι απόφασης επιτρέπουν τη διακλάδωση της ροής με βάση συνθήκες (guard conditions), οι οποίες είναι λογικές εκφράσεις που καθορίζουν τη διαδρομή εκτέλεσης (Πρέντζα, 2025)

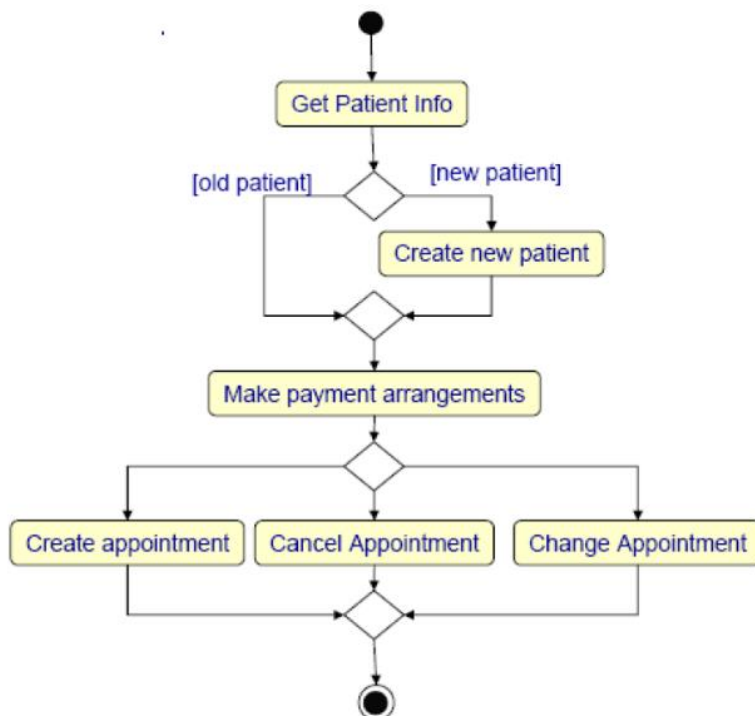
Τα στοιχεία των διαγραμμάτων δραστηριοτήτων φαίνονται στον παρακάτω πίνακα (Πρέντζα, 2025):

Στοιχεία Διαγραμμάτων Δραστηριοτήτων	Σύμβολο
Ενώσεις – Merge: Σημείο/κόμβος ένωσης/συγχώνευσης χρησιμοποιείται συνήθως μετά από ένα σημείο απόφασης	
Ταυτόχρονες Δραστηριότητες - Concurrent Activities: Μερικές δραστηριότητες εκτελούνται ταυτόχρονα.	
Διακλάδωση/Ένωση - Fork/Join:	
Τελική Δραστηριότητα - Final Activity: Το τέλος ενός διαγράμματος δραστηριοτήτων.	

Εικ. 4-14 Στοιχεία Διαγραμμάτων Δραστηριοτήτων

Παραδείγματα Διαγράμματος Δραστηριοτήτων

Α. Το παραπάνω διάγραμμα απεικονίζει μια τυπική ροή διαχείρισης ραντεβού νοσοκομείου.

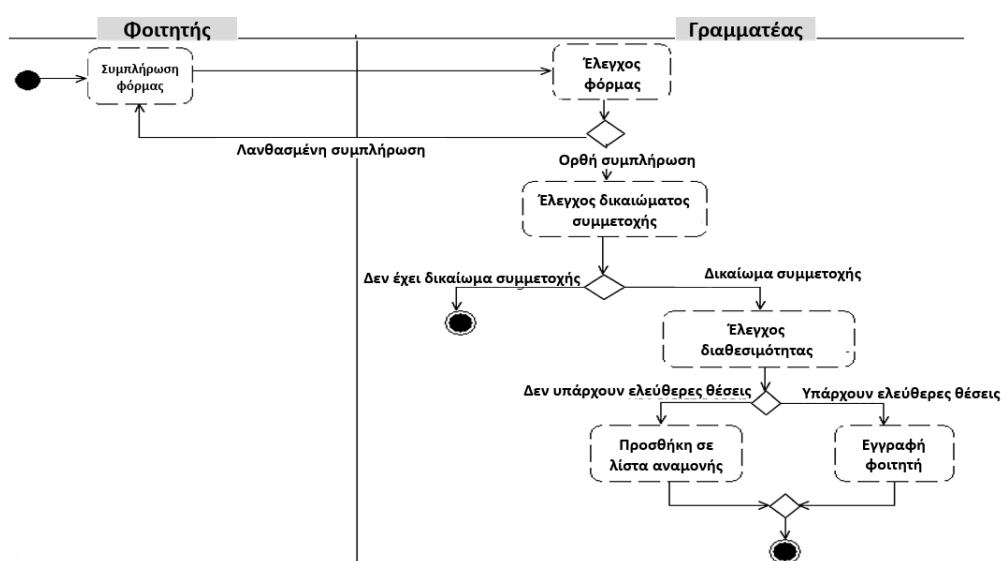


Εικ. 4-15 Διαχείριση ραντεβού νοσοκομείου

Β. Σύστημα Διαχείρισης σεμιναρίων ενός Πανεπιστημίου

Το διάγραμμα δραστηριοτήτων θα πρέπει να περιλαμβάνει:

- Τη συμπλήρωση φόρμας για συμμετοχή σε σεμινάριο από το φοιτητή
- Τον έλεγχο της φόρμας από τη Γραμματεία
- Αν βρεθεί λάθος, ο φοιτητής συμπληρώνει τη φόρμα ξανά, αλλιώς η γραμματεία ελέγχει αν ο φοιτητής έχει δικαίωμα να συμμετάσχει στο σεμινάριο.
- Σε περίπτωση που ο φοιτητής δεν έχει δικαίωμα να εγγραφεί στο σεμινάριο, η διαδικασία ολοκληρώνεται, αλλιώς ο γραμματέας ελέγχει αν υπάρχουν ελεύθερες θέσεις στο σεμινάριο.
- Αν υπάρχουν ελεύθερες θέσεις, η γραμματεία γράφει το φοιτητή στο σεμινάριο, διαφορετικά τον βάζει σε λίστα αναμονής και η διαδικασία ολοκληρώνεται.
-

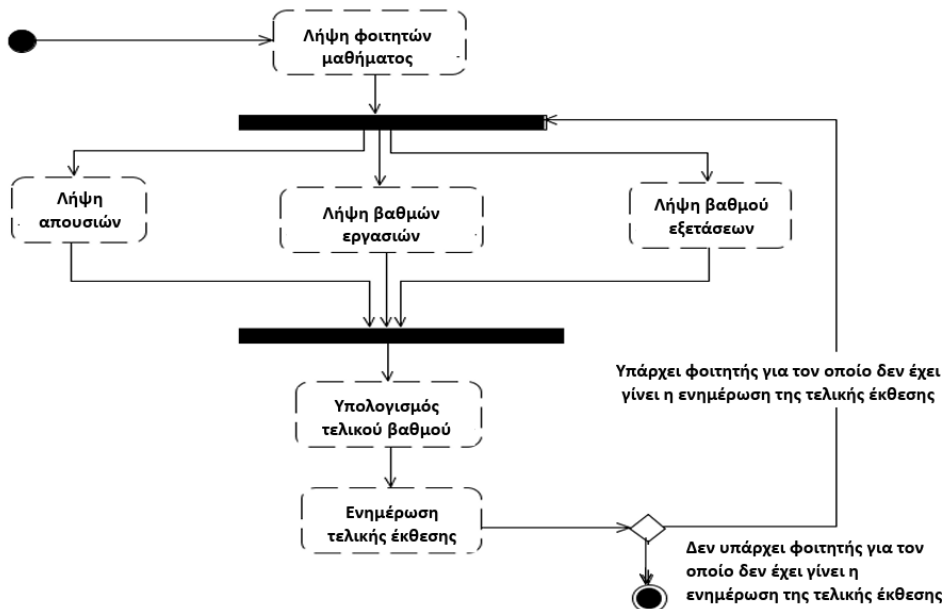


Εικ. 4-16 Εγγραφή φοιτητή σε σεμινάριο (Μπεληγιάννης, 2015)

Γ. Διάγραμμα Δραστηριοτήτων για σύστημα διαχείρισης μαθημάτων για καθηγητές

Περιλαμβάνονται οι παρακάτω δραστηριότητες:

- Λήψη των φοιτητών του μαθήματος
- Για κάθε φοιτητή απαιτούνται τα παρακάτω:
 - Παράλληλη εκτέλεση των λειτουργιών «Λήψη απουσιών», «Λήψη βαθμών Εργασιών», Λήψη «Βαθμού Εξετάσεων»
 - Υπολογίζεται ο τελικός βαθμός
 - Ενημερώνεται η τελική βαθμολογία για κάθε φοιτητή (Μπεληγιάννης, 2015)



Εικ. 4-17 Σύστημα Διαχείρισης μαθημάτων για καθηγητές (Μπεληγιάννης, 2015)

4.4 Σημασιολογία Διαγραμμάτων Κατάστασης (State Machine Diagrams)

Εισαγωγή

Τα Διαγράμματα Κατάστασης (State Machine Diagrams) αποτελούν ένα από τα βασικά εργαλεία της UML για τη μοντελοποίηση της δυναμικής συμπεριφοράς συστημάτων. Περιγράφουν τις καταστάσεις στις οποίες μπορεί να βρεθεί ένα σύστημα, καθώς και τις μεταβάσεις μεταξύ αυτών, ως αποτέλεσμα γεγονότων και συνθηκών.

Η σημασιολογία τους αφορά την ακριβή ερμηνεία των στοιχείων τους, όπως καταστάσεις, μεταβάσεις, γεγονότα και ενέργειες. Σε αντίθεση με πιο αφηρημένα διαγράμματα, τα State Machine Diagrams έχουν ισχυρή θεωρητική βάση που συνδέεται με τα πεπερασμένα αυτόματα (finite state machines) (Booch, Rumbaugh, & Jacobson, 2005· Sommerville, 2007).

Ρόλος στη UML

Τα διαγράμματα κατάστασης χρησιμοποιούνται για:

- Μοντελοποίηση της συμπεριφοράς αντικειμένων
- Περιγραφή κύκλου ζωής ενός συστήματος
- Ανάλυση αντιδραστικών (reactive) συστημάτων
- Μοντελοποίηση event-driven εφαρμογών

Σε αντίθεση με τα Activity Diagrams, που εστιάζουν στη ροή εργασιών, τα State Machine Diagrams εστιάζουν στις καταστάσεις και στις μεταβάσεις που προκαλούνται από γεγονότα.

Βασικά Στοιχεία και Σημασιολογία

Καταστάσεις (States)

Μια κατάσταση αναπαριστά μια συγκεκριμένη συνθήκη στην οποία βρίσκεται ένα σύστημα σε μια χρονική στιγμή. Μπορεί να περιλαμβάνει:

- entry actions (ενέργειες εισόδου)
- do activities (ενέργειες κατά τη διάρκεια)
- exit actions (ενέργειες εξόδου)

Μεταβάσεις (Transitions)

Οι μεταβάσεις συνδέουν καταστάσεις και ενεργοποιούνται από γεγονότα. Μια μετάβαση μπορεί να περιλαμβάνει:

- γεγονός (event)

- συνθήκη (guard condition)
- ενέργεια (action)

Γεγονότα (Events)

Τα γεγονότα ενεργοποιούν μεταβάσεις και μπορεί να είναι:

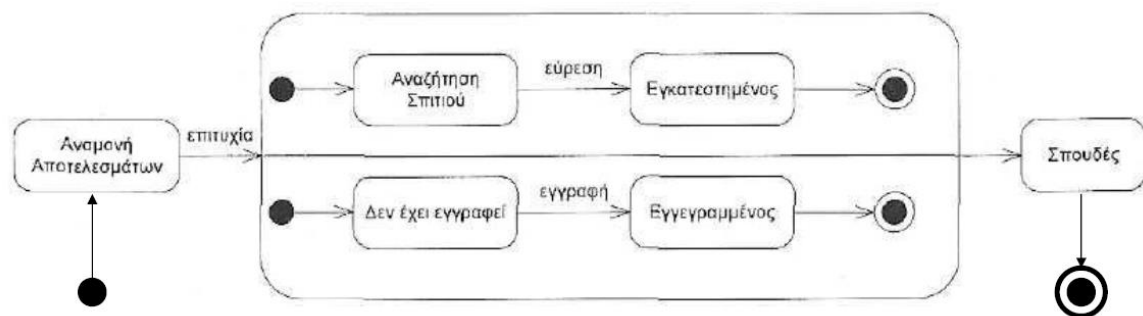
- εξωτερικά (π.χ. input χρήστη)
- χρονικά (time events)
- σήματα (signals)

Αρχική και Τελική Κατάσταση

Αρχική κατάσταση: σημείο εκκίνησης

Τελική κατάσταση: τερματισμός της εκτέλεσης

Παράδειγμα Διαγράμματος Κατάστασης



Εικ. 4-18 Διάγραμμα με ταυτόχρονες καταστάσεις για έναν υποψήφιο πανελληνίων εξετάσεων

Σημασιολογία Εκτέλεσης

Η σημασιολογία των state machines βασίζεται στην έννοια:

- της ενεργής κατάστασης

- της επεξεργασίας γεγονότων
- της αλλαγής κατάστασης (state transition)

Η εκτέλεση είναι event-driven: ένα γεγονός οδηγεί σε μετάβαση μόνο εάν πληρούνται οι συνθήκες.

Σύνθετες Καταστάσεις (Composite States)

Οι σύνθετες καταστάσεις επιτρέπουν ιεραρχική μοντελοποίηση:

- περιέχουν υποκαταστάσεις
- μειώνουν την πολυπλοκότητα
- επιτρέπουν επαναχρησιμοποίηση μοντέλων

Παράλληλες Καταστάσεις (Orthogonal Regions)

Η UML υποστηρίζει παράλληλη εκτέλεση μέσω orthogonal regions:

- πολλαπλές καταστάσεις ενεργές ταυτόχρονα
- συγχρονισμός μέσω γεγονότων

Τυπική (Formal) Σημασιολογία

Η συμπεριφορά ενός State Machine μπορεί να οριστεί ως:

$$SM = (S, E, T, s_0, F)$$

όπου:

- S : σύνολο καταστάσεων
- E : σύνολο γεγονότων
- $T \subseteq S \times E \times S$: σύνολο μεταβάσεων
- s_0 : αρχική κατάσταση

- F : σύνολο τελικών καταστάσεων

Συνάρτηση Μετάβασης

Η μετάβαση ορίζεται ως:

$$\delta: S \times E \rightarrow S$$

Η συνάρτηση δ καθορίζει σε ποια κατάσταση θα μεταβεί το σύστημα δεδομένης μιας κατάστασης και ενός γεγονότος.

Εκτέλεση Συστήματος

Η εκτέλεση μπορεί να περιγραφεί ως ακολουθία:

$$s_0 \xrightarrow{e_1} s_1 \xrightarrow{e_2} s_2 \xrightarrow{e_3} \dots$$

Guard Conditions

Οι μεταβάσεις ενεργοποιούνται μόνο αν:

$$g(s, e) = true$$

Πίνακας 4-1. Σύγκριση με Activity Diagrams

Χαρακτηριστικό	Activity Diagrams	State Machines
Εστίαση	Ροή εργασιών	Καταστάσεις
Εκτέλεση	Flow-based	Event-driven
Παραλληλία	Fork/Join	Orthogonal states

Συμπεράσματα

Τα Διαγράμματα Κατάστασης αποτελούν ένα ισχυρό εργαλείο για τη μοντελοποίηση της συμπεριφοράς συστημάτων με έμφαση σε γεγονότα και μεταβάσεις. Η σημασιολογία τους, ιδιαίτερα όταν εκφράζεται τυπικά, επιτρέπει την αυστηρή ανάλυση και επαλήθευση συστημάτων. Η χρήση τους είναι κρίσιμη σε reactive και real-time συστήματα, όπου η σωστή διαχείριση καταστάσεων και γεγονότων είναι απαραίτητη.

4.5 Συνηθισμένα λάθη και ελλειψεις στη σημασιολογική ερμηνεία

Η Ενοποιημένη Γλώσσα Μοντελοποίησης (UML) αποτελεί ένα από τα πιο διαδεδομένα πρότυπα για την οπτική αναπαράσταση συστημάτων λογισμικού. Παρά τη δημοτικότητα της, η σημασιολογική της ερμηνεία συχνά παρουσιάζει ασάφειες και οδηγεί σε παρερμηνείες, ιδιαίτερα όταν χρησιμοποιείται από ομάδες με διαφορετικό επίπεδο εμπειρίας. Στην ενότητα αυτή αναλύονται τα πιο συνηθισμένα λάθη και οι ελλείψεις που εντοπίζονται στη σημασιολογική κατανόηση της UML.

Ασάφεια στη σημασιολογία των διαγραμμάτων

Ένα από τα βασικότερα προβλήματα της UML είναι ότι, ενώ παρέχει σαφείς συντακτικούς κανόνες, η σημασιολογία πολλών στοιχείων δεν είναι πλήρως τυποποιημένη. Αυτό οδηγεί σε διαφορετικές ερμηνείες του ίδιου διαγράμματος από διαφορετικούς αναλυτές (Booch et al., 2005). Για παράδειγμα, ένα διάγραμμα κλάσεων μπορεί να αποδίδει είτε λογική είτε φυσική αρχιτεκτονική, χωρίς αυτό να είναι πάντα ξεκάθαρο.

Σύγχυση μεταξύ εννοιολογικού και υλοποιητικού επιπέδου

Συχνά παρατηρείται σύγχυση μεταξύ του εννοιολογικού (conceptual) και του υλοποιητικού (implementation) επιπέδου μοντελοποίησης. Οι σχεδιαστές ενσωματώνουν λεπτομέρειες υλοποίησης σε πρώιμα στάδια ανάλυσης, γεγονός που περιορίζει την αφαιρετικότητα του μοντέλου και δυσχεραίνει την επαναχρησιμοποίηση (Fowler, 2004).

Εσφαλμένη χρήση σχέσεων μεταξύ στοιχείων

Οι σχέσεις όπως η κληρονομικότητα, η συσχέτιση και η σύνθεση χρησιμοποιούνται συχνά λανθασμένα. Ένα συνηθισμένο λάθος είναι η χρήση της σύνθεσης (composition) αντί της απλής συσχέτισης, χωρίς να πληρούνται τα αυστηρά κριτήρια ιδιοκτησίας και κύκλου ζωής (Rumbaugh et al., 1999). Αυτό οδηγεί σε λανθασμένη κατανόηση της δομής του συστήματος.

Υπερβολική ή ανεπαρκής λεπτομέρεια

Η υπερβολική λεπτομέρεια μπορεί να καταστήσει τα διαγράμματα δυσνόητα, ενώ η ανεπαρκής λεπτομέρεια οδηγεί σε ασαφή μοντέλα. Η ισορροπία μεταξύ των δύο είναι

κρίσιμη, αλλά συχνά παραβλέπεται, ιδιαίτερα σε μεγάλα έργα λογισμικού (Eriksson & Penker, 2000).

Έλλειψη συνέπειας μεταξύ διαφορετικών διαγραμμάτων

Η UML περιλαμβάνει πολλαπλούς τύπους διαγραμμάτων (π.χ. διαγράμματα κλάσεων, ακολουθίας, δραστηριοτήτων). Συχνά, παρατηρείται έλλειψη συνέπειας μεταξύ αυτών, με αποτέλεσμα αντιφάσεις στη σημασιολογική ερμηνεία. Για παράδειγμα, μια ακολουθία μηνυμάτων σε διάγραμμα ακολουθίας μπορεί να μην αντικατοπτρίζεται σωστά στις μεθόδους των κλάσεων (Larman, 2005).

Παρερμηνεία της δυναμικής συμπεριφοράς

Τα διαγράμματα συμπεριφοράς, όπως τα διαγράμματα καταστάσεων και ακολουθίας, συχνά παρερμηνεύονται λόγω της πολυπλοκότητάς τους. Η μη σωστή κατανόηση των χρονικών περιορισμών ή των μεταβάσεων καταστάσεων μπορεί να οδηγήσει σε σφάλματα σχεδιασμού (Harel & Rumpfe, 2004).

Έλλειψη τυποποίησης στη χρήση συμβολισμών

Παρότι η UML είναι τυποποιημένη, η χρήση της στην πράξη διαφέρει σημαντικά. Οργανισμοί και ομάδες ανάπτυξης συχνά υιοθετούν δικές τους «παραλλαγές», γεγονός που δημιουργεί ασυνέπειες και δυσκολεύει την επικοινωνία (Object Management Group, 2017).

Περιορισμοί στην εκφραστικότητα

Η UML, αν και ισχυρή, δεν μπορεί να εκφράσει πλήρως όλες τις σημασιολογικές πτυχές ενός συστήματος. Για παράδειγμα, πολύπλοκοι επιχειρησιακοί κανόνες ή περιορισμοί συχνά δεν αποτυπώνονται επαρκώς χωρίς τη χρήση συμπληρωματικών γλωσσών, όπως η OCL (Object Constraint Language) (Warmer & Kleppe, 2003).

Αντιμετώπιση Σημασιολογικών Σφαλμάτων μέσω Τυπικών Προσεγγίσεων

Τα σημασιολογικά προβλήματα που παρουσιάστηκαν στην προηγούμενη ενότητα δεν αποτελούν απλώς πρακτικές δυσκολίες μοντελοποίησης αλλά συνδέονται άμεσα με την έλλειψη αυστηρού ορισμού της σημασίας των UML μοντέλων. Η βιβλιογραφία επισημαίνει

ότι ένα σημαντικό ποσοστό σφαλμάτων στις προδιαγραφές λογισμικού προκύπτει από διαφορετικές ερμηνείες των ίδιων μοντελοποιητικών στοιχείων (France & Rumpe, 2007).

Η ύπαρξη τυπικής σημασιολογίας επιτρέπει τη σαφή αντιστοίχιση των γραφικών στοιχείων της UML σε μαθηματικά ή λογικά μοντέλα, μειώνοντας την αμφισημία και διευκολύνοντας την αυτοματοποιημένη επαλήθευση (Object Management Group, 2017). Στο Κεφάλαιο 3 παρουσιάστηκαν διάφορες τυπικές σημασιολογικές προσεγγίσεις, όπως η Object Constraint Language (OCL), τα Petri Nets, η operational semantics και οι προσεγγίσεις model checking. Οι προσεγγίσεις αυτές μπορούν να χρησιμοποιηθούν ως μηχανισμοί πρόληψης και ανίχνευσης των λαθών που συναντώνται συχνότερα στα UML μοντέλα.

Πίνακας 4-2 Συσχέτιση Σημασιολογικών Σφαλμάτων και Τυπικών Προσεγγίσεων

Σημασιολογικό πρόβλημα	Επιπτώσεις	Προτεινόμενη τυπική προσέγγιση
Ασάφεια στη σημασιολογία διαγραμμάτων	Διαφορετικές ερμηνείες μοντέλων	Formal Semantics
Σύγχυση εννοιολογικού και υλοποιητικού επιπέδου	Μειωμένη αφαιρετικότητα	Meta-modeling
Ασυνέπεια μεταξύ διαγραμμάτων	Αντιφατικές προδιαγραφές	Consistency Checking
Παρερμηνεία δυναμικής συμπεριφοράς	Λανθασμένη εκτέλεση διαδικασιών	Petri Nets
Μη προσβάσιμες καταστάσεις	Ελλιπής συμπεριφορά συστήματος	State Verification
Περιορισμένη εκφραστικότητα UML	Ελλιπής περιγραφή κανόνων	OCL
Λανθασμένη χρήση πληθικότητων	Παραβίαση επιχειρησιακών κανόνων	OCL Constraints

Η χρήση της OCL θεωρείται ιδιαίτερα αποτελεσματική για τη διατύπωση περιορισμών που δεν μπορούν να εκφραστούν αποκλειστικά με τα γραφικά στοιχεία της UML. Μέσω προ-συνθηκών, μετα-συνθηκών και αμετάβλητων συνθηκών (invariants), καθίσταται δυνατή η ακριβής περιγραφή επιχειρησιακών κανόνων και η ανίχνευση ασυνεπειών σε διαγράμματα κλάσεων (Warmer & Kleppe, 2003).

Αντίστοιχα, τα Petri Nets χρησιμοποιούνται εκτενώς για τη σημασιολογική θεμελίωση των Activity Diagrams, καθώς επιτρέπουν την ανάλυση ιδιοτήτων όπως η προσβασιμότητα, ο

συγχρονισμός, η παράλληλη εκτέλεση και η ανίχνευση αδιεξόδων (deadlocks) (Harel & Rumpe, 2004). Η δυνατότητα αυτή είναι ιδιαίτερα σημαντική σε συστήματα με πολύπλοκες ροές εργασιών. Στην περίπτωση των State Machine Diagrams, οι προσεγγίσεις operational semantics και model checking επιτρέπουν την επαλήθευση μεταβάσεων, τη διαπίστωση μη προσβάσιμων καταστάσεων και τον εντοπισμό λογικών ασυνεπειών πριν από την υλοποίηση του συστήματος (Störrle, 2021).

Παρά τα πλεονεκτήματα των τυπικών προσεγγίσεων, η εφαρμογή τους στην πράξη παραμένει περιορισμένη λόγω της αυξημένης πολυπλοκότητας και της ανάγκης εξειδικευμένων γνώσεων. Όπως επισημαίνουν οι Brambilla, Cabot και Wimmer (2017), οι περισσότερες ομάδες ανάπτυξης εξακολουθούν να χρησιμοποιούν τη UML σε ημι-τυπικό επίπεδο. Ωστόσο, σε κρίσιμα συστήματα, όπου η αξιοπιστία αποτελεί βασική απαίτηση, οι τυπικές σημασιολογικές τεχνικές θεωρούνται απαραίτητες για τη διασφάλιση της ποιότητας και της ορθότητας των μοντέλων (Sommerville, 2016).

Συνεπώς, η αντιμετώπιση των σημασιολογικών λαθών δεν εξαρτάται αποκλειστικά από την εμπειρία του αναλυτή αλλά και από τον βαθμό τυπικότητας που ενσωματώνεται στη διαδικασία μοντελοποίησης. Η σύνδεση της UML με τυπικές σημασιολογικές τεχνικές συμβάλλει ουσιαστικά στη βελτίωση της συνέπειας, της πληρότητας και της αξιοπιστίας των μοντέλων λογισμικού.

Συμπεράσματα

Η σημασιολογική ερμηνεία της UML αποτελεί κρίσιμο παράγοντα για την επιτυχή μοντελοποίηση συστημάτων. Τα συνηθισμένα λάθη που παρουσιάστηκαν αναδεικνύουν την ανάγκη για καλύτερη εκπαίδευση, αυστηρότερη τυποποίηση και προσεκτική χρήση των διαγραμμάτων. Η κατανόηση των περιορισμών της UML μπορεί να συμβάλει στη δημιουργία πιο σαφών και αποτελεσματικών μοντέλων.

5. UML Activity Diagrams: Σημασιολογία και εφαρμογές

5.1 Ο ρόλος των Activity Diagrams στη μοντελοποίηση συστημάτων

Τα Activity Diagrams της UML αποτελούν ένα από τα βασικότερα εργαλεία για τη μοντελοποίηση της δυναμικής συμπεριφοράς ενός συστήματος. Χρησιμοποιούνται ευρέως για την αναπαράσταση ροών εργασίας (workflows), επιχειρησιακών διαδικασιών και αλληλεπιδράσεων μεταξύ διαφορετικών λειτουργικών μονάδων. Σε αντίθεση με τα διαγράμματα κλάσεων που επικεντρώνονται στη στατική δομή, τα Activity Diagrams εστιάζουν στη ροή ελέγχου και δεδομένων μέσα σε ένα σύστημα (Object Management Group, 2017).

Βασικά χαρακτηριστικά και έννοιες

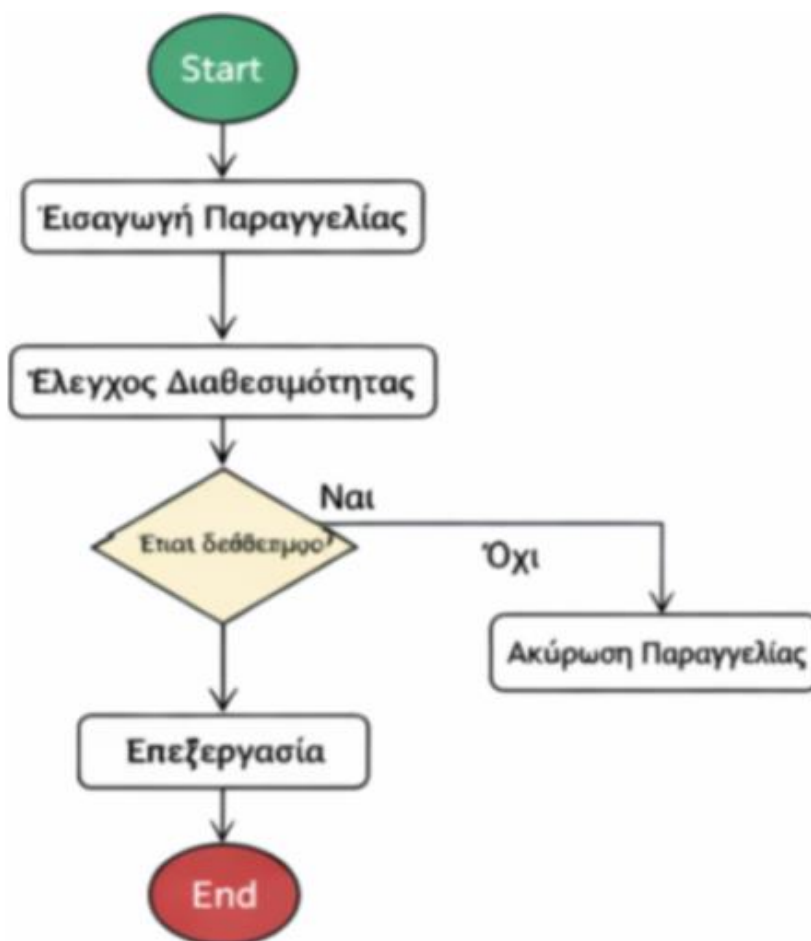
Τα Activity Diagrams βασίζονται σε έννοιες όπως οι δραστηριότητες (activities), οι ενέργειες (actions), οι ροές ελέγχου (control flows) και οι ροές αντικειμένων (object flows). Επιπλέον, περιλαμβάνουν κόμβους απόφασης (decision nodes), συγχρονισμού (fork/join) και αρχής/τέλους, οι οποίοι επιτρέπουν την αναπαράσταση πολύπλοκων λογικών ροών (Booch et al., 2005).

Ένα σημαντικό χαρακτηριστικό είναι η δυνατότητα αναπαράστασης παράλληλων διεργασιών, μέσω των κόμβων fork και join. Αυτό καθιστά τα Activity Diagrams ιδιαίτερα χρήσιμα για τη μοντελοποίηση συστημάτων όπου εκτελούνται ταυτόχρονες λειτουργίες (Harel & Rumpe, 2004).

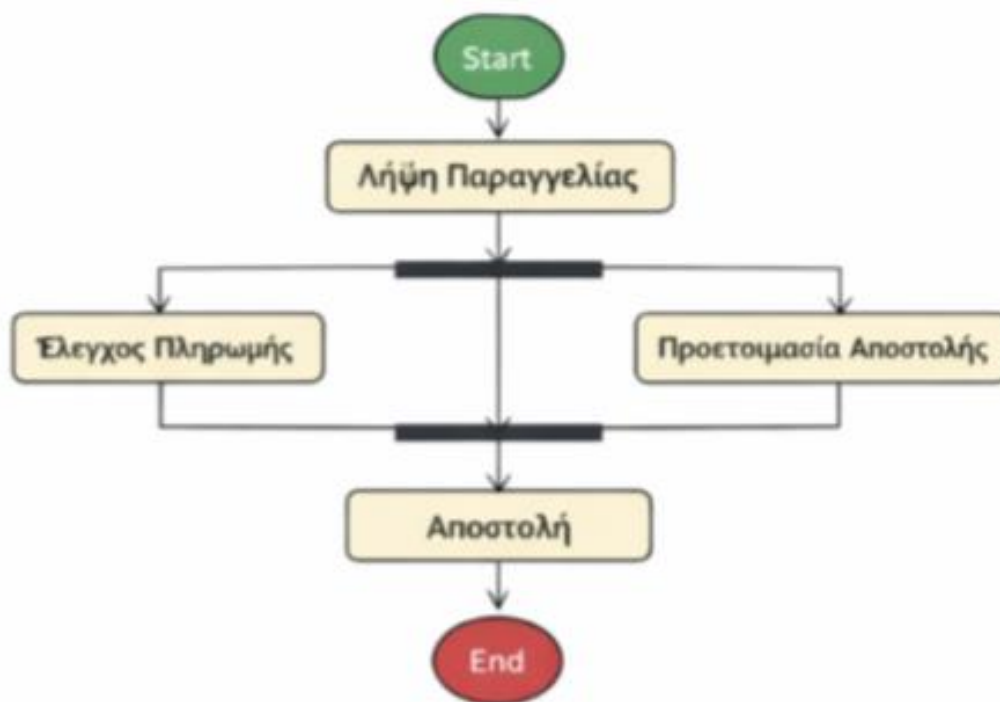
Χρήση στη μοντελοποίηση επιχειρησιακών διαδικασιών

Τα Activity Diagrams χρησιμοποιούνται εκτενώς στη μοντελοποίηση επιχειρησιακών διαδικασιών (business process modeling). Επιτρέπουν την οπτική αναπαράσταση της αλληλουχίας ενεργειών που απαιτούνται για την ολοκλήρωση μιας διαδικασίας, καθώς και την αναγνώριση σημείων απόφασης και πιθανών βελτιώσεων (Eriksson & Penker, 2000).

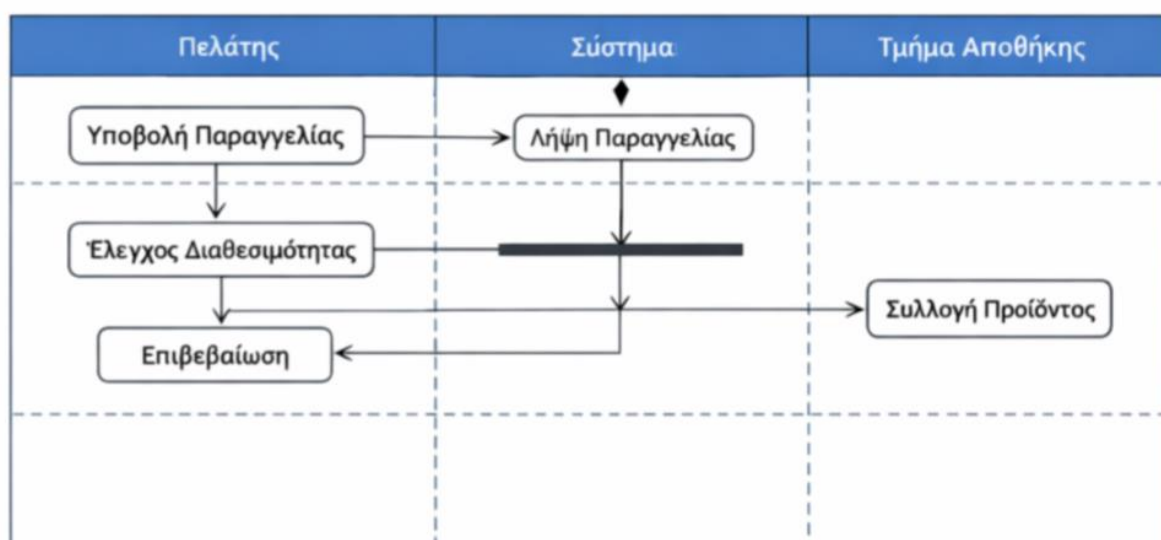
Η χρήση τους συμβάλλει στην καλύτερη κατανόηση των απαιτήσεων από τους εμπλεκόμενους φορείς, ακόμη και όταν αυτοί δεν διαθέτουν τεχνικό υπόβαθρο. Έτσι, λειτουργούν ως γέφυρα επικοινωνίας μεταξύ αναλυτών, προγραμματιστών και επιχειρησιακών στελεχών (Fowler, 2004).



Εικ. 5.1 Διαδικασία Παραγγελίας Προϊόντος



Εικ. 5.2 Παράλληλη Εκτέλεση Διεργασιών



Εικ. 5-3 Activity Diagram με Swimlanes

Υποστήριξη της ανάλυσης και του σχεδιασμού συστημάτων

Κατά τη φάση της ανάλυσης, τα Activity Diagrams βοηθούν στον εντοπισμό βασικών λειτουργιών και σεναρίων χρήσης. Στη φάση του σχεδιασμού, μπορούν να χρησιμοποιηθούν για τη λεπτομερή αποτύπωση της λογικής εκτέλεσης των διαδικασιών και την κατανομή αρμοδιοτήτων μεταξύ υποσυστημάτων (Larman, 2005).

Επιπλέον, μέσω της χρήσης swimlanes, είναι δυνατή η απεικόνιση της κατανομής ευθυνών μεταξύ διαφορετικών ρόλων ή οργανωτικών μονάδων, ενισχύοντας την κατανόηση της αρχιτεκτονικής του συστήματος.

Σχέση με άλλα διαγράμματα UML

Τα Activity Diagrams δεν λειτουργούν απομονωμένα, αλλά συμπληρώνουν άλλα διαγράμματα της UML. Για παράδειγμα, μπορούν να χρησιμοποιηθούν σε συνδυασμό με Use Case Diagrams για την ανάλυση σεναρίων χρήσης, καθώς και με Sequence Diagrams για τη λεπτομερή αποτύπωση αλληλεπιδράσεων (Booch et al., 2005).

Ενώ τα Sequence Diagrams επικεντρώνονται στη χρονική σειρά ανταλλαγής μηνυμάτων, τα Activity Diagrams δίνουν έμφαση στη ροή εργασίας, προσφέροντας μια πιο αφαιρετική και συνολική εικόνα της διαδικασίας.

Πλεονεκτήματα και περιορισμοί

Τα βασικά πλεονεκτήματα των Activity Diagrams περιλαμβάνουν:

- Την απλότητα και ευκολία κατανόησης
- Την ικανότητα αναπαράστασης σύνθετων ροών εργασίας
- Τη δυνατότητα μοντελοποίησης παράλληλων διεργασιών

Ωστόσο, παρουσιάζουν και ορισμένους περιορισμούς. Σε ιδιαίτερα πολύπλοκα συστήματα, τα διαγράμματα μπορεί να γίνουν δυσανάγνωστα και δύσκολα στη διαχείριση. Επιπλέον, η έλλειψη αυστηρής σημασιολογίας σε ορισμένες περιπτώσεις μπορεί να οδηγήσει σε διαφορετικές ερμηνείες από διαφορετικούς χρήστες (Harel & Rumpe, 2004).

Συμβολή στην τεκμηρίωση και επαναχρησιμοποίηση

Τα Activity Diagrams συμβάλλουν σημαντικά στην τεκμηρίωση του συστήματος, παρέχοντας σαφή απεικόνιση των διαδικασιών. Αυτό διευκολύνει τη συντήρηση και την εξέλιξη του συστήματος, καθώς και την επαναχρησιμοποίηση μοντέλων σε παρόμοια έργα (Eriksson & Penker, 2000).

Συμπεράσματα

Τα Activity Diagrams αποτελούν ένα ισχυρό εργαλείο για τη μοντελοποίηση της συμπεριφοράς συστημάτων και επιχειρησιακών διαδικασιών. Η ικανότητά τους να αναπαριστούν τόσο σειριακές όσο και παράλληλες ροές τα καθιστά ιδιαίτερα χρήσιμα σε όλα τα στάδια ανάπτυξης λογισμικού. Παρά τους περιορισμούς τους, η σωστή χρήση τους μπορεί να βελτιώσει σημαντικά την κατανόηση, την επικοινωνία και την ποιότητα του σχεδιασμού ενός συστήματος.

5.2 Σημασιολογικές δομές στα Activity Diagrams

Τα Activity Diagrams αποτελούν ένα από τα βασικά διαγραμματικά εργαλεία της UML (Unified Modeling Language) για την αναπαράσταση της δυναμικής συμπεριφοράς ενός συστήματος. Εστιάζουν κυρίως στη ροή ελέγχου και δεδομένων μεταξύ δραστηριοτήτων, επιτρέποντας την περιγραφή πολύπλοκων επιχειρησιακών διαδικασιών και αλγοριθμικών ροών. Η σημασιολογική τους δομή είναι κρίσιμη για την κατανόηση και τη σωστή ερμηνεία των μοντέλων, καθώς καθορίζει το πώς εκτελούνται οι δραστηριότητες και πώς μεταφέρονται τα δεδομένα μέσα στο σύστημα (Object Management Group, 2017).

Βασικές έννοιες σημασιολογίας

Η σημασιολογία των Activity Diagrams βασίζεται στην έννοια των tokens, τα οποία χρησιμοποιούνται για να αναπαραστήσουν τη ροή εκτέλεσης. Τα tokens κινούνται κατά μήκος των ακμών (edges) του διαγράμματος και ενεργοποιούν τις δραστηριότητες (actions) όταν πληρούνται συγκεκριμένες συνθήκες. Υπάρχουν δύο βασικοί τύποι tokens: τα control tokens και τα object tokens. Τα control tokens καθορίζουν τη ροή ελέγχου, ενώ τα object tokens μεταφέρουν δεδομένα μεταξύ ενεργειών (Rumbaugh et al., 2005).

Η εκτέλεση ενός Activity Diagram ξεκινά από έναν αρχικό κόμβο (initial node) και προχωρά μέσω των ενεργειών και των κόμβων ελέγχου, μέχρι να φτάσει σε έναν ή περισσότερους τελικούς κόμβους (final nodes). Η σημασιολογία της εκτέλεσης είναι μη-ντετερμινιστική σε ορισμένες περιπτώσεις, επιτρέποντας παράλληλη εκτέλεση και εναλλακτικές ροές (Eriksson & Penker, 2000).

Κόμβοι ελέγχου και σημασιολογική συμπεριφορά

Οι κόμβοι ελέγχου (control nodes) παίζουν καθοριστικό ρόλο στη σημασιολογική δομή των Activity Diagrams. Οι βασικοί τύποι περιλαμβάνουν:

- Decision nodes: Κατευθύνουν τη ροή σε μία από πολλές εναλλακτικές διαδρομές, βάσει συνθηκών (guards). Η σημασιολογία τους επιβάλλει ότι μόνο μία εξερχόμενη ακμή μπορεί να ενεργοποιηθεί ανά εκτέλεση.
- Merge nodes: Συγχωνεύουν εναλλακτικές ροές χωρίς συγχρονισμό.
- Fork nodes: Δημιουργούν παράλληλες ροές εκτέλεσης, διανέμοντας tokens σε πολλαπλές ακμές.
- Join nodes: Συγχρονίζουν παράλληλες ροές, απαιτώντας την άφιξη tokens από όλες τις εισερχόμενες ακμές πριν συνεχιστεί η εκτέλεση.

Η σημασιολογία των fork και join κόμβων είναι ιδιαίτερα σημαντική για την κατανόηση της ταυτόχρονης εκτέλεσης (concurrency), καθώς καθορίζει τον τρόπο συγχρονισμού και αλληλεπίδρασης μεταξύ παράλληλων διεργασιών (Booch et al., 2007).

Ροές δεδομένων και αντικειμένων

Πέρα από τη ροή ελέγχου, τα Activity Diagrams υποστηρίζουν και ροές δεδομένων μέσω object nodes και object flows. Τα object nodes λειτουργούν ως αποθηκευτικοί χώροι ή σημεία μεταφοράς δεδομένων, ενώ τα object flows μεταφέρουν αντικείμενα μεταξύ ενεργειών.

Η σημασιολογία των object flows βασίζεται στην έννοια της κατανάλωσης και παραγωγής tokens: μια ενέργεια μπορεί να εκτελεστεί μόνο όταν όλα τα απαραίτητα εισερχόμενα object tokens είναι διαθέσιμα. Αυτό επιτρέπει την ακριβή μοντελοποίηση εξαρτήσεων δεδομένων και περιορισμών εκτέλεσης (Dumas et al., 2018).

Ιεραρχία και σύνθετες δραστηριότητες

Τα Activity Diagrams υποστηρίζουν ιεραρχική μοντελοποίηση μέσω της χρήσης σύνθετων δραστηριοτήτων (structured activities). Αυτές επιτρέπουν την οργάνωση πολύπλοκων διαδικασιών σε υπο-δραστηριότητες, διευκολύνοντας την κατανόηση και επαναχρησιμοποίηση του μοντέλου.

Η σημασιολογία των σύνθετων δραστηριοτήτων βασίζεται στην έννοια της ενθυλάκωσης (encapsulation), όπου η εσωτερική δομή μιας δραστηριότητας δεν είναι άμεσα ορατή στο εξωτερικό επίπεδο. Ωστόσο, η εκτέλεση εξακολουθεί να ακολουθεί τους ίδιους κανόνες ροής tokens (Object Management Group, 2017).

Σημασιολογικές προεκτάσεις και εκτελεσιμότητα

Σε πιο προχωρημένες εφαρμογές, τα Activity Diagrams μπορούν να χρησιμοποιηθούν ως εκτελέσιμα μοντέλα (executable models), ιδιαίτερα σε περιβάλλοντα Model-Driven Engineering (MDE). Σε αυτές τις περιπτώσεις, η σημασιολογία τους πρέπει να είναι πλήρως καθορισμένη και τυπική, ώστε να επιτρέπεται η αυτόματη μετατροπή σε κώδικα ή η προσομοίωση της εκτέλεσης (Mellor et al., 2004).

Η τυπική σημασιολογία των Activity Diagrams έχει μελετηθεί εκτενώς και συχνά συνδέεται με μαθηματικά μοντέλα, όπως τα Petri Nets, τα οποία προσφέρουν αυστηρό πλαίσιο για την ανάλυση ιδιοτήτων όπως η συγχρονισμένη εκτέλεση και η αποφυγή αδιεξόδων (Peterson, 1981).

5.3 Χρήση των Activity Diagrams για μοντελοποίηση πειραματικών διαδικασιών σε εικονικά εργαστήρια

Τα εικονικά εργαστήρια (virtual laboratories) αποτελούν ένα σύγχρονο εργαλείο εκπαίδευσης και έρευνας, επιτρέποντας την προσομοίωση πειραματικών διαδικασιών σε ένα ψηφιακό περιβάλλον. Η αναπαράσταση των διαδικασιών αυτών απαιτεί σαφή και δομημένη μοντελοποίηση, ώστε να διασφαλίζεται η κατανόηση, η επαναληψιμότητα και η ορθότητα των πειραμάτων. Στο πλαίσιο αυτό, τα Activity Diagrams της UML προσφέρουν

ένα ισχυρό μέσο για τη μοντελοποίηση των ροών ενεργειών και των αλληλεπιδράσεων που λαμβάνουν χώρα σε ένα εικονικό εργαστήριο (Object Management Group, 2017).

Αναπαράσταση πειραματικών ροών

Τα Activity Diagrams είναι ιδιαίτερα κατάλληλα για την απεικόνιση πειραματικών διαδικασιών, καθώς επιτρέπουν τη γραφική αποτύπωση της ακολουθίας βημάτων που απαιτούνται για την εκτέλεση ενός πειράματος. Κάθε ενέργεια (action) μπορεί να αντιστοιχεί σε ένα στάδιο του πειράματος, όπως η προετοιμασία εξοπλισμού, η εκτέλεση μετρήσεων ή η καταγραφή αποτελεσμάτων.

Η χρήση κόμβων ελέγχου, όπως οι decision και fork nodes, επιτρέπει τη μοντελοποίηση εναλλακτικών σεναρίων και παράλληλων διαδικασιών. Για παράδειγμα, σε ένα εικονικό εργαστήριο φυσικής, η ταυτόχρονη συλλογή δεδομένων από διαφορετικούς αισθητήρες μπορεί να αναπαρασταθεί μέσω παράλληλων ροών (Ma & Nickerson, 2006).

Διαχείριση δεδομένων και αλληλεπιδράσεων

Στα εικονικά εργαστήρια, η διαχείριση δεδομένων αποτελεί κρίσιμο στοιχείο της πειραματικής διαδικασίας. Τα Activity Diagrams υποστηρίζουν τη ροή δεδομένων μέσω object flows, επιτρέποντας την αναπαράσταση της μεταφοράς και επεξεργασίας δεδομένων μεταξύ των επιμέρους δραστηριοτήτων.

Για παράδειγμα, τα δεδομένα που συλλέγονται από ένα πείραμα μπορούν να περάσουν από στάδια φιλτραρίσματος, ανάλυσης και αποθήκευσης. Η σημασιολογία των object tokens εξασφαλίζει ότι κάθε ενέργεια εκτελείται μόνο όταν τα απαραίτητα δεδομένα είναι διαθέσιμα, ενισχύοντας τη συνέπεια και την ακρίβεια της προσομοίωσης (Dumas et al., 2018).

Υποστήριξη επαναληψιμότητας και ελέγχου ροής

Ένα από τα βασικά χαρακτηριστικά των επιστημονικών πειραμάτων είναι η επαναληψιμότητα. Τα Activity Diagrams επιτρέπουν τη μοντελοποίηση επαναληπτικών διαδικασιών μέσω βρόχων (loops) και συνθηκών ελέγχου, διασφαλίζοντας ότι τα πειράματα μπορούν να εκτελούνται πολλαπλές φορές υπό διαφορετικές παραμέτρους.

Επιπλέον, η χρήση guards σε decision nodes επιτρέπει τον έλεγχο της ροής βάσει συγκεκριμένων κριτηρίων, όπως η εγκυρότητα των δεδομένων ή η επίτευξη επιθυμητών αποτελεσμάτων. Αυτό είναι ιδιαίτερα χρήσιμο σε εκπαιδευτικά περιβάλλοντα, όπου οι χρήστες μπορούν να εξερευνήσουν διαφορετικά σενάρια εκτέλεσης (Sommerville, 2016).

Μοντελοποίηση αλληλεπίδρασης χρήστη

Στα εικονικά εργαστήρια, η αλληλεπίδραση του χρήστη με το σύστημα αποτελεί βασικό στοιχείο. Τα Activity Diagrams μπορούν να ενσωματώσουν ενέργειες που σχετίζονται με τον χρήστη, όπως η εισαγωγή δεδομένων, η επιλογή παραμέτρων και η ενεργοποίηση διαδικασιών.

Η χρήση swimlanes (διαχωριστικών περιοχών) επιτρέπει την οργάνωση των δραστηριοτήτων ανάλογα με τον ρόλο (π.χ. χρήστης, σύστημα, εξομοιωτής), διευκολύνοντας την κατανόηση της κατανομής ευθυνών και της αλληλεπίδρασης μεταξύ των επιμέρους οντοτήτων (Booch et al., 2007).

Χρήση των Activity Diagrams για Μοντελοποίηση Πειραματικών Διαδικασιών σε Εικονικά Εργαστήρια: Κριτική Προσέγγιση

Τα Activity Diagrams χρησιμοποιούνται ευρέως για την αναπαράσταση εκπαιδευτικών και πειραματικών διαδικασιών σε εικονικά εργαστήρια, καθώς παρέχουν έναν σαφή και διαισθητικό τρόπο περιγραφής της ακολουθίας ενεργειών, των σημείων απόφασης και των παράλληλων δραστηριοτήτων. Ωστόσο, η χρήση τους δεν είναι απαλλαγμένη από περιορισμούς, ιδιαίτερα όταν απαιτείται αυστηρή σημασιολογική ερμηνεία ή εκτελέσιμη μοντελοποίηση (Dumas et al., 2018).

- **Παράδειγμα από Εικονικό Εργαστήριο Φυσικής**

Ένα χαρακτηριστικό παράδειγμα αποτελεί το πείραμα της Άνωσης στο περιβάλλον Klett Book-App και Publior Virtual Lab. Η διαδικασία περιλαμβάνει:

1. Εκκίνηση του εικονικού εργαστηρίου.
2. Επιλογή πειράματος.
3. Μέτρηση βάρους σώματος στον αέρα.
4. Βύθιση σώματος στο νερό.

5. Μέτρηση φαινόμενου βάρους.
6. Υπολογισμό άνωσης.
7. Σύγκριση με το βάρος του εκτοπισμένου νερού.
8. Εξαγωγή συμπερασμάτων.

Η συγκεκριμένη διαδικασία μοντελοποιείται αποτελεσματικά μέσω Activity Diagram, καθώς περιλαμβάνει σαφή ακολουθία ενεργειών και συγκεκριμένα σημεία ελέγχου. Παράλληλα, οι ροές δεδομένων μπορούν να αναπαραστήσουν τη μεταφορά των μετρήσεων μεταξύ των διαφορετικών σταδίων της πειραματικής διαδικασίας.

- **Παράδειγμα από Εικονικό Εργαστήριο Χημείας**

Σε ένα εικονικό πείραμα ογκομέτρησης, ο μαθητής:

- επιλέγει αντιδραστήρια,
- πραγματοποιεί βαθμονόμηση οργάνων,
- εκτελεί την προσθήκη αντιδραστηρίου,
- παρακολουθεί τη μεταβολή χρώματος,
- καταγράφει το τελικό σημείο ισοδυναμίας.

Η διαδικασία περιλαμβάνει επαναληπτικούς βρόχους και πολλαπλά σημεία ελέγχου εγκυρότητας. Τα Activity Diagrams μπορούν να αποτυπώσουν τη λογική ακολουθία των ενεργειών, αλλά δυσκολεύονται να αναπαραστήσουν με ακρίβεια χρονικούς περιορισμούς και γεγονότα που ενεργοποιούνται από εξωτερικές συνθήκες.

- **Παράδειγμα από Εικονικό Εργαστήριο Βιολογίας**

Σε ένα πείραμα μικροσκοπικής παρατήρησης κυττάρων, οι μαθητές:

- προετοιμάζουν το δείγμα,
- επιλέγουν μεγέθυνση,
- παρατηρούν το παρασκεύασμα,
- καταγράφουν παρατηρήσεις,
- συγκρίνουν τα αποτελέσματα με πρότυπα δείγματα.

Η ύπαρξη πολλαπλών χρηστών και ταυτόχρονων δραστηριοτήτων καθιστά απαραίτητη τη χρήση Swimlanes και παράλληλων ροών. Ωστόσο, η πολυπλοκότητα αυξάνεται σημαντικά

όταν επιχειρείται η αναπαράσταση συνεργατικών δραστηριοτήτων μεταξύ πολλών χρηστών.

Σύγκριση Activity Diagrams και BPMN

Παρόλο που τα Activity Diagrams χρησιμοποιούνται συχνά για τη μοντελοποίηση εκπαιδευτικών διαδικασιών, δεν αποτελούν τη μοναδική διαθέσιμη επιλογή. Η Business Process Model and Notation (BPMN) αποτελεί μια εναλλακτική προσέγγιση η οποία έχει σχεδιαστεί ειδικά για τη μοντελοποίηση διαδικασιών.

Πίνακας 5-1 Σύγκριση Activity Diagrams και BPMN

Κριτήριο	Activity Diagram	BPMN
Ευκολία κατανόησης	Υψηλή	Μέτρια
Σύνδεση με UML	Άμεση	Περιορισμένη
Εκτελεσιμότητα	Περιορισμένη	Υψηλότερη
Αναπαράσταση επιχειρησιακών διαδικασιών	Καλή	Πολύ καλή
Χρονικά γεγονότα	Περιορισμένα	Ισχυρή υποστήριξη
Εκπαιδευτικά σενάρια	Πολύ κατάλληλα	Κατάλληλα

Η BPMN προσφέρει πλουσιότερους μηχανισμούς για γεγονότα, εξαιρέσεις και χρονικούς περιορισμούς, ενώ τα Activity Diagrams διατηρούν το πλεονέκτημα της στενής ενσωμάτωσής τους στο οικοσύστημα της UML (Silver, 2020).

Εκτελέσιμη Σημασιολογία μέσω Petri Nets: Κριτική Αξιολόγηση

Η βιβλιογραφία προτείνει τη μετατροπή των Activity Diagrams σε Petri Nets ώστε να επιτευχθεί αυστηρότερη σημασιολογική θεμελίωση και δυνατότητα τυπικής επαλήθευσης (Störrle, 2021).

Η προσέγγιση αυτή επιτρέπει:

- ανίχνευση deadlocks,
- έλεγχο προσβασιμότητας καταστάσεων,
- επαλήθευση συγχρονισμού παράλληλων διεργασιών,
- μαθηματική ανάλυση της συμπεριφοράς.

Ωστόσο, η χρήση Petri Nets συνοδεύεται από σημαντικούς περιορισμούς. Η μετάβαση από ένα Activity Diagram σε ένα ισοδύναμο Petri Net αυξάνει σημαντικά την πολυπλοκότητα του μοντέλου, ενώ απαιτεί εξειδικευμένες γνώσεις μαθηματικής μοντελοποίησης. Επιπλέον, η αναγνωσιμότητα μειώνεται σημαντικά για εκπαιδευτικούς και αναλυτές που δεν διαθέτουν σχετική εξειδίκευση.

Συνεπώς, η εκτελέσιμη σημασιολογία μέσω Petri Nets είναι ιδιαίτερα χρήσιμη σε κρίσιμα συστήματα υψηλής αξιοπιστίας, αλλά δεν αποτελεί πάντοτε την καταλληλότερη λύση για εκπαιδευτικά περιβάλλοντα ή για αρχικά στάδια ανάλυσης απαιτήσεων.

Παράδειγμα Σημασιολογικής Ασάφειας και Αποτυχίας Υλοποίησης

Ένα συνηθισμένο πρόβλημα εμφανίζεται σε Activity Diagrams όπου χρησιμοποιείται κόμβος απόφασης χωρίς αμοιβαία αποκλειόμενες συνθήκες (guards).

Σε ένα εικονικό εργαστήριο φυσικής, η δραστηριότητα:

«Έλεγχος εγκυρότητας μετρήσεων»

μπορεί να ακολουθείται από δύο εξερχόμενες ροές:

- Μετρήσεις αποδεκτές
- Επανάληψη πειράματος

Εάν οι συνθήκες δεν είναι πλήρως καθορισμένες, διαφορετικά εργαλεία ή διαφορετικοί αναλυτές ενδέχεται να ερμηνεύσουν διαφορετικά τη ροή εκτέλεσης. Ως αποτέλεσμα, η προσομοίωση μπορεί να οδηγηθεί είτε σε αδιέξοδο είτε σε λανθασμένη επανάληψη της διαδικασίας.

Το παράδειγμα αυτό αναδεικνύει ότι η συντακτική ορθότητα ενός Activity Diagram δεν διασφαλίζει απαραίτητα και τη σημασιολογική του ορθότητα. Η ύπαρξη τυπικής

σημασιολογίας ή η χρήση τεχνικών επαλήθευσης μπορεί να συμβάλει σημαντικά στον εντοπισμό τέτοιων προβλημάτων πριν από την υλοποίηση.

Συμπεράσματα

Η χρήση των Activity Diagrams σε εικονικά εργαστήρια φυσικών επιστημών προσφέρει σημαντικά πλεονεκτήματα ως προς την αναπαράσταση πειραματικών διαδικασιών, τη διαχείριση δεδομένων και την υποστήριξη εκπαιδευτικών δραστηριοτήτων. Ωστόσο, η αποτελεσματικότητά τους εξαρτάται από τη σημασιολογική ακρίβεια των μοντέλων και από τον βαθμό τυπικότητας που απαιτεί η εκάστοτε εφαρμογή. Η σύγκριση με τη BPMN και η κριτική αξιολόγηση της χρήσης Petri Nets καταδεικνύουν ότι δεν υπάρχει μία καθολικά βέλτιστη λύση· η επιλογή της κατάλληλης προσέγγισης εξαρτάται από τους στόχους, την πολυπλοκότητα και τις απαιτήσεις του συστήματος.

6. Από τη μοντελοποίηση στην υλοποίηση: Μετατροπή UML σε εκτελέσιμο κώδικα

6.1 Model-Driven Engineering (MDE) και σημασιολογία

Η Model-Driven Engineering (MDE) αποτελεί μια σύγχρονη προσέγγιση στην ανάπτυξη λογισμικού, η οποία δίνει έμφαση στη χρήση μοντέλων ως βασικών στοιχείων καθ' όλη τη διάρκεια του κύκλου ζωής ενός συστήματος. Αντί η ανάπτυξη να επικεντρώνεται αποκλειστικά στον κώδικα, η MDE προωθεί τη δημιουργία αφηρημένων μοντέλων που περιγράφουν τη δομή, τη συμπεριφορά και τις λειτουργικές απαιτήσεις του συστήματος. Στο πλαίσιο αυτό, η σημασιολογία των μοντέλων αποκτά ιδιαίτερη σημασία, καθώς καθορίζει την ακριβή ερμηνεία και εκτέλεση των μοντέλων (Schmidt, 2006).

Βασικές αρχές της MDE

Η MDE βασίζεται σε τρεις θεμελιώδεις έννοιες: τα μοντέλα (models), τα μετα-μοντέλα (metamodels) και τους μετασχηματισμούς (model transformations). Τα μοντέλα αποτελούν αφαιρετικές αναπαραστάσεις ενός συστήματος, ενώ τα μετα-μοντέλα καθορίζουν τη γλώσσα και τους κανόνες με τους οποίους κατασκευάζονται τα μοντέλα. Οι μετασχηματισμοί επιτρέπουν τη μετατροπή ενός μοντέλου σε άλλο, συχνά από υψηλότερο επίπεδο αφαίρεσης σε χαμηλότερο (π.χ. από μοντέλο σε κώδικα) (Kleppe et al., 2003).

Η επιτυχία της MDE εξαρτάται σε μεγάλο βαθμό από την ύπαρξη σαφούς και αυστηρά ορισμένης σημασιολογίας. Χωρίς σαφή σημασιολογία, τα μοντέλα μπορεί να ερμηνευθούν διαφορετικά από διαφορετικά εργαλεία ή χρήστες, οδηγώντας σε ασυνέπειες και σφάλματα (Seidewitz, 2003).

Έννοια της σημασιολογίας στα μοντέλα

Η σημασιολογία αναφέρεται στο νόημα των στοιχείων ενός μοντέλου και στον τρόπο με τον οποίο αυτά ερμηνεύονται και εκτελούνται. Στο πλαίσιο της MDE, διακρίνονται τρεις βασικοί τύποι σημασιολογίας:

- **Συντακτική σημασιολογία (syntactic semantics):** Καθορίζει τη δομή και τους κανόνες σύνταξης των μοντέλων, όπως ορίζονται από το μετα-μοντέλο.

- **Σημασιολογία εκτέλεσης (execution semantics):** Περιγράφει τη δυναμική συμπεριφορά των μοντέλων κατά την εκτέλεσή τους.
- **Τυπική σημασιολογία (formal semantics):** Βασίζεται σε μαθηματικά μοντέλα για την ακριβή και μη αμφίσημη περιγραφή της συμπεριφοράς (Harel & Rumpe, 2004).

Η ύπαρξη τυπικής σημασιολογίας είναι ιδιαίτερα σημαντική σε κρίσιμα συστήματα, όπου απαιτείται υψηλός βαθμός αξιοπιστίας και επαλήθευσης.

Σημασιολογία και μετασχηματισμοί μοντέλων

Οι μετασχηματισμοί μοντέλων αποτελούν βασικό μηχανισμό της MDE και εξαρτώνται άμεσα από τη σημασιολογία των εμπλεκόμενων μοντέλων. Ένας μετασχηματισμός πρέπει να διατηρεί τη σημασιολογική συνέπεια, δηλαδή το νόημα του αρχικού μοντέλου να παραμένει ισοδύναμο στο παραγόμενο μοντέλο.

Για παράδειγμα, κατά τη μετατροπή ενός UML Activity Diagram σε εκτελέσιμο κώδικα, η σημασιολογία της ροής ελέγχου και δεδομένων πρέπει να διατηρείται, ώστε το τελικό σύστημα να συμπεριφέρεται όπως περιγράφεται στο αρχικό μοντέλο (Mellor et al., 2004).

Η πρόκληση έγκειται στο γεγονός ότι διαφορετικές γλώσσες μοντελοποίησης μπορεί να έχουν διαφορετικά σημασιολογικά θεμέλια, γεγονός που καθιστά τη διαλειτουργικότητα (interoperability) ένα σημαντικό ζήτημα (Sendall & Kozaczynski, 2003).

Εκτελέσιμα μοντέλα και σημασιολογική ακρίβεια

Η έννοια των εκτελέσιμων μοντέλων (executable models) αποτελεί βασικό στόχο της MDE. Σε αυτή την περίπτωση, τα μοντέλα δεν χρησιμοποιούνται μόνο για τεκμηρίωση, αλλά και για την άμεση εκτέλεση ή προσομοίωση της συμπεριφοράς ενός συστήματος.

Για να καταστεί αυτό εφικτό, απαιτείται πλήρως καθορισμένη σημασιολογία, η οποία να επιτρέπει την αυτόματη ερμηνεία από εργαλεία λογισμικού. Η χρήση τυπικών μεθόδων, όπως οι καταστάσεις (state machines) ή τα Petri nets, συμβάλλει στην επίτευξη αυτής της ακρίβειας (Schmidt, 2006).

Η σημασιολογική σαφήνεια επιτρέπει επίσης την επαλήθευση και επικύρωση των μοντέλων μέσω τεχνικών όπως η προσομοίωση, ο έλεγχος ιδιοτήτων και η ανάλυση συμπεριφοράς.

Πλεονεκτήματα και προκλήσεις

Η ενσωμάτωση της σημασιολογίας στη MDE προσφέρει σημαντικά πλεονεκτήματα, όπως:

- Βελτιωμένη κατανόηση και επικοινωνία μεταξύ των εμπλεκομένων.
- Δυνατότητα αυτοματοποίησης μέσω μετασχηματισμών.
- Υποστήριξη ανάλυσης και επαλήθευσης μοντέλων.

Ωστόσο, υπάρχουν και προκλήσεις, όπως η πολυπλοκότητα στον ορισμό τυπικής σημασιολογίας και η ανάγκη για εξειδικευμένα εργαλεία και γνώσεις. Επιπλέον, η ετερογένεια των γλωσσών μοντελοποίησης καθιστά δύσκολη την ενοποίηση και τη διαλειτουργικότητα μεταξύ διαφορετικών συστημάτων (Harel & Rumpe, 2004).

6.2 Τεχνικές αυτόματης παραγωγής κώδικα

Η μετατροπή μοντέλων UML σε εκτελέσιμο κώδικα αποτελεί βασικό στόχο της Model-Driven Engineering (MDE), καθώς επιδιώκει τη γεφύρωση του χάσματος μεταξύ αφηρημένου σχεδιασμού και υλοποίησης. Στο σύγχρονο πλαίσιο ανάπτυξης λογισμικού, η αυτόματη παραγωγή κώδικα βασίζεται σε τυπικά ορισμένα μοντέλα και σημασιολογικά συνεπείς μετασχηματισμούς, επιτρέποντας την αύξηση της παραγωγικότητας και τη μείωση των σφαλμάτων (Schmidt, 2006· Di Ruscio et al., 2021).

Από τα μοντέλα στον κώδικα

Η διαδικασία μετατροπής UML σε εκτελέσιμο κώδικα ακολουθεί συνήθως την αρχιτεκτονική Model-Driven Architecture (MDA), η οποία περιλαμβάνει τη μετάβαση από Platform-Independent Models (PIM) σε Platform-Specific Models (PSM) και τελικά σε πηγαίο κώδικα. Τα UML διαγράμματα χρησιμοποιούνται για την περιγραφή της δομής και της συμπεριφοράς του συστήματος, ενώ η σημασιολογία τους καθορίζει τον τρόπο αντιστοίχισης σε στοιχεία προγραμματισμού (Kleppe et al., 2003).

Σύγχρονες προσεγγίσεις επεκτείνουν αυτή τη διαδικασία ενσωματώνοντας τεχνικές αυτοματοποίησης που επιτρέπουν τη δημιουργία πλήρως λειτουργικών εφαρμογών

απευθείας από μοντέλα, ιδιαίτερα σε περιβάλλοντα low-code και model-driven πλατφορμών (Di Ruscio et al., 2021).

Τεχνικές μετασχηματισμού μοντέλων

Οι τεχνικές αυτόματης παραγωγής κώδικα βασίζονται σε μετασχηματισμούς μοντέλων, οι οποίοι διακρίνονται σε:

- Model-to-Model (M2M): Μετατροπή μοντέλων μεταξύ διαφορετικών επιπέδων αφαίρεσης.
- Model-to-Text (M2T): Παραγωγή πηγαίου κώδικα από μοντέλα.

Οι M2T μετασχηματισμοί υλοποιούνται μέσω προτύπων (templates), τα οποία καθορίζουν τη σύνδεση μεταξύ στοιχείων UML και συντακτικών στοιχείων γλωσσών προγραμματισμού. Σύγχρονες μελέτες δείχνουν ότι η χρήση επαναχρησιμοποιήσιμων προτύπων βελτιώνει σημαντικά την ποιότητα και τη συντηρησιμότητα του παραγόμενου κώδικα (García-Magariño et al., 2020).

Επιπλέον, νέες προσεγγίσεις αξιοποιούν τη MDE για την αυτόματη παραγωγή εξειδικευμένου κώδικα, όπως γραφικές διεπαφές χρήστη (GUI) και εφαρμογές κινητών συσκευών, ενισχύοντας τη χρήση της σε σύγχρονα περιβάλλοντα ανάπτυξης (Usman et al., 2024).

Αντιστοίχιση UML στοιχείων σε κώδικα

Η αντιστοίχιση UML μοντέλων σε κώδικα περιλαμβάνει τη μετατροπή:

- Class Diagrams → Κλάσεις και δομές δεδομένων
- Activity Diagrams → Δομές ελέγχου (if, loops, concurrency)
- State Machines → Διαχείριση καταστάσεων
- Η ορθότητα αυτής της διαδικασίας εξαρτάται από τη σημασιολογική σαφήνεια των μοντέλων. Σε σύγχρονες εφαρμογές, η αντιστοίχιση επεκτείνεται ώστε να καλύπτει και πιο σύνθετες έννοιες, όπως η ενσωμάτωση επιχειρησιακής λογικής και η παραγωγή κώδικα για κατανεμημένα συστήματα (Lano & Siala, 2024).

Παράλληλα, η χρήση της MDE σε συνδυασμό με τεχνολογίες όπως η μηχανική μάθηση οδηγεί σε νέα μοντέλα παραγωγής κώδικα, όπου τα μοντέλα χρησιμοποιούνται για τη δημιουργία εξειδικευμένων εφαρμογών σε τομείς όπως η ανάλυση δεδομένων και τα ευφυή συστήματα (Rädler et al., 2024).

Εκτελέσιμη UML και σύγχρονα εργαλεία

Η έννοια της Εκτελέσιμης UML (Executable UML – xUML) επιτρέπει την άμεση εκτέλεση μοντέλων χωρίς την ανάγκη χειροκίνητης υλοποίησης. Σήμερα, αυτή η προσέγγιση ενισχύεται από σύγχρονα εργαλεία και frameworks που υποστηρίζουν πλήρως αυτοματοποιημένες διαδικασίες ανάπτυξης.

Πλαίσια όπως το MontiGem αποδεικνύουν ότι η MDE μπορεί να εφαρμοστεί επιτυχώς σε πολλαπλά πεδία εφαρμογών, προσφέροντας επαναχρησιμοποίηση και υψηλό επίπεδο αυτοματοποίησης (MontiGem Study, 2024). Παράλληλα, διεθνή συνέδρια όπως το MODELS 2024 παρουσιάζουν τις τελευταίες εξελίξεις στον τομέα, συμπεριλαμβανομένων τεχνικών για την ενίσχυση της εκτελεσιμότητας των μοντέλων (Wimmer et al., 2024).

Επιπλέον, πρόσφατες ερευνητικές τάσεις συνδυάζουν τη MDE με τεχνητή νοημοσύνη, αξιοποιώντας μεγάλα γλωσσικά μοντέλα (LLMs) για την αυτόματη παραγωγή κώδικα από UML διαγράμματα, ανοίγοντας νέες προοπτικές στην αυτοματοποίηση της ανάπτυξης (Bates et al., 2025).

Παράδειγμα Μετασχηματισμού Activity Diagram σε Java

Για να καταδειχθεί η πρακτική εφαρμογή της MDE, παρουσιάζεται ένα απλό παράδειγμα μετασχηματισμού ενός UML Activity Diagram σε κώδικα Java.

Το Activity Diagram περιγράφει τη διαδικασία ελέγχου εγκυρότητας μετρήσεων σε ένα εικονικό εργαστήριο φυσικής:

1. Έναρξη πειράματος.
2. Εισαγωγή μετρήσεων.
3. Έλεγχος εγκυρότητας.
4. Απόφαση:

✓ Αν οι μετρήσεις είναι έγκυρες → Υπολογισμός αποτελεσμάτων.

✓ Αν οι μετρήσεις δεν είναι έγκυρες → Επανάληψη πειράματος.

5. Τερματισμός.

Η σημασιολογία του Activity Diagram αντιστοιχίζεται σε δομές ελέγχου της Java ως εξής:

Πίνακας 6-1 Activity Diagram σε Java

UML Activity Element	Java Construct
Action	Method call
Decision Node	if / else
Loop	while
Initial Node	main() entry point
Final Node	return

Παράδειγμα κώδικα:

```
public class Experiment {  
  
    public static void main(String[] args) {  
  
        boolean validMeasurements = checkMeasurements();  
  
        if(validMeasurements){  
            calculateResults();  
        }  
        else{  
            repeatExperiment();  
        }  
    }  
}
```

```
static boolean checkMeasurements(){  
    return true;  
}  
  
static void calculateResults(){  
    System.out.println("Results calculated");  
}  
  
static void repeatExperiment(){  
    System.out.println("Experiment repeated");  
}  
}
```

Το παράδειγμα αυτό δείχνει ότι η μετατροπή ενός Activity Diagram σε κώδικα δεν αποτελεί απλή συντακτική αντιστοίχιση. Η σωστή παραγωγή κώδικα προϋποθέτει ότι οι σημασιολογικές έννοιες του μοντέλου – όπως οι συνθήκες απόφασης, οι βρόχοι και οι μεταβάσεις – έχουν οριστεί με σαφή και μη αμφίσημο τρόπο. Διαφορετικά, διαφορετικά εργαλεία μετασχηματισμού μπορεί να παράγουν διαφορετικές υλοποιήσεις για το ίδιο μοντέλο (Brambilla, Cabot & Wimmer, 2017).

Πλεονεκτήματα και περιορισμοί

Η αυτόματη παραγωγή κώδικα από UML παρουσιάζει σημαντικά πλεονεκτήματα:

- Μείωση ανθρώπινων σφαλμάτων
- Επιτάχυνση της ανάπτυξης
- Διατήρηση συνέπειας μεταξύ μοντέλων και υλοποίησης
- Διευκόλυνση συντήρησης και επαναχρησιμοποίησης

Ωστόσο, εξακολουθούν να υπάρχουν προκλήσεις:

- Η ποιότητα του παραγόμενου κώδικα εξαρτάται από την ποιότητα των μοντέλων
- Η ανάγκη για εξειδικευμένα εργαλεία και τεχνογνωσία
- Περιορισμοί στην πλήρη αυτοματοποίηση πολύπλοκων συστημάτων

Επιπλέον, αν και οι σύγχρονες τεχνολογίες, όπως τα LLMs, βελτιώνουν σημαντικά την παραγωγή κώδικα, εξακολουθούν να απαιτούνται μηχανισμοί επαλήθευσης για τη διασφάλιση της ορθότητας και της σημασιολογικής συνέπειας (Sadik et al., 2024).

6.3 Ο ρόλος της σημασιολογικής συνέπειας στη μετατροπή μοντέλων

Η μετατροπή UML μοντέλων σε εκτελέσιμο κώδικα αποτελεί μια σύνθετη διαδικασία, η οποία απαιτεί όχι μόνο συντακτική αντιστοίχιση αλλά και διατήρηση της σημασιολογικής συνέπειας (semantic consistency). Η σημασιολογική συνέπεια διασφαλίζει ότι το νόημα και η συμπεριφορά που περιγράφονται στα μοντέλα παραμένουν αμετάβλητα κατά τη μετάβαση σε εκτελέσιμο κώδικα, αποφεύγοντας λειτουργικές αποκλίσεις και ασάφειες (Seidewitz, 2003· Schmidt, 2006).

Έννοια της σημασιολογικής συνέπειας

Η σημασιολογική συνέπεια αναφέρεται στη διατήρηση του νοήματος μεταξύ διαφορετικών αναπαραστάσεων ενός συστήματος, είτε αυτές είναι μοντέλα είτε υλοποιήσεις. Σε αντίθεση με τη συντακτική συνέπεια, η οποία αφορά τη συμμόρφωση με τους κανόνες μιας γλώσσας μοντελοποίησης, η σημασιολογική συνέπεια σχετίζεται με την ορθότητα της ερμηνείας των μοντέλων (Seidewitz, 2003).

Σύμφωνα με πιο πρόσφατες μελέτες, η συνέπεια μπορεί να διακριθεί σε οριζόντια και κατακόρυφη. Η οριζόντια αφορά τη συμφωνία μεταξύ διαφορετικών UML διαγραμμάτων, ενώ η κατακόρυφη αφορά τη συμφωνία μεταξύ επιπέδων αφαίρεσης, όπως από μοντέλα σε κώδικα (Engels et al., 2023). Η κατακόρυφη συνέπεια είναι ιδιαίτερα κρίσιμη στη MDE, καθώς η υλοποίηση πρέπει να αντικατοπτρίζει πιστά τη συμπεριφορά που περιγράφεται στο μοντέλο.

Σημασιολογία και μετασχηματισμοί μοντέλων

Οι μετασχηματισμοί μοντέλων αποτελούν τον βασικό μηχανισμό μετατροπής UML σε εκτελέσιμο κώδικα. Για να είναι αξιόπιστοι, πρέπει να διατηρούν τη σημασιολογική ισοδυναμία μεταξύ του αρχικού και του παραγόμενου μοντέλου (Sendall & Kozaczynski, 2003).

Η σημασιολογική συνέπεια διασφαλίζει ότι οι μετασχηματισμοί δεν αλλοιώνουν τη λογική του συστήματος. Για παράδειγμα, κατά τη μετατροπή ενός Activity Diagram σε κώδικα, η ροή ελέγχου και οι εξαρτήσεις δεδομένων πρέπει να διατηρούνται ακριβώς όπως ορίζονται στο μοντέλο (Kleppe et al., 2003).

Επιπλέον, σύγχρονες προσεγγίσεις εισάγουν την έννοια των transformation contracts, τα οποία καθορίζουν τυπικά τις σχέσεις μεταξύ εισόδου και εξόδου ενός μετασχηματισμού, επιτρέποντας τον έλεγχο της σημασιολογικής ορθότητας (García-Magariño et al., 2020).

Σημασιολογική ενοποίηση πολλαπλών διαγραμμάτων UML

Η UML περιλαμβάνει πολλαπλά διαγράμματα που περιγράφουν διαφορετικές πτυχές ενός συστήματος, γεγονός που δημιουργεί προκλήσεις στη διατήρηση της συνολικής σημασιολογικής συνέπειας. Για παράδειγμα, ένα Class Diagram περιγράφει τη δομή, ενώ ένα Activity Diagram τη συμπεριφορά.

Η ενοποίηση αυτών των διαγραμμάτων απαιτεί κοινή σημασιολογική βάση, ώστε να αποφεύγονται αντιφάσεις. Τυπικές προσεγγίσεις, όπως η χρήση δυναμικής λογικής και μαθηματικών μοντέλων, έχουν προταθεί για την ενοποίηση της σημασιολογίας (O’Keefe, 2002).

Επιπλέον, η διαχείριση ασυνεπειών μεταξύ μοντέλων αποτελεί κρίσιμο ζήτημα στη μηχανική λογισμικού, καθώς οι ασυνέπειες μπορεί να οδηγήσουν σε σφάλματα κατά την παραγωγή κώδικα (Spanoudakis & Zisman, 2001).

Σύγχρονες προσεγγίσεις και προκλήσεις

Σύγχρονες εξελίξεις στον τομέα της MDE δίνουν έμφαση στη σημασιολογική συνέπεια ως βασική προϋπόθεση για την αυτοματοποίηση της παραγωγής κώδικα. Η ανάπτυξη low-code πλατφορμών βασίζεται σε μεγάλο βαθμό σε αξιόπιστους σημασιολογικούς μετασχηματισμούς (Di Ruscio et al., 2021).

Παράλληλα, η ενσωμάτωση τεχνητής νοημοσύνης και μεγάλων γλωσσικών μοντέλων (LLMs) στη διαδικασία παραγωγής κώδικα δημιουργεί νέες προκλήσεις. Αν και τα μοντέλα αυτά μπορούν να παράγουν κώδικα από UML διαγράμματα, η διασφάλιση της σημασιολογικής ακρίβειας παραμένει κρίσιμο ζήτημα (Bates et al., 2025).

Επιπλέον, η διαλειτουργικότητα μεταξύ διαφορετικών μοντελοποιητικών γλωσσών απαιτεί προηγμένες τεχνικές σημασιολογικού μετασχηματισμού, ώστε να διατηρείται το νόημα κατά τη μεταφορά μεταξύ διαφορετικών περιβαλλόντων (Lano & Siala, 2024).

Σημασία για την ποιότητα λογισμικού

Η σημασιολογική συνέπεια επηρεάζει άμεσα την ποιότητα του παραγόμενου λογισμικού. Η έλλειψή της μπορεί να οδηγήσει σε ασυνέπειες μεταξύ σχεδίασης και υλοποίησης, καθώς και σε σφάλματα που είναι δύσκολο να εντοπιστούν (Schmidt, 2006).

Αντίθετα, η διατήρηση της σημασιολογικής συνέπειας επιτρέπει την αξιόπιστη χρήση μοντέλων ως βάση για την αυτόματη παραγωγή κώδικα και την επαλήθευση της ορθότητας του συστήματος. Μελέτες δείχνουν ότι η συστηματική διαχείριση συνέπειας μεταξύ UML διαγραμμάτων συμβάλλει σημαντικά στη μείωση σφαλμάτων και στη βελτίωση της συντηρησιμότητας του λογισμικού (García-Magariño et al., 2020).

6.4 Μελέτες περίπτωσης από τη βιβλιογραφία

Η σύγχρονη βιβλιογραφία παρουσιάζει σημαντική εξέλιξη στη μετατροπή UML μοντέλων σε εκτελέσιμο κώδικα, με έμφαση σε πραγματικές εφαρμογές και νέες τεχνολογικές κατευθύνσεις. Οι μελέτες περίπτωσης καταδεικνύουν ότι η Model-Driven Engineering (MDE) αποτελεί πλέον ώριμη προσέγγιση, η οποία εφαρμόζεται σε ποικίλους τομείς, από βιομηχανικά συστήματα έως εφαρμογές τεχνητής νοημοσύνης (Di Ruscio et al., 2021).

Μελέτες περίπτωσης σε βιομηχανικά και σύνθετα συστήματα

Η χρήση της MDE σε βιομηχανικά και μηχανικά συστήματα αποτελεί σημαντικό πεδίο εφαρμογής. Στη μελέτη των Koltun και Pundel (2022), παρουσιάζονται περιπτώσεις μετασχηματισμού μοντέλων σε εκτελέσιμα συστήματα με χρήση του εργαλείου VIATRA. Τα αποτελέσματα δείχνουν ότι η μοντελοκεντρική προσέγγιση συμβάλλει στη μείωση της πολυπλοκότητας και στη βελτίωση της συνεργασίας μεταξύ διαφορετικών ομάδων ανάπτυξης.

Παράλληλα, πιο πρόσφατες μελέτες δείχνουν ότι η UML μπορεί να αξιοποιηθεί ακόμη και σε προηγμένα υπολογιστικά περιβάλλοντα, όπως υβριδικά και κβαντικά συστήματα, επιβεβαιώνοντας τη διεύρυνση των εφαρμογών της MDE (Lano et al., 2024).

Μελέτες περίπτωσης σε web και mobile εφαρμογές

Η ανάπτυξη web και mobile εφαρμογών αποτελεί έναν από τους πιο δυναμικούς τομείς εφαρμογής της αυτόματης παραγωγής κώδικα. Στη μελέτη των Muthumanikandan et al. (2024), παρουσιάζεται το σύστημα Screen2Web, το οποίο συνδυάζει UML με τεχνικές μηχανικής μάθησης για την παραγωγή HTML κώδικα. Τα αποτελέσματα δείχνουν σημαντική μείωση του χρόνου ανάπτυξης και βελτίωση της αποδοτικότητας.

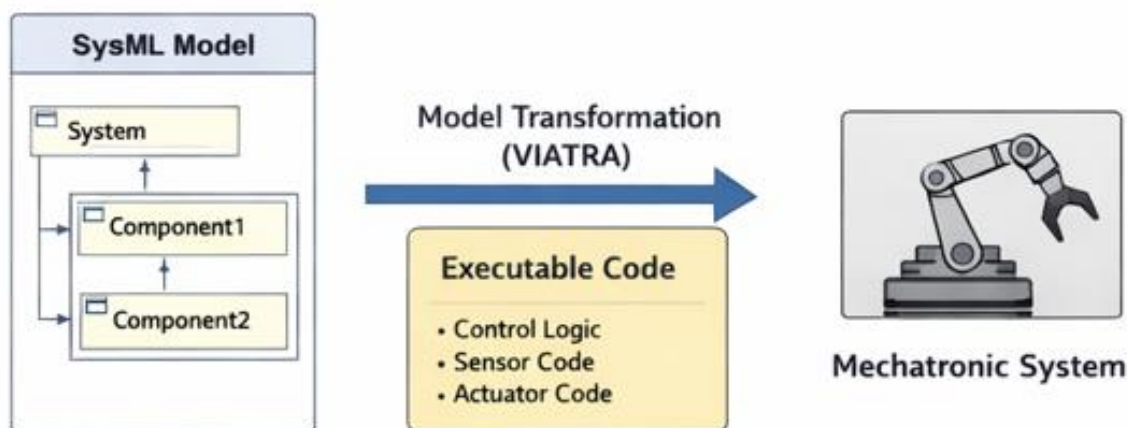
Αντίστοιχα, η μελέτη των Usman et al. (2024) για το σύστημα MOBICAT καταδεικνύει ότι η χρήση UML μοντέλων μπορεί να αυτοματοποιήσει τη δημιουργία γραφικών διεπαφών για εφαρμογές Android. Η προσέγγιση αυτή μειώνει τα σφάλματα και επιταχύνει τη διαδικασία ανάπτυξης.

Μελέτες περίπτωσης σε μετατροπή και επαναχρησιμοποίηση λογισμικού

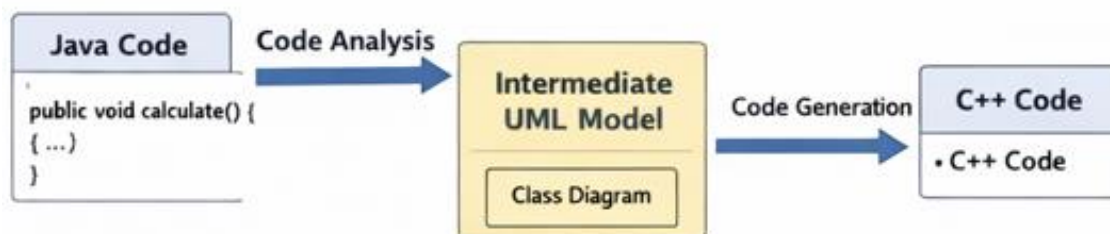
Η μετατροπή λογισμικού μεταξύ διαφορετικών γλωσσών αποτελεί ένα σημαντικό πεδίο εφαρμογής της MDE. Οι Lano και Siala (2024) παρουσιάζουν μια προσέγγιση όπου τα μοντέλα χρησιμοποιούνται ως ενδιάμεσο επίπεδο αναπαράστασης για την αυτόματη μετάφραση προγραμμάτων. Τα αποτελέσματα δείχνουν ότι η προσέγγιση αυτή διευκολύνει τη διαλειτουργικότητα και μειώνει την πολυπλοκότητα της διαδικασίας.

Επιπλέον, η επαναχρησιμοποίηση legacy συστημάτων αποτελεί ένα κρίσιμο ζήτημα στη σύγχρονη ανάπτυξη λογισμικού. Σύμφωνα με τους Lano et al. (2024), η MDE μπορεί να

υποστηρίζει τη σταδιακή ανακατασκευή παλαιών συστημάτων, μειώνοντας το κόστος και τον χρόνο συντήρησης.



Εικ. 6-1 Μετατροπή κώδικα μέσω ενδιάμεσου μοντέλου



Εικ. 6-2 Μετατροπή κώδικα μέσω ενδιάμεσου μοντέλου

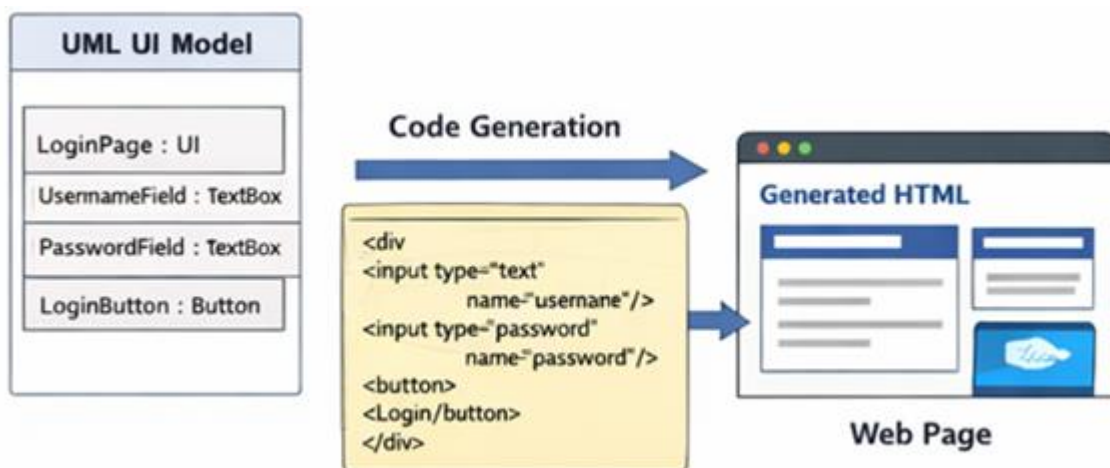
Μελέτες περίπτωσης με χρήση τεχνητής νοημοσύνης

Οι πιο πρόσφατες εξελίξεις συνδέουν τη μετατροπή UML σε κώδικα με την τεχνητή νοημοσύνη. Στη μελέτη των Al-Azzoni et al. (2025), παρουσιάζεται η ενσωμάτωση MDE και τεχνικών AI για την ανάπτυξη smart contracts, επιτρέποντας την αυτοματοποιημένη παραγωγή κώδικα με βάση μοντέλα.

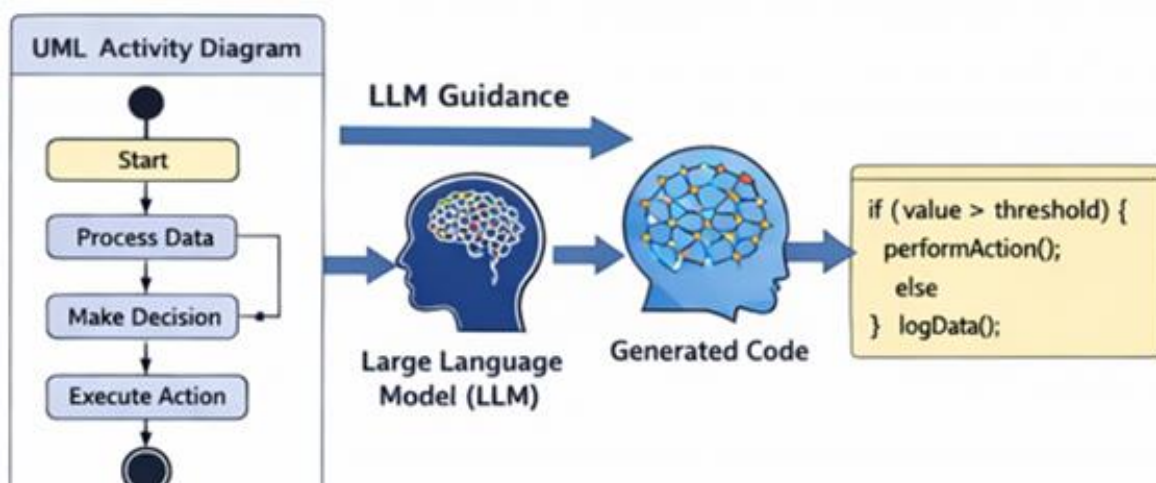
Παράλληλα, το σύστημα StructGen προτείνει τη χρήση UML Activity Diagrams ως ενδιάμεσο βήμα για την καθοδήγηση μεγάλων γλωσσικών μοντέλων (LLMs) στην

παραγωγή κώδικα. Η προσέγγιση αυτή βελτιώνει την ακρίβεια και μειώνει την αβεβαιότητα στη διαδικασία παραγωγής (StructGen, 2026).

Επιπλέον, σχετικές μελέτες δείχνουν ότι τα LLMs μπορούν να μετατρέψουν UML διαγράμματα σε κώδικα με σχετικά υψηλή ακρίβεια, αν και η πολυπλοκότητα των μοντέλων επηρεάζει την απόδοση (Antal et al., 2024).



Εικ. 6-3 Οδηγία LLM στην παραγωγή κώδικα μέσω UML Activity Diagrams



Εικ. 6-4 Οδηγία LLM στην παραγωγή κώδικα μέσω UML Activity Diagrams

Συνολική αξιολόγηση σύγχρονων μελετών

Οι σύγχρονες μελέτες περίπτωσης καταδεικνύουν ότι η MDE αποτελεί πλέον ένα ισχυρό εργαλείο ανάπτυξης λογισμικού. Συγκεκριμένα:

- Η χρήση μοντέλων μειώνει τον χρόνο ανάπτυξης και τα σφάλματα (Di Ruscio et al., 2021).
- Η εφαρμογή της επεκτείνεται σε πολλαπλούς τομείς, όπως mobile, βιομηχανία και τεχνητή νοημοσύνη (Usman et al., 2024; Al-Azzoni et al., 2025).
- Η σημασιολογική συνέπεια παραμένει βασική προϋπόθεση για την επιτυχία της μετατροπής (Lano & Siala, 2024).

Συνολικά, οι μελέτες επιβεβαιώνουν ότι η μετατροπή UML σε εκτελέσιμο κώδικα εξελίσσεται δυναμικά, με την ενσωμάτωση νέων τεχνολογιών να ανοίγει νέες προοπτικές αλλά και προκλήσεις.

Κριτική Αξιολόγηση της Χρήσης LLMs στην Παραγωγή Κώδικα

Παρόλο που οι πρόσφατες μελέτες αναφέρουν υψηλά ποσοστά επιτυχίας κατά τη μετατροπή UML μοντέλων σε κώδικα μέσω μεγάλων γλωσσικών μοντέλων (LLMs), τα αποτελέσματα αυτά πρέπει να αξιολογούνται με προσοχή. Για παράδειγμα, αναφορές σε ποσοστά ακρίβειας της τάξης του 88% δεν συνεπάγονται απαραίτητα ότι ο παραγόμενος κώδικας είναι σημασιολογικά ορθός ή έτοιμος για παραγωγική χρήση.

Η πλειονότητα των σχετικών μελετών αξιολογεί τη συντακτική ορθότητα του παραγόμενου κώδικα ή τη δομική αντιστοίχιση με το UML μοντέλο. Ωστόσο, η σημασιολογική ορθότητα αποτελεί διαφορετικό και πολύ πιο απαιτητικό κριτήριο (Pearce et al., 2024). Ένα LLM μπορεί να παράγει κώδικα που μεταγλωττίζεται επιτυχώς, αλλά να παραβιάζει επιχειρησιακούς κανόνες ή να αλλοιώνει τη λογική του αρχικού μοντέλου.

Ιδιαίτερα προβληματική είναι η περίπτωση Activity Diagrams που περιλαμβάνουν παράλληλες ροές, συγχρονισμό ή πολύπλοκες συνθήκες απόφασης. Σε τέτοιες περιπτώσεις, τα LLMs συχνά παράγουν λειτουργικά ισοδύναμο αλλά όχι σημασιολογικά ισοδύναμο κώδικα, δημιουργώντας αποκλίσεις μεταξύ μοντέλου και υλοποίησης (Hou et al., 2024).

Επιπλέον, τα LLMs παρουσιάζουν το φαινόμενο των hallucinations, δηλαδή την παραγωγή συντακτικά ορθού αλλά λογικά λανθασμένου κώδικα. Το πρόβλημα αυτό γίνεται ιδιαίτερα κρίσιμο σε περιβάλλοντα Model-Driven Engineering, όπου ο στόχος είναι η αυστηρή διατήρηση της σημασιολογίας κατά τη διαδικασία μετασχηματισμού.

Συνεπώς, τα LLMs δεν μπορούν ακόμη να θεωρηθούν πλήρεις αντικαταστάτες των κλασικών σημασιολογικών μετασχηματισμών. Αντίθετα, λειτουργούν αποτελεσματικότερα ως βοηθητικά εργαλεία παραγωγής κώδικα, τα οποία απαιτούν επαλήθευση μέσω τυπικών σημασιολογικών μηχανισμών και ανθρώπινο έλεγχο πριν από την τελική αξιοποίηση του παραγόμενου λογισμικού.

6.5 Εργαλεία UML και δυνατότητες παραγωγής κώδικα

Σύγχρονος ρόλος των UML εργαλείων

Παρά την κριτική που έχει δεχθεί η UML τα τελευταία χρόνια, εξακολουθεί να αποτελεί βασικό εργαλείο για τον σχεδιασμό σύνθετων συστημάτων, ιδιαίτερα σε βιομηχανικά και enterprise περιβάλλοντα. Σύγχρονες μελέτες δείχνουν ότι η UML δεν έχει εγκαταλειφθεί, αλλά έχει εξελιχθεί μέσω της ενσωμάτωσης τεχνητής νοημοσύνης και αυτοματισμών code generation (Tsang, 2025; Visual Paradigm Research, 2025).

Η νέα γενιά εργαλείων δεν περιορίζεται πλέον στη στατική μοντελοποίηση, αλλά λειτουργεί ως “AI-assisted modeling environments” που υποστηρίζουν παραγωγή κώδικα, validation και synchronization με repositories (Tsang, 2025).

Σύγχρονα UML εργαλεία και δυνατότητες

Τα πιο διαδεδομένα σύγχρονα εργαλεία UML περιλαμβάνουν:

- Sparx Systems Enterprise Architect (έκδοση 17+)
- Cameo Systems Modeler (MagicDraw)
- Visual Paradigm 17/18+
- StarUML v6+

Τα εργαλεία αυτά υποστηρίζουν πλέον πλήρη round-trip engineering, δηλαδή:

- UML → Code generation
- Code → UML reverse engineering
- Συγχρονισμό μεταξύ μοντέλου και πηγαίου κώδικα

Η δυνατότητα αυτή θεωρείται βασικό στοιχείο των Model-Driven Engineering (MDE) εργαλείων (OMG, 2023; Selic, 2023).

Παραγωγή κώδικα (Code Generation) από UML

Η σύγχρονη παραγωγή κώδικα από UML έχει εξελιχθεί σημαντικά λόγω της ενσωμάτωσης machine learning μοντέλων και LLMs.

Κλασική Model-Driven code generation

Τα εργαλεία MDE παράγουν:

- Java / C# / C++ κλάσεις
- Interfaces
- Βασικές δομές αρχιτεκτονικής

Η διαδικασία βασίζεται σε transformation rules (model-to-text transformations) όπως περιγράφεται από την OMG (2023).

AI-based UML code generation

Πρόσφατες έρευνες δείχνουν ότι μεγάλα γλωσσικά μοντέλα μπορούν να μετατρέψουν UML διαγράμματα σε λειτουργικό κώδικα με υψηλή ακρίβεια.

Για παράδειγμα:

- GPT-4 Vision μπορεί να αναγνωρίσει UML class diagrams και να παράγει Java code με 88% ακρίβεια στοιχείων (Antal et al., 2024)
- Multimodal LLMs επιτυγχάνουν υψηλές επιδόσεις σε sequence και activity diagrams (Bates et al., 2025).

Αυτό σημαίνει ότι η UML πλέον λειτουργεί όχι μόνο ως σχεδιαστικό εργαλείο, αλλά και ως “intermediate programming representation” για AI-based development pipelines.

Σύγχρονες τάσεις στα UML εργαλεία

Οι σημαντικότερες σύγχρονες τάσεις είναι:

AI-assisted modeling

- Τα εργαλεία UML ενσωματώνουν AI assistants που:
- δημιουργούν διαγράμματα από φυσική γλώσσα
- προτείνουν σχέσεις μεταξύ κλάσεων
- εντοπίζουν design inconsistencies (Visual Paradigm, 2025)

Integration με DevOps

Η UML πλέον συνδέεται με:

- CI/CD pipelines
- Git-based modeling
- cloud repositories

Model-to-code automation σε enterprise συστήματα

Η παραγωγή boilerplate κώδικα μειώνει σημαντικά τον χρόνο ανάπτυξης, αλλά δεν αντικαθιστά πλήρως τον προγραμματισμό (Selic, 2023).

Περιορισμοί των σύγχρονων UML εργαλείων

Παρά την πρόοδο, υπάρχουν ακόμα περιορισμοί:

- Η UML δεν εκφράζει πλήρως business logic
- Η παραγωγή κώδικα παραμένει κυρίως boilerplate
- Υπάρχει εξάρτηση από vendor-specific εργαλεία

Τα AI-generated models απαιτούν ανθρώπινη επιβεβαίωση (Bates et al., 2025). Έρευνες δείχνουν ότι η πλήρης αυτοματοποίηση δεν είναι ακόμη εφικτή, ιδιαίτερα σε multi-class complex systems (Antal et al., 2024).

Σύνδεση της Σημασιολογίας με τη Model-Driven Engineering

Η σημασία της σημασιολογίας στη Model-Driven Engineering γίνεται σαφέστερη όταν εξεταστεί υπό το πρίσμα των προηγούμενων κεφαλαίων της παρούσας εργασίας.

Στο Κεφάλαιο 2 αναδείχθηκε η διάκριση μεταξύ συντακτικού και σημασιολογικού επιπέδου της UML. Η διάκριση αυτή είναι θεμελιώδης για τη MDE, καθώς η παραγωγή κώδικα απαιτεί όχι μόνο σωστά UML διαγράμματα αλλά και σαφώς ορισμένο νόημα των στοιχείων τους.

Στο Κεφάλαιο 3 παρουσιάστηκαν οι τυπικές σημασιολογικές προσεγγίσεις, όπως τα Petri Nets, η Object Constraint Language (OCL) και οι formal semantics. Οι προσεγγίσεις αυτές αποτελούν τη θεωρητική βάση που επιτρέπει τη διατήρηση της σημασιολογικής ισοδυναμίας κατά τους μετασχηματισμούς μοντέλων.

Στο Κεφάλαιο 4 αναλύθηκαν σημασιολογικά τα βασικά UML διαγράμματα και παρουσιάστηκαν συνηθισμένα λάθη ερμηνείας. Τα σφάλματα αυτά αποκτούν ιδιαίτερη σημασία στη MDE, διότι οποιαδήποτε σημασιολογική ασάφεια ενσωματώνεται αυτομάτως στον παραγόμενο κώδικα.

Για παράδειγμα, ένα Activity Diagram που περιλαμβάνει ασαφείς συνθήκες σε decision nodes μπορεί να οδηγήσει σε διαφορετικές υλοποιήσεις από διαφορετικά εργαλεία μετασχηματισμού. Αντίστοιχα, λανθασμένες πληθικότητες σε Class Diagrams μπορούν να οδηγήσουν σε εσφαλμένες δομές δεδομένων στον παραγόμενο κώδικα.

Η MDE επομένως δεν αποτελεί απλώς μια τεχνική αυτοματοποίησης της ανάπτυξης λογισμικού. Αποτελεί μια διαδικασία που εξαρτάται άμεσα από τη σημασιολογική ποιότητα των μοντέλων. Όσο μεγαλύτερη είναι η σημασιολογική πληρότητα και συνέπεια ενός UML μοντέλου, τόσο μεγαλύτερη είναι η πιθανότητα επιτυχούς και αξιόπιστης παραγωγής εκτελέσιμου κώδικα.

Συμπεράσματα

Τα σύγχρονα UML εργαλεία έχουν μετασχηματιστεί από απλά diagram editors σε ολοκληρωμένα model-driven και AI-enhanced περιβάλλοντα ανάπτυξης λογισμικού. Η δυνατότητα παραγωγής κώδικα αποτελεί πλέον βασικό χαρακτηριστικό, ενώ η ενσωμάτωση τεχνητής νοημοσύνης αναμένεται να ενισχύσει περαιτέρω τον ρόλο της UML στη μηχανική λογισμικού.

7. Μελέτη Περίπτωσης 1: Μοντελοποίηση πειραματικών διαδικασιών σε εικονικό εργαστήριο

7.1 Εικονικά Εργαστήρια

Η ανάπτυξη των τεχνολογιών πληροφορικής, των συστημάτων προσομοίωσης και των ψηφιακών μαθησιακών περιβαλλόντων έχει επηρεάσει σημαντικά τον τρόπο υλοποίησης της εκπαιδευτικής διαδικασίας στις φυσικές επιστήμες, στη μηχανική και γενικότερα στην εκπαίδευση STEM. Τα εικονικά εργαστήρια (virtual laboratories) αποτελούν ένα σύγχρονο εκπαιδευτικό εργαλείο που επιτρέπει την προσομοίωση πειραματικών διαδικασιών μέσω διαδραστικών υπολογιστικών περιβαλλόντων. Η αξιοποίησή τους ενισχύθηκε ιδιαίτερα κατά την περίοδο της πανδημίας COVID-19, όταν η εξ αποστάσεως εκπαίδευση κατέστη αναγκαία για την εκπαιδευτική συνέχεια (Raman et al., 2022).

Η μοντελοποίηση πειραματικών διαδικασιών σε εικονικό εργαστήριο αναφέρεται στη δημιουργία ψηφιακών αναπαραστάσεων πραγματικών εργαστηριακών δραστηριοτήτων μέσω μαθηματικών, υπολογιστικών και γραφικών μοντέλων. Μέσω αυτών των προσομοιώσεων, οι εκπαιδευόμενοι μπορούν να αλληλεπιδρούν με εικονικό εξοπλισμό, να τροποποιούν παραμέτρους και να παρατηρούν αποτελέσματα σε πραγματικό χρόνο, χωρίς να απαιτείται φυσική παρουσία σε εργαστηριακό χώρο. Σύμφωνα με τη διεθνή βιβλιογραφία, τα εικονικά εργαστήρια συμβάλλουν σημαντικά στην κατανόηση θεωρητικών εννοιών, στη διερευνητική μάθηση και στην ανάπτυξη δεξιοτήτων επίλυσης προβλημάτων (Brinson, 2015; Potkonjak et al., 2016).

Παρακάτω, παρουσιάζονται οι θεωρητικές βάσεις, οι τεχνολογίες, οι μεθοδολογίες και οι σύγχρονες ερευνητικές τάσεις που σχετίζονται με τη μοντελοποίηση πειραματικών διαδικασιών σε εικονικά εργαστήρια, με έμφαση στις εκπαιδευτικές εφαρμογές και στην αποτελεσματικότητά τους.

Εικονικά Εργαστήρια - Ορισμός και χαρακτηριστικά

Σύμφωνα με τους (Ma & Nickerson, 2006), τα εικονικά εργαστήρια αποτελούν ψηφιακά περιβάλλοντα προσομοίωσης, τα οποία αναπαριστούν τη λειτουργία πραγματικών εργαστηρίων μέσω υπολογιστικών συστημάτων. Οι χρήστες έχουν τη δυνατότητα να

εκτελούν πειράματα, να μεταβάλλουν παραμέτρους και να λαμβάνουν άμεση ανατροφοδότηση σε πραγματικό χρόνο (Ma & Nickerson, 2006).

Τα βασικά χαρακτηριστικά των εικονικών εργαστηρίων περιλαμβάνουν:

- διαδραστικότητα,
- προσομοίωση πραγματικών συνθηκών,
- δυνατότητα επανάληψης πειραμάτων,
- ασφαλές περιβάλλον μάθησης,
- προσβασιμότητα από απόσταση,
- χαμηλότερο λειτουργικό κόστος.

Η χρήση τους συνδέεται στενά με τις θεωρίες της ενεργητικής και εποικοδομητικής μάθησης, σύμφωνα με τις οποίες οι μαθητές οικοδομούν τη γνώση μέσω της αλληλεπίδρασης με το μαθησιακό περιβάλλον (Piaget, 1970; Vygotsky, 1978).

Η ανάπτυξη των εικονικών εργαστηρίων ξεκίνησε κατά τη δεκαετία του 1990 με την πρόοδο των τεχνολογιών πολυμέσων και του διαδικτύου. Αρχικά, τα συστήματα αυτά περιορίζονταν σε απλές δισδιάστατες προσομοιώσεις. Ωστόσο, η εξέλιξη των τρισδιάστατων γραφικών, των μηχανών παιχνιδιών και των τεχνολογιών εικονικής πραγματικότητας οδήγησε στη δημιουργία ιδιαίτερα ρεαλιστικών περιβαλλόντων (Mikropoulos & Natsis, 2011). Σύμφωνα με τη βιβλιομετρική μελέτη των Raman et al. (2022), περισσότερο από το 70% των δημοσιεύσεων που σχετίζονται με τα εικονικά εργαστήρια δημοσιεύθηκαν μετά το 2011, γεγονός που καταδεικνύει τη ραγδαία ανάπτυξη του συγκεκριμένου ερευνητικού πεδίου.

Μοντελοποίηση Πειραματικών Διαδικασιών

Η μοντελοποίηση αποτελεί τη διαδικασία αναπαράστασης πραγματικών συστημάτων μέσω μαθηματικών και υπολογιστικών μοντέλων. Στα εικονικά εργαστήρια, η μοντελοποίηση χρησιμοποιείται για την προσομοίωση φυσικών, χημικών, βιολογικών και μηχανικών φαινομένων. Η ποιότητα του μοντέλου επηρεάζει άμεσα την αξιοπιστία και την εκπαιδευτική αποτελεσματικότητα του εικονικού περιβάλλοντος. Τα μοντέλα πρέπει να είναι επιστημονικά έγκυρα, αλλά παράλληλα επαρκώς απλοποιημένα ώστε να είναι κατανοητά από τους εκπαιδευόμενους.

Μαθηματικά και υπολογιστικά μοντέλα

Για την ανάπτυξη εικονικών εργαστηρίων αξιοποιούνται μαθηματικά και υπολογιστικά μοντέλα: διαφορικές εξισώσεις, στοχαστικά μοντέλα, αριθμητικές προσομοιώσεις, μοντέλα πεπερασμένων στοιχείων, δυναμικά συστήματα.

Εργαλεία όπως τα MATLAB/Simulink, Unity 3D, LabVIEW και OpenModelica χρησιμοποιούνται ευρέως για την ανάπτυξη εκπαιδευτικών προσομοιώσεων (Potkonjak et al., 2016).

Διαδικασία ανάπτυξης εικονικού εργαστηρίου

Η διαδικασία ανάπτυξης ενός εικονικού εργαστηρίου περιλαμβάνει συνήθως τα ακόλουθα στάδια:

1. Ανάλυση του πραγματικού πειράματος
2. Καθορισμός μαθησιακών στόχων
3. Δημιουργία μαθηματικού μοντέλου
4. Ανάπτυξη γραφικής διεπαφής
5. Ενσωμάτωση αλληλεπίδρασης χρήστη
6. Έλεγχος εγκυρότητας της προσομοίωσης
7. Εκπαιδευτική αξιολόγηση

Η εγκυρότητα του μοντέλου αποτελεί κρίσιμο παράγοντα, καθώς επηρεάζει την αξιοπιστία των πειραματικών αποτελεσμάτων και τη συνολική εμπειρία μάθησης (Chacon et al., 2025; Sypsas & Kalles, 2023).

Εκπαιδευτικές Θεωρίες και Εικονικά Εργαστήρια

Η χρήση εικονικών εργαστηρίων στηρίζεται σε σύγχρονες παιδαγωγικές θεωρίες, όπως ο εποικοδομητισμός και η βιωματική μάθηση. Σύμφωνα με τον Kolb (1984), η μάθηση πραγματοποιείται αποτελεσματικότερα όταν ο εκπαιδευόμενος συμμετέχει ενεργά σε βιωματικές δραστηριότητες. Τα εικονικά εργαστήρια ενισχύουν αυτή τη διαδικασία, επιτρέποντας στους φοιτητές να πειραματίζονται χωρίς φόβο αποτυχίας ή κινδύνου.

Παράλληλα, η θεωρία της μάθησης μέσω ανακάλυψης υποστηρίζει ότι οι μαθητές κατανοούν βαθύτερα τις επιστημονικές έννοιες όταν διερευνούν μόνοι τους τα αποτελέσματα διαφορετικών παραμέτρων και μεταβλητών (Dewey, 1938). Μελέτες

δείχνουν ότι τα εικονικά εργαστήρια ενισχύουν την αυτορρυθμιζόμενη μάθηση, την ενεργό συμμετοχή και την ανάπτυξη δεξιοτήτων επίλυσης προβλημάτων (Sapriati et al., 2023).

Τεχνολογίες Υλοποίησης

Τρισδιάστατα περιβάλλοντα και μηχανές γραφικών

Η ανάπτυξη σύγχρονων εικονικών εργαστηρίων βασίζεται σε τεχνολογίες τρισδιάστατης απεικόνισης, όπως:

- Unity,
- Unreal Engine,
- WebGL,
- Blender.

Οι τεχνολογίες αυτές επιτρέπουν τη δημιουργία ρεαλιστικών περιβαλλόντων και διαδραστικών αντικειμένων.

Εικονική πραγματικότητα (VR)

Η τεχνολογία εικονικής πραγματικότητας προσφέρει πλήρη εμβύθιση του χρήστη σε τρισδιάστατο περιβάλλον. Σύμφωνα με μετα-ανάλυση των Merchant et al. (2014), τα περιβάλλοντα VR μπορούν να βελτιώσουν σημαντικά τα μαθησιακά αποτελέσματα και την εμπλοκή των εκπαιδευομένων.

Η VR χρησιμοποιείται ιδιαίτερα στην ιατρική εκπαίδευση, στη μηχανική και στις φυσικές επιστήμες, καθώς επιτρέπει την ασφαλή προσομοίωση σύνθετων ή επικίνδυνων διαδικασιών.

Επαυξημένη πραγματικότητα (AR)

Η επαυξημένη πραγματικότητα συνδυάζει πραγματικά και ψηφιακά αντικείμενα, επιτρέποντας την αλληλεπίδραση με εικονικά στοιχεία στον φυσικό χώρο. Η τεχνολογία αυτή χρησιμοποιείται ευρέως στην ιατρική εκπαίδευση, στη βιολογία και στη μηχανική.

Τεχνητή νοημοσύνη και learning analytics

Οι σύγχρονες εφαρμογές εικονικών εργαστηρίων ενσωματώνουν τεχνολογίες τεχνητής νοημοσύνης και learning analytics, με σκοπό:

- την εξατομίκευση της μάθησης,
- την προσαρμογή της δυσκολίας,

- την αυτόματη αξιολόγηση,
- την παρακολούθηση της προόδου των χρηστών.

Η συστηματική ανασκόπηση των Elmoazen et al. (2023) επισημαίνει ότι τα learning analytics μπορούν να συμβάλουν σημαντικά στην αξιολόγηση της συμμετοχής και της απόδοσης των φοιτητών στα εικονικά εργαστήρια.

Πλεονεκτήματα και Περιορισμοί

Πλεονεκτήματα

Η διεθνής βιβλιογραφία αναδεικνύει πολλαπλά οφέλη από τη χρήση εικονικών εργαστηρίων:

- μείωση κόστους εξοπλισμού και αναλωσίμων,
- ασφαλές περιβάλλον για επικίνδυνα πειράματα,
- δυνατότητα εξ αποστάσεως πρόσβασης,
- επαναληψιμότητα πειραμάτων,
- ενίσχυση της αυτορρυθμιζόμενης μάθησης.

Επιπλέον, έρευνες δείχνουν ότι τα εικονικά εργαστήρια μπορούν να επιτύχουν ίσα ή και καλύτερα γνωστικά μαθησιακά αποτελέσματα σε σύγκριση με τα παραδοσιακά εργαστήρια (Chan et al., 2021).

Περιορισμοί

Παρά τα σημαντικά πλεονεκτήματα, υπάρχουν και περιορισμοί:

- μειωμένη φυσική αλληλεπίδραση με πραγματικό εξοπλισμό,
- υψηλό αρχικό κόστος ανάπτυξης,
- ανάγκη τεχνικής υποδομής,
- δυσκολία προσομοίωσης ορισμένων φυσικών φαινομένων με απόλυτο ρεαλισμό.

Μελέτες στη φυσιολογία και στις επιστήμες υγείας επισημαίνουν ότι τα εικονικά εργαστήρια λειτουργούν αποτελεσματικότερα ως συμπληρωματικό εργαλείο και όχι ως πλήρης αντικατάσταση των φυσικών εργαστηρίων (Zhang et al., 2021).

Σύγχρονες Τάσεις και Μελλοντικές Προοπτικές

Οι σύγχρονες ερευνητικές τάσεις επικεντρώνονται:

- στην αξιοποίηση τεχνολογιών metaverse,
- στη συνεργατική εικονική μάθηση,
- στην ενσωμάτωση τεχνητής νοημοσύνης,
- στην ανάπτυξη προσαρμοστικών περιβαλλόντων μάθησης,
- στη χρήση cloud υποδομών και learning analytics.

Πρόσφατες μελέτες στον χώρο της ιατρικής εκπαίδευσης δείχνουν ότι τα υβριδικά και εικονικά εργαστήρια ενισχύουν την ικανοποίηση των φοιτητών, τη γνωστική κατανόηση και την ανάπτυξη επαγγελματικών δεξιοτήτων (Bamashmous et al., 2025).

Συμπεράσματα

Η μοντελοποίηση πειραματικών διαδικασιών σε εικονικά εργαστήρια αποτελεί έναν ιδιαίτερα σημαντικό και ταχέως αναπτυσσόμενο τομέα της σύγχρονης εκπαιδευτικής τεχνολογίας. Τα εικονικά εργαστήρια προσφέρουν ευελιξία, ασφάλεια, χαμηλότερο κόστος και αυξημένες δυνατότητες αλληλεπίδρασης, συμβάλλοντας ουσιαστικά στην υποστήριξη της εξ αποστάσεως και βιωματικής μάθησης.

Η βιβλιογραφία υποδεικνύει ότι τα εικονικά εργαστήρια μπορούν να βελτιώσουν τη γνωστική κατανόηση και τη μαθησιακή εμπειρία, ιδιαίτερα όταν συνδυάζονται με παραδοσιακές εργαστηριακές δραστηριότητες. Παράλληλα, η συνεχής πρόοδος στις τεχνολογίες VR, AR, AI και learning analytics αναμένεται να ενισχύσει περαιτέρω την αποτελεσματικότητα και τη ρεαλιστικότητα των εικονικών περιβαλλόντων μάθησης.

7.2 Περιγραφή σεναρίου: 1^η μελέτη περίπτωσης: Σύστημα Ελέγχου LED με Arduino Uno στο Tinkercad

Το σενάριο απευθύνεται σε μαθητές Β/θμιας Εκπαίδευσης στο πλαίσιο του γνωστικού αντικειμένου της Εκπαιδευτικής Ρομποτικής. Η εικόνα 1, απεικονίζει πιθανό σενάριο στο πλαίσιο μιας δραστηριότητας που αφορά στην υλοποίηση ενός απλού συστήματος αυτόματου φωτισμού με τη χρήση αισθητήρα φωτός LDR. Ο αισθητήρας ανιχνεύει το επίπεδο φωτεινότητας στο περιβάλλον και όταν αυτό πέσει κάτω από ένα καθορισμένο όριο, ενεργοποιεί αυτόματα ένα LED. Αν υπάρχει σκοτάδι, το LED ανάβει, αλλιώς σβήνει. Η λειτουργία του κυκλώματος είναι να προσομοιώνει έναν έξυπνο φωτισμό που ενεργοποιείται μόνο όταν απαιτείται. Έτσι, το κύκλωμα συμβάλλει στην εξοικονόμηση ενέργειας και κατ' επέκταση στην προστασία του περιβάλλοντος. Γίνεται προσομοίωση της λειτουργίας του κυκλώματος στην πλατφόρμα Tinkercad. Η κατασκευή έχει πρακτική εφαρμογή σε συστήματα έξυπνου φωτισμού και εξοικονόμησης ενέργειας.

Σκοπός της δραστηριότητας

Ο σκοπός της δραστηριότητας είναι διττός: (α) η ευαισθητοποίηση των μαθητών αναφορικά με τα περιβαλλοντικά θέματα (β) η αξιοποίηση της τεχνολογίας για την επίλυση περιβαλλοντικών προβλημάτων.

Προσδοκώμενα Μαθησιακά Αποτελέσματα

Μετά την εκπόνηση της δραστηριότητας οι μαθητές θα είναι σε θέση να:

- παρουσιάζουν τις βασικές αρχές των μικροελεγκτών και της εκπαιδευτικής ρομποτικής
- σχεδιάζουν και να υλοποιούν κύκλωμα στην πλατφόρμα Arduino UNO, σε περιβάλλον προσομοίωσης, στην πλατφόρμα Tinkercad.

Παρουσίαση του Κυκλώματος

Υλικά: Το κύκλωμα περιλαμβάνει τα παρακάτω υλικά:

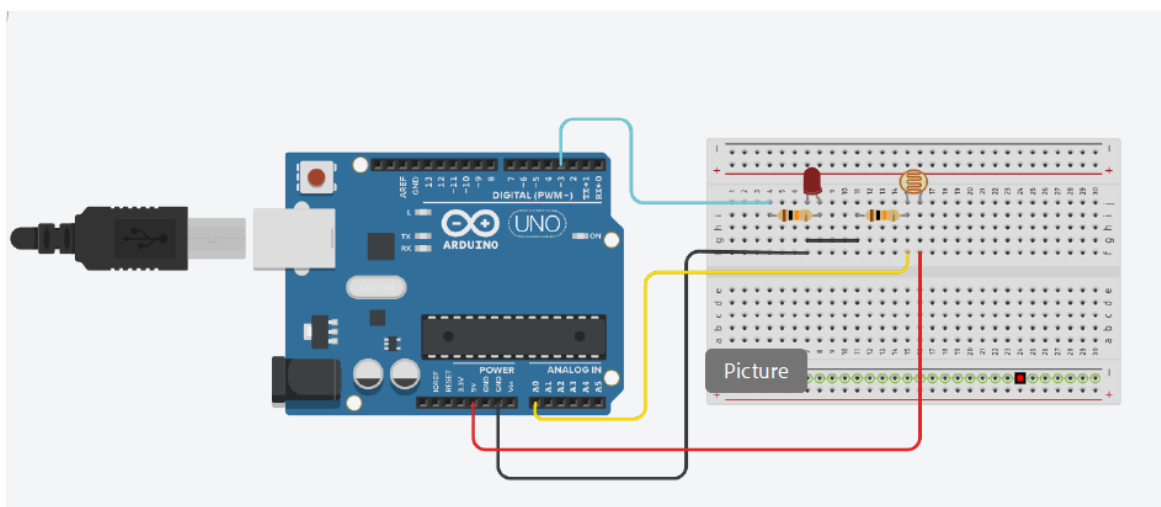
Arduino Uno

Breadboard

Αισθητήρα φωτός (photoresistor)

Led για οπτική ειδοποίηση

Αντιστάσεις και Καλώδια Σύνδεσης



Εικ.7-1 Προσομοίωση έξυπνου φωτισμού

Φωτοαντίσταση (LDR): Η φωτοαντίσταση είναι μια αντίσταση μεταβλητής τιμής η οποία εξαρτάται από την ένταση του φωτός στο περιβάλλον. Το ένα άκρο συνδέεται με μια αντίσταση και η αντίσταση με τη σειρά της συνδέεται με το GND του Arduino. Έτσι, μετράμε την αντίσταση του φωτοαντιστάτη για να μετρήσουμε την ένταση του φωτός στο περιβάλλον. Το άλλο άκρο του LDR συνδέεται με την αναλογική είσοδο A0. Η επιλογή της αναλογικής εισόδου επιτρέπει την ανάγνωση διαβαθμίσεων φωτεινότητας αντί των δυαδικών καταστάσεων (ON/OFF). Με αυτόν τον τρόπο, μπορούμε να παρακολουθούμε με μεγαλύτερη ακρίβεια την ένταση του φωτός και να προγραμματίζουμε το σύστημα να ενεργοποιείται όταν η ένταση υποχωρεί κάτω από ένα καθορισμένο όριο.

LED: Το LED χρησιμοποιείται για να δώσει οπτική ένδειξη όταν το σύστημα ενεργοποιείται λόγω σκοταδιού. Το ένα άκρο του (κάθοδος) συνδέεται στο GND ενώ το άλλο άκρο (άνοδος) περνά από μία αντίσταση, η οποία προστατεύει το LED από υπερβολική ένταση ρεύματος. Στη συνέχεια, συνδέεται στο pin 3 του Arduino, το οποίο

λειτουργεί ως ψηφιακή έξοδος. Μέσω του προγράμματος αυτό το pin δέχεται τιμές ON ή OFF.

Αντιστάσεις: η φωτοαντίσταση μπορεί να έχει αντίσταση που φτάνει έως και 100 kΩ σε συνθήκες χαμηλού φωτισμού. Επιλέχθηκε μια αντίσταση 10 kΩ, η οποία εξασφαλίζει την ομαλή λειτουργία του αισθητήρα σε συνθήκες μέτριου φωτισμού (π.χ. ένα φωτισμένο δωμάτιο). Η αντίσταση αυτή διασφαλίζει τη σταθερή και αξιόπιστη λειτουργία του κυκλώματος υπό κανονικές συνθήκες. Αν χρησιμοποιούσαμε μικρότερη αντίσταση (π.χ. 1 kΩ), το κύκλωμα θα γινόταν υπερβολικά ευαίσθητο και ασταθές. Αν στο κύκλωμα χρησιμοποιήσουμε μια πολύ μεγάλη αντίσταση (π.χ. 100 kΩ), το κύκλωμα θα αντιδρούσε μόνο σε έντονο σκοτάδι, παραβλέποντας τις ενδιάμεσες καταστάσεις φωτεινότητας.

Κώδικας

```
1 const int LIGHT_SENSOR_PIN = A0; // Arduino pin connected to light sensor's pin, Analog Input
2 const int LED_PIN = 3; // Arduino pin connected to LED's pin, Digital Output
3 const int ANALOG_THRESHOLD = 500; // Brightness limit
4
5 int analogValue; // stores the brightness value read by the sensor
6
7 void setup() {
8   pinMode(LED_PIN, OUTPUT); // set arduino pin to output mode, It means that LED_PIN it's the output
9 }
10
11 void loop() {
12   analogValue = analogRead(LIGHT_SENSOR_PIN); // read the input on analog pin
13
14   if(analogValue < ANALOG_THRESHOLD)
15     digitalWrite(LED_PIN, HIGH); //if it's dark, turn on LED
16   else
17     digitalWrite(LED_PIN, LOW); // turn off LED
18 }
```

Εικ.7-2 Κώδικας δραστηριότητας

Επεξήγηση κώδικα

- 1) **const int LIGHT_SENSOR_PIN= A0;** Ορίζεται μια σταθερά (const int) με το όνομα LIGHT_SENSOR_PIN, η οποία αντιστοιχεί στην αναλογική είσοδο A0 του Arduino, όπου είναι συνδεδεμένος ο αισθητήρας φωτός (LDR). Η αναλογική είσοδος, επιτρέπει την ανάγνωση μεταβαλλόμενων τιμών φωτεινότητας.

- 2) **const int LED_PIN=3;** Ορίζεται η σταθερά LED_PIN για το ψηφιακό pin 3, στο οποίο είναι συνδεδεμένο το LED. Αυτό το pin λειτουργεί ως έξοδος (OUTPUT) και στέλνει σήμα HIGH (ενεργοποίηση) ή LOW (απενεργοποίηση) ανάλογα με τις συνθήκες του φωτός.
- 3) **const int ANALOG_THRESHOLD=500;:** Ορίζεται η σταθερά ANALOG_THRESHOLD για το κατώφλι φωτεινότητας. Αν η αναγνωσμένη φωτεινότητα είναι μικρότερη από 500 (δηλαδή σκοτεινά), το LED ενεργοποιείται. Αν είναι μεγαλύτερη από 500 (έχει φως), απενεργοποιείται.
- 4) **Int analog Value; :** Ορίζεται η μεταβλητή analogValue η οποία αποθηκεύει την τιμή φωτεινότητας που διαβάζεται από τον αισθητήρα.
- 5) **Void setup() {pinMode(LED_PIN, OUTPUT);}**: Στη συνάρτηση setup(), η οποία εκτελείται μια φορά κατά την εκκίνηση, καθορίζεται ότι το pin του LED είναι έξοδος. Δηλ. θα στέλνει ρεύμα προς το LED για να το ανάψει ή να το σβήσει.
- 6) **Void loop() {analogValue = analogRead(LIGHT_SENSOR_PIN); :** Στη συνάρτηση loop() (που εκτελείται επαναληπτικά), διαβάζεται η τρέχουσα τιμή από τον αισθητήρα φωτός μέσω της εντολής analogRead(). Η τιμή αποθηκεύεται στη μεταβλητή analogValue.
- 7) **If (analogValue < ANALOG_THRESHOLD)**
 digitalWrite(LED_PIN, HIGH);
Else
 digitalWrite(LED_PIN, LOW);}
Αν η τιμή της φωτεινότητας είναι μικρότερη από το κατώφλι (ANALOG_THRESHOLD), τότε το πρόγραμμα ανάβει το LED (digitalWrite(LED_PIN, HIGH)). Αν είναι μεγαλύτερη, που σημαίνει ότι ο φωτισμός είναι επαρκής, το LED παραμένει σβηστό (digitalWrite(LED_PIN, LOW)).

7.3 Δημιουργία Use Case Models για Σύστημα Ελέγχου LED με Arduino Uno στο Tinkercad

Εισαγωγή

Όπως ειπώθηκε στα παραπάνω κεφάλαια, τα Use Case Models χρησιμοποιούνται για την περιγραφή των λειτουργιών ενός συστήματος από την οπτική γωνία του χρήστη (Booch, Rumbaugh & Jacobson, 2005).

Στην παρούσα εργασία πραγματοποιείται η ανάπτυξη Use Case Models για ένα εκπαιδευτικό σύστημα προσομοίωσης ηλεκτρονικού κυκλώματος με χρήση Arduino Uno και LED στο περιβάλλον Tinkercad. Το σύστημα επιτρέπει στον χρήστη να δημιουργήσει κύκλωμα, να συνδέσει ηλεκτρονικά εξαρτήματα, να γράψει κώδικα Arduino και να εκτελέσει προσομοίωση λειτουργίας LED.

Η χρήση Use Case Models συμβάλλει:

- στην κατανόηση των απαιτήσεων του συστήματος,
- στην αναγνώριση των λειτουργικών αλληλεπιδράσεων,
- στη βελτίωση της επικοινωνίας μεταξύ αναλυτών και χρηστών,
- στη δημιουργία βάσης για περαιτέρω UML μοντελοποίηση (Sommerville, 2016).

Τα Use Case Models αποτελούν βασικό στοιχείο της UML (Unified Modeling Language) και χρησιμοποιούνται για την αναπαράσταση των λειτουργικών απαιτήσεων ενός συστήματος (OMG, 2023).

Ένα Use Case Diagram περιλαμβάνει:

- Actors,
- Use Cases,
- σχέσεις Association,
- σχέσεις Include,

- σχέσεις Extend,
- System Boundary.

Οι actors αναπαριστούν εξωτερικές οντότητες που αλληλεπιδρούν με το σύστημα, ενώ τα use cases περιγράφουν τις υπηρεσίες ή λειτουργίες που παρέχονται από το σύστημα. Σύμφωνα με τους Jacobson et al. (1992), τα use cases αποτελούν μηχανισμό σύνδεσης μεταξύ λειτουργικών απαιτήσεων και συμπεριφορικής ανάλυσης του συστήματος.

Περιγραφή του Συστήματος Ελέγχου LED με Arduino Uno στο Tinkercad

Το υπό μελέτη σύστημα αφορά στην προσομοίωση κυκλώματος LED στο Tinkercad με χρήση Arduino Uno.

Το κύκλωμα περιλαμβάνει:

- Arduino Uno,
- breadboard,
- LED,
- αντίσταση 220Ω,
- καλώδια σύνδεσης.

Η λειτουργία του συστήματος βασίζεται στην αποστολή ψηφιακού σήματος από το Pin 8 του Arduino προς το LED, ώστε αυτό να αναβοσβήνει περιοδικά.

Ο χρήστης αλληλεπιδρά με το περιβάλλον Tinkercad για:

- δημιουργία κυκλώματος,
- σύνδεση εξαρτημάτων,
- συγγραφή κώδικα,
- εκτέλεση προσομοίωσης,
- έλεγχο λειτουργίας.

Αναγνώριση Actors

Η πρώτη φάση δημιουργίας του Use Case Model αφορά τον εντοπισμό των actors που συμμετέχουν στο σύστημα.

✓ Χρήστης / Μαθητής

Ο βασικός actor του συστήματος είναι ο χρήστης (μαθητής), ο οποίος:

- δημιουργεί το κύκλωμα,

- προσθέτει εξαρτήματα,
- γράφει τον κώδικα,
- εκκινεί την προσομοίωση,
- ελέγχει τη λειτουργία του LED.

Ο actor αυτός θεωρείται primary actor, καθώς ξεκινά όλες τις βασικές λειτουργίες του συστήματος.

✓ **Tinkercad**

Το Tinkercad λειτουργεί ως περιβάλλον προσομοίωσης και υποστηρίζει:

- τη δημιουργία κυκλωμάτων,
- την τοποθέτηση εξαρτημάτων,
- την εκτέλεση simulation.

Αποτελεί supporting actor του συστήματος.

✓ **Arduino Uno**

Το Arduino Uno λειτουργεί ως embedded execution unit και είναι υπεύθυνο για:

- την εκτέλεση του κώδικα,
- την ενεργοποίηση του Pin 8,
- τον έλεγχο του LED.

Αναγνώριση Use Cases

Με βάση τις λειτουργικές απαιτήσεις του συστήματος, αναγνωρίστηκαν τα ακόλουθα use cases.

Πίνακας 7- 1. Use Cases

Use Case	Περιγραφή
Δημιουργία νέου κυκλώματος	Δημιουργία νέου project στο Tinkercad
Προσθήκη εξαρτημάτων	Εισαγωγή Arduino, LED, αντίστασης
Σύνδεση κυκλώματος	Δημιουργία ηλεκτρικών συνδέσεων
Συγγραφή κώδικα	Ανάπτυξη Arduino code
Εκκίνηση προσομοίωσης	Εκτέλεση simulation
Έλεγχος λειτουργίας LED	Επιβεβαίωση σωστής λειτουργίας
Διόρθωση σφαλμάτων	Εντοπισμός και διόρθωση προβλημάτων

Σχέσεις Μεταξύ Use Cases

✓ Σχέση Association

Η σχέση Association χρησιμοποιείται για τη σύνδεση actors και use cases.

- Ο χρήστης συνδέεται με όλα τα use cases.
- Το Arduino Uno συνδέεται με τη συγγραφή κώδικα και τον έλεγχο LED.
- Το Tinkercad συνδέεται με την προσομοίωση.

✓ Σχέση Include

Η σχέση <<include>> χρησιμοποιείται όταν μία λειτουργία απαιτεί υποχρεωτικά μία δεύτερη λειτουργία. Στο σύστημα: Η εκκίνηση προσομοίωσης οδηγεί υποχρεωτικά σε έλεγχο λειτουργίας.

Εκκίνηση προσομοίωσης <<include>> Έλεγχος λειτουργίας LED.

✓ Σχέση Extend

Η σχέση <<extend>> χρησιμοποιείται για εναλλακτικές ροές.

Στο μοντέλο:

Διόρθωση σφαλμάτων <<extend>> Σύνδεση κυκλώματος
και

Διόρθωση σφαλμάτων <<extend>> Συγγραφή κώδικα

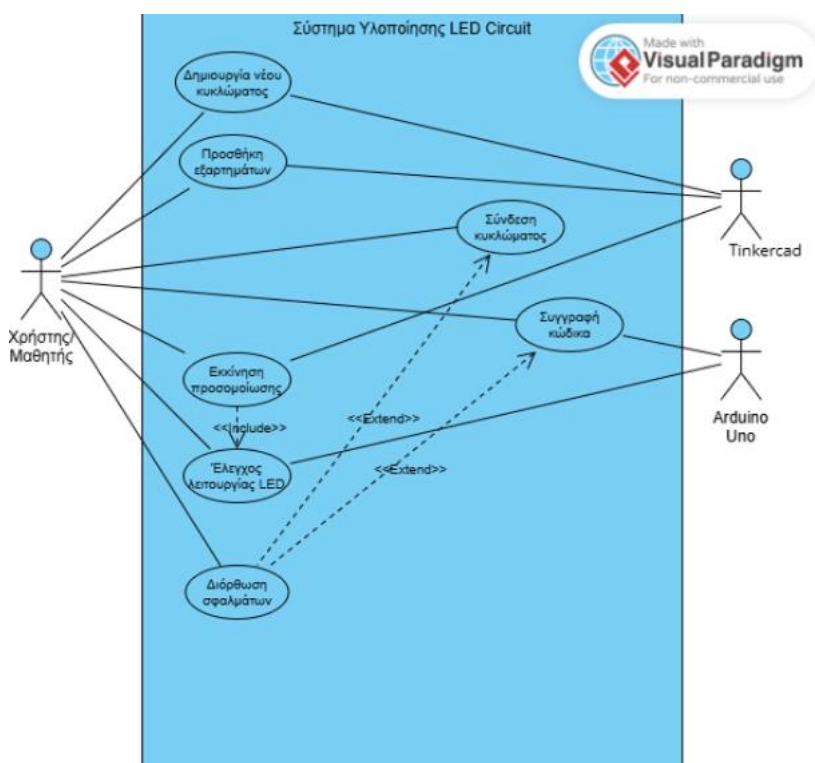
Οι σχέσεις αυτές εκφράζουν ότι η διαδικασία debugging ενεργοποιείται μόνο όταν παρουσιαστεί σφάλμα.

Δημιουργία του Use Case Diagram

Το τελικό Use Case Diagram αναπαριστά:

- τους actors,
- τις λειτουργίες του συστήματος,
- τις σχέσεις αλληλεπίδρασης.

Η δομή του διαγράμματος ακολουθεί τις προδιαγραφές UML 2.5.1 του Object Management Group (OMG, 2023), υλοποιήθηκε με το λογισμικό visual – paradigm και παρουσιάζεται στην εικόνα 7-3.



Εικ. 7- 3 Δημιουργία Use Case Models για Σύστημα Ελέγχου LED με Arduino Uno στο Tinkercad

Το σύστημα οριοθετείται μέσω system boundary με τίτλο:

Σύστημα Υλοποίησης LED Circuit στο Tinkercad

Στο εσωτερικό του boundary βρίσκονται όλα τα use cases του συστήματος.

Σημασιολογική ανάλυση του μοντέλου

Η σημασιολογική ανάλυση του μοντέλου έδειξε ότι:

- αποτυπώνονται όλες οι βασικές λειτουργίες,
- υπάρχει σαφής κατανομή ευθυνών,
- οι αλληλεπιδράσεις actors–system είναι ορθές,
- οι εναλλακτικές ροές μοντελοποιούνται επαρκώς.

Το μοντέλο παρουσιάζει υψηλή λειτουργική πληρότητα και μπορεί να χρησιμοποιηθεί ως βάση για:

- Activity Diagrams,
- Sequence Diagrams,
- State Machines,
- SysML μοντελοποίηση.

Πλεονεκτήματα- Περιορισμοί της Χρήσης Use Case Models

Η χρήση Use Case Models προσφέρει:

- Κατανόηση Απαιτήσεων: Το μοντέλο βοηθά στην κατανόηση των λειτουργικών απαιτήσεων του συστήματος.
- Επικοινωνία με Χρήστες: διευκολύνει την επικοινωνία μεταξύ αναλυτών και τελικών χρηστών.
- Βάση για UML Ανάλυση: Τα use cases αποτελούν τη βάση για περαιτέρω σχεδίαση UML.
- Υποστήριξη Embedded Modeling: Παρόλο που το σύστημα είναι απλό, η UML επιτρέπει τη βασική αναπαράσταση embedded behavior.

Παρά την πληρότητα του μοντέλου, υπάρχουν οι παρακάτω περιορισμοί:

- δεν αποτυπώνονται ηλεκτρικά χαρακτηριστικά,
- δεν περιγράφονται real-time constraints,
- δεν μοντελοποιείται η φυσική συμπεριφορά hardware,
- δεν περιλαμβάνονται hardware states.

Για πιο προηγμένη μοντελοποίηση embedded systems απαιτείται χρήση SysML ή UML profiles για real-time systems (Douglass, 2016).

Συμπεράσματα

Η δημιουργία του Use Case Model για το σύστημα Arduino LED στο Tinkercad επέτρεψε τη σαφή αναπαράσταση των λειτουργικών απαιτήσεων και των αλληλεπιδράσεων του συστήματος.

Το μοντέλο:

- είναι UML-consistent,
- παρουσιάζει υψηλή λειτουργική πληρότητα,
- υποστηρίζει περαιτέρω ανάλυση και σχεδίαση,
- αποτελεί κατάλληλη βάση για εκπαιδευτικές εφαρμογές embedded systems.

Η χρήση UML Use Case Modeling αποδεικνύεται ιδιαίτερα αποτελεσματική για την κατανόηση και τεκμηρίωση της λειτουργίας του συστήματος.

7.4 Δημιουργία Activity Diagrams με σημασιολογική τεκμηρίωση

Μελέτη Περίπτωσης 1: Προσομοίωση ενός ηλεκτρονικού κυκλώματος Arduino Uno με LED στο περιβάλλον Tinkercad

Στην παρούσα μελέτη περίπτωσης αναπτύχθηκε Activity Diagram για την προσομοίωση ενός ηλεκτρονικού κυκλώματος Arduino Uno με LED στο περιβάλλον Tinkercad. Στόχος του διαγράμματος είναι η σχεδίαση και η αναπαράσταση όλων των ενεργειών που απαιτούνται από τη δημιουργία του κυκλώματος μέχρι την επιτυχή εκτέλεση της προσομοίωσης και τη διόρθωση πιθανών σφαλμάτων. Η χρήση Activity Diagrams είναι ιδιαίτερα σημαντική στα embedded systems, καθώς επιτρέπει τη μοντελοποίηση της λειτουργικής συμπεριφοράς τόσο του λογισμικού όσο και των αλληλεπιδράσεων με το υλικό (Douglass, 2016).

Τα Activity Diagrams αποτελούν μία από τις βασικές κατηγορίες συμπεριφορικών διαγραμμάτων της UML. Σύμφωνα με την προδιαγραφή UML 2.5.1, χρησιμοποιούνται για την περιγραφή της ροής ελέγχου (control flow) και της ακολουθίας εκτέλεσης δραστηριοτήτων μέσα σε ένα σύστημα (OMG, 2023).

Ένα Activity Diagram περιλαμβάνει τα παρακάτω στοιχεία:

- Initial Node (κόμβος έναρξης)
- Activity Actions (ενέργειες)
- Control Flows (ροές ελέγχου)
- Decision Nodes (κόμβοι απόφασης)
- Merge Nodes
- Activity Final Nodes (τερματισμός)
- Partitions ή Swimlanes

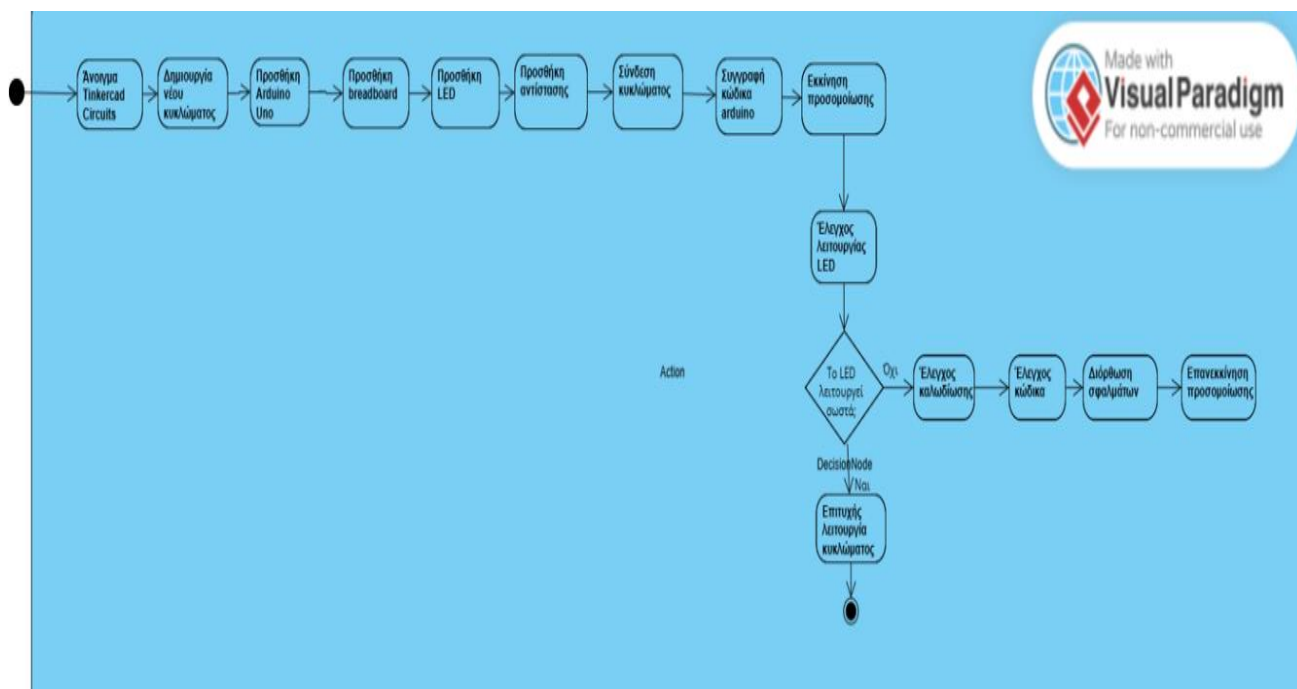
Η χρήση των παραπάνω στοιχείων επιτρέπει τη σαφή αποτύπωση της συμπεριφοράς ενός συστήματος και τη διάκριση των υπευθυνοτήτων μεταξύ των εμπλεκόμενων οντοτήτων (Sommerville, 2016).

Για τη μοντελοποίηση της συμπεριφοράς του συστήματος που παρουσιάστηκε παραπάνω, σχεδιάστηκε και δημιουργήθηκε Activity Diagram το οποίο περιγράφει:

1. Τη δημιουργία του κυκλώματος.
2. Τη σύνδεση των εξαρτημάτων.
3. Τη συγγραφή του κώδικα.
4. Την εκτέλεση της προσομοίωσης.
5. Τον έλεγχο της λειτουργίας του LED.
6. Τη διαδικασία διόρθωσης σφαλμάτων.

Δημιουργία του Activity Diagram

Η ανάπτυξη του Activity Diagram πραγματοποιήθηκε ακολουθώντας τις αρχές της UML και τη λογική ροή των λειτουργιών του συστήματος. Το διάγραμμα σχεδιάστηκε μέσω του λογισμικού Visual-Paradigm και παρουσιάζεται στην εικόνα 7.4.



Εικ.7- 4 Activity Diagram για Σύστημα Ελέγχου LED με Arduino Uno στο Tinkercad

- ✓ **Κόμβος Έναρξης:** Το διάγραμμα ξεκινά με έναν Initial Node, ο οποίος σηματοδοτεί την έναρξη της διαδικασίας μοντελοποίησης. Η πρώτη δραστηριότητα αφορά στην είσοδο του χρήστη στο περιβάλλον Tinkercad και τη δημιουργία νέου κυκλώματος.
- ✓ **Προσθήκη και Σύνδεση Εξαρτημάτων:** Περιγράφονται οι δραστηριότητες:
 - Προσθήκη Arduino Uno,
 - Προσθήκη breadboard,
 - Προσθήκη LED,
 - Προσθήκη αντίστασης,
 - Σύνδεση καλωδίων.

Οι δραστηριότητες αυτές αναπαριστούν τη φυσική συναρμολόγηση του κυκλώματος και αποτελούν απαραίτητη προϋπόθεση για την ορθή λειτουργία του συστήματος.

- ✓ **Συγγραφή Κώδικα:** Μετά την ολοκλήρωση των ηλεκτρικών συνδέσεων ακολουθεί η ανάπτυξη του προγράμματος Arduino. Η δραστηριότητα αυτή περιλαμβάνει:
 - άνοιγμα του Code Editor,
 - συγγραφή του κώδικα,

- έλεγχο σύνταξης.

Η διαδικασία αυτή συνδέει τη συμπεριφορά του λογισμικού με τη λειτουργία του φυσικού κυκλώματος.

- ✓ **Εκτέλεση Προσομοίωσης:** Μετά την ολοκλήρωση του κώδικα εκτελείται η προσομοίωση μέσω του Tinkercad. Η δραστηριότητα αυτή ενεργοποιεί την εκτέλεση του προγράμματος στο Arduino Uno και επιτρέπει την παρατήρηση της συμπεριφοράς του LED.
- ✓ **Έλεγχος Αποτελέσματος:** Στο Activity Diagram περιλαμβάνεται Decision Node με ερώτημα: «Το LED λειτουργεί σωστά;» Η απόφαση αυτή δημιουργεί δύο εναλλακτικές ροές:
 - Θετική ροή (ΝΑΙ)
 - Αρνητική ροή (ΟΧΙ)

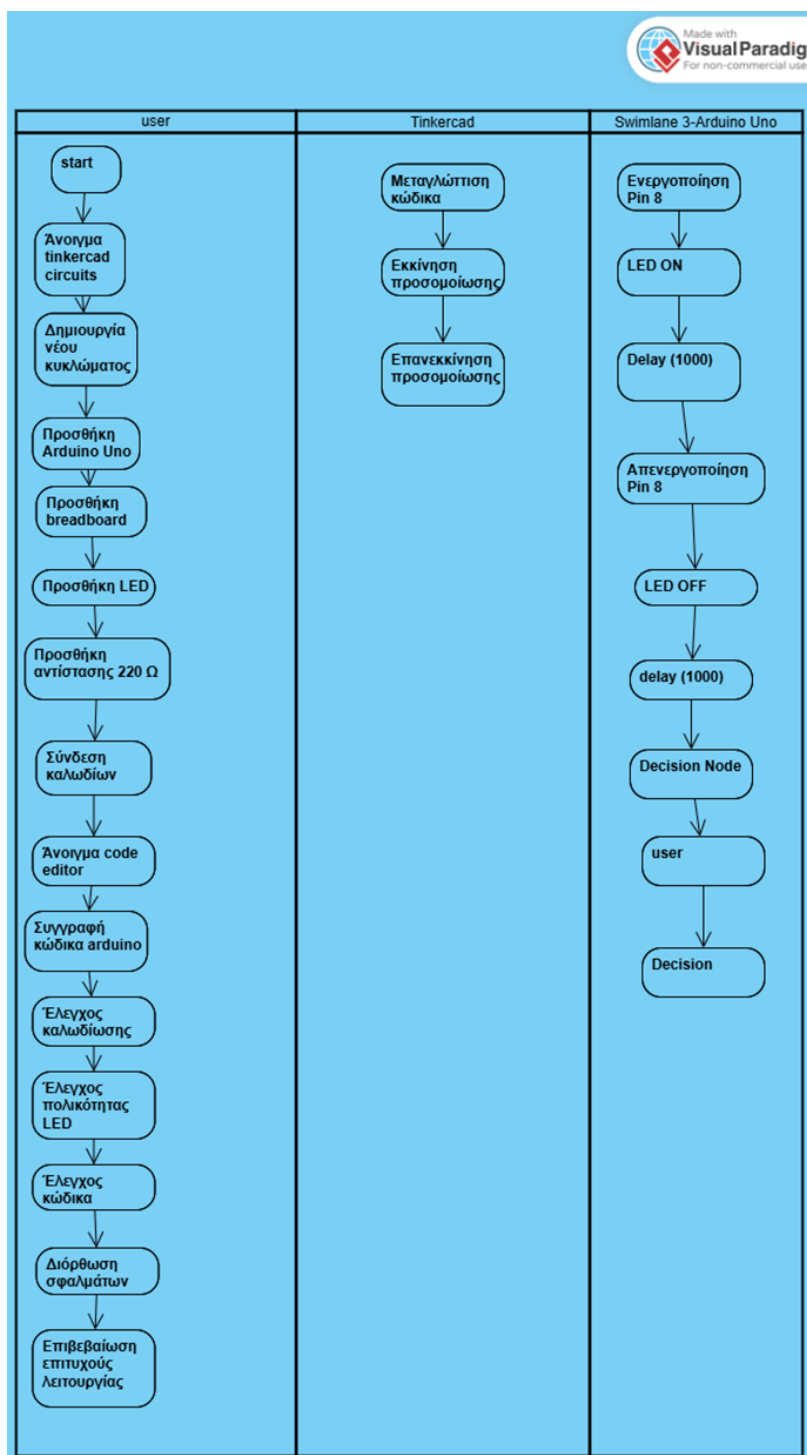
Η χρήση decision nodes θεωρείται κρίσιμη για την αποτύπωση εναλλακτικών συμπεριφορών και διαδικασιών ελέγχου (Brambilla, Cabot & Wimmer, 2017).

- ✓ **Διαχείριση Σφαλμάτων:** Στην περίπτωση αποτυχίας εκτελούνται οι δραστηριότητες:
 - Έλεγχος καλωδίωσης,
 - Έλεγχος πολικότητας LED,
 - Έλεγχος κώδικα,
 - Διόρθωση σφαλμάτων.

Μετά την ολοκλήρωση των διορθώσεων πραγματοποιείται επανεκκίνηση της προσομοίωσης. Η διαδικασία αυτή επιτρέπει την αναπαράσταση exception handling και debugging διαδικασιών μέσα στο Activity Diagram.

Χρήση Swimlanes για Κατανομή Ευθυνών

Για τη βελτίωση της αναγνωσιμότητας και της σημασιολογικής πληρότητας του μοντέλου χρησιμοποιήθηκαν Swimlanes (Partitions). Τα Swimlanes διαχωρίζουν τις δραστηριότητες ανάλογα με την οντότητα που τις εκτελεί. Στη συγκεκριμένη μελέτη περίπτωσης χρησιμοποιήθηκαν τρία Swimlanes και παρουσιάζονται στην εικόνα 2:



Εικ.7-5 Χρήση Swimlanes για Κατανομή Ευθυνών για Σύστημα Ελέγχου LED με Arduino Uno στο Tinkercad

✓ **Χρήστης:** Περιλαμβάνει δραστηριότητες όπως:

- δημιουργία κυκλώματος,
- προσθήκη εξαρτημάτων,
- συγγραφή κώδικα,
- έλεγχος λειτουργίας,
- διόρθωση σφαλμάτων.

✓ **Tinkercad:** Περιλαμβάνει:

- μεταγλώττιση κώδικα,
- εκκίνηση προσομοίωσης,
- επανεκκίνηση προσομοίωσης.

✓ **Arduino Uno:** Περιλαμβάνει:

- ενεργοποίηση Pin 8,
- LED ON,
- LED OFF,
- χρονικές καθυστερήσεις.

Η χρήση Swimlanes βελτιώνει σημαντικά τη σαφήνεια του διαγράμματος και υποστηρίζει την κατανομή ευθυνών μεταξύ των εμπλεκόμενων οντοτήτων (Dennis, Wixom & Tegarden, 2020).

Σημασιολογική Τεκμηρίωση του Activity Diagram

Η σημασιολογική τεκμηρίωση αφορά στην ερμηνεία κάθε δραστηριότητας και στη συμβολή της στη συνολική λειτουργία του συστήματος. Η σημασιολογική τεκμηρίωση του Activity Diagram παρουσιάζεται στον παρακάτω πίνακα:

Πίνακας 7-2. Σημασιολογική Τεκμηρίωση του Activity Diagram

Δραστηριότητα	Σημασιολογική Τεκμηρίωση
Άνοιγμα Tinkercad Circuits	Αντιπροσωπεύει την έναρξη της αλληλεπίδρασης του χρήστη με το περιβάλλον προσομοίωσης.
Δημιουργία Νέου Κυκλώματος	Αποτελεί τη δημιουργία του ψηφιακού workspace μέσα στο οποίο θα αναπτυχθεί το κύκλωμα.
Προσθήκη Arduino Uno	Εισάγει τον μικροελεγκτή που θα εκτελέσει τον κώδικα.
Προσθήκη LED και Αντίστασης	Αναπαριστά την ενσωμάτωση των ηλεκτρονικών εξαρτημάτων που θα χρησιμοποιηθούν για την υλοποίηση της εξόδου.
Συγγραφή Κώδικα Arduino	Αποτελεί το σημείο όπου καθορίζεται η λειτουργική συμπεριφορά του συστήματος.
Εκκίνηση Προσομοίωσης	Μεταφέρει το σύστημα από τη φάση της σχεδίασης στη φάση της εκτέλεσης.
Έλεγχος Λειτουργίας LED	Αποτελεί τη διαδικασία επαλήθευσης της σωστής λειτουργίας του κυκλώματος.
Διόρθωση Σφαλμάτων	Αναπαριστά τον μηχανισμό ανάδρασης και βελτίωσης του συστήματος.

Αξιολόγηση ως προς της Σημασιολογική Πληρότητα -Συμπεράσματα

Το παραπάνω Activity Diagram παρουσιάζει υψηλό βαθμό σημασιολογικής πληρότητας, καθώς αποτυπώνει:

- όλες τις κύριες λειτουργίες του συστήματος,
- τις εναλλακτικές ροές εκτέλεσης,

- τη διαδικασία ελέγχου και debugging,
- την κατανομή ευθυνών μέσω Swimlanes.

Ωστόσο, δεν αποτυπώνονται λεπτομέρειες όπως:

- καταστάσεις μικροελεγκτή,
- hardware interrupts,
- χρονικοί περιορισμοί πραγματικού χρόνου,
- ηλεκτρικά χαρακτηριστικά κυκλώματος.

Οι περιορισμοί αυτοί είναι αναμενόμενοι, καθώς η UML επικεντρώνεται κυρίως στη λειτουργική και συμπεριφορική αναπαράσταση και όχι στη φυσική προσομοίωση hardware (Douglass, 2016).

Συνολικά, το Activity Diagram αποδείχθηκε ιδιαίτερα χρήσιμο εργαλείο τεκμηρίωσης και ανάλυσης της λειτουργίας του συστήματος, παρέχοντας μια σαφή και ολοκληρωμένη εικόνα της συμπεριφοράς του. Η ανάπτυξη Activity Diagram με Swimlanes για το κύκλωμα Arduino Uno-LED επέτρεψε την αποτελεσματική αναπαράσταση της λειτουργικής και συμπεριφορικής λογικής του συστήματος. Η χρήση Swimlanes συνέβαλε στη σαφή κατανομή ευθυνών μεταξύ χρήστη, περιβάλλοντος προσομοίωσης και μικροελεγκτή, ενώ η χρήση decision nodes επέτρεψε τη μοντελοποίηση διαδικασιών ελέγχου και διαχείρισης σφαλμάτων.

7.5 Αξιολόγηση των Μοντέλων ως προς τη Σημασιολογική Πληρότητα

Η ποιότητα ενός μοντέλου λογισμικού δεν εξαρτάται μόνο από τη συντακτική ορθότητά του, αλλά και από τον βαθμό στον οποίο αποτυπώνει με ακρίβεια και πληρότητα τη σημασία (semantics) του συστήματος που αναπαριστά. Η έννοια της σημασιολογικής πληρότητας (semantic completeness) αναφέρεται στην ικανότητα ενός μοντέλου να περιγράφει όλες τις απαραίτητες λειτουργίες, συμπεριφορές, αλληλεπιδράσεις και περιορισμούς του υπό μελέτη συστήματος (Brambilla, Cabot & Wimmer, 2017).

Στην παρούσα εργασία αναπτύχθηκαν δύο βασικές κατηγορίες UML μοντέλων:

- Use Case Model
- Activity Diagram με Swimlanes

Τα μοντέλα αυτά χρησιμοποιήθηκαν για την αναπαράσταση ενός εκπαιδευτικού embedded συστήματος που βασίζεται σε Arduino Uno και LED, στο περιβάλλον προσομοίωσης Tinkercad.

Στόχος της παρούσας ενότητας είναι η αξιολόγηση των μοντέλων ως προς τη σημασιολογική τους πληρότητα, δηλαδή κατά πόσο αποτυπώνουν αποτελεσματικά τις απαιτήσεις, τη συμπεριφορά και τις αλληλεπιδράσεις του συστήματος.

Η έννοια της σημασιολογικής πληρότητας

Σύμφωνα με το Object Management Group (OMG), ένα UML μοντέλο θεωρείται πλήρες όταν αποτυπώνει τις ουσιώδεις πτυχές του συστήματος χωρίς να παραλείπει κρίσιμες λειτουργίες ή σχέσεις (OMG, 2023).

Η σημασιολογική πληρότητα αξιολογείται συνήθως μέσω των παρακάτω διαστάσεων:

- Functional Completeness
- Behavioral Completeness
- Actor Completeness
- Interaction Completeness
- Exception Handling Completeness
- Domain Semantics Representation

Η ύπαρξη υψηλής σημασιολογικής πληρότητας συμβάλλει στη βελτίωση της ποιότητας των μοντέλων και στη μείωση των παρερμηνειών κατά τη διαδικασία ανάπτυξης λογισμικού (Dennis, Wixom & Tegarden, 2020).

Αξιολόγηση του Use Case Model

✓ Λειτουργική Πληρότητα

Το Use Case Model που αναπτύχθηκε περιλαμβάνει όλες τις βασικές λειτουργίες του συστήματος:

- Δημιουργία νέου κυκλώματος
- Προσθήκη εξαρτημάτων
- Σύνδεση κυκλώματος
- Συγγραφή κώδικα
- Εκκίνηση προσομοίωσης
- Έλεγχος λειτουργίας LED

- Διόρθωση σφαλμάτων

Η κάλυψη των παραπάνω λειτουργιών επιτρέπει την πλήρη αναπαράσταση του κύκλου ζωής της εφαρμογής, από τη δημιουργία έως τον έλεγχο της λειτουργίας της.

Σύμφωνα με τους Booch, Rumbaugh και Jacobson (2005), ένα Use Case Model θεωρείται λειτουργικά πλήρες όταν αποτυπώνει όλες τις υπηρεσίες που προσφέρει το σύστημα στους χρήστες του. Με βάση αυτό το κριτήριο, το συγκεκριμένο μοντέλο παρουσιάζει υψηλό βαθμό λειτουργικής πληρότητας.

✓ Πληρότητα Actors

Το μοντέλο περιλαμβάνει τρεις βασικούς actors:

- Χρήστης / Μαθητής
- Tinkercad
- Arduino Uno

Η επιλογή των actors επιτρέπει τη σαφή αναπαράσταση των αλληλεπιδράσεων μεταξύ ανθρώπου, λογισμικού προσομοίωσης και μικροελεγκτή.

Η ύπαρξη ξεχωριστού actor για το Arduino Uno είναι ιδιαίτερα σημαντική, καθώς αναδεικνύει τον ρόλο του embedded hardware ως ενεργού μέρους της λειτουργίας του συστήματος (Douglass, 2016).

✓ Σημασιολογική Ορθότητα των Σχέσεων

Το Use Case Diagram περιλαμβάνει σχέσεις:

- Association
- Include
- Extend

Η σχέση μεταξύ των use cases «Εκκίνηση Προσομοίωσης» και «Έλεγχος Λειτουργίας LED» αποτυπώνει σωστά μία υποχρεωτική λειτουργική εξάρτηση. Παράλληλα, οι σχέσεις που συνδέουν τη «Διόρθωση Σφαλμάτων» με τη «Σύνδεση Κυκλώματος» και τη «Συγγραφή Κώδικα» περιγράφουν με ορθό τρόπο εξαιρεσιακές ή εναλλακτικές ροές λειτουργίας. Η χρήση αυτών των σχέσεων συμβαδίζει πλήρως με τη σημασιολογία της UML 2.5.1 (OMG, 2023).

✓ Περιορισμοί του Use Case Model

Παρά τον υψηλό βαθμό πληρότητας, το μοντέλο δεν αποτυπώνει:

- ηλεκτρικά χαρακτηριστικά κυκλώματος,
- χρονικές απαιτήσεις εκτέλεσης,
- hardware constraints,
- καταστάσεις λειτουργίας του μικροελεγκτή.

Οι παραπάνω πληροφορίες υπερβαίνουν το πεδίο εφαρμογής ενός Use Case Diagram και συνήθως αποτυπώνονται μέσω άλλων UML ή SysML μοντέλων (Friedenthal, Moore & Steiner, 2015).

Αξιολόγηση του Activity Diagram

✓ Συμπεριφορική Πληρότητα

Το Activity Diagram αναπαριστά ολόκληρη τη ροή λειτουργίας του συστήματος.

Συγκεκριμένα περιλαμβάνει:

- Δημιουργία κυκλώματος.
- Προσθήκη εξαρτημάτων.
- Σύνδεση καλωδίων.
- Συγγραφή κώδικα.
- Μεταγλώττιση.
- Εκκίνηση προσομοίωσης.
- Εκτέλεση από το Arduino.
- Έλεγχο αποτελέσματος.
- Διαχείριση σφαλμάτων.

Η ακολουθία αυτή αντιστοιχεί πλήρως στη λειτουργική διαδικασία που ακολουθεί ο χρήστης στο περιβάλλον Tinkercad. Σύμφωνα με τον Sommerville (2016), η πλήρης αναπαράσταση της ροής εργασίας αποτελεί βασικό κριτήριο αξιολόγησης των Activity Diagrams.

✓ Χρήση Swimlanes και Κατανομή Ευθυνών

Η χρήση Swimlanes προσφέρει επιπλέον σημασιολογική πληροφορία μέσω της κατανομής ευθυνών. Οι δραστηριότητες κατανέμονται σε:

Χρήστης

- Δημιουργία κυκλώματος
- Προσθήκη εξαρτημάτων
- Συγγραφή κώδικα
- Debugging

Tinkercad

- Μεταγλώττιση
- Εκκίνηση προσομοίωσης

Arduino Uno

- Ενεργοποίηση Pin 8
- LED ON/OFF
- Χρονικές καθυστερήσεις

Η κατανομή αυτή βελτιώνει τη σαφήνεια του μοντέλου και διευκολύνει την κατανόηση των αλληλεπιδράσεων μεταξύ των επιμέρους οντοτήτων (Dennis et al., 2020).

✓ Αναπαράσταση Εναλλακτικών Ροών

Η ύπαρξη Decision Node με ερώτημα: «*Το LED λειτουργεί σωστά;*» επιτρέπει την αναπαράσταση δύο διαφορετικών σεναρίων:

- επιτυχής λειτουργία,
- διαδικασία διόρθωσης σφαλμάτων.

Η μοντελοποίηση exception flows αποτελεί σημαντικό στοιχείο σημασιολογικής πληρότητας, καθώς αναπαριστά όχι μόνο την ιδανική λειτουργία αλλά και τις αποκλίσεις από αυτήν (Brambilla et al., 2017).

✓ Περιορισμοί του Activity Diagram

Παρά τη σημαντική του πληρότητα, το Activity Diagram δεν αποτυπώνει:

- hardware interrupts,
- πραγματικό χρόνο εκτέλεσης,
- εσωτερικές καταστάσεις καταχωρητών,
- ηλεκτρικές μεταβολές τάσης και ρεύματος.

Οι πληροφορίες αυτές απαιτούν εξειδικευμένα μοντέλα όπως State Machine Diagrams, Timing Diagrams ή SysML Block Definition Diagrams (Douglass, 2016).

Συγκριτική Αξιολόγηση των Μοντέλων - Συμπεράσματα

Τα δύο μοντέλα καλύπτουν διαφορετικές αλλά συμπληρωματικές πτυχές του συστήματος. Το Use Case Model επικεντρώνεται στις λειτουργικές απαιτήσεις και στις αλληλεπιδράσεις χρηστών-συστήματος, ενώ το Activity Diagram περιγράφει τη συμπεριφορά και τη ροή εκτέλεσης των δραστηριοτήτων. Ο συνδυασμός των δύο μοντέλων προσφέρει σημαντικά υψηλότερη σημασιολογική πληρότητα σε σχέση με τη χρήση ενός μόνο μοντέλου.

Στον παρακάτω πίνακα παρουσιάζονται τα δυνατά και τα λιγότερο δυνατά σημεία των δύο μοντέλων:

Πίνακας 7-3 Βαθμός κάλυψης των κριτηρίων για κάθε μοντέλο

Κριτήριο	Use case Model	Activity Diagram
Λειτουργικές απαιτήσεις	Υψηλή κάλυψη	Μέτρια κάλυψη
Συμπεριφορά Συστήματος	Περιορισμένη	Υψηλή κάλυψη
Actors	Υψηλή κάλυψη	Μέτρια κάλυψη
Ροή Ελέγχου	Χαμηλή κάλυψη	Υψηλή κάλυψη
Exception Handling	Μέτρια κάλυψη	Υψηλή κάλυψη
Κατανομή Ευθυνών	Περιορισμένη	Υψηλή κάλυψη

Από τα παραπάνω προκύπτει ό,τι η συμπληρωματική χρήση των δύο μοντέλων αυξάνει σημαντικά τη συνολική αναπαράσταση της σημασίας του συστήματος. Παρά τους περιορισμούς τους ως προς τη μοντελοποίηση hardware-level και real-time χαρακτηριστικών, τα δύο μοντέλα κρίνονται επαρκή για την αναπαράσταση των βασικών λειτουργιών και της συμπεριφοράς του συστήματος, αποτελώντας μια ισχυρή βάση για περαιτέρω UML ή SysML ανάλυση. Το Use Case Model αποτυπώνει αποτελεσματικά τις λειτουργικές απαιτήσεις και τις αλληλεπιδράσεις μεταξύ των εμπλεκόμενων οντοτήτων, ενώ το Activity Diagram αναπαριστά με σαφήνεια τη συμπεριφορά και τη ροή λειτουργίας του συστήματος. Η χρήση Swimlanes ενισχύει περαιτέρω τη σημασιολογική αξία του Activity Diagram, καθώς επιτρέπει την ορθή κατανομή ευθυνών μεταξύ χρήστη, περιβάλλοντος προσομοίωσης και μικροελεγκτή.

7.6 Συμπεράσματα της Μελέτης Περίπτωσης

Η παρούσα μελέτη περίπτωσης είχε ως αντικείμενο τη μοντελοποίηση ενός εκπαιδευτικού embedded συστήματος βασισμένου σε Arduino Uno και LED στο περιβάλλον προσομοίωσης Tinkercad. Η διαδικασία περιλάμβανε την ανάλυση των λειτουργικών απαιτήσεων, τη δημιουργία Use Case Models, την ανάπτυξη Activity Diagrams με Swimlanes και την αξιολόγηση των παραγόμενων μοντέλων ως προς τη σημασιολογική τους πληρότητα.

Η χρήση UML στη μοντελοποίηση συστημάτων λογισμικού και ενσωματωμένων συστημάτων έχει αναγνωριστεί ως μία από τις πιο αποτελεσματικές μεθόδους τεκμηρίωσης και ανάλυσης, καθώς επιτρέπει την αναπαράσταση τόσο των λειτουργικών απαιτήσεων όσο και της συμπεριφοράς των συστημάτων με τυποποιημένο τρόπο (OMG, 2023). Η συγκεκριμένη μελέτη επιβεβαίωσε τη χρησιμότητα της UML ως εργαλείου κατανόησης, σχεδίασης και αξιολόγησης εκπαιδευτικών εφαρμογών που συνδυάζουν στοιχεία λογισμικού και υλικού.

Συμπεράσματα από την Ανάλυση Απαιτήσεων

Ένα από τα σημαντικότερα συμπεράσματα της μελέτης είναι ότι η διαδικασία ανάλυσης απαιτήσεων συνέβαλε ουσιαστικά στην κατανόηση της λειτουργίας του συστήματος και στον εντοπισμό των βασικών αλληλεπιδράσεων μεταξύ των εμπλεκόμενων οντοτήτων.

Η αναγνώριση των actors (Χρήστης, Tinkercad και Arduino Uno) καθώς και των βασικών λειτουργιών του συστήματος επέτρεψε τη δημιουργία ενός ολοκληρωμένου Use Case Model, το οποίο περιέγραψε με σαφήνεια τον τρόπο με τον οποίο ο χρήστης αλληλεπιδρά με το περιβάλλον προσομοίωσης και το embedded hardware.

Σύμφωνα με τους (Jacobson et al., 1992· Dennis, Wixom & Tegarden, 2020), η ανάλυση απαιτήσεων μέσω Use Cases διευκολύνει σημαντικά τη μετάβαση από την περιγραφή των απαιτήσεων στη σχεδίαση του συστήματος, μειώνοντας την πιθανότητα παρερμηνειών και αστοχιών κατά την ανάπτυξη.

Συμπεράσματα από τη Δημιουργία των Use Case Models

Η ανάπτυξη του Use Case Diagram ανέδειξε τη σημασία της λειτουργικής μοντελοποίησης ως μέσου καταγραφής των υπηρεσιών που παρέχει το σύστημα.

Το μοντέλο κατέγραψε με σαφήνεια:

- τη δημιουργία του κυκλώματος,
- την προσθήκη και σύνδεση των εξαρτημάτων,
- τη συγγραφή κώδικα,
- την εκτέλεση προσομοίωσης,
- τον έλεγχο λειτουργίας του LED,
- τη διαδικασία διόρθωσης σφαλμάτων.

Η χρήση των σχέσεων συνέβαλε στη σωστή αναπαράσταση υποχρεωτικών και εναλλακτικών λειτουργιών, γεγονός που ενίσχυσε τη σαφήνεια και την εκφραστικότητα του μοντέλου.

Τα αποτελέσματα αυτά συμφωνούν με τα ευρήματα των Booch, Rumbaugh και Jacobson (2005), σύμφωνα με τα οποία τα Use Case Models αποτελούν θεμελιώδες εργαλείο για τη μοντελοποίηση λειτουργικών απαιτήσεων και τη διασύνδεση των απαιτήσεων με τη σχεδίαση του συστήματος.

Συμπεράσματα από τη Δημιουργία των Activity Diagrams

Η ανάπτυξη των Activity Diagrams επέτρεψε την αναπαράσταση της συμπεριφοράς του συστήματος και της ακολουθίας των δραστηριοτήτων που εκτελούνται κατά τη λειτουργία του.

Το Activity Diagram περιέγραψε:

- τη διαδικασία δημιουργίας του κυκλώματος,
- τη σύνδεση των ηλεκτρονικών εξαρτημάτων,
- τη συγγραφή και μεταγλώττιση του κώδικα,
- την εκτέλεση της προσομοίωσης,
- τον έλεγχο της λειτουργίας του LED,
- τις διαδικασίες debugging.

Ιδιαίτερα σημαντική αποδείχθηκε η χρήση Swimlanes, οι οποίες επέτρεψαν την κατανομή των δραστηριοτήτων μεταξύ των εμπλεκόμενων οντοτήτων. Η διάκριση μεταξύ Χρήστη, Tinkercad και Arduino Uno συνέβαλε στη βελτίωση της αναγνωσιμότητας του μοντέλου και στην καλύτερη κατανόηση των αλληλεπιδράσεων μεταξύ λογισμικού και υλικού. Η σημασία των Swimlanes ως μηχανισμού κατανομής ευθυνών έχει επισημανθεί και στη διεθνή βιβλιογραφία, καθώς διευκολύνουν την κατανόηση πολύπλοκων διαδικασιών και υποστηρίζουν την ανάλυση οργανωτικών και τεχνικών ροών εργασίας (Dennis et al., 2020).

Συμπεράσματα από την Αξιολόγηση της Σημασιολογικής Πληρότητας

Η αξιολόγηση των μοντέλων έδειξε ότι τόσο το Use Case Diagram όσο και το Activity Diagram παρουσιάζουν υψηλό βαθμό σημασιολογικής πληρότητας για το συγκεκριμένο εκπαιδευτικό σύστημα.

Το Use Case Model αποτυπώνει αποτελεσματικά:

- τις λειτουργικές απαιτήσεις,
- τους εμπλεκόμενους actors,
- τις σχέσεις μεταξύ λειτουργιών.

Αντίστοιχα, το Activity Diagram αποτυπώνει:

- τη ροή ελέγχου,
- τις αλληλεπιδράσεις μεταξύ των οντοτήτων,
- τις διαδικασίες λήψης αποφάσεων,
- τους μηχανισμούς διαχείρισης σφαλμάτων.

Η συνδυαστική χρήση των δύο μοντέλων προσέφερε μια περισσότερο ολοκληρωμένη αναπαράσταση του συστήματος, γεγονός που επιβεβαιώνει τη συμπληρωματικότητα των διαφορετικών τύπων UML μοντέλων, όπως αναφέρεται και από τους Brambilla, Cabot και Wimmer (2017).

Παράλληλα, εντοπίστηκαν ορισμένοι περιορισμοί, καθώς τα μοντέλα δεν αποτυπώνουν:

- ηλεκτρικά χαρακτηριστικά του κυκλώματος,
- χρονικές απαιτήσεις πραγματικού χρόνου,
- καταστάσεις μικροελεγκτή,

- hardware interrupts.

Σύμφωνα με τους (Friedenthal, Moore & Steiner, 2015), οι περιορισμοί αυτοί είναι σύμφωνοι με τα γνωστά όρια της UML στη μοντελοποίηση embedded συστημάτων, όπου συχνά απαιτείται η χρήση επεκτάσεων όπως η SysML για την αναπαράσταση φυσικών και τεχνικών χαρακτηριστικών του υλικού.

Συμβολή της UML στη Μοντελοποίηση Embedded Συστημάτων

Η μελέτη επιβεβαίωσε ότι η UML μπορεί να αποτελέσει ένα ιδιαίτερα χρήσιμο εργαλείο για την ανάλυση και τεκμηρίωση εκπαιδευτικών embedded συστημάτων.

Η χρήση UML προσφέρει:

- τυποποιημένη αναπαράσταση απαιτήσεων,
- σαφή περιγραφή συμπεριφοράς,
- βελτιωμένη επικοινωνία μεταξύ αναλυτών και χρηστών,
- ευκολότερη τεκμηρίωση και συντήρηση του συστήματος.

Παρόλο που η UML σχεδιάστηκε αρχικά για την ανάπτυξη λογισμικού, η εφαρμογή της σε συστήματα Arduino και γενικότερα σε embedded platforms αποδεικνύεται ιδιαίτερα αποτελεσματική όταν χρησιμοποιείται σε συνδυασμό με κατάλληλες τεχνικές μοντελοποίησης (Douglass, 2016).

Εκπαιδευτική και Ερευνητική Αξία της Μελέτης

Η συγκεκριμένη μελέτη περίπτωσης παρουσιάζει σημαντική εκπαιδευτική αξία, καθώς συνδυάζει τρία διαφορετικά γνωστικά αντικείμενα:

- ηλεκτρονική και κυκλώματα,
- προγραμματισμό μικροελεγκτών,
- τεχνικές μοντελοποίησης λογισμικού.

Η προσέγγιση αυτή επιτρέπει στους εκπαιδευόμενους να αναπτύξουν δεξιότητες συστημικής σκέψης και να κατανοήσουν καλύτερα τη σχέση μεταξύ σχεδίασης, υλοποίησης και λειτουργίας ενός συστήματος.

Από ερευνητικής πλευράς, η μελέτη αναδεικνύει τη δυνατότητα αξιοποίησης UML τεχνικών σε απλά embedded systems και δημιουργεί προϋποθέσεις για μελλοντική διερεύνηση πιο σύνθετων εφαρμογών που περιλαμβάνουν αισθητήρες, ασύρματες επικοινωνίες και Internet of Things (IoT) αρχιτεκτονικές.

Σύνοψη

Από την παρούσα μελέτη περίπτωσης φαίνεται ότι η χρήση UML αποτελεί αποτελεσματική προσέγγιση για την ανάλυση, τεκμηρίωση και κατανόηση εκπαιδευτικών embedded συστημάτων. Τα Use Case Models και τα Activity Diagrams που αναπτύχθηκαν κατέγραψαν με σαφήνεια τόσο τις λειτουργικές απαιτήσεις όσο και τη συμπεριφορά του συστήματος, παρουσιάζοντας υψηλό βαθμό σημασιολογικής πληρότητας.

Η συνδυαστική χρήση των δύο μοντέλων παρείχε μια ολοκληρωμένη αναπαράσταση του συστήματος και ανέδειξε τη σημασία της UML ως εργαλείου γεφύρωσης μεταξύ θεωρητικής ανάλυσης και πρακτικής υλοποίησης. Παρά τους περιορισμούς της στη μοντελοποίηση hardware-level χαρακτηριστικών, η UML φαίνεται επαρκής για τη μοντελοποίηση της συγκεκριμένης εφαρμογής και μπορεί να αποτελέσει τη βάση για περαιτέρω ανάπτυξη και μελέτη πιο σύνθετων embedded συστημάτων.

Προτάσεις για Μελλοντική Εργασία

Μελλοντική επέκταση της παρούσας εργασίας θα μπορούσε να περιλαμβάνει:

- ανάπτυξη State Machine Diagrams για την αναπαράσταση των καταστάσεων του LED,
- δημιουργία Sequence Diagrams για τη μοντελοποίηση χρονικών αλληλεπιδράσεων,
- αξιοποίηση SysML για τη μοντελοποίηση hardware χαρακτηριστικών,
- ανάπτυξη πιο σύνθετων κυκλωμάτων με αισθητήρες και πολλαπλές συσκευές,
- εφαρμογή τεχνικών Model-Driven Engineering (MDE) για αυτόματη παραγωγή κώδικα από UML μοντέλα.

Οι προσεγγίσεις αυτές θα μπορούσαν να αυξήσουν περαιτέρω τη σημασιολογική πληρότητα των μοντέλων και να ενισχύσουν τη σύνδεση μεταξύ μοντελοποίησης και υλοποίησης.

8. Μελέτη Περίπτωσης 2: Μοντελοποίηση πειραματικών διαδικασιών σε εικονικό εργαστήριο

8.1 2^η Μελέτη Περίπτωσης: Περιγραφή του σεναρίου του Πειράματος της Άνωσης στο Εικονικό Εργαστήριο Klett / Publior Virtual Lab

Το πείραμα της Άνωσης υλοποιείται στο εικονικό εργαστήριο Publior Virtual Lab, το οποίο είναι ενσωματωμένο στο ψηφιακό εκπαιδευτικό περιβάλλον της Klett για τη Φυσική της Β' Γυμνασίου. Το περιβάλλον αυτό παρέχει στους μαθητές τη δυνατότητα να εκτελούν εικονικά πειράματα, να πραγματοποιούν μετρήσεις, να καταγράφουν δεδομένα και να εξάγουν επιστημονικά συμπεράσματα χωρίς την ανάγκη φυσικού εργαστηριακού εξοπλισμού.

Η δραστηριότητα συνδυάζει τη θεωρητική διδασκαλία της Αρχής του Αρχιμήδη με τη διερευνητική μάθηση, επιτρέποντας στους μαθητές να αλληλεπιδρούν ενεργά με το εικονικό εργαστήριο και να ανακαλύπτουν μόνοι τους τις σχέσεις μεταξύ φυσικών μεγεθών.

Πείραμα: Άνωση – Αρχή του Αρχιμήδη
Εικονικό εργαστήριο Klett Physics – Β' Γυμνασίου

- Στόχος**
Να επαληθεύσουμε την Αρχή του Αρχιμήδη:
«Ένα σώμα που βυθίζεται σε υγρό δέχεται άνωση ίση με το βάρος του υγρού που εκτοπίζει.»
- Όργανα – Υλικά**
 - Δυναμόμετρο
 - Ογκομετρικός κύλινδρος ή δοχείο υπερχειλίσσης
 - Νερό
 - Μεταλλικό σώμα (κύβος ή κύλινδρος με γαντζάκι)
 - Ζυγαριά (για μέτρηση μάζας εκτοπισμένου νερού)
- Διαδικασία (Βήμα-βήμα)**
 - Μέτρηση βάρους σώματος στον αέρα**
 - Κρεμάμε το σώμα στο δυναμόμετρο και καταγράφουμε την ένδειξη. Αυτή είναι το βάρος του σώματος στον αέρα ($B_{\alpha\epsilon\rho\alpha}$).
 - Μέτρηση βάρους σώματος μέσα στο νερό**
 - Βυθίζουμε πλήρως το σώμα μέσα στο νερό (χωρίς να ακουμπά στα τοιχώματα ή στον πυθμένα) και καταγράφουμε τη νέα ένδειξη του δυναμόμετρου. Αυτή είναι το φαινόμενο βάρος του σώματος μέσα στο νερό ($B_{\upsilon\gamma\rho\rho\upsilon}$).
 - Υπολογισμός άνωσης**
 - Η άνωση (A) που δέχεται το σώμα δίνεται από τη διαφορά:
 $A = B_{\alpha\epsilon\rho\alpha} - B_{\upsilon\gamma\rho\rho\upsilon}$
 - Μέτρηση εκτοπισμένου νερού**
 - Χρησιμοποιούμε ογκομετρικό κύλινδρο ή δοχείο υπερχειλίσσης.
 - Συλλέγουμε το νερό που εκτοπίζεται όταν βυθίζουμε το σώμα.
 - Μετράμε τον όγκο (V) του εκτοπισμένου νερού.
 - Υπολογισμός βάρους εκτοπισμένου νερού**
 - Υπολογίζουμε το βάρος του εκτοπισμένου νερού:
 $B_{\epsilon\kappa\tau.\ \nu\epsilon\rho\upsilon} = \rho_{\nu\epsilon\rho\upsilon} \cdot g \cdot V$
 - Σύγκριση**
 - Συγκρίνουμε την άνωση (A) με το βάρος του εκτοπισμένου νερού.
 - Αναμένουμε $A = B_{\epsilon\kappa\tau.\ \nu\epsilon\rho\upsilon}$ σύμφωνα με την Αρχή του Αρχιμήδη.
- Συμπέρασμα**
Αν η άνωση που υπολογίσαμε ισούται (εντός πειραματικού σφάλματος) με το βάρος του εκτοπισμένου νερού, τότε επιβεβαιώνεται η Αρχή του Αρχιμήδη.

Klett Physics – Εικονικό Εργαστήριο Άνωση – Αρχή του Αρχιμήδη

1. Βάρος στον αέρα

Ένδειξη: 2,94 N

2. Βάρος μέσα στο νερό

Ένδειξη: 1,76 N

3. Δοχείο υπερχειλίσσης

4. Εκτοπισμένο νερό

Όγκος: 60 mL

Υπολογισμοί

Βάρος στον αέρα ($B_{\alpha\epsilon\rho\alpha}$): 2,94 N
 Φαινόμενο βάρος στο νερό ($B_{\upsilon\gamma\rho\rho\upsilon}$): 1,76 N
 Άνωση (A): $2,94 - 1,76 = 1,18\ \text{N}$
 Όγκος εκτοπισμένου νερού (V): $60\ \text{mL} = 60\ \text{cm}^3 = 6,0 \times 10^{-5}\ \text{m}^3$
 Βάρος εκτοπισμένου νερού:
 $B_{\epsilon\kappa\tau.\ \nu\epsilon\rho\upsilon} = \rho \cdot g \cdot V = 1000\ \text{kg/m}^3 \cdot 9,81\ \text{m/s}^2 \cdot 6,0 \times 10^{-5}\ \text{m}^3 = 1,18\ \text{N}$
Άρα $A = B_{\epsilon\kappa\tau.\ \nu\epsilon\rho\upsilon} \rightarrow$ Επαληθεύεται η Αρχή του Αρχιμήδη.

Τι παρατηρούμε;

- Το σώμα δέχεται μικρότερη ένδειξη (βάρος) μέσα στο νερό.
- Το νερό που εκτοπίζεται έχει βάρος ίσο με την άνωση.
- Όσο μεγαλύτερος ο όγκος του εκτοπισμένου νερού, τόσο μεγαλύτερη η άνωση.

Εικ.8-1 Πείραμα: Άνωση-Αρχή του Αρχιμήδη-Εικονικό Εργαστήριο Klett Physics- Β' Γυμνασίου

- **Είσοδος στο Ψηφιακό Περιβάλλον**

Η διαδικασία ξεκινά με την είσοδο του μαθητή στο ψηφιακό περιβάλλον της Klett.

Ο μαθητής:

1. Εκκινεί την εφαρμογή Klett Book-App.
2. Επιλέγει το μάθημα της Φυσικής Β΄ Γυμνασίου.
3. Μεταβαίνει στο αντίστοιχο κεφάλαιο που αφορά την Άνωση και την Αρχή του Αρχιμήδη.
4. Επιλέγει το σύνδεσμο προς το Publior Virtual Lab.
5. Ανοίγει το εικονικό εργαστήριο του πειράματος.

Κατά τη διάρκεια αυτής της διαδικασίας το σύστημα φορτώνει τα απαραίτητα γραφικά στοιχεία, τα εικονικά όργανα μέτρησης και τις οδηγίες εκτέλεσης του πειράματος.

- **Εξοικείωση με το Εικονικό Εργαστήριο**

Με την είσοδό του στο εργαστήριο, ο μαθητής παρατηρεί την εικονική διάταξη του πειράματος.

Στην οθόνη εμφανίζονται:

- το δυναμόμετρο,
- το μεταλλικό σώμα,
- το δοχείο με νερό,
- το δοχείο υπερχειλίσης,
- ο ογκομετρικός κύλινδρος,
- τα εργαλεία καταγραφής μετρήσεων.

Ο μαθητής μπορεί να επιλέγει αντικείμενα με το ποντίκι και να τα μετακινεί στον εικονικό χώρο εργασίας. Το περιβάλλον παρέχει άμεση ανατροφοδότηση για κάθε ενέργεια που πραγματοποιείται.

- **Αλληλεπίδραση με τα Εικονικά Όργανα**

Πριν από την εκτέλεση του πειράματος ο μαθητής αλληλεπιδρά με τα διαθέσιμα όργανα ώστε να κατανοήσει τη λειτουργία τους.

Συγκεκριμένα:

- επιλέγει το δυναμόμετρο και παρατηρεί την κλίμακα μέτρησης,

- εξετάζει το σώμα που θα χρησιμοποιηθεί,
- παρατηρεί το δοχείο με το νερό,
- εξοικειώνεται με τον τρόπο μέτρησης του εκτοπισμένου νερού.

Η διαδικασία αυτή συμβάλλει στην ανάπτυξη δεξιοτήτων εργαστηριακού χειρισμού και προετοιμάζει τον μαθητή για τη σωστή εκτέλεση του πειράματος.

Εκτέλεση του Πειράματος

- **Μέτρηση Βάρους στον Αέρα**

Ο μαθητής σύρει το μεταλλικό σώμα και το τοποθετεί στο δυναμόμετρο.

Το εικονικό όργανο εμφανίζει την ένδειξη του βάρους του σώματος στον αέρα.

Η τιμή καταγράφεται στο φύλλο εργασίας ή στο Physics Lab του συστήματος.

- **Βύθιση του Σώματος στο Νερό**

Στη συνέχεια ο μαθητής μετακινεί το σώμα μέσα στο δοχείο με το νερό.

Το εικονικό περιβάλλον προσομοιώνει τη βύθιση και ενημερώνει δυναμικά την ένδειξη του δυναμόμετρου.

Ο μαθητής παρατηρεί ότι η ένδειξη μειώνεται.

- **Καταγραφή Φαινομένου Βάρους**

Η νέα ένδειξη του δυναμόμετρου αντιστοιχεί στο φαινόμενο βάρος του σώματος μέσα στο νερό.

Η τιμή καταγράφεται για χρήση στους επόμενους υπολογισμούς.

- **Μέτρηση Εκτοπισμένου Νερού**

Κατά τη βύθιση του σώματος το σύστημα προσομοιώνει την εκτόπιση νερού.

Ο μαθητής παρατηρεί την ποσότητα νερού που μεταφέρεται στο δοχείο συλλογής και διαβάσει τον όγκο του εκτοπισμένου νερού στον ογκομετρικό κύλινδρο.

Η μέτρηση αυτή καταγράφεται επίσης στο φύλλο εργασίας.

- **Υπολογισμοί και Επεξεργασία Δεδομένων**

Μετά την ολοκλήρωση των μετρήσεων ο μαθητής χρησιμοποιεί τα δεδομένα για να πραγματοποιήσει τους απαραίτητους υπολογισμούς.

Αρχικά υπολογίζει την άνωση:

Άνωση = Βάρος στον αέρα – Φαινόμενο βάρος

Στη συνέχεια υπολογίζει το βάρος του εκτοπισμένου νερού με βάση τον όγκο που μετρήθηκε.

Το περιβάλλον μπορεί είτε να παρέχει αυτόματα τους υπολογισμούς είτε να ζητά από τον μαθητή να τους εκτελέσει χειροκίνητα.

- **Σύγκριση Αποτελεσμάτων**

Ο μαθητής συγκρίνει:

- την τιμή της άνωσης,
- την τιμή του βάρους του εκτοπισμένου νερού.

Η σύγκριση αυτή αποτελεί το κεντρικό σημείο του πειράματος.

Εάν οι δύο τιμές είναι ίσες ή σχεδόν ίσες, επιβεβαιώνεται η θεωρία του Αρχιμήδη.

Εάν παρατηρηθούν αποκλίσεις, το σύστημα δίνει τη δυνατότητα επανάληψης των μετρήσεων.

- **Διατύπωση Συμπεράσματος**

Με βάση τα αποτελέσματα που προέκυψαν, ο μαθητής καλείται να διατυπώσει το τελικό συμπέρασμα.

Το συμπέρασμα που αναμένεται είναι ότι:

«Η άνωση που δέχεται ένα σώμα μέσα σε υγρό είναι ίση με το βάρος του υγρού που εκτοπίζει.»

Η διατύπωση αυτή αποτελεί την πειραματική επιβεβαίωση της Αρχής του Αρχιμήδη.

Εκπαιδευτική Αξία της Δραστηριότητας

Η δραστηριότητα δεν περιορίζεται στην εκτέλεση ενός εικονικού πειράματος αλλά ενσωματώνει βασικές αρχές διερευνητικής μάθησης.

Ο μαθητής:

- αλληλεπιδρά ενεργά με το περιβάλλον,
- πραγματοποιεί μετρήσεις,
- συλλέγει δεδομένα,
- επεξεργάζεται πληροφορίες,
- διατυπώνει συμπεράσματα.

Με τον τρόπο αυτό αναπτύσσονται τόσο οι γνώσεις φυσικής όσο και δεξιότητες επιστημονικής σκέψης, παρατήρησης και ανάλυσης δεδομένων.

8.2 Δημιουργία Use Case Models για την περιγραφή του Πειράματος της Άνωσης στο Εικονικό Εργαστήριο Klett / Publior Virtual Lab

Για το συγκεκριμένο πλαίσιο **Klett Φυσική Β' Γυμνασίου / Publior Virtual Lab**, τα use cases μπορούν να οργανωθούν γύρω από τη χρήση του εικονικού εργαστηρίου, τη λήψη μετρήσεων και την επιβεβαίωση της Αρχής του Αρχιμήδη. Η Klett αναφέρει ότι το εικονικό εργαστήριο Publior Virtual Lab χρησιμοποιείται μαζί με το διδακτικό πακέτο της Φυσικής Β' Γυμνασίου, ενώ το Book-App περιλαμβάνει τετράδιο Physics Lab για καταγραφή μετρήσεων και συμπερασμάτων.

Πίνακας 8-1 Actors για το πείραμα της Άνωσης

Actor	Ρόλος
Μαθητής	Εκτελεί το πείραμα και καταγράφει μετρήσεις
Εκπαιδευτικός	Καθοδηγεί και ελέγχει τα συμπεράσματα
Klett Book-App	Παρέχει πρόσβαση στο μάθημα και στο Physics Lab
Publior Virtual Lab	Εκτελεί την εικονική προσομοίωση
Εικονικά όργανα μέτρησης	Δυναμόμετρο, δοχείο νερού, ογκομετρικός κύλινδρος

Πίνακας 8-2 Use Cases για το πείραμα της Άνωσης

Use Case	Σύντομη περιγραφή
Σύνδεση στο Klett Book-App	Ο μαθητής εισέρχεται στην εφαρμογή με τα στοιχεία του
Επιλογή ενότητας 4.5 Άνωση – Αρχή του Αρχιμήδη	Ο μαθητής ανοίγει τη σχετική ενότητα Φυσικής Β' Γυμνασίου
Άνοιγμα Publior Virtual Lab	Μετάβαση στο εικονικό εργαστήριο πειραμάτων
Επιλογή πειράματος Άνωσης	Επιλογή του πειράματος για την Αρχή του Αρχιμήδη
Παρατήρηση εικονικών οργάνων	Αναγνώριση δυναμόμετρου, σώματος, δοχείου με νερό και ογκομετρικού κυλίνδρου
Μέτρηση βάρους σώματος στον αέρα	Καταγραφή ένδειξης δυναμόμετρου πριν τη βύθιση
Βύθιση σώματος στο νερό	Το σώμα βυθίζεται στο υγρό χωρίς να ακουμπά στον πυθμένα

Μέτρηση φαινόμενου βάρους	Καταγραφή νέας μειωμένης ένδειξης δυναμόμετρου
Μέτρηση εκτοπισμένου νερού	Καταγραφή όγκου νερού που εκτοπίζεται
Υπολογισμός άνωσης	Υπολογισμός από τη διαφορά βάρους στον αέρα και φαινόμενου βάρους
Υπολογισμός βάρους εκτοπισμένου νερού	Υπολογισμός από τη μάζα/όγκο του εκτοπισμένου νερού
Σύγκριση αποτελεσμάτων	Σύγκριση άνωσης και βάρους εκτοπισμένου νερού
Διατύπωση συμπεράσματος	Επιβεβαίωση της Αρχής του Αρχιμήδη
Καταγραφή στο Physics Lab	Καταχώριση μετρήσεων και συμπερασμάτων
Επανάληψη πειράματος	Επανεκτέλεση αν υπάρχει λάθος ή απόκλιση
Έλεγχος από εκπαιδευτικό	Ο εκπαιδευτικός ελέγχει τις μετρήσεις και το συμπέρασμα

- **Βασικές σχέσεις include**

Υπολογισμός άνωσης

<<include>>

Μέτρηση βάρους σώματος στον αέρα

Υπολογισμός άνωσης

<<include>>

Μέτρηση φαινόμενου βάρους

Υπολογισμός βάρους εκτοπισμένου νερού

<<include>>

Μέτρηση εκτοπισμένου νερού

Σύγκριση αποτελεσμάτων

<<include>>

Υπολογισμός άνωσης

Σύγκριση αποτελεσμάτων

<<include>>

Υπολογισμός βάρους εκτοπισμένου νερού

- **Βασικές σχέσεις extend**

Επανάληψη πειράματος

<<extend>>

Μέτρηση βάρους σώματος στον αέρα

Επανάληψη πειράματος

<<extend>>

Μέτρηση φαινόμενου βάρους

Επανάληψη πειράματος

<<extend>>

Μέτρηση εκτοπισμένου νερού

Έλεγχος από εκπαιδευτικό

<<extend>>

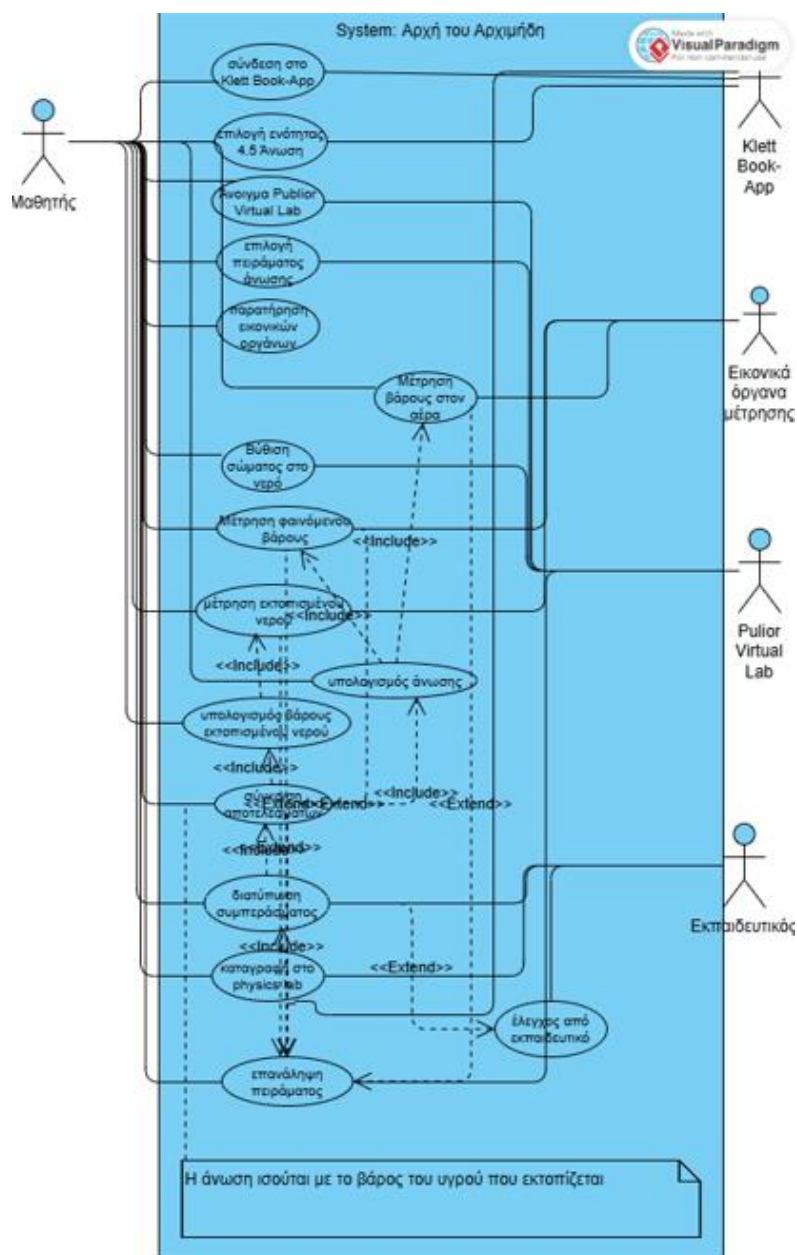
Διατύπωση συμπεράσματος

Δημιουργία του Use Case Diagram

Το τελικό Use Case Diagram αναπαριστά:

- τους actors,
- τις λειτουργίες του συστήματος,
- τις σχέσεις αλληλεπίδρασης.

Η δομή του διαγράμματος ακολουθεί τις προδιαγραφές UML 2.5.1 του Object Management Group (OMG, 2023), υλοποιήθηκε με το λογισμικό visual – paradigm και παρουσιάζεται στην εικόνα 8-2.



Εικ. 8-2. Δημιουργία Use Case Models για την περιγραφή του Πειράματος της Άνωσης στο Εικονικό Εργαστήριο Klett / Publior Virtual Lab

Σημασιολογική Ανάλυση του Use Case Model για τη Μελέτη Περίπτωσης του Πειράματος της Άνωσης

Η μοντελοποίηση απαιτήσεων μέσω Use Case Diagrams αποτελεί μία από τις πλέον διαδεδομένες τεχνικές της Unified Modeling Language (UML), καθώς επιτρέπει την αναπαράσταση των λειτουργιών ενός συστήματος από την οπτική γωνία των χρηστών του (Object Management Group [OMG], 2023). Τα Use Case Models χρησιμοποιούνται για την περιγραφή των υπηρεσιών που παρέχει ένα σύστημα, των εμπλεκόμενων οντοτήτων και των αλληλεπιδράσεων που αναπτύσσονται μεταξύ αυτών.

Στην παρούσα μελέτη περίπτωσης το Use Case Model σχεδιάστηκε με σκοπό την αποτύπωση της αλληλεπίδρασης του μαθητή με το ψηφιακό περιβάλλον Klett Book-App και το Publior Virtual Lab κατά την εκτέλεση του πειράματος της Άνωσης. Η σημασιολογική ανάλυση του μοντέλου αποσκοπεί στην αξιολόγηση της ικανότητάς του να αναπαριστά με πληρότητα τις λειτουργικές απαιτήσεις, τους εμπλεκόμενους ρόλους και τις διαδικασίες που χαρακτηρίζουν το συγκεκριμένο εκπαιδευτικό σενάριο.

Θεωρητικό Υπόβαθρο της Σημασιολογικής Ανάλυσης Use Case Models

Η σημασιολογική ανάλυση εξετάζει κατά πόσο τα στοιχεία ενός μοντέλου αποτυπώνουν με ακρίβεια το νόημα και τη λειτουργία του πραγματικού συστήματος που αναπαριστούν (Wand & Weber, 2002). Στα Use Case Models η σημασιολογική ποιότητα σχετίζεται κυρίως με:

- την ορθή αναγνώριση των actors,
- την πληρότητα των use cases,
- την ορθότητα των σχέσεων μεταξύ των στοιχείων,
- την κάλυψη των λειτουργικών απαιτήσεων,
- την αναπαράσταση εναλλακτικών και εξαιρεσιακών σεναρίων.

Σύμφωνα με τους Dennis, Wixom και Tegarden (2020), ένα Use Case Diagram θεωρείται σημασιολογικά επαρκές όταν αποτυπώνει με σαφήνεια τις λειτουργίες που προσφέρονται στους χρήστες και τις σχέσεις που αναπτύσσονται μεταξύ αυτών των λειτουργιών.

Σημασιολογική Ανάλυση των Actors

Ο Actor «Μαθητής»

Ο μαθητής αποτελεί τον κύριο actor του συστήματος και είναι ο βασικός αποδέκτης των λειτουργιών που παρέχονται από το εικονικό εργαστήριο.

Σημασιολογικά, ο ρόλος του μαθητή περιλαμβάνει:

- την πρόσβαση στο εκπαιδευτικό περιβάλλον,
- την επιλογή του πειράματος,
- τη λήψη μετρήσεων,
- την επεξεργασία δεδομένων,
- τη διατύπωση συμπερασμάτων.

Η αναπαράσταση αυτή συνάδει με τις αρχές της διερευνητικής και εποικοδομητικής μάθησης, σύμφωνα με τις οποίες ο μαθητής συμμετέχει ενεργά στην οικοδόμηση της γνώσης μέσω πειραματισμού και ανακάλυψης (Hmelo-Silver, 2019).

Ο Actor «Εκπαιδευτικός»

Ο εκπαιδευτικός λειτουργεί ως δευτερεύων actor, ο οποίος υποστηρίζει και επιβλέπει τη μαθησιακή διαδικασία.

Οι λειτουργίες που συνδέονται με τον εκπαιδευτικό αφορούν:

- την παροχή καθοδήγησης,
- τον έλεγχο των αποτελεσμάτων,
- την επιβεβαίωση των συμπερασμάτων.

Η παρουσία του actor στο μοντέλο είναι σημασιολογικά απαραίτητη, καθώς αντανακλά τον πραγματικό ρόλο του εκπαιδευτικού στο σχολικό εργαστήριο.

Ο Actor «Publior Virtual Lab»

Το Publior Virtual Lab αποτελεί τον μηχανισμό εκτέλεσης της προσομοίωσης.

Ο actor αυτός παρέχει:

- τα εικονικά όργανα,

- τις προσομοιώσεις φυσικών φαινομένων,
- τις εργαστηριακές μετρήσεις.

Η παρουσία του στο μοντέλο εξασφαλίζει ότι η πειραματική διαδικασία αναπαρίσταται ως αλληλεπίδραση μεταξύ χρήστη και λογισμικού.

Σημασιολογική Ανάλυση των Use Cases

Use Cases Πρόσβασης στο Σύστημα

Τα use cases:

- Σύνδεση στο Klett Book-App,
- Επιλογή ενότητας Άνωση,
- Άνοιγμα Publior Virtual Lab,

αντιστοιχούν στη διαδικασία πρόσβασης του μαθητή στο εκπαιδευτικό περιβάλλον. Η σημασιολογική τους λειτουργία είναι ιδιαίτερα σημαντική, καθώς περιγράφουν τα βήματα που προηγούνται της πειραματικής δραστηριότητας.

Use Cases Πειραματικής Εκτέλεσης

Η δεύτερη κατηγορία περιλαμβάνει:

- Παρατήρηση εικονικών οργάνων,
- Μέτρηση βάρους στον αέρα,
- Βύθιση σώματος στο νερό,
- Μέτρηση φαινόμενου βάρους,
- Μέτρηση εκτοπισμένου νερού.

Τα use cases αυτά αντιπροσωπεύουν τα βασικά βήματα του εργαστηριακού πρωτοκόλλου.

Η αντιστοίχισή τους με τις πραγματικές ενέργειες του μαθητή συμβάλλει στη σημασιολογική εγκυρότητα του μοντέλου.

Use Cases Υπολογισμών

Τα use cases:

- Υπολογισμός άνωσης,
- Υπολογισμός βάρους εκτοπισμένου νερού,

μετατρέπουν τις πειραματικές μετρήσεις σε φυσικά μεγέθη.

Η συμπερίληψή τους είναι αναγκαία, καθώς χωρίς αυτά δεν θα ήταν δυνατή η επιβεβαίωση της Αρχής του Αρχιμήδη.

Use Cases Επαλήθευσης

Η τελική κατηγορία περιλαμβάνει:

- Σύγκριση αποτελεσμάτων,
- Διατύπωση συμπεράσματος,
- Καταγραφή στο Physics Lab.

Τα συγκεκριμένα use cases συνδέονται άμεσα με την οικοδόμηση της επιστημονικής γνώσης και την αξιολόγηση της ορθότητας των μετρήσεων.

Σημασιολογική Ανάλυση των Σχέσεων Include

Οι σχέσεις $\diamond$ χρησιμοποιούνται για την αναπαράσταση λειτουργιών που αποτελούν υποχρεωτικό τμήμα άλλων λειτουργιών.

Στο μοντέλο παρατηρούνται οι εξής χαρακτηριστικές περιπτώσεις:

- Ο Υπολογισμός Άνωσης περιλαμβάνει τη Μέτρηση Βάρους στον Αέρα.
- Ο Υπολογισμός Άνωσης περιλαμβάνει τη Μέτρηση Φαινόμενου Βάρους.
- Ο Υπολογισμός Βάρους Εκτοπισμένου Νερού περιλαμβάνει τη Μέτρηση Εκτοπισμένου Νερού.

Η χρήση των σχέσεων include θεωρείται σημασιολογικά ορθή, καθώς οι υπολογισμοί δεν μπορούν να πραγματοποιηθούν χωρίς τις αντίστοιχες μετρήσεις. Επιπλέον, η σχέση include αποτυπώνει τη λογική εξάρτηση μεταξύ των λειτουργιών, βελτιώνοντας τη σαφήνεια του μοντέλου (Satzinger, Jackson & Burd, 2021).

Σημασιολογική Ανάλυση των Σχέσεων Extend

Οι σχέσεις $\diamond$ χρησιμοποιούνται για την αναπαράσταση προαιρετικών ή εναλλακτικών λειτουργιών.

Στο μοντέλο εμφανίζονται κυρίως στις περιπτώσεις:

- Επανάληψη Πειράματος,
- Έλεγχος από Εκπαιδευτικό.

Η ύπαρξη των σχέσεων αυτών επιτρέπει την αναπαράσταση καταστάσεων που δεν εκτελούνται πάντα αλλά ενεργοποιούνται μόνο υπό συγκεκριμένες συνθήκες. Η δυνατότητα αυτή ενισχύει σημαντικά τη σημασιολογική πληρότητα του μοντέλου, καθώς επιτρέπει την περιγραφή πραγματικών σεναρίων χρήσης και όχι μόνο της ιδανικής εκτέλεσης.

Σύνδεση του Use Case Model με τους Μαθησιακούς Στόχους

Η σημασιολογική ανάλυση δείχνει ότι τα use cases δεν περιγράφουν μόνο τεχνικές λειτουργίες αλλά συνδέονται άμεσα με μαθησιακούς στόχους.

Πίνακας 8-3 Σύνδεση του Use Case Model με τους Μαθησιακούς Στόχους

Use Case	Μαθησιακός Στόχος
Παρατήρηση εικονικών οργάνων	Εξοικείωση με εργαστηριακά όργανα
Μέτρηση βάρους στον αέρα	Κατανόηση της έννοιας του βάρους
Βύθιση σώματος	Παρατήρηση φυσικού φαινομένου
Υπολογισμός άνωσης	Εφαρμογή φυσικών νόμων
Σύγκριση αποτελεσμάτων	Ανάλυση δεδομένων
Διατύπωση συμπεράσματος	Ανάπτυξη επιστημονικής σκέψης

Η σύνδεση αυτή αποδεικνύει ότι το μοντέλο δεν περιγράφει απλώς μία ακολουθία ενεργειών, αλλά αποτυπώνει μία ολοκληρωμένη μαθησιακή διαδικασία.

Αξιολόγηση της Σημασιολογικής Πληρότητας

Η αξιολόγηση του μοντέλου δείχνει ότι καλύπτονται όλες οι βασικές λειτουργικές απαιτήσεις της μελέτης περίπτωσης.

Ειδικότερα:

- αναγνωρίζονται όλοι οι βασικοί actors,
- αναπαρίστανται όλες οι κύριες δραστηριότητες του πειράματος,
- αποτυπώνονται οι σχέσεις εξάρτησης μεταξύ των λειτουργιών,

- περιλαμβάνονται εναλλακτικά σενάρια εκτέλεσης,
- υποστηρίζεται η μαθησιακή και πειραματική διάσταση του σεναρίου.

Κατά συνέπεια, το Use Case Model μπορεί να χαρακτηριστεί ως σημασιολογικά πλήρες για το συγκεκριμένο πεδίο εφαρμογής.

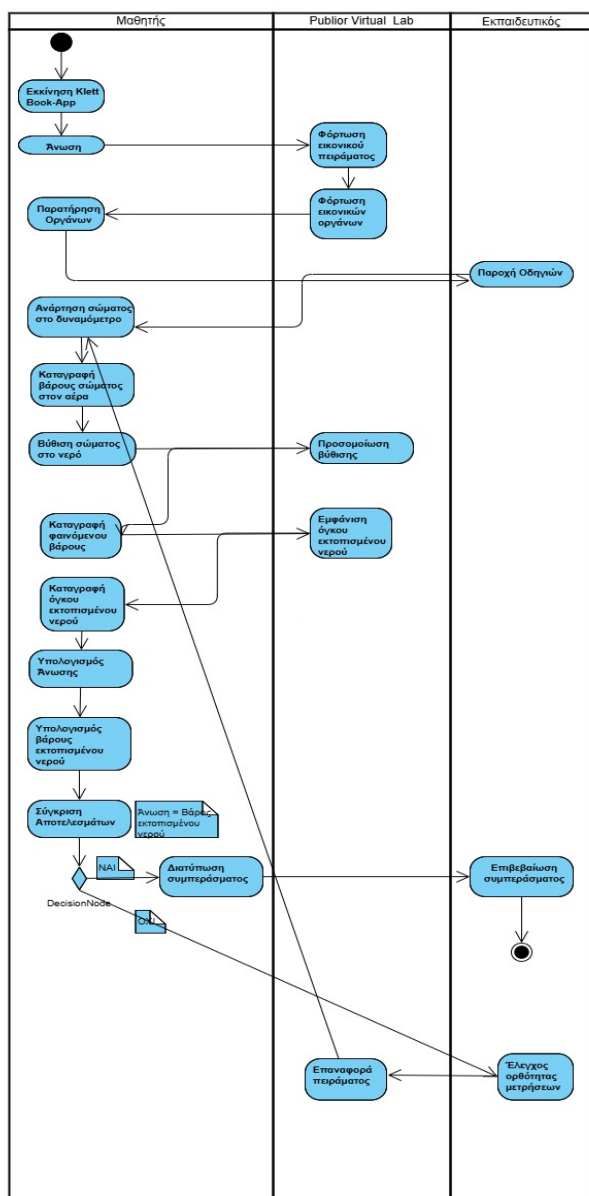
Συμπεράσματα

Η σημασιολογική ανάλυση του Use Case Model κατέδειξε ότι το μοντέλο αναπαριστά με πληρότητα και συνέπεια τη διαδικασία εκτέλεσης του πειράματος της Άνωσης στο περιβάλλον Klett Book-App και Publior Virtual Lab. Οι actors ανταποκρίνονται στους πραγματικούς συμμετέχοντες της εκπαιδευτικής διαδικασίας, τα use cases καλύπτουν το σύνολο των λειτουργικών απαιτήσεων και οι σχέσεις include και extend αποτυπώνουν με ακρίβεια τις λογικές εξαρτήσεις και τις εναλλακτικές ροές.

Συνεπώς, το μοντέλο μπορεί να θεωρηθεί κατάλληλο για την ανάλυση και τεκμηρίωση της συγκεκριμένης μελέτης περίπτωσης, παρέχοντας μια ολοκληρωμένη αναπαράσταση των αλληλεπιδράσεων μεταξύ χρηστών και συστήματος.

8.3 Δημιουργία Activity Diagrams με σημασιολογική τεκμηρίωση

Η ανάπτυξη του Activity Diagram πραγματοποιήθηκε ακολουθώντας τις αρχές της UML και τη λογική ροή των λειτουργιών του συστήματος. Το διάγραμμα σχεδιάστηκε μέσω του λογισμικού Visual-Paradigm και παρουσιάζεται στην εικόνα 8-3.



Εκ. 8-3 Δημιουργία Activity Diagrams

Όπως φαίνεται στην παραπάνω εικόνα, δημιουργήθηκαν 3 swimlanes τα οποία αντιστοιχούν :

- Στο μαθητή
 - Στο εικονικό εργαστήριο
 - Στον εκπαιδευτικό
-
- **Activities στο swimlane «Μαθητής»**

Εκκίνηση εικονικού εργαστηρίου
Επιλογή πειράματος άνωσης
Παρατήρηση διαθέσιμων οργάνων
Ανάρτηση σώματος στο δυναμόμετρο
Καταγραφή βάρους σώματος στον αέρα
Βύθιση σώματος στο νερό
Καταγραφή φαινόμενου βάρους στο νερό
Καταγραφή όγκου εκτοπισμένου νερού
Υπολογισμός άνωσης
Υπολογισμός βάρους εκτοπισμένου νερού
Σύγκριση αποτελεσμάτων
Διατύπωση συμπεράσματος

- **Activities στο swimlane «Εικονικό Εργαστήριο»**

Φόρτωση πειράματος
Εμφάνιση οργάνων και υλικών
Εμφάνιση ένδειξης δυναμόμετρου στον αέρα
Προσομοίωση βύθισης σώματος
Εμφάνιση ένδειξης δυναμόμετρου στο νερό
Προσομοίωση εκτόπισης νερού
Εμφάνιση όγκου εκτοπισμένου νερού
Επαναφορά / επανάληψη πειράματος

- **Activities στο swimlane «Εκπαιδευτικός»**

Παροχή οδηγιών
Έλεγχος ορθότητας μετρήσεων
Επιβεβαίωση συμπεράσματος

- **Βασικές συνδέσεις Control Flow**

Initial Node

→ Εκκίνηση εικονικού εργαστηρίου

Εκκίνηση εικονικού εργαστηρίου

→ Φόρτωση πειράματος

Φόρτωση πειράματος

→ Επιλογή πειράματος άνωσης

Επιλογή πειράματος άνωσης

→ Εμφάνιση οργάνων και υλικών

Εμφάνιση οργάνων και υλικών

→ Παρατήρηση διαθέσιμων οργάνων

Παρατήρηση διαθέσιμων οργάνων

→ Παροχή οδηγιών

Παροχή οδηγιών

→ Ανάρτηση σώματος στο δυναμόμετρο

- **Συνδέσεις μέτρησης βάρους στον αέρα**

Ανάρτηση σώματος στο δυναμόμετρο

→ Εμφάνιση ένδειξης δυναμόμετρου στον αέρα

Εμφάνιση ένδειξης δυναμόμετρου στον αέρα

→ Καταγραφή βάρους σώματος στον αέρα

- **Συνδέσεις μέτρησης μέσα στο νερό**

Καταγραφή βάρους σώματος στον αέρα

→ Βύθιση σώματος στο νερό

Βύθιση σώματος στο νερό

→ Προσομοίωση βύθισης σώματος

Προσομοίωση βύθισης σώματος

→ Εμφάνιση ένδειξης δυναμόμετρου στο νερό

Εμφάνιση ένδειξης δυναμόμετρου στο νερό

→ Καταγραφή φαινομένου βάρους στο νερό

- **Συνδέσεις εκτοπισμένου νερού**

Καταγραφή φαινομένου βάρους στο νερό

→ Προσομοίωση εκτόπισης νερού

Προσομοίωση εκτόπισης νερού

→ Εμφάνιση όγκου εκτοπισμένου νερού

Εμφάνιση όγκου εκτοπισμένου νερού
→ Καταγραφή όγκου εκτοπισμένου νερού

- **Συνδέσεις υπολογισμών**

Καταγραφή όγκου εκτοπισμένου νερού
→ Υπολογισμός άνωσης

Υπολογισμός άνωσης
→ Υπολογισμός βάρους εκτοπισμένου νερού

Υπολογισμός βάρους εκτοπισμένου νερού
→ Σύγκριση αποτελεσμάτων

- **Προσθήκη Decision Node**

Σύγκριση αποτελεσμάτων
→ Άνωση $\approx$ Βάρος εκτοπισμένου νερού;

- **Ροή «Ναι»**

Διατύπωση συμπεράσματος
→ Επιβεβαίωση συμπεράσματος

Επιβεβαίωση συμπεράσματος
→ Activity Final Node

- **Ροή «Όχι»**

Έλεγχος ορθότητας μετρήσεων
→ Επαναφορά / επανάληψη πειράματος

Επαναφορά / επανάληψη πειράματος
→ Ανάρτηση σώματος στο δυναμόμετρο

(Το πείραμα επαναλαμβάνεται αν υπάρχει λάθος).

- **Προσθήκη Final Node**

Activity Final Node

- **Προτεινόμενη συνολική ροή**

Start

→ Εκκίνηση εργαστηρίου
→ Φόρτωση πειράματος
→ Επιλογή πειράματος
→ Εμφάνιση οργάνων
→ Παρατήρηση οργάνων

- Παροχή οδηγιών
- Μέτρηση βάρους στον αέρα
- Μέτρηση φαινομένου βάρους στο νερό
- Μέτρηση εκτοπισμένου νερού
- Υπολογισμοί
- Σύγκριση
- Απόφαση
- Συμπέρασμα ή Επανάληψη
- Stop

- **Προσθήκη σημείωσης για σημασιολογική τεκμηρίωση**

Προσθέτουμε την παρακάτω σημείωση κοντά στο Decision Note:

“Η Αρχή του Αρχιμήδη επιβεβαιώνεται όταν η υπολογισμένη άνωση είναι ίση ή περίπου ίση με το βάρος του εκτοπισμένου νερού”.

Σημασιολογική Τεκμηρίωση του Activity Diagram για το Πείραμα της Αρχής του Αρχιμήδη στο Klett Book-App

Η ανάπτυξη Activity Diagrams αποτελεί μία από τις σημαντικότερες τεχνικές συμπεριφορικής μοντελοποίησης στην Unified Modeling Language (UML), καθώς επιτρέπει την αναπαράσταση της αλληλουχίας ενεργειών, των ροών ελέγχου και των αλληλεπιδράσεων μεταξύ των εμπλεκόμενων οντοτήτων ενός συστήματος (Object Management Group [OMG], 2023). Στο πλαίσιο της παρούσας μελέτης περίπτωσης, το Activity Diagram χρησιμοποιήθηκε για τη μοντελοποίηση της διαδικασίας εκτέλεσης του πειράματος της Άνωσης στο ψηφιακό εκπαιδευτικό περιβάλλον Klett Book-App και στο εικονικό εργαστήριο Publior Virtual Lab.

Η σημασιολογική τεκμηρίωση του διαγράμματος αποσκοπεί στην ερμηνεία της λειτουργίας κάθε δραστηριότητας και στη σύνδεσή της με τους εκπαιδευτικούς και επιστημονικούς στόχους του πειράματος. Με τον τρόπο αυτό διασφαλίζεται ότι το μοντέλο δεν αποτελεί απλώς μία γραφική αναπαράσταση διαδικασιών, αλλά ένα ολοκληρωμένο μέσο περιγραφής της μαθησιακής και πειραματικής διαδικασίας.

Γενική Περιγραφή του Activity Diagram

Το Activity Diagram αναπαριστά το σύνολο των ενεργειών που εκτελούνται από τη στιγμή που ο μαθητής εισέρχεται στο ψηφιακό περιβάλλον έως τη διατύπωση του τελικού συμπεράσματος για την Αρχή του Αρχιμήδη.

Η ροή του διαγράμματος οργανώνεται σε τρεις διακριτές περιοχές ευθύνης (Swimlanes):

- Μαθητής,
- Publior Virtual Lab,
- Εκπαιδευτικός.

Η χρήση Swimlanes επιτρέπει τη σαφή κατανομή των δραστηριοτήτων μεταξύ των εμπλεκόμενων οντοτήτων και βελτιώνει σημαντικά τη σημασιολογική σαφήνεια του μοντέλου (Dennis, Wixom & Tegarden, 2020).

Το διάγραμμα ξεκινά με την είσοδο του μαθητή στο περιβάλλον Klett και ολοκληρώνεται με την επιβεβαίωση της Αρχής του Αρχιμήδη ή, εναλλακτικά, με την επανάληψη της διαδικασίας σε περίπτωση αποκλίσεων στις μετρήσεις.

Σημασιολογική Τεκμηρίωση των Δραστηριοτήτων του Μαθητή

3.1 Εκκίνηση Klett Book-App

Η δραστηριότητα αυτή σηματοδοτεί την έναρξη της αλληλεπίδρασης του μαθητή με το ψηφιακό μαθησιακό περιβάλλον. Αποτελεί προϋπόθεση για την πρόσβαση στο εκπαιδευτικό περιεχόμενο και στο εικονικό εργαστήριο.

Σημασιολογικά, η δραστηριότητα αντιπροσωπεύει τη μετάβαση από το εξωτερικό περιβάλλον χρήσης στο οργανωμένο μαθησιακό περιβάλλον της εφαρμογής.

Επιλογή Ενότητας Άνωση

Η συγκεκριμένη δραστηριότητα εκφράζει την επιλογή του γνωστικού αντικειμένου που θα μελετηθεί. Ο μαθητής μεταβαίνει στη θεματική ενότητα που αφορά την Άνωση και την Αρχή του Αρχιμήδη, ενεργοποιώντας το αντίστοιχο εκπαιδευτικό σενάριο.

Η ενέργεια αυτή συνδέεται με την έννοια της πλοήγησης σε ψηφιακά μαθησιακά περιβάλλοντα και αποτελεί τυπικό στοιχείο των εκπαιδευτικών συστημάτων ηλεκτρονικής μάθησης (Herber & Wheaton, 2024).

Παρατήρηση Οργάνων

Η δραστηριότητα αυτή επιτρέπει στον μαθητή να εξοικειωθεί με τα εικονικά όργανα του πειράματος. Η σημασιολογική της αξία είναι ιδιαίτερα σημαντική, καθώς λειτουργεί ως φάση προετοιμασίας πριν από την πραγματοποίηση των μετρήσεων. Στην εκπαιδευτική έρευνα έχει διαπιστωθεί ότι η εξοικείωση με τα όργανα προηγείται της πειραματικής διαδικασίας και συμβάλλει στη μείωση λαθών κατά την εκτέλεση των δραστηριοτήτων (Spector, 2022).

Καταγραφή Βάρους Σώματος στον Αέρα

Η δραστηριότητα αντιστοιχεί στην πρώτη βασική μέτρηση του πειράματος. Ο μαθητής καταγράφει την ένδειξη του δυναμόμετρου όταν το σώμα βρίσκεται στον αέρα. Η τιμή αυτή αποτελεί τη βάση για τον μεταγενέστερο υπολογισμό της άνωσης και επομένως η σημασιολογική της σημασία είναι καθοριστική για την εξέλιξη της πειραματικής διαδικασίας.

Βύθιση Σώματος στο Νερό

Η βύθιση του σώματος αντιπροσωπεύει το σημείο μετάβασης από την κατάσταση ισορροπίας στον αέρα στη φυσική αλληλεπίδραση σώματος και υγρού. Κατά τη δραστηριότητα αυτή ενεργοποιείται το φυσικό φαινόμενο της άνωσης, το οποίο αποτελεί το αντικείμενο μελέτης του πειράματος.

Καταγραφή Φαινόμενου Βάρους

Η δραστηριότητα αυτή αποτυπώνει τη μέτρηση του φαινόμενου βάρους του σώματος μέσα στο νερό. Η ένδειξη του δυναμόμετρου είναι μικρότερη από την αρχική λόγω της δύναμης της άνωσης. Σημασιολογικά, η δραστηριότητα αποτελεί τη δεύτερη κρίσιμη μέτρηση του πειράματος και συνδέεται άμεσα με τον υπολογισμό της άνωσης.

Καταγραφή Όγκου Εκτοπισμένου Νερού

Η μέτρηση του εκτοπισμένου νερού αποτελεί την πειραματική σύνδεση μεταξύ της βύθισης του σώματος και της Αρχής του Αρχιμήδη. Η δραστηριότητα αυτή επιτρέπει τον υπολογισμό του βάρους του εκτοπισμένου νερού και συνεπώς τη σύγκριση με την υπολογισμένη άνωση.

Υπολογισμός Άνωσης

Η άνωση υπολογίζεται από την παρακάτω σχέση:

Άνωση = Βάρος στον αέρα – Φαινόμενο βάρος

Η δραστηριότητα μετατρέπει τις πειραματικές μετρήσεις σε επιστημονικό μέγεθος και αποτελεί τη γέφυρα μεταξύ παρατήρησης και επιστημονικής ερμηνείας.

Υπολογισμός Βάρους Εκτοπισμένου Νερού

Η δραστηριότητα αυτή επιτρέπει τη μετατροπή της μέτρησης του όγκου σε φυσικό μέγεθος βάρους. Σημασιολογικά, πρόκειται για την εφαρμογή της θεωρητικής γνώσης στη διαδικασία επεξεργασίας των δεδομένων.

Σύγκριση Αποτελεσμάτων

Η δραστηριότητα της σύγκρισης αποτελεί το κεντρικό σημείο του πειράματος. Εδώ αξιολογείται κατά πόσο η υπολογισμένη άνωση είναι ίση με το βάρος του εκτοπισμένου νερού. Η δραστηριότητα αυτή αντιπροσωπεύει το στάδιο επαλήθευσης μιας επιστημονικής υπόθεσης, διαδικασία που θεωρείται θεμελιώδης στη διερευνητική μάθηση (Bransford, Brown & Cocking, 2020).

Διατύπωση Συμπεράσματος

Η τελική δραστηριότητα του μαθητή αφορά τη διατύπωση του επιστημονικού συμπεράσματος. Ο μαθητής καταλήγει στο συμπέρασμα ότι: «*Η άνωση που δέχεται ένα σώμα μέσα σε υγρό είναι ίση με το βάρος του υγρού που εκτοπίζει*». Η δραστηριότητα αυτή ολοκληρώνει τον κύκλο της επιστημονικής διερεύνησης.

Σημασιολογική Τεκμηρίωση των Δραστηριοτήτων του Publior Virtual Lab

Οι δραστηριότητες του εικονικού εργαστηρίου περιλαμβάνουν:

- τη φόρτωση του πειράματος,
- την εμφάνιση των οργάνων,
- την προσομοίωση της βύθισης,
- την εμφάνιση των μετρήσεων,
- την επαναφορά του πειράματος.

Οι δραστηριότητες αυτές λειτουργούν υποστηρικτικά προς τον μαθητή και παρέχουν το απαραίτητο περιβάλλον αλληλεπίδρασης. Σημασιολογικά αντιπροσωπεύουν τον ρόλο του λογισμικού ως μεσολαβητή μεταξύ του χρήστη και του υπό μελέτη φυσικού φαινομένου.

Σημασιολογική Τεκμηρίωση των Δραστηριοτήτων του Εκπαιδευτικού

Ο εκπαιδευτικός συμμετέχει σε δύο κρίσιμα σημεία του διαγράμματος:

- παροχή οδηγιών,
- επιβεβαίωση συμπεράσματος.

Οι δραστηριότητες αυτές αναδεικνύουν τον καθοδηγητικό και υποστηρικτικό ρόλο του εκπαιδευτικού στη μαθησιακή διαδικασία. Η παρουσία του εκπαιδευτικού δεν υποκαθιστά την αυτενέργεια του μαθητή αλλά λειτουργεί ως μηχανισμός ανατροφοδότησης και παιδαγωγικής υποστήριξης (Laurillard, 2023).

Σημασιολογία του Κόμβου Απόφασης

Ο κόμβος απόφασης με το ερώτημα: «*Η άνωση είναι ίση με το βάρος του εκτοπισμένου νερού;*» αποτελεί το σημαντικότερο σημείο ελέγχου του Activity Diagram.

Η ύπαρξή του επιτρέπει τη δημιουργία δύο εναλλακτικών ροών:

1. Επιβεβαίωση της θεωρίας.
2. Επανεκτέλεση του πειράματος.

Η χρήση Decision Nodes για την αναπαράσταση εναλλακτικών σεναρίων θεωρείται βασική πρακτική στην UML και συμβάλλει ουσιαστικά στη σημασιολογική πληρότητα των Activity Diagrams (OMG, 2023).

Συμπεράσματα

Η σημασιολογική ανάλυση του Activity Diagram έδειξε ότι το μοντέλο αποτυπώνει με πληρότητα τόσο τη μαθησιακή διαδικασία όσο και τη φυσική πειραματική διαδικασία που εκτελείται στο περιβάλλον Klett Book-App και Publior Virtual Lab.

Η χρήση Swimlanes επιτρέπει τη σαφή διάκριση των ευθυνών μεταξύ μαθητή, εικονικού εργαστηρίου και εκπαιδευτικού, ενώ η ύπαρξη Decision Node υποστηρίζει τη μοντελοποίηση της επιστημονικής διαδικασίας ελέγχου και επαλήθευσης. Συνολικά, το Activity Diagram παρουσιάζει υψηλό βαθμό σημασιολογικής πληρότητας και μπορεί να θεωρηθεί κατάλληλο για την αναπαράσταση του πειράματος της Άνωσης σε ένα σύγχρονο ψηφιακό μαθησιακό περιβάλλον.

8.4 Αξιολόγηση των Μοντέλων ως προς τη Σημασιολογική Πληρότητα

Η αξιολόγηση της σημασιολογικής πληρότητας αποτελεί βασικό κριτήριο ποιότητας κατά τη μοντελοποίηση πληροφοριακών και εκπαιδευτικών συστημάτων, καθώς εξετάζει κατά πόσο τα μοντέλα αποτυπώνουν με ακρίβεια το νόημα, τις λειτουργίες και τη συμπεριφορά του πραγματικού συστήματος που αναπαριστούν (Wand & Weber, 2002). Στην παρούσα μελέτη περίπτωσης η αξιολόγηση πραγματοποιείται τόσο για το Use Case Model όσο και για το Activity Diagram που αναπτύχθηκαν για το πείραμα της Άνωσης στο περιβάλλον Klett Book-App και Publior Virtual Lab.

Αξιολόγηση του Use Case Model

Το Use Case Model παρουσιάζει υψηλό βαθμό σημασιολογικής πληρότητας, καθώς αποτυπώνει με σαφήνεια όλους τους βασικούς συμμετέχοντες της πειραματικής διαδικασίας και το σύνολο των λειτουργικών απαιτήσεων του συστήματος. Οι actors που αναγνωρίστηκαν (Μαθητής, Εκπαιδευτικός, Klett Book-App, Publior Virtual Lab και Εικονικά Όργανα Μέτρησης) ανταποκρίνονται στους πραγματικούς φορείς που εμπλέκονται στην εκτέλεση του πειράματος και καλύπτουν τόσο τη μαθησιακή όσο και την τεχνολογική διάσταση του σεναρίου.

Παράλληλα, τα use cases περιγράφουν με πληρότητα όλα τα στάδια της πειραματικής διαδικασίας, από τη σύνδεση στο περιβάλλον Klett έως τη διατύπωση του τελικού

συμπεράσματος και την καταγραφή των αποτελεσμάτων στο Physics Lab. Η ύπαρξη σχέσεων <<include>> επιτρέπει την αναπαράσταση των υποχρεωτικών εξαρτήσεων μεταξύ μετρήσεων και υπολογισμών, ενώ οι σχέσεις <<extend>> υποστηρίζουν τη μοντελοποίηση εναλλακτικών καταστάσεων, όπως η επανάληψη του πειράματος και ο έλεγχος από τον εκπαιδευτικό (OMG, 2023).

Ωστόσο, παρά την υψηλή λειτουργική του πληρότητα, το Use Case Model δεν αποτυπώνει τη χρονική αλληλουχία των ενεργειών ούτε τις εσωτερικές καταστάσεις του συστήματος. Επιπλέον, δεν περιγράφει τη φυσική συμπεριφορά του φαινομένου της άνωσης ούτε τους μαθηματικούς υπολογισμούς που πραγματοποιούνται κατά την επεξεργασία των δεδομένων. Οι περιορισμοί αυτοί είναι σύμφωνοι με τη φύση των Use Case Diagrams, τα οποία εστιάζουν κυρίως στις λειτουργικές απαιτήσεις και όχι στη συμπεριφορά του συστήματος (Dennis, Wixom & Tegarden, 2020).

Αξιολόγηση του Activity Diagram

Το Activity Diagram παρουσιάζει ακόμη μεγαλύτερο βαθμό σημασιολογικής πληρότητας, καθώς αποτυπώνει αναλυτικά τη ροή εκτέλεσης του πειράματος και τις αλληλεπιδράσεις μεταξύ των εμπλεκόμενων οντοτήτων. Μέσω των swimlanes επιτυγχάνεται σαφής διαχωρισμός ευθυνών μεταξύ μαθητή, εικονικού εργαστηρίου και εκπαιδευτικού, γεγονός που διευκολύνει την κατανόηση της διαδικασίας και μειώνει τις πιθανότητες εσφαλμένης ερμηνείας του μοντέλου.

Ιδιαίτερα σημαντική είναι η παρουσία του Decision Node που εξετάζει κατά πόσο η υπολογισμένη άνωση είναι ίση με το βάρος του εκτοπισμένου νερού. Ο συγκεκριμένος κόμβος επιτρέπει τη μοντελοποίηση δύο εναλλακτικών ροών: της επιβεβαίωσης της Αρχής του Αρχιμήδη και της επανεκτέλεσης του πειράματος σε περίπτωση απόκλισης των αποτελεσμάτων. Με τον τρόπο αυτό το μοντέλο δεν περιγράφει μόνο την ιδανική εκτέλεση της διαδικασίας, αλλά ενσωματώνει και μηχανισμούς ελέγχου και ανατροφοδότησης, στοιχείο που ενισχύει σημαντικά τη σημασιολογική του ποιότητα (Brambilla, Cabot & Wimmer, 2017). Παρά τα πλεονεκτήματά του, το Activity Diagram εξακολουθεί να αφαιρεί ορισμένες πτυχές της πραγματικής πειραματικής διαδικασίας. Δεν αποτυπώνει με λεπτομέρεια τις φυσικές μεταβλητές του συστήματος, τις αβεβαιότητες των μετρήσεων ή

τις γνωστικές διεργασίες που αναπτύσσει ο μαθητής κατά την ερμηνεία των αποτελεσμάτων. Επιπλέον, η UML δεν παρέχει εγγενείς μηχανισμούς για την αναπαράσταση μαθηματικών περιορισμών ή φυσικών νόμων, γεγονός που περιορίζει τη δυνατότητα πλήρους σημασιολογικής αναπαράστασης επιστημονικών φαινομένων (Douglass, 2016).

Συνολική Αποτίμηση

Η συνδυαστική χρήση Use Case Model και Activity Diagram αποδείχθηκε ιδιαίτερα αποτελεσματική για την αναπαράσταση του πειράματος της Άνωσης. Το Use Case Model αποτυπώνει τις λειτουργικές απαιτήσεις και τους εμπλεκόμενους ρόλους, ενώ το Activity Diagram αναπαριστά τη δυναμική συμπεριφορά και τη ροή εκτέλεσης της πειραματικής διαδικασίας. Η συμπληρωματική χρήση των δύο μοντέλων οδηγεί σε υψηλό επίπεδο σημασιολογικής πληρότητας και παρέχει μια ολοκληρωμένη αναπαράσταση τόσο της εκπαιδευτικής όσο και της τεχνολογικής διάστασης του σεναρίου.

8.5 Συμπεράσματα της Μελέτης Περίπτωσης

Η δεύτερη μελέτη περίπτωσης επικεντρώθηκε στη μοντελοποίηση του πειράματος της Άνωσης στο περιβάλλον Klett Book-App και Publior Virtual Lab, με στόχο την αξιολόγηση της δυνατότητας της UML να αναπαριστά εκπαιδευτικές και πειραματικές διαδικασίες σε εικονικά εργαστήρια. Τα αποτελέσματα της ανάλυσης έδειξαν ότι η UML μπορεί να χρησιμοποιηθεί αποτελεσματικά για τη μοντελοποίηση όχι μόνο πληροφοριακών συστημάτων αλλά και μαθησιακών δραστηριοτήτων που περιλαμβάνουν αλληλεπίδραση χρήστη, επιστημονική διερεύνηση και πειραματική διαδικασία.

Η ανάπτυξη του Use Case Model επέτρεψε την αποτύπωση των λειτουργικών απαιτήσεων του συστήματος και των σχέσεων μεταξύ των εμπλεκόμενων οντοτήτων. Παράλληλα, το Activity Diagram κατέστησε δυνατή την αναπαράσταση της πλήρους ακολουθίας ενεργειών που εκτελεί ο μαθητής, από την είσοδο στο περιβάλλον έως τη διατύπωση του τελικού συμπεράσματος για την Αρχή του Αρχιμήδη. Η χρήση swimlanes και decision nodes συνέβαλε σημαντικά στη σαφήνεια και στην αναγνωσιμότητα του μοντέλου.

Ένα ιδιαίτερα σημαντικό εύρημα της μελέτης είναι ότι η σημασιολογική τεκμηρίωση των μοντέλων διευκολύνει την κατανόηση του πειράματος και καθιστά εμφανείς τις σχέσεις μεταξύ των επιμέρους ενεργειών και των μαθησιακών στόχων. Η διαδικασία αυτή συμβάλλει στην ενίσχυση της ποιότητας των μοντέλων και στη μείωση πιθανών παρερμηνειών κατά την ανάλυση και σχεδίαση εκπαιδευτικών συστημάτων (Sommerville, 2016).

Παράλληλα, η μελέτη ανέδειξε και τους περιορισμούς της UML στην αναπαράσταση φυσικών φαινομένων. Αν και τα μοντέλα αποτυπώνουν αποτελεσματικά τη διαδικασία εκτέλεσης του πειράματος, δεν μπορούν να αναπαραστήσουν πλήρως τις φυσικές μεταβλητές, τις μαθηματικές σχέσεις και τις γνωστικές διεργασίες που εμπλέκονται στη μαθησιακή διαδικασία. Για την κάλυψη τέτοιων αναγκών ενδέχεται να απαιτηθεί η αξιοποίηση περισσότερο εξειδικευμένων προσεγγίσεων, όπως η SysML ή οι τυπικές σημασιολογικές προσεγγίσεις που παρουσιάστηκαν στα προηγούμενα κεφάλαια.

Συνολικά, η μελέτη περίπτωσης επιβεβαιώνει ότι η UML αποτελεί ένα ιδιαίτερα χρήσιμο εργαλείο για τη μοντελοποίηση εικονικών εργαστηρίων και εκπαιδευτικών δραστηριοτήτων. Η συνδυαστική χρήση Use Case Models και Activity Diagrams επέτρεψε την ολοκληρωμένη αναπαράσταση του πειράματος της Άνωσης και ανέδειξε τον καθοριστικό ρόλο της σημασιολογικής πληρότητας στη δημιουργία ποιοτικών μοντέλων λογισμικού και εκπαιδευτικών συστημάτων.

9. Συζήτηση – Αποτελέσματα

9.1 Επισκόπηση ευρημάτων

Η παρούσα εργασία διερεύνησε τον ρόλο της σημασιολογίας στη μοντελοποίηση λογισμικού μέσω της UML, εξετάζοντας τόσο τις θεωρητικές διαστάσεις της σημασιολογικής πληρότητας όσο και την εφαρμογή τους σε πραγματικά εκπαιδευτικά περιβάλλοντα. Τα αποτελέσματα της έρευνας καταδεικνύουν ότι η ποιότητα ενός UML μοντέλου δεν εξαρτάται αποκλειστικά από τη συντακτική του ορθότητα αλλά κυρίως από τον βαθμό στον οποίο αποτυπώνει με ακρίβεια το νόημα των οντοτήτων, των διαδικασιών και των αλληλεπιδράσεων που αναπαριστά.

Η ανάλυση των Use Case Diagrams, Class Diagrams, Activity Diagrams και State Machine Diagrams ανέδειξε ότι η σημασιολογική πληρότητα αποτελεί κρίσιμο παράγοντα κατανόησης, επικοινωνίας και αξιοπιστίας των μοντέλων. Παρότι η UML παρέχει ένα τυποποιημένο σύνολο συμβολισμών, η ύπαρξη συντακτικά ορθών διαγραμμάτων δεν διασφαλίζει απαραίτητα την ορθή ερμηνεία τους από διαφορετικούς αναλυτές ή προγραμματιστές (Seidewitz, 2003).

Ιδιαίτερα σημαντικό εύρημα της εργασίας είναι ότι τα περισσότερα προβλήματα που εμφανίζονται κατά τη μοντελοποίηση δεν σχετίζονται με παραβιάσεις των κανόνων σύνταξης αλλά με σημασιολογικές ασάφειες. Η παρατήρηση αυτή επιβεβαιώθηκε τόσο από τη θεωρητική ανάλυση όσο και από τις δύο μελέτες περίπτωσης που εξετάστηκαν.

Στην περίπτωση του Arduino Uno και του κυκλώματος LED στο περιβάλλον Tinkercad, η σημασιολογική συνέπεια μεταξύ των λειτουργικών απαιτήσεων, των δραστηριοτήτων και των αλληλεπιδράσεων αποδείχθηκε καθοριστική για τη δημιουργία κατανοητών και πλήρων UML μοντέλων. Αντίστοιχα, στο πείραμα της Άνωσης στο περιβάλλον Klett Book-App, η ορθή αναπαράσταση των εκπαιδευτικών και πειραματικών διαδικασιών συνέβαλε στη δημιουργία μοντέλων που αποτυπώνουν με ακρίβεια τη μαθησιακή εμπειρία και τη λογική της πειραματικής διερεύνησης.

Ένα ακόμη σημαντικό εύρημα αφορά τα Activity Diagrams, τα οποία αποδείχθηκαν ιδιαίτερα αποτελεσματικά για τη μοντελοποίηση εκπαιδευτικών διαδικασιών και εικονικών εργαστηρίων. Ωστόσο, η αποτελεσματικότητά τους εξαρτάται από τη σαφή περιγραφή των

συνθηκών απόφασης, των παράλληλων ροών και των μεταβάσεων μεταξύ δραστηριοτήτων. Όπως έδειξε η ανάλυση, ακόμη και μικρές σημασιολογικές ασάφειες μπορούν να οδηγήσουν σε διαφορετικές ερμηνείες και, τελικά, σε διαφορετικές υλοποιήσεις.

9.2 Συσχέτιση με τη βιβλιογραφία

Τα αποτελέσματα της παρούσας έρευνας παρουσιάζουν σημαντική συμφωνία με τη σύγχρονη διεθνή βιβλιογραφία σχετικά με τη σημασιολογία των μοντέλων και τη Model-Driven Engineering.

Η διάκριση μεταξύ συντακτικής και σημασιολογικής ορθότητας, η οποία αναδείχθηκε ως κεντρικό συμπέρασμα της εργασίας, βρίσκεται σε συμφωνία με τον Seidewitz (2003), ο οποίος υποστηρίζει ότι η αξία ενός μοντέλου εξαρτάται από το νόημα που μεταφέρει και όχι μόνο από τη μορφή του. Αντίστοιχα, οι France και Rumpé (2007) επισημαίνουν ότι οι περισσότερες ασυνέπειες στα μοντέλα λογισμικού οφείλονται σε σημασιολογικές αποκλίσεις και όχι σε συντακτικά σφάλματα.

Η ανάλυση των Activity Diagrams και η σύνδεσή τους με εκπαιδευτικά περιβάλλοντα επιβεβαιώνει τα ευρήματα των Ma και Nickerson (2006), οι οποίοι υποστηρίζουν ότι η μοντελοποίηση εικονικών εργαστηρίων απαιτεί σαφή αναπαράσταση διαδικασιών, αλληλεπιδράσεων και ροών δεδομένων. Παράλληλα, η σύγκριση των Activity Diagrams με τη BPMN ανέδειξε ότι διαφορετικές τεχνικές μοντελοποίησης παρουσιάζουν διαφορετικά πλεονεκτήματα ανάλογα με το πεδίο εφαρμογής, γεγονός που συμφωνεί με τα συμπεράσματα των Dumas et al. (2018).

Επιπλέον, η ανάλυση των τυπικών σημασιολογικών προσεγγίσεων επιβεβαιώνει τη θέση των Harel και Rumpé (2004) ότι η χρήση αυστηρών σημασιολογικών μηχανισμών, όπως τα Petri Nets και η OCL, μπορεί να μειώσει σημαντικά την αμφισημία και να ενισχύσει την αξιοπιστία των μοντέλων.

Τέλος, τα ευρήματα σχετικά με τη χρήση μεγάλων γλωσσικών μοντέλων (LLMs) στη Model-Driven Engineering συμφωνούν με πρόσφατες μελέτες που υποστηρίζουν ότι η συντακτική επιτυχία της παραγόμενης υλοποίησης δεν εγγυάται απαραίτητα τη σημασιολογική της ορθότητα (Hou et al., 2024· Pearce et al., 2024).

9.3 Σημασία της Σημασιολογίας για την Ανάπτυξη Λογισμικού

Ένα από τα σημαντικότερα συμπεράσματα της παρούσας εργασίας αφορά τον ρόλο της σημασιολογίας στη διαδικασία ανάπτυξης λογισμικού και εκπαιδευτικών συστημάτων. Η σημασιολογία αναφέρεται στη σημασία που αποδίδεται στα στοιχεία ενός μοντέλου και στη σχέση τους με τις πραγματικές έννοιες και διαδικασίες του πεδίου εφαρμογής. Ένα μοντέλο θεωρείται σημασιολογικά πλήρες όταν περιλαμβάνει όλες τις απαραίτητες έννοιες και αλληλεπιδράσεις που απαιτούνται για την κατανόηση του συστήματος χωρίς σημαντικές παραλείψεις. (Wand & Weber, 2002).

Η ανάλυση των δύο μελετών περίπτωσης έδειξε ότι η σημασιολογική πληρότητα συμβάλλει ουσιαστικά στην ποιότητα των μοντέλων. Μέσω της σημασιολογικής ανάλυσης κατέστη δυνατή η αξιολόγηση της αντιστοίχισης μεταξύ των στοιχείων των μοντέλων και των πραγματικών δραστηριοτήτων που εκτελούνται στα εικονικά περιβάλλοντα.

Στην περίπτωση του Arduino, η σημασιολογική πληρότητα συνδέεται με την ορθή αναπαράσταση της διαδικασίας ανάπτυξης και εκτέλεσης του κυκλώματος. Στην περίπτωση του πειράματος της Άνωσης, η σημασιολογική πληρότητα συνδέεται με την ορθή αναπαράσταση της επιστημονικής διαδικασίας, από τη λήψη των μετρήσεων έως την εξαγωγή του τελικού συμπεράσματος.

Η σημασία της σημασιολογίας είναι ιδιαίτερα εμφανής στο πλαίσιο της ανάπτυξης λογισμικού, όπου τα μοντέλα λειτουργούν ως μέσο επικοινωνίας μεταξύ αναλυτών, προγραμματιστών, εκπαιδευτικών και τελικών χρηστών. Η ύπαρξη σημασιολογικά συνεπών μοντέλων μειώνει την πιθανότητα παρερμηνειών, διευκολύνει τη συντήρηση των συστημάτων και συμβάλλει στη βελτίωση της ποιότητας του τελικού προϊόντος (Satzinger, Jackson & Burd, 2021).

Επιπλέον, οι σύγχρονες προσεγγίσεις του Model-Driven Engineering (MDE) και του MBSE βασίζονται στην υπόθεση ότι τα μοντέλα μπορούν να αποτελέσουν το κεντρικό τεχνούργημα ανάπτυξης ενός συστήματος. Για να καταστεί αυτό εφικτό, απαιτείται υψηλό επίπεδο σημασιολογικής ακρίβειας και πληρότητας (Brambilla et al., 2017).

Συνεπώς, η σημασιολογία δεν αποτελεί απλώς ένα θεωρητικό χαρακτηριστικό των μοντέλων αλλά έναν κρίσιμο παράγοντα που επηρεάζει άμεσα την αποτελεσματικότητα της ανάλυσης απαιτήσεων, τη σχεδίαση συστημάτων και την ανάπτυξη λογισμικού.

Η παρούσα εργασία κατέδειξε ότι η χρήση UML μοντέλων μπορεί να υποστηρίξει αποτελεσματικά την ανάλυση και τεκμηρίωση εκπαιδευτικών δραστηριοτήτων που υλοποιούνται σε εικονικά περιβάλλοντα μάθησης. Οι δύο μελέτες περίπτωσης απέδειξαν ότι τα Use Case Models και τα Activity Diagrams μπορούν να αναπαραστήσουν με σαφήνεια τόσο τις λειτουργικές απαιτήσεις όσο και τη συμπεριφορά σύνθετων εκπαιδευτικών συστημάτων.

Η σημασιολογική ανάλυση ανέδειξε τον καθοριστικό ρόλο της σημασιολογικής πληρότητας στην ποιότητα των μοντέλων και επιβεβαίωσε ότι η συστηματική εφαρμογή των αρχών της UML συμβάλλει στη δημιουργία ακριβέστερων, κατανοητών και επαναχρησιμοποιήσιμων μοντέλων. Ως εκ τούτου, η σημασιολογικά τεκμηριωμένη μοντελοποίηση μπορεί να αποτελέσει σημαντικό εργαλείο τόσο για την εκπαιδευτική τεχνολογία όσο και για τη σύγχρονη ανάπτυξη λογισμικού.

Σημασιολογία και Model-Driven Engineering

Η σχέση μεταξύ σημασιολογίας και Model-Driven Engineering αποτέλεσε έναν από τους βασικούς άξονες της παρούσας εργασίας.

Η Model-Driven Engineering βασίζεται στην υπόθεση ότι τα μοντέλα μπορούν να χρησιμοποιηθούν ως πρωτεύοντα τεχνουργήματα ανάπτυξης λογισμικού και να μετασχηματιστούν σε εκτελέσιμο κώδικα μέσω αυτοματοποιημένων διαδικασιών (Schmidt, 2006). Ωστόσο, η αποτελεσματικότητα αυτής της προσέγγισης εξαρτάται άμεσα από τη σημασιολογική ποιότητα των μοντέλων.

Η ανάλυση που πραγματοποιήθηκε στο Κεφάλαιο 6 έδειξε ότι οι σημασιολογικές ασάφειες μεταφέρονται αναπόφευκτα στον παραγόμενο κώδικα. Επομένως, η ύπαρξη τυπικής σημασιολογίας και σαφώς καθορισμένων κανόνων μετασχηματισμού αποτελεί προϋπόθεση για την επιτυχή εφαρμογή της MDE.

Ιδιαίτερο ενδιαφέρον παρουσιάζει η αυξανόμενη χρήση τεχνητής νοημοσύνης και μεγάλων γλωσσικών μοντέλων στη διαδικασία παραγωγής κώδικα. Παρότι τα εργαλεία αυτά παρουσιάζουν εντυπωσιακές δυνατότητες αυτοματοποίησης, η παρούσα εργασία υποστηρίζει ότι η σημασιολογική επαλήθευση παραμένει απαραίτητη. Η παραγωγή συντακτικά ορθού κώδικα δεν διασφαλίζει ότι η λογική του παραγόμενου συστήματος είναι συνεπής με το αρχικό μοντέλο. Συνεπώς, η σημασιολογία αποτελεί θεμελιώδη προϋπόθεση για την αξιοπιστία της MDE και για την ασφαλή αξιοποίηση τεχνικών τεχνητής νοημοσύνης στην ανάπτυξη λογισμικού.

Επιπτώσεις για τα Εκπαιδευτικά Συστήματα και τα Εικονικά Εργαστήρια

Οι δύο μελέτες περίπτωσης που εξετάστηκαν στην παρούσα εργασία αναδεικνύουν τη σημασία της σημασιολογικής μοντελοποίησης στον χώρο της εκπαιδευτικής τεχνολογίας.

Η περίπτωση του Arduino Uno στο Tinkercad έδειξε ότι η UML μπορεί να χρησιμοποιηθεί αποτελεσματικά για την αναπαράσταση αλληλεπιδράσεων μεταξύ χρηστών, λογισμικού και φυσικών συσκευών. Αντίστοιχα, η περίπτωση του πειράματος της Άωσης στο Klett Book-App κατέδειξε ότι τα UML μοντέλα μπορούν να περιγράψουν με ακρίβεια σύνθετες μαθησιακές και πειραματικές διαδικασίες.

Τα αποτελέσματα αυτά υποδηλώνουν ότι η σημασιολογική μοντελοποίηση μπορεί να λειτουργήσει ως εργαλείο σχεδιασμού, αξιολόγησης και βελτίωσης εκπαιδευτικών εφαρμογών. Παράλληλα, η χρήση Activity Diagrams επιτρέπει την αναπαράσταση της μαθησιακής διαδικασίας με τρόπο που διευκολύνει την κατανόηση, την επαναχρησιμοποίηση και τη μελλοντική επέκταση των εκπαιδευτικών σεναρίων.

Συνεπώς, η σημασιολογική προσέγγιση που προτάθηκε στην παρούσα εργασία μπορεί να αξιοποιηθεί όχι μόνο σε παραδοσιακά πληροφοριακά συστήματα αλλά και σε περιβάλλοντα ψηφιακής μάθησης και εικονικών εργαστηρίων.

10. Συμπεράσματα και Μελλοντικές Κατευθύνσεις

10.1 Απάντηση στα Ερευνητικά Ερωτήματα

Η παρούσα διπλωματική εργασία διερεύνησε τον ρόλο της σημασιολογίας στη μοντελοποίηση λογισμικού μέσω της UML, καθώς και τη συμβολή της σημασιολογικής πληρότητας στην ανάπτυξη αξιόπιστων μοντέλων και συστημάτων λογισμικού. Η έρευνα οργανώθηκε γύρω από έξι βασικά ερευνητικά ερωτήματα, τα οποία απαντώνται συνοπτικά στη συνέχεια:

EE1: Ποιος είναι ο ρόλος της σημασιολογίας στη σωστή ερμηνεία και κατανόηση των UML μοντέλων;

Η ανάλυση έδειξε ότι η σημασιολογία αποτελεί τον βασικό μηχανισμό μέσω του οποίου αποδίδεται νόημα στα στοιχεία των UML μοντέλων. Η σωστή κατανόηση ενός διαγράμματος δεν εξαρτάται μόνο από τη γνώση των συμβολισμών της UML αλλά και από τη σαφή ερμηνεία των εννοιών που αναπαριστώνται. Η διάκριση μεταξύ συντακτικής και σημασιολογικής ορθότητας αποδείχθηκε καθοριστική για την κατανόηση της ποιότητας των μοντέλων (Seidewitz, 2003).

EE2: Πώς επηρεάζει η σημασιολογική πληρότητα την ποιότητα και την αξιοπιστία των UML διαγραμμάτων;

Τα αποτελέσματα της έρευνας έδειξαν ότι η σημασιολογική πληρότητα αποτελεί βασικό παράγοντα ποιότητας των μοντέλων. Όσο πληρέστερα αποτυπώνονται οι έννοιες, οι διαδικασίες και οι σχέσεις ενός συστήματος, τόσο μειώνεται η πιθανότητα παρερμηνειών και σφαλμάτων κατά τη φάση της υλοποίησης. Η σημασιολογική πληρότητα λειτουργεί επομένως ως μηχανισμός διασφάλισης της συνέπειας μεταξύ απαιτήσεων, σχεδίασης και υλοποίησης.

EE3: Ποια είναι τα συνηθέστερα σημασιολογικά σφάλματα στα UML διαγράμματα;

Η ανάλυση των Use Case, Class, Activity και State Machine Diagrams ανέδειξε ότι τα συνηθέστερα σφάλματα αφορούν ασάφειες στις απαιτήσεις, λανθασμένη χρήση σχέσεων, ελλιπή περιγραφή συνθηκών απόφασης, μη προσβάσιμες καταστάσεις και ασυνέπειες

μεταξύ διαφορετικών διαγραμμάτων. Τα σφάλματα αυτά εμφανίζονται ακόμη και σε συντακτικά ορθά μοντέλα, γεγονός που αναδεικνύει τη σημασία της σημασιολογικής αξιολόγησης (France & Rumpe, 2007).

EE4: Πώς μπορούν οι τυπικές σημασιολογικές προσεγγίσεις να συμβάλουν στη μείωση των σημασιολογικών ασυνεπειών;

Οι τυπικές προσεγγίσεις, όπως η Object Constraint Language (OCL), τα Petri Nets και οι formal semantics, μπορούν να περιορίσουν σημαντικά την αμφισημία και να υποστηρίξουν την επαλήθευση της ορθότητας των μοντέλων. Παράλληλα, η χρήση τους επιτρέπει τον έγκαιρο εντοπισμό σφαλμάτων πριν από τη φάση της υλοποίησης (Harel & Rumpe, 2004).

EE5: Ποιος είναι ο ρόλος της σημασιολογικής πληρότητας στη Model-Driven Engineering;

Η έρευνα έδειξε ότι η σημασιολογική πληρότητα αποτελεί προϋπόθεση για την επιτυχή εφαρμογή της Model-Driven Engineering. Η παραγωγή κώδικα μέσω αυτοματοποιημένων μετασχηματισμών προϋποθέτει σαφώς ορισμένη σημασιολογία, καθώς οποιαδήποτε ασάφεια ή ασυνέπεια μεταφέρεται στον παραγόμενο κώδικα (Schmidt, 2006).

EE6: Σε ποιο βαθμό τα UML μοντέλα που αναπτύσσονται για εκπαιδευτικά περιβάλλοντα μπορούν να θεωρηθούν σημασιολογικά πλήρη;

Η αξιολόγηση των δύο μελετών περίπτωσης έδειξε ότι τα UML μοντέλα μπορούν να επιτύχουν υψηλό βαθμό σημασιολογικής πληρότητας όταν συνοδεύονται από σαφείς περιγραφές απαιτήσεων, συνεπή χρήση συμβολισμών και κατάλληλη σημασιολογική τεκμηρίωση. Η σημασιολογική πληρότητα αποδείχθηκε ιδιαίτερα σημαντική για την αναπαράσταση πειραματικών διαδικασιών και μαθησιακών αλληλεπιδράσεων.

10.2 Συμπεράσματα της Εργασίας

Η παρούσα εργασία κατέδειξε ότι η σημασιολογία δεν αποτελεί δευτερεύον χαρακτηριστικό της μοντελοποίησης αλλά θεμελιώδη παράγοντα ποιότητας των UML μοντέλων.

Η θεωρητική ανάλυση ανέδειξε ότι η ύπαρξη κοινής σύνταξης δεν συνεπάγεται απαραίτητα και κοινή κατανόηση των μοντέλων. Αντίθετα, η σημασιολογική διάσταση καθορίζει τον τρόπο με τον οποίο οι αναλυτές, οι σχεδιαστές και οι προγραμματιστές ερμηνεύουν τα στοιχεία ενός μοντέλου και τα μετατρέπουν σε λειτουργικό λογισμικό.

Οι μελέτες περίπτωσης επιβεβαίωσαν ότι η σημασιολογική πληρότητα μπορεί να εφαρμοστεί αποτελεσματικά σε διαφορετικά πεδία, από κυβερνοφυσικά συστήματα όπως το Arduino έως εικονικά εργαστήρια φυσικών επιστημών. Παράλληλα, η ανάλυση των Activity Diagrams ανέδειξε τη χρησιμότητά τους για την αναπαράσταση εκπαιδευτικών διαδικασιών, ενώ η σύγκριση με τη BPMN και η σύνδεση με τα Petri Nets κατέδειξαν ότι η επιλογή της κατάλληλης τεχνικής μοντελοποίησης εξαρτάται από τον επιδιωκόμενο βαθμό σημασιολογικής αυστηρότητας.

Ιδιαίτερη σημασία έχει επίσης το συμπέρασμα ότι η επιτυχία της Model-Driven Engineering εξαρτάται άμεσα από τη σημασιολογική ποιότητα των μοντέλων. Η παραγωγή συντακτικά ορθού κώδικα δεν διασφαλίζει από μόνη της την ορθή συμπεριφορά του συστήματος, γεγονός που καθιστά απαραίτητη τη σημασιολογική επαλήθευση των μοντέλων πριν από τους μετασχηματισμούς.

Τέλος, η εργασία ανέδειξε ότι η αυξανόμενη χρήση τεχνητής νοημοσύνης και μεγάλων γλωσσικών μοντέλων στην ανάπτυξη λογισμικού δεν μειώνει τη σημασία της σημασιολογίας. Αντίθετα, η ύπαρξη σαφών και σημασιολογικά συνεπών μοντέλων καθίσταται ακόμη πιο κρίσιμη σε περιβάλλοντα όπου η παραγωγή κώδικα πραγματοποιείται μερικώς ή πλήρως αυτοματοποιημένα.

10.3 Περιορισμοί της Μελέτης

Παρά τα ευρήματα της έρευνας, η παρούσα εργασία παρουσιάζει ορισμένους περιορισμούς:

Πρώτον, η αξιολόγηση της σημασιολογικής πληρότητας πραγματοποιήθηκε μέσω δύο συγκεκριμένων μελετών περίπτωσης που αφορούν εκπαιδευτικά περιβάλλοντα. Επομένως, τα αποτελέσματα δεν μπορούν να γενικευθούν χωρίς επιφύλαξη σε όλα τα είδη πληροφοριακών ή βιομηχανικών συστημάτων.

Δεύτερον, η έρευνα επικεντρώθηκε κυρίως στα Use Case, Class, Activity και State Machine Diagrams. Άλλοι τύποι UML διαγραμμάτων, όπως τα Sequence Diagrams, Component Diagrams ή Deployment Diagrams, δεν εξετάστηκαν με την ίδια λεπτομέρεια.

Τρίτον, οι τυπικές σημασιολογικές προσεγγίσεις παρουσιάστηκαν κυρίως σε θεωρητικό επίπεδο και δεν εφαρμόστηκαν εκτενώς σε πλήρως εκτελέσιμα βιομηχανικά συστήματα. Ως εκ τούτου, η πρακτική αξιολόγηση της αποτελεσματικότητάς τους απαιτεί περαιτέρω διερεύνηση.

Τέλος, η ανάλυση των εργαλείων τεχνητής νοημοσύνης και των μεγάλων γλωσσικών μοντέλων βασίστηκε κυρίως σε πρόσφατη βιβλιογραφία και όχι σε εκτεταμένα πειραματικά δεδομένα που προέκυψαν από την παρούσα εργασία.

10.4 Προτάσεις για Περαιτέρω Έρευνα

Τα αποτελέσματα της παρούσας εργασίας δημιουργούν αρκετές προοπτικές για μελλοντική έρευνα.

Μία πρώτη κατεύθυνση αφορά την ανάπτυξη μεθοδολογιών αυτόματης αξιολόγησης της σημασιολογικής πληρότητας UML μοντέλων μέσω τεχνικών τεχνητής νοημοσύνης και μηχανικής μάθησης. Η κατεύθυνση αυτή θα μπορούσε να οδηγήσει σε εργαλεία υποστήριξης αναλυτών και σχεδιαστών κατά τη διαδικασία μοντελοποίησης.

Μία δεύτερη προοπτική αφορά τη διερεύνηση της σημασιολογικής πληρότητας σε πρόσθετους τύπους UML διαγραμμάτων και σε σύνθετα βιομηχανικά συστήματα, όπου οι απαιτήσεις αξιοπιστίας είναι ιδιαίτερα αυξημένες.

Επιπλέον, ιδιαίτερο ενδιαφέρον παρουσιάζει η περαιτέρω μελέτη της σχέσης μεταξύ UML, BPMN και άλλων γλωσσών μοντελοποίησης διαδικασιών, με στόχο την ανάπτυξη ενοποιημένων πλαισίων σημασιολογικής ανάλυσης.

Τέλος, η ταχεία εξέλιξη των μεγάλων γλωσσικών μοντέλων δημιουργεί την ανάγκη διερεύνησης νέων μηχανισμών σημασιολογικής επαλήθευσης του παραγόμενου κώδικα. Η συνδυαστική χρήση τυπικών σημασιολογικών τεχνικών και τεχνητής νοημοσύνης αποτελεί ένα ιδιαίτερα υποσχόμενο ερευνητικό πεδίο για τα επόμενα χρόνια.

Βιβλιογραφικές Αναφορές

- Al-Azzoni, I., et al. (2025). Model-driven engineering and artificial intelligence for smart contract generation. Information.
- Ambler, S. W. (2004). *The object primer: Agile model-driven development with UML 2.0* (3rd ed.). Cambridge University Press.
- Antal, G., Vozár, R., & Ferenc, R. (2024). Assessing GPT-4-Vision's Capabilities in UML-Based Code Generation. arXiv:2404.14370.
- Antal, M., et al. (2024). UML diagram to code generation using large language models.
- Bates, A. et al. (2025). Unified Modeling Language Code Generation from Diagram Images Using Multimodal Large Language Models. arXiv:2503.12293.
- Bates, A., et al. (2025). UML Code Generation using Multimodal Large Language Models. arXiv.
- Bézivin, J. (2005). On the unification power of models. *Software and Systems Modeling*, 4(2), 171–188. <https://doi.org/10.1007/s10270-005-0079-0>
- Booch, G., Rumbaugh, J., & Jacobson, I. (2007). *The Unified Modeling Language User Guide* (2nd ed.). Addison-Wesley.
- Booch, G. (1991). *Object-oriented design with applications*. Benjamin/Cummings.
- Brambilla, M., Cabot, J., & Wimmer, M. (2017). *Model-driven software engineering in practice* (2nd ed.). Morgan & Claypool Publishers.
- Bransford, J. D., Brown, A. L., & Cocking, R. R. (2020). *How People Learn II: Learners, Contexts, and Cultures*. National Academies Press.
- Brinson, J. R. (2015). Learning outcome achievement in non-traditional (virtual and remote) versus traditional (hands-on) laboratories: A review of the empirical research. *Computers & Education*, 87, 218–237.
- Broy, M., & Stølen, K. (2001). *Specification and development of interactive systems: Focus on streams, interfaces, and refinement*. Springer.

- Cabot, J., & Teniente, E. (2006). Constraint inference in UML conceptual models. *Software and Systems Modeling*, 10(3), 309–330.
- Chan, P., Van Gerven, T., Dubois, J., & Bernaerts, K. (2021). Virtual chemical laboratories: A systematic literature review of research, technologies and instructional design. *Computers and Education Open*, 2, 100053. <https://doi.org/10.1016/j.caeo.2021.100053>
- Cockburn, A. (2001). *Writing Effective Use Cases*. Addison-Wesley.
- Dennis, A., Wixom, B. H., & Tegarden, D. (2020). *Systems Analysis and Design: An Object-Oriented Approach with UML* (6th ed.). Wiley.
- Dewey, J. (1938). *Experience and education*. Macmillan.
- Di Ruscio, D., Kolovos, D., de Lara, J. *et al.* Low-code development and model-driven engineering: Two sides of the same coin?. *Softw Syst Model* **21**, 437–446 (2022). <https://doi.org/10.1007/s10270-021-00970-2>
- Douglass, B. P. (2016). *Real-Time UML Workshop for Embedded Systems* (3rd ed.). Elsevier.
- Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. A. (2018). *Fundamentals of Business Process Management* (2nd ed.). Springer.
- Elmoazen, R., Saqr, M., Khalil, M., & Wasson, B. (2023). Learning analytics in virtual laboratories: A systematic literature review of empirical research. *Smart Learning Environments*, 10(23). <https://doi.org/10.1186/s40561-023-00244-y>
- Engels, G., et al. (2023). Consistency in software models and architectures.
- Eriksson, H. E., & Penker, M. (2000). *Business Modeling with UML: Business Patterns at Work*. Wiley.
- Fowler, M. (2004). *UML Distilled: A Brief Guide to the Standard Object Modeling Language* (3rd ed.). Addison-Wesley.
- Fowler, M. (2018). *UML Distilled: A Brief Guide to the Standard Object Modeling Language* (3rd ed., Kindle ed.). Addison-Wesley Professional.
- France, R., & Rumpe, B. (2007). Model-driven development of complex software: A research roadmap. In *Future of software engineering (FOSE 2007)* (pp. 37–54). IEEE Computer Society. <https://doi.org/10.1109/FOSE.2007.14>

- Friedenthal, S., Moore, A., & Steiner, R. (2015). *A Practical Guide to SysML* (3rd ed.). Morgan Kaufmann.
- García-Magariño, I., et al. (2020). Code generation using model driven architecture: A systematic mapping study. *Journal of Computer Languages*, 56.
- Guarino, N. (1998). Formal ontology and information systems. In N. Guarino (Ed.), *Proceedings of FOIS 1998: International Conference on Formal Ontology in Information Systems* (pp. 3–15). IOS Press.
- Guizzardi, G. (2005). *Ontological foundations for structural conceptual models*. Universal Press.
- Harel, D., & Rumpe, B. (2004). Meaningful modeling: What's the semantics of “semantics”? *Computer*, 37(10), 64–72. <https://doi.org/10.1109/MC.2004.172>
- Herber, D. R., & Wheaton, J. S. (2024). *Digital Requirements Engineering with an INCOSE-Derived SysML Meta-Model*. arXiv.
- Hitzler, P., Krötzsch, M., & Rudolph, S. (2023). *Foundations of semantic web technologies* (2nd ed.). CRC Press.
- Hmelo-Silver, C. E. (2019). Problem-Based Learning. In K. Sawyer (Ed.), *The Cambridge Handbook of the Learning Sciences* (3rd ed.). Cambridge University Press.
- Hou, X., Wang, Z., Yin, D., & Liu, B. (2024). Large Language Models for Code Generation: A Systematic Literature Review. *ACM Computing Surveys*. DOI: <https://doi.org/10.1145/3702518>
- Jacobson, I., Christerson, M., Jonsson, P., & Övergaard, G. (1992). *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley.
- Juhás, G., Lehocki, F., & Lorenz, P. (2007). Modeling of Concurrent Systems Using Petri Nets.
- Kent, S. (2002). Model-driven engineering. In *Proceedings of IFM 2002*. Springer.
- Kleppe, A., Warmer, J., & Bast, W. (2003). *MDA Explained: The Model Driven Architecture*. Addison-Wesley.

- Kolb, D. A. (1984). *Experiential learning: Experience as the source of learning and development*. Prentice Hall.
- Koltun, G., & Pundel, M. (2022). Model-driven engineering of mechatronic systems using VIATRA. *Software and Systems Modeling*.
- Κόμης, Β. (2011). *Εισαγωγή στις εκπαιδευτικές εφαρμογές των ΤΠΕ*. Νέων Τεχνολογιών.
- Lano, K., & Siala, H. (2024). Using model-driven engineering to automate software language translation. *Automated Software Engineering*, 31(20)
- Lano, K., et al. (2024). Agile model-driven re-engineering. *Innovations in Systems and Software Engineering*.
- Larman, C. (2004). *Applying UML and patterns: An introduction to object-oriented analysis and design and iterative development* (3rd ed.). Prentice Hall.
- Laurillard, D. (2023). *Teaching as a Design Science: Building Pedagogical Patterns for Learning and Technology*. Routledge.
- Levesque, H., Pirri, F., Belle, V., & Lakemeyer, G. (2019). *The Situation Calculus: A Foundation for Knowledge Representation and Reasoning*.
- Λιοναράκης, Α. (2006). *Θεωρίες μάθησης και εξ αποστάσεως εκπαίδευση*. Προπομπός
- Ma, J., & Nickerson, J. V. (2006). Hands-On, Simulated, and Remote Laboratories: A Comparative Literature Review. *ACM Computing Surveys (CSUR)*, 38(3). DOI: <https://doi.org/10.1145/1132960.1132961>
- Μακράκης, Β. (2000). *Υπερμέσα στην εκπαίδευση*. Μεταίχμιο.
- Marriott, K., Stuckey, P., & Wallace, M. (2006). *Constraint Logic Programming and Operational Semantics*.
- Mellor, S. J., Scott, K., Uhl, A., & Weise, D. (2004). *MDA Distilled: Principles of Model-Driven Architecture*. Addison-Wesley.
- Mellor, S., Clark, A., & Futagami, T. (2003). Model-driven development—Guest editor introduction. *IEEE Software*, 20(5), 14–18.

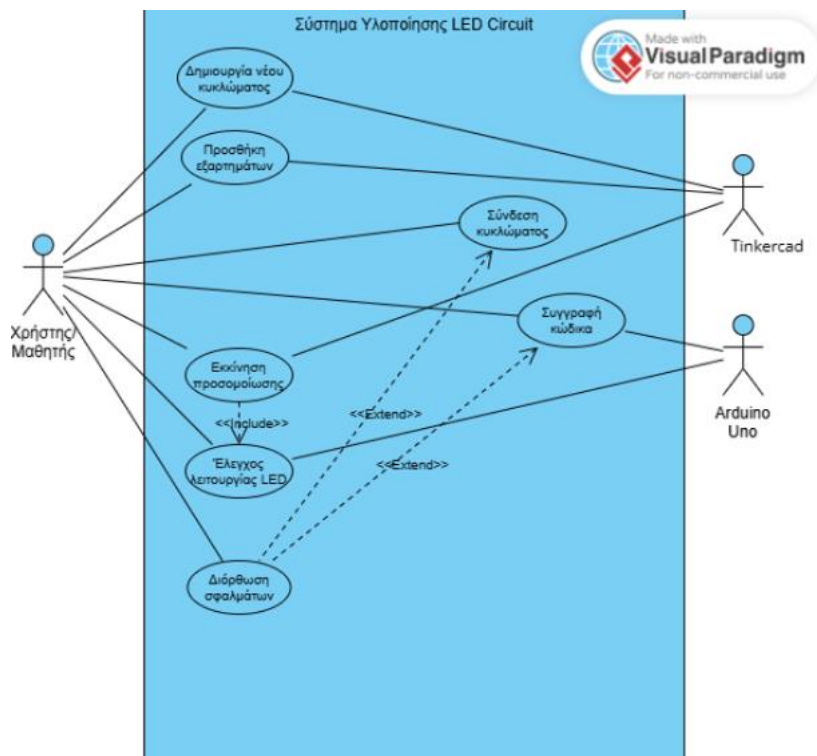
- Merchant, Z., Goetz, E. T., Cifuentes, L., Keeney-Kennicutt, W., & Davis, T. J. (2014). Effectiveness of virtual reality-based instruction on students' learning outcomes in K–12 and higher education: A meta-analysis. *Computers & Education*, 70, 29–40.
- Mikropoulos, T. A., & Natsis, A. (2011). Educational virtual environments: A ten-year review of empirical research. *Computers & Education*, 56(3), 769–780.
- MontiGem Study (2024). Lessons learned from applying model-driven engineering in multiple domains. Science of Computer Programming.
- Μπεληγιάννης Γρηγόριος. «Πληροφοριακά Συστήματα Διοίκησης. Διαγράμματα Δραστηριοτήτων». Έκδοση: 1.0. Πανεπιστήμιο Πατρών. Πάτρα 2015. Διαθέσιμο από τη δικτυακή διεύθυνση:
<https://eclass.upatras.gr/modules/document/document.php?course=DEAPT111>.
- Muthumanikandan, V., et al. (2024). Screen2Web: UML-based web code generation. Information Discovery and Delivery.
- Noy, N., Gao, Y., Jain, A., Narayanan, A., Patterson, A., & Taylor, J. (2019). Industry-scale knowledge graphs: Lessons and challenges. *Communications of the ACM*, 62(8), 36–43.
<https://doi.org/10.1145/3331166>
- O’Keefe, G. (2002). Dynamic logic semantics for UML consistency.
- Object Management Group (OMG). (2023). *Unified Modeling Language (UML) Version 2.5.1 Specification*. Available at: [OMG UML Specification](#)
- Object Management Group. (2017). *OMG unified modeling language (OMG UML), version 2.5.1*. Object Management Group. <https://www.omg.org/spec/UML/2.5.1>
- Pan, J. Z., Vetere, G., Gomez-Perez, J. M., & Wu, H. (Eds.). (2024). *Knowledge graphs: Fundamentals, techniques and applications*. Springer. <https://doi.org/10.1007/978-3-031-19433-7>
- Pearce, H., Ahmad, B., Tan, B., et al. (2024). Assessing the Security and Reliability of AI-Assisted Code Generation. *IEEE Symposium on Security and Privacy*. DOI: <https://doi.org/10.1109/SP54263.2024.00145>

- Peterson, J. L. (1981). Petri Net Theory and the Modeling of Systems. Prentice Hall.
- Piaget, J. (1970). Science of education and the psychology of the child. Viking Press.
- Potkonjak, V., Gardner, M., Callaghan, V., Mattila, P., Guetl, C., Petrović, V. M., & Jovanović, K. (2016). Virtual laboratories for education in science, technology, and engineering: A review. *Computers & Education*, 95, 309–327.
- Πρέντζα Ανδριάνα (2024). *Τεχνολογία Λογισμικού*, Πανεπιστήμιο Πειραιώς.
- Pressman, R. S., & Maxim, B. R. (2020). *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill.
- Pressman, R. S., & Maxim, B. R. (2015). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill.
- Rädler, S., Rupp, M., Rigger, E., & Rinderle-Ma, S. (2024). Model-Driven Engineering for Machine Learning Code Generation using SysML. In *Modellierung 2024*.
- Raman, R., Achuthan, K., Nair, V. K., & Nedungadi, P. (2022). Virtual laboratories: A historical review and bibliometric analysis of the past three decades. *Education and Information Technologies*, 27(8), 11055–11087. <https://doi.org/10.1007/s10639-022-11058-9>
- Rumbaugh, J., Jacobson, I., & Booch, G. (2005). The Unified Modeling Language Reference Manual (2nd ed.). Addison-Wesley.
- Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., & Lorensen, W. (1991). *Object-oriented modeling and design*. Prentice Hall.
- Sadik, A. R., et al. (2024). LLM as a code generator in Agile Model Driven Development. arXiv.
- Sapriati, A., Suhandoko, A. D. J., Yundayani, A., Karim, R. A., Kusmawan, U., Adnan, A. H. M., & Suhandoko, A. A. (2023). The effect of virtual laboratories on improving students' self-regulated learning: An umbrella systematic review. *Education Sciences*, 13(3), 222. <https://doi.org/10.3390/educsci13030222>

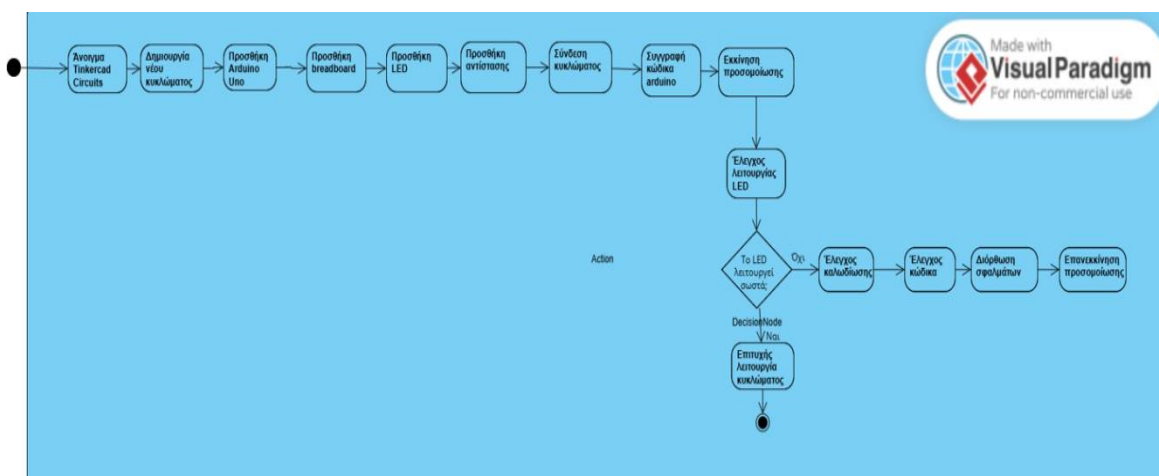
- Satzinger, J. W., Jackson, R. B., & Burd, S. D. (2021). *Systems Analysis and Design in a Changing World* (8th ed.). Cengage Learning.
- Schmidt, D. C. (2006). Model-Driven Engineering. *Computer*, 39(2), 25–31. DOI: <https://doi.org/10.1109/MC.2006.58>
- Seidewitz, E. (2003). What Models Mean. *IEEE Software*, 20(5), 26–32. DOI: <https://doi.org/10.1109/MS.2003.1231147>
- Selic, B. (2023). Model-Driven Engineering in Practice: Current Challenges and Future Directions. *IEEE Software*.
- Selic, B. (2012). What will it take? A view on adoption of model-based methods in practice. *Software and Systems Modeling*, 11(4), 513–526.
- Sendall, S., & Kozaczynski, W. (2003). Model Transformation: The Heart and Soul of Model-Driven Software Development. *IEEE Software*, 20(5), 42–45. DOI: **10.1109/MS.2003.1231150**
- Silver, B. (2020). *BPMN Method and Style* (3rd ed.). Cody-Cassidy Press
- Sommerville, I. (2016). *Software Engineering* (10th Global Edition). Pearson Education
- Spanoudakis, G., & Zisman, A. (2001). Inconsistency management in software engineering.
- Spector, J. M. (2022). *Learning, Design, and Technology: An International Compendium of Theory, Research, Practice, and Policy*. Springer.
- Störrle, H. (2021). *Semantics and Verification of UML Models*. Springer
- StructGen. (2026). Structure-guided UML-based code generation using large language models. *Neurocomputing*.
- Usman, M., et al. (2024). MOBICAT: Model-driven GUI code generation for Android applications. *Frontiers in Computer Science*.
- van der Aalst, W. (2016). *Process Mining: Data Science in Action* (2nd ed.). Springer. DOI: **10.1007/978-3-662-49851-4**
- Visual Paradigm Research & Innovation Lab. (2025). UML in the Age of AI: Intelligent Modeling Ecosystems. Visual Paradigm Guides.

- Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes*. Harvard University Press.
- Tsang, C. (2025). *AI-Augmented Software Modeling and UML Evolution in Enterprise Systems*. Visual Paradigm Publications.
- Wand, Y., & Weber, R. (2002). Research Commentary: Information Systems and Conceptual Modeling—A Research Agenda. *Information Systems Research*, 13(4), 363–376.
- Warmer, J., & Kleppe, A. (2003). *The Object Constraint Language: Getting Your Models Ready for MDA*. Addison-Wesley
- Whittle, J., Hutchinson, J., & Rouncefield, M. (2014). The state of practice in model-driven engineering. *IEEE Software*, 31(3), 79–85. <https://doi.org/10.1109/MS.2013.65>
- Wimmer, M., Egyed, A., Combemale, B., & Chechik, M. (Eds.). (2024). *Proceedings of the ACM/IEEE MODELS 2024*.
- Wohed, P., van der Aalst, W., Dumas, M., ter Hofstede, A., & Russell, N. (2006). On the Suitability of BPMN for Business Process Modelling. DOI: 10.1007/11837862_8
- Zhang, X., Al-Mekhled, D., & Choate, J. (2021). Are virtual physiology laboratories effective for student learning? A systematic review. *Advances in Physiology Education*, 45(3), 467–480. <https://doi.org/10.1152/advan.00016.2021>

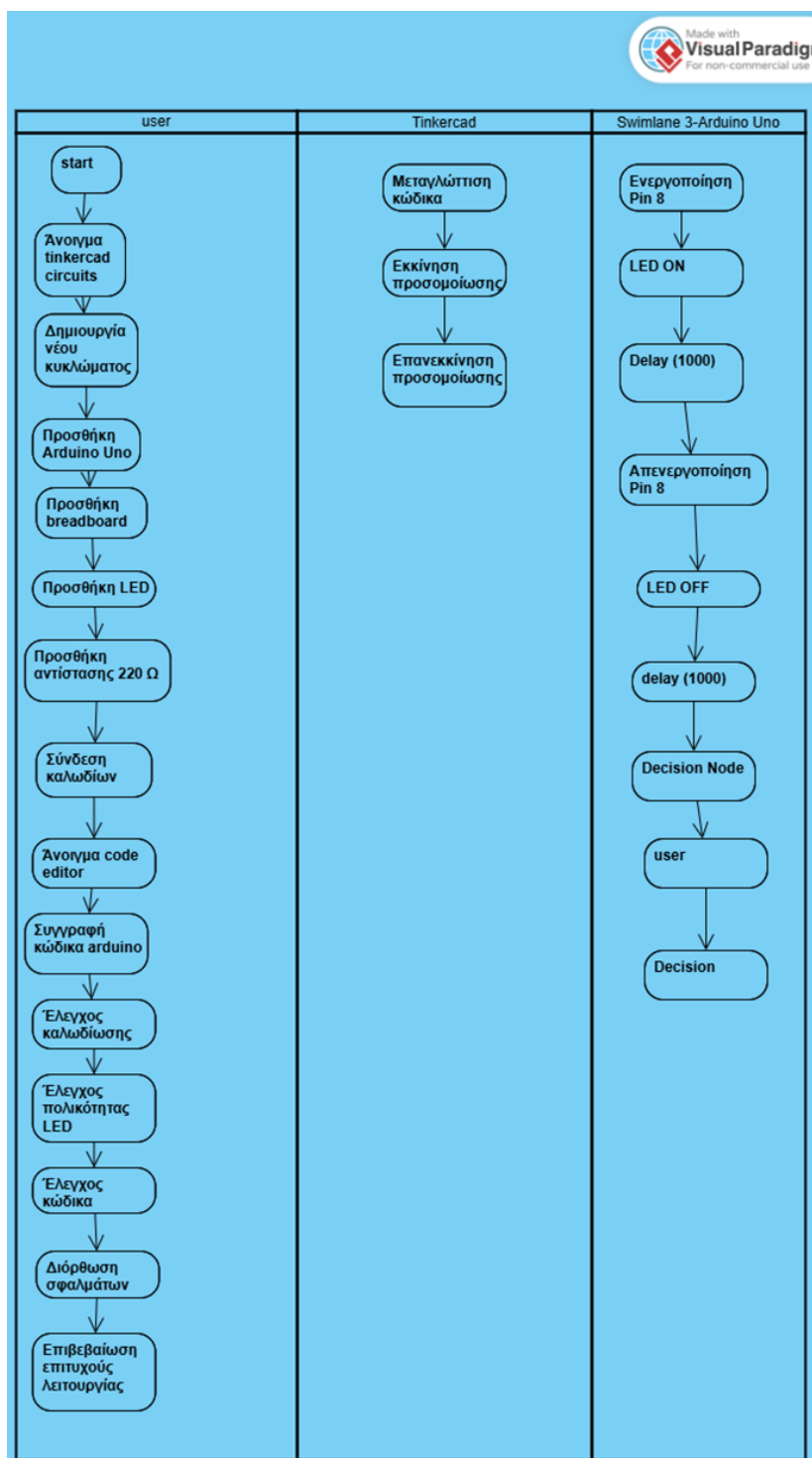
Παράρτημα Α: «1^η μελέτη περίπτωσης: Σύστημα Ελέγχου LED με Arduino Uno στο Tinkercad»



Εικ. 7- 3 Δημιουργία Use Case Models για Σύστημα Ελέγχου LED με Arduino Uno στο Tinkercad

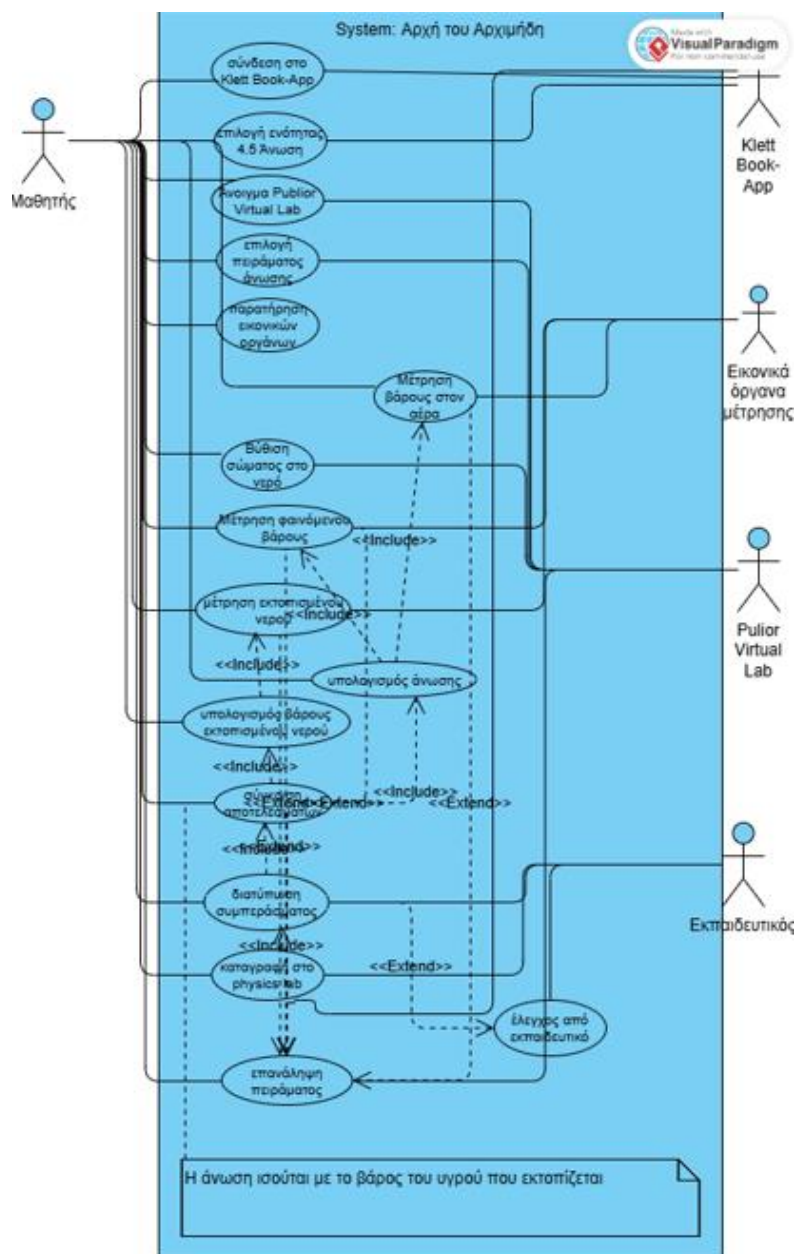


Εικ.7.4 Activity Diagram για Σύστημα Ελέγχου LED με Arduino Uno στο Tinkercad

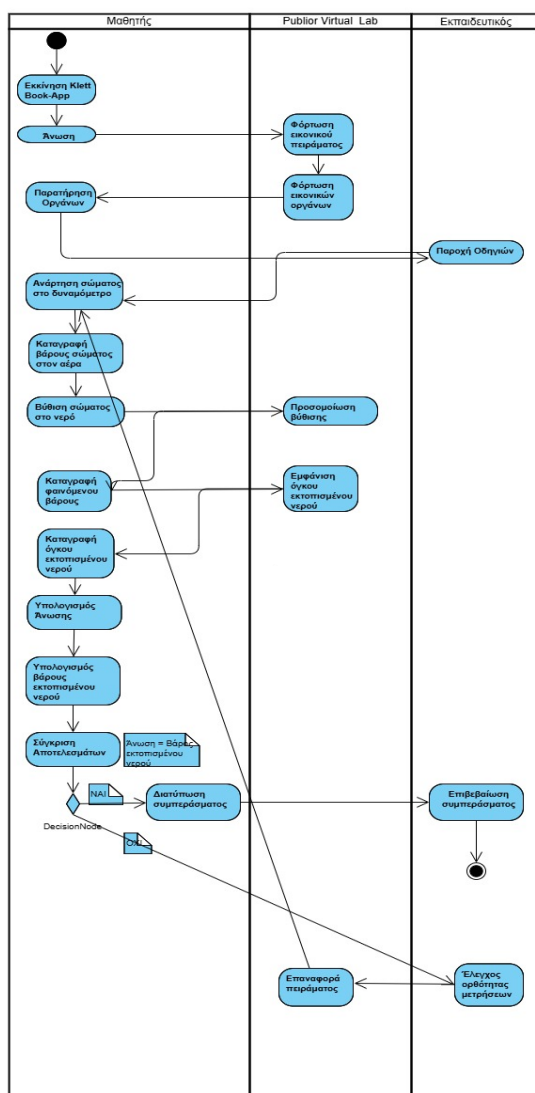


Εικ.7.5 Χρήση Swimlanes για Κατανομή Ευθυνών για Σύστημα Ελέγχου LED με Arduino Uno στο Tinkercad

Παράρτημα Β: «2^η μελέτη περίπτωσης: Περιγραφή του Πειράματος της Άνωσης στο Εικονικό Εργαστήριο Klett / Publior Virtual Lab»



Εικ. 8-2. Δημιουργία Use Case Models για την περιγραφή του Πειράματος της Άνωσης στο Εικονικό Εργαστήριο Klett / Publior Virtual Lab



Εικ. 8-3 Δημιουργία Activity Diagrams για την περιγραφή του Πειράματος της Άνωσης στο Εικονικό Εργαστήριο Klett / Publior Virtual Lab

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.