ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Πληροφορική και Τηλεματική
“Πληροφοριακά Συστήματα στη Διοίκηση Επιχειρήσεων”

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Προσομοίωση Γενικών Μοντέλων Συστημάτων (SysML)

ΠΑΠΑΔΗΜΟΣ ΠΑΝΑΓΙΩΤΗΣ

Αθήνα,
Ιούλιος 2021

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Πληροφορική και Τηλεματική
“Πληροφοριακά Συστήματα στη Διοίκηση Επιχειρήσεων”

Τριμελής Εξεταστική Επιτροπή

Γεώργιος-Δημήτριος Κάπος (Επιβλέπων)

Διδάσκων με Ανάθεση, Πληροφορική και Τηλεματική, Χαροκόπειο Πανεπιστήμιο

Βασίλειος Δαλάκας

Ε.ΔΙ.Π., Πληροφορική και Τηλεματική, Χαροκόπειο Πανεπιστήμιο

Ανάργυρος Τσαδήμας

Ε.ΔΙ.Π., Πληροφορική και Τηλεματική, Χαροκόπειο Πανεπιστήμιο

Ο Παναγιώτης Παπαδήμος

δηλώνω υπεύθυνα ότι:

1) Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλει τα πνευματικά δικαιώματα τρίτων.

2) Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή/και σε οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονείται στα πλαίσια του Π.Μ.Σ. "Πληροφορικής και Τηλεματικής" του τμήματος Πληροφορικής και Τηλεματικής, του Χαροκοπείου Πανεπιστημίου.

Με την περάτωση της παρούσας εργασίας, θα ήθελα να ευχαριστήσω τον επιβλέποντα Διδάσκοντα Γεώργιο-Δημήτριο Κάπο. Η υποστήριξη και η διαθεσιμότητά του καθ΄ όλη τη διάρκεια της εκπόνησης της εργασίας αποτέλεσε σπουδαία βοήθεια, συμβάλλοντας ποικιλοτρόπως στην ολοκλήρωσή της, παρέχοντας μεταξύ άλλων πολύτιμες συμβουλές και καθοδήγηση όπου κρίθηκε αναγκαίο.

Επίσης, θα ήθελα να ευχαριστήσω τον κ. Δαλάκα Βασίλειο, Ε.ΔΙ.Π. του τμήματος Πληροφορικής και Τηλεματικής και τον Ε.ΔΙ.Π. του τμήματος Πληροφορικής και Τηλεματικής κ. Ανάργυρο Τσαδήμα, για τη συμμετοχή τους στην τριμελή εξεταστική επιτροπή της παρούσας εργασίας.

Επιπλέον, νιώθω την ανάγκη να ευχαριστήσω όλους τους καθηγητές και τους συμφοιτητές με τους οποίους συνεργάστηκα κατά τη διάρκεια του μεταπτυχιακού.

Ιδιαίτερα θερμές ευχαριστίες θέλω να δώσω στην οικογένειά μου για την συνεχή συμπαράστασή τους, για τις πολύτιμες συμβουλές τους και για όλα όσα μου έχουν προσφέρει όλα αυτά τα χρόνια της ζωής μου αλλά και των σπουδών μου.

Πίνακας Περιεχομένων

Περίληψη

Τα φυσικά συστήματα που συναντάμε καθημερινά και τα οποία άλλοτε μας διευκολύνουν και άλλοτε όχι, πλέον μπορούν μέσω της μοντελοποίησης συστημάτων να αναλυθούν και να σχεδιαστούν, ώστε να υπάρχει μια πιο καθαρή εικόνα για τη χρήση τους. Επιπλέον, μέσα από την προσομοίωση μπορούμε να προλάβουμε κάποια δυσλειτουργία του συστήματος αλλά και να περιορίσουμε τυχόν αστοχίες που μπορούν να υπάρξουν. Ένα ακόμα σημαντικό στοιχείο της προσομοίωσης, είναι ότι μας παρέχεται μια πληρέστερη εικόνα της λειτουργίας του συστήματος.

Στην παρούσα εργασία, θα ακολουθήσουμε έναν γενικό τρόπο μοντελοποίησης συστημάτων SysML, χωρίς κάποια επέκταση για συγκεκριμένο πεδίο εφαρμογής ή περιβάλλον προσομοίωσης και θα μπορεί να αξιοποιηθεί σε οποιοδήποτε μοντέλο συστήματος. Για να υλοποιήσουμε αυτόν τον τρόπο μοντελοποίησης, θα χρησιμοποιήσουμε βασικά διαγράμματα της SySML και βασικές ιδιότητες που μας παρέχει η γλώσσα.

Στη διπλωματική, γίνεται χρήση εργαλείων με τα οποία παρέχεται ένα περιβάλλον μοντελοποίησης συστημάτων, βασισμένο στη γλώσσα μοντελοποίησης SysML και μέσα από το σχεδιαστικό περιβάλλον θα εξάγουμε το μοντέλο και στη συνέχεια με τους κατάλληλους μετασχηματισμούς θα δημιουργηθεί ένα αντίστοιχο μοντέλο προσομοίωσης. Με την εκτέλεση της προσομοίωσης θα είμαστε σε θέση να εξάγουμε χρήσιμα αποτελέσματα και σημαντικά συμπεράσματα, στοχεύοντας στην άρτια και σωστή μελέτη της απόδοσης του συστήματος μας.

Στόχος της εργασίας είναι η εξοικείωση με την χρήση της SySML, της μοντελοποίησης, των διαδικασιών αυτόματης παραγωγής κώδικα και της προσομοίωσης. Καθοριστικό ρόλο στην επίτευξη του στόχου μας έπαιξαν τα εργαλεία που χρησιμοποιήσαμε, όπου μεταξύ άλλων είναι το Modelio, Eclipse και DEVS Suite.

Λέξεις Κλειδιά: Μοντελοποίηση, SysML, QVT, προσομοίωση, DEVS.

Abstract

The physical systems that we daily encounter and sometimes facilitate us and sometimes not, can now be analyzed and designed through systems modeling, so that there is a clearer picture of their use. In additional, through simulation, we can prevent a system malfunction but also reduce any failures that may occur. Another important element of the simulation, is that it provide us with a more complete picture of the operation of the system.

In the present context, we will follow a general way of modeling SysML systems, with no extension for a specific scope or simulation environment and can be be used in any system model. To implement this model, we will use basic SysML diagrams and basic properties provided to us by the language.

In the master thesis, we use tools with which, a system modeling environment is provided, based on the SysML modeling language and through the design environment we will export the model and then with the appropriate transformations a relevant simulation model will be created. By performing the simulation we will be able to draw useful results and important conclusions,, aiming at a thorough and correct study of our system performance.

The aim of the context is the familiarity with the use of SysML, the modeling, the automated code generation procedures and the simulation. A key role in achieving our goal, played the tools that we used, which among others are Modelio, Eclipse and DEVS Suite.

KEYWORDS: Modeling, SysML, QVT, simulation, DEVS.

Ευρετήριο Εικόνων

Ευρετήριο Πινάκων

Κεφάλαιο 1ο

1.1 Εισαγωγή

Στο κεφάλαιο αυτό δίνεται η περιγραφή του προβλήματος που εξετάζεται στην παρούσα διπλωματική εργασία, το αντικείμενο της μελέτης, παρουσιάζεται η δομή της και η βιβλιογραφική έρευνα του προβλήματος.

1.2 Περιγραφή του προβλήματος

Η SysML προτυποποιήθηκε και προτείνεται από τον οργανισμό OMG για την υποστήριξη της Βασισμένης σε Μοντέλα Ανάπτυξης Συστημάτων (Model-Based Systems Engineering - MBSE). Ερευνητικές προσπάθειες έχουν πετύχει την αυτοματοποίηση της παραγωγής εκτελέσιμων μοντέλων προσομοίωσης είτε αξιοποιώντας βιβλιοθήκες προσομοίωσης για συγκεκριμένο πεδίο, είτε επεκτείνοντας τη SysML για συγκεκριμένο περιβάλλον προσομοίωσης. Για τη δυνατότητα απρόσκοπτης προσομοίωσης οποιουδήποτε μοντέλου SysML, θα πρέπει να γίνει διερεύνηση και σχετική υλοποίηση της αυτοματοποίησης της παραγωγής εκτελέσιμων μοντέλων προσομοίωσης (διακριτού ή/και συνεχούς χρόνου) από γενικά μοντέλα συστήματος SysML (χωρίς επεκτάσεις για συγκεκριμένο πεδίο ή περιβάλλον προσομοίωσης).

1.3 Αντικείμενο της μελέτης

Οι βασικοί στόχοι της εργασίας αυτής συνοψίζονται στα παρακάτω σημεία:

- Κατανόηση της γλώσσας μοντελοποίησης SysML.
- Μελέτη της υπάρχουσας βιβλιογραφίας στα γενικά μοντέλα συστήματος SysML χωρίς επεκτάσεις για συγκεκριμένο πεδίο ή περιβάλλον προσομοίωσης.
- Δημιουργία ενός γενικού τρόπου μοντελοποίησης συστημάτων με SySML.
- Οι ελάχιστες παράμετροι προσομοίωσης που πρέπει να διατίθενται ή να συμπληρωθούν για να τρέξει η προσομοίωση.
- Τα αναμενόμενα αποτελέσματα της προσομοίωσης.
- Υλοποίηση της αυτοματοποίησης της παραγωγής εκτελέσιμων μοντέλων προσομοίωσης (διακριτού ή/και συνεχούς χρόνου) από γενικά μοντέλα συστήματος SysML.
- Εξαγωγή συμπερασμάτων για την μελέτη που έχει διενεργηθεί και παρουσίαση προτάσεων για μελλοντική έρευνα.

1.4 Βιβλιογραφική Έρευνα

Αναζητώντας στη διεθνή βιβλιογραφία, δε βρέθηκε κάποια σχετική έρευνα όπου να μελετά τη διερεύνηση και σχετική υλοποίηση της αυτοματοποίησης της παραγωγής εκτελέσιμων μοντέλων προσομοίωσης (διακριτού ή/και συνεχούς χρόνου) από γενικά μοντέλα συστήματος SysML. Αντιθέτως υπήρχε αρκετή βιβλιογραφική ανασκόπηση σχετικά με τη μελέτη συστημάτων που περιέχουν επεκτάσεις για διάφορα συγκεκριμένα πεδία ή περιβάλλοντα προσομοίωσης.**[1][2][3]**

Επιπλέον, υπάρχουν αρκετά εμπορικά εργαλεία μοντελοποίησης SysML/UML που προσφέρουν δυνατότητες προσομοίωσης επιμέρους διαγραμμάτων (activity diagrams, state machine diagrams, parametric diagrams), χωρίς όμως να έχουν τη λογική της διασύνδεσης των διαγραμμάτων και της δημιουργίας ενός συνολικού μοντέλου.

Κεφάλαιο 2: Μοντελοποίηση Συστημάτων και Προσομοίωση

2.1 Μοντέλα Συστημάτων

Μοντέλο είναι μία αναπαράσταση ενός φυσικού συστήματος ή ενός οργανισμού. Για να γίνει σωστή εξαγωγή συμπερασμάτων από το μοντέλο και κατ' επέκταση από το σύστημα, θα πρέπει το μοντέλο να αναπαριστά το σύστημα όσο το δυνατόν πιο πιστά. Έτσι, για να περιγραφεί ένα φυσικό σύστημα από ένα μοντέλο, θα πρέπει να αποτυπωθούν στις οργανωτικές δομές (πακέτα) και στα υποσυστήματα του μοντέλου όλα τα στοιχεία του φυσικού συστήματος. Για τη μελέτη ενός συστήματος είναι απαραίτητη η δημιουργία ενός αντιπροσωπευτικού μοντέλου.
Τα μοντέλα λοιπόν χρησιμοποιούνται :
• για να μελετηθούν υπάρχοντα συστήματα χωρίς να επηρεάζεται η λειτουργία τους.
• για να μελετηθούν υπάρχοντα συστήματα χωρίς να καταστραφούν.
• για να μελετηθούν συστήματα που δεν υπάρχουν αλλά θέλουμε να κατασκευάσουμε ή να προσαρμώσουμε.

Με άλλα λόγια, ένα μοντέλο περιγράφει σε δομημένη μορφή το σύστημα, επιτυγχάνοντας την οπτικοποίηση, τον καθορισμό της δομής και της συμπεριφοράς του συστήματος, καθώς και την δημιουργία ενός προτύπου που συμβάλλει στην καθοδήγηση της κατασκευής του συστήματος. Έτσι, η μοντελοποίηση περιγράφει ένα σύστημα διατηρώντας μόνο ένα περιορισμένο αριθμό από τις αρχικές λεπτομέρειες του συστήματος, κάτι που βοηθά τον αναλυτή να κατανοήσει τις λειτουργικές δυνατότητες του συστήματος για τις πτυχές λειτουργίας του συστήματος που κρίνονται ως σημαντικές.

2.2 Εισαγωγή στη UML

Η Ενοποιημένη Γλώσσα Μοντελοποίησης (Unified Modeling Language, UML) είναι μια διαγραμματική γλώσσα για την οπτική αναπαράσταση, τη διαμόρφωση προδιαγραφών και την τεκμηρίωση συστημάτων, κυρίως συστημάτων λογισμικού. Συγκεκριμένα, ορίζεται ως μία γλώσσα μοντελοποίησης που εφαρμόζεται στην αναπαράσταση ενός φυσικού συστήματος με οπτικό τρόπο.

Η UML είναι ένα αποδοτικό πρότυπο για την αντικειμενοστραφή μοντελοποίηση. Ένα αντικειμενοστραφές σύστημα μοντελοποιείται ως μία συλλογή αντικειμένων που αλληλεπιδρούν για την εκτέλεση μίας λειτουργίας η οποία είναι τελικά αξιοποιήσιμη από τον χρήστη του συστήματος.

Η UML είναι ορισμένη σύμφωνα με την Υποδομή Μετα-Αντικειμένων (Meta-Object Facility) (MOF) ως μετα-μοντέλο.**[4]**

Τα τρία βασικά στοιχεία της UML είναι οι οντότητες, οι σχέσεις και τα διαγράμματα.
Η UML αποτυπώνει τη στατική δομή, η οποία καθορίζει τα είδη των αντικειμένων και τις συσχετίσεις μεταξύ τους, καθώς και τη δυναμική συμπεριφορά, όπου προσδιορίζει την εξέλιξη των αντικειμένων σε σχέση με τον χρόνο και την επικοινωνία μεταξύ τους.

Συγκεντρωτικά τα διαγράμματα που περιγράφουν τη στατική δομή της UML είναι:

- Διαγράμματα δομής
    - Διάγραμμα κλάσεων (*class diagram*)
    - Διάγραμμα αντικειμένων (*object diagram*)
- Διαγράμματα δομής υλοποίησης
    - Διάγραμμα εξαρτημάτων (*component diagram*)
    - Διάγραμμα ανάπτυξης (*deployment diagram*)

Ενώ τα διαγράμματα που περιγράφουν τη δυναμική συμπεριφορά της UML είναι:

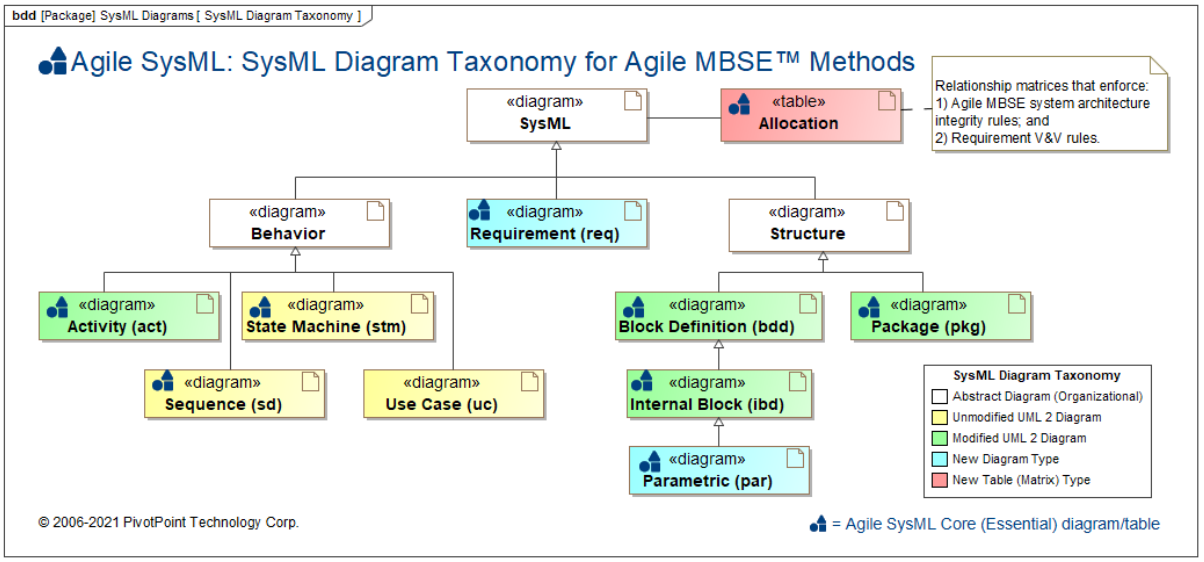
- Διάγραμμα περιπτώσεων χρήσης (*use case diagram*)
- Διαγράμματα συμπεριφοράς
    - Διάγραμμα καταστάσεων (*statechart diagram*)
    - Διάγραμμα δραστηριοτήτων (*activity diagram*)
    - Διαγράμματα αλληλεπίδρασης
        - Διάγραμμα ακολουθίας (*sequence diagram*)
        - Διάγραμμα συνεργασίας (*collaboration diagram*)

2.3 Εισαγωγή στη SySML

Η Γλώσσα Μοντελοποίησης Συστημάτων (Systems Modeling Language, SysML) είναι μια γλώσσα μοντελοποίησης γενικής χρήσης για εφαρμογές μηχανικής συστημάτων. Υποστηρίζει την προδιαγραφή, την ανάλυση, το σχεδιασμό, την επαλήθευση και την επικύρωση ενός ευρέος φάσματος συστημάτων και συστημάτων-συστημάτων (systems of systems).

Η SysML ορίζεται ως η προσαρμογή και επέκταση της (UML) στον τομέα της μηχανικής συστημάτων (systems engineering).

Τα διαγράμματα της SysML χωρίζονται σε τρεις κατηγορίες: τα διαγράμματα δομής (structure diagrams), συμπεριφοράς (behaviour diagrams) και απαιτήσεων (requirement diagrams).



Εικόνα 2.1: SysML διαγράμματα και η σύνδεση μεταξύ τους
Πηγή εικόνας [2.1]

Η SysML ορίζει τα παρακάτω διαγράμματα , όπως παρουσιάζονται και στην Εικόνα 2.1
Τα διαγράμματα δομής περιλαμβάνουν:

• Το διάγραμμα καθορισμού τμημάτων (block definition diagram).
• Το διάγραμμα εσωτερικών τμημάτων (internal block diagram).
• Το διάγραμμα παραμέτρων (parametric diagram).
• Το διάγραμμα πακέτων (package diagram).

Τα διαγράμματα συμπεριφοράς περιλαμβάνουν:
• Τα διαγράμματα δραστηριοτήτων (activity diagram).
• Τα διαγράμματα ακολουθίας (sequence diagram).
• Τα διαγράμματα κατάστασης (state machine diagram).
• Το διάγραμμα περιπτώσεων χρήσης (use case diagram).

Τα Διαγράμματα Απαιτήσεων (Requirement diagrams) όπου παρουσιάζουν τις απαιτήσεις του έργου και τη σχέση τους με άλλες απαιτήσεις, στοιχεία του σχεδιασμού και περιπτώσεις ελέγχου, οι οποίες υποστηρίζουν την ανίχνευση απαιτήσεων. Απαίτηση είναι η ιδιότητα ή συμπεριφορά ενός συστήματος, η οποία θα πρέπει πάντα να ικανοποιείται.**[5]**
Όλα τα παραπάνω διαγράμματα φαίνονται στην **(Εικ. 2.1).**

2.4 Εργαλείο Μοντελοποίησης (Modelio)

Για να αναπτύξουμε το μοντέλο συστήματος στην παρούσα μελέτη, χρησιμοποιήσαμε το Modelio το οποίο είναι ένα εργαλείο UML ανοιχτού κώδικα που αναπτύχθηκε από τη Modeliosoft.

Το Modelio είναι πρώτα απ'όλα ένα περιβάλλον μοντελοποίησης. Είναι μια λύση μοντελοποίησης που προσφέρει ένα ευρύ φάσμα λειτουργιών με βάση τα κύρια πρότυπα της αρχιτεκτονικής των επιχειρήσεων, της ανάπτυξης λογισμικού και της μηχανικής συστημάτων. Είναι σχεδιασμένο για προγραμματιστές, αναλυτές και αρχιτέκτονες λογισμικού και συστημάτων.

Το Modelio υποστηρίζει όλα τα διαγράμματα της UML, της BPMN και της SySML όπου θα χρησιμοποιήσουμε για την δημιουργία του μοντέλου μας. Επιπλέον, υποστηρίζει όλα τα διαγράμματα SysML, από διαγράμματα μπλοκ έως εσωτερικά και παραμετρικά διαγράμματα.

Τέλος, η Modelio SA SysML υποστηρίζει την αρχιτεκτονική του συστήματος και τη μοντελοποίηση λειτουργικών προδιαγραφών χρησιμοποιώντας την SysML.**[6]**

2.5 Προσομοίωση

Η προσομοίωση είναι μια μέθοδος όπου μελετάται ένα φυσικό σύστημα με τη βοήθεια του υπολογιστή. Η προσομοίωση αναπαριστά τη λειτουργία του συστήματος για κάποιο χρονικό διάστημα. Με την προσομοίωση δεν υλοποιείται το πραγματικό σύστημα αλλά η αποτύπωση των χαρακτηριστικών ενός συστήματος στον υπολογιστή, όπου θα ληφθούν οι βέλτιστες αποφάσεις για την υλοποίηση του πραγματικού συστήματος μέσω των πειραμάτων που θα πραγματοποιηθούν. Η εφαρμογή της προσομοίωσης μπορεί να γίνει σε διάφορα είδη προβλημάτων. Η επιλογή ή όχι της εφαρμογής της προσομοίωσης σε ένα πρόβλημα εξαρτάται από τα χαρακτηριστικά του συστήματος που πρόκειται να μελετηθεί αλλά και το είδος του προβλήματος. Η προσομοίωση χρησιμοποιείται για να μελετηθεί η συμπεριφορά ενός συστήματος, για τον έλεγχο θεωριών και υποθέσεων της παρατηρούμενης συμπεριφοράς του συστήματος, για την βελτιστοποίηση απόδοσης αλλά και για την πρόβλεψη της μελλοντικής του συμπεριφοράς **[7]**.

Συνοψίζοντας, η προσομοίωση εξυπηρετεί συγκεκριμένους σκοπούς:
• **Αξιολόγηση:** Μέσω της προσομοίωσης αξιολογείται το υπό μελέτη σύστημα.
• **Σύγκριση:** Συγκρίνονται εναλλακτικοί τρόποι σχεδίασης και λειτουργίας του υπό μελέτη συστήματος.
• **Πρόβλεψη:** Εκτιμάται η μελλοντική απόδοση του συστήματος κάτω από συγκεκριμένες συνθήκες λειτουργίας και παραδοχές για το περιβάλλον του συστήματος.
• **Ανάλυση ευαισθησίας:** Με την ανάλυση ευαισθησίας (sensitivity analysis) καθορίζονται εκείνοι οι παράγοντες που επηρεάζουν περισσότερο τη λειτουργία του συστήματος.

Έτσι λοιπόν, τα παραπάνω εφαρμόζονται σε διάφορους τομείς της καθημερινότητας μεταξύ των οποίων:

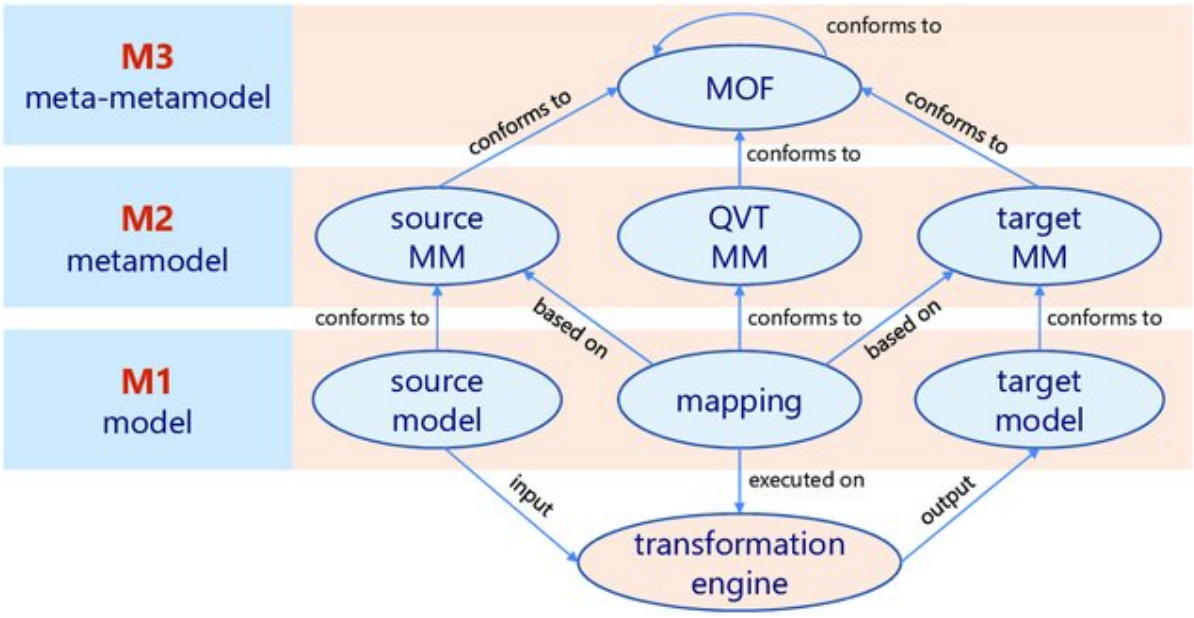
• στη βιομηχανία, για την ανάλυση και σχεδίαση συστημάτων παραγωγής
• στις εμπορικές επιχειρήσεις, για τον έλεγχο των αποθεμάτων
• στη μελέτη κυκλοφοριακών συστημάτων
• στις τράπεζες και στα νοσοκομεία, για τη μελέτη συστημάτων εξυπηρέτησης πελατών.

2.6 Γλώσσα QVT

Η QVT (Query / View / Transformation) χρησιμοποιείται ως ένα πρότυπο για μετασχηματισμούς μοντέλων και περιλαμβάνει τρεις γλώσσες μετασχηματισμού μοντέλου. Ορίζεται από την Object Management Group (OMG) **[8]** και καθορίζει έναν τρόπο για να μετατρέπει ένα μοντέλο-πηγή (Source model) σε ένα μοντέλο-στόχο (Target model).

Τόσο τα μοντέλα πηγής, όσο και τα μοντέλα προορισμού συμμορφώνονται με τα μεταμοντέλα που έχουν οριστεί σε όρους Meta-Object Facility (MOF) 2.0**[9]**. Το QVT είναι ένα κατάλληλο πρότυπο για τον ορισμό ενός τέτοιου μετασχηματισμού, καθώς ο κύριος σκοπός του είναι να καθορίσει την αντιστοιχία και τον μετασχηματισμό μεταξύ δύο μετα-μοντέλων. Το πρότυπο QVT αξιοποιεί το πρότυπο OCL 2.0 και το επεκτείνει με λειτουργίες συμπερασμού (imperative features).

Η εικόνα 2.2 αποτυπώνει την σχέση των μοντέλων, των μεταμοντέλων και των μετασχηματισμών με το MOF.



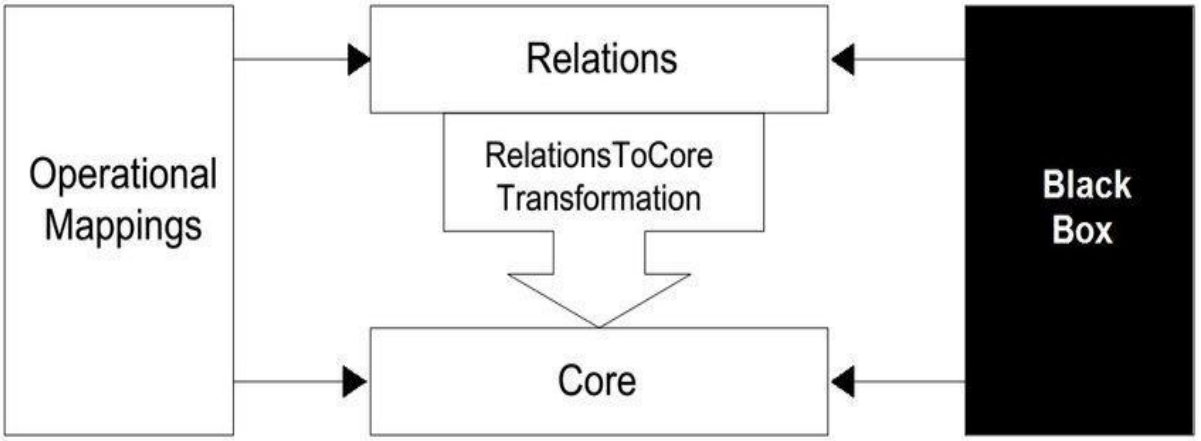
Εικόνα 2.1: Concept of QVT model-to-model transformations

Πηγή εικόνας [2.2]

Το πρότυπο της QVT περιλαμβάνει τρεις γλώσσες, την QVT-Relations, την QVT-Core και την QVT-Operational. Αναλυτικότερα έχουμε:

- Την QVT-Relations: μια δηλωτική γλώσσα που έχει σχεδιαστεί για να επιτρέπει τη σύνταξη μονόδρομων και αμφίδρομων μετασχηματισμών μοντέλων. Ένας μετασχηματισμός ενσωματώνει μια σχέση συνέπειας σε σύνολα μοντέλων. Η συνεκτικότητα μπορεί να ελεγχθεί εκτελώντας τον μετασχηματισμό σε λειτουργία checkonly. Η γλώσσα QVT-Relations έχει συντακτικό κειμένου και γραφικής αναπαράστασης και καθορίζει relations (σχέσεις) πάνω από στοιχεία του μοντέλου.
- Την QVT-Core: μια απλή δηλωτική γλώσσα, που λειτουργεί ως μια απλούστευση της γλώσσας QVT-Relations. Ωστόσο, η QVT-Core δεν είχε ποτέ πλήρη εφαρμογή και στην πραγματικότητα δεν είναι τόσο εκφραστική όσο το QVT-Relations.
- Την QVT-Operational: μια επιτακτική γλώσσα σχεδιασμένη για τη σύνταξη μονοκατευθυντικών μετασχηματισμών και επεκτείνει τις γλώσσες QVT-Relations και QVT-Core **[10].** Στην παρούσα εργασία αξιοποιήθηκε αυτή η παραλλαγή της γλώσσας QVT.

Ακόμα, έχουμε τον μηχανισμό QVT-BlackBox όπου χρησιμοποιείται για την κωδικοποίηση πολύπλοκων αλγορίθμων σε οποιαδήποτε γλώσσα προγραμματισμού και λειτουργεί ως μία εξωτερική γλώσσα προσφέροντας κάποιες υλοποιήσεις. Ο μηχανισμός QVT-BlackBox είναι σημαντικός για την υλοποίηση αλγορίθμων, διότι η QVT στηρίζεται στη γλώσσα OCL, η οποία δεν είναι γλώσσα προγραμματισμού και συνεπώς ορισμένοι αλγόριθμοι δεν μπορούν να υλοποιηθούν με την OCL.**[11]**



Εικόνα 2.1: Αρχιτεκτονική QVT
Πηγή εικόνας [2.3]

2.7 Εργαλείο εκτέλεσης προσομοίωσης

Η υλοποίηση του κώδικα γραμμένου σε γλώσσα QVT-O πραγματοποιήθηκε με τη χρήση του εργαλείου Eclipse IDE 2021-03**[12]** με ενσωματωμένα τα UML Tools και Eclipse Suite**[13]**.

Το Eclipse είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) που χρησιμοποιείται στον προγραμματισμό υπολογιστών. Στη δική μας περίπτωση το Eclipse QVT είναι μια υλοποίηση της λειτουργικής γλώσσας μετασχηματισμού που ορίζεται από το Meta Object Facility (MOF). Μακροπρόθεσμα, στοχεύει στην πλήρη εφαρμογή του λειτουργικού μέρους του προτύπου. Μια υψηλού επιπέδου επισκόπηση της λειτουργικής γλώσσας QVT διατίθεται με τη μορφή παρουσιάσεων Model Transformation with Operational QVT και The Art of Model Transformation with Operational QVT.

Κεφάλαιο 3: Προϋποθέσεις και αναμενόμενα αποτελέσματα προσομοίωσης

3.1 Ελάχιστες προϋποθέσεις μοντέλων SysML για να μπορούν να προσομοιωθούν

Στη μελέτη που κάνουμε, για να μπορούμε να προσομοιώσουμε ένα μοντέλο SySML θα πρέπει να υπάρχουν κάποιες βασικές προϋποθέσεις, ώστε να μπορεί να προσομοιωθεί το μοντέλο μας. Οι βασικές προϋποθέσεις που έχουμε ορίσει είναι ότι τα μοντέλα SysML θα πρέπει να περιέχουν τα εξής:

- Use Case Diagram (UCD)
- Block Definition Diagram (BDD)
- State Machine Diagrams (SMD)
- Internal Block Diagrams (IBD)
- Parametric Diagram (PD)
- Constraints στα UCDs και στα BDDs
- Information Flows (IF)

Αρχικά λοιπόν, το μοντέλο μας θα περιέχει ένα Use Case Diagram(UCD) για την περιγραφή της σχέσης ανάμεσα στον χρήστη και στο σύστημα και συγκεκριμένων περιπτώσεων χρήσης. Ο σκοπός ενός διαγράμματος περίπτωσης χρήσης είναι να καταγράψει τη δυναμική πλευρά του συστήματος.

Ένα ακόμα σημαντικό κομμάτι είναι τα BDDs. Τα Block Definition Diagrams ορίζουν μια συλλογή χαρακτηριστικών που χρησιμοποιούνται για να περιγράψουν ένα σύστημα, ένα υποσύστημα, ένα στοιχείο ή άλλο αντικείμενο μηχανικού ενδιαφέροντος. Στη μελέτη μας τα BDDs περιέχουν τα blocks που απαρτίζουν το σύστημα.

Με τα State Machine Diagrams (SMDs), που ορίζουμε για κάποια από τα blocks, έχουμε τη δυνατότητα να καθορίζουμε την κατάσταση στην οποία μπορεί να βρίσκεται κάθε στιγμή μια οντότητα στο μοντέλο μας. Τα SMDs αφορούν συγκεκριμένα blocks και μπορούν να συνδέονται με συγκεκριμένα use cases του Use Case Diagram.

Τα Internal Block Diagrams (IBDs) είναι εκείνα που καταγράφουν την εσωτερική δομή ενός στοιχείου μπλοκ, όσον αφορά τις ιδιότητές του (θύρες και μέρη) και τις συνδέσεις μεταξύ αυτών των ιδιοτήτων. Στην δική μας περίπτωση, χρησιμοποιούμε τα IBD για να μεταφέρουμε διάφορες σημαντικές τιμές (δύναμη, ταχύτητα) από ένα υποσύστημα σε ένα άλλο. Αυτή η δυνατότητα θα καταστήσει εφικτό το να προσομοιώσουμε επιμέρους τμήματα του συστήματός μας.

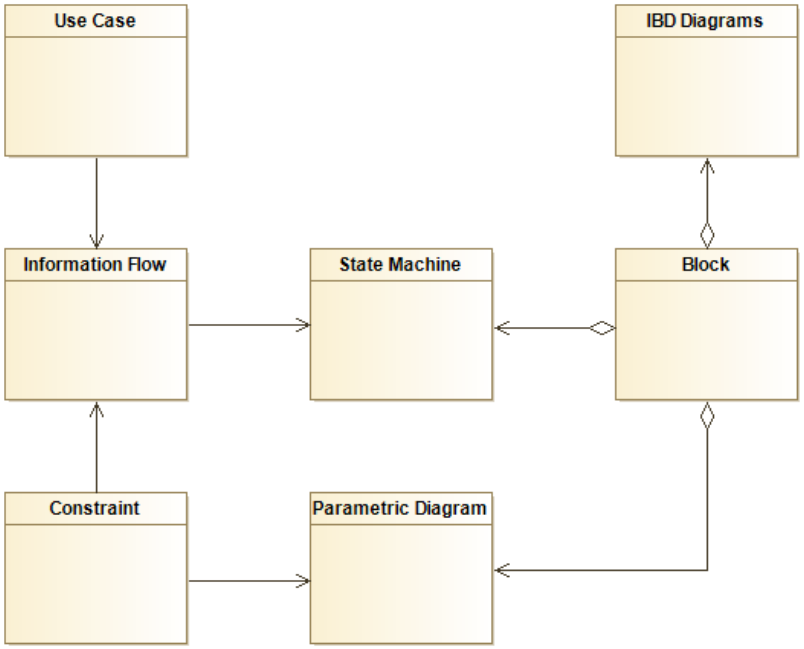
Τα Parametric Diagrams χρησιμοποιούνται για τη δημιουργία συστημάτων εξισώσεων που μπορούν να περιορίσουν τις ιδιότητες των μπλοκ. Το Parametric Diagram θα μας βοηθήσει στην προσομοίωση, διότι αξιοποιεί τον μαθηματικό τύπο που έχει οριστεί μέσα στο Constraint, αξιοποιώντας έτσι τις τιμές που λαμβάνουμε από το IBD και γενικότερα τις πληροφορίες που προκύπτουν απο τη διασύνδεση του Use Case με τα State Machine.

Το Constraint Block ορίζει έναν μαθηματικό κανόνα (περιορισμός) και παραμέτρους κανόνα, όπου οι τελευταίες δεσμεύονται σε Block Value Properties, ώστε οι αλλαγές σε μία ιδιότητα τιμής μπλοκ να μεταδοθούν σε άλλες Block Value Properties κατά τρόπο συνεπή με τον μαθηματικό κανόνα.

Όπως αναφέραμε και παραπάνω, μέσα στο UCD περιέχονται οι περιπτώσεις χρήσης. Επίσης ενσωματώνονται και State Machine Diagrams. Η διασύνδεση των Use Cases με τα State Machine Diagrams των blocks, είναι σημαντικός παράγοντας για τη σύνθεση του μοντέλου της προσομοίωσης. Από τη σύνδεση αυτή, μπορούν να προκύψουν οι δυνατές καταστάσεις από τις οποίες μπορούν να περνούν τα συστατικά στοιχεία του συστήματος.

Τέλος, για τη διασύνδεση των Use Case με τα SMD, τη διασύνδεση μεταξύ των IBD αλλά και του Parametric Diagram, σημαντική είναι η λειτουργία των Information Flows μέσω των οποίων μεταφέρεται όλη η απαραίτητη πληροφορία για την εκτέλεση της προσομοίωσης.

Οι δομές της SysML, που αξιοποιούνται και περιγράφηκαν παραπάνω αποτυπώνονται στο μετα-μοντέλο του σχήματος 3.1.



Εικόνα 3.1: Δομές SysML

3.2 Γλώσσα QVT που επιλέχθηκε

Όπως αναφέρθηκε και παραπάνω η QVT χρησιμοποιείται για το μετασχηματισμό μοντέλων. Η αρχική μας κατεύθυνση ήταν να μετασχηματίσουμε το μοντέλο μας με την δηλωτική γλώσσα QVT-Relations, η οποία παρέχει υψηλού επιπέδου δυνατότητες ενδοσκόπησης των μοντέλων του μετασχηματισμού. Με τους μετασχηματισμούς της QVT-Relations οι σχέσεις εφαρμόζονται και στα δύο μοντέλα, το οποίο σημαίνει ότι από το ένα μέρος διαβάζουμε τη δομή της πηγής και από το άλλο μέρος δημιουργούμε τη δομή του στόχου.

Έτσι, ξεκινήσαμε από το επίπεδο των μεταμοντέλων UML και DEVS. Έπειτα, προσπαθώντας να περάσουμε στο δεύτερο επίπεδο συσχετισμού της κλάσης με το DEVS μοντέλο μας, εντοπίστηκε ότι δημιουργούνταν αρκετές φορές το μοντέλο DEVS χωρίς να ενσωματώνει το ένα επίπεδο μέσα στο άλλο που ήταν και ο στόχος μας. Έτσι, παρατηρήθηκαν περιορισμοί του εργαλείου Eclipse (QVT-Declarative) στην υλοποίηση του μετασχηματισμού με δηλωτικό τρόπο (αντίστοιχο του QVT-R) κατά τη διαδικασία συγγραφής και ελέγχου του κώδικα. Το παλαιό λογισμικό Medini-QVT, που υποστηρίζει QVT-R δεν είναι πλέον λειτουργικό (σε υπολογιστές με σύγχρονες εκδόσεις λειτουργικών συστημάτων). Επιπλέον, η QVT-Relations είναι πιο χρήσιμη-απαραίτητη για αμφίδρομους μετασχηματισμούς, κάτι το οποίο δεν ήταν απαραίτητο για τον δικό μας μετασχηματισμό. Επίσης, η πρόσβαση στο υλικό και στις πληροφορίες ως προς την ανάπτυξη του κώδικα της QVT-Operational**[14][15]** ήταν ευκολότερη σε σχέση με την QVT-Relations, ενώ και η υποστήριξη της QVT-Operational γίνεται από πιο σύγχρονα εργαλεία λογισμικού.

Επομένως, για τους παραπάνω λόγους αποφασίσαμε να στραφούμε στη λύση της QVT-Operational. Όπως γνωρίζουμε, η QVT-Operational είναι πιο διαδικαστική σε σχέση με την QVT Relation, ενώ η σύνταξη της αλλάζει σε μικρό βαθμό χωρίς επί της ουσίας να υπάρχουν πολλές διαφορές με την QVT-Relations. Κατά συνέπεια, μετά την διερεύνηση και των δύο γλωσσών, η QVT Operational ήταν εκείνη που επιλέχθηκε για την υλοποίηση του μετασχηματισμού των μοντέλων μας.

3.3 Συστήματα συνεχούς ή διακριτού χρόνου

Τα συστήματα χωρίζονται σε κατηγορίες ανάλογα με τις μεταβολές της κατάστασης τους ή τη σχέση τους με το περιβάλλον. Συγκεκριμένα, όσον αφορά τις μεταβολές της κατάστασης, διακρίνονται σε συστήματα διακριτού ή συνεχούς χρόνου.

Στα συστήματα συνεχούς χρόνου, βλέπουμε ότι οι διαφορικές εξισώσεις με τις οποίες εκφράζεται η συμπεριφορά τους, η κατάσταση του μοντέλου μεταβάλλεται με συνεχή τρόπο κατά την πάροδο του χρόνου. Αντιθέτως, στα συστήματα διακριτού χρόνου, η κατάσταση του μοντέλου μεταβάλλεται στιγμιαία σε διακριτές χρονικές στιγμές, μέσω υπολογισμών που λαμβάνουν υπόψη την τρέχουσα κατάσταση, το είδος του συμβάντος και το χρόνο που έχει παρέλθει.
Στην παρούσα εργασία, επιλέξαμε να ασχοληθούμε με το κομμάτι αυτό των μοντέλων SysML που θα μπορούσαν να προσομοιωθούν με προσομοίωση διακριτού χρόνου.**[16]**

3.4 Είδος προσομοίωσης που ακολουθήθηκε

Για να ελέγξουμε ότι το μοντέλο μας αντιπροσωπεύει το φυσικό σύστημα απόλυτα και πληρεί όλες τις προϋποθέσεις, θα πρέπει να μελετήσουμε το μοντέλο μας και να αξιολογήσουμε τις επιδόσεις του. Αυτή η διαδικασία προσφέρεται με την προσομοίωση. Σύμφωνα με το **[17]**, η μεθοδολογία προσομοίωσης διακριτού χρόνου DEVS είναι κατάλληλη για την προσομοίωση μοντέλων SysML. Έτσι, μέσα από το DEVS μας δίνεται η δυνατότητα να δημιουργήσουμε ένα αντίστοιχο περιβάλλον, το οποίο θα μας προσφέρει μεγάλη αξιοπιστία και ακριβή αποτελέσματα. Αυτό συμβαίνει διότι το DEVS είναι ένας ιεραρχικός φορμαλισμός, κάτι το οποίο σημαίνει ότι το πρότυπο μηχανής προσομοίωσης που δημιουργείται έχει ως στόχο την εκτέλεση των μοντέλων και την αναπαραγωγή της δυναμικής συμπεριφοράς τους.

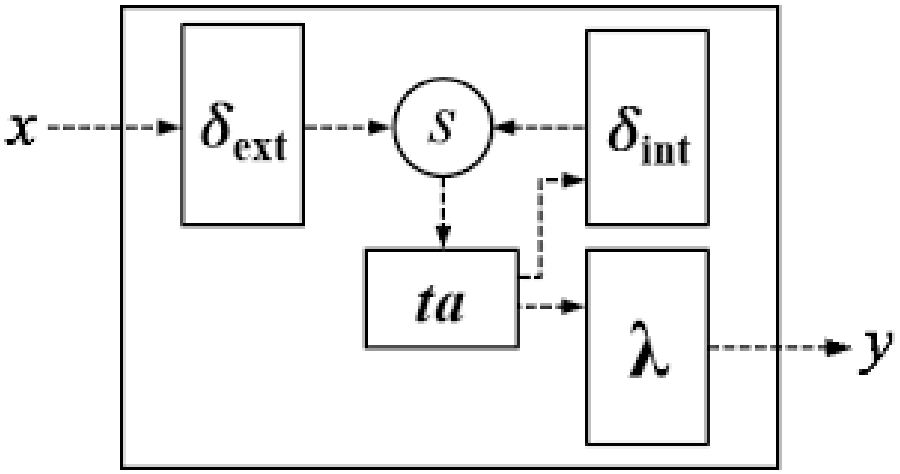
Επιπλέον, κοιτάζοντας τη δομή των μοντέλων SysML, με την οποία θα ασχοληθούμε στην παρούσα εργασία, και τη δομή των μοντέλων DEVS παρατηρούμε ότι παρουσιάζουν πολλές ομοιότητες ως προς τη δομή τους. Η SysML χρησιμοποιεί τα Blocks που περιλαμβάνουν ιδιότητες, Ports και διασυνδέσεις που μπορούν να υπάρξουν ανάμεσα τους.

Το DEVS για να περιγράψει τη δομή και τη συμπεριφορά των μοντέλων χρησιμοποιεί κάποια μαθηματικά σύνολα και ορίζει δύο είδη μοντέλων, τα coupled models που περιγράφουν τη διασύνδεση των atomic και των coupled μοντέλων, και τα atomic models που ασχολούνται με τη συμπεριφορά των μοντέλων.

Ένα atomic DEVS μοντέλο παραμένει σε μια κατάσταση για συγκεκριμένο χρονικό διάστημα και παράγει μια έξοδο πριν μεταβεί στην επόμενη κατάσταση. Το atomic DEVS μοντέλο περιγράφεται διαγραμματικά στην **Εικόνα 3.2** και περιέχει τα παρακάτω επτά στοιχεία:

M =< X, Y, S, ta, δext, δint, λ >

- Το X είναι ένα σύνολο συμβάντων εισόδου.
- Το Y είναι ένα σύνολο συμβάντων εξόδου.
- Το S είναι ένα σύνολο καταστάσεων.
- t: S → T∞ είναι η συνάρτηση χρονικής προώθησης (time advance) όπου T∞ είναι το σύνολο μη αρνητικών λογικών αριθμών έως το άπειρο. Αυτή η λειτουργία χρησιμοποιείται για τον προσδιορισμό της διάρκειας ζωής μιας κατάστασης.
- δext : Q × X → S είναι η συνάρτηση μετάβασης εξωτερικής κατάστασης που ορίζει πώς ένα συμβάν εισαγωγής αλλάζει κατάσταση.
- δint : S → S είναι η συνάρτηση μετάβασης εσωτερικής κατάστασης, που προσδιορίζει την επόμενη κατάσταση που θα μεταβεί το σύστημα, όταν έχει τελειώσει ο χρόνος παραμονής στην τρέχουσα κατάσταση
- λ : S → Yφ είναι η συνάρτηση εξόδου, όπου Yφ = Y ∪ {φ} και φ / ∈ Y δηλώνει το αθόρυβο συμβάν. Αυτη η έξοδος και η συνάρτηση μετάβασης εσωτερικής κατάστασης καθορίζει τον τρόπο κατάστασης **[18]**

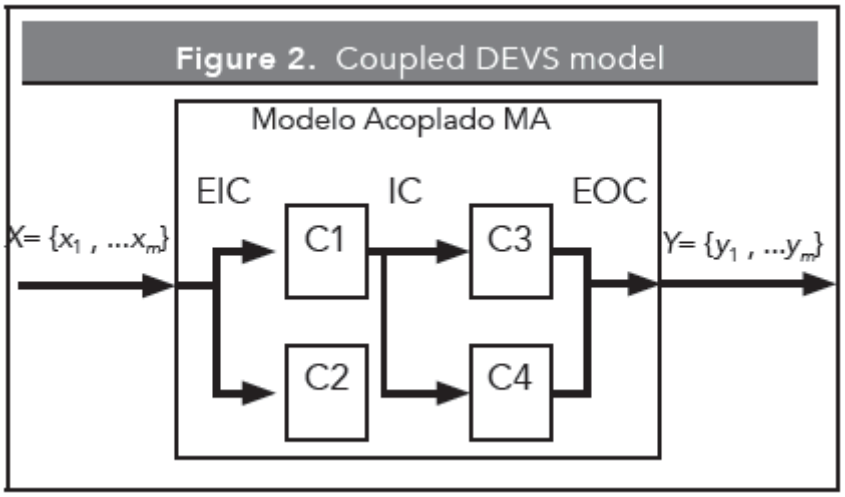


Εικόνα 3.2: Atomic DEVS Model
Πηγή εικόνας [3.2]

Από την άλλη, ένα coupled DEVS μοντέλο μπορεί να περιέχει κι άλλα DEVS μοντέλα, τα οποία μπορούν να είναι είτε coupled είτε atomic, διασυνδεδεμένα μεταξύ τους μέσω των εισόδων και των εξόδων των μοντέλων. Το coupled DEVS μοντέλο περιγράφεται διαγραμματικά από την **Εικόνα 3.3** και περιέχει τα οκτώ στοιχεία:

N =< X, Y, D, {Mi}, Cxx, Cyx, Cyy, Select >
όπου:

- Το X είναι ένα σύνολο συμβάντων εισαγωγής.
- Το Y είναι ένα σύνολο συμβάντων εξόδου.
- Το D είναι ένα σύνολο ονομάτων των υποστοιχείων
- Το {Mi} είναι ένα σύνολο των περιεχομένων στοιχείων, όπου $\forall i \in D$, Mi είναι ένα atomic ή ένα coupled DEVS μοντέλο.
- $Cxx \subseteq X \cup i \in D$ είναι μια λειτουργία σύνδεσης εξωτερικής εισόδου, όπου το Xi είναι το σύνολο συμβάντων εισαγωγής του υπό στοιχείου $i \in D$.
- $Cyx \cup i \in DYi \times \cup i \in D\ Xi$ είναι μια λειτουργία σύνδεσης εξωτερικής εξόδου.
- $Cyy : \cup i \in DYi \rightarrow Y^{\varnothing}$ είναι μια λειτουργία σύνδεσης εξωτερικής εξόδου
- Select : 2D → D είναι η συνάρτηση επίλυσης προτεραιότητας για να ρυθμίσει την ταυτόχρονη εμφάνιση συμβάντων. **[19]**



Εικόνα 3.3: Coupled DEVS Model
Πηγή εικόνας [3.3]

3.5 Στοιχεία του μοντέλου συστήματος που μπορούν να αξιοποιηθούν στην προσομοίωση

Αφού δημιουργηθεί το μοντέλο συστήματος και υλοποιηθεί ο μετασχηματισμός, θα φτάσουμε στο τελικό στάδιο το οποίο είναι η προσομοίωση και η εξόρυξη αποτελεσμάτων. Στόχος μας είναι να τρέξουμε μια προσομοίωση που είναι αντιπροσωπευτική του συστήματος που έχει υλοποιηθεί. Όπως αναφέραμε και στο κεφάλαιο 2, ο σκοπός μας με την προσομοίωση είναι να μελετήσουμε σε βάθος τη συμπεριφορά του υπό εξέταση συστήματος, βελτιστοποιώντας την απόδοσή του και προβλέποντας μελλοντικές συμπεριφορές.

Για να πετύχουμε το σκοπό αυτό, θα πρέπει να αξιοποιήσουμε σωστά όλα εκείνα τα δεδομένα του συστήματός μας, που θα μας επιτρέψουν να εξάγουμε ασφαλή και όσο το δυνατόν πιο ξεκάθαρα συμπεράσματα για τη λειτουργία του συστήματος ή μέρους αυτού.

Ορισμένα από τα στοιχεία του μοντέλου που θα μας βοηθήσουν να αντλήσουμε αντιπροσωπευτικά αποτελέσματα, είναι τα δομικά στοιχεία του μοντέλου, καθώς και οι συσχετίσεις (dependency, association, information flow, containment) που υπάρχουν μεταξύ των συστατικών. Μεταξύ των δομικών στοιχείων είναι τα Blocks που χρησιμοποιούνται στο μοντέλο και τα οποία είναι εκείνα που μας δείχνουν τη δομή του. Τα Blocks αποτελούνται από τις ιδιότητες τιμών (value properties) όπου είναι οι μεταβλητές για τη διατήρηση τιμών, τις λειτουργίες (Operations) και τους περιορισμούς (Constraints), που περιορίζουν ποσοτικά τα συστατικά ή κάποιες παραμέτρους ενός block. Τα στοιχεία αυτά μπορούν να δώσουν στο μοντέλο κατευθύνσεις ως προς το τι θέλουμε να πετύχουμε και αποτελούν βασική πηγή για την εξαγωγή των αντιπροσωπευτικών μοντέλων προσομοίωσης.**[20]**

Επίσης, για τα blocks προσδιορίζονται οι καταστάσεις και οι μεταβάσεις μεταξύ αυτών με SMDs, ενώ από τα IBDs για τα blocks, μπορεί να εξαχθεί ο τρόπος διασύνδεσης των

συστατικών blocks, ενός σύνθετου block. Παράλληλα τα parametric diagrams επιτρέπουν τον συνδυασμό συστατικών στοιχείων των blocks (ports, properties) με constraints, ορίζοντας με αυτόν τον τρόπο περιορισμούς λειτουργίας του συστήματος, που μπορούν να αξιοποιηθούν στο μοντέλο προσομοίωσης που θα παραχθεί.

Τέλος, από τα UCDs θα μπορούσαν να προκύψουν σενάρια εκτέλεσης προσομοίωσης, ιδιαίτερα εφόσον τα Use Cases ήταν συνδεδεμένα με συγκεκριμένες καταστάσεις του μοντέλου συστήματος.

3.6 Αποτελέσματα προσομοίωσης που μπορούν να παραχθούν

Για την εκτέλεση της προσομοίωσης χρησιμοποιήσαμε τη λειτουργία ενός Λεωφορείου ως παράδειγμα, το οποίο θα μας βοηθήσει να κατανοήσουμε ότι κάθε φυσικό σύστημα μπορεί να προσομοιωθεί και να εξαχθούν χρήσιμες πληροφορίες για τη λειτουργία του. Στη συγκεκριμένη περίπτωση του Λεωφορείου, στόχος μας είναι να παραχθούν αποτελέσματα που είναι αρκετά αντιπροσωπευτικά, με σκοπό να χρησιμοποιηθούν για τη σωστή συντήρηση και απόδοσή του.

Εξίσου σημαντικό αποτέλεσμα με την εκτέλεση της προσομοίωσης είναι η εξαγωγή συμπερασμάτων για τα όρια λειτουργίας ή κορεσμού του συστήματος, τα οποία θα μας δείξουν σημαντικά στοιχεία για την εξέλιξη του συστήματος στο χρόνο.

Επιπλέον, τα χαρακτηριστικά απόδοσης αλλά και οι μεταβολές που υπάρχουν στο σύστημα σχετικά με την κινητική κατάσταση και τη δυναμική ενέργεια του οχήματος, είναι παράγοντες αρκετά σημαντικοί για την απόδοση του συστήματος. Έχοντας μια συνολική και ασφαλή εικόνα για τα επιμέρους συστήματα μέσω της προσομοίωσης, μπορούμε να ελαχιστοποιήσουμε τα σφάλματα και να αυξήσουμε την απόδοση του συστήματος μας. Συμπερασματικά, με τα παραπάνω αποτελέσματα μας δίνεται η δυνατότητα αφενός να κατανοήσουμε καλύτερα το σύστημα και αφετέρου να καταλάβουμε τι είναι αυτό που θέλουμε να κερδίσουμε από το σύστημα.

Από την άλλη μεριά, μέσα από την προσομοίωση μπορούμε να αποκομίσουμε κάποια αποτελέσματα που κατατάσσονται στα τεχνικά αποτελέσματα. Μεταξύ αυτών είναι η καταγραφή του χρόνου παραμονής του συστήματος σε μια κατάσταση πριν μεταβεί στην επόμενη, αλλά και η εύρεση των μεγίστων/ελαχίστων χρόνων που χρειάζεται να μεταβεί από τη μια κατάσταση στην άλλη. Ένα ακόμα τεχνικό αποτέλεσμα που μπορεί να προκύψει μέσα από την προσομοίωση, είναι η εκτίμηση μέσης/ελάχιστης/μέγιστης τιμής σε μια παράμετρο λειτουργίας του συστήματος (π.χ. δείκτης φθοράς).

Κεφάλαιο 4: Προτεινόμενος Μετασχηματισμός

4.1 Μεθοδολογία

Η γενική μεθοδολογία που ακολουθήθηκε για να φτάσουμε από την δημιουργία του Μοντέλου Συστήματος (μέσω του Modelio), στη δημιουργία του μετασχηματισμού περιγράφεται ως εξής:

1. Καθορισμός υψηλού επιπέδου απαιτήσεων λειτουργίας του συστήματος ως περιπτώσεις χρήσης (use cases).
2. Δημιουργία του κεντρικού SysML μοντέλου μέσω του προγράμματος Modelio, ορίζοντας τα διαγράμματα αλλά και τις διασυνδέσεις μεταξύ των υποσυστημάτων του λεωφορείου.
3. Δημιουργία των Ports και των Constraints στο μοντέλο, που είναι απαραίτητα για τον υπολογισμό και την μετάδοση της πληροφορίας.
4. Εξαγωγή του τελικού μοντέλου SysML σε μορφή UML XMI από το Modelio και εισαγωγή του στο Eclipse.
5. Εισαγωγή UML-XMI file στο εργαλείο Eclipse IDE.
6. Μετατροπή του μοντέλου και των στοιχείων του από SysML σε εκτελέσιμο μοντέλο DEVS, με τη χρήση QVT-Operational μέσω του Eclipse IDE.
7. Εκτέλεση προσομοίωσης.
8. Αξιοποίηση αποτελεσμάτων.

Στον παρακάτω πίνακα παρουσιάζονται τα βήματα που περιγράφηκαν:

Βήμα	**Δραστηριότητα**	**Τρόπος/Μέθοδος**	**Εργαλείο**	**Αποτέλεσμα**
1	Καθορισμός απαιτήσεων	Μελέτη Συστήματος	-	Case Study
2	Δημιουργία Κεντρικού Μοντέλου	SysML	Modelio	Εισαγωγή SysML Διαγραμμάτων
3	Εισαγωγή Ports και Constraints	BDD	Modelio	Ολοκληρωμένο SysML μοντέλο
4	Εξαγωγή SysML Model	SysML Model	Modelio	UML-XMI file

5	Εισαγωγή UML-XMI file	-	Eclipse IDE	Μοντέλο Συστήματος στο Project
6	Μετασχηματισμός SysML σε εκτελέσιμο μοντέλο DEVS	QVT Operational, Μετα-μοντέλο DEVS & Μετασχηματισμός	Eclipse IDE	DEVS model
7	Εκτέλεση	Προσομοίωση διακριτού χρόνου	DEVS Suite Plus	Αποτελέσματα προσομοίωσης
8	Αξιοποίηση Αποτελεσμάτων	Ανάλυση	Εργαλείο ανάλυσης	Αποτελέσματα μελέτης

Πίνακας 4.1: Μεθοδολογία Εργασίας

Ειδικότερα, ως προς τον μετασχηματισμό επιλέξαμε να ακολουθήσουμε ένα συγκεκριμένο μονοπάτι το οποίο θα μας επιτρέψει να πάρουμε την τελική πληροφορία για το μοντέλο μας, η οποία είναι η φθορά των ελαστικών που θα έχει το Λεωφορείο.

Έτσι λοιπόν ξεκινάμε από το Use Case Diagram όπου τα Use Cases είναι συνδεδεμένα τα SMDs μέσω Information Flows. Στη συνέχεια, θα βρούμε τα States που υπάρχουν μέσα στα SMD και τα Constraints που περιέχονται σε αυτά.

Επόμενο βήμα θα είναι ο εντοπισμός του Parametric Diagram όπου περιέχει το Constraint που είναι υπεύθυνο για τον υπολογισμό της φθοράς των ελαστικών. Για τον εντοπισμό της πληροφορίας που εισάγεται στο Parametric θα αξιοποιηθούν οι συνδέσεις στο αντίστοιχο IBD.

4.2 Υλοποίηση Project μέσω του Eclipse

Όπως αναφέραμε και στο Κεφάλαιο 3, η υλοποίηση του κώδικα γραμμένου σε γλώσσα QVTo πραγματοποιήθηκε με τη χρήση του εργαλείου Eclipse IDE.

Για να ξεκινήσουμε, λοιπόν, την υλοποίηση του κώδικα ακολουθούμε τα εξής βήματα:

1. Δημιουργία του Project
    - File
        - ➔ New
            - ➔ Project

➔ Examples

➔ QVT Operational Transformation

➔ SimpleUML to RDB Transformation Project
(επιλέχθηκε το διαθέσιμο demo project για αξιοποίηση των στοιχείων του εξοικείωση με την QVT-O)

2. Δημιουργία των αρχείων του Project

• Εύρεση του διαθέσιμου από το **[21]** αρχείου "DEVSMM".ecore από Browser και εισαγωγή στο Project

• Εύρεση των αρχείων .uml που έχουμε εξάγει από το εργαλείο Modelio και εισαγωγή στο Project

• Για τη δημιουργία του QVT αρχείου κάνουμε Δεξί κλικ στο Project μας

➔ New

➔ Other

➔ Model to Model Transformation

➔ QVT Operational Transformation

➔ Next

➔ Δίνουμε το όνομα που θέλουμε

➔ Finish

• Για να μπορεί να τρέξει το αρχείο "DEVSMM".ecore θα πρέπει να το προσθέσουμε και από:

➔ Project

➔ Properties

➔ QVT Settings

➔ Metamodel Mappings

➔ Add

➔ Επιλογή "DEVSMM".ecore μέσω του Browser

➔ OK

➔ Apply

➔ Apply and Close

3. Συγγραφή κώδικα QVT στο αρχείο "*όνομα_αρχείου"*.qvto

(για τον μετασχηματισμό της UML σε DEVS)

4. Εκτέλεση κώδικα QVT Operational

• Για να τρέξουμε τον κώδικα επιλέγουμε δεξί κλικ στο QVTo αρχείο μας και στη συνέχεια:

➔Run As

➔ Run Configurations

Εκτελώντας λοιπόν τα παραπάνω βήματα, το αρχείο που θα εξαχθεί θα μπορούμε να το αξιοποιήσουμε για την εκτέλεση προσομοίωσης μέσω του εργαλείου DEVS Suite.

4.3 Αντιστοίχιση εννοιών/δομών SysML με έννοιες/δομές προσομοίωσης

Μεταξύ της γλώσσας SysML και του φορμαλισμού DEVS μπορούμε να διακρίνουμε αρκετές ομοιότητες. Αρχικά, στη SysML η ιεραρχική σύνθεση των στοιχείων της παραπέμπει στα Blocks, τα οποία μπορούν να περιέχουν και άλλα Blocks.

Επιπλέον, μέσω των Block Definition Diagrams (BDDs) μπορούμε να περιγράψουμε την ιεραρχία των blocks και μέσω των Internal Block Diagrams(IBD) βλέπουμε την ανάλυση των Ports και τα περιεχόμενα των Blocks. Αντίστοιχα στο DEVS, για την δημιουργία πολύπλοκων μοντέλων αλλά και τη διασύνδεση των ports εισόδου και εξόδου χρησιμοποιούνται τα coupled DEVS μοντέλα.

Μια ακόμα ομοιότητα μπορεί να βρεθεί, στα value properties που διαθέτουν τα SysML blocks και τα οποία μπορούν να συσχετίζονται μέσω constraint blocks σε Parametric Diagrams (PDs). Αντίστοιχα, τα atomic DEVS μοντέλα περιγράφουν την κατάσταση με state variables και με τις υπάρχουσες συσχετίσεις που υπάρχουν μεταξύ τους.

Επίσης, για την περιγραφή της συμπεριφοράς του μοντέλου βλέπουμε ότι στη SysML γίνεται με τη βοήθεια των State Machine Diagrams (SMD), ενω στο DEVS υλοποιείται με τις συναρτήσεις των atomic μοντέλων μέσω των εσωτερικών μεταβάσεων **(Πιν. 4.2)**.

Οι βασικές αντιστοιχίσεις μεταξύ του SySml και του DEVS αποτυπώνονται καλύτερα στον παρακάτω πίνακα:

SySml	**DEVS**
BDDs, IBDs	Coupled Model
Block	Atomic & Coupled Model
BDDs	Atomic Model
Value Property	Μεταβλητή κατάστασης

Constraints in PDs	Σχέσεις μεταξύ μεταβλητών κατάστασης

Πίνακας 4.2: Γενική Αντιστοίχιση εννοιών μεταξύ SySml και DEVS

Λαμβάνοντας υπόψη τα παραπάνω, για να υλοποιήσουμε το μετασχηματισμό μας, δημιουργήσαμε τις οντότητες και τις διασυνδέσεις που υπάρχουν μεταξύ αυτών. Μέσα από αυτή τη διαδικασία προέκυψε το μοντέλο SysML το οποίο χρησιμοποιούμε ως μοντέλο πηγή (Source model) για τον μετασχηματισμό. Από την άλλη πλευρά έχουμε το μετα-μοντέλο DEVS με βάση το οποίο δημιουργείται το μοντέλο-στόχος (Target model). Μεταξύ των μοντέλων πηγής και στόχου θα δημιουργήσουμε την 1-1 (ένα προς ένα) αντιστοιχία **(Πιν. 4.3)**. Έτσι, δημιουργούνται οι εξής αντιστοιχίες:

AA	**SysML**	**DEVS**
1	Model	DEVS
2	Block	Devs Atomic
3	Block	Devs Coupled
4	Information Flow	Internal Coupling
5	State Machine	T_States
6	Region	T_State_Set
7	State(atomic)	T_States_Set_Values
8	Parametric	T Condition
9	Block	Devs Atomic
10	Block	Devs Coupled
11	Value Property	State Variable
12	Constraint	DEVS_ATOMIC/STATES
13	IBD	External Output Coupling(coupled)
14	Activity	External Transition Function(atomic)

Πίνακας 4.3: Αντιστοίχιση εννοιών μεταξύ SySml και DEVS

4.4 Εκτελέσιμη QVT

Σχετικά με το τμήμα της QVT Operational, βασισμένοι στη μεθοδολογία μας, δίνουμε τα βασικά κομμάτια της λειτουργίας κάθε μέρους του αναγκαίου κώδικα. Παρακάτω λοιπόν παραθέτουμε ενδεικτικά κομμάτια του QVT Operational κώδικα όπως αυτά υλοποιήθηκαν στο εργαλείο Eclipse:

- **Ορισμός μετα μοντέλων**

```
modeltype UML uses 'http://www.eclipse.org/uml2/2.0.0/UML';
modeltype DEVS uses 'urn:DEVS_MM.ecore';
```

- **Ορισμός μετασχηματισμού από UML σε DEVS:**

```
transformation SysML_To_DEVS(in uml : UML, out devs : DEVS);
```

- **Σημείο έναρξης μετασχηματισμού από Uml σε DEVS**

```
main() {
        uml.rootObjects()[UML::Model]->map model2DEVSModel();
        uml.rootObjects()[UML::Model]->map model2DEVSScenario();
       }
```

- **Mapping Block με DEVS Coupled**

```
mapping UML::Class::class2DEVSCoupled() : DEVS::T_DEVS_COUPLED
when { self.ownedElement[Class] -> notEmpty() }
{
        MODEL_NAME := self -> map namedElement2modelName('') -> any(true);
        self.ownedElement[Class] -> map class2DEVSCoupled() -> any(true);
        self.ownedElement[Class] -> map class2DEVSAtomic() -> any(true);
}
```

- **Mapping Block με DEVS Atomic**

```
mapping UML::Class::class2DEVSAtomic() : DEVS::T_DEVS_ATOMIC
when { self.ownedElement[Class] -> isEmpty() }
{
        MODEL_NAME := self -> map namedElement2modelName('') -> any(true);
        STATES :=  self.ownedElement[StateMachine] -> map
stateMachineSub2DEVSAtomicStates() -> any(true);
}
```

- **Mapping Information Flows με DEVS Atomic**

```
mapping UML::InformationFlow::informationFlow2DEVSAtomic() :
DEVS::T_DEVS_ATOMIC
```

- **Mapping Information Flows με DEVS Atomic με περιορισμό πηγής το Use case και προορισμού το State Machine**

```
when {
        self.informationSource.oclIsKindOf(UML::UseCase) -> any(true) and
        self.informationTarget.oclIsKindOf(UML::StateMachine) -> any(true)
      }
```

- **Mapping State Machine με T_States/ Εμφάνιση Region**

```
mapping UML::StateMachine::stateMachineSub2DEVSAtomicStates() : DEVS::T_States
{
 STATE_SET := self.submachineState.region -> map states2DEVSAtomicStateSet() -> any(true);
}

mapping UML::StateMachine::stateMachine2DEVSAtomicStates() : DEVS::T_States
{
        STATE_SET := self.region -> map states2DEVSAtomicStateSet() -> any(true);
}
```

- **Mapping State Machine Region με T_States_Set**

```
mapping UML::Region::states2DEVSAtomicStateSet() : DEVS::T_State_Set
{
        STATE_SET_VALUES := self -> map region2DEVSAtomicStateSetValues() -> any(true);
}
```

- **Mapping State Machine Region με T_States_Set_Values**

```
mapping UML::Region::region2DEVSAtomicStateSetValues() : DEVS::T_State_Set_Values
{
        STATE_SET_VALUE := self.ownedElement[UML::State] -> map states2DEVSAtomicStateSetValues();
}
```

- **Αντιστοίχιση State με T_States_Set_Values**

```
mapping UML::State::states2DEVSAtomicStateSetValues() : DEVS::T_State_Set_Value
{
        text := self.name;
}
```

Τρέχοντας τον κωδικα της QVT Operational που αναφέραμε παραπάνω, έχουμε καταφέρει σε πρώτο στάδιο να δημιουργήσουμε όλα τα coupled και τα atomic models. Όσα από τα blocks έχουν άλλα blocks ενσωματωμένα θα εμφανίζονται ως coupled, ενώ όσα blocks δεν περιέχουν άλλα blocks θα εμφανίζονται ως atomic. Έπειτα δημιουργούμε το DEVS Atomic και Model. Στη συνέχεια στα στοιχεία του μοντέλου, που είναι τύπου Information Flow,

εφαρμόσαμε το Mapping έχοντας ως πηγή τα USE CASES και ως προορισμό τα State Machines. Με τον τρόπο αυτό θα εμφανίσουμε όλα τα States που είναι συνδεδεμένα με κάποιο Use Case μέσω των Information Flows.

Στο επόμενο βήμα, για να εμφανίσουμε όλα τα States που υπάρχουν στο μοντέλο μας, εφαρμόστηκε το mapping για State Machine σε DEVS Atomic States. Επειδή όμως έχουμε αρκετά επίπεδα και από τις δύο πλευρές (UML και DEVS), χρειάστηκε να δημιουργήσουμε διαδοχικά mappings. Αρχικά πήγαμε στο Region, που είναι ο χώρος που έχει μέσα τα States, κάνοντας mapping του Region με τα T_State_SET. Επιπλέον, πήγαμε για δεύτερη φορά στο Region, ώστε να μπούμε στα δύο επίπεδα του DEVS που περιέχονται οι αντιστοιχίες για τα State Machines, δηλαδή τα T_States_Set_Values. Μετά την υλοποίηση των παραπάνω, καταλήξαμε να εμφανίσουμε όλα τα States που έχουν δημιουργηθεί από την πλευρά του UML και να φτιάξουμε τα αντίστοιχα T_State_Set_Value από την πλευρά του DEVS.

Κεφάλαιο 5: Παράδειγμα

Επιλογή παραδείγματος

5.1 Λειτουργία Λεωφορείου

Στο παράδειγμα που ακολουθεί δημιουργούμε ένα SysML Μοντέλο το οποίο περιγράφει την διαδικασία που ακολουθείται για να εξάγουμε χρήσιμη πληροφορία για τη μέτρηση της φθοράς των ελαστικών ενός λεωφορείου. Στη συνέχεια, γίνεται ανάλυση του κάθε διαγράμματος που υλοποιήθηκε και των επιμέρων στοιχείων που συνθέτουν το κάθε διάγραμμα. Η δημιουργία του μοντέλου έγινε με το εργαλείο Modelio που αναλύθηκε στο Κεφάλαιο 2.
Τα διαγράμματα που υλοποιήθηκαν συγκεντρωτικά είναι τα εξής:

- Block Definition Diagram
- State Definition Diagram
- Use case
- Activity Diagram
- IBD Mechanical System
- IBD Electrical System
- IBD Steering/Transmission System
- Parametric Diagram

5.1.1 Block Definition Diagram

Ξεκινώντας με το Block Definition Diagram, έχουμε δημιουργήσει το βασικό μας Block που είναι εκείνο του λεωφορείου (Vehicle-Bus). Το Vehicle-Bus, περιέχει όλα τα Blocks - συστήματα που είναι τα βασικά μέρη για τη δημιουργία και τη σωστή λειτουργία του μοντέλου. Τα βασικά Blocks-συστήματα που έχουμε κατασκευάσει είναι τα εξής:

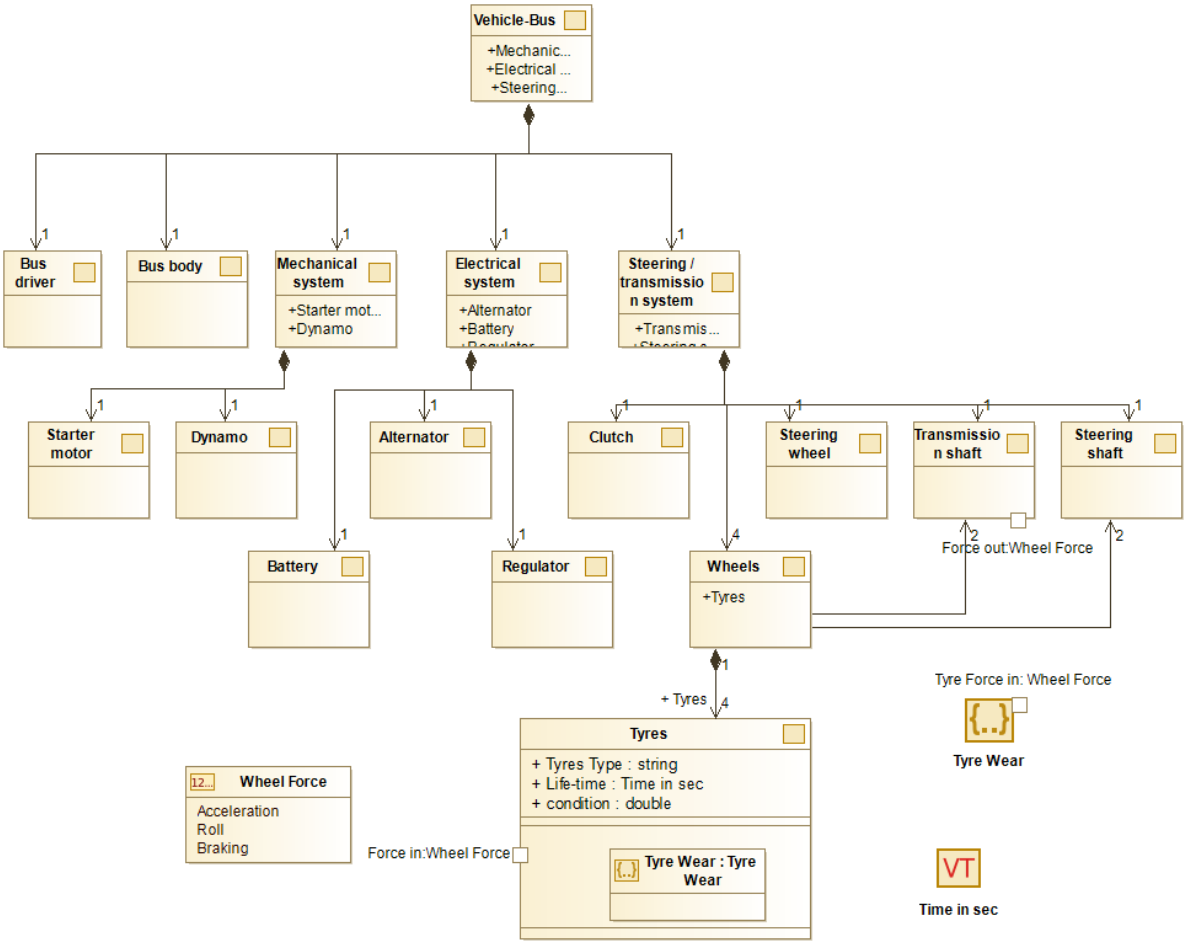
1. Bus driver
2. Bus body
3. Mechanical System
4. Electrical System
5. Steering/Transmission System

Τα παραπάνω συστήματα περιέχουν άλλα υποσυστήματα τα οποία συνθέτουν την ολοκληρωμένη λειτουργία του μοντέλου, αλλά και την εύρυθμη λειτουργία του λεωφορείου.

Ένα σημαντικό υποσύστημα, στο οποίο έχει στηριχθεί η παρούσα μελέτη, είναι τα ελαστικά (Tyres). Τα ελαστικά είναι μέρος του συστήματος Steering/Transmission System και είναι το

υποσύστημα που θέλουμε να μελετήσουμε ως προς τη φθορά που μπορεί να έχουν τα ελαστικά με την πάροδο του χρόνου και της κίνησης του οχήματος.

Στο Block Tyres έχει ενσωματωθεί ένα constraint στο οποίο έχει οριστεί ένας μαθηματικός κανόνας που υπολογίζει τη φθορά των ελαστικών. Έτσι λοιπόν, κάθε φορά που δίνεται εντολή για κάποια ενέργεια κίνησης (Accelerate, Roll, Braking) στο λεωφορείο, αυξάνεται αντίστοιχα και το condition (που περιέχεται μέσα στο constraint) που είναι η μεταβλητή για τον υπολογισμό της φθοράς των ελαστικών. Το constraint Tyre Wear λαμβάνει ως είσοδο, μέσω του port Tyre Force in: Wheel Force, την ενέργεια κίνησης (Accelerate, Roll, Braking) που προέρχεται από την port Force in: Wheel Force του Parametric Diagram **(Εικ. 5.1)**.



Εικόνα 5.1 :Block Definition Diagram

5.1.2 State Machine Diagram

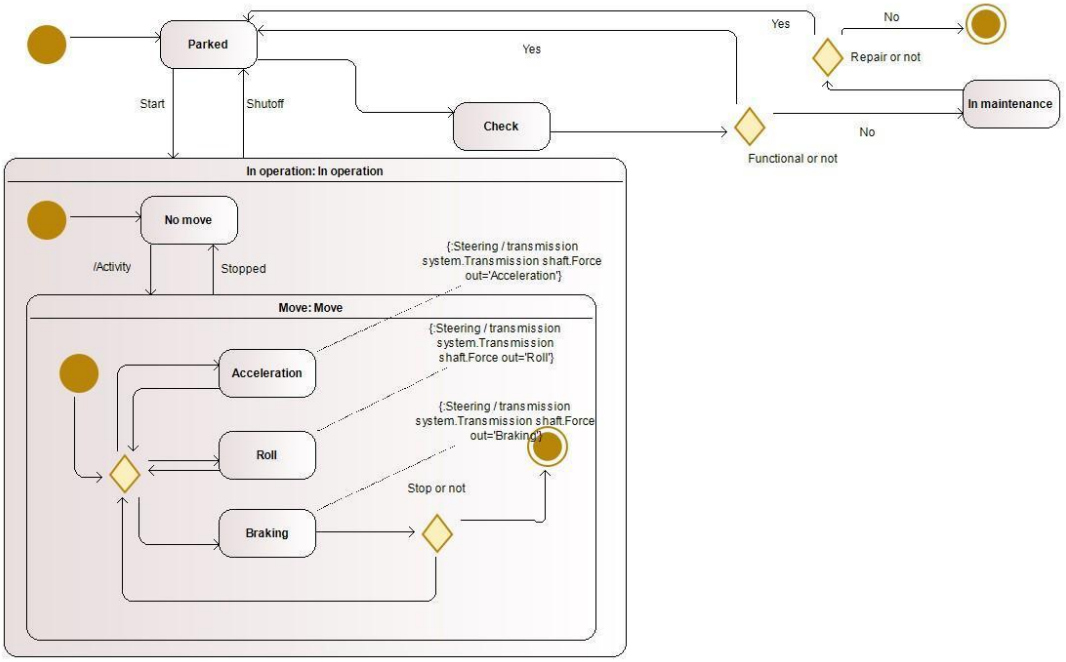
Τα διαγράμματα μηχανών κατάστασης δείχνουν τις διάφορες καταστάσεις μιας οντότητας. Μπορούν επίσης να δείξουν πώς μια οντότητα ανταποκρίνεται σε διάφορα συμβάντα αλλάζοντας από τη μία κατάσταση στην άλλη.

Στο παράδειγμά μας έχουμε δημιουργήσει τρία State Machine Diagrams όπου έχει γίνει πολυεπίπεδη εφαρμογή τους, ενσωματώνοντας το ένα μέσα στο άλλο.

Έτσι, δημιουργήσαμε το βασικό μας State όπου το Bus είναι στη κατάσταση Parked. Έπειτα, με την ενέργεια Start θα εισέλθει στο ενσωματωμένο State In operation και στην κατάσταση No move. Έτσι, το Bus θα είναι σε λειτουργία αλλά είναι ακινητοποιημένο.

Με την ενέργεια Accelerate όπου την ενεργοποιεί το Activity Diagram, το Bus μπαίνει στην επόμενη κατάσταση "Move". Σε αυτό το σημείο υπάρχουν οι καταστάσεις Acceleration, Roll και Braking, όπου όταν ενεργοποιείται η κάθε μία από αυτές τις καταστάσεις στέλνει την πληροφορία στην Port του Transmission shaft του IBD μέσω του Transmission shaft.Force out="Acceleration".

Όταν θα φτάσει στον τερματισμό των States αυτών, μέσω της ενέργειας Stopped θα μεταφερθεί στην κατάσταση Ακίνητο και στη συνέχεια μέσω της ενέργειας Shutoff, θα φτάσει στην αρχική κατάσταση Parked. Έπειτα θα πρέπει να γίνει ο έλεγχος των ελαστικών μέσω της κατάστασης Check και αν τα ελαστικά είναι λειτουργικά θα επανέλθει στην αρχική κατάσταση Parked, αλλιώς θα πρέπει να συνεχίσει στην κατασταση In maintenance. Τέλος, αν επισκευαστούν θα επιστρέψει στο Parked, ενώ αν όχι τότε θα υπάρχει τερματισμός **(Εικ. 5.2)**.

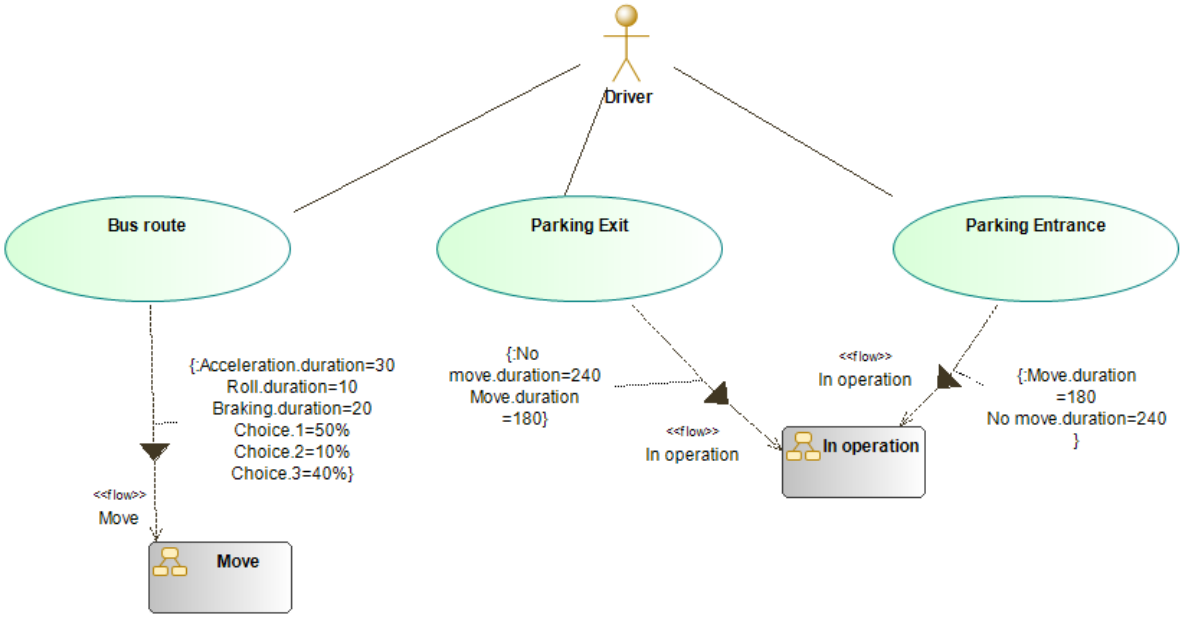


Εικόνα 5.2 :State Machine Diagram

5.1.3 Use Case

Με το Use Case βλέπουμε την αλληλεπίδραση που έχει ο χρήστης με το σύστημα. Στο παράδειγμά μας έχουμε μία οντότητα (Driver) που συνδέεται με τις περιπτώσεις χρήσης (Bus route, Parking Exit και Parking Entrance).

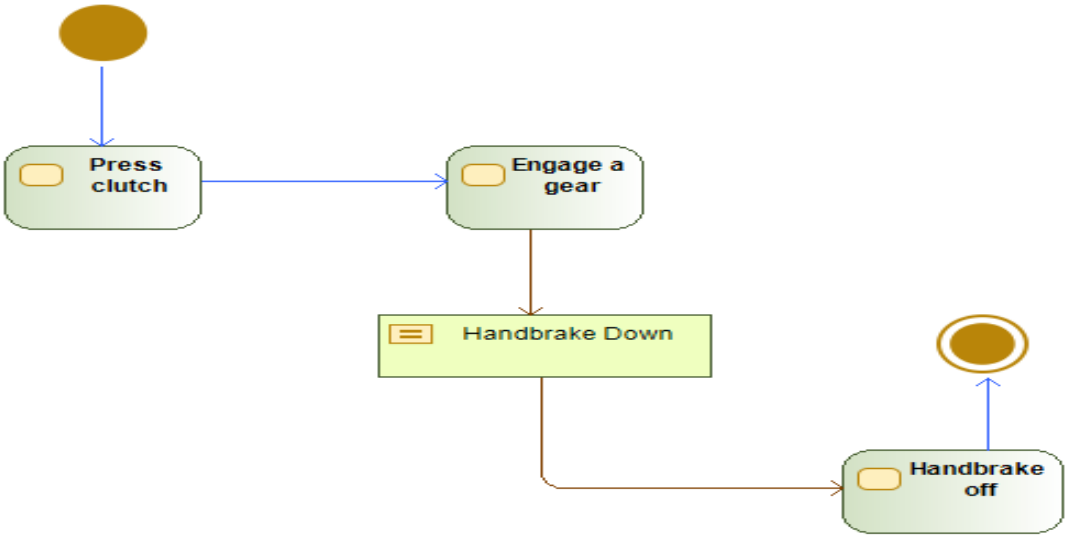
Κάθε μια από τις περιπτώσεις χρήσεις την έχουμε συνδέσει με τα αντίστοιχα States του State Machine Diagram όπου κινείται η κάθε περίπτωση χρήσης. Η σύνδεση έχει γίνει με Information Flow και έχει δοθεί σε κάθε σύνδεση ο χρόνος σε δευτερόλεπτα που το σύστημα παραμένει σε κάθε κατάσταση (No move.duration=240) με τη μορφή constraints. Επιπλέον, έχουμε δώσει και τις πιθανότητες που υπάρχουν να συμβεί μια κατάσταση , όταν είμαστε σε έναν κόμβο επιλογής (Choice). Επομένως, η πιθανότητα να βρεθεί το σύστημά μας στην κατάσταση Acceleration (Choice.1) είναι 50%, στην κατάσταση Roll (Choice.2) είναι 10% και στην κατάσταση Braking (Choice.3) είναι 40% **(Εικ. 5.3)**.



Εικόνα 5.3 :Use Case

5.1.4 Activity Diagram

Με το Activity Diagram αποτυπώνονται με γραφικό τρόπο οι ροές εργασίας (workflows) των δραστηριοτήτων και των επιμέρους ενεργειών του συστήματος. Στην περίπτωση του παραδείγματος μας, με το Activity Diagram ενεργοποιούμε την ενέργεια που γίνεται, για να πάμε από την κατάσταση Ακίνητο στο In operation. Αναλύοντας λοιπόν το Activity Diagram, αρχικά ενεργοποιείται ο συμπλέκτης (Press clutch) και στη συνέχεια ενεργοποιούνται οι ταχύτητες. Αφού στο τέλος απενεργοποιήσουμε το χειρόφρενο θα φτάσει στον τελικό κόμβο, όπου θα έχει γίνει η μεταφορά από το State Ακίνητο στο State In operation **(Εικ. 5.4)**.

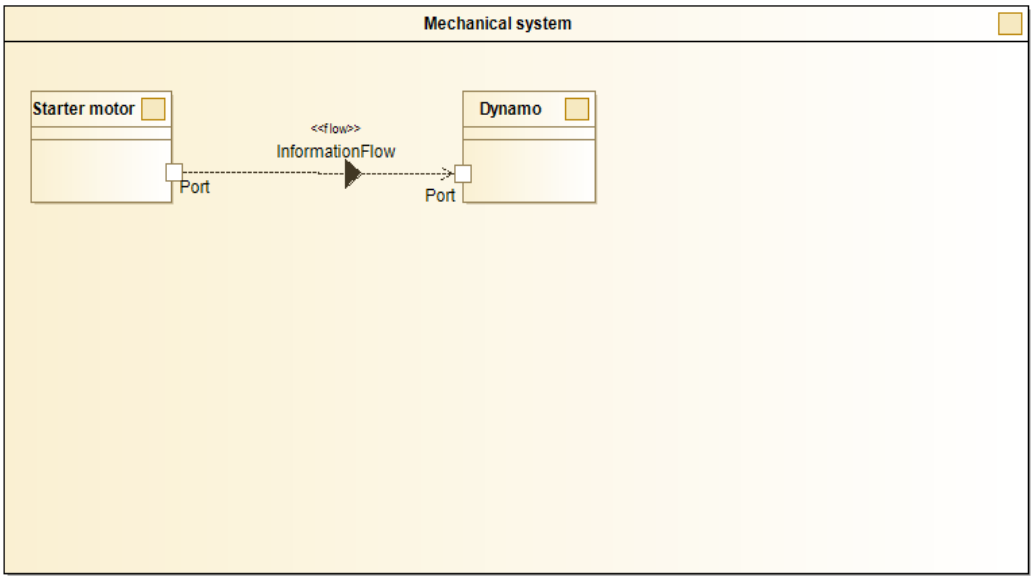


Εικόνα 5.4: Activity Diagram

Στη συνέχεια αποτυπώνονται οι διασυνδέσεις των υποσυστημάτων που περιέχονται στο BDD.

5.1.5 IBD Mechanical system

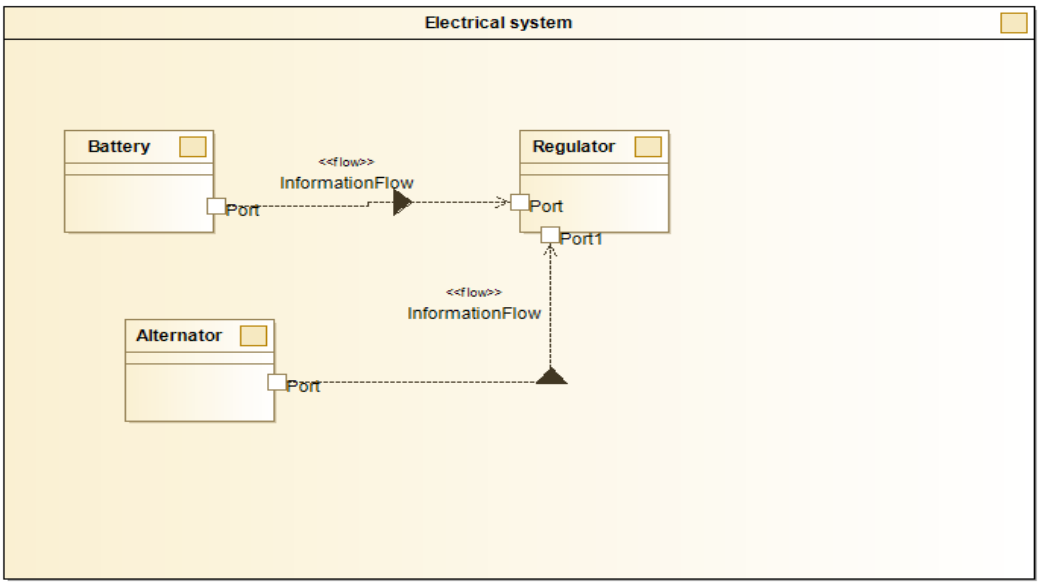
Αρχικά βλέπουμε τη σύνδεση του Μηχανικού μέρους και την επαφή των υποσυστημάτων Starter motor με το Dynamo μέσω των Ports **(Εικ.5.5)**.



Εικόνα 5.5: IBD Mechanical system

5.1.6 IBD Electrical system

Το Ηλεκτρικό σύστημα περιέχει την Μπαταρία (Battery), τον Ρυθμιστή (Regulator) και τον εναλλάκτη (Alternator). Οι συνδέσεις γίνονται με το Information Flow μέσω των αντίστοιχων Ports **(Εικ. 5.6)**.

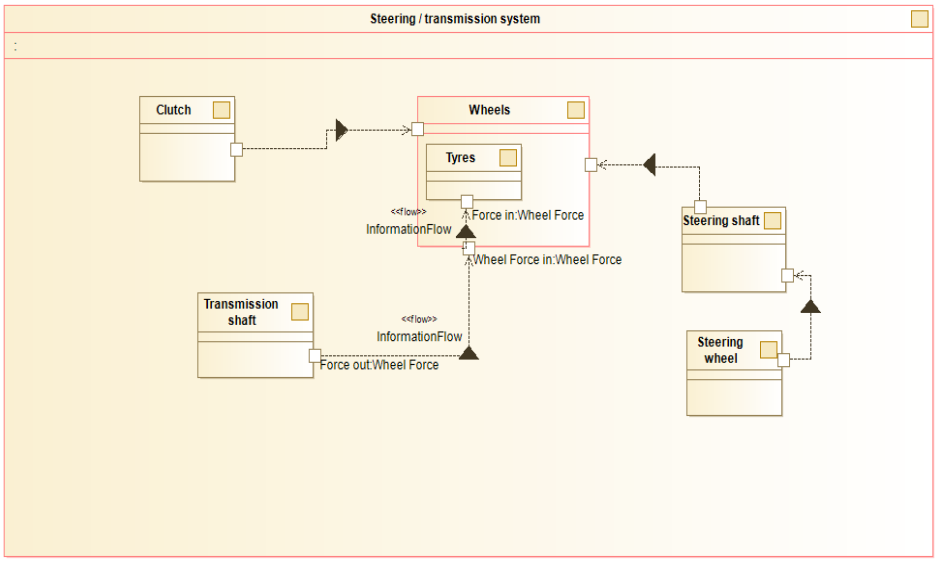


Εικόνα 5.6: IBD Electrical system

5.1.7 IBD Steering/transmission system

Μέσω του IBD Steering/transmission system βλέπουμε την σύνδεση των υποσυστημάτων που περιέχονται στο Σύστημα Διεύθυνσης/Μετάδοσης. Για να υπολογίσουμε την φθορά των ελαστικών στο Constraint, χρειαζόμαστε την κατάσταση στην οποία βρίσκεται κάθε φορά το μοντέλο μας και περιέχεται στο State Machine Diagram. Η πληροφορία αυτή, στέλνεται από το State Machine Diagram κάθε φορά που αλλάζει η κατάσταση.

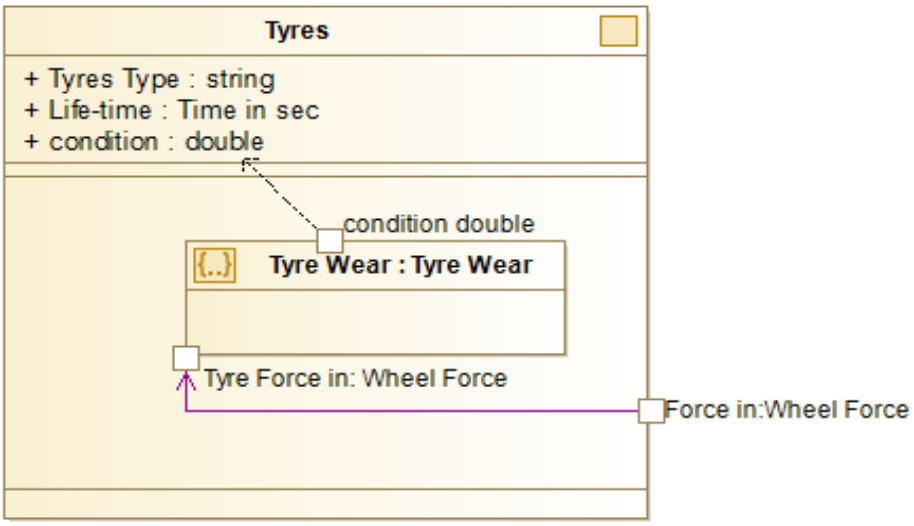
Επομένως, η βασική πληροφορία που χρειαζόμαστε στο IBD Steering/transmission system λαμβάνεται μέσω του Transmission shaft και το port Force out:Wheel Force. Έπειτα, μεταφέρεται στο port Wheel Force in:Wheel Force του Wheel και στη συνέχεια στην Port Force in:Wheel Force του Tyres. Όλες οι συνδέσεις στα IBD γίνονται με το Information Flow **(Εικ. 5.7)**.



Εικόνα 5.7: IBD Steering/transmission system

5.1.8 Parametric Diagram

Το Parametric Diagram περιέχει το constraint Tyre Wear που δημιουργήσαμε στο Block Definition Diagram**,** το αλφαριθμητικό string, τον χρόνο Time in sec και το condition που είναι double δηλαδή για την αναπαράσταση των τιμών κινητής υποδιαστολής. Το Parametric Diagram μέσω του port Force in:Wheel Force λαμβάνει ως είσοδο την κατάσταση στην οποία βρίσκεται το bus και μεταφέρει αυτή την τιμή στο in port Tyre Force (Wheel Force) του constraint. Τέλος, υπάρχει το Port condition (double) όπου συνδέεται με την μεταβλητή condition που περιέχεται στο constraint **(Εικ. 5.8)**.



Εικόνα 5.8: Parametric Diagram

5.1.9 Constraint

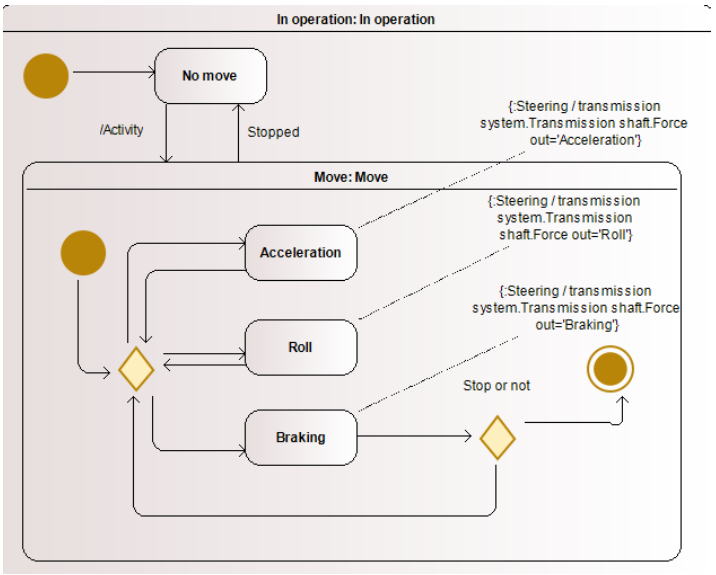
Το βασικό constraint που έχουμε χρησιμοποιήσει στο BDD είναι αυτό που κάνει τον υπολογισμό της φθοράς των ελαστικών. Ανάλογα με την κατάσταση στη οποία βρίσκεται κάθε φορά, γίνεται και ο αντίστοιχος υπολογισμός **(Εικ. 5.9)**.

```
{:if condition<=0 then return
     end if
  Tyre Wear
  if 'Tyre Force in'=='Acceleration'
     then condition=condition-0.00025*condition
        and if
  if 'Tyre Force in'=Roll=''
     then condition=condition-0.00001*condition
        and if
if 'Tyre Force in'=='Braking'
     then condition=condition-0.00030*condition
        and if
         }
```

Εικόνα 5.9 Constraint Υπολογισμού φθοράς ελαστικών

Για να γίνει ο παραπάνω υπολογισμός θα πρέπει να γνωρίζουμε σε ποια κατάσταση βρίσκεται κάθε φορά το μοντέλο. Την πληροφορία αυτή την λαμβάνουμε μέσα από τα Constraints που έχουν οριστεί στο SMD **(Εικ.5.10)**.

Κάθε φορά που το μοντέλο θα βρίσκεται σε μία από τις τρεις καταστάσεις (Acceleration, Roll, Braking) θα στέλνει την πληροφορία στο Transmission Shaft του IBD Steering/transmission system, ώστε στη συνέχεια να φτάσει στο τελικό Constraint που γίνεται ο υπολογισμός της φθορας των ελαστικών.



Εικόνα 5.10 Constraint καταστάσεων μοντέλου

5.2 Εξαγώγιμο αρχείο του Μοντέλου

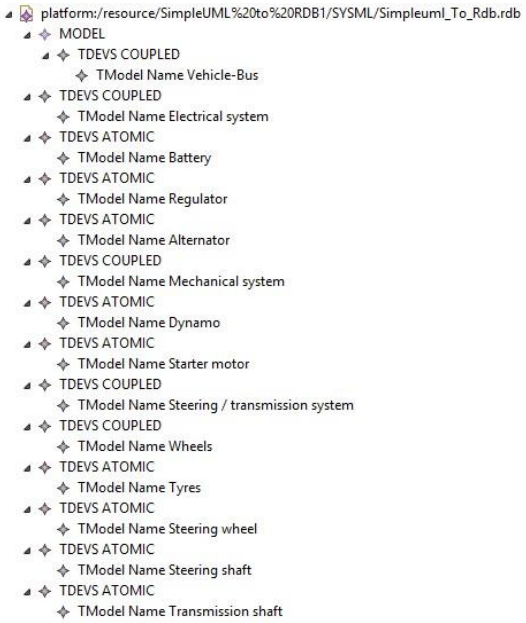
Αφού ολοκληρωθεί το μοντέλο μας, εξάγοντας το μέσω του Modelio, θα λάβουμε το ".uml" αρχείο που θα το μεταφορτώσουμε στο Eclipse για να συνεχίσουμε τη διαδικασία που περιγράφηκε στο κεφάλαιο 4. Εκτός από το παραγόμενο μοντέλο που περιέχει όλα τα διαγράμματα, παράγεται και ένα αρχείο profile.uml σχετικό με την SysML.
Η επόμενη εικόνα **(Εικ. 5.11)** παρουσιάζει ένα απόσπασμα του bus.uml:


```
<?xml version="1.0" encoding="UTF-8"?>
<uml:Model xmi:version="2.1" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1" xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore"
xmlns:uml="http://www.eclipse.org/uml2/3.0.0/UML" xmi:id="_OQqXIK0BEeumkP7s5T74Dg" name="bus">
  <eAnnotations xmi:id="_OQqXIa0BEeumkP7s5T74Dg" source="Objing">
    <contents xmi:type="uml:Property" xmi:id="_OQqXIq0BEeumkP7s5T74Dg" name="exporterVersion">
      <defaultValue xmi:type="uml:LiteralString" xmi:id="_OQqXI60BEeumkP7s5T74Dg" value="3.0.0"/>
    </contents>
  </eAnnotations>
  <ownedRule xmi:id="_OQqXJK0BEeumkP7s5T74Dg" constrainedElement="_OQqXta0BEeumkP7s5T74Dg">
    <specification xmi:type="uml:LiteralString" xmi:id="_OQqXJa0BEeumkP7s5T74Dg" value="{:if condition&lt;=0 then return&#xA;&#x9;end if&#xA;  Tyre Wear&#xA;  if 'Tyre
Force in'=='Acceleration'&#xA;&#x9;then condition=condition-0.00025*condition&#xA;&#x9;     and if&#xA;  if 'Tyre Force in'=Roll='&#xA;&#x9;then condition=condition-
0.00001*condition&#xA;&#x9;     and if&#xA;if 'Tyre Force in'=='Braking'&#xA;&#x9;then condition=condition-0.00030*condition&#xA;             and if&#xA;&#x9;&#x9;}"/>
  </ownedRule>
  <ownedRule xmi:id="_OQqXJq0BEeumkP7s5T74Dg" constrainedElement="_OQqX2a0BEeumkP7s5T74Dg">
    <specification xmi:type="uml:LiteralString" xmi:id="_OQqXJ60BEeumkP7s5T74Dg" value="{:if condition&lt;=0 then return&#xA;&#x9;end if&#xA;  Tyre Wear&#xA;  if 'Tyre
Force in'=='Acceleration'&#xA;&#x9;then condition=condition-0.00025*condition&#xA;&#x9;     and if&#xA;  if 'Tyre Force in'=Roll='&#xA;&#x9;then condition=condition-
0.00001*condition&#xA;&#x9;     and if&#xA;if 'Tyre Force in'=='Braking'&#xA;&#x9;then condition=condition-0.00030*condition&#xA;             and if&#xA;&#x9;&#x9;}"/>
  </ownedRule>
  <ownedRule xmi:id="_OQqXKK0BEeumkP7s5T74Dg" constrainedElement="_OQqXW60BEeumkP7s5T74Dg">
    <specification xmi:type="uml:LiteralString" xmi:id="_OQqXKa0BEeumkP7s5T74Dg" value="Steering / transmission system.Transmission shaft.Force out='Acceleration'"/>
  </ownedRule>
  <ownedRule xmi:id="_OQqXKq0BEeumkP7s5T74Dg" constrainedElement="_OQqXXK0BEeumkP7s5T74Dg">
    <specification xmi:type="uml:LiteralString" xmi:id="_OQqXK60BEeumkP7s5T74Dg" value="Steering / transmission system.Transmission shaft.Force out='Roll'"/>
  </ownedRule>
  <ownedRule xmi:id="_OQqXLK0BEeumkP7s5T74Dg" constrainedElement="_OQqXXa0BEeumkP7s5T74Dg">
    <specification xmi:type="uml:LiteralString" xmi:id="_OQqXLa0BEeumkP7s5T74Dg" value="Steering / transmission system.Transmission shaft.Force out='Braking'"/>
  </ownedRule>
  <ownedRule xmi:id="_OQqXLq0BEeumkP7s5T74Dg" constrainedElement="_OQ0IIK0BEeumkP7s5T74Dg">
    <specification xmi:type="uml:LiteralString" xmi:id="_OQqXL60BEeumkP7s5T74Dg" value="No move.duration=240&#xA;Move.duration&#xA;=180"/>
  </ownedRule>
  <ownedRule xmi:id="_OQqXMK0BEeumkP7s5T74Dg" constrainedElement="_OQ0IIq0BEeumkP7s5T74Dg">
    <specification xmi:type="uml:LiteralString" xmi:id="_OQqXMa0BEeumkP7s5T74Dg"
value="Acceleration.duration=30&#xA;Roll.duration=10&#xA;Braking.duration=20&#xA;Choice.1=50%&#xA;Choice.2=10%&#xA;Choice.3=40%"/>
  </ownedRule>
  <ownedRule xmi:id="_OQqXMq0BEeumkP7s5T74Dg" constrainedElement="_OQ0IJK0BEeumkP7s5T74Dg">
    <specification xmi:type="uml:LiteralString" xmi:id="_OQqXM60BEeumkP7s5T74Dg" value="Move.duration&#xA;=180&#xA;No move.duration=240&#xA;"/>
```


Εικόνα 5.11: Εξαγωγή UML Μοντέλου

5.3 Υλοποίηση και Αποτέλεσμα Μετασχηματισμού

Εκτελώντας τον κώδικα QVT Operational που δημιουργήσαμε, καταφέραμε αρχικά να δημιουργήσουμε όλα τα coupled και τα atomic models. Όσα από τα blocks διαθέτουν άλλα blocks θα εμφανίζονται ως coupled, ενώ όσα blocks δεν έχουν άλλα blocks θα εμφανίζονται ως atomic. Έτσι, με αυτόν τον τρόπο θα πάρουμε όλα τα blocks που περιέχονται στο μοντέλο μας όπως βλέπουμε στην παρακάτω εικόνα **(Εικ. 5.12)**.



Εικόνα 5.12: Αποτελέσματα Μετασχηματισμού Coupled & Atomic

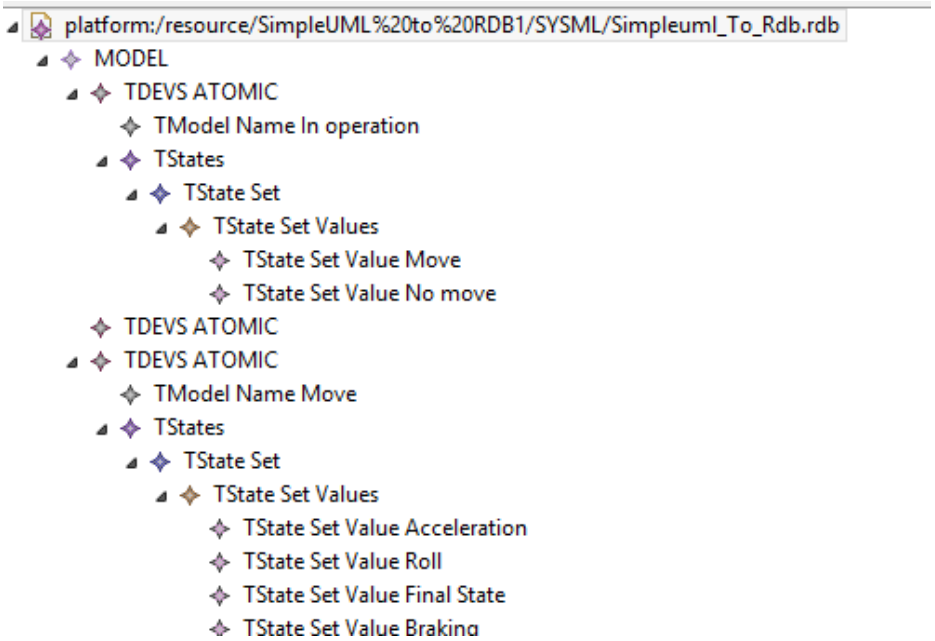
Στη συνέχεια, δημιουργούμε το DEVS Atomic, το Model και όλα τα State που υπάρχουν μέσα στο μοντέλο μας. Αρχικά βλέπουμε στο Use Case Diagram, τα State Machines (Move, In Operation) που είναι συνδεδεμένα με τα Use Case μέσω των Information Flows.

Έπειτα, εμφανίζεται η αντιστοίχιση που έγινε μεταξύ των στοιχείων UML, όπου περιέχει όλα τα States (Move, No Move) που υπάρχουν μέσα στο In Operational και του DEVS με τα επίπεδα State Set και State Set Value. Τέλος, εμφανίζουμε όλα τα States μέσα στο Move, δηλαδή τα (Acceleration, Roll, Braking και Final State) και φαίνονται και τα αντίστοιχα επίπεδα του DEVS (TStates, TState Set, TState Set Value).

Βασιζόμενοι στην υπάρχουσα υλοποίηση του μετασχηματισμού, μια από τις μελλοντικές επέκτασεις που μπορούν να υλοποιηθούν είναι η αξιοποίηση των Parametric Diagrams, τα οποία θα μας δώσουν τον υπολογισμό της φθοράς των ελλαστικών με την πάροδο του χρόνου.

Επιπλέον, η αξιοποίηση των constraints που υπάρχουν μεταξύ των συνδέσεων των Use Case-State και των Machine Diagrams, θα μας επιτρέψουν να γνωρίζουμε ποια είναι η πιθανότητα να πάει το σύστημά μας σε μια κατάσταση αλλά και πόσο χρόνο θα μείνει σε αυτή την κατάσταση.

Στην επόμενη εικόνα **(Εικ. 5.13)** φαίνονται τα States που έχουν δημιουργηθεί από την εκτέλεση του κώδικα QVT Operational μέσα από το εργαλείο Eclipse.



Εικόνα 5.13:Αποτελέσματα Μετασχηματισμού States

5.4 Αναμενόμενα Αποτελέσματα Προσομοίωσης

Κατασκευάζοντας το μοντέλο και υλοποιώντας τον μετασχηματισμό θα μπορούμε να εξάγουμε αποτελέσματα που είναι αντιπροσωπευτικά, ώστε να αξιοποιηθούν για τη σωστή λειτουργία και απόδοση του συστήματος. Για την προσομοίωση έχει επιλεγεί το εργαλείο DEVS-Suite, με το οποίο έχουμε κάνει δοκιμές προσομοίωσης μοντέλων και λάβαμε χρήσιμη γνώση για τη λειτουργία του εργαλείου. Μετά την εξοικείωση με το DEVS-Suite είμαστε σε θέση να χρησιμοποιήσουμε σωστά το εργαλείο και να εξάγουμε αξιοποιήσιμα αποτελέσματα για κάθε μοντέλο προσομοίωσης.

Έτσι, αφού τρέξουμε την προσομοίωση αναμένουμε να πάρουμε αποτελέσματα για κάθε σενάριο (use case) που έχει υλοποιηθεί στο μοντέλο μας, τα οποία θα μας βοηθήσουν να κατανοήσουμε τη λειτουργία του συστήματος. Ένα ακόμη αναμενόμενο σημαντικό αποτέλεσμα, που μπορεί να μας δώσει πολλές πληροφορίες σχετικά με τη λειτουργία του μοντέλου, είναι ο χρόνος παραμονής του συστήματος στις καταστάσεις (parked, in operation, no move, move, acceleration, braking, roll). Γνωρίζοντας πόσο χρόνο παρέμεινε σε μία κατάσταση το συστήμά μας, θα μπορούμε να αξιοποιήσουμε τα δεδομένα για να υπολογίσουμε την αντίστοιχη επιβάρυνση που μπορεί να έχει το σύστημα στο μέλλον.

Επιπλέον, μέσα από την προσομοίωση αναμένουμε να λάβουμε την τιμή της παραμέτρου λειτουργίας "condition" των ελαστικών, όπου με την παρακολούθηση της τιμής αυτής μας δίνεται η δυνατότητα να γνωρίζουμε την φθορά που θα έχουν τα ελαστικά με την πάροδο του χρόνου. Αντίστοιχα αν είχαμε και άλλες παραμέτρους θα λαμβάναμε την αντίστοιχη πληροφορία.

Κεφάλαιο 6: Συμπεράσματα

Στο κεφάλαιο αυτό παρατίθενται η προσφορά της μελέτης, οι μελλοντικές επεκτάσεις για την βελτίωση και συνέχιση της συγκεκριμένης διπλωματικής, καθώς επίσης καταλήγουμε και σε χρήσιμα συμπεράσματα σχετικά με την παρούσα εργασία.

6.1 Προσφορά της Μελέτης

Τα σημαντικότερα σημεία της προσφοράς της μελέτης μας είναι τα εξής:

- Η συγκεκριμένη μελέτη δίνει μια λύση για προσομοίωση με συνολική αξιοποίηση μοντέλου συστήματος. Στην παρούσα εργασία ασχοληθήκαμε με την συνολική αξιοποίηση μοντέλου συστήματος μέσω των διασυνδέσεων για την προσομοίωση και όχι με κάποιο διάγραμμα μεμονωμένα για την προσομοίωση του.

- Ανεξάρτητα από το βαθμό υλοποίησης του μετασχηματισμού, αν θέλουμε να μελετήσουμε ένα μοντέλο SysML με γενικό τρόπο, αυτό που μπορούμε να λάβουμε σαν αποτέλεσμα είναι ο χρόνος παραμονής σε καταστάσεις και οι τιμές για κάποια properties των blocks.

- Υπάρχει απλότητα μοντελοποίησης στη συγκεκριμένη μελέτη, χωρίς επεκτάσεις για συγκεκριμένο πεδίο ή περιβάλλον προσομοίωσης, που ήταν και το ζητούμενο της εργασίας.

- Για την ερευνητική ομάδα του Χαροκόπειου πανεπιστημίου ήταν μια εισαγωγή για πρώτη φορά στην QVT-Operational και των σχετικών εργαλείων.

- Η λύση υλοποιείται με αξιοποίηση standards (MOF/UML/SysML/QVT). Τα εργαλεία που χρησιμοποιήθηκαν ήταν open source και standards based(Modelio-UML, Eclipse-QVT-O) και δεν χρειάστηκε να ακολουθήσουμε κάποια λύση πάνω σε κάποιο εμπορικό προϊόν.

- Έγινε αξιοποίηση προηγούμενης ερευνητικής δραστηριότητας για το DEVS Meta-model.

6.2 Μελλοντικές επεκτάσεις

Για την υλοποίηση της παρούσας διπλωματικής στηριχτήκαμε στη UML, στη SysML, στη QVT-Operational, στη DEVS, στα εργαλεία Modelio, Eclipse και DEVS Suite. Κατασκευάσαμε ένα μοντέλο το οποίο περιείχε βασικά διαγράμματα της SysML και λίγους περιορισμούς, ώστε να μπορεί από τη μια να χρησιμοποιηθεί σαν γενικό μοντέλο συστήματος SysML (χωρίς επεκτάσεις για συγκεκριμένο πεδίο ή περιβάλλον προσομοίωσης) και από την άλλη να αντιπροσωπεύει ένα αληθινό σύστημα λεωφορείου. Μέσω κατάλληλων διεργασιών έγινε η υλοποίηση του μετασχηματισμού με χρήση της QVT Operational και δημιουργήθηκε ένα εκτελέσιμο μοντέλο προσομοίωσης.

Έχοντας ως οδηγό τις διαπιστώσεις αυτές αλλά και τα αποτελέσματα της μέχρι τώρα δουλειάς, προτείνουμε για τη μελλοντική έρευνα τα ακόλουθα:

- Επέκταση και εξέλιξη του μετασχηματισμού προσομοίωσης μέσω της διαδικασίας αυτόματης παραγωγής κώδικα, με σκοπό στην εκτέλεση προσομοίωσης να έχουμε εξαγωγή ασφαλών συμπερασμάτων για τη λειτουργία του συστήματος αλλά και να λάβουμε ακριβή αποτελέσματα για τα επιμέρους συστήματα του μοντέλου.

- Επέκταση του υπάρχοντος μοντέλου προσθέτοντας επιπλέον νέες οντότητες και λειτουργίες για την εξαγωγή επιπλέον πληροφοριών για τη λειτουργία του συστήματος.

- Μελέτη μοντέλων από άλλα πεδία.

- Ανίχνευση άλλων τμημάτων του μοντέλου συστήματος, όπου θα είχε καλύτερη εφαρμογή η προσομοίωση συνεχούς χρόνου (π.χ. collision detection, ασκούμενες δυνάμεις, ροή υγρών, κλπ.).

- Εξαγωγή αναλυτικών αποτελεσμάτων της προσομοίωσης και διαγραμματική ανάλυσής τους.

- Μοντελοθεωρητική μελέτη και θεμελίωση προσέγγισης.

6.3 Συμπεράσματα - Ανακεφαλαίωση

Ο στόχος της παρούσας διπλωματικής εργασίας ήταν να δημιουργηθεί ένα γενικό μοντέλο συστήματος SysML, το οποίο δεν θα περιέχει καμία επέκταση για συγκεκριμένο πεδίο ή περιβάλλον προσομοίωσης. Μέσω της δημιουργίας του μοντέλου αυτού, στοχεύουμε στην εξοικείωση με τη χρησιμότητα και τη χρήση της SysML, τη μοντελοποίηση, τη λογική της προσομοίωσης και τις διαδικασίες αυτόματης παραγωγής κώδικα. Επιπλέον, στοχεύσαμε στο κομμάτι αυτό των μοντέλων SysML, που θα μπορούσαν να προσομοιωθούν με προσομοίωση διακριτού χρόνου.

Έτσι, σχεδιάστηκε και υλοποιήθηκε ένα μοντέλο όπου περιέχει βασικά στοιχεία της SysML. Ξεκινώντας από ένα πλήρως γραφικό περιβάλλον σχεδιάστηκε και δημιουργήθηκε το μοντέλο βήμα-βήμα μέσω του εργαλείου Modelio και στη συνέχεια μέσω του Eclipse υλοποιήθηκε ο μετασχηματισμός με τη χρήση της QVT-Operational. Συμπεραίνουμε λοιπόν, ότι για την υλοποίηση μεγάλου κομματιού της διπλωματικής χρησιμοποιήθηκαν εργαλεία που είναι open source και δεν απαιτούσαν αρκετή συγγραφή κώδικα, κάτι το οποίο μας βοήθησε να επικεντρωθούμε στην ανάλυση του συστήματος που είχαμε για να υλοποιήσουμε.

Για να δημιουργήσουμε τις προϋποθέσεις του μοντέλου μας, πήραμε ως βάση το παράδειγμα του Λεωφορείου το οποίο είναι πολύπλοκο και μπορεί να γίνει αρκετά σύνθετο σύστημα ως προς τις διασυνδέσεις που έχει. Στην υλοποίηση του συστήματος μας προσπαθήσαμε να δημιουργήσουμε οντότητες και λειτουργίες οι οποίες μπορούν μέσω μικρών αλλαγών να χρησιμοποιηθούν σε οποιοδήποτε άλλο σύστημα. Η χρήση των εργαλείων για την μοντελοποίηση συστημάτων στη SySML παρέχουν σημαντική βοήθεια και ευελιξία στη δημιουργία τους. Επιπλέον, δίνεται η δυνατότητα εξαγωγής του μοντέλου μοντελοποίησης ώστε να προσομοιωθεί με τη χρήση μετασχηματισμών.

Μέσω της εργασίας αυτής δόθηκε το κίνητρο για την περαιτέρω ενασχόληση με μετασχηματισμούς μοντέλων (QVT-O), διερεύνηση μεθόδων προσομοίωσης και βελτίωση της αυτοματοποίησης της παραγωγής εκτελέσιμων μοντέλων προσομοίωσης από γενικά μοντέλα συστήματος SysML, χωρίς επεκτάσεις για συγκεκριμένο πεδίο ή περιβάλλον προσομοίωσης. Η ευχρηστία και η απλότητα των εργαλείων που χρησιμοποιήθηκαν σε αυτή την εργασία, δίνουν δυνατότητα ακόμα και σε αρχάριους να ασχοληθούν με τον τομέα αυτό, για την διεύρυνση των γνώσεων τους, αλλά και την εξοικείωση με το πεδίο της μοντελοποίησης και προσομοίωσης.

Παράρτημα

Βιβλιογραφία

[1], Κοτρώνης Χρήστος 2016, "Μία SysML προσέγγιση για την μελέτη Συστημάτων Μεταφορών", Χαροκόπειο Πανεπιστήμιο

[2] SmartCitySysML: A SysML Profile for Smart Cities Applications https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7974970/

[3] Marco Mori, Andrea Ceccarelli, Paolo Lollini, Bernhard Frömel, Francesco Brancati, Andrea Bondavalli, "Systems-of-systems modeling using a comprehensive viewpoint-based SysML profile" https://onlinelibrary.wiley.com/doi/epdf/10.1002/smr.1878

[4] UML, Meta Meta Models and Profiles, https://www.uml-diagrams.org/uml-meta-models.html

[5] [20] ΔΟΥΜΠΙΩΤΟΥ, ΚΑΝΔΑΛΗΣ 2013, "Μοντελοκεντρική Ανάπτυξη Ενσωματωμένων Συστημάτων", Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης.

[6] MODELIO, https://www.modeliosoft.com/en/other-products/modelio-sa-sysml.html

[7] Εισαγωγή στα μοντέλα προσομοίωσης, https://repository.kallipos.gr/bitstream/11419/2489/1/02_chapter_1.pdf

[8] Object Management Group (OMG), https://www.omg.org/

[9] QVT (Query/View/Transformation), https://en.wikipedia.org/wiki/QVT

[10] [11] L. Lengyel, "An overview of QVT" https://www.researchgate.net/publication/264883432_An_Overview_of_QVT

[12] Eclipse IDE 2021-03, https://www.eclipse.org/downloads/packages/release/2021-03/r

[13] DEVS-Suite Simulator, https://sourceforge.net/projects/devs-suitesim/

[14] QVT transformations with Eclipse, http://reqpro.blogspot.com/search/label/QVT

[15] QVT Operational, https://sdqweb.ipd.kit.edu/wiki/QVT#QVT_Operational

[16][19] Κάπος Γεώργιος-Δημήτριος 2016, "Μια Μοντελοστραφής Προσέγγιση για την Αυτοματοποίηση της Προσομοίωσης SySML Μοντέλων Συστημάτων", Χαροκόπειο Πανεπιστήμιο.

[17] [21] George-Dimitrios Kapos, Vassilis Dalakas, Mara Nikolaidou and Dimosthenis Anagnostopoulos 2014, SIMULATION "An integrated framework for automated simulation of SysML models using DEVS"

[18] Moon Ho Hwang, Su Kyeong Cho, Bernard P. Zeigler, Feng Lin 2007, "Processing Time Bounds of Schedule-Preserving DEVS", USA.

Links Εικόνων

[2.1] https://sysml.org/sysml-faq/

[2.2] Vitali Schneider 2018, "A Standards-based Framework for Test-driven Agile Simulation", The technical faculty of the Friedrich-Alexander-University Erlangen-Nuremberg https://www.researchgate.net/publication/337030135_A_Standards-based_Framework_for_Test-driven_Agile_Simulation

[2.3] https://www.researchgate.net/figure/Architecture-du-Standard-QVT_fig19_260552707

[3.2] https://www.researchgate.net/figure/DEVS-Atomic-model-definition_fig1_327853815

[3.3] http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S1794-12372014000200002

Tool Links

Modelio, https://store.modelio.org/resource/modules/sysml-architect-open-source.html

Eclipse IDE 2021-03, https://www.eclipse.org/downloads/packages/release/2021-03/r

DEVS-Suite Simulator, https://sourceforge.net/projects/devs-suitesim/