ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Μοντελοκεντρική Διαχείριση της Δομής του Διαδικτύου των Πραγμάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μιχάλης Α. Μπαχράς

Επιβλέπων : Κωνσταντίνος Κοντογιάννης
Καθηγητής ΕΜΠ

Αθήνα, Ιούλιος 2018

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Μοντελοκεντρική Διαχείριση της Δομής του Διαδικτύου των Πραγμάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μιχάλης Α. Μπαχράς

Επιβλέπων : Κωνσταντίνος Κοντογιάννης
Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 24η Ιουλίου 2018.

.....	.....	.....
Νικόλαος Παπασπύρου	Ανδρέας-Γεώργιος Σταφυλοπάτης	Γιώργος Στάμου
Αναπληρωτής Καθηγητής	Καθηγητής	Αναπληρωτής Καθηγητής

Αθήνα, Ιούλιος 2018

.....

Μιχάλης, Α. Μπαχράς

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Copyright © Μιχάλης, Μπαχράς, 2018.

Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Περίληψη

Η ολοένα και μεγαλύτερη εισχώρηση των συστημάτων του Διαδικτύου των Πραγμάτων καθιστά απαραίτητη τη δημιουργία ενός καθολικού τρόπου παραγωγής των μοντέλων που αφορούν τη δομή του εκάστοτε συστήματος. Ταυτόχρονα, η ολοένα μεγαλύτερη κλιμακωσιμότητα και πολυπλοκότητα αυτών των συστημάτων κάνει επιτακτική τη δημιουργία αλγορίθμων για τη διαχείριση και την εύρυθμη λειτουργία τους.

Η συντριπτική πλειοψηφία αυτών των συστημάτων προϋποθέτουν συγκεκριμένα χαρακτηριστικά και κομμάτια στη δομή τους, ανεξάρτητα από τον τομέα στον οποίο βρίσκουν εφαρμογή. Όλα αυτά τα συστήματα έχουν σκοπό να εκπληρώσουν κάποιο στόχο, που είναι άλλωστε και ο λόγος ύπαρξης τους, ο οποίος έχει καθοριστεί από το πλαίσιο στο οποίο λειτουργεί το κάθε σύστημα και από τον μηχανικό λογισμικού που το σχεδιάζει.

Ο σκοπός αυτής της εργασίας είναι η δημιουργία ενός μεταμοντέλου της δομής του Διαδικτύου των Πραγμάτων, από το οποίο θα εκπορεύονται όλα τα πιθανά μοντέλα για τέτοιου είδους συστήματα, η δημιουργία ενός μεταμοντέλου για τα δέντρα στόχων των συστημάτων και η σχεδίαση αλγορίθμου για την εύρεση κατάλληλης αλληλουχίας ατομικών και απλών ενεργειών(Actions) για την εκπλήρωση των διάφορων υποστόχων του δέντρου στόχων και κατ' επέκταση του στόχου ρίζα.

Τα μεταμοντέλα αυτά κατασκευάστηκαν με βάση το πρότυπο Meta Object facility(MOF) ενώ ειδικά για τα μοντέλα στόχων χρησιμοποιήθηκαν πηγές από την σύγχρονη βιβλιογραφία[30][31]. Επιπλέον, προτείνονται διάφορες επεκτάσεις των μοντέλων - στόχων που έχουν ως σκοπό την παραγωγή εκφραστικότερων και πιο λεπτομερών μοντέλων - στόχων για την ικανοποίηση κάθε σύγχρονης ανάγκης των συστημάτων του Διαδικτύου των Πραγμάτων.

Εξάλλου, οι αλληλουχίες των ενεργειών που πρέπει να εκτελεστούν αποτελούν θεμελιώδες κομμάτι για τη λειτουργία αυτών των συστημάτων και καθορίζουν τη συμπεριφορά του συστήματος, ανάλογα με ποια αλληλουχία έχει επιλεχθεί και το πλαίσιο στο οποίο λειτουργεί(context).

Λέξεις κλειδιά:

Μοντελοκεντρική διαχείριση, τεχνολογία λογισμικού, δέντρα στόχων, μοντέλα στόχων, μεταμοντέλο στόχων, ακολουθίες ενεργειών, Διαδίκτυο των Πραγμάτων.

Abstract

The increasing penetration of the Internet of Things systems makes it necessary to create a global way of producing the models that are relevant to the structure of the system. At the same time, the growing scalability and complexity of these systems makes it imperative to create algorithms for managing and securing their regular operation.

The overwhelming majority of these systems require specific features and components in their structure, regardless of the domain they are applying to. All of these systems are designed to fulfill a goal, which is also the reason for their existence, which is determined by the context in which each system works and by the software engineer who designs it.

The purpose of this work is to create a metamodel of the structure of the Internet of Things, from which all the possible models for such systems will emerge, the creation of a metamodel for goal trees and the design of an algorithm to find a suitable sequence of Actions to fulfill the various sub-goals of the goal tree and thus the root goal node.

These metamodels were constructed on the basis of the Meta Object facility (MOF), while sources from the modern literature [30] [31] were used specifically for the goal models. In addition, various extensions of the goal models are proposed that aim to produce more expressive and more detailed goal models to meet every modern need of the Internet of Things systems.

In addition, the sequences of the actions to be performed are a fundamental part of the functioning of these systems and determine the behavior of the system, depending on which sequence is chosen and the context in which it operates.

Keywords:

Model-driven Architecture, Software Engineering, Goal Tree, Goal Model, Goal Metamodel, Action Plans, Internet of Things

Πίνακας Περιεχομένων

Κεφάλαιο 1: Εισαγωγή 15
1.1 Κίνητρο 15
1.2 Περιγραφή Προβλήματος 16
1.3 Συνεισφορά της Διπλωματικής Εργασίας 17
1.4 Οργάνωση του Κειμένου 18
Κεφάλαιο 2: Σχετικές Εργασίες 21
2.1 Εισαγωγή 21
2.2 Μοντελοποίηση Συστημάτων Λογισμικού 21
2.2.1 MOF 21
2.2.2 XMI 23
2.2.3 Eclipse Modelling Framework 23
2.3 Δέντρα Στόχων 24
2.4 Προσωποποιημένο Περιβάλλον-Πλαίσιο 27
2.5 Διαδίκτυο των Πραγμάτων 27
Κεφάλαιο 3: Αρχιτεκτονική του Συστήματος 31
3.1 Εισαγωγή 31
3.2 Επισκόπηση της Αρχιτεκτονικής 33
3.3 Αναλυτική Περιγραφή των Components 35
3.3.1 Finder Component 35
3.3.2 Rule Manager Component 36
3.3.3 Transformer Component 38
3.3.4 Component Sorter 38
3.3.5 Component Combiner 39
3.3.6 Component Context Manager 40
3.4 Επικοινωνία μεταξύ των components του Συστήματος 41
3.4.1 Ακολουθιακό Διάγραμμα Δημιουργίας Δέντρου - Στόχων 41
3.4.2 Ακολουθιακό Διάγραμμα Ανανέωσης Δέντρου - Στόχων 42
3.4.3 Ακολουθιακό Διάγραμμα Εύρεσης Πλάνων Ατομικών Ενεργειών 43
Κεφάλαιο 4:Εννοιολογικά Μοντέλα 45
4.1 Εισαγωγή 45
4.2 Εννοιολογικό Μοντέλο Διαδικτύου των Πραγμάτων..... 45
4.2.1 Κλάση Entity 45
4.2.2 Κλάση User 45
4.2.3 Κλάση Attribute 46
4.2.4 Κλάση Thing 46
4.2.5 Κλάση Property 46
4.2.6 Κλάση Physical Thing 46

4.2.7 Κλάση Digital Artifact 46
4.2.8 Κλάση Active Digital Artifact 47
4.2.9 Κλάση Passive Digital Artifact 47
4.2.10 Κλάση Virtual Thing 47
4.2.11 Κλάση Device 47
4.2.12 Κλάση Communicator 47
4.2.13 Κλάση Port 47
4.2.14 Κλάση Device State 48
4.2.15 Κλάση Sensor 48
4.2.16 Κλάση Actuator 48
4.2.17 Κλάση Action 48
4.2.18 Κλάση Tag 48
4.2.19 Κλάση Transition 49
4.2.20 Κλάση Rule 49
4.2.21 Κλάση Information Resource 49
4.2.22 Κλάση Information 49
4.2.23 Κλάση Fog 50
4.2.24 Κλάση Fog Node 50
4.2.25 Κλάση Fog Services 50
4.2.26 Κλάση Operations 50
4.2.27 Κλάση Database 51
4.2.28 Κλάση Policy Repository 51
4.3 Εννοιολογικό Μοντέλο Δέντρων Στόχων 61
4.3.1 Κλάση Goal Model 61
4.3.2 Κλάση Node 61
4.3.3 Κλάση GoalNode 62
4.3.4 Κλάση Task 62
4.3.5 Κλάση Action 62
4.3.6 Κλάση Decomposition 62
4.3.7 Κλάση Resource 62
4.3.8 Κλάση Control Node 63
4.3.9 Κλάση Parallel Node 63
4.3.10 Κλάση Link 63
4.3.11 Κλάση Dependency Link 63
4.3.12 Κλάση LogicalPrecondition 63
4.3.13 Κλάση TemporalLink 63
4.3.14 Κλάση TimeoutLink 64
4.3.15 Κλάση TimeDifferenceLink 64
4.3.16 Κλάση ParallelLink 64
4.3.17 Κλάση ResourceDependency 64
4.3.18 Κλάση Contribution 64
4.3.19 Κλάση ContributionType 64
4.3.20 Κλάση ContextualComponent 65

4.3.21 Κλάση TemporalPrecondition 65
4.4 Εννοιολογικό Μοντέλο Γράφου Εξάρτησης Ενεργειών..72
4.4.1 Κλάση ADG 72
4.4.2 Κλάση Link 72
4.4.3 Κλάση Node 72
4.4.4 Κλάση ActionNode 72
4.4.5 Κλάση SplitNode 72
4.4.6 Κλάση JoinNode 73
Κεφάλαιο 5:Αλγόριθμοι Μετασχηματισμού και εύρεσης Ενεργειών 75
5.1 Κανόνες Μετασχηματισμού 75
5.1.1 Κανόνας Συνεισφοράς 75
5.1.2 Κανόνας Παραλληλισμού 77
5.1.3 Κανόνας Λογικής Προϋπόθεσης 79
5.1.4 Κανόνας Χρονικής Προϋπόθεσης 81
5.1.5 Κανόνας Εξάρτησης Πηγών Δεδομένων 82
5.1.6 Κανόνας Χρονικής Διαφοράς 83
5.1.7 Κανόνας Χρονικής Απόστασης 84
5.2 Αλγόριθμος Ενορχήστρωσης 85
Κεφάλαιο 6:Πειράματα και Μετρήσεις Απόδοσης 91
6.1 Επίδραση Μεγέθους Μοντέλου Δέντρου Στόχων και OR αποσυνθέσεων στην Απόδοση του Συστήματος 91
6.1.1 Χρονική απόδοση 91
6.1.2 Απόδοση Μνήμης 94
6.2 Επίδραση Αριθμού Παραλληλιών και OR Αποσυνθέσεων στην Απόδοση του Συστήματος 96
Κεφάλαιο 7: Επίλογος 101
7.1 Σύνοψη και Συμπεράσματα 101
7.2 Μελλοντικές επεκτάσεις 102
Κεφάλαιο 8: Βιβλιογραφία 103

Πίνακας Σχημάτων

Σχήμα 2.1 Αρχιτεκτονική Mof 22
Σχήμα 2.2 Παράδειγμα δέντρου - στόχων 26
Σχήμα 2.3 Αφηρημενη αρχιτεκτονικη IoT 29
Σχήμα 3.1 Περίγραμμα λειτουργιων 32
Σχήμα 3.2 Ψηφιδικό διάγραμμα συστήματος 33
Σχήμα 3.3 Ψηφιδικό διάγραμμα Finder 36
Σχήμα 3.4 Ψηφιδικό διάγραμμα Rule Manager 37
Σχήμα 3.5 Ψηφιδικό διάγραμμα Component 38
Σχήμα 3.6 Ψηφιδικό διάγραμμα Sorter 39
Σχήμα 3.7 Ψηφιδικό διάγραμμα Combiner 40
Σχήμα 3.8 Ψηφιδικό διάγραμμα Context Manager 40
Σχήμα 3.9 Ακολουθιακό διάγραμμα Δημιουργίας Δέντρου - Στόχων 42
Σχήμα 3.10 Ακολουθιακό Διάγραμμα Ανανέωσης Δέντρου - Στόχων 43
Σχήμα 3.11 Ακολουθιακό Διάγραμμα Εύρεσης Πλάνων Ατομικών Ενεργειών 44
Σχήμα 4.1-Διάγραμμα IoT metamodel 1 52
Σχήμα 4.2-Διάγραμμα IoT metamodel 2 53
Σχήμα 4.3-Διάγραμμα IoT metamodel 3 54
Σχήμα 4.4-Διάγραμμα IoT metamodel 4.1 55
Σχήμα 4.5-Διάγραμμα IoT metamodel 4.2 56
Σχήμα 4.6-Διάγραμμα IoT metamodel 5 57
Σχήμα 4.7-Διάγραμμα IoT metamodel 6.1 58
Σχήμα 4.8-Διάγραμμα IoT metamodel 6.2 59
Σχήμα 4.9-Διάγραμμα IoT metamodel 60
Σχήμα 4.10-Διάγραμμα goal metamodel 1 66
Σχήμα 4.11-Διάγραμμα goal metamodel 2 67
Σχήμα 4.12-Διάγραμμα goal metamodel 3 68
Σχήμα 4.13-Διάγραμμα goal metamodel 4 69
Σχήμα 4.14-Διάγραμμα goal metamodel 5 70
Σχήμα 4.15-Διάγραμμα goal metamodel 71
Σχήμα 4.16-Διάγραμμα ADG metamodel 73
Διάγραμμα 6.1 92
Διάγραμμα 6.2 92
Διάγραμμα 6.3 92
Διάγραμμα 6.4 93
Διάγραμμα 6.5 93
Διάγραμμα 6.6 94
Διάγραμμα 6.7 94

Διάγραμμα 6.8 95
Διάγραμμα 6.9 95
Διάγραμμα 6.10 96
Διάγραμμα 6.11 97
Διάγραμμα 6.12 97
Διάγραμμα 6.13 98
Διάγραμμα 6.14 98
Διάγραμμα 6.15 99
Διάγραμμα 6.16 99

Κεφάλαιο 1
Εισαγωγή

1.1 Κίνητρο

Η βιομηχανία παραγωγής λογισμικού παγκοσμίως έχει γνωρίσει τα τελευταία χρόνια μία τεράστια άνθηση το οποίο έχει σαν αποτέλεσμα να εγείρονται ζητήματα διαλειτουργικότητας και ανεξαρτησίας από συγκεκριμένες υλοποιήσεις. Μια λύση σε αυτό είναι η δημιουργία αφαιρετικών προτύπων και πρωτοκόλλων τα οποία εξασφαλίζουν τη διαλειτουργικότητα διάφορων υλοποιήσεων, χωρίς να εξαρτώνται από την υλοποίηση αυτή καθαυτή.

Σε αυτό το πλαίσιο έχουν αναπτυχθεί τεχνικές οι οποίες εξασφαλίζουν αυτές τις προϋποθέσεις με σημαντικότερη τη Μοντελοκεντρική Μηχανική(Model Driven Engineering,MDE), όπου χρησιμοποιούνται συστηματικά μοντέλα σαν αντικείμενα πρώτης τάξης για να αναπαραστήσουν το σύστημα και τις διάφορες φάσεις του κύκλου ζωής του. Το βασικό χαρακτηριστικό της Μοντελοκεντρικής Μηχανικής είναι ότι κάθε σύστημα μπορεί να αναπαρασταθεί με ένα μοντέλο, το οποίο βασίζεται συντακτικά σε ένα μεταμοντέλο[1]. Η σημαντικότερη προσέγγιση στο πεδίο της Μοντελοκεντρικής Μηχανικής είναι η Μοντελοκεντρική Αρχιτεκτονική (Model Driven Architecture,MDA) του Object Management Group(OMG).

Κυρίαρχο ρόλο στην MDA προσέγγιση έχει το μοντέλο που αναπαριστά το σύστημα, καθώς αποτελεί το πρωτεύον αντικείμενο για τη σχεδίαση, δημιουργία, συντήρηση και αποσφαλμάτωση του συστήματος. Το μοντέλο αναπαριστά αφαιρετικά και τυπικά ένα σύστημα όσον αφορά τη δομή και οργάνωσή του, τη λειτουργία του και τη σχέση του με το περιβάλλον. Σκοπός, εξάλλου, της MDA προσέγγισης είναι η παραγωγή και δημιουργία έργων λογισμικού μέσα από μια αφαιρετική διαδικασία υψηλού επιπέδου. Τέτοια διαδικασία είναι η σταδιακή δημιουργία ενός συστήματος λογισμικού μέσα από μια ακολουθία μετασχηματισμών μοντέλων, ξεκινώντας από ένα μοντέλο ανεξάρτητο από την πλατφόρμα(Platform Independent Model,PIM) και καταλήγοντας σε ένα μοντέλο συνδεδεμένο με μία συγκεκριμένη πλατφόρμα (Platform Specific Model,PSM)[2]. Εξέχων ρόλο στην MDA προσέγγιση παίζουν τεχνολογίες όπως: Unified Modelling Language (UML), Meta Object Facility(MOF), Common Warehouse Metamodel (CWM) και άλλα.

Παράλληλα, τεράστια ανάπτυξη έχει τα τελευταία χρόνια ο τομέας του Διαδικτύου των Πραγμάτων. Μέχρι στιγμής πάνω από 23 δισεκατομμύρια συσκευές είναι συνδεδεμένες στο Διαδίκτυο των Πραγμάτων ενώ προβλέψεις αναφέρουν ότι μέχρι το 2025 οι συσκευές αυτές θα τετραπλασιαστούν. Όπως είναι λογικό η πολυπλοκότητα και η κλιμακωσιμότητα τέτοιων συστημάτων αποτελούν τεράστια πρόκληση για τους σχεδιαστές έργων λογισμικού, τα οποία σχετίζονται με το Διαδίκτυο των Πραγμάτων. Ως εκ τούτου, η παραπάνω προσέγγιση και τεχνική είναι κατάλληλη για τη διαχείριση συστημάτων τέτοιας έκτασης και πολυπλοκότητας.

Η ύπαρξη μεταμοντέλου για τη γενική δομή του Διαδικτύου των Πραγμάτων αλλά και για το δέντρο στόχων του συστήματος δίνει μια εποπτική εικόνα στο μηχανικό λογισμικού για το πώς θα σχεδιάσει το σύστημα ενώ προσφέρει και έναν καθολικό τρόπο απεικόνισης, ανεξάρτητα από τον τομέα στον οποίο βρίσκουν εφαρμογή, πράγμα πολύ χρήσιμο στη μετέπειτα συντήρηση του συστήματος, η οποία πιθανώς να μην γίνεται από το ίδιο προσωπικό. Έτσι, υπάρχει κέρδος σε χρόνο και σε χρήματα και η βελτίωση του συστήματος καθίσταται ευκολότερη και πιο γρήγορη.

Βέβαια, η εύρυθμη λειτουργία των συστημάτων αυτών απαιτεί την εύρεση συγκεκριμένων ακολουθιών ατομικών και απλών ενεργειών, οι οποίες θα ικανοποιούν το δέντρο - στόχων που έχει καθοριστεί, με βάση το περιβάλλον στο οποίο βρίσκεται το σύστημα, την κατάστασή του αλλά και τις επιθυμίες του χρήστη σχετικά με τη λειτουργία του. Έτσι, είναι αναγκαία η αυτόματη παράγωγη όλων αυτών των ακολουθιών, με τη μικρότερη δυνατή ανθρώπινη παρέμβαση, καθώς ο ανθρώπινος παράγοντας δημιουργεί λάθη σε συστήματα τέτοιας πολυπλοκότητας και μεγέθους.

1.2 Περιγραφή Προβλήματος

Στόχος της παρούσας εργασίας είναι η ανάπτυξη ενός μεταμοντέλου για την οργάνωση της δομής του Διαδικτύου των Πραγμάτων που θα αποτελέσει τη βάση για την ανάπτυξη κοινών συντακτικά μοντέλων στο συγκεκριμένο ερευνητικό πεδίο, ενός μεταμοντέλου για την έκφραση των στόχων του συστήματος και ένα λογισμικό αυτόματης παραγωγής ακολουθιών από ενέργειες των συστημάτων IoT για την ικανοποίηση των στόχων. Πιο συγκεκριμένα, το μεταμοντέλο θα πρέπει να ικανοποιεί κάθε ανάγκη οποιουδήποτε συστήματος IoT,όπως είναι η μοντελοποίηση όλων των δυνατών συστατικών μερών ενός πολύπλοκου IoT συστήματος και η διασύνδεση αυτών των μερών. Επιπλέον, το λογισμικό θα πρέπει να λαμβάνει υπόψη του το περιβάλλον - πλαίσιο και τις επιδράσεις που έχει αυτό στο σύστημα, τις επιθυμίες του χρήστη και τους στόχους που έχουν καθοριστεί στο δέντρο-στόχων, το οποίο εκπορεύεται συντακτικά και εννοιολογικά από το μεταμοντέλο που έχει εισαχθεί.

Η ανάπτυξη όλων των παραπάνω προϋποθέτει την εύρεση όλων εκείνων των χαρακτηριστικών που καθορίζουν ένα IoT σύστημα, τις λειτουργικές απαιτήσεις που έχουν και τον τρόπο με τον οποίο ικανοποιούνται οι στόχοι.

Έτσι, προκύπτουν τα παρακάτω ερωτήματα που μας απασχόλησαν κατά τη σχεδίαση των μεταμοντέλων και του λογισμικού:

- Ποια είναι σε αφαιρετικό επίπεδο τα αναγκαία συστατικά μέρη που σχηματίζουν ένα IoT σύστημα;
- Πώς είναι το σύστημα συνδεδεμένο με το περιβάλλον και πώς αλληλεπιδρά με αυτό;

- Ποιες είναι οι πιθανές διασυνδέσεις μεταξύ αυτών των συστατικών μερών;
- Ποια είναι τα χαρακτηριστικά τους;
- Από ποια υποσύνολα αποτελούνται;
- Τι είδους στόχους εκπληρώνουν;
- Τι σχέσεις έχουν οι στόχοι μεταξύ τους;
- Τι εξαρτήσεις έχουν οι στόχοι μεταξύ τους;
- Τι εξαρτήσεις έχουν οι στόχοι με το σύστημα;
- Με ποιόν τρόπο θα απεικονιστούν αυτές οι εξαρτήσεις;
- Με ποιόν τρόπο θα παραχθούν οι διάφορες ακολουθίες των ενεργειών;
- Πώς επηρεάζονται από το περιβάλλον-πλαίσιο;

1.3 Συνεισφορά της Διπλωματικής Εργασίας

Η παρούσα εργασία παρουσιάζει το λογισμικό που αναπτύχθηκε για την εύρεση όλων των αλληλουχιών από απλές ενέργειες που ικανοποιούν το δέντρο - στόχων καθώς και όλα τα σχετικά μεταμοντέλα που δημιουργήθηκαν για την υλοποίηση του λογισμικού. Πιο συγκεκριμένα, η συνεισφορά της διπλωματικής εργασίας συνοψίζεται στα εξής σημεία:

- Τον ορισμό και τη σχεδίαση του μεταμοντέλου για τη δομή του Διαδικτύου των Πραγμάτων. Εισάγονται όλα εκείνα τα μέρη που είναι απαραίτητα για τη δημιουργία του Διαδικτύου των Πραγμάτων, καθορίζονται τα θεμελιώδη χαρακτηριστικά αυτών και οι συσχετίσεις τόσο μεταξύ τους όσο και με εξωτερικούς παράγοντες, όπως χρήστες και υπηρεσίες, αλλά και οι διάφοροι τρόποι με τους οποίους είναι δυνατή η επικοινωνία αυτού του δικτύου με τον «έξω κόσμο».

- Την επέκταση του Μοντέλου Στόχων, που αποτελεί εργαλείο της Μηχανικής Απαιτήσεων και το σχεδιασμό του μεταμοντέλου του δέντρου - στόχων που ενσωματώνει αυτές τις επεκτάσεις. Επισημειώνονται επεκτάσεις που αφορούν την πιθανή παραλληλία εκτέλεσης στόχων, τις χρονικές και λογικές εξαρτήσεις μεταξύ κάποιων στόχων καθώς και τη διαθεσιμότητα αυτών λόγω επιδράσεων του περιβάλλοντος ή του χρήστη. Το μεταμοντέλο αυτό είναι επεκτάσιμο και μπορεί να επεκταθεί στο μέλλον αν παραστεί ανάγκη για πιο σύνθετα μοντέλα. Επιπλέον, είναι εύκολα διαχειρίσιμο τόσο από τους υπεύθυνους διαχειριστές, όσο και προγραμματιστικά. Για την κατασκευή του δέντρου στόχων χρησιμοποιήθηκε το EMF, εργαλείο της πλατφόρμας Eclipse για τον σχεδιασμό μοντέλων και μεταμοντέλων, όπου παρέχεται αυτόματη παραγωγή κώδικα και γραφική απεικόνιση του σχεδιαθέντος μεταμοντέλου / μοντέλου.

- Τη δημιουργία αλγορίθμου για την εύρεση όλων των δυνατών ακολουθιών από απλές ενέργειες με γνώμονα την ικανοποίηση του δέντρου - στόχων και του περιβάλλοντος - πλαισίου. Κατά τη διάρκεια υλοποίησης του λογισμικού δημιουργήθηκε ακόμα ένα μεταμοντέλο που έχει ως σκοπό να περιγράψει σε αφαιρετικό επίπεδο ένα Γράφο εξαρτήσεων Ενεργειών[3], όπου τοποθετούνται απλές ενέργειες του συστήματος καθώς και οι συνδέσεις μεταξύ τους. Ως αποτέλεσμα, είναι εύκολο να εξαχθεί η αλληλουχία των ενεργειών που θα εξασφαλίσουν την ορθή λειτουργία του συστήματος.

1.4 Οργάνωση του Κειμένου

Το υπόλοιπο κείμενο έχει οργανωθεί σε 6 κεφάλαια ως εξής:

- Στο κεφάλαιο 2 παρουσιάζεται η υφιστάμενη γνώση, πάνω στην οποία στηρίχθηκε η παρούσα εργασία, και οι τεχνολογίες όπως και τα εργαλεία που χρησιμοποιήθηκαν για την εκπόνησή της. Πιο συγκεκριμένα, θα παρουσιαστεί με ποιο τρόπο γίνεται η μοντελοποίηση συστημάτων λογισμικού, με σημαντικότερο πρότυπο το MOF ενώ θα αναλυθεί η έννοια του μεταμοντέλου καθώς και η χρησιμότητα του. Επιπλέον, θα αναφερθεί η ιδέα των δέντρων - στόχων όπως και ο ορισμός τους. Θα παρουσιαστούν τα εργαλεία που χρησιμοποιήθηκαν, με κυριότερο το EMF ενώ θα γίνει μια ανάλυση της τεχνολογίας του Διαδικτύου των Πραγμάτων.

- Στο κεφάλαιο 3 θα παρουσιαστεί η γενική αρχιτεκτονική του συστήματος και μια εποπτική περιγραφή του βασικού βρόχου εκτέλεσης μέσω ενός διαγράμματος δραστηριοτήτων.

- Στο κεφάλαιο 4 παρουσιάζονται τα διάφορα μεταμοντέλα που εκπονήθηκαν καθώς και το εννοιολογικό τους περιεχόμενο. Επιπλέον, αναλύονται οι επεκτάσεις που πραγματοποιήθηκαν στο μοντέλο στόχων και η συμβολή που έχουν στον εμπλουτισμό της εκφραστικότητας των παραχθέντων μοντέλων - στόχων.

- Στο κεφάλαιο 5 αναλύεται ο αλγόριθμος υπολογισμού των ακολουθιών των ενεργειών βήμα προς βήμα και παρουσιάζονται όλες οι ιδέες και τεχνικές που χρησιμοποιήθηκαν για την κατασκευή του.

- Στο κεφάλαιο 6 θα παρουσιαστούν τα αποτελέσματα και τα συμπεράσματα που εξήχθησαν από τις πειραματικές δοκιμές που πραγματοποιήθηκαν και θα συγκριθούν μεταξύ τους με μετρικές σχετικές με το χρόνο και τη μνήμη, μεταβάλλοντας διάφορα χαρακτηριστικά όπως ο αριθμός των κόμβων του δέντρου - στόχου.

- Στο κεφάλαιο 7 θα γίνει μια ανακεφαλαίωση βασικών σημείων της διπλωματικής, θα αναφερθούν τα τελικά συμπεράσματα και θα γίνουν προτάσεις για μελλοντική έρευνα.

- Τέλος, στο κεφάλαιο 8 θα παρουσιαστεί η βιβλιογραφία που χρησιμοποιήθηκε κατά την εκπόνηση της παρούσας διπλωματικής.

Κεφάλαιο 2: Σχετικές Εργασίες

2.1 Εισαγωγή

Στο κεφάλαιο που ακολουθεί παρουσιάζεται συνοπτικά το σύνολο των τεχνολογιών που χρησιμοποιήθηκαν για την εκπόνηση της παρούσας διπλωματικής καθώς και το θεωρητικό υπόβαθρο που απαιτείται για την κατανόηση της. Το γνωστικό αντικείμενο των θεμάτων που πραγματεύεται αυτό το κεφάλαιο είναι αρκετά ευρύ και ξεπερνά τα πλαίσια αυτής της διπλωματικής. Γι' αυτό, το παρόν κεφάλαιο περιορίζεται μόνο στις βασικές έννοιες ενώ ο αναγνώστης μπορεί να ανατρέξει στις βιβλιογραφικές αναφορές για περισσότερες πληροφορίες.

2.2 Μοντελοποίηση Συστημάτων Λογισμικού

Η Μοντελοκεντρική Αρχιτεκτονική (Model Driven Architecture,MDA) είναι μια σύγχρονη προσέγγιση για τη σχεδίαση και την ανάπτυξη λογισμικού καθώς παρέχει ένα σύνολο κατευθυντήριων γραμμών για τη διάρθρωση ενός συστήματος λογισμικού και πιο συγκεκριμένα στην αναπαράστασή του με τη μορφή μοντέλου. Σκοπός της μοντελοποίησης ενός συστήματος είναι η αναπαράστασή του σε ένα μοντέλο, το οποίο μπορεί να γίνει εύκολα κατανοητό από έναν μηχανικό λογισμικού. Βασίζεται σε κάποιες υπάρχουσες τεχνολογίες, όπως είναι η Unified Modeling Language(UML), το Meta Object Facility(MOF), το XML Metadata Interchange(XMI) τα οποία περιγράφονται στη συνέχεια.

2.2.1 MOF

Η έννοια της μετα-μοντελοποίησης αποτελεί τον θεμέλιο λίθο για τη δημιουργία και την επεξεργασία γλωσσών μοντελοποίησης και των μοντέλων τους. Το Meta Object Facility[13][14] προτάθηκε από τον οργανισμό Object Management Group(OMG) ως ένα πρότυπο μοντελοποίησης συστημάτων. Είναι μια αφηρημένη γλώσσα που προσφέρει το απαραίτητο περιβάλλον - πλαίσιο για τη σχεδίαση, ανάπτυξη και διαχείριση μεταμοντέλων. Επιπλέον, δίνει τη δυνατότητα διαλειτουργικότητας μεταξύ διαφορετικών ειδών προϊόντων λογισμικού μέσω του ανεξάρτητου πλαισίου διαχείρισης μεταδεδομένων που παρέχεται. Η αρχιτεκτονική που ορίζεται στο πρότυπο Meta Object Facility χωρίζεται σε τέσσερα επίπεδα[15][16] και περιγράφονται παρακάτω:

1. Το επίπεδο M3 ή αλλιώς επίπεδο μέτα-μεταμοντέλου (meta-metamodel layer). Αποτελεί το ανώτερο επίπεδο της ιεραρχίας και η χρησιμότητά του έγκειται στην περιγραφή μοντέλων του επιπέδου M2, δηλαδή μεταμοντέλων(όπως η UML). Τα μέτα-μεταμοντέλα που βρίσκονται σε αυτό το επίπεδο είναι ορισμοί γλωσσών που χρησιμοποιούνται στην

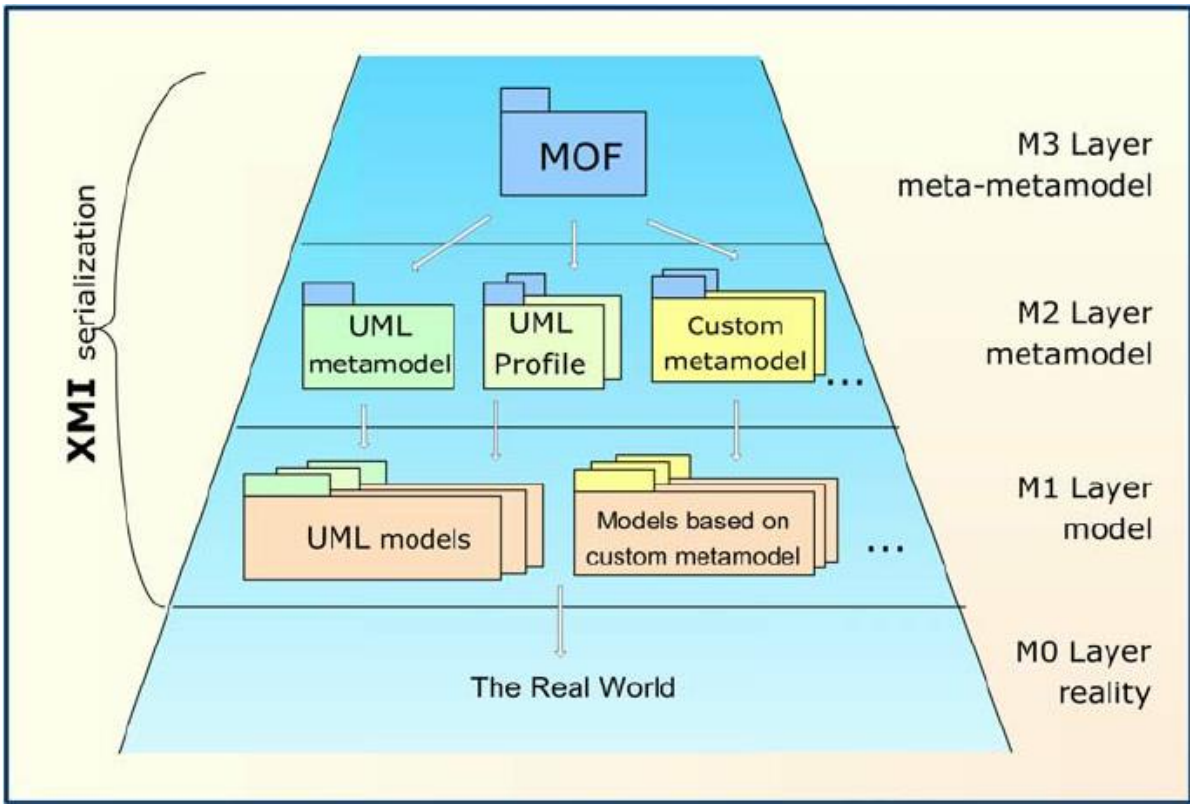
προδιαγραφή μεταμοντέλων και είναι MOF μοντέλα. Ο ορισμός των μέτα-μεταμοντέλων γίνεται χρησιμοποιώντας μέτα-μεταμοντέλα, δηλαδή τα στοιχεία αυτού του επιπέδου ορίζονται αυτοαναφορικά(το MOF είναι ορισμένο με χρήση του ίδιου του MOF).

2. Το επίπεδο M2 ή αλλιώς επίπεδο μεταμοντέλου(meta-model layer). Στοιχεία αυτού του επιπέδου είναι τα μεταμοντέλα, δηλαδή γλώσσες μοντελοποίησης που χρησιμοποιούνται για τον ορισμό μοντέλων του επιπέδου M1. Τα μεταμοντέλα υπακούουν σε κάποιο μέτα-μεταμοντέλο. Χαρακτηριστικό παράδειγμα μεταμοντέλου είναι το UML μεταμοντέλο, το οποίο αποτελεί στιγμιότυπο του MOF.

3. Το επίπεδο M1 ή αλλιώς επίπεδο μοντέλου (model layer). Στοιχεία αυτού του επιπέδου είναι τα μοντέλα, τα οποία αποτελούν στιγμιότυπα μεταμοντέλου του επιπέδου M2. Χαρακτηριστικό παράδειγμα μοντέλου είναι τα μοντέλα λογισμικών συστημάτων που περιγράφονται με UML μοντέλα (ακολουθιακά διαγράμματα, διαγράμματα κλάσεων κτλ).

4. Το επίπεδο M0 ή αλλιώς επίπεδο πληροφορίας (information layer) . Σε αυτό το επίπεδο βρίσκονται όλα τα δεδομένα και αντικείμενα που θέλουμε να μοντελοποιήσουμε. Τα στοιχεία αυτού του επιπέδου είναι ένα στιγμιότυπο του μοντέλου του επιπέδου M1.

Στο παρακάτω σχήμα φαίνεται η αρχιτεκτονική των τεσσάρων επιπέδων που περιγράφηκε.



Σχήμα 2.1

Η παραπάνω αρχιτεκτονική δεν είναι απόλυτη. Το MOF έχει σχεδιαστεί ώστε να μπορεί να χρησιμοποιηθεί σε αρχιτεκτονικές μοντελοποίησης με μεγαλύτερο αριθμό επιπέδων. Τέλος, το MOF επιτρέπει την αποθήκευση των μοντέλων σε άλλες μορφές όπως το XMI και έτσι παρέχει πλεονεκτήματα όπως είναι η μεταφορά σε άλλη εφαρμογή και η δυνατότητα επεκτασιμότητας μεταγενέστερα, πιθανόν από διαφορετικό σχεδιαστή.

2.2.2 XMI

Το XMI Metadata Interchange (XMI) είναι ένα πρότυπο της OMG που έχει δημιουργηθεί και πιστοποιηθεί κατά ISO για την ανταλλαγή μεταδεδομένων μέσω της eXtensible Markup Language(XML). Χρησιμοποιείται για μεταδεδομένα όπου το μεταμοντέλο του αναπαρίσταται στο Meta Object Facility (MOF). Αποτελεί, δηλαδή, το μοντέλο XMI ένα στιγμιότυπο ενός μοντέλου MOF. Συνηθέστερη χρήση της XMI είναι η ανταλλαγή UML μοντέλων και η σειριοποίηση μεταμοντέλων.

Το OMG ορίζει δύο είδη μοντέλων, τα αφηρημένα (abstract models) και τα συγκεκριμένα (concrete models). Τα αφηρημένα μοντέλα χρησιμοποιούνται για να περιγράψουν τη σημασιολογία της πληροφορίας και τα συγκεκριμένα μοντέλα αναπαριστούν εποπτικά διαγράμματα. Χαρακτηριστικά παραδείγματα αφηρημένων μοντέλων είναι οι γλώσσες μοντελοποίησης UML και SysML [17], που βασίζονται στο πρότυπο MOF. Για την ανταλλαγή εποπτικών διαγραμμάτων χρησιμοποιείται η τυποποίηση ανταλλαξιμότητας διαγραμμάτων ή αλλιώς XMIDI.

Μερικά από τα πλεονεκτήματα της μορφής XMI είναι[18]:

- Αυτοματοποιείται η παραγωγή κώδικα με τη χρήση UML με συνέπεια αλλαγές που γίνονται στο μοντέλο XMI να διαδίδονται αυτόματα στον κώδικα του λογισμικού.
- Διευκολύνεται η ανταλλαγή μεταδεδομένων μεταξύ εργαλείων μοντελοποίησης που βασίζονται στην UML και αποθήκες μεταδεδομένων μεταξύ ετερογενών υπολογιστικών συστημάτων.

2.2.3 Eclipse Modelling Framework

Το λογισμικό που αναπτύχθηκε κατά την εκπόνηση της παρούσας διπλωματικής βασίζεται στο αντικειμενοστραφές μοντέλο προγραμματισμού. Η μοντελοποίηση τέτοιου είδους συστημάτων απαιτούν εργαλεία όπως είναι η UML, που προσφέρουν τη δυνατότητα μιας καθολικής και αφηρημένης απεικόνισης για το σύστημα λογισμικού. Ένα πολύ χρήσιμο εργαλείο για την μοντελοποίηση συστημάτων σε UML είναι το EMF (Eclipse Modeling Framework). Το Eclipse Modelling Framework[19][20] αποτελεί προσάρτημα του περιβάλλοντος λογισμικού Eclipse. Διευκολύνει σημαντικά τη σχεδίαση και την ανάπτυξη λογισμικού που βασίζεται στη

μοντελοκεντρική αρχιτεκτονική και δίνει τη δυνατότητα της αυτόματης μετατροπής των μοντέλων σε κώδικα. Προσφέρει τη δυνατότητα στον χρήστη να σχεδιάσει μοντέλα που ακολουθούν τις προδιαγραφές του προτύπου MOF και περιγράφουν ένα σύστημα λογισμικού[21]. Τα μοντέλα που σχεδιάζονται μπορούν να μετατραπούν με αυτόματο τρόπο σε πηγαίο κώδικα και να γίνουν αλλαγές που ενσωματώνονται στον αναπτυσσόμενο κώδικα. Για παράδειγμα, ο χρήστης μπορεί να προσθέσει νέες κλάσεις, μεθόδους και μεταβλητές και αυτόματα το EMF παράγει τον αντίστοιχο πηγαίο κώδικα. Αλλαγές στο μοντέλο διαδίδονται, επίσης αυτόματα στον πηγαίο κώδικα. Μια άλλη σημαντική δυνατότητα του EMF είναι η δυνατότητα παραγωγής στιγμιότυπων των συστημάτων που έχουν αναπαρασταθεί ως MOF μοντέλα και τα οποία μπορούν να αποθηκευτούν σε μορφή XMI και να χρησιμοποιηθούν από τον παραγόμενο κώδικα του μοντέλου.

Στα πλαίσια αυτής της διπλωματικής, το EMF χρησιμοποιήθηκε για τη μοντελοποίηση της δομής του Διαδικτύου των Πραγμάτων, του Μοντέλου Δένδρων - Στόχων και του Μοντέλου Γράφων Εξάρτησης Ενεργειών. Για όλα τα παραπάνω, δημιουργήθηκαν με βάση το MOF τα απαραίτητα μεταμοντέλα και στη συνέχεια χρησιμοποιήθηκε ο παραγόμενος κώδικας για τον ορισμό και τη διαχείριση των δέντρων - στόχων και των γράφων.

2.3 Δέντρα-Στόχων

Ένας σημαντικός κλάδος της Τεχνολογίας Λογισμικού είναι η Μηχανική Απαιτήσεων. Ο κλάδος αυτός ασχολείται με τη συλλογή και την ανάλυση απαιτήσεων, χαρακτηριστικά πολύ σημαντικά κατά την διαδικασία ανάπτυξης του λογισμικού. Ασαφείς και ελλιπείς απαιτήσεις του συστήματος λογισμικού έχει σαν αποτέλεσμα μείωση της ποιότητας του παραγόμενου λογισμικού, αύξηση του κόστους και του χρόνου παραγωγής. Επομένως, είναι κατανοητό πόσο σημαντικό είναι αυτό το στάδιο ανάπτυξης του λογισμικού.

Η Μηχανική Απαιτήσεων προσανατολισμένη σε στόχους αποτελεί κυρίαρχη προσέγγιση σε αυτόν τον κλάδο. Τα πλεονεκτήματα αυτής αναλύονται διεξοδικά στα [22] και [23]. Συνέπεια της ευρείας χρήσης της παραπάνω μεθόδου είναι η ανάπτυξη πολλαπλών μεθοδολογιών για την προσανατολισμένη σε στόχους Μηχανική Απαιτήσεων. Μερικά από τα πιο δημοφιλή παραδείγματα είναι: KAOS[24], GRL[25], GBRAM[26], TROPOS[27], CREWS[28], i*[29].

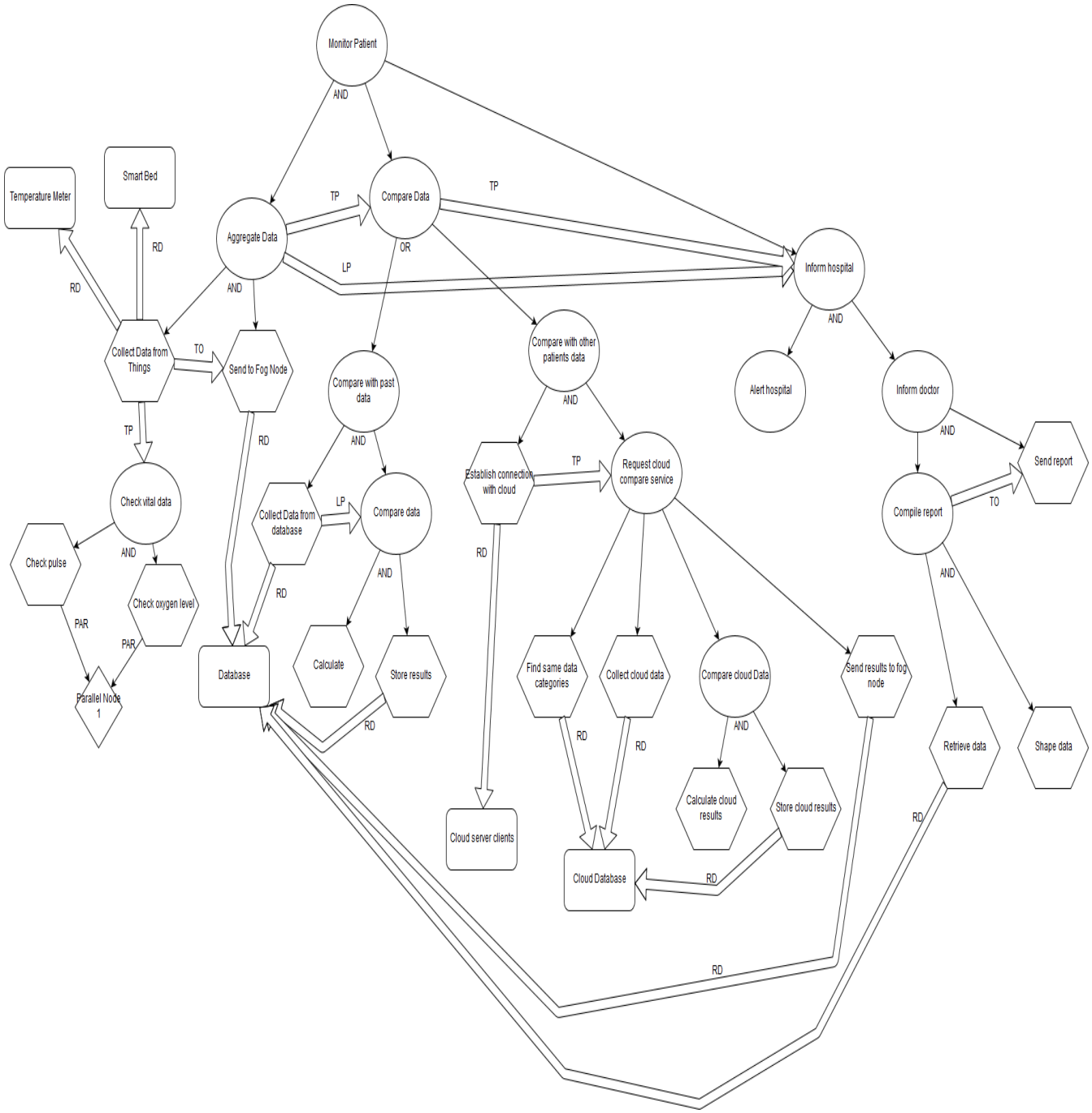
Σημαντικό μέρος ης Μηχανικής Απαιτήσεων αποτελεί η μοντελοποίηση των απαιτήσεων. Γι' αυτό το λόγο έχουν προταθεί πολυάριθμες μέθοδοι για τη μοντελοποίηση των λειτουργικών και μη λειτουργικών απαιτήσεων ενός συστήματος λογισμικού ενώ έχουν αναπτυχθεί αρκετά εργαλεία που στηρίζονται στις μεθόδους αυτές. Εμφανίστηκαν πρώτη φορά στο χώρο της Μηχανικής Απαιτήσεων στο [30].

Τα δέντρα - στόχων μοντελοποιούν τους στόχους σε μια δενδρική δομή, όπου ο κάθε κόμβος του δέντρου αποτελεί στόχος(goal). Κάθε στόχος μπορεί να αποτελεί στόχος κάποιου άλλου στόχου στο δέντρο, δηλαδή είναι υποστόχος (subgoal). Οι υποστόχοι συνθέτουν κάποιον πιο σύνθετο στόχο, μέσω λογικής σύζευξης (ΚΑΙ/AND) ή διάζευξης(Ή/OR) και η ικανοποίηση τους ή μη επηρεάζει την ικανοποίηση ή μη του στόχου - πατέρα. Πιο συγκεκριμένα, στην περίπτωση της σύζευξης για να ικανοποιηθεί ο στόχος - πατέρας απαιτείται η ικανοποίηση όλων των στόχων - παιδιών. Αντίστοιχα, για τη διάζευξη απαιτείται η ικανοποίηση τουλάχιστον ενός από τους στόχους - παιδιά για την ικανοποίηση του στόχου - πατέρα.

Επιπλέον αυτών των χαρακτηριστικών έχουν αναπτυχθεί επιπρόσθετες επεκτάσεις που αφορούν πράκτορες (Agents), ρόλους (Roles), δεσμεύσεις (Commitments) μεταξύ πρακτόρων και συνεισφορές (Contributions) μεταξύ κόμβων, λογικές και χρονικές προϋποθέσεις, τα οποία παρουσιάζονται αναλυτικά στα [31],[32],[33],[2],[9].

Στην παρούσα διπλωματική χρησιμοποιήθηκε η επέκταση για τις συνεισφορές μεταξύ κόμβων αλλά και άλλες που αναπτύχθηκαν για την εκπόνηση της παρούσας διπλωματικής όπως είναι η παραλληλία μεταξύ κόμβων και η χρονική διαφορά εκτέλεσης.

Ακολουθεί ένα πλήρες παράδειγμα δέντρου - στόχων με όλα τα παραπάνω χαρακτηριστικά:



Σχήμα 2.2

Στην παρούσα διπλωματική χρησιμοποιήθηκαν τα δέντρα - στόχων για να μοντελοποιηθούν οι στόχοι που πρέπει να εκπληρωθούν σε ένα σύστημα για την εύρυθμη λειτουργία του. Θα πρέπει, δηλαδή, να ικανοποιηθεί ο κόμβος - ρίζα του δέντρου για να λειτουργήσει το σύστημα όπως έχει καθοριστεί. Το λογισμικό που έχει αναπτυχθεί μπορεί να εφαρμοστεί όχι μόνο για το Διαδίκτυο των Πραγμάτων(όπως στην παρούσα διπλωματική) αλλά για οποιαδήποτε δομή συστήματος.

2.4 Προσωποποιημένο Περιβάλλον – Πλαίσιο (Personalised Web Tasking)

Το προσωποποιημένο πλαίσιο διαδικτυακών εργασιών είναι η αυτοματοποίηση απλών και επαναλαμβανόμενων διαδικτυακών αλληλεπιδράσεων που σχετίζονται με το χρήστη και τις επιθυμίες του, λαμβάνοντας υπόψη το περιβάλλον - πλαίσιο στο οποίο εκτελείται η εφαρμογή για να βελτιώσει και να διευκολύνει την εμπειρία του χρήστη[34][35].

Η γνώση του περιβάλλοντος - πλαισίου είναι θεμελιώδης για τέτοιου είδους εφαρμογές. Τέτοια γνώση είναι η κατανόηση του περιβάλλοντος εκτέλεσης της εφαρμογής καθώς και των επιθυμιών του χρήστη. Στις εφαρμογές προσωποποιημένου πλαισίου διαδικτυακών εφαρμογών υπάρχουν 2 βασικοί στόχοι σχετικά με τη γνώση του περιβάλλοντος - πλαισίου:

1. Η εγγύηση ότι το λογισμικό θα αυτοπροσαρμόζεται στο περιβάλλον - πλαίσιο ανάλογα με τις αλλαγές που προκύπτουν[36][37][38].
2. Να εκμεταλλευτεί τις επιθυμίες του χρήστη για καλύτερη προσαρμογή σε αυτόν κατά την διάρκεια εκτέλεσης του λογισμικού.

Υπάρχουν αρκετές μέθοδοι για την εύρεση αυτής της γνώσης. Ένα χαρακτηριστικό παράδειγμα είναι οι πληροφορίες για τις επιθυμίες του χρήστη που μπορούν να αντληθούν από τα κοινωνικά δίκτυα[39].

Στην παρούσα διπλωματική ανάλογα με την γνώση του περιβάλλοντος - πλαισίου και των επιλογών του χρήστη καθορίζεται ποιοι κόμβοι - στόχοι θα είναι διαθέσιμοι στο δέντρο - στόχων για την εύρεση των πλάνων ατομικών ενεργειών για την αναμενόμενη λειτουργία του συστήματος.

2.5 Διαδίκτυο των Πραγμάτων (Internet of Things)

Το Διαδίκτυο των Πραγμάτων είναι μια καινοτομία με ραγδαία ανάπτυξη τα τελευταία χρόνια που έχει επιφέρει μεγάλες αλλαγές στον τομέα του Λογισμικού και της Πληροφορίας. Σκοπός του Διαδικτύου των Πραγμάτων είναι η διασύνδεση όσο το δυνατόν περισσότερων αντικειμένων του πραγματικού κόσμου με το Διαδίκτυο για να επικοινωνούν μεταξύ τους και με συστήματα λογισμικού, χωρίς την ανθρώπινη παρέμβαση[40]. Αρχικός στόχος αυτής της καινοτομίας ήταν να μειωθούν τα δεδομένα που εισάγονται από ανθρώπινο παράγοντα σε διάφορα συστήματα και την αυτοματοποίηση συλλογής και εισαγωγή δεδομένων από διάφορους αισθητήρες[41][42].

Εξαιτίας της μειωμένης απόδοσης των συσκευών IoT σε ενέργεια και αποθηκευτικό χώρο, προκύπτουν προβλήματα απόδοσης, ασφάλειας, ιδιωτικότητας και αξιοπιστίας. Η ενσωμάτωση του Διαδικτύου των Πραγμάτων με το Cloud, επονομαζόμενο Cloud of Things, λύνει σε ένα βαθμό τα παραπάνω προβλήματα[43]. Το Cloud of Things απλοποιεί τη

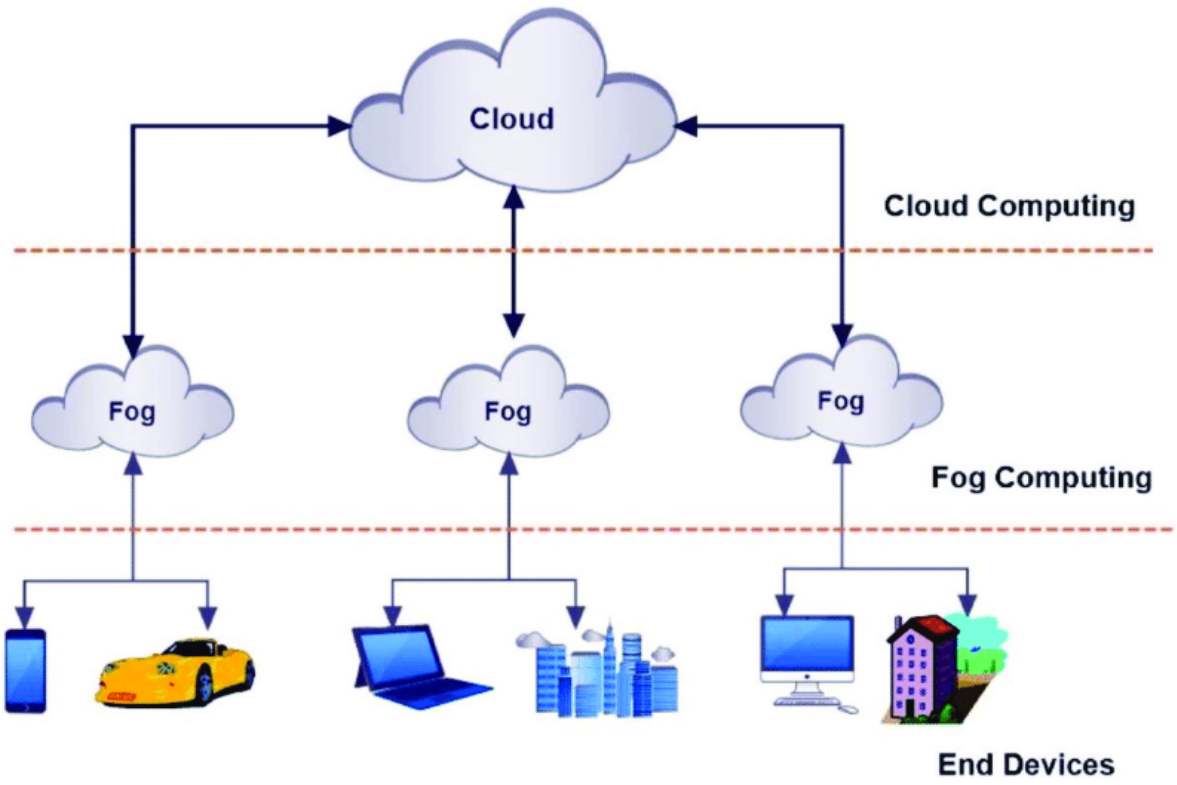
συλλογή και επεξεργασία δεδομένων που δημιουργούνται από το Διαδίκτυο των Πραγμάτων και προσφέρει γρήγορη, χαμηλή σε κόστος και εύκολη επεξεργασία μεγάλων και πολύπλοκων δεδομένων[43][44].

Η ενσωμάτωση του Διαδικτύου των Πραγμάτων με το Cloud έχει πολυάριθμα πλεονεκτήματα σε πολλές εφαρμογές. Παρόλα αυτά, ο μεγάλος αριθμός IoT συσκευών με ετερογενή συστήματα καθιστά την ανάπτυξη τέτοιου είδους εφαρμογών δύσκολη έως αδύνατη[44]. Αυτό συμβαίνει γιατί εφαρμογές του Διαδικτύου των Πραγμάτων παράγουν τεράστιο όγκο δεδομένων από αισθητήρες και άλλες συσκευές. Αυτά τα δεδομένα, στη συνέχεια, αναλύονται για να καθοριστούν οι ενέργειες που θα εκτελέσει το σύστημα. Στέλνοντας αυτόν τον τεράστιο όγκο δεδομένων στο Cloud απαιτείται μεγάλο δικτυακό εύρος ζώνης[44]. Έτσι, αναπτύχθηκε η έννοια του Fog Computing.

Ο όρος Fog Computing ορίστηκε πρώτη φορά από τη Cisco[45]. Είναι μια νέα τεχνολογία που προσφέρει πολλά πλεονεκτήματα σε διάφορους τομείς, ιδιαίτερα στον κλάδο του Διαδικτύου των Πραγμάτων. Παρόμοια με το Cloud, το Fog Computing προσφέρει υπηρεσίες στο Διαδίκτυο των Πραγμάτων, όπως είναι η επεξεργασία των δεδομένων και η αποθήκευση αυτών. Το Fog Computing στηρίζεται στην ιδέα του υπολογισμού κοντά στην πηγή αντί να στέλνονται τα δεδομένα προς επεξεργασία στο Cloud, το οποίο επιφέρει κόστος τόσο σε εύρος ζώνης όσο και σε αποκρισιμότητα.

Σκοπός του Fog Computing στο Διαδίκτυο των Πραγμάτων είναι να βελτιώσει την αποδοτικότητα, την απόδοση και να μειώσει την όγκο των δεδομένων που μεταφέρονται στο Cloud. Τα δεδομένα που συλλέγονται παραμένουν σε τοπικούς κόμβους ή συσκευές που βρίσκονται στην άκρη του δικτύου για επεξεργασία και ανάλυση. Ως αποτέλεσμα υπάρχουν μεγάλες μειώσεις στο χρόνο απόκρισης και στην κίνηση του δικτύου.

Παρακάτω παρουσιάζεται ένα σχήμα με την αφηρημένη αρχιτεκτονική που περιγράφτηκε παραπάνω.



Σχήμα 2.3

Σημαντικό χαρακτηριστικό του Fog Computing αποτελεί η κλιμακωσιμότητα, η απόκριση πραγματικού χρόνου και η διαλειτουργικότητα[44][46], χαρακτηριστικά πολύ σημαντικά για εφαρμογές Διαδικτύου των Πραγμάτων.

Στην παρούσα διπλωματική αναπτύσσεται ένα μεταμοντέλο της δομής του Διαδικτύου των Πραγμάτων, όπου έχει ενσωματωθεί και η τεχνολογία του Fog Computing, για καλυφθούν οι σύγχρονες ανάγκες μιας εφαρμογής Διαδικτύου των Πραγμάτων, με το μεγάλο όγκο δεδομένων που τη συνοδεύει, και διευκολύνει σε μεγάλο βαθμό τη μοντελοποίηση τους.

Κεφάλαιο 3: Αρχιτεκτονική του Συστήματος

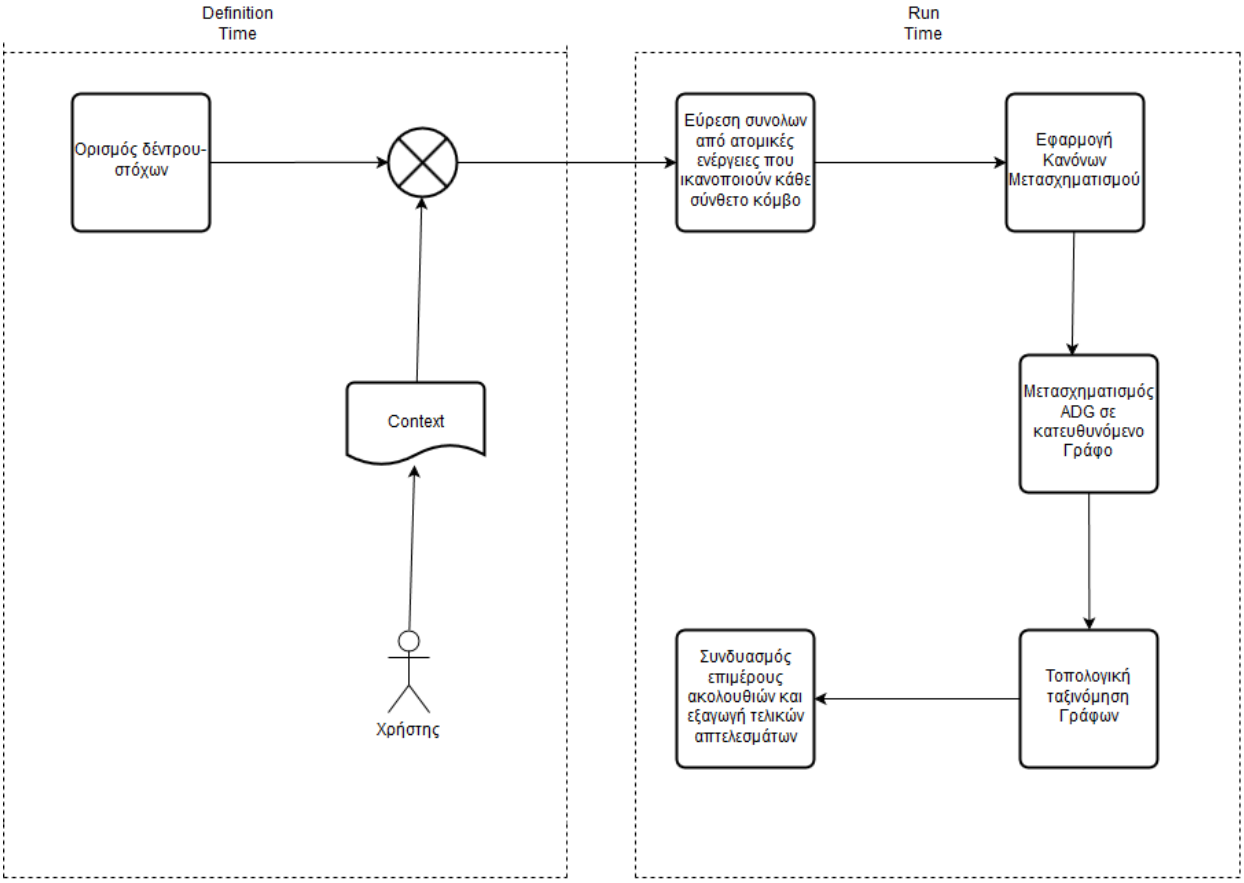
3.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζουμε την αρχιτεκτονική του προτεινόμενου συστήματος. Αναφέρονται όλα τα δομικά συστατικά του συστήματος, αναλύονται οι λειτουργικότητες τους και ο τρόπος διασύνδεσης μεταξύ τους. Για την ανάλυση αυτή χρησιμοποιούνται ψηφιδικά διαγράμματα (component diagrams).

Κεντρικό ζήτημα κατά την σχεδίαση και ανάπτυξη ενός συστήματος λογισμικού είναι η επεκτασιμότητα και η ευκολία προσαρμογής σε νέες, μελλοντικές ανάγκες, διαφορετικές από αυτές που λήφθηκαν υπόψη κατά την αρχική σχεδίαση του. Νέες λειτουργικότητες προστίθενται κατά τη διάρκεια ζωής του συστήματος λογισμικού αφού οι απαιτήσεις του περιβάλλοντος στο οποίο εκτελείται και οι ανάγκες των χρηστών διαρκώς μεταβάλλονται. Επομένως, είναι απαραίτητο το προτεινόμενο σύστημα να υποστηρίζει μελλοντικές αλλαγές και επεκτάσεις. Προς την κατεύθυνση αυτή, λοιπόν, κατά τη σχεδίαση του παρόντος συστήματος αναγνωρίστηκαν όλες εκείνες οι λειτουργίες και υπο-λειτουργίες που απαιτούνται για την επίλυση του προβλήματος ενώ ανατέθηκαν αυτές σε διάφορα components. Η λειτουργία του κάθε component είναι ανεξάρτητη από τα υπόλοιπα components ενώ η διασύνδεση μεταξύ τους πραγματοποιείται με διαπροσωπίες που έχουν καθοριστεί κατά την ανάπτυξη.

Βασική έννοια για την περιγραφή του συστήματος είναι αυτή του Fog Computing[12]. Το Fog αποτελεί ένα επιπλέον στρώμα μεταξύ του Cloud και της άκρης του δικτύου για την καλύτερη διαχείριση των πόρων και των δεδομένων. Με αυτόν τον τρόπο μεταφέρεται η επεξεργασία των δεδομένων κοντά στην πηγή με χαμηλότερες και πιο προβλέψιμες καθυστερήσεις. Βασίζεται στις ίδιες αρχές που διέπουν και το Cloud Computing και ουσιαστικά οι κόμβοι του Fog(Fog Nodes) λειτουργούν σαν επιμέρους κέντρα δεδομένων που βρίσκονται κοντά στην πηγή. Κάθε τέτοιος κόμβος συντονίζει τη ροή των δεδομένων, κάνει υπολογισμούς, κατανέμει την επεξεργασία των δεδομένων και καθορίζει ποια από αυτά πρέπει να σταλθούν στο Cloud για υπολογισμό. Στο παρόν σύστημα, ο Fog Node εισάγει νέα δεδομένα με την μορφή δέντρου - στόχου, καθορίζει το περιβάλλον – πλαίσιο (context) μέσα στο οποίο ορίζεται το δέντρο - στόχων και ενορχηστρώνει την εύρεση των πλάνων από ατομικές ενέργειες.

Μια άλλη σημαντική έννοια είναι το context. Το context καθορίζεται τόσο από το περιβάλλον στο οποίο εκτελείται το σύστημα όσο και από τις απαιτήσεις του χρήστη. Αυτό επηρεάζει το δέντρο - στόχων που έχει καθοριστεί και τα παραγόμενα πλάνα για την ικανοποίηση αυτού. Το παρακάτω διάγραμμα απεικονίζει τις βασικές λειτουργίες που εκτελούνται στο σύστημα.



Σχήμα 3.1 - Περίγραμμα λειτουργιών

Από τις παραπάνω λειτουργίες είναι σαφές ότι χρειάζονται τα εξής components:

- Ένα component για τον καθορισμό του δέντρου - στόχων
- Ένα component για τον καθορισμό και την διαχείριση του context
- Ένα component για την εύρεση όλων εκείνων των συνόλων από ατομικές ενέργειες που ικανοποιούν κάθε σύνθετο στόχο(Task)
- Ένα component για την εφαρμογή των κανόνων
- Ένα component για το μετασχηματισμό των αρχικών, ακατέργαστων ADG σε κατευθυνόμενους γράφους
- Ένα component που θα ταξινομεί τοπολογικά τους γράφους
- Ένα component για το συνδυασμό όλων των επιμέρους αλληλουχιών και την εξαγωγή των τελικών αλληλουχιών
- Ένα κεντρικό component που θα είναι ο Fog Node που θα διαχειρίζεται όλες τις παραπάνω λειτουργίες

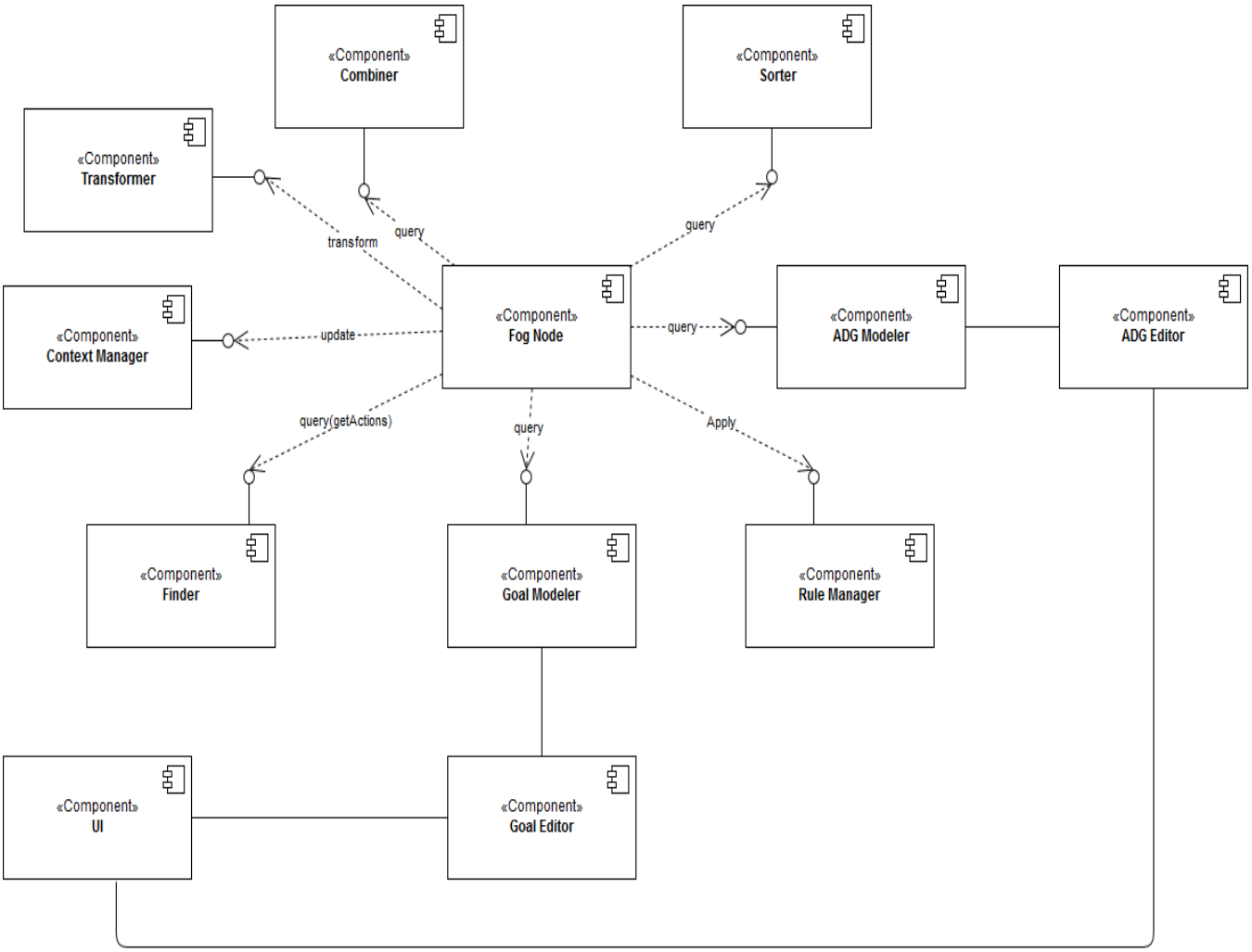
Τα παραπάνω αποτελούν τα βασικά components του προτεινόμενου συστήματος και είναι θεμελιώδη για την εύρεση όλων εκείνων των πλάνων από ατομικές ενέργειες που ικανοποιούν το μοντέλο.

3.2 Επισκόπηση της Αρχιτεκτονικής

Στην παρούσα ενότητα παρουσιάζεται αναλυτικά η αρχιτεκτονική του προτεινόμενου συστήματος για την εύρεση των πλάνων που ικανοποιούν το μοντέλο - στόχων. Περιλαμβάνονται τόσο τα βασικά components που αναφέρθηκαν παραπάνω, όσο και κάποια επιπρόσθετα components που το επεκτείνουν.

Η αρχιτεκτονική που προτείνεται βασίζεται στον Fog Node, που αποτελεί το κύριο component του συστήματος. Ο Fog Node προσφέρει τις υπηρεσίες στο χρήστη, διαχειρίζεται τα δεδομένα που παράγονται και ενορχηστρώνει το σύστημα για την διεκπεραίωση των λειτουργιών. Όλα τα components του συστήματος συνδέονται και επικοινωνούν με τον Fog Node, ο οποίος διαχειρίζεται τόσο τους γράφους όσο και τις παραγόμενες αλληλουχίες από ατομικές ενέργειες.

Στη συνέχεια, ακολουθεί το ψηφιδικό διάγραμμα του συστήματος που περιλαμβάνει όλα τα components και τη διασύνδεση μεταξύ τους.



Σχήμα 3.2

Όπως φαίνεται από το παραπάνω διάγραμμα η αρχιτεκτονική αποτελείται από τα εξής components:

- Fog Node: Όπως αναφέρθηκε και παραπάνω, είναι το κεντρικό δομικό στοιχείο της αρχιτεκτονικής γύρω από το οποίο περιστρέφονται όλες οι λειτουργίες που πρέπει να εκτελεστούν. Διαχειρίζεται το δέντρο - στόχων και ενορχηστρώνει τα υπόλοιπα components για την εύρεση των πλάνων.

- Goal Modeler: Αυτό το component είναι υπεύθυνο για την αναπαράσταση του μεταμοντέλου στο οποίο βασίζεται η μοντελοποίηση των δέντρων - στόχων.

- Goal Editor: Είναι το component που χρησιμοποιείται για τον ορισμό νέων δέντρων - στόχων. Κάθε νέο μοντέλο που δημιουργείται πρέπει να ακολουθεί τις προδιαγραφές που έχουν καθοριστεί στο μεταμοντέλο τόσο με τους κόμβους που περιέχει όσο και με τα χαρακτηριστικά του (π.χ. συνδέσεις).

- UI: Χρησιμοποιείται για την αλληλεπίδραση του χρήστη με το μοντέλο στόχων του συστήματος. Στο παρουσιασθέν σύστημα ως UI χρησιμοποιήθηκε το πλαίσιο Eclipse Modelling Framework (EMF), το οποίο έχει παρουσιαστεί αναλυτικά στο Κεφάλαιο 2. Το EMF δίνει στο χρήστη μια εποπτική εικόνα του μοντέλου στο χρήστη και μπορεί με ευκολία να το τροποποιήσει. Τέλος, προσφέρει τη δυνατότητα παραγωγής μεθόδων που διευκολύνουν τη διαχείριση του.

- ADG Modeler: Αυτό το component είναι υπεύθυνο για την αναπαράσταση του μεταμοντέλου στο οποίο βασίζεται η μοντελοποίηση των Γράφων Εξάρτησης Ενεργειών(ADG).

- ADG Editor: Χρησιμοποιείται για τον ορισμό των Γράφων και κάθε νέο μοντέλο Γράφου που κατασκευάζεται ακολουθεί τις προδιαγραφές που έχουν καθοριστεί στο μεταμοντέλο.

- Rule Manager: Το component αυτό διαχειρίζεται και εφαρμόζει τους κανόνες Μετασχηματισμού σε ADG. Εξετάζει για κάθε κανόνα τις αντίστοιχες συνδέσεις και διαμορφώνει τα ADG, τα οποία τοποθετούνται σε συλλογές ανάλογα με το αν οι κόμβοι του ADG εκτελούνται παράλληλα.

- Finder: Το component αυτό έχει σαν σκοπό την εύρεση όλων των συνόλων από ατομικές ενέργειες που ικανοποιούν κάθε σύνθετο στόχο (Task).

- Context Manager: Είναι το component που καθορίζει αν κάποιος κόμβος του δέντρου ή κάποια πηγή δεδομένων θα είναι διαθέσιμη ανάλογα με το περιβάλλον και τις επιθυμίες του χρήστη. Στη συνέχεια, ανάλογα με το context καθορίζονται και οι αλληλουχίες που ικανοποιούν το δέντρο - στόχων.

- Sorter: Το component αυτό χρησιμοποιείται για την τοπολογική ταξινόμηση των κατευθυνόμενων γράφων, κάτι που είναι απαραίτητο για την εξαγωγή των αλληλουχιών από ατομικές ενέργειες.

- Combiner: Αυτό το component είναι υπεύθυνο για το συνδυασμό των παραγόμενων αλληλουχιών από τις διάφορες συλλογές γράφων και την εξαγωγή των τελικών αλληλουχιών από ατομικές ενέργειες που ικανοποιούν το δέντρο - στόχων.

- Transformer: Είναι το component που μετασχηματίζει τα σύνολα που περιέχουν ατομικές ενέργειες και συνδέσεις σε κατευθυνόμενους γράφους, με σκοπό τον υπολογισμό των αλληλουχιών από ατομικές ενέργειες.

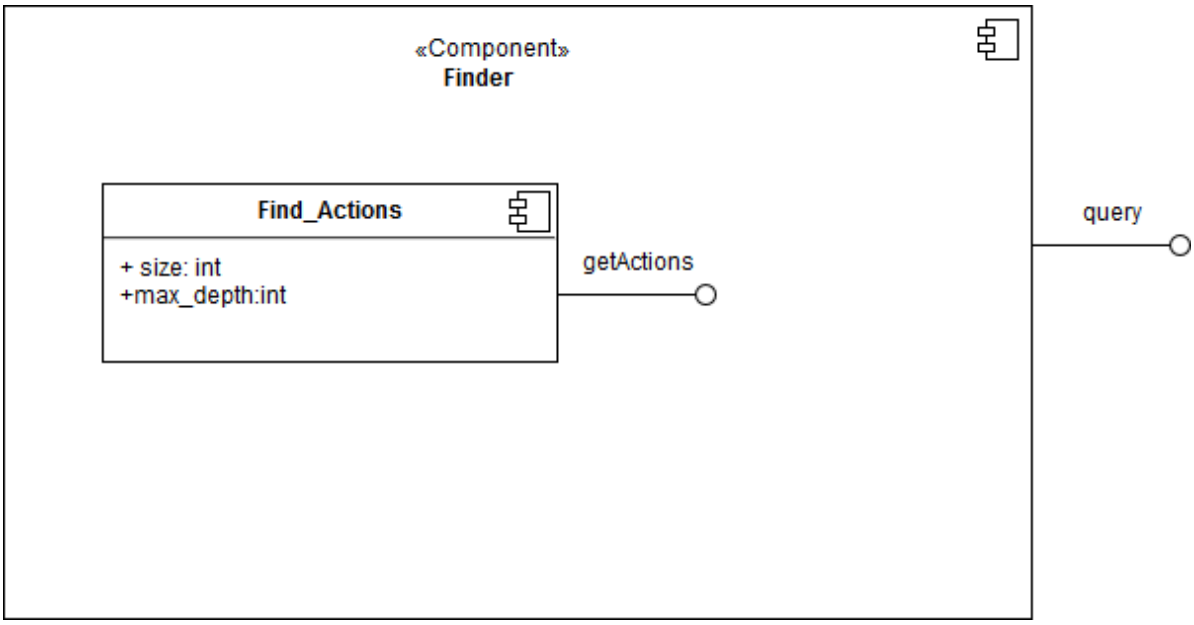
Το EMF αποτελεί καλή λύση σε επίπεδο υλοποίησης για τον ορισμό των components Goal Modeler, ADG Modeler και UI και αυτό χρησιμοποιήθηκε στο πρωτότυπο που αναπτύχθηκε.

3.3 Αναλυτική Περιγραφή των Components

Σε αυτή την ενότητα θα εξεταστούν αναλυτικά τα components που παρουσιάστηκαν στην προηγούμενη ενότητα. Θα χρησιμοποιηθούν UML διαγράμματα για να την αναπαράσταση των components και της εσωτερικής τους δομής. Τέλος, θα παρουσιαστούν οι διεπαφές τους και οι αλληλεπιδράσεις με άλλα components.

3.3.1 Finder Component

Το component Finder αναλαμβάνει να βρει όλα τα σύνολα (άτακτα στο εσωτερικό τους) που ικανοποιούν κάθε σύνθετο στόχο. Τα σύνολα αυτά προκύπτουν από συνδυασμούς ατομικών ενεργειών που ικανοποιούν τους κόμβους - παιδιά και κατ' επέκταση το σύνθετο στόχο. Ο υπολογισμός των συνόλων ξεκινάει από τους σύνθετους στόχους που βρίσκονται στο χαμηλότερο επίπεδο του δέντρου, για να είναι δυνατή η εύρεση των αντίστοιχων συνόλων από σύνθετους στόχους που βρίσκονται ψηλότερα στο δέντρο - στόχων και οι οποίοι εξαρτώνται από την ικανοποίηση των χαμηλότερων στόχων. Επιπλέον, λαμβάνονται υπόψη μόνο κόμβοι του δέντρου που είναι διαθέσιμοι εξαιτίας του context και υπολογίζεται η περίπτωση που ο κόμβος - ρίζα του δέντρου δεν ικανοποιείται, όταν δεν υπάρχει, δηλαδή, αλληλουχία ατομικών ενεργειών για την ικανοποίηση του μοντέλου.
Τα παραπάνω φαίνονται στο ψηφιδικό διάγραμμα που ακολουθεί:

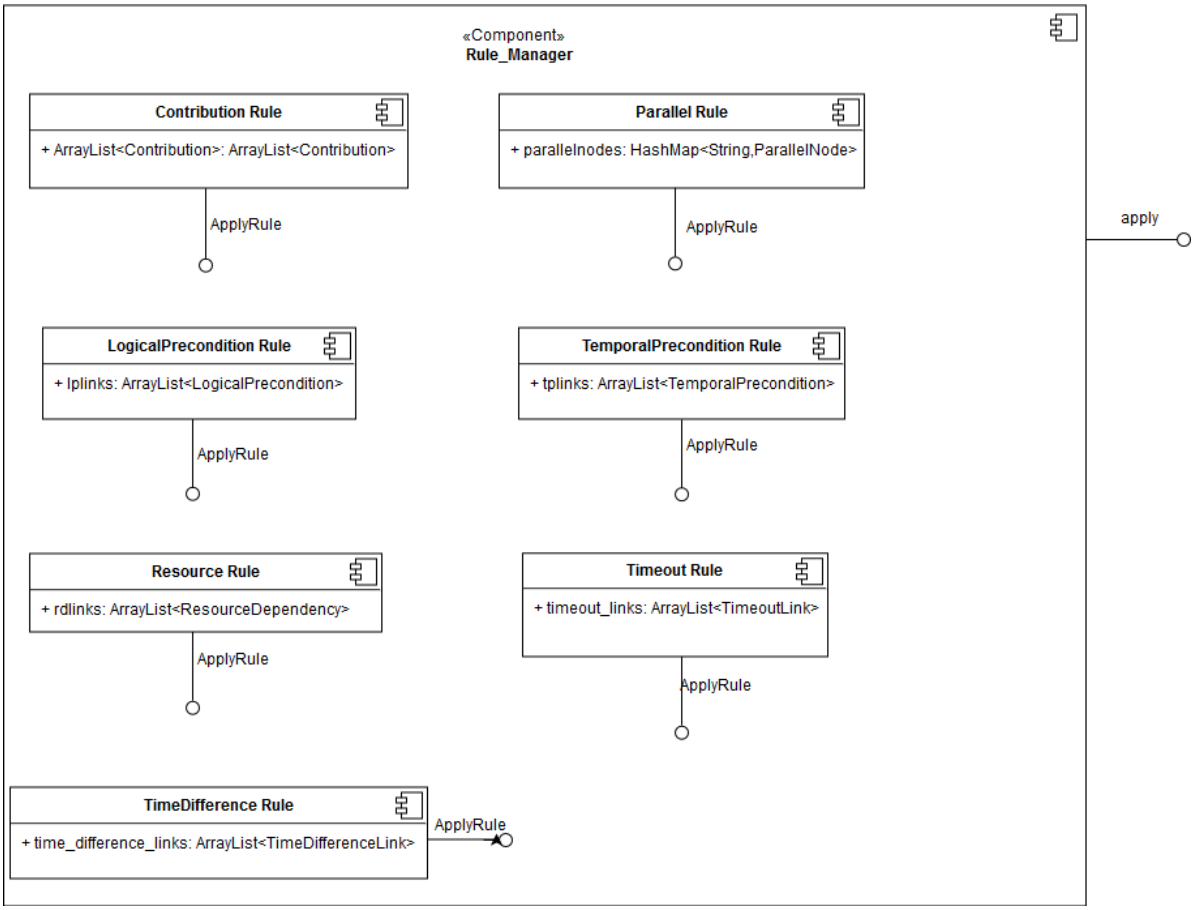


Σχήμα 3.3

Το component του Finder αποτελείται από το sub-component Find_Actions πραγματοποιεί την εύρεση των συνόλων από ατομικές ενέργειες. Επιπλέον, ορίζει τη διεπαφή query μέσω της οποίας ο Fog Node βρίσκει τα σύνολα που ικανοποιούν τους σύνθετους στόχους.

3.3.2 Rule Manager Component

Αυτό το component εφαρμόζει στο δέντρο - στόχων τους κανόνες μετασχηματισμού για να δημιουργήσει σύνολα με κόμβους και συνδέσεις, από όπου στη συνέχεια θα προκύψουν οι αλληλουχίες των ατομικών ενεργειών. Οι κανόνες μετασχηματισμού εφαρμόζονται με τη σειρά καλώντας κάθε φορά και την αντίστοιχη κλάση. Δημιουργούν Γράφους Εξάρτησης Ενεργειών, τοποθετούν ατομικές ενέργειες σε κάθε Γράφο, δημιουργούν συνδέσεις μεταξύ αυτών και τοποθετούν περιορισμούς που πρέπει να τηρηθούν κατά την πραγματοποίηση των στόχων. Η σειρά εφαρμογής των κανόνων είναι: Contribution Rule, Parallel Rule, LogicalPrecondition Rule, TemporalPrecondition Rule, Resource Rule, Timeout Rule και Timedifference Rule. Μετά την εφαρμογή των κανόνων έχουν προκύψει συνδέσεις μεταξύ ατομικών ενεργειών στα διάφορα σύνολα που αναπαριστούν τις εξαρτήσεις που υπάρχουν μεταξύ στόχων για την ικανοποίηση του μοντέλου και που χρησιμοποιούνται για την εξαγωγή αποτελεσματικών πλάνων ενεργειών.
Το ψηφιδικό διάγραμμα του component φαίνεται παρακάτω:



Σχήμα 3.4

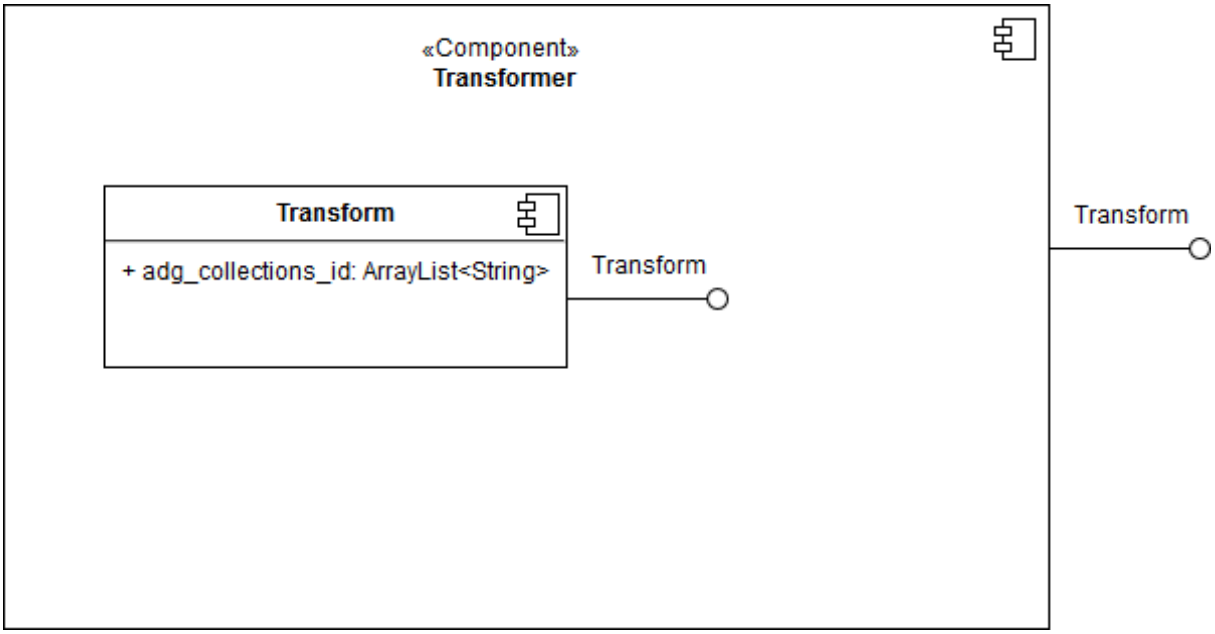
Όπως φαίνεται παραπάνω, το component Rule Manager ορίζει τα sub-components:

- Parallel Rule, που εφαρμόζει τους κανόνες που έχουν καθοριστεί για τους παράλληλους κόμβους.
- Contribution Rule, που κάνει τις απαραίτητες ενέργειες στις συνδέσεις συνεισφοράς.
- LogicalPrecondition Rule, που εφαρμόζει τον κανόνα για τις LogicalPrecondition συνδέσεις.
- TemporalPrecondition Rule, που εφαρμόζει τον κανόνα για τις TemporalPrecondition συνδέσεις.
- Resource Rule, που εφαρμόζει τον αντίστοιχο κανόνα για τις εξαρτήσεις που υπάρχουν στο δέντρο - στόχων μεταξύ ατομικών ενεργειών και πηγών δεδομένων.
- Timeout Rule, που εφαρμόζει τον κανόνα για τις χρονικές εξαρτήσεις μεταξύ ατομικών ενεργειών που πρέπει να εκτελεστούν μέσα σε συγκεκριμένο χρονικό διάστημα.
- Timedifference Rule, που εφαρμόζει τον κανόνα για τις χρονικές εξαρτήσεις μεταξύ ατομικών ενεργειών που πρέπει να εκτελεστούν με διαφορά συγκεκριμένου χρονικού διαστήματος.

Μέσω της διεπαφής Apply ο Fog Node έχει πρόσβαση σε αυτό το component για την εφαρμογή όλων των κανόνων μετασχηματισμού.

3.3.3 Transformer Component

Το component αυτό μετασχηματίζει τα σύνολα με τις ατομικές ενέργειες και τις συνδέσεις που έχουν προκύψει από την εφαρμογή των κανόνων μετασχηματισμού σε κατευθυνόμενους γράφους για να είναι ευκολότερη η διαχείριση τους και ποιο γρήγορη η εξαγωγή των τελικών αλληλουχιών που ικανοποιούν το δέντρο - στόχων. Για κάθε σύνολο ατομικών ενεργειών δημιουργείται ένας κατευθυνόμενος γράφος με τις κατάλληλες συνδέσεις, ο οποίος στη συνέχεια ταξινομείται τοπολογικά.
Το ψηφιδικό διάγραμμα αυτού του component είναι:

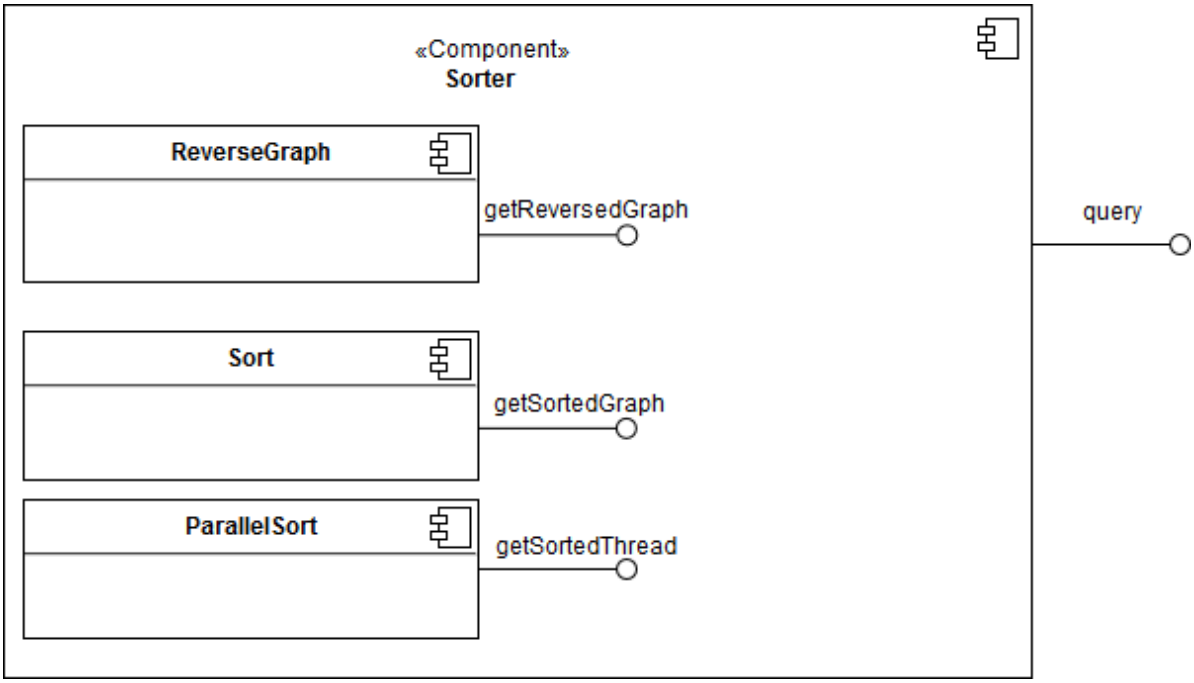


Σχήμα 3.5

Όπως φαίνεται και από το διάγραμμα το component Transformer αποτελείται από το sub-component Transform που μετασχηματίζει τα σύνολα που βρίσκονται στις ADG συλλογές σε κατευθυνόμενους γράφους ενώ προσφέρει τη διεπαφή Transform.

3.3.4 Component Sorter

Το component Sorter ταξινομεί τοπολογικά τους κατευθυνόμενους γράφους που έχουν δημιουργηθεί σε προηγούμενα βήματα του αλγορίθμου με σκοπό να εξαχθούν με ευκολία οι αλληλουχίες ατομικών ενεργειών. Η ταξινόμηση βασίζεται στις συνδέσεις που έχουν μεταξύ τους οι ατομικές ενέργειες χρησιμοποιώντας τον αλγόριθμο Tarjan[11].
Το ψηφιδικό διάγραμμα είναι:



Σχήμα 3.6

Το component Sorter αποτελείται από τα εξής sub-components:

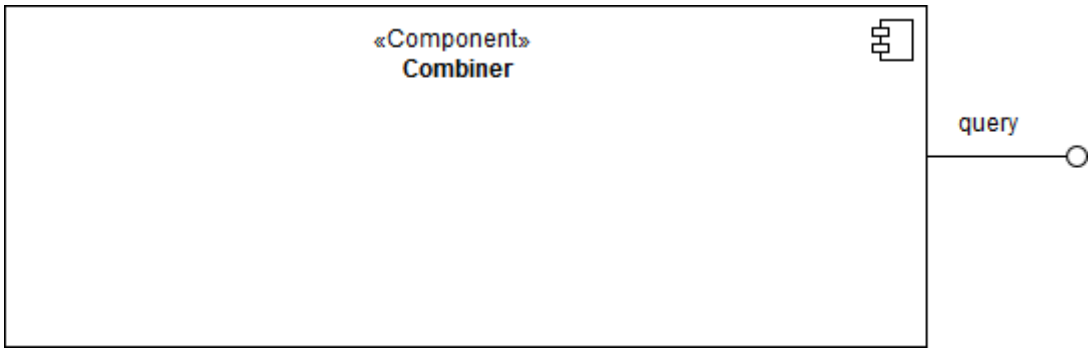
- ReverseGraph, όπου δημιουργείται μια ανάστροφη έκδοση του κατευθυνόμενου γράφου για την εφαρμογή του αλγορίθμου. Επιστρέφει μέσω της διεπαφής getReversedGraph τον ανάστροφο γράφο.
- Sort, όπου εφαρμόζεται DFS σε κατευθυνόμενο γράφο με ένα νήμα ατομικών ενεργειών για να βρεθεί η σειρά με την οποία πρέπει να εκτελεστούν οι ατομικές ενέργειες. Μέσω της διεπαφής getSortedGraph επιστρέφεται ο γράφος με καθορισμένη τη σειρά εκτέλεσης για κάθε ατομική ενέργεια.
- ParallelSort, όπου εφαρμόζεται DFS σε όλα τα νήματα του κατευθυνόμενου γράφου για να βρεθούν οι παράλληλες αλληλουχίες ατομικών ενεργειών. Μέσω της διεπαφής getSortedThread επιστρέφεται με καθορισμένη σειρά το νήμα ατομικών ενεργειών.

Το component Fog Node επικοινωνεί με το component Sorter μέσω της διεπαφής query για την τοπολογική ταξινόμηση των γράφων.

3.3.5 Component Combiner

Το component Combiner είναι επιφορτισμένο με το συνδυασμό των διάφορων αλληλουχιών που έχουν προκύψει από τις συλλογές των Γράφων Εξάρτησης Ενεργειών. Αντικαθιστά, δηλαδή, τους ψευδοκόμβους που έχουν δημιουργηθεί για κάθε παραλληλιά με τις κατάλληλες αλληλουχίες από την αντίστοιχη συλλογή γράφων. Οι συνδυασμοί που προκύπτουν αποτελούν τις κατάλληλες αλληλουχίες ατομικών ενεργειών που ικανοποιούν το δέντρο - στόχων, με όλους τους περιορισμούς που έχουν καθοριστεί.

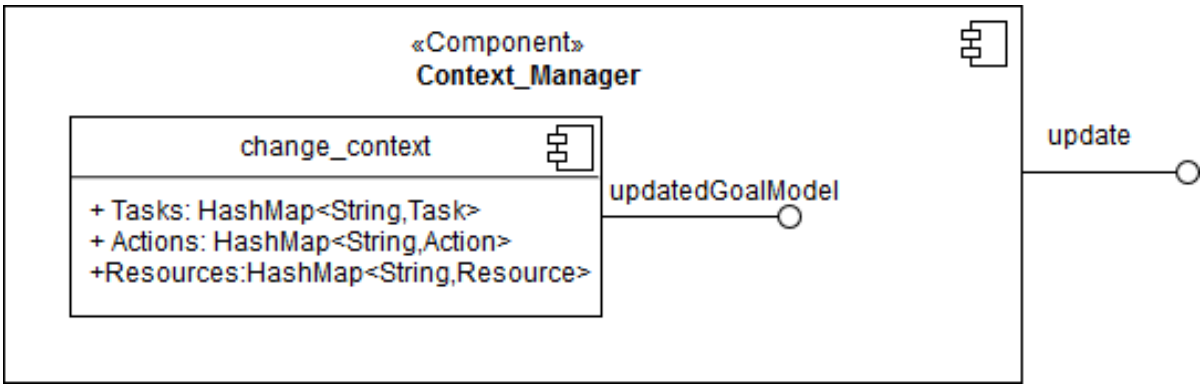
Το ψηφιδικό διάγραμμα είναι:



Σχήμα 3.7

3.3.6 Component Context Manager

Το component αυτό είναι υπεύθυνο για την παρακολούθηση του context βάσει του περιβάλλοντος και των επιλογών του χρήστη ενώ καθορίζει τους διαθέσιμους κόμβους του δέντρο - στόχων αλλά και τις διαθέσιμες πηγές δεδομένων. Αποτελεί σημαντικό κομμάτι του συστήματος καθώς η διαθεσιμότητα των κόμβων και των πηγών δεδομένων επηρεάζει όχι μόνο τις αλληλουχίες των ατομικών ενεργειών αλλά και αν υπάρχει κάποια τέτοια που ικανοποιεί το δέντρο - στόχων. Τέλος, μέσω αυτού του component καθορίζονται και οι κόμβοι που τελικά θα είναι διαθέσιμοι για να εκτελεστούν παράλληλα.
Το ψηφιδικό διάγραμμα φαίνεται παρακάτω:



Σχήμα 3.8

Το component Context Manager περιλαμβάνει το sub-component change context που αλλάζει τη διαθεσιμότητα των κόμβων ανάλογα με τα δεδομένα που δέχεται και ανανεώνει το μοντέλο μέσω της διεπαφής updateGoalModel. Το Fog Node επικοινωνεί με αυτό το component μέσω της διεπαφής update για τις αλλαγές στο δέντρο - στόχων.

3.4 Επικοινωνία μεταξύ των components του Συστήματος

Σε αυτή την ενότητα θα παρουσιαστούν με τη βοήθεια ακολουθιακών διαγραμμάτων η λειτουργία του συστήματος και οι αλληλεπιδράσεις που έχουν μεταξύ τους τα διάφορα components που το αποτελούν.

Κεντρικό κομμάτι του συστήματος, όπως έχει αναφερθεί και παραπάνω, είναι το Fog Node το οποίο επικοινωνεί με όλα τα υπόλοιπα components του συστήματος και συντονίζει τις λειτουργίες που πρέπει να εκτελεστούν. Ο Fog Node επικοινωνεί με το κατάλληλο component για να εκτελέσει συγκεκριμένες λειτουργίες και να επιστραφούν σε αυτόν δεδομένα όπως Γράφοι Εξάρτησης Ενεργειών, συλλογές γράφων κτλ. Όλα τα components επικοινωνούν μόνο με το Fog Node, το οποίο είναι υπεύθυνο άλλωστε για τη συνολική διαχείριση του συστήματος ενώ το καθένα από αυτά εκτελεί μια βασική υπο-λειτουργία του συστήματος. Οι λειτουργίες αυτές αφορούν τη δημιουργία νέων δέντρων - στόχων, την ανανέωση των μοντέλων - στόχων με βάση το περιβάλλον - πλαίσιο που εκτελούνται(context) και την εύρεση των πλάνων ατομικών ενεργειών. Κάθε component για την εκτέλεση της λειτουργίας που του έχει ανατεθεί και επιστρέφει τα δεδομένα που υπολογίζει στον Fog Node.

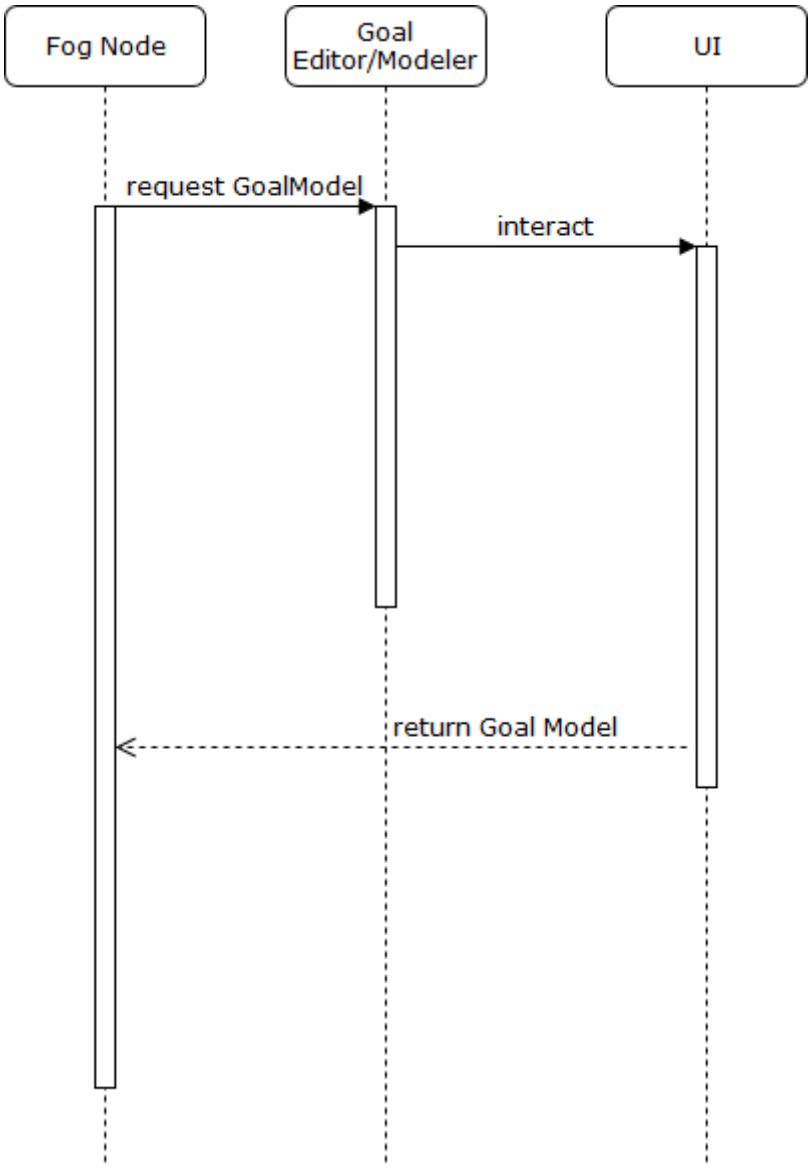
Η δημιουργία των νέων δέντρων - στόχων είναι ανεξάρτητη των υπόλοιπων λειτουργιών του συστήματος και επομένως θα παρουσιαστεί ξεχωριστά σαν ακολουθιακό διάγραμμα. Το ίδιο ισχύει και για την ανανέωση των δέντρων - στόχων αφού εκτελείται πριν την εύρεση των πλάνων. Αντίθετα, η εύρεση των πλάνων στηρίζεται σε διάφορες υπο-λειτουργίες που εκτελούνται σειριακά και επομένως εξαρτώνται μεταξύ τους.

Παρατίθενται, στη συνέχεια, τα ακολουθιακά διαγράμματα για τις βασικές λειτουργίες του συστήματος.

3.4.1 Ακολουθιακό Διάγραμμα Δημιουργίας Δέντρου - Στόχων

Στο διάγραμμα που ακολουθεί παρουσιάζονται τα βήματα και οι αλληλεπιδράσεις των components για την δημιουργία νέων δέντρων - στόχων.
Τα βήματα που ακολουθούνται είναι τα εξής:

1. **Request GoalModel**: Ο Fog Node αλληλεπιδρά με τους Goal Editor και Goal Modeler για την υλοποίηση μοντέλου - στόχων με τις προϋποθέσεις που έχουν καθοριστεί στο Goal Modeler.
2. **Interact:** Ο Goal Editor αλληλεπιδρά με το UI και κατ΄ επέκταση με το χρήστη ο οποίος δημιουργεί ανάλογα με τις ανάγκες του ένα δέντρο – στόχων.
3. **Return Goal Model:** Μετά τη δημιουργία του μοντέλου, αυτό επιστρέφεται στον Fog Node για διαχείριση.



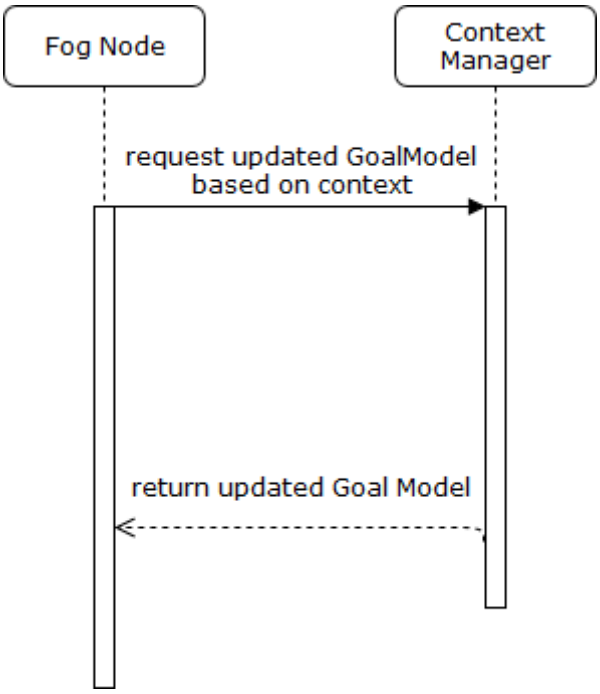
Σχήμα 3.9

3.4.2 Ακολουθιακό Διάγραμμα Ανανέωσης Δέντρου-Στόχων

Το παρακάτω ακολουθιακό διάγραμμα απεικονίζει τη σειρά ενεργειών που εκτελείται για την ανανέωση ενός δέντρου - στόχου με βάση το περιβάλλον – πλαίσιο (context).

Τα βήματα είναι τα εξής:

1. **Requested updated Goal Model:** Ο Fog Node αλληλεπιδρά με τον Context Manager και "δίνει εντολή" για την ανανέωση του παρόντος δέντρου - στόχων με βάση το περιβάλλον – πλαίσιο (context).
2. **Return updated GoalModel**: Μετά την ανανέωση του μοντέλου - στόχων αυτό επιστρέφεται στον Fog Node για την περαιτέρω ανάλυση.



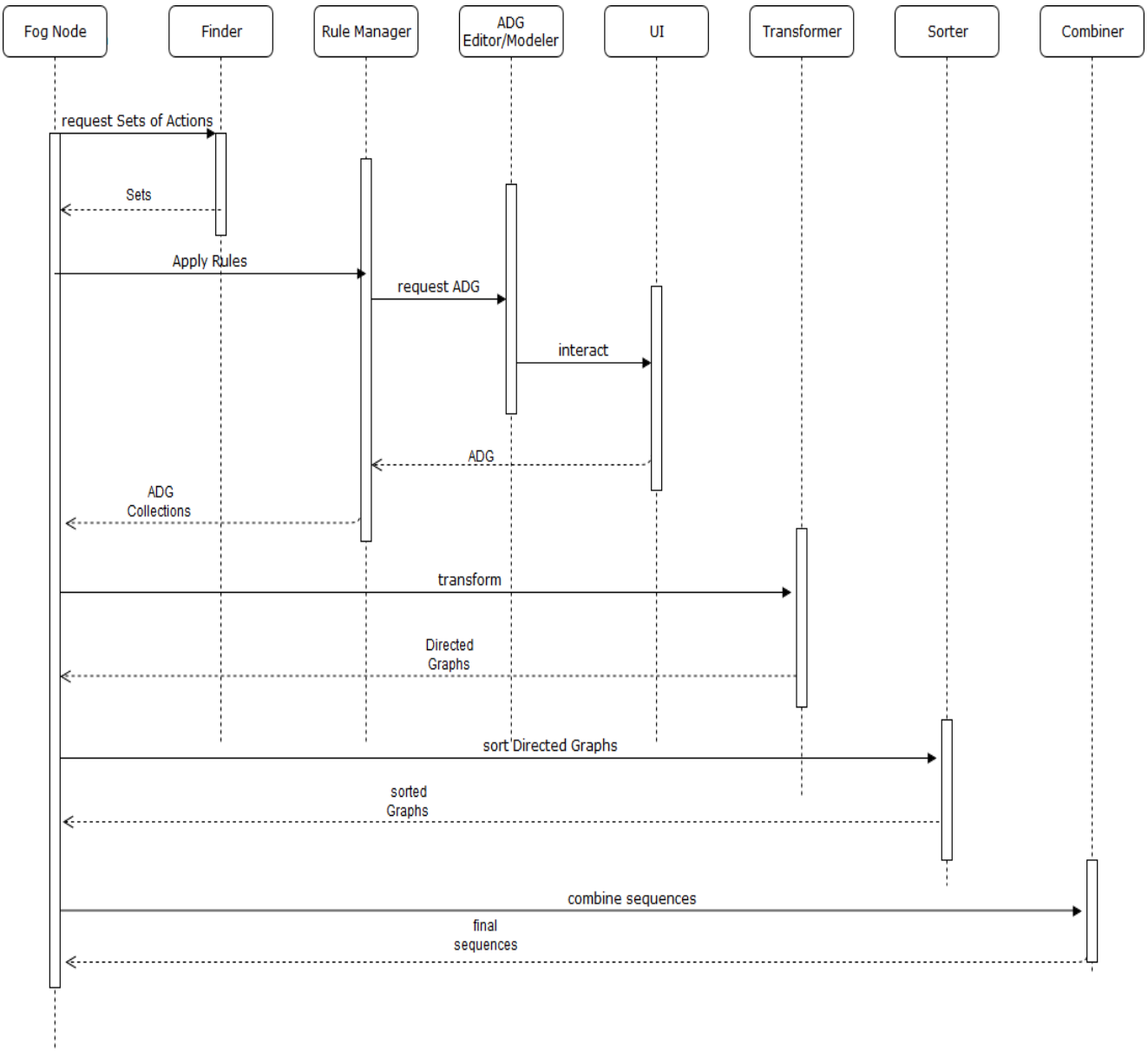
Σχήμα 3.10

3.4.3 Ακολουθιακό Διάγραμμα Εύρεσης Πλάνων Ατομικών Ενεργειών

Στην παρούσα υποενότητα θα παρουσιαστεί το ακολουθιακό διάγραμμα της εύρεσης των πλάνων από ατομικές ενέργειες και θα αναλυθούν τα βήματα εκτέλεσης της λειτουργίας.
Τα βήματα είναι τα εξής:

1. **Request Sets of Actions:** Ο Fog Node αναζητά όλα τα σύνολα ατομικών ενεργειών που ικανοποιεί κάθε σύνθετο στόχο (Task) και αλληλεπιδρά με το component Finder για την εκτέλεση της λειτουργίας.
2. **Sets:** Το component Finder επιστρέφει τα σύνολα που υπολογίστηκαν, τα οποία στη συνέχεια χρησιμοποιούνται για την εξαγωγή των αλληλουχιών.
3. **Apply Rules:** Ο Fog Node καλεί το component Rule Manager για να εφαρμόσει τους κανόνες μετασχηματισμού στα σύνολα ατομικών ενεργειών που έχουν προκύψει και να τα μετατρέψει σε μια αρχική μορφή ADG.
4. **Request ADG:** Ο Rule Manager αλληλεπιδρά με τους ADG Editor και ADG Modeler για την δημιουργία ADG με τις προϋποθέσεις που έχουν καθοριστεί στον ADG Modeler.
5. **Interact:** Ο ADG Editor αλληλεπιδρά με το UI και κατ' επέκταση με το χρήστη ο οποίος δημιουργεί ADG ανάλογα με τις επιθυμίες του.
6. **ADG:** Επιστρέφεται στον Rule Manager το ADG που δημιουργήθηκε.
7. **ADG Collections:** Επιστρέφονται όλες οι συλλογές με ADG που έχουν δημιουργηθεί κατά την εφαρμογή των κανόνων μετασχηματισμού.

8. **Transform:** Καλείται το component Transformer από τον Fog Node για την μετατροπή των πρώιμων ADG σε κατευθυνόμενους γράφους.
9. **Directed Graphs:** Επιστρέφονται στον Fog Node όλοι οι μετασχηματισμένοι κατευθυνόμενοι γράφοι.
10. **Sort Directed Graphs:** Καλεί ο Fog Node το component Sorter για την τοπολογική ταξινόμηση των γράφων για την μετέπειτα εξαγωγή των αλληλουχιών.
11. **Sorted Graphs:** Επιστρέφονται οι ταξινομημένοι γράφοι στον Fog Node για περαιτέρω επεξεργασία.
12. **Combine sequences:** Καλείται το component Combiner για τον συνδυασμό κατάλληλων αλληλουχιών ατομικών ενεργειών από διαφορετικές συλλογές ADG για την εξαγωγή των τελικών αποτελεσμάτων.
13. **Final sequences:** Επιστρέφονται τα τελικά αποτελέσματα στον Fog Node.



Σχήμα 3.11

Κεφάλαιο 4:Εννοιολογικά Μοντέλα

4.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζονται τα εννοιολογικά μοντέλα της δομής του Διαδικτύου των Πραγμάτων και των δέντρων - στόχων που προτείνονται στην παρούσα διπλωματική. Επιπλέον, προς το τέλος του κεφαλαίου δίνεται το μεταμοντέλο του Γράφου Εξάρτησης Ενεργειών(Action Dependency Graph,ADG) το οποίο χρησιμοποιήθηκε για την εύρεση των ακολουθιών από ενέργειες που ικανοποιούν το στόχο - ρίζα του δέντρου[9]. Αναλύονται όλες οι οντότητες που περιλαμβάνονται σαν εννοιολογικά μοντέλα και οι σχέσεις που αναπτύσσουν μεταξύ τους. Τα μοντέλα που εκπονήθηκαν στηρίζονται στο πρότυπο MOF και αποτελούν τη βάση του λογισμικού που αναπτύχθηκε λόγω της αυτόματης παραγωγής κώδικα που προσφέρεται μέσω του EMF.

4.2 Εννοιολογικό Μοντέλο Διαδικτύου των Πραγμάτων

Το εννοιολογικό μοντέλο της δομής του Διαδικτύου των Πραγμάτων σχεδιάστηκε για να καλύπτει τις σύγχρονες ανάγκες τέτοιων ειδών συστημάτων και να καλύπτει ένα ευρύ φάσμα εφαρμογών, από εφαρμογές e-health και έξυπνου σπιτιού μέχρι εφαρμογές συλλογής επιστημονικών δεδομένων. Επιπλέον, παρέχει ένα καθολικό συντακτικό πρότυπο εκπόνησης IoT μοντέλων, κάτι το οποίο διευκολύνει σε μεγάλο βαθμό τη διαχείριση τους, και επεκτείνει υπάρχουσες έρευνες πάνω σε αυτό το κομμάτι[4][5].

Στη συνέχεια του παρόντος υποκεφαλαίου παρουσιάζονται μία προς μία οι κλάσεις του εννοιολογικού μοντέλου και δίνεται ένα διάγραμμα κλάσεων για μία εποπτική εικόνα όλου του μεταμοντέλου.

4.2.1 Κλάση Entity

Αποτελεί τη βασική κλάση του μεταμοντέλου. Εκφράζει σαν κλάση τις οντότητες που περιέχει το μεταμοντέλο και ουσιαστικά αποτελεί την κλάση - ρίζα για τις βασικές κλάσεις του μεταμοντέλου(User,Thing,Attribute).

4.2.2 Κλάση User

Η κλάση User αντιπροσωπεύει όλους τους δυνατούς χρήστες που αλληλεπιδρούν με κάποια οντότητα του Διαδικτύου των Πραγμάτων(κάποια συσκευή του συστήματος, κάποιος άλλος χρήστης κτλ). Ο χρήστης κάνει αίτηση ή εγγραφή σε fog υπηρεσίες οι οποίες με τη σειρά τους ενορχηστρώνουν κατάλληλα τις συσκευές του Διαδικτύου των Πραγμάτων για να ικανοποιήσουν τις επιθυμίες του χρήστη. Ο χρήστης μπορεί να συνδεθεί με όσες υπηρεσίες επιθυμεί, όπως φαίνεται και από την σύνδεση

subscribes_invokes πολλαπλότητας 0...*. User μπορεί να είναι είτε κάποιο φυσικό πρόσωπο είτε κάποιος πράκτορας λογισμικού (software agent).

4.2.3 Κλάση Attribute

Αποτελεί μία αφηρημένη κλάση η οποία αντιπροσωπεύει όλα τα δυνατά attributes που μπορούν να δημιουργηθούν για κάθε κλάση. Μπορεί να γίνει extend για όποια κλάση επιθυμούμε και οποιοδήποτε είδος attributes θέλουμε να ορίσουμε. Η αφηρημένη αυτή κλάση περιέχει κάποιες βασικές ιδιότητες όπως είναι το όνομα του attribute και ο τύπος του.

4.2.4 Κλάση Thing

Η κλάση αυτή αποτελεί κεντρικό κομμάτι του μεταμοντέλου καθώς αντιπροσωπεύει τα «Πράγματα» από τα οποία αποτελείται το Διαδίκτυο των Πραγμάτων. Αποτελείται από 2 κομμάτια, τη φυσική του απεικόνιση στον πραγματικό κόσμο (Physical Thing) και την εικονική του απεικόνιση στον ψηφιακό κόσμο (Virtual Thing). Κάθε αντικείμενο αυτής της κλάσης συνδέεται με ένα σύστημα fog computing από το οποίο ζητά υπηρεσίες για την ικανοποίηση του χρήστη, όπως φαίνεται και από την αμφίδρομη σχέση με την κλάση fog(request_service 1..*,1..1). Τέλος, ένα αντικείμενο τύπου Thing μπορεί να αποτελείται από ένα ή περισσότερα αντικείμενα του ίδιου τύπου.

4.2.5 Κλάση Property

Είναι μία αφηρημένη κλάση που απεικονίζει τις ιδιότητες που έχει ένα αντικείμενο τύπου Thing. Είναι επεκτάσιμη για να καθορίσει ο εκάστοτε μηχανικός λογισμικού τις ιδιότητες που είναι απαραίτητες και περιέχει το βασικό πεδίο για το αν είναι μεταβλητές οι ιδιότητες ή όχι.

4.2.6 Κλάση Physical Thing

Είναι αντικείμενο του πραγματικού κόσμου και αποτελεί τη φυσική υπόσταση των αντικειμένων τύπου Thing, δηλαδή των «Πραγμάτων» που σχηματίζουν το δίκτυο. Περιέχει μία ή περισσότερες συσκευές (κλάση Device) που επιτελούν διάφορες λειτουργίες στο Διαδίκτυο των Πραγμάτων.

4.2.7 Κλάση Digital Artifact

Είναι κάποιο ψηφιακό τεχνούργημα το οποίο εκτελεί κάποιες λειτουργίες. Χωρίζεται σε 2 κατηγορίες: ενεργό και παθητικό.

4.2.8 Κλάση Active Digital Artifact

Είναι οποιοσδήποτε ενεργός κώδικας που τρέχει, όπως κάποια ενσωματωμένη εφαρμογή ή κάποιος πράκτορας λογισμικού. Τέτοιου είδους ψηφιακά τεχνουργήματα είναι δυνατόν να αποτελέσουν χρήστες του συστήματος.

4.2.9 Κλάση Passive Digital Artifact

Είναι οποιοδήποτε ψηφιακό αντικείμενο το οποίο είναι αποθηκευμένο σε μορφή βάσης δεδομένων ή σε οποιαδήποτε τέτοια μορφή, δηλαδή δεν αλληλεπιδρά από μόνο του με άλλα αντικείμενα.

4.2.10 Κλάση Virtual Thing

Είναι ένα ψηφιακό τεχνούργημα το οποίο απεικονίζει το πραγματικό αντικείμενο στον ψηφιακό κόσμο, όπως φαίνεται και από τη σχέση *represents[1..1]*. Το κάθε αντικείμενο κλάσης Virtual Thing χαρακτηρίζεται από ένα μοναδικό URI που έχει ως σκοπό την μεταξύ τους διάκριση. Επιπλέον, συνδέεται μέσω της σχέσης *is_associated_with[0..*]* με πηγές δεδομένων που αφορούν με το ίδιο το αντικείμενο, τις συσκευές του δικτύου αλλά και πληροφορίες για το δίκτυο.

4.2.11 Κλάση Device

Η κλάση Device αντιπροσωπεύει τις συσκευές οι οποίες βρίσκονται συνδεδεμένες στο δίκτυο. Κάθε αντικείμενο τύπου Device μπορεί να αποτελείται από ένα ή περισσότερα αντικείμενα ίδιου τύπου όπως και από άλλα συστατικά μέρη όπως είναι: Communicator, On-device Resource, Rule, Device State, τα οποία αναλύονται παρακάτω. Παράλληλα, χωρίζεται σε 3 κατηγορίες: Tag, Actuator και Sensor.

4.2.12 Κλάση Communicator

Αντιπροσωπεύει το κομμάτι εκείνο της συσκευής που είναι υπεύθυνο για την επικοινωνία της με άλλες συσκευές και με το περιβάλλον. Ενσωματώνει κάποια χαρακτηριστικά όπως είναι ο τύπος της επικοινωνίας καθώς και ο αριθμός των θυρών που υπάρχουν για επικοινωνία. Συνδέεται με την κλάση Port μέσω της σχέσης *has_ports[0..*]* και αποτελεί ένα από τα συστατικά μέρη των αντικειμένων τύπου Device.

4.2.13 Κλάση Port

Αναπαριστά τις θύρες επικοινωνιών που είναι διαθέσιμες σε κάθε συσκευή.

4.2.14 Κλάση Device State

Κάθε αντικείμενο του πραγματικού κόσμου βρίσκεται ανά πάσα στιγμή σε μία συγκεκριμένη κατάσταση. Η κατάσταση αυτή αναπαρίσταται με την κλάση Device State ενώ οι ενεργές και ανενεργές καταστάσεις διακρίνονται από το χαρακτηριστικό *Enabled*. Μπορούν να οριστούν και πιο σύνθετες καταστάσεις αντικειμένων μέσω της κλάσης CompositeState που είναι υποκλάση της DeviceState.

4.2.15 Κλάση Sensor

Είναι μία αφηρημένη κλάση που αντιπροσωπεύει τους αισθητήρες που υπάρχουν συνήθως σε ένα Διαδίκτυο των Πραγμάτων. Η κλάση Sensor έχει μία σχέση *monitors* πολλαπλότητας 0..* με την κλάση Physical Thing καθώς και μία σχέση *observes* πολλαπλότητας 0..1 με την κλάση DeviceState. Οι σχέσεις αυτές απεικονίζουν την λειτουργία που εκτελούν οι αισθητήρες να παρατηρούν αντικείμενα του φυσικού κόσμου όπως και την κατάσταση τους. Χαρακτηριστικά των αισθητήρων είναι το όνομα τους, η συχνότητα παρατήρησης και μέτρησης και το αν είναι ενεργοποιημένοι. Χωρίζονται σε 2 κατηγορίες: σε αισθητήρες που βρίσκονται ενσωματωμένοι σε συσκευές - «Πράγματα» του δικτύου και σε εξωτερικούς αισθητήρες. Επιπλέον, συνδέονται με την κλάση Action που περιλαμβάνει δράσεις που μπορούν να γίνουν και οι οποίες πυροδοτούνται από πληροφορίες που έχουν συλλέξει οι αισθητήρες.

4.2.16 Κλάση Actuator

Αντιπροσωπεύει τους ενεργοποιητές που βρίσκονται στο σύστημα, οι οποίοι έχουν τη δυνατότητα να αλλάξουν τη φυσική κατάσταση των αντικειμένων ή/και να ενεργοποιήσουν ή να απενεργοποιήσουν διάφορες λειτουργικότητες σε αυτά. Ο κάθε ενεργοποιητής διακρίνεται από το όνομα του και περιλαμβάνει σχέσεις τόσο με την κλάση Physical Thing(*acts[0..*])* όσο και με την κλάση Device State (*observes[0..1])* οι οποίες απεικονίζουν τις δυνατότητες που περιγράφτηκαν παραπάνω. Χωρίζονται σε 2 κατηγορίες: Device Actuator και External Actuator ενώ συνδέονται με την κλάση Action με τις δράσεις που έχει τη δυνατότητα να πραγματοποιήσει ο κάθε ενεργοποιητής.

4.2.17 Κλάση Action

Είναι οι δράσεις που μπορούν να εφαρμοστούν στις διάφορες συσκευές του συστήματος και στα μέρη αυτών.

4.2.18 Κλάση Tag

Τα αντικείμενα της κλάσης Tag χρησιμοποιούνται για να αναγνωρίσουν φυσικές οντότητες(*identifies [0..*])* ενώ συνήθως είναι προσκολλημένα πάνω σε αυτές. Ένα τέτοιο αντικείμενο μπορεί να είναι για παράδειγμα ένα barcode ή ένα QR code.

4.2.19 Κλάση Transition

Η κλάση αποτελείται από δράσεις που αντικείμενα των κλάσεων Sensor και Actuator έχουν την δυνατότητα να πραγματοποιήσουν και αναπαριστά τις δυνατές μεταβάσεις καταστάσεων κάποιας συσκευής.

4.2.20 Κλάση Rule

Πέρα από τα αντικείμενα τύπου Sensor και Actuator που έχουν τη δυνατότητα να πυροδοτήσουν κάποιες ενέργειες σε μια συσκευή ή στο σύστημα, η ίδια η συσκευή μπορεί να διαθέτει κάποιους κανόνες για την αυτόματη εκτέλεση κάποιων ενεργειών ή ακόμα να καλεί τα παραπάνω αντικείμενα να εκτελέσουν κάποιες ενέργειες, με βάση κάποιες συνθήκες οι οποίες έχουν καθοριστεί για την συγκεκριμένη συσκευή και ικανοποιούνται. Οι ιδιότητες που έχουν αυτά τα αντικείμενα είναι το είδος της συνθήκης(condition literal),π.χ. μεγαλύτερο από, και η τιμή που πυροδοτεί τον κανόνα(*condition value).*

4.2.21 Κλάση Information Resource

Είναι κάποια πηγή δεδομένων με την οποία συσχετίζεται κάθε αντικείμενο τύπου Virtual Thing, δηλαδή η ψηφιακή απεικόνιση των «Πραγμάτων», για να έχει πρόσβαση σε πληροφορίες που αφορούν το ίδιο το αντικείμενο, το περιβάλλον καθώς και άλλα αντικείμενα. Χωρίζεται σε 2 κατηγορίες: On-device Resource και Network Resource, ανάλογα με το πού βρίσκεται η πηγή των δεδομένων. Κάθε τέτοια πηγή δεδομένων μπορεί να περιέχει 2 ειδών πηγές δεδομένων: αυτές που αφορούν την ίδια τη συσκευή (*Device_Resource)* και περιέχουν δεδομένα και μεταδεδομένα σχετικά με αυτήν και αυτές που σχετίζονται με το σύστημα και τις υπηρεσίες που τρέχουν σε αυτό (*Service Resources),* όπου περιέχονται στοιχεία του συστήματος καθώς και κάποιοι αλγόριθμοι. Κάθε αντικείμενο αυτής της κλάσης αποτελείται από *Information* και η συσχέτιση με την αφηρημένη κλάση *Attributes* δίνει την δυνατότητα να καθοριστούν ανάλογα με το σύστημα τα χαρακτηριστικά αυτών.

4.2.22 Κλάση Information

Αποτελούν τα συστατικά μέρη των αντικειμένων Information Resource και αντιπροσωπεύουν τις πληροφορίες που υπάρχουν για το σύστημα και για τις συσκευές που βρίσκονται σε αυτό. Η κλάση αποτελείται από 2 ειδών δεδομένα : Από τα *AtomicData* και από *DataStreams.* Τα *AtomicData* είναι μεμονωμένα δεδομένα χωρίς απαραίτητα να έχουν κάποια συσχέτιση μεταξύ τους και συνδέονται με την αφηρημένη κλάση *AtomicDataAttributes (*υποκλάση της αφηρημένης κλάσης Attributes) όπου περιέχονται βασικά χαρακτηριστικά τα οποία διακρίνουν αυτού του είδους τα δεδομένα, όπως είναι η κωδικοποίηση των δεδομένων και ποια συσκευή αφορούν. Τα *DataStreams* είναι ροές δεδομένων και αποτελούνται από *AtomicData* τα

οποία συσχετίζονται μεταξύ τους. Συνδέεται με την αφηρημένη κλάση *DataStreamAttribute* (υποκλάση της αφηρημένης κλάσης Attributes) και περιέχει τα βασικά χαρακτηριστικά των DataStream όπως ο μέσος ρυθμός bit, χρονοσφραγίδα της ροής, μέγιστος ρυθμός bit, μορφή των δεδομένων της ροής, κωδικοποίηση , συσκευή/συσκευές που αφορά και περιγραφή της ροής αυτής. Προφανώς, αφού είναι αφηρημένη κλάση μπορούν να εισαχθούν και παραπάνω χαρακτηριστικά ανάλογα με την εφαρμογή.

4.2.23 Κλάση Fog

Αντιπροσωπεύει το fog σύστημα με το οποίο συνδέεται το Διαδίκτυο των Πραγμάτων και είναι απαραίτητο για την λειτουργία του καθώς μέσω αυτού ενορχηστρώνονται οι διάφορες υπηρεσίες που προσφέρονται. Βασίζεται στην αρχιτεκτονική που διέπει το fog computing[6] και προσφέρει αρκετά πλεονεκτήματα σε σχέση με την πιο απομακρυσμένη αρχιτεκτονική του cloud computing[7]. Αποτελείται από διάφορους τοπικούς fog κόμβους και συνδέεται με μία βάση δεδομένων. Τέλος, χρησιμοποιεί έναν Authorizer για να καθορίζει την πρόσβαση στο σύστημα με κάποια control policies ενώ συνδέεται με ένα cloud σύστημα.

4.2.24 Κλάση Fog Node

Είναι οι τοπικοί κόμβοι fog οι οποίοι σχηματίζουν το σύστημα fog. Κάθε κόμβος περιέχει ένα analytics engine, που προσφέρει ανάλυση δεδομένων σε πραγματικό χρόνο[8] και συσχετίζεται με μία ή πολλές υπηρεσίες που προσφέρονται από το σύστημα fog(Fog Services). Εφαρμογές και υπηρεσίες του fog node μπορούν να τρέχουν είτε σε κάποιον Container είτε σε κάποια εικονική μηχανή.

4.2.25 Κλάση Fog Services

Αποτελούν τις υπηρεσίες που προσφέρονται από το σύστημα και οι οποίες ενορχηστρώνονται από το fog και εκτελούνται στο Διαδίκτυο των Πραγμάτων. Τα Fog Services εκθέτουν τις πηγές δεδομένων που διατίθενται από το Διαδίκτυο των Πραγμάτων (Information Resources), γνωρίζουν πώς να τα διαχειριστούν και προσφέρουν στον χρήστη μια διεπαφή για να αλληλεπιδράσει με την κάθε υπηρεσία. Ο χρήστης έχει τη δυνατότητα να καλέσει κάποιο από αυτά τα Services ή/και να κάνει εγγραφή σε αυτά. Είναι συνδεδεμένα με τις ψηφιακές απεικονίσεις των αντικειμένων του Διαδικτύου των Πραγμάτων(*is_connected_with [1..*]*) και εκτελούνται μέσα σε κάποιον Container. Αποτελούνται από microperations οι οποίες καλούνται και υλοποιούν την υπηρεσία.

4.2.26 Κλάση Operations

Τα Fog Services αποτελούνται από operations που αποτελούν βασικά συστατικά για την εκτέλεση των Fog Services. Τρέχουν σε container. Συσχετίζονται με διάφορους evaluators, οι οποίοι υλοποιούν την προγραμματιστική λογική των operations με διάφορους τρόπους(πχ java,python κτλ) και γι αυτό απεικονίζεται στο παρόν μεταμοντέλο ως

αφηρημένη κλασή. Υπάρχει κάποια μέθοδο do σε κάθε operation, την οποία κάνει extend ο evaluator και μέσα εκεί υλοποιεί την εκάστοτε λειτουργικότητα. Εγγράφεται σε διάφορες πηγές δεδομένων του Διαδικτύου των Πραγμάτων από όπου αντλούν πληροφορίες για την εκτέλεση των operations.

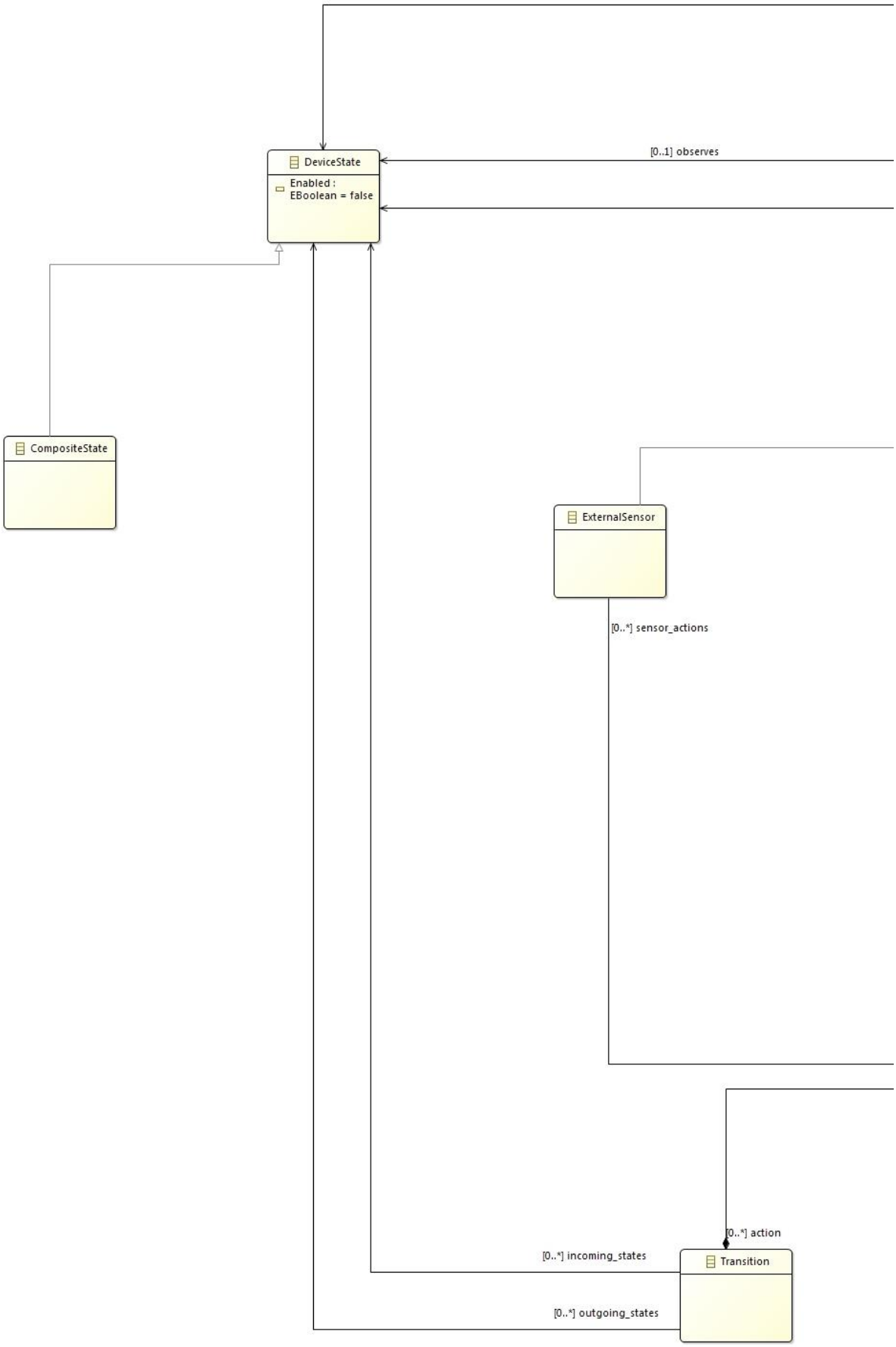
4.2.27 Κλάση Database

Απεικονίζει την βάση δεδομένων του fog συστήματος, περιέχει δεδομένα για το σύστημα και έναν Policy repository στον οποίο αποθηκεύονται οι πολιτικές του συστήματος.

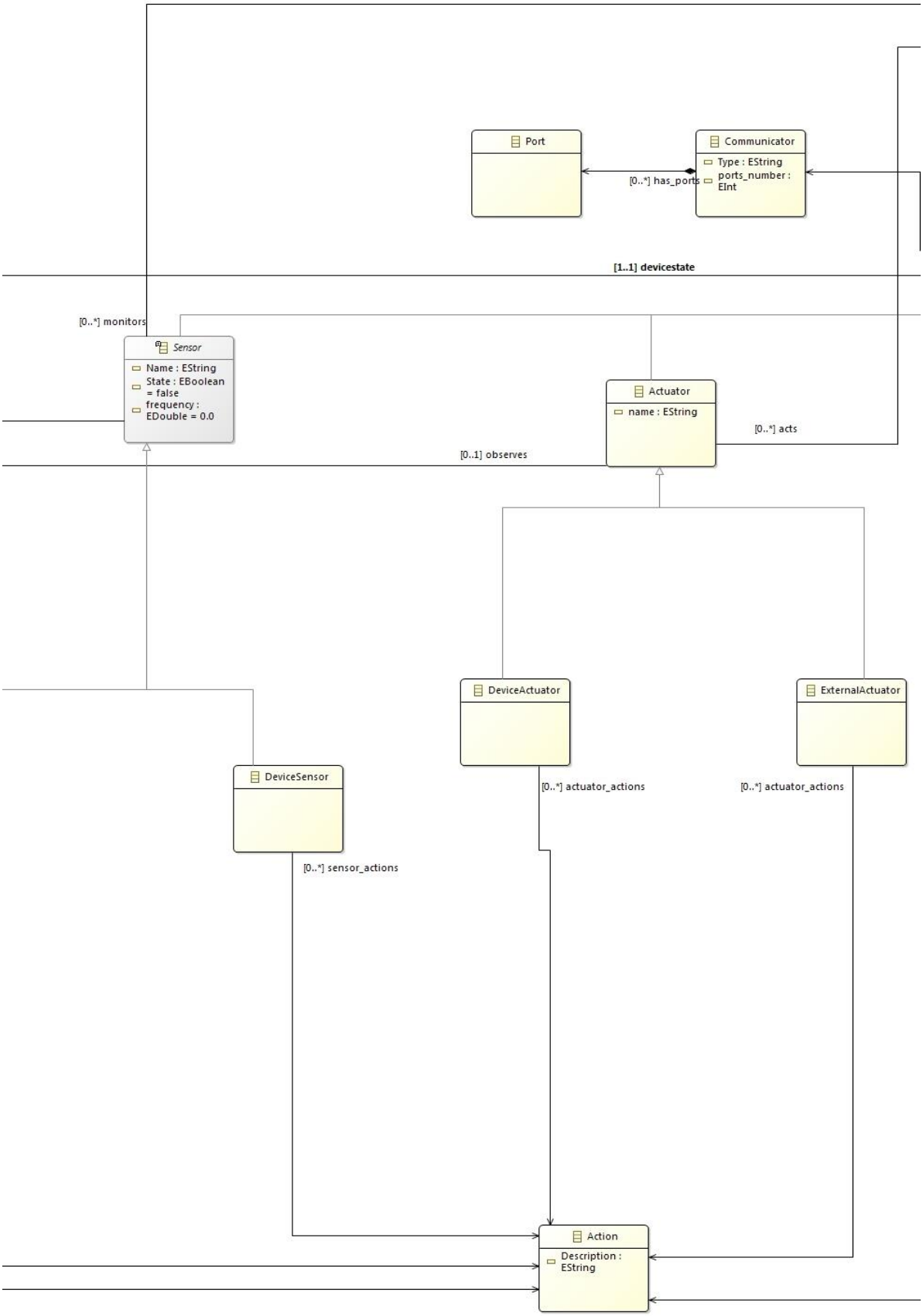
4.2.28 Κλάση Policy Repository

Περιέχει τις πολιτικές που εφαρμόζονται στο σύστημα και συσχετίζεται με ένα Reference Monitor, το οποίο ενορχηστρώνει αυτές τις πολιτικές στο σύστημα και εφαρμόζει τους κανόνες πρόσβασης στον Authorizer, συστατικό μέρος του Policy Repository.

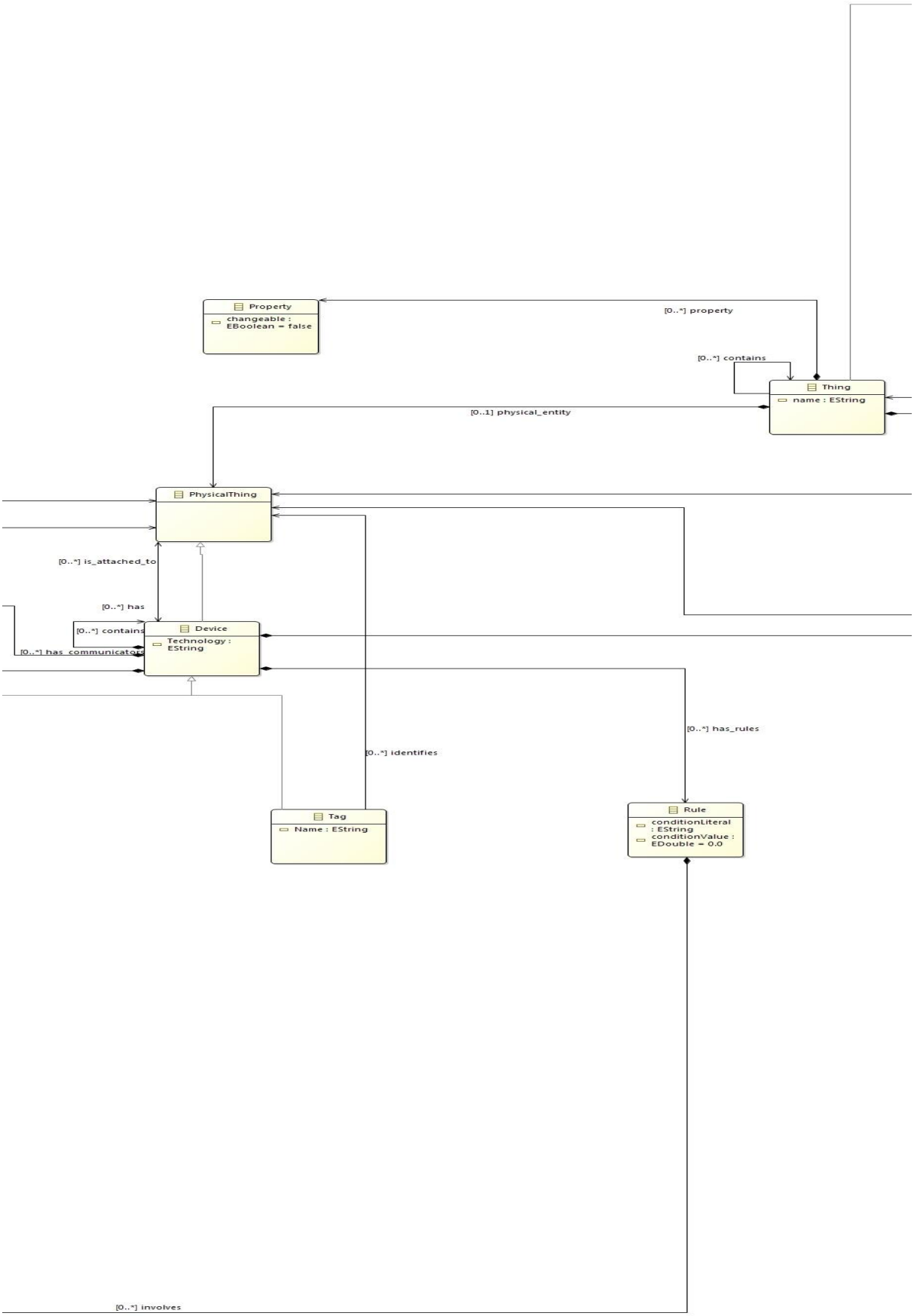
Παρακάτω παρατίθεται σε κομμάτια το μεταμοντέλο του IoT καθώς και το πλήρες διάγραμμα του.



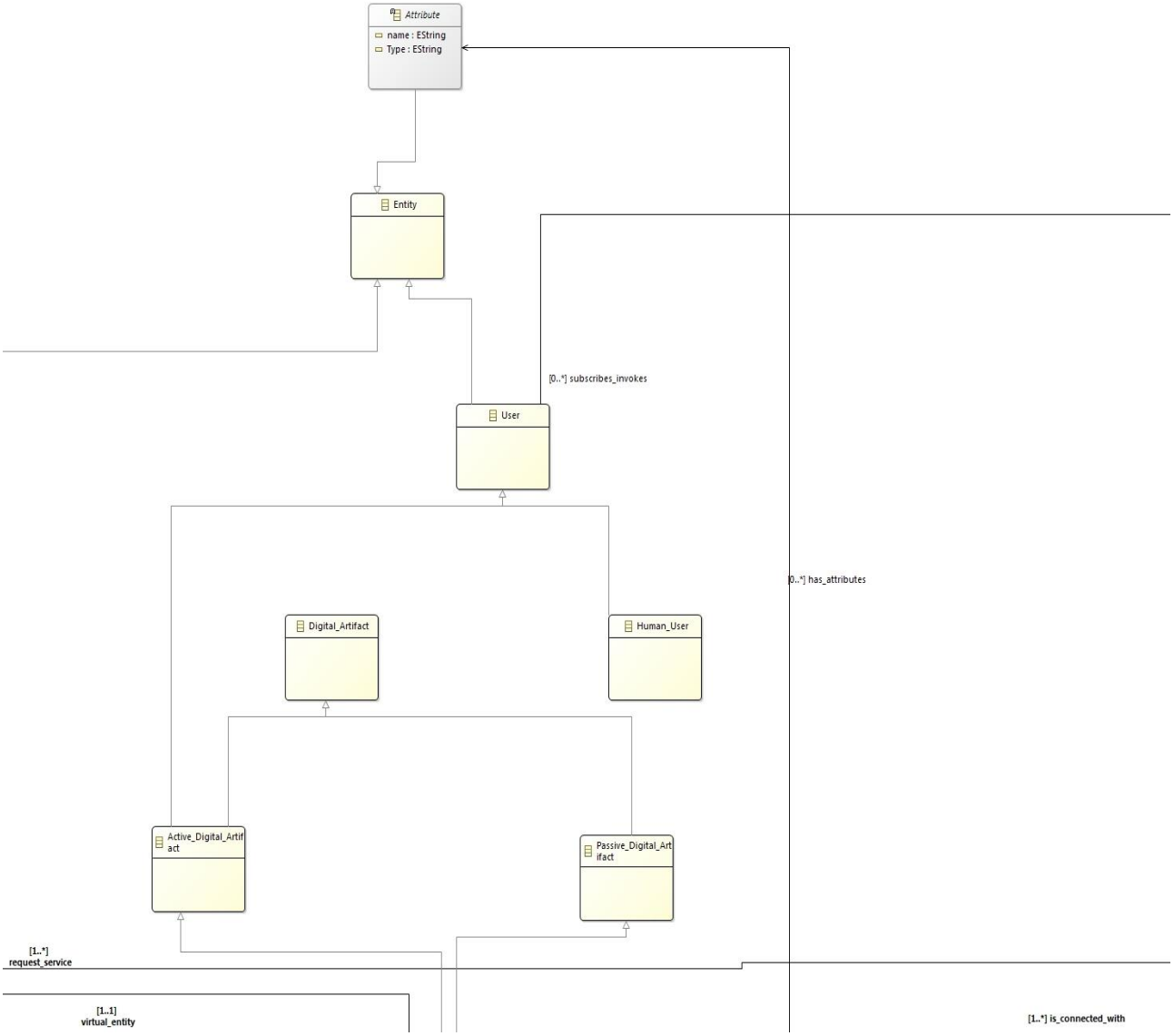
Σχήμα 4.1-Διάγραμμα IoT metamodel 1



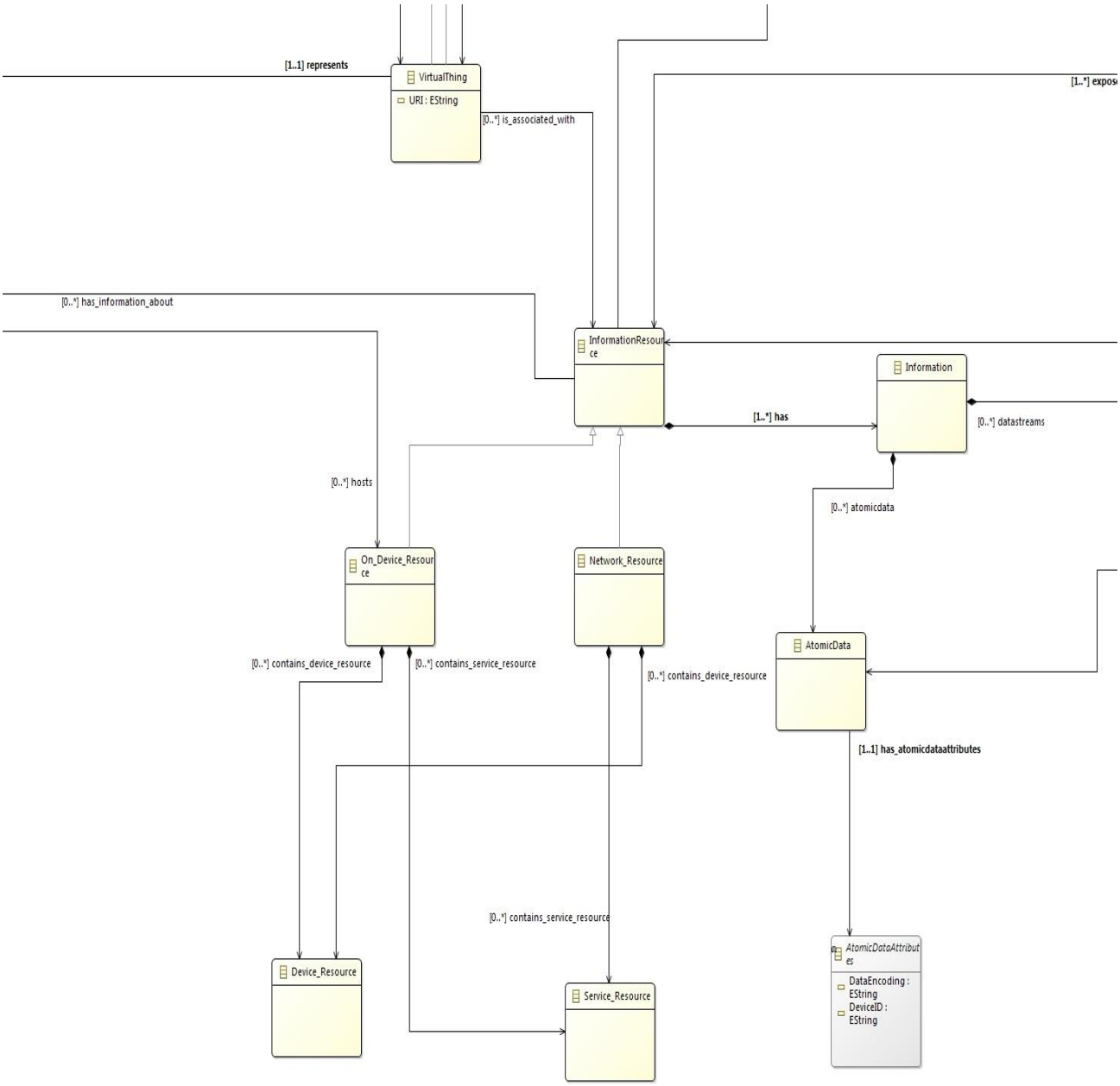
Σχήμα 4.2-Διάγραμμα IoT metamodel 2



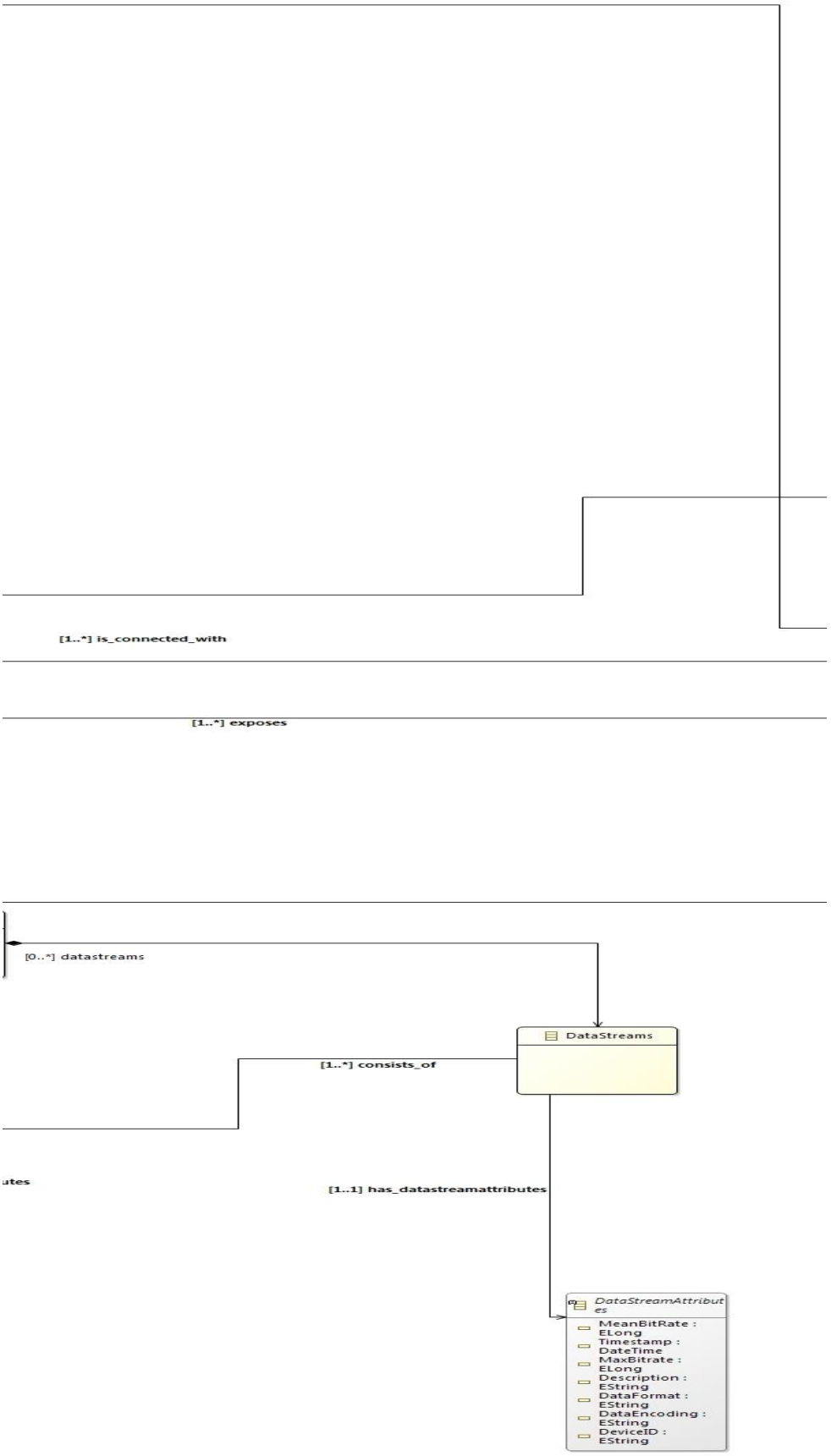
Σχήμα 4.3-Διάγραμμα IoT metamodel 3



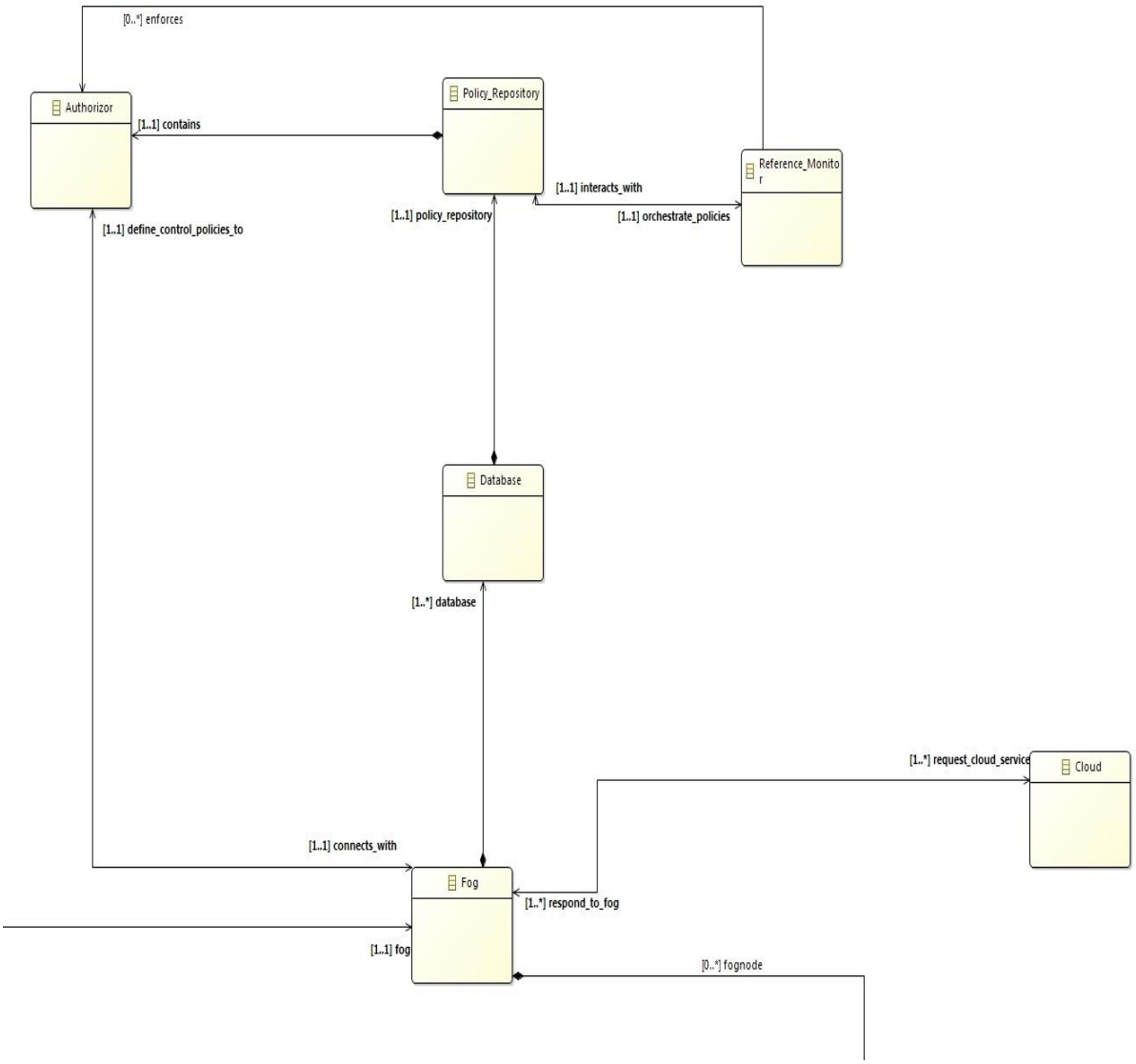
Σχήμα 4.4-Διάγραμμα IoT metamodel 4.1



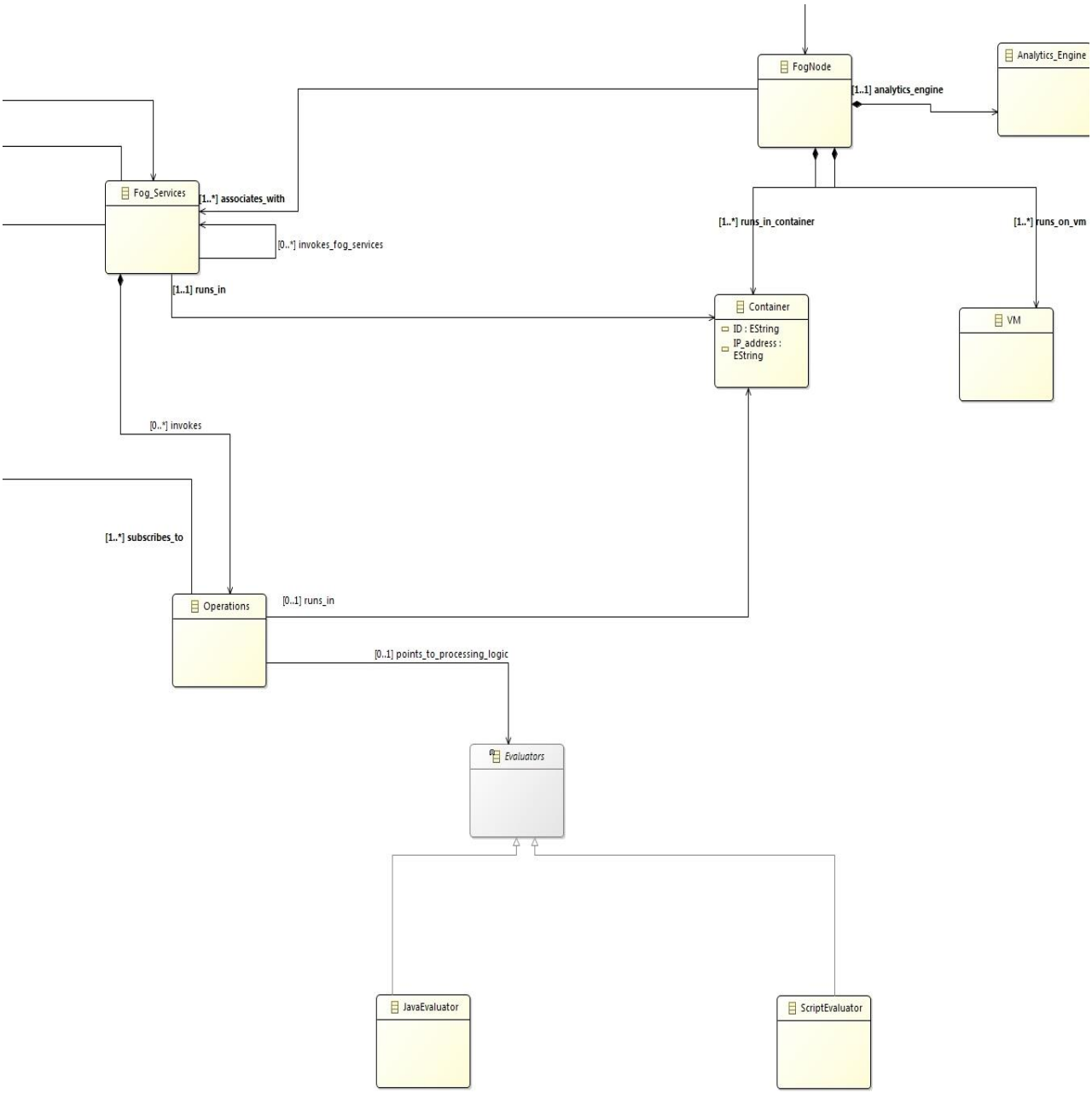
Σχήμα 4.5-Διάγραμμα IoT metamodel 4.2



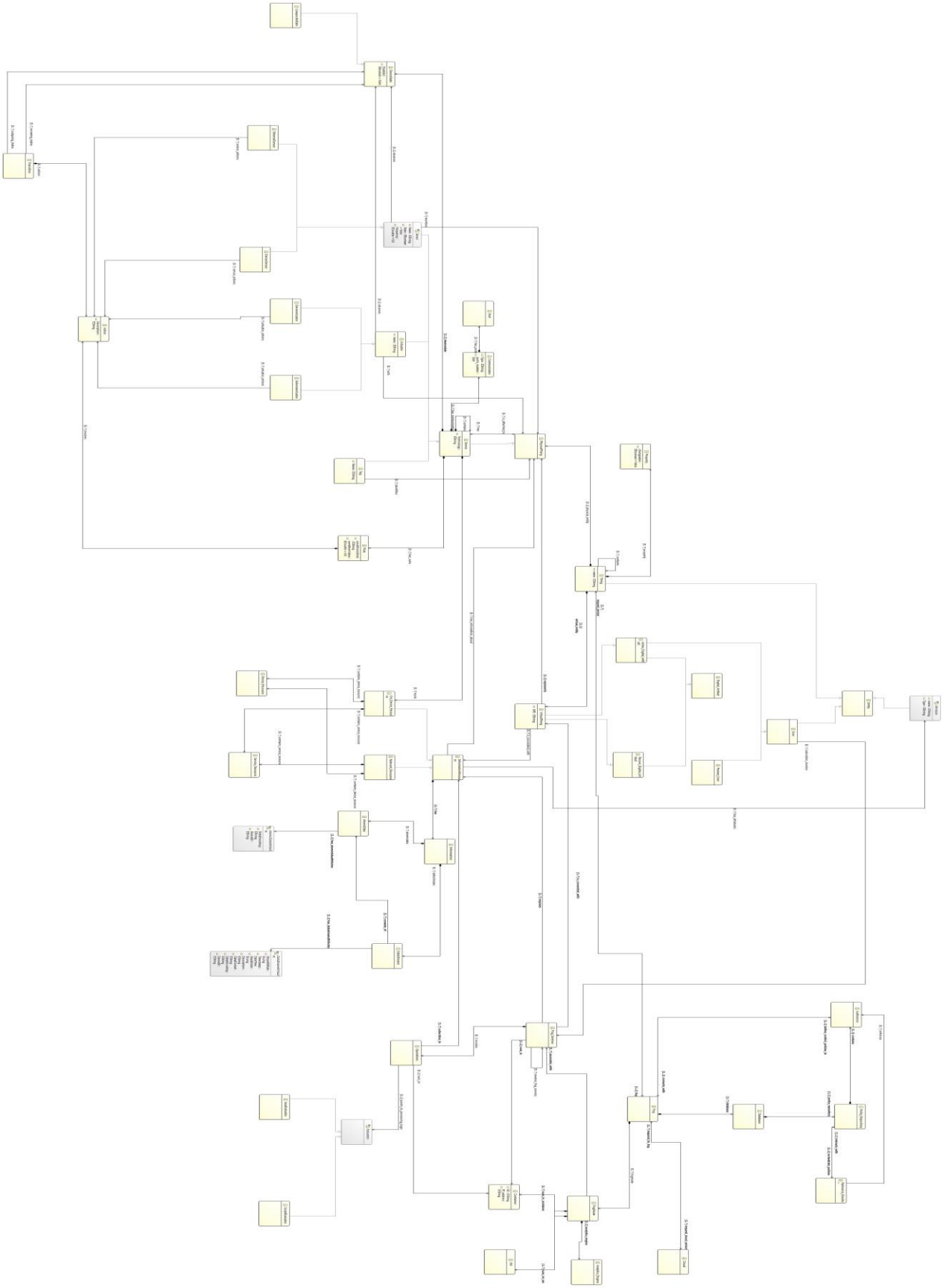
Σχήμα 4.6-Διάγραμμα IoT metamodel 5



Σχήμα 4.7-Διάγραμμα IoT metamodel 6.1



Σχήμα 4.8-Διάγραμμα IoT metamodel 6.2



Σχήμα 4.9-Διάγραμμα IoT metamodel

4.3 Εννοιολογικό Μοντέλο Δέντρων - Στόχων

Στην παρούσα διπλωματική χρησιμοποιήθηκε το εννοιολογικό μοντέλο των δέντρων - στόχων το οποίο αναπτύχθηκε σε ερευνητικές εργασίες τα προηγούμενα χρόνια[10][18][22] και ορίζει Δέντρα - Στόχων με τη μορφή Ή/ΚΑΙ αποσύνθεσης, συνεισφορών μεταξύ των στόχων και εφαρμογή του περιβάλλοντος-πλαισίου σε αυτούς. Επιπλέον, αναπτύχθηκαν κάποιες επεκτάσεις του εννοιολογικού μοντέλου για να καλύψει τις ανάγκες της παρούσας διπλωματικής και του ερευνητικού πεδίου στο οποίο δραστηριοποιείται. Πιο συγκεκριμένα, αναπτύχθηκαν επεκτάσεις που αφορούν την παραλληλοποίηση των στόχων, τη διαφορά που πρέπει να έχουν στο χρόνο εκτέλεσης στην μέγιστη όσο και στην ελάχιστη τιμή, τις λογικές και χρονικές προϋποθέσεις που πρέπει να έχουν υλοποιηθεί για την επίτευξη κάποιου στόχου. Η παραπάνω δομή των Δέντρων - στόχων επιλέχθηκε λόγω της απλότητας και πλούσιας εκφραστικότητας που παρέχει για την μοντελοποίηση πολύπλοκων στόχων του συστήματος, κάτι το οποίο κάνει ευκολότερη την ανάπτυξη, διαχείριση και βελτίωση του παρόντος λογισμικού για την εύρεση όλων των ακολουθιών από ατομικές ενέργειες που ικανοποιούν το Δέντρο - Στόχων.

Παρακάτω παρατίθενται και αναλύονται οι κλάσεις του εννοιολογικού μοντέλου των δέντρων - στόχων και στο τέλος του κεφαλαίου δίνεται το διάγραμμα κλάσεων του.

4.3.1 Κλάση Goal Model

Η κλάση αυτή είναι η ρίζα του μεταμοντέλου και αντιπροσωπεύει τη βασική δομή ενός δέντρου - στόχων. Μέσω αυτής της δομής μοντελοποιούνται οι στόχοι του συστήματος καθώς και οι μεταξύ τους σχέσεις. Έχει την ιδιότητα *name* για να διακρίνεται από άλλες δομές ίδιου τύπου και την ιδιότητα *description* για περιγραφή του μοντέλου σε μορφή κειμένου. Έχει 3 σχέσεις: την σχέση σύνθεσης *link[1..*]* με την κλάση Link που απεικονίζει τις συνδέσεις που ανήκουν στο μοντέλο που έχει αναπτυχθεί, τη σχέση *refers_to[1..1]* με την κλάση Task όπου υποδεικνύεται ο στόχος - ρίζα του δέντρου και τη σχέση σύνθεσης *contextcondition[0..*]* με την κλάση ContextCondition που απεικονίζονται οι συνθήκες του περιβάλλοντος - πλαισίου που επηρεάζουν το μοντέλο στόχων.

4.3.2 Κλάση Node

Αποτελεί ένα από τα βασικά στοιχεία του μεταμοντέλου αφού εκφράζει τους κόμβους που θα δημιουργηθούν στο δέντρο-στόχων. Έχει σαν ιδιότητα το *name* που λειτουργεί σαν πρωτεύον κλειδί για την διάκριση του κάθε κόμβου και την ιδιότητα *description* για να περιγραφούν σε κείμενο οι λόγοι ύπαρξης αυτού του κόμβου. Στην παρούσα υλοποίηση αναπτύσσονται 3 είδη κόμβων: GoalNode, Resource και ControlNode που αναλύονται παρακάτω.

4.3.3 Κλάση GoalNode

Είναι μία από τις 3 κατηγορίες κόμβων του μεταμοντέλου και αντιπροσωπεύει τους στόχους οι οποίοι τοποθετούνται στο δέντρο. Κάθε αντικείμενο τύπου GoalNode (εκτός του στόχου ρίζα) αποτελεί μέρος μιας αποσύνθεσης καθώς κάθε σύνθετος στόχος αποδομείται σε άλλους απλούστερους ή σύνθετους στόχους. Οι σύνθετοι και απλοί κόμβοι αποτελούν υποκατηγορίες της κλάσης GoalNode και απεικονίζονται ως Task και Action αντίστοιχα.

4.3.4 Κλάση Task

Αναπαριστούν τους σύνθετους στόχους του μοντέλου. Κάθε σύνθετος στόχος πρέπει να αναλυθεί σε άλλους απλούς ή σύνθετους στόχους το οποίο εκφράζεται στο μεταμοντέλο με την σχέση *hasdecomposition* πολλαπλότητας 1.

4.3.5 Κλάση Action

Είναι απλοί στόχοι του συστήματος οι οποίοι ουσιαστικά εκφράζουν απλές, ατομικές ενέργειες και συμβάλλουν στην ικανοποίηση μεγαλύτερων, συνθετότερων και πολυπλοκότερων στόχων(Tasks). Είναι πιθανό αυτές οι ατομικές ενέργειες να χρειάζονται δεδομένα από μια πηγή δεδομένων για να εκτελεστούν και κατ' επέκταση να ικανοποιήσουν τους απλούς στόχους που θα ικανοποιήσουν. Περιέχει την ιδιότητα *resource_enabled*, η οποία εκφράζει το αν μπορεί να ικανοποιηθεί ο στόχος ανάλογα με την διαθεσιμότητα των πηγών δεδομένων με τις οποίες συνδέεται.

4.3.6 Κλάση Decomposition

Είναι η κλάση που απεικονίζει την αποσύνθεση σύνθετων στόχων σε άλλους στόχους - παιδιά. Οι στόχοι - παιδιά μπορεί να είναι είτε σύνθετοι στόχοι (Task) είτε απλοί στόχοι (Action) το οποίο εκφράζεται στο μεταμοντέλο με τη σχέση *decomposed_to[1..*]*. Έχει σαν ιδιότητες τον τύπο της αποσύνθεσης (type) η οποία παίρνει τις τιμές AND και OR, όπως έχει οριστεί στον τύπο πεδίου DecompositionType.

4.3.7 Κλάση Resource

Η κλάση αυτή αναπαριστά πηγές δεδομένων από τις οποίες οι απλοί στόχοι(Action) αντλούν δεδομένα για να εκτελέσουν τις ατομικές ενέργειες που είναι απαραίτητες για την ικανοποίηση του στόχου. Έχει σαν ιδιότητα μια Boolean τιμή η οποία εκφράζει την διαθεσιμότητα της πηγής δεδομένων.

4.3.8 Κλάση Control Node

Απεικονίζει μια 3η κατηγορία κόμβων η οποία δεν αφορά στόχους που πρέπει να επιτευχθούν αλλά χρησιμοποιούνται για να εμπλουτίσουν την εκφραστικότητα του δέντρου - στόχων και να προσθέσουν πληροφορίες για το μοντέλο και το σύστημα που αυτό αναφέρεται αλλά και για τις σχέσεις μεταξύ των στόχων του δέντρου.

4.3.9 Κλάση Parallel Node

Είναι υποκλάση του Control Node και σε αυτόν συνδέονται κόμβοι - στόχοι οι οποίοι τρέχουν παράλληλα. Έχει σαν ιδιότητες τον αριθμό των κόμβων που πρέπει να τρέξουν παράλληλα (*NumberofNodes)* και το μέγιστο χρονικό διάστημα στο οποίο πρέπει να έχουν ολοκληρωθεί όλες οι εκτελέσεις των παράλληλων κόμβων.

4.3.10 Κλάση Link

Αποτελεί θεμελιώδες κομμάτι του μεταμοντέλου καθώς εκφράζει τις σχέσεις που έχουν μεταξύ τους οι κόμβοι - στόχοι του δέντρου. Στην παρούσα υλοποίηση αναπτύσσονται 2 κατηγορίες συνδέσεων: οι συνδέσεις οι οποίες εκφράζουν την εξαρτήσεις των κόμβων που συνδέονται μεταξύ τους και οι συνδέσεις συνεισφοράς στην ικανοποίηση κάποιου άλλου στόχου.

4.3.11 Κλάση Dependency Link

Συνδέει κόμβους - στόχους που έχουν εξάρτηση μεταξύ τους είτε χρονική είτε λογική. Περιλαμβάνει τις εξής υποκατηγορίες: LogicalPrecondition, TemporalLink, ParallelLink και ResourceDependency για τις οποίες ακολουθεί ανάλυση παρακάτω.

4.3.12 Κλάση LogicalPrecondition

Η σύνδεση τέτοιου τύπου εκφράζει την εξάρτηση ενός κόμβου ότι για να εκτελεστεί και να ικανοποιηθεί ο κόμβος - στόχος θα πρέπει προηγουμένως να έχει ικανοποιηθεί ο κόμβος - πηγή. Συνδέεται με τις σχέσεις *from_goalnode [1..1]* και *to_goalnode[1..1]* για να καθοριστούν οι κόμβοι που έχουν μεταξύ τους μια λογική εξάρτηση.

4.3.13 Κλάση TemporalLink

Η σύνδεση αυτή καθορίζει τις χρονικές εξαρτήσεις που έχουν μεταξύ τους οι κόμβοι-στόχοι του δέντρου. Χωρίζεται σε 2 κατηγορίες: TimeoutLink και TimeDifferenceLink.

4.3.14 Κλάση TimeoutLink

Αναπαριστά την χρονική εξάρτηση που έχουν μεταξύ τους 2 απλοί κόμβοι του δέντρου (Action) όσον αφορά τη μέγιστη χρονική διαφορά που μπορούν να έχουν από την αρχή εκτέλεσης του ενός κόμβου μέχρι την ολοκλήρωση εκτέλεσης του άλλου κόμβου. Η μέγιστη αυτή διαφορά αναπαρίσταται με την ιδιότητα *TimeoutPeriod* και έχει σαν μονάδα μέτρησης τα δευτερόλεπτα. Συνδέεται με 2 σχέσεις οι οποίες εκφράζουν τους 2 απλούς κόμβους - στόχους.

4.3.15 Κλάση TimeDifferenceLink

Εκφράζει την ελάχιστη διαφορά που πρέπει να έχουν 2 κόμβοι για την εκτέλεση τους, δηλαδή ο κόμβος στόχος της σύνδεσης θα πρέπει να εκτελεστεί τουλάχιστον μετά από κάποιο χρονικό διάστημα μετά από την ολοκλήρωση του κόμβου πηγής της σύνδεσης. Αυτό το χρονικό διάστημα καθορίζεται με την ιδιότητα *TimeDifference* και έχει σαν μονάδα μέτρησης τα δευτερόλεπτα.

4.3.16 Κλάση ParallelLink

Η κλάση αυτή συνδέει κόμβους του δέντρου που πρέπει να τρέξουν παράλληλα με έναν κόμβο τύπου ParallelNode. Έχει τις σχέσεις *from[1..1]*, η οποία αναφέρεται σε έναν κόμβο τύπου GoalNode, και *to[1..1]* η οποία πραγματοποιεί τη σύνδεση με τον ένα παράλληλο κόμβο, στον οποίο συνδέονται όλοι οι κόμβοι της ομάδας που πρέπει να τρέξουν παράλληλα.

4.3.17 Κλάση ResourceDependency

Η κλάση αυτή συνδέει απλούς κόμβους στόχους(Action) με πηγές δεδομένων (Resource) και δείχνει την εξάρτηση της εκτέλεσης αυτών από τη διαθεσιμότητα κάποιων πηγών δεδομένων.

4.3.18 Κλάση Contribution

Η κλάση Contribution εκφράζει τη συνεισφορά που έχει η ικανοποίηση ή η άρνηση ενός κόμβου - στόχου στην ικανοποίηση ή την άρνηση ενός άλλου κόμβου - στόχου. Ο τύπος της συνεισφοράς καθορίζεται από την ιδιότητα *type* η οποία είναι τύπου *ContributionType*.

4.3.19 Κλάση ContributionType

Ορίζει τους 4 δυνατούς τύπους συνεισφορών που υπάρχουν στο παρόν μεταμοντέλο:

- PPS: η ικανοποίηση του στόχου - πηγή συνεισφέρει θετικά στην ικανοποίηση του στόχου - προορισμού.
- MMS: η ικανοποίηση του στόχου - πηγή συνεισφέρει αρνητικά στην ικανοποίηση του στόχου – προορισμού.

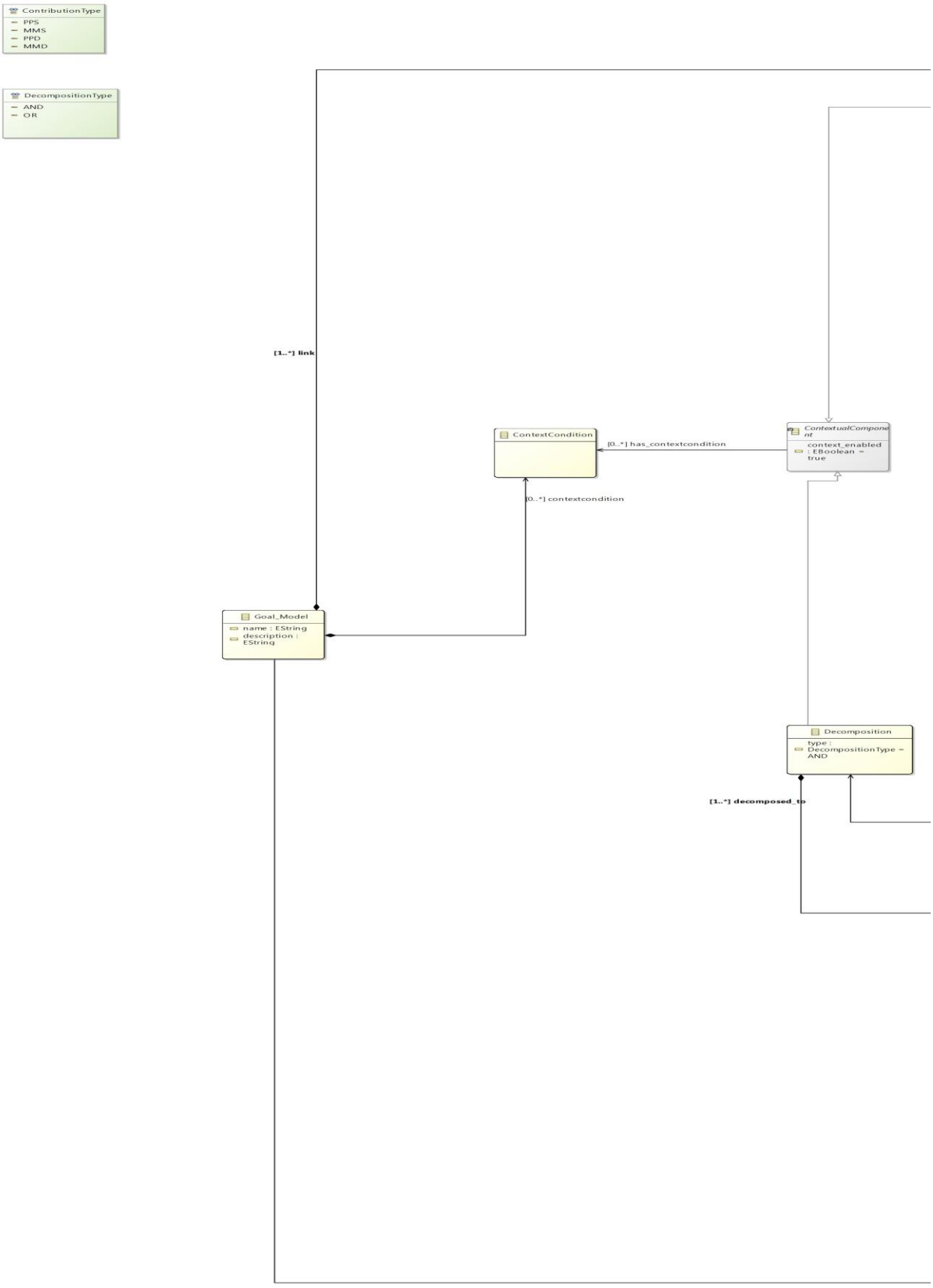
- PPD: η μη ικανοποίηση του στόχου - πηγή συνεισφέρει αρνητικά στην ικανοποίηση του στόχου - προορισμού.
- MMD: η μη ικανοποίηση του στόχου - πηγή συνεισφέρει θετικά στην ικανοποίηση του στόχου - προορισμού

4.3.20 Κλάση ContextualComponent

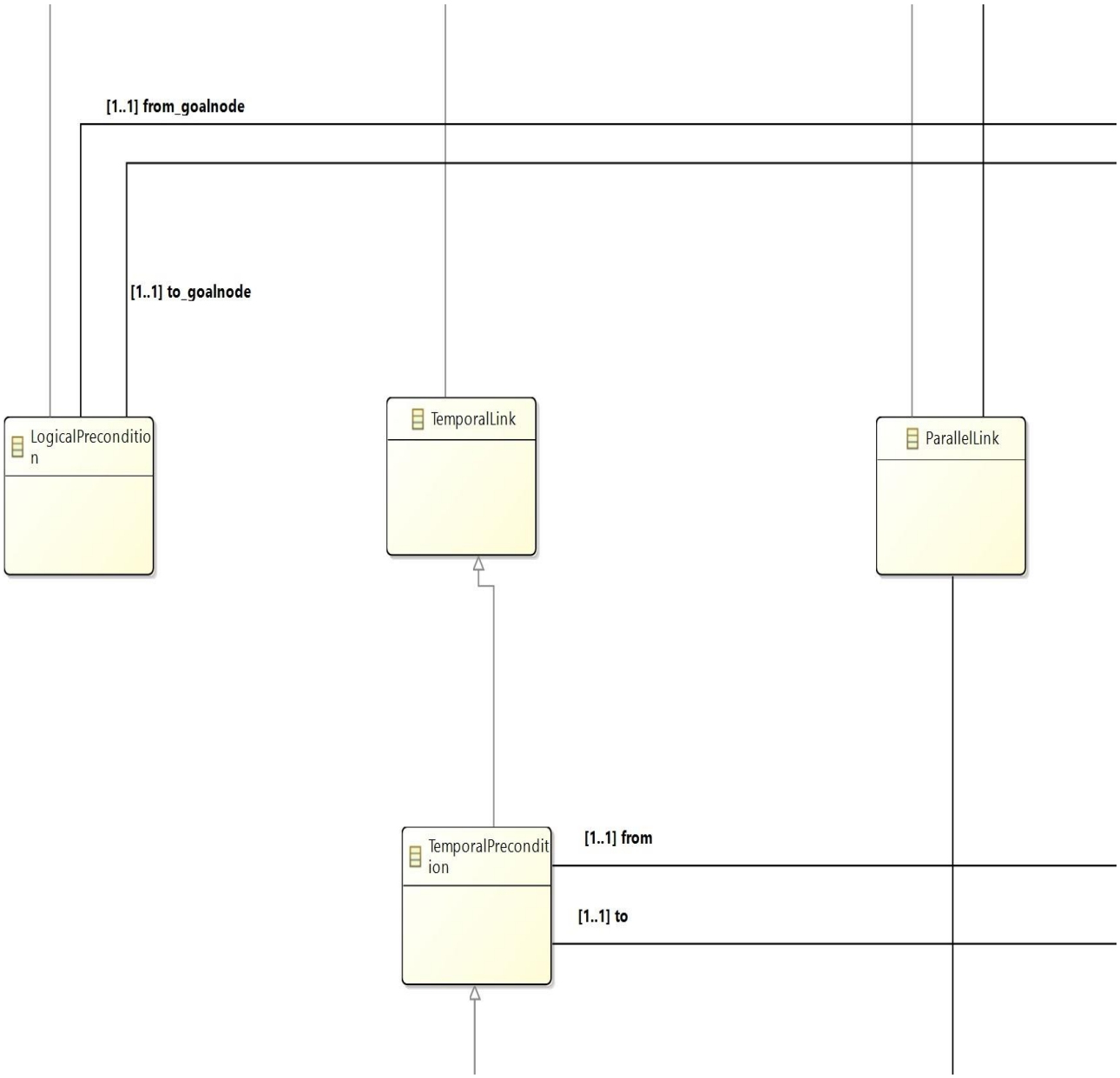
Αφηρημένη κλάση και περιλαμβάνει τα Links και τα Nodes του μεταμοντέλου ενώ συνδέεται με την κλάση ContextCondition που καθορίζει το πλαίσιο και τις συνθήκες μέσα στις οποίες τρέχει το μοντέλο, όπου ενεργοποιούνται και απενεργοποιούνται Nodes και Links με βάση το context.

4.3.21 Κλάση TemporalPrecondition

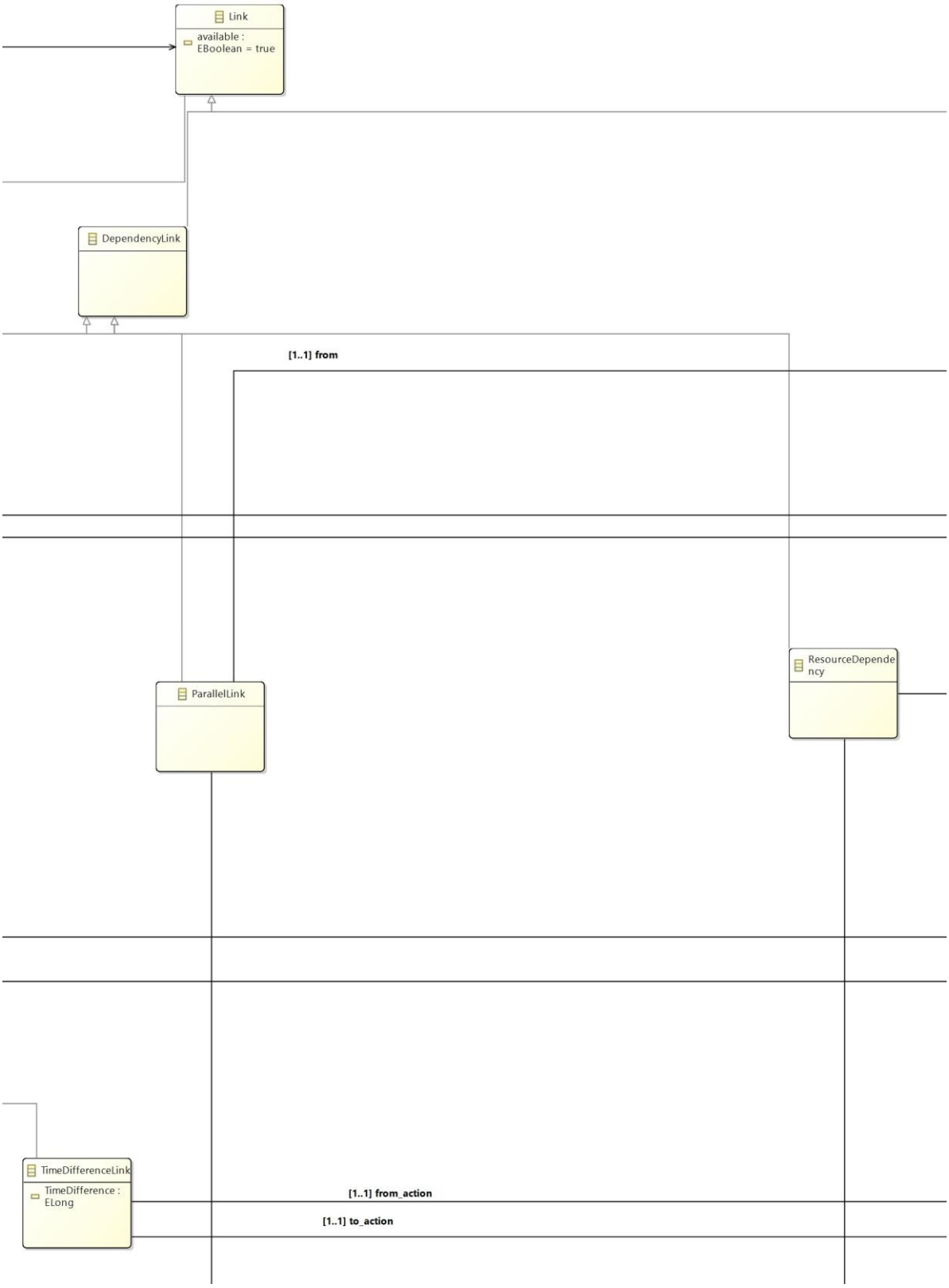
Η σύνδεση τέτοιου τύπου εκφράζει την εξάρτηση ενός κόμβου ότι για να εκτελεστεί και να ικανοποιηθεί ο κόμβος - στόχος θα πρέπει να έχει προηγηθεί η ικανοποίηση του κόμβου - πηγή. Συνδέεται με τις σχέσεις *from [1..1]* και *to [1..1]* για να καθοριστούν οι κόμβοι που έχουν μεταξύ τους μια λογική εξάρτηση.



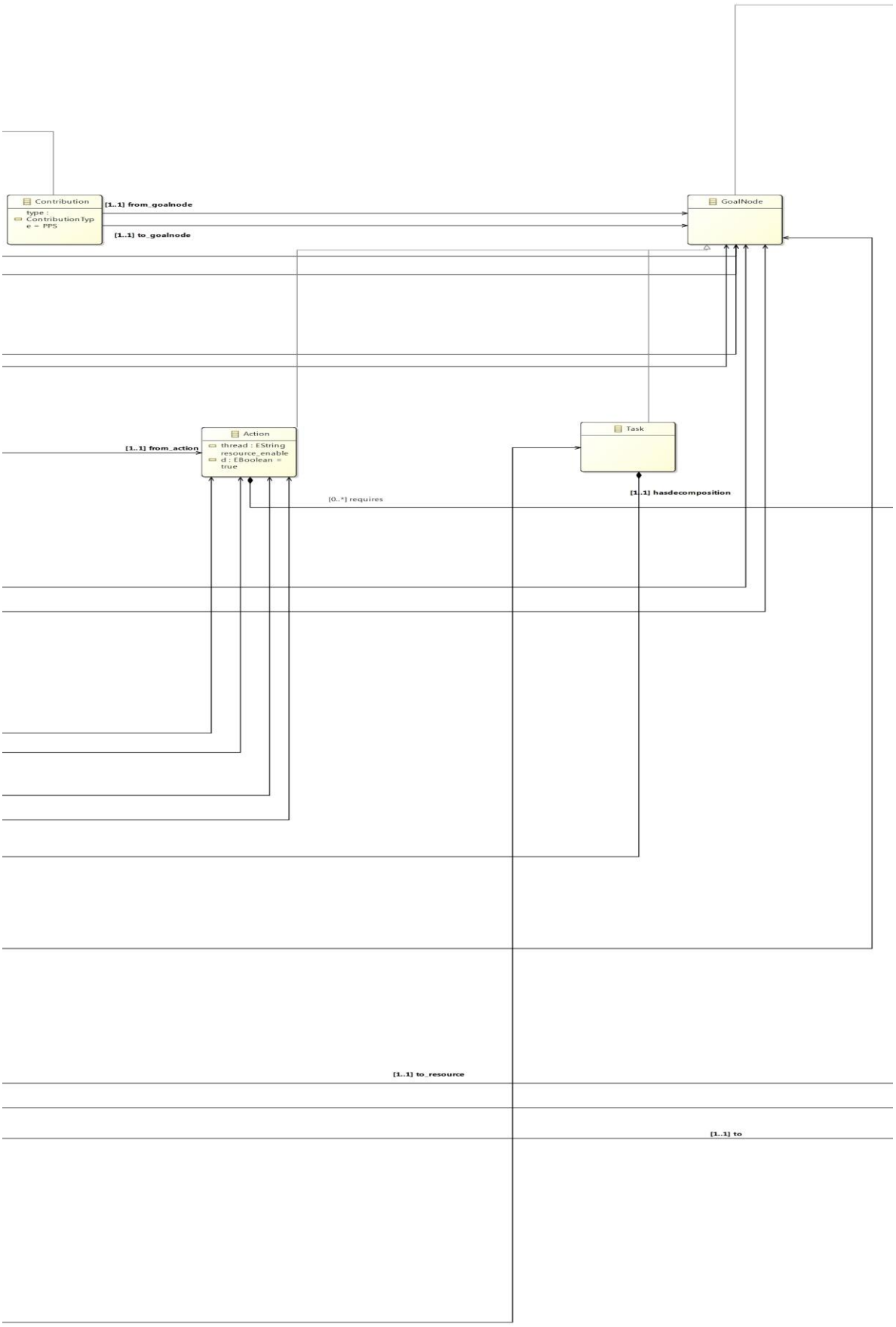
Σχήμα 4.10-Διάγραμμα goal metamodel 1



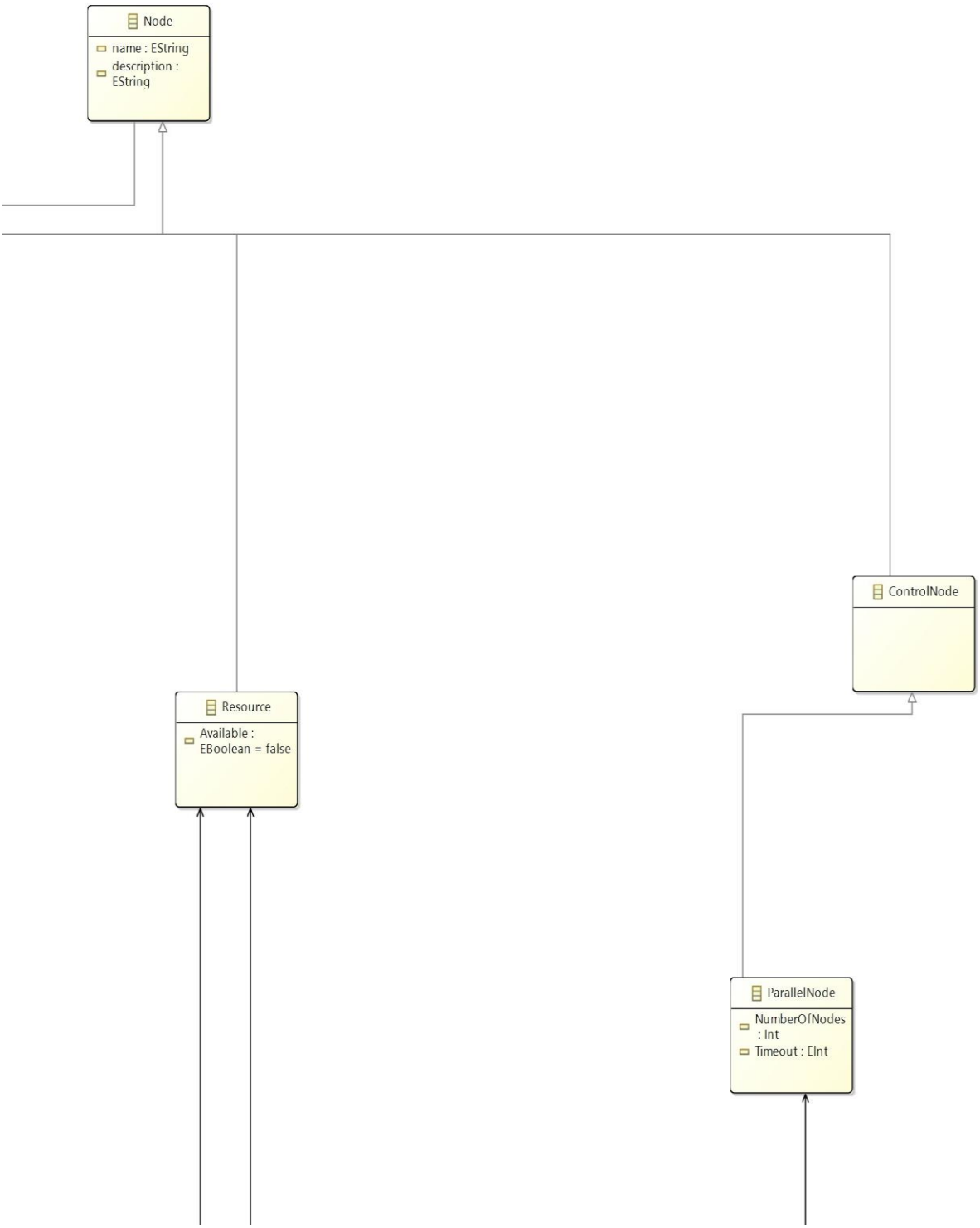
Σχήμα 4.11-Διάγραμμα goal metamodel 2



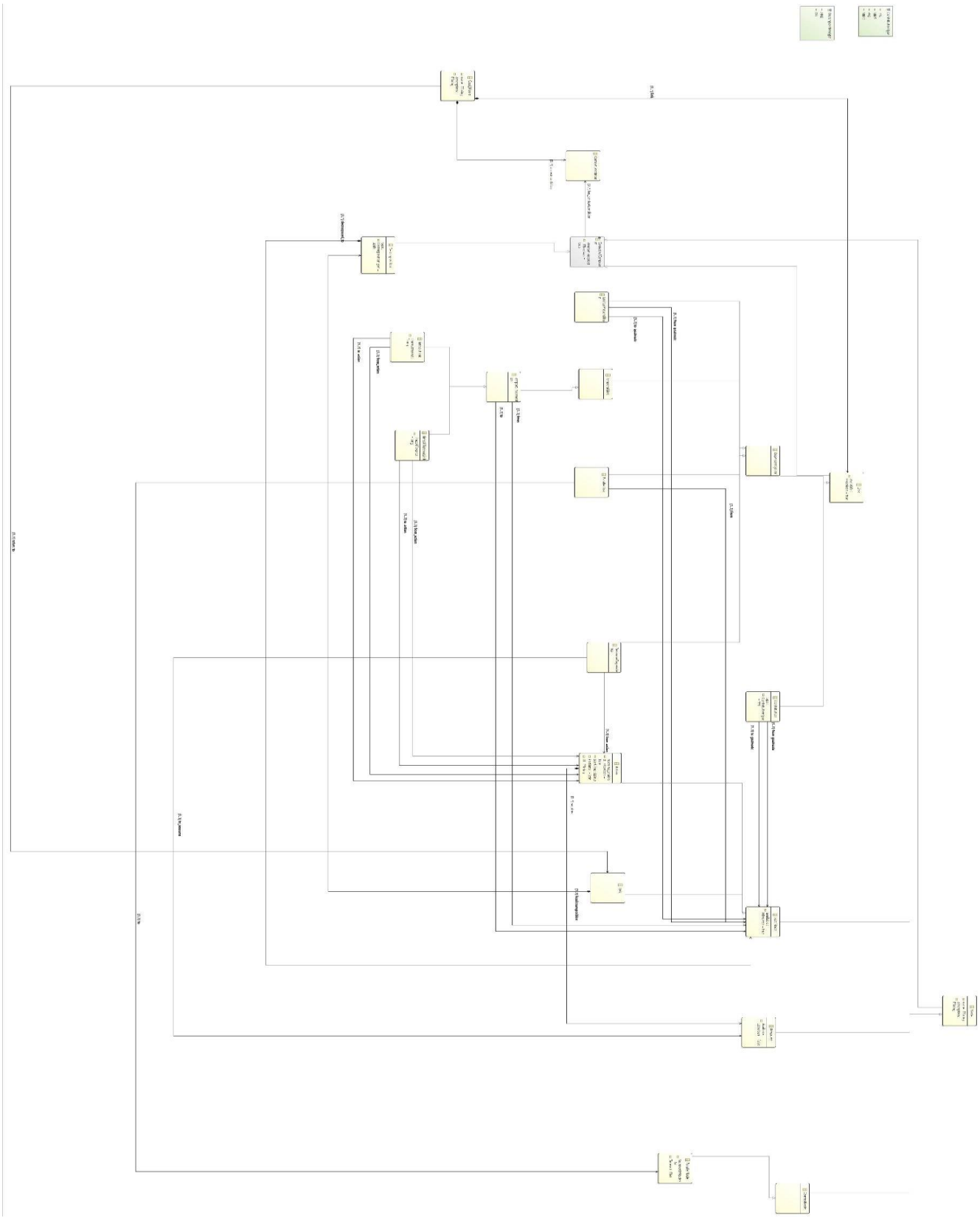
Σχήμα 4.12-Διάγραμμα goal metamodel 3



Σχήμα 4.13-Διάγραμμα goal metamodel 4



Σχήμα 4.14-Διάγραμμα goal metamodel 5



Σχήμα 4.15-Διάγραμμα goal metamodel

4.4 Εννοιολογικό Μοντέλο Γράφου Εξάρτησης Ενεργειών

Για την εύρεση των πλάνων που αφορούν τις ενέργειες που πρέπει να γίνουν (κλάση Action του goal metamodel) και με ποια σειρά, χρησιμοποιήθηκε η έννοια του Γράφου Εξάρτησης Ενεργειών (ADG)[9] ο οποίος απεικονίζει με εύγλωττο τρόπο αυτήν ακριβώς την ακολουθία. Ο γράφος αυτός προκύπτει από κάποιους κανόνες που εφαρμόζονται στο δέντρο - στόχων και εξαρτάται από τις εξαρτήσεις που έχουν οι κόμβοι μεταξύ τους καθώς και το περιβάλλον - πλαίσιο στο οποίο λειτουργεί το σύστημα.

4.4.1 Κλάση ADG

Είναι ο κόμβος - ρίζα του μεταμοντέλου και αποτελεί την απεικόνιση ολόκληρου του γράφου. Αποτελείται από κόμβους (Node) και συνδέσεις (Link) ενώ διαθέτει την ιδιότητα *ParallelStatus* όπου δηλώνεται αν ο γράφος αποτελεί μέρος κάποιας παραλληλίας.

4.4.2 Κλάση Link

Αναπαριστά τις συνδέσεις που υπάρχουν στον γράφο ενώ οι σχέσεις *from[1..1]* και *to[1..1]* με την κλάση Node απεικονίζουν τους 2 κόμβους που δημιουργούν την σύνδεση.

4.4.3 Κλάση Node

Η κλάση αυτή αντιπροσωπεύει τους κόμβους που υπάρχουν στον γράφο. Χωρίζονται σε 3 κατηγορίες: ActionNode, SplitNode και JoinNode

4.4.4 Κλάση ActionNode

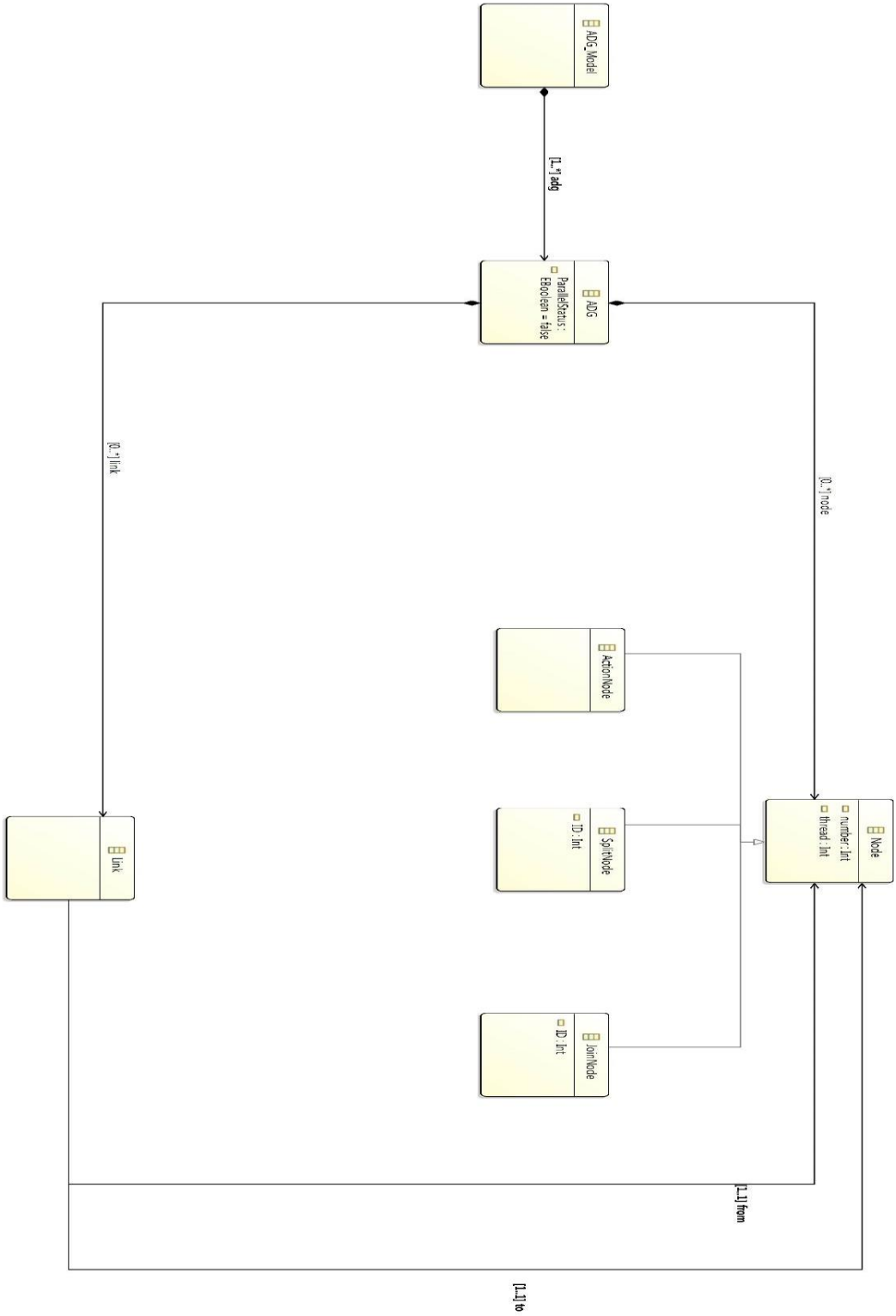
Αποτελεί τη μετασχηματισμένη κλάση Action του goal metamodel και απεικονίζει απλές ενέργειες του συστήματος. Το κάθε αντικείμενο τύπου ActionNode έχει ένα αντίστοιχο αντικείμενο τύπου Action στο goal metamodel. Οι κόμβοι αυτοί δημιουργούνται κατά τη διάρκεια εφαρμογής των κανόνων μετασχηματισμού, όπως περιγράφεται στο κεφάλαιο 5. Κάθε τέτοιο αντικείμενο διαθέτει την ιδιότητα *name* που αποτελεί το πρωτεύον κλειδί για κάθε κόμβο.

4.4.5 Κλάση SplitNode

Είναι ο κόμβος - ρίζα του γράφου και δηλώνει την αφετηρία της ακολουθίας ενεργειών που έχει τοποθετηθεί στον γράφο. Επίσης, αποτελεί αφετηρία για τα διάφορα νήματα από ενέργειες που πρέπει να εκτελεστούν παράλληλα.

4.4.6 Κλάση JoinNode

Δηλώνει τον κόμβο-τερματισμό του γράφου για την περιεχόμενη ακολουθία των ενεργειών ενώ σε αυτόν τον κόμβο τερματίζουν τα πιθανά νήματα των ενεργειών που εκτελούνται παράλληλα.



Σχήμα 4.16-Διάγραμμα ADG metamodel

Κεφάλαιο 5:Αλγόριθμοι Μετασχηματισμού και εύρεσης Ενεργειών

Στο παρόν κεφάλαιο παρουσιάζονται και αναλύονται οι αλγόριθμοι μετασχηματισμού που αναπτύχθηκαν και χρησιμοποιήθηκαν για τον μετασχηματισμό του δέντρου - στόχων στους διάφορους Γράφους Εξάρτησης Ενεργειών. Στους γράφους αυτούς στη συνέχεια γίνεται μια τοπολογική ταξινόμηση για να βρεθούν οι κατάλληλες ακολουθίες ενεργειών που ικανοποιούν το δέντρο - στόχων του συστήματος. Επιπλέον, παρουσιάζεται και ο αλγόριθμος - πελάτη, ο οποίος κάνει την απαραίτητη προεργασία στο δέντρο - στόχων, ενορχηστρώνει τους κανόνες μετασχηματισμού και στη συνέχεια εξάγει όλες τι δυνατές ακολουθίες ενεργειών. Παράλληλα, θα παρουσιαστούν μικρά παραδείγματα εφαρμογής των παραπάνω αλγορίθμων για μια πιο εποπτική και πλήρη εικόνα των αποτελεσμάτων αυτών. Τέλος, θα δοθούν 2 παραδείγματα δέντρων - στόχων πραγματικών συστημάτων, όπου θα παρουσιαστούν τα αποτελέσματα που εξάγονται.

5.1 Κανόνες Μετασχηματισμού

Θα αναλυθούν όλοι οι κανόνες μετασχηματισμού που αναπτύχθηκαν και επεκτάθηκαν[9] για την δημιουργία του Γράφου Εξάρτησης Ενεργειών και είναι οι:

- Κανόνας Συνεισφοράς (Contribution Rule)
- Κανόνας Παραλληλισμού (Parallel Rule)
- Κανόνας Λογικής Προϋπόθεσης (LogicalPrecondition Rule)
- Κανόνας Χρονικής Προϋπόθεσης (TemporalPrecondition Rule)
- Κανόνας Εξάρτησης Πηγών Δεδομένων (Resource Dependency Rule)
- Κανόνας Χρονικής Διαφοράς (Timeout Rule)
- Κανόνας Χρονικής Απόστασης (TimeDifference Rule)

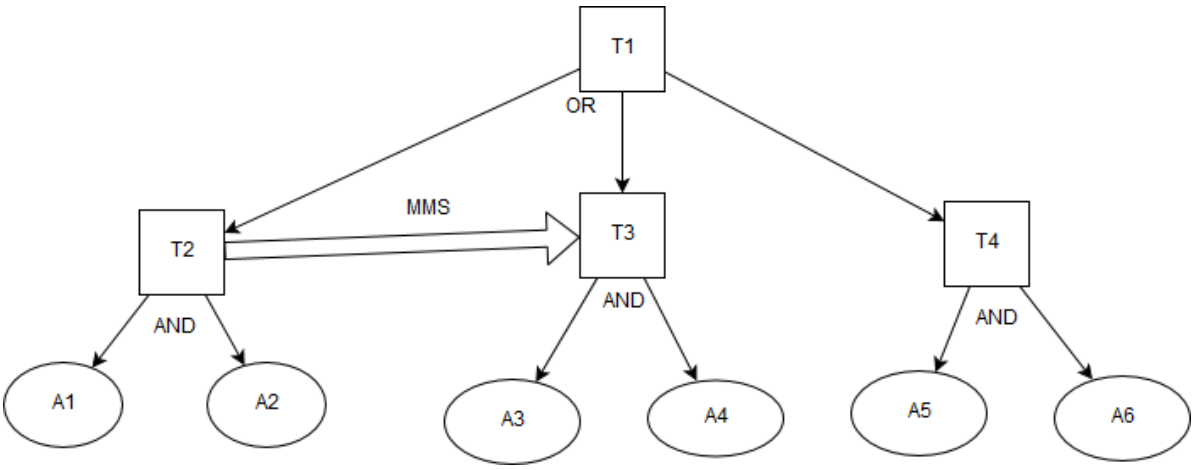
5.1.1 Κανόνας Συνεισφοράς

Ο κανόνας αυτός έχει σαν σκοπό να μεταφέρει στον Γράφο Εξάρτησης Ενεργειών (ADG) τις σχέσεις συνεισφοράς που υπάρχουν στο δέντρο - στόχων. Για παράδειγμα, αν υπάρχει μια σύνδεση συνεισφοράς MMS στο δέντρο - στόχων τότε οι ατομικές ενέργειες που ικανοποιούν τον στόχο - πηγή δεν θα πρέπει να βρίσκονται στο ίδιο πλάνο ενεργειών με τις ατομικές ενέργειες που ικανοποιούν το στόχο - κατάληξη. Ο κανόνας αυτός εφαρμόζεται και έχει ουσιαστικό νόημα μόνο για τις συνεισφορές MMS και PPD καθώς για τις υπόλοιπες δύο οι περιορισμοί που θέτουν στην δημιουργία των ADG και των πλάνων ενεργειών, λαμβάνονται υπόψη στον αλγόριθμο - πελάτη, όπως θα αναλυθεί παρακάτω. Για την απεικόνιση των περιορισμών που δημιουργούν οι σχέσεις συνεισφοράς κάθε στόχος - κατάληξη περιέχει μία λίστα από ατομικές ενέργειες που ικανοποιούν το στόχο - πηγή και για τις οποίες γίνεται έλεγχος, στη συνέχεια, για να διαπιστωθεί αν στο ADG, στο οποίο βρίσκεται ο στόχος - κατάληξη, οι

περιορισμοί που έχουν τεθεί από τις σχέσεις συνεισφοράς κάνουν απαγορευτική την εκπόνηση πλάνου ενεργειών από αυτό.
Πιο αναλυτικά ο αλγόριθμος είναι:

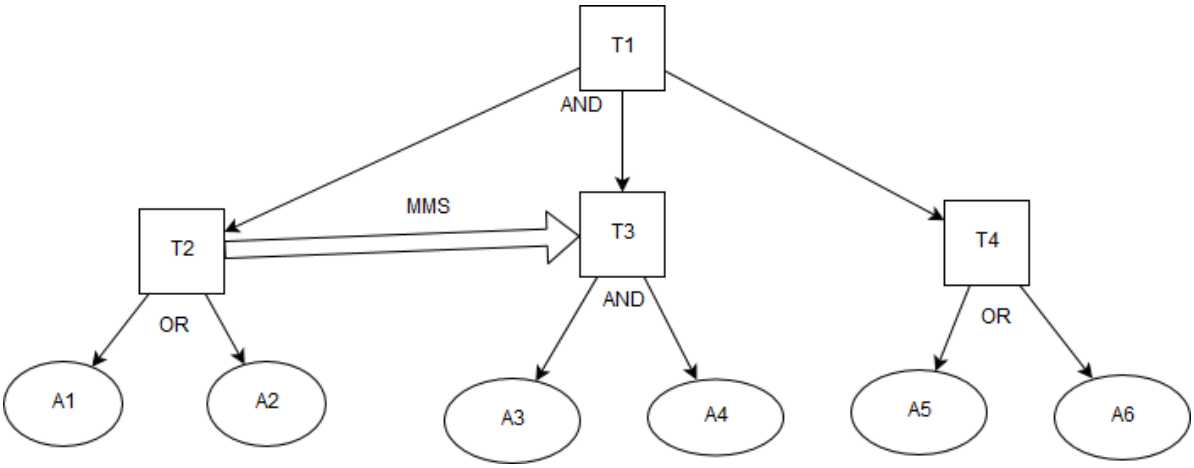
```
for all C in contribution links do
  if C is not available then
    continue
  end if
  T1 = C.get-from-node
  T2 = C.get-to-node
  if C.get − type == MMS then
    for all satisfiable − list in T2.getList do
      for all a in satisfiable-list do
        a.set-to-be-examined
        put every satisfiable-list of T1 in a.exist-node-list
      end for
    end for
  end if
  if C.get − type == PPD then
    for all satisfiable − list in T2.getList do
      for all a in satisfiable-list do
        a.set-to-be-examined
        put every satisfiable-list of T1 in a.non-exist-node-list
      end for
    end for
  end if
end for
```

Τέλος, παρουσιάζονται 2 μικρές εφαρμογές του κανόνα για εποπτικότερη εικόνα της εφαρμογής του:



Παράδειγμα 5.1

Στο παραπάνω παράδειγμα, δεν θα πρέπει κανένα πλάνο ενεργειών που ικανοποιεί το σύνθετο στόχο T3(A3,A4) να ικανοποιεί και το στόχο T2(A1,A2). Επομένως, μετά την εφαρμογή του κανόνα οι στόχοι A3 και A4 θα έχουν μία λίστα με τους στόχους A1 και A2 για να μπορεί να γίνει έλεγχος και να απορριφθούν ADG που περιλαμβάνουν τους στόχους A1,A2,A3,A4.



Παράδειγμα 5.2

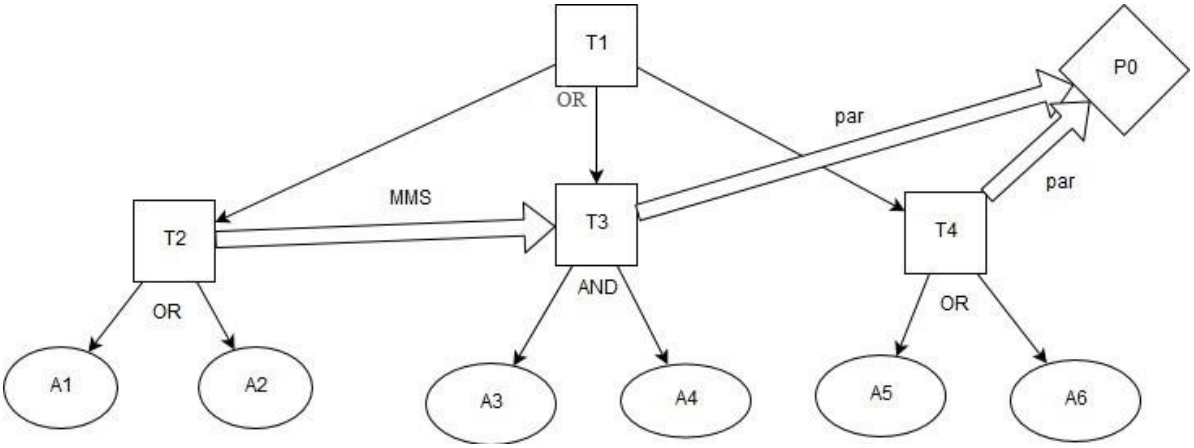
Στο Παράδειγμα 2 δεν είναι δυνατή η δημιουργία κανενός πλάνου ενεργειών λόγω των περιορισμών της σχέσης συνεισφοράς. Αυτό συμβαίνει γιατί κάθε πλάνο που θα εκπονηθεί θα περιλαμβάνει όλους τους απλούς στόχους, πράγμα απαγορευτικό καθώς η ικανοποίηση του σύνθετου στόχου T2 έχει ως συνέπεια την άρνηση του σύνθετου στόχου T3.

5.1.2 Κανόνας Παραλληλισμού

Αυτός ο κανόνας δημιουργεί συλλογές Γράφων, όπου σε κάθε συλλογή βρίσκονται όλοι εκείνοι οι Γράφοι Εξάρτησης Ενεργειών που ικανοποιούν ομάδες στόχων οι οποίες πρέπει να εκτελεστούν παράλληλα. Δηλαδή, κάθε παράλληλος κόμβος που υπάρχει στο μοντέλο - στόχων δημιουργεί, με την εφαρμογή του παραπάνω κανόνα, μια συλλογή από γράφους (ADGCollection).
Ο κανόνας υπολογίζει όλα τα δυνατά σύνολα ενεργειών (Action) που ικανοποιούν τους στόχους που είναι συνδεδεμένοι με έναν παράλληλο κόμβο και για κάθε συνδυασμό δημιουργείται ένας γράφος ο οποίος τοποθετείται στην αντίστοιχη συλλογή. Πιο αναλυτικά:

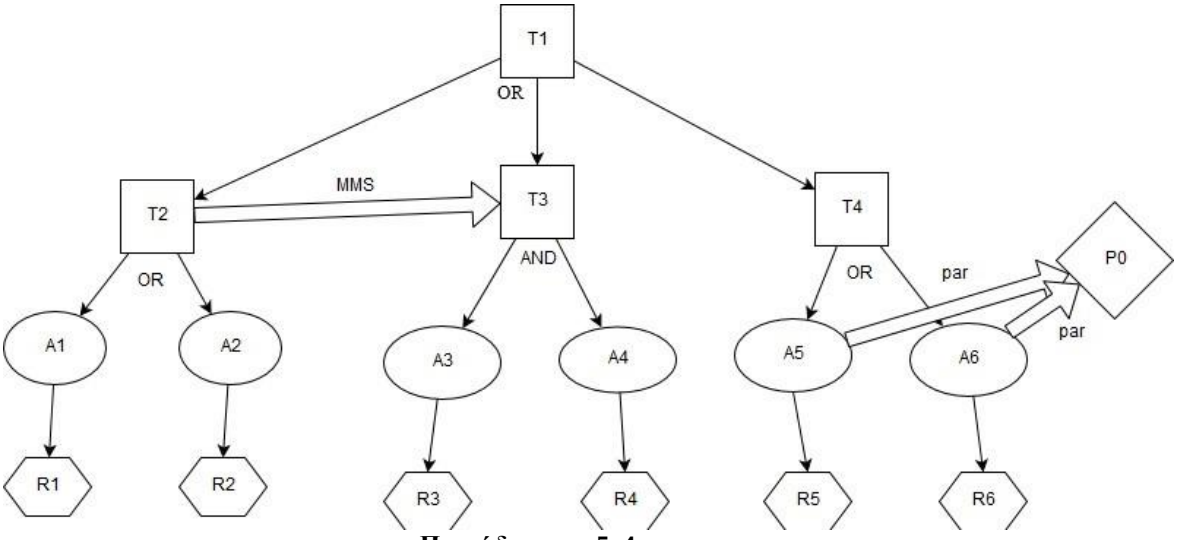
```
for all Pi in parallel nodes do
   Create ActionNode Pi
   Put ActionNode Pi in every ADG in parent ADG-Collection
   Create ADG-Collection with name Pi
   Find all combinations of Action that satisfy the GoalNodes connected to
   parallel node Pi
   for all combination in combinations do
      Create ADG
      if ActionsofADG cannot be executed due to contribution rules then
         continue
      end if
      Put the ADG in appropriate ADG-Collection
   end for
end for
```

Κάποιες χαρακτηριστικές εφαρμογές του κανόνα παρουσιάζονται παρακάτω:



Παράδειγμα 5.3

Στο παραπάνω παράδειγμα θα δημιουργεί μία ακόμα συλλογή από γράφους η οποία θα έχει σαν ID το όνομα του παράλληλου κόμβου που «εκπροσωπεί». Μέσα σε αυτήν τη συλλογή θα υπάρχουν 3 ADG τα οποία ικανοποιούν τους κόμβους T3 και T4 που είναι συνδεδεμένους με τον P0. Επομένως, στη συλλογή γράφων P0 θα υπάρχουν 3 ADG, τα οποία περιλαμβάνουν τις ενέργειες {A3,A4,A5},{A3,A4,A6}{A3,A4,A5,A6}.



Παράδειγμα 5.4

Στο παράδειγμα 4 τα ADG που θα δημιουργηθούν στην συλλογή P0 θα περιέχουν τις εξής ενέργειες:{A5},{A6},{A5,A6}. Αυτοί οι συνδυασμοί καλύπτουν όλες τις επιλογές που ικανοποιούν τον κόμβο - στόχο T4.

5.1.3 Κανόνας Λογικής Προϋπόθεσης

Ο κανόνας αυτός χρησιμοποιεί τις συνδέσεις μεταξύ στόχων που έχουν καθοριστεί στο δέντρο - στόχων για να απεικονίσει στον Γράφο Εξάρτησης Ενεργειών τη λογική σειρά με την οποία πρέπει να εκτελεστούν οι ατομικές ενέργειες (Action), όπως έχει καθοριστεί με τις παραπάνω συνδέσεις. Δημιουργεί και σβήνει δηλαδή συνδέσεις στον Γράφο για να απεικονιστεί μια λογική αλληλουχία εκτέλεσης, η οποία θα καθοριστεί επακριβώς σε επόμενα βήματα του αλγορίθμου με την τοπολογική ταξινόμηση. Πιο συγκεκριμένα τα βήματα του κανόνα είναι:

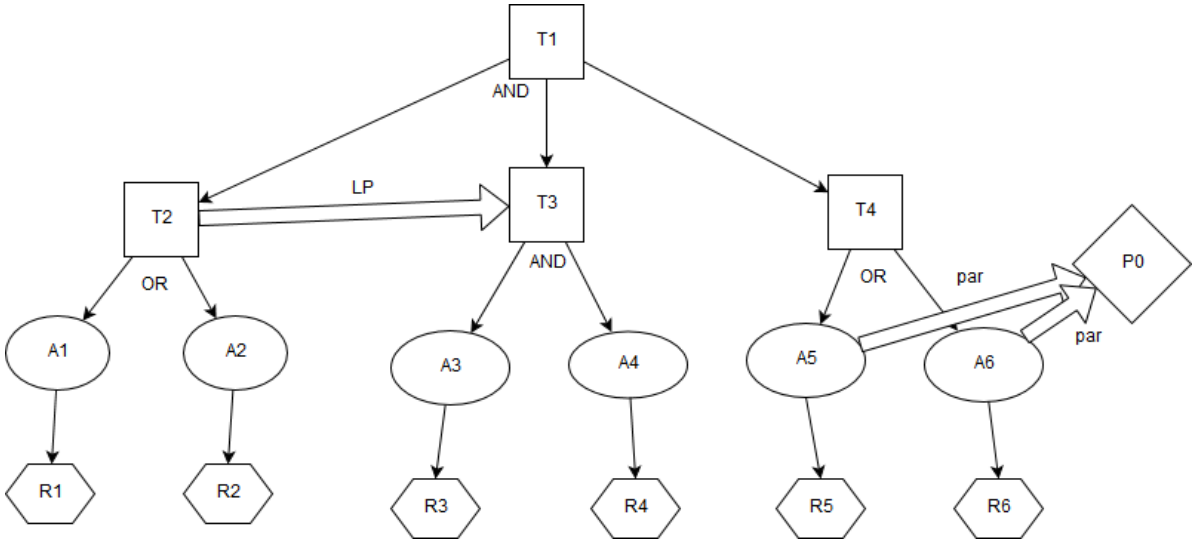
1)$TaskT1 \overset{LP}{\rightarrow} TaskT2$:
$\forall a \in T1, \forall b \in T2$: if exists link $a \rightarrow JPk$ for some k,remove that link //if exists link $SPk \rightarrow b$ for some k,remove that link
i)if T1 only has a parallel link with Pi: $\forall b \in T2 : Pi \rightarrow b$
ii)if T2 only has a parallel link with Pi: $\forall a \in T1 : a \rightarrow Pi$
iii)if T1 and T2 have parallel link with Pi,Pj respectively:
$Pi \rightarrow Pj$
iv)if none has parallel link: $\forall a \in T1, \forall b \in T2 : a \rightarrow b$

2)$TaskT \overset{LP}{\rightarrow} Action\ x$:
$\forall a \in T$: if exists link $a \rightarrow JPk$ for some k,remove that link //if exists link $SPk \rightarrow x$ for some k,remove that link
i)if T only has a parallel link Pi: $Pi \rightarrow x$
ii)if x only has a parallel link Pi: $\forall a \in T : a \rightarrow Pi$
iii)if T and x have parallel link with Pi,Pj respectively:
$Pi \rightarrow Pj$
iv)if none has parallel link: $\forall a \in T : a \rightarrow x$

3)$Action\ x \overset{LP}{\rightarrow} TaskT$:
$\forall a \in T$: if exists link $x \rightarrow JPk$ for some k,remove that link //if exists link $SPk \rightarrow a$ for some k,remove that link
i)if x only has a parallel link and Pi: $\forall a \in T : Pi \rightarrow a$
ii)if T only has a parallel link with Pi: $x \rightarrow Pi$
iii)if x and T have parallel link with Pi,Pj respectively:
$Pi \rightarrow Pj$
iv)if none has parallel link and sync : $\forall a \in T : x \rightarrow a$

4)$Action\ x \overset{LP}{\rightarrow} Action\ y$:
if exists link $x \rightarrow JPk$ for some k,remove that link //if exists link $SPk \rightarrow y$ for some k,remove that link
i)if x only has a parallel link with Pi: $Pi \rightarrow y$
ii)if y only has a parallel link with Pi: $x \rightarrow Pi$
iii)if x and y have parallel link with Pi,Pj respectively:
$Pi \rightarrow Pj$
iv)if none has parallel link: $x \rightarrow y$

Κάποια παραδείγματα εφαρμογής του κανόνα είναι:

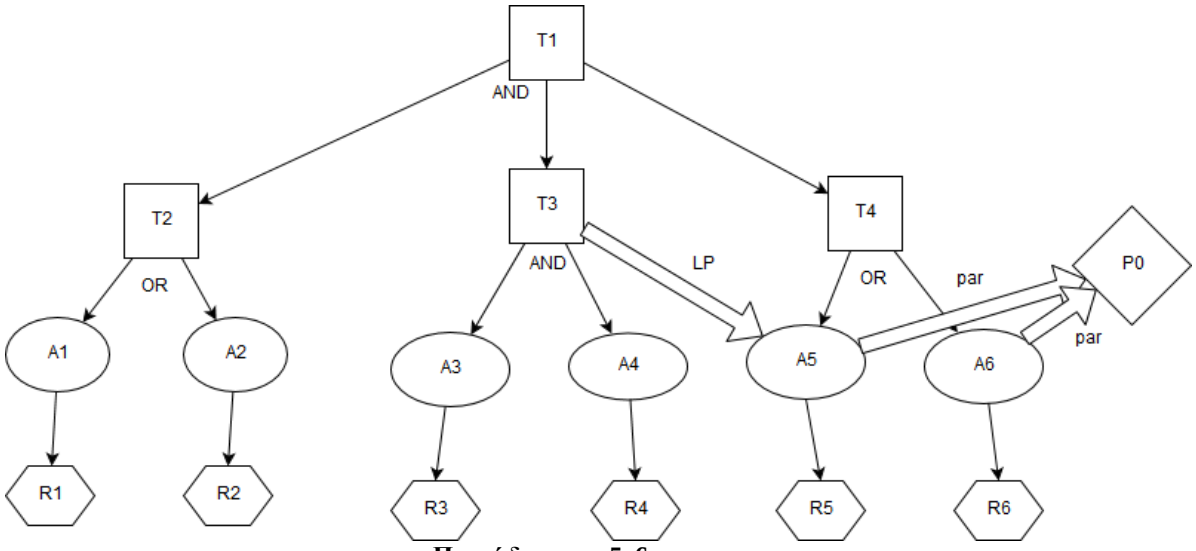


Παράδειγμα 5.5

Στο παραπάνω παράδειγμα λόγω της λογικής σύνδεσης των T2,T3 στους Γράφους που ικανοποιούν τα T2,T3 και επομένως θα περιέχουν τις αντίστοιχες ενέργειες, θα έχουν δημιουργηθεί οι εξής συνδέσεις (όπου βέβαια υπάρχουν και τα 2 μέλη):

1. A1->A3
2. A1->A4
3. A2->A3
4. A2->A4

Παράλληλα, θα έχουν διαγραφεί οι συνδέσεις των A1,A2 με το JoinPoint του Γράφου αφού δεν θα εκτελεστούν σίγουρα τελευταίες αυτές οι ενέργειες όπως και οι συνδέσεις των A3,A4 με το StartPoint καθώς δεν θα εκτελεστούν σίγουρα πρώτες.



Παράδειγμα 5.6

Από την εφαρμογή του κανόνα στο Παράδειγμα 6 θα δημιουργηθούν συνδέσεις μεταξύ των ενεργειών A3 και A4 με τον ψευδοκόμβο P0, ο οποίος απεικονίζει γράφους που περιέχουν όλες τις ενέργειες που εκτελούνται παράλληλα και ικανοποιούν τους συνδεδεμένους στον παράλληλο κόμβο P0.

5.1.4 Κανόνας Χρονικής Προϋπόθεσης

Ο κανόνας χρονικής προϋπόθεσης χρησιμοποιεί τις χρονικές συνδέσεις του δέντρου - στόχων για να απεικονίσει στο γράφο κάποια αλληλουχία ενεργειών, όπως και ο κανόνας λογικής προϋπόθεσης. Η αλγοριθμική λογική του κανόνα είναι παρόμοια με την αλγοριθμική λογική του κανόνα λογικής προϋπόθεσης. Πιο συγκεκριμένα, παρατίθεται αναλυτικά ο αλγόριθμος καθώς και ένα παράδειγμα εφαρμογής του για πληρότητα.

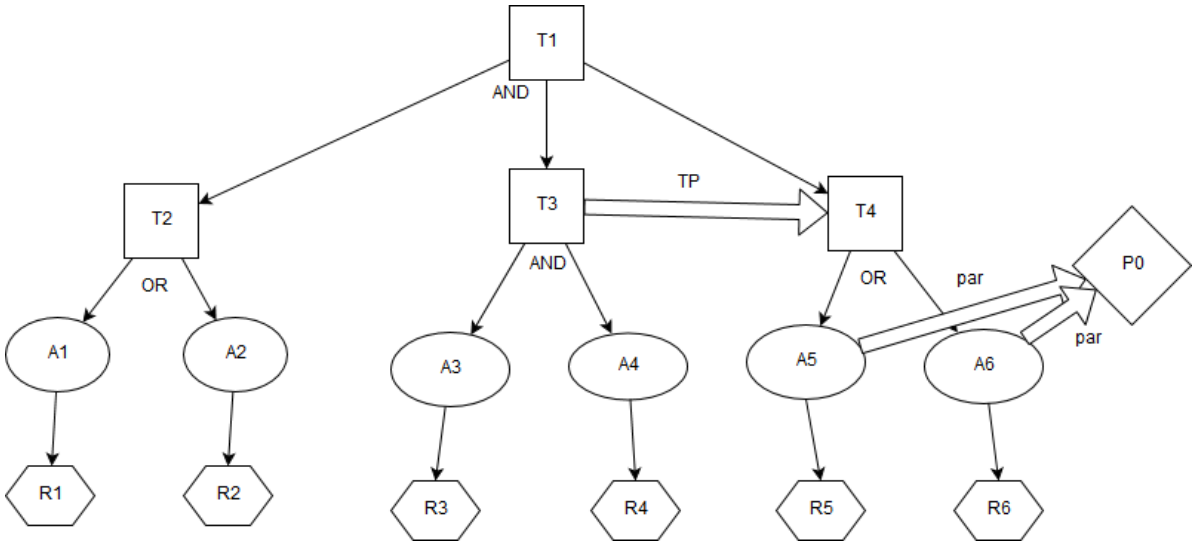
1)$TaskT1 \overset{TP}{\rightarrow} TaskT2$:
$\forall a \in T1, \forall b \in T2$: if exists link $a \rightarrow JPk$ for some k,remove that link //if exists link $SPk \rightarrow b$ for some k,remove that link
i)if T1 only has a parallel link with Pi: $\forall b \in T2 : Pi \rightarrow b$
ii)if T2 only has a parallel link with Pi: $\forall a \in T1 : a \rightarrow Pi$
iii)if T1 and T2 have parallel link with Pi,Pj respectively:
$Pi \rightarrow Pj$
iv)if none has parallel link: $\forall a \in T1, \forall b \in T2 : a \rightarrow b$

2)$TaskT \overset{TP}{\rightarrow} Action\ x$:
$\forall a \in T$: if exists link $a \rightarrow JPk$ for some k,remove that link //if exists link $SPk \rightarrow x$ for some k,remove that link
i)if T only has a parallel link Pi: $Pi \rightarrow x$
ii)if x only has a parallel link Pi: $\forall a \in T : a \rightarrow Pi$
iii)if T and x have parallel link with Pi,Pj respectively:
$Pi \rightarrow Pj$
iv)if none has parallel link: $\forall a \in T : a \rightarrow x$

3)$Action\ x \overset{TP}{\rightarrow} TaskT$:
$\forall a \in T$: if exists link $x \rightarrow JPk$ for some k,remove that link //if exists link $SPk \rightarrow a$ for some k,remove that link
i)if x only has a parallel link and Pi: $\forall a \in T : Pi \rightarrow a$
ii)if T only has a parallel link with Pi: $x \rightarrow Pi$
iii)if x and T have parallel link with Pi,Pj respectively:
$Pi \rightarrow Pj$
iv)if none has parallel link and sync : $\forall a \in T : x \rightarrow a$

4)$Action\ x \overset{TP}{\rightarrow} Action\ y$:
if exists link $x \rightarrow JPk$ for some k,remove that link //if exists link $SPk \rightarrow y$ for some k,remove that link
i)if x only has a parallel link with Pi: $Pi \rightarrow y$
ii)if y only has a parallel link with Pi: $x \rightarrow Pi$
iii)if x and y have parallel link with Pi,Pj respectively:
$Pi \rightarrow Pj$
iv)if none has parallel link: $x \rightarrow y$

Χαρακτηριστικό παράδειγμα της εφαρμογής αυτού του κανόνα είναι:



Παράδειγμα 5.7

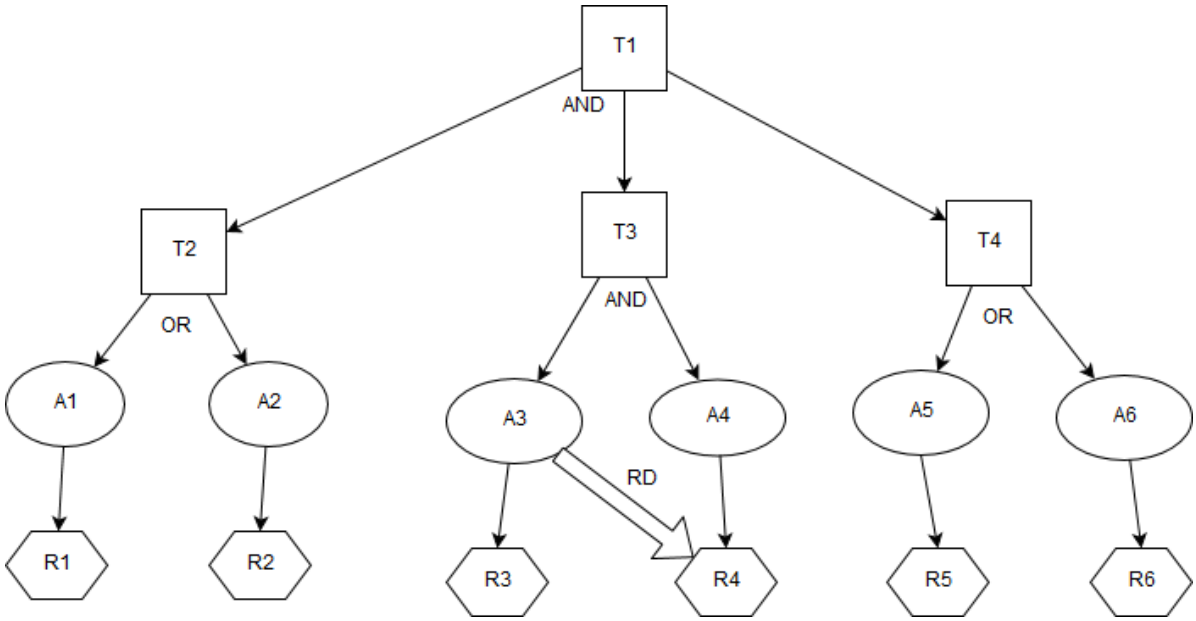
Η εφαρμογή του κανόνα της χρονικής προϋπόθεσης θα δημιουργήσει συνδέσεις των A3,A4 με τον ψευδοκόμβο P0 που απεικονίζει με την σειρά του γράφους ενεργειών που εκτελούνται παράλληλα.

5.1.5 Κανόνας Εξάρτησης Πηγών Δεδομένων

Μέσω του κανόνα αυτού απεικονίζονται οι εξαρτήσεις που αναπτύσσονται μεταξύ απλών στόχων - ενεργειών (Action) με άλλους απλούς στόχους - ενέργειες. Η εξάρτηση αυτή αφορά δεδομένα που δημιουργούνται από την εκτέλεση των ενεργειών του στόχου - πηγή και είναι απαραίτητα για την πηγή δεδομένων, μέσω της οποίας εκτελείται η ατομική ενέργεια που θα ικανοποιήσει τον απλό στόχο. Ως αποτέλεσμα, υποδεικνύεται έμμεσα μια συγκεκριμένη σειρά στην εκτέλεση των στόχων και κατ επέκταση των ενεργειών που τους ικανοποιούν.

$Action\ x \overset{rd}{\rightarrow} Resource\ a : x \rightarrow y$, where y parent action node of Resource a

Το παράδειγμα 8 και η εφαρμογή του κανόνα αναδεικνύει τη σημασία του:



Παράδειγμα 5.8

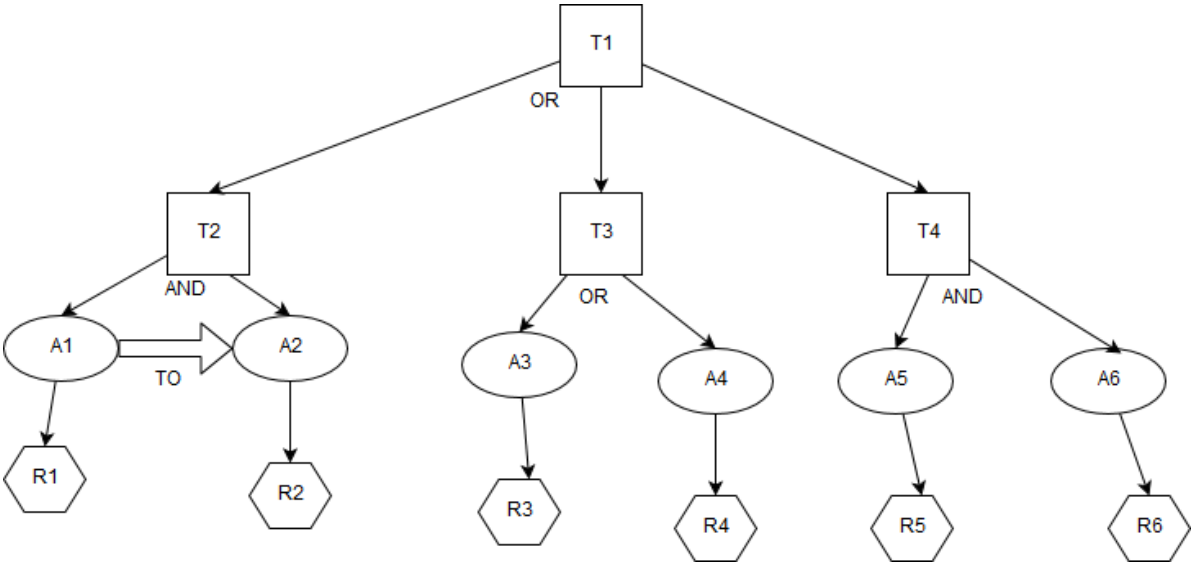
Στη συγκεκριμένη περίπτωση η πηγή δεδομένων R4 περιμένει την ολοκλήρωση της ενέργειας A3 για να λάβει κάποια δεδομένα από αυτήν για να καταστεί διαθέσιμή για το σύστημα. Η ενέργεια A4 για να εκτελεστεί θα πρέπει η πηγή R4 να είναι διαθέσιμη και επομένως συνάγουμε το συμπέρασμα ότι η ενέργεια A4 θα εκτελεστεί μετά την ενέργεια A3. Έτσι, δημιουργούμε τη σύνδεση A3->A4 σε κάθε γράφο που περιέχονται αυτές ενώ ταυτόχρονα διαγράφουμε τυχόν συνδέσεις του A3 με JoinPoint και του A4 με SplitPoint.

5.1.5 Κανόνας Χρονικής Διαφοράς

Ο κανόνας αυτός μετατρέπει τις σχέσεις χρονικής διαφοράς του δέντρου-στόχων σε δημιουργία συνδέσεων στους αντίστοιχους γράφους αλλά και διαγραφή συνδέσεων όπου είναι απαραίτητο. Παράλληλα, τοποθετεί σε μια λίστα του περιορισμούς σε χρόνο που δημιουργούνται από την εφαρμογή του κανόνα και οι οποίοι θα ληφθούν υπόψιν κατά την εκτέλεση του πλάνου ενεργειών.

$$Action\ x \xrightarrow{to} Action\ y : x \rightarrow y, \text{y.starttime-x.endtime} <= \text{timeout}$$

Παράδειγμα εφαρμογής:



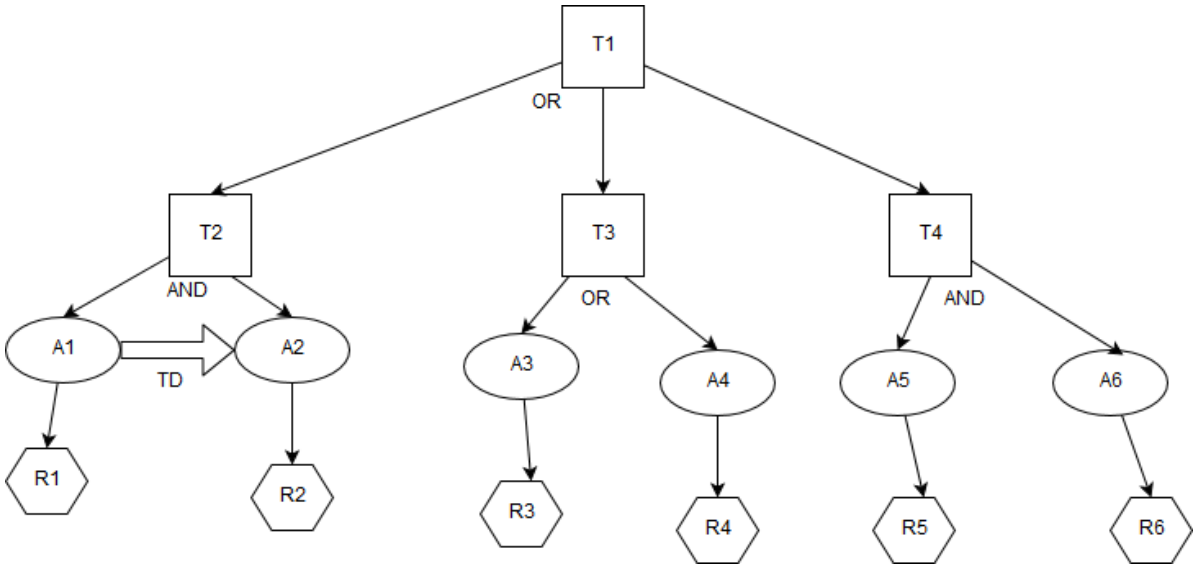
Παράδειγμα 5.9

Μετά την εφαρμογή του κανόνα στο παραπάνω παράδειγμα θα δημιουργηθεί μία σύνδεση A1->A2 σε κάθε ADG που περιλαμβάνονται ενώ θα σβηστούν τυχόν συνδέσεις του A1 και A2 με κάποιο JoinPoint και SplitPoint αντίστοιχα. Τέλος, θα δημιουργηθεί ένας περιορισμός που θα αφορά τα A1,A2 της μορφής *Timeout period between A1 and A2 is TO.get_timeout_period*.

5.1.6 Κανόνας Χρονικής Απόστασης

Ο κανόνας χρονικής απόστασης, όπως και ο κανόνας χρονικής διαφοράς, απεικονίζουν τις σχέσεις χρονικής απόστασης του δέντρου - στόχων στους Γράφους Εξάρτησης Ενεργειών. Επιπλέον, δημιουργεί και τοποθετεί σε λίστα τους αντίστοιχους περιορισμούς που είναι απαραίτητοι να ελεγχθούν κατά την εκτέλεση των πλάνων ενεργειών.

$Action\ x \xrightarrow{td} Action\ y : x \rightarrow y$, with the constraint y.starttime-x.endtime >= timedifference



Παράδειγμα 5.10

Η εφαρμογή του κανόνα στο παραπάνω παράδειγμα θα έχει σαν αποτέλεσμα να δημιουργηθεί η σύνδεση A1->A2 και να διαγραφούν οι συνδέσεις που πιθανώς υπάρχουν μεταξύ του A1 με κάποιο JoinPoint και μεταξύ του A2 με κάποιο SplitPoint.

5.2 Αλγόριθμος Ενορχήστρωσης

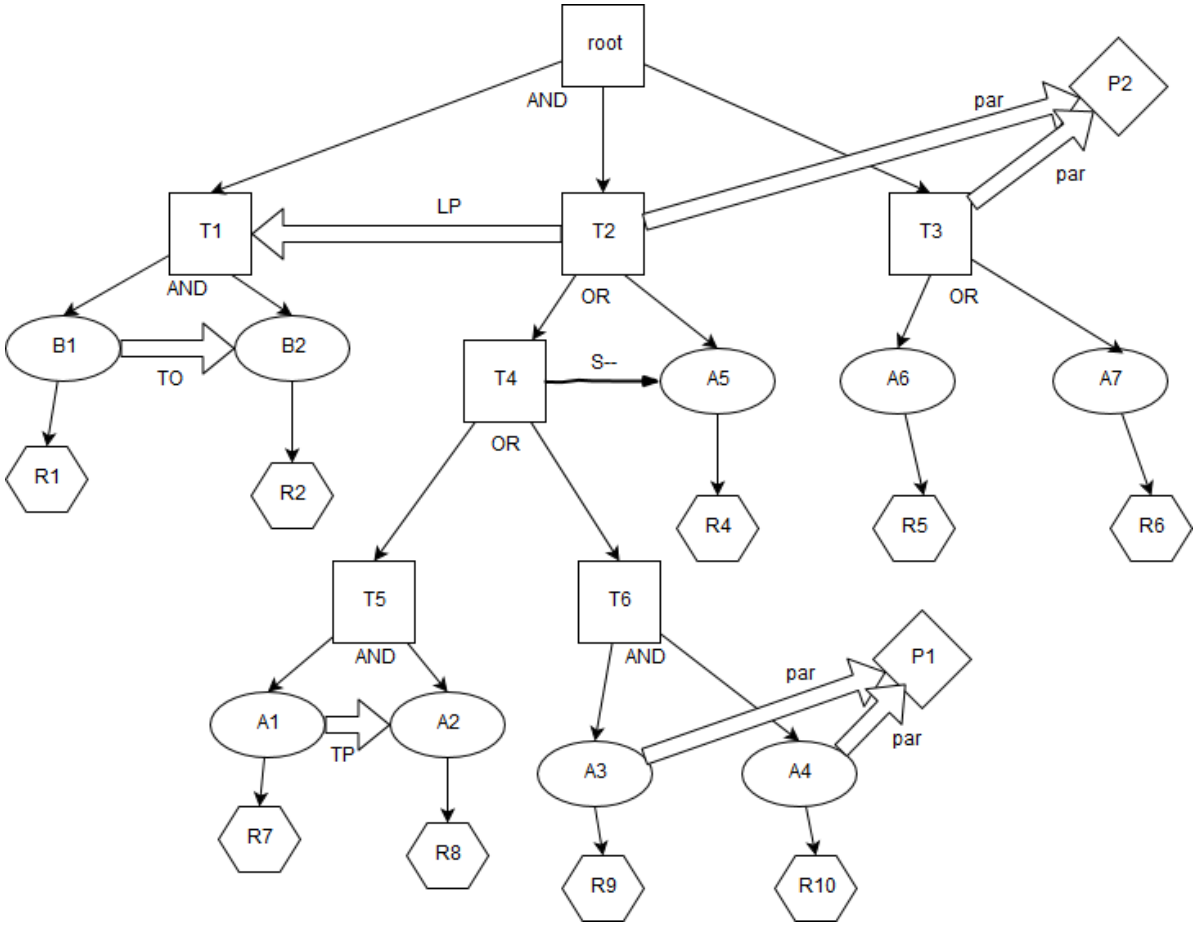
Ο αλγόριθμος ενορχήστρωσης αποτελεί βασικό κομμάτι της παρούσας διπλωματικής καθώς καθοδηγεί τους υπόλοιπους επιμέρους αλγορίθμους που έχουν αναπτυχθεί, με τελικό αποτέλεσμα την εύρεση όλων των δυνατών πλάνων ενεργειών που ικανοποιούν το δέντρο στόχων. Ο αλγόριθμος αυτός βρίσκει όλες τις δυνατές ενέργειες (Action) που ικανοποιούν κάθε κόμβο - στόχο του δέντρου, κάτι το οποίο είναι αναγκαίο για την δημιουργία των Γράφων Εξάρτησης Ενεργειών και την εύρεση των πλάνων. Η εύρεση αυτών των συνόλων από ενέργειες ξεκινάει από τους σύνθετους κόμβους που βρίσκονται χαμηλά στο δέντρο - στόχων καθώς η ικανοποίηση των ανώτερων σύνθετων στόχων εξαρτάται από τον τρόπο ικανοποίησης χαμηλότερων κόμβων - στόχων. Ενορχηστρώνει την εφαρμογή των κανόνων που παρουσιάστηκαν στο κεφάλαιο 5.1 και ταξινομεί τοπολογικά τους διάφορους γράφους έτσι ώστε να είναι δυνατή η εκπόνηση των πλάνων ενεργειών[11]. Τέλος, συνδυάζει τα διάφορα πλάνα ενεργειών των διαφορετικών παραλληλιών που έχουν δημιουργηθεί για την εκπόνηση ενιαίων αλληλουχιών από ενέργειες. Ο αλγόριθμος αυτός εκτελείται στο κεντρικό κομμάτι τους συστήματος που είναι ο Fog Node.

Πιο αναλυτικά ο αλγόριθμος είναι:

for all *Task* in Tasks **do**
  Find all sets of Actions that satisfy Task
end for
Apply Contribution Rule
Create ADG-Collection with id '0',which is the initial ADG-Collection
For each set of Actions that satisfy root node create ADG and put it in initial ADG-Collection
Apply Parallel Rule
Apply LogicalPrecondition Rule
Apply TemporalPrecondition Rule
Apply ResourceDependency Rule
Apply Timeout Rule
Apply Timedifference Rule
for all *ADG − Collection* in ADG-Collections **do**
  for all *ADG* in *ADG − Collection* **do**
    Transform ADG to Directed Graph
    Do topological sorting to Directed Graph
  end for
end for
Combine Directed Graphs between various ADG-Collections in order to extract uniform Action Plans
Print Results

Στη συνέχεια, παρουσιάζονται 2 παραδείγματα δέντρων - στόχων και παρατίθενται τα αποτελέσματα τους μετά την εφαρμογή του αλγορίθμου.

Παράδειγμα 5.11



Για την εποπτικότερη εικόνα του αλγορίθμου θα δοθούν πρώτα όλα τα σύνολα ενεργειών που ικανοποιούν κάθε κόμβο (με την κατάλληλη σειρά βάση των εξαρτήσεων που υπάρχουν στο μοντέλο) και στη συνέχεια τα τελικά αποτελέσματα.

T1:{B1,B2}
T5:{A1,A2}
T6:{P1}
T4:{A1,A2},{P1},{A1,A2,P1}
T2: {A1,A2},{P1},{A1,A2,P1},{A5}
T3:{A6},{A7},{A6,A7}
Root:{P2,B1,B2}

Οι ψευδοκόμβοι P1,P2 αναπαριστούν πλάνα με αλληλουχίες ενεργειών που εκτελούνται παράλληλα για να ικανοποιήσουν στόχους που ανήκουν σε κάποια ομάδα παραλληλίας
Τα τελικά πλάνα που θα εκπονηθούν είναι:

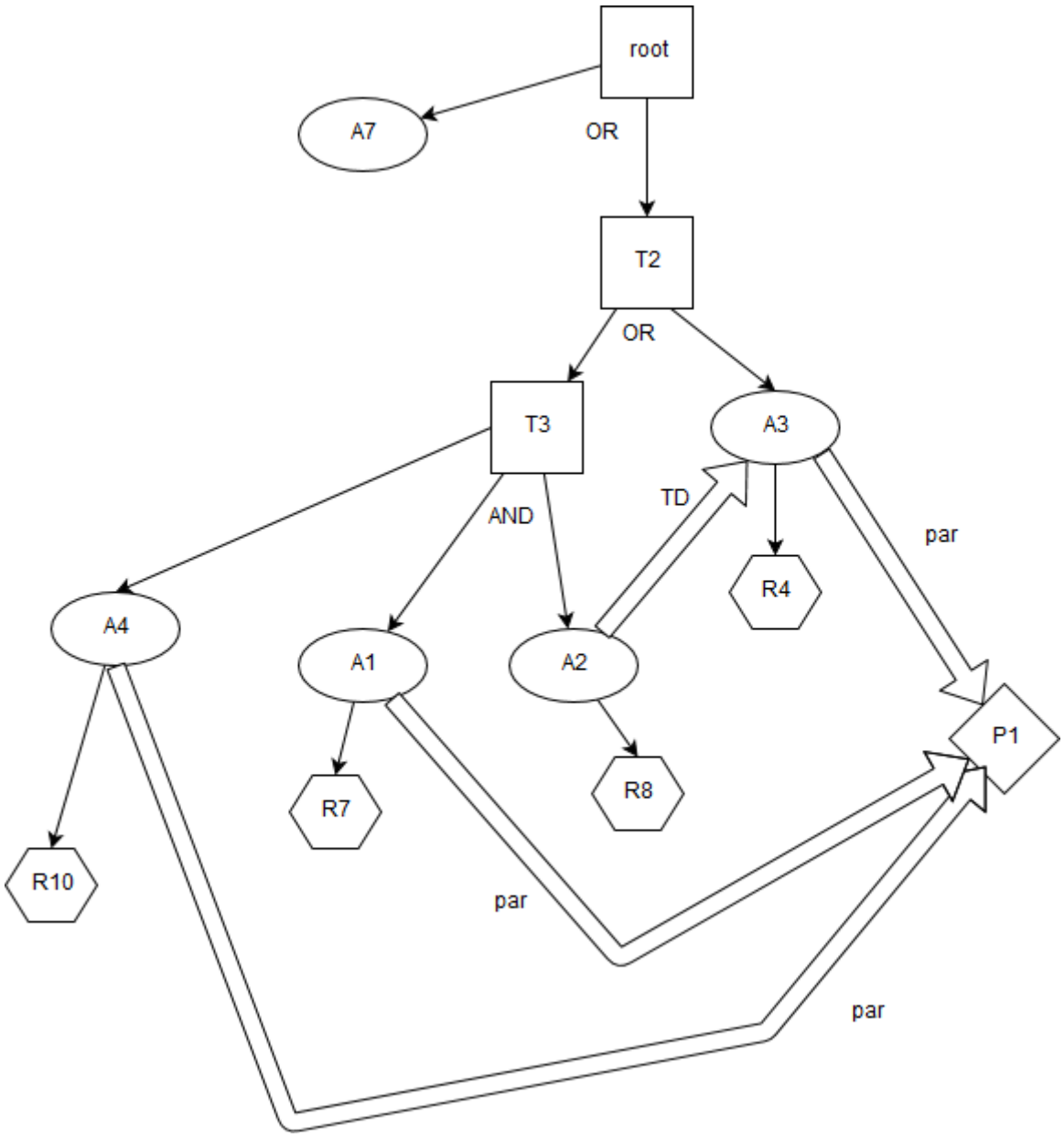
1. {B1,B2,P2}
   P2-> {A1,A2},{A6} (τρέχουν παράλληλα οι αλληλουχίες)
2. {B1,B2,P2}
   P2-> {A1,A2},{A7}
3. {B1,B2,P2}
   P2-> {A1,A2},{A6,A7}
4. {B1,B2,P2}
   P2-> {P1},{A6}
   P1->{A3},{A4}
5. {B1,B2,P2}
   P2-> {P1},{A7}
   P1->{A3},{A4}
6. {B1,B2,P2}
   P2-> {P1},{A6,A7}
   P1->{A3},{A4}
7. {B1,B2,P2}
   P2-> {A1,A2,P1},{A6}
   P1->{A3},{A4}
8. {B1,B2,P2}
   P2-> {A1,A2,P1},{A7}
   P1->{A3},{A4}
9. {B1,B2,P2}
   P2-> {A1,A2,P1},{A6,A7}
   P1->{A3},{A4}
10. {B1,B2,P2}
   P2-> {A5},{A6}
11. {B1,B2,P2}
   P2-> {A5},{A7}
12. {B1,B2,P2}
   P2-> {A5},{A6,A7}

Επιπλέον, θα υπάρχουν περιορισμοί από τις συνδέσεις διαφοράς χρόνου για τη στιγμή εκτέλεσης των B1,B2 αλλά και από τις παραλληλίες σχετικά με το

χρόνο που θα έχουν οι παράλληλες αλληλουχίες ενεργειών για να ολοκληρωθούν, ξεκινώντας τη χρονομέτρηση από τη στιγμή που ολοκληρώνεται η πρώτη αλληλουχία.

Σημείωση: Στην παραλληλία P2 συνδέονται οι σύνθετοι κόμβοι T2 και T3. Αυτοί αποσυντίθενται μέσω διάζευξης σε απλούστερους στόχους. Εξαιτίας αυτού, λαμβάνεται υπόψη κατά την εκτέλεση του αλγορίθμου και υπολογίζονται όλες οι ακολουθίες που ικανοποιούν τους παράλληλους κόμβους. Στην προκειμένη, για παράδειγμα, για την ομάδα γράφων P2 θα υπολογιστούν όλες οι ακολουθίες ενεργειών που ικανοποιούν τον T3, δηλαδή τα {A6},{A7},{A6,A7}. Το ίδιο ισχύει και για τον κόμβο T2. Τέλος, αυτά τα σύνολα θα συνδυαστούν μεταξύ τους και θα δημιουργήσουν τα «παράλληλα» σύνολα του P2, όπως φαίνεται και παραπάνω.

Παράδειγμα 5.12



T3:{P1,A2}
T2:{A3},{P1},{A3,P1}
Root:{A7},{P1},{A7,P1},{A2,P1},{A2,P1,A7}
Τα τελικά πλάνα που θα εκπονηθούν είναι:

1. {A7}
2. {P1}
   P1->{A3}
3. {P1,A7}
   P1->{A3}
4. {A2,P1}
   P1->{A4},{A1},{A3}
   P1->{A4},{A1}
5. {A2,P1,A7}
   P1->{A4},{A1},{A3}
   P1->{A4},{A1}

Επίσης, θα υπάρχουν οι περιορισμοί που δημιουργούνται από τις συνδέσεις χρόνου απόστασης και τις παραλληλίες.

Σημείωση: Στον παράλληλο κόμβο P1 συνδέονται οι ατομικές ενέργειες A4,A1,A3. Η ενέργεια, όμως, A3, μπορεί να μην έχει επιλεγεί για την ικανοποίηση του σύνθετου στόχου T2 εξαιτίας της διάζευξης. Αυτό απεικονίζεται στις παραπάνω λύσεις όπου φαίνονται οι ακολουθίες στις οποίες περιλαμβάνεται η ενέργεια A3 και αυτές που δεν περιλαμβάνεται.

Κεφάλαιο 6: Πειράματα και Μετρήσεις Απόδοσης

Στο κεφάλαιο αυτό καταγράφονται τα αποτελέσματα μιας σειράς δοκιμών που πραγματοποιήθηκαν για την εκτίμηση της απόδοσης του συστήματος. Μετρήθηκε ο απαιτούμενος χρόνος εκτέλεσης καθώς και οι απαιτήσεις σε μνήμη για διάφορα μεγέθη μοντέλων. Ακόμα, εξετάστηκαν οι επιδράσεις που έχουν στην απόδοση του συστήματος ο αριθμός των κόμβων που τρέχουν παράλληλα καθώς και οι ομάδες παραλληλίας που αυτοί σχηματίζουν. Τέλος, στις μετρήσεις που πραγματοποιήθηκαν λήφθηκαν υπόψη ο αριθμός των OR αποσυνθέσεων, το μέσο βάθος των OR αποσυνθέσεων και των παράλληλων κόμβων στα δέντρα στόχων. Τα μεγέθη αυτά αποτελούν κρίσιμο παράγοντα για την απόδοση τους συστήματος. Οι OR αποσυνθέσεις όσο ψηλότερα βρίσκονται στο δέντρο - στόχων δημιουργούν περισσότερους συνδυασμούς από ατομικές ενέργειες που ικανοποιούν τον κόμβο πατέρα, λόγω του μεγάλου υποδέντρου που πρέπει να εξεταστεί χαμηλότερα από αυτόν. Επιπλέον, το βάθος των παράλληλων κόμβων και κατ'επέκταση το μέγεθος των υποδένδρων που πρέπει να εξεταστούν, καθορίζουν το μέγεθος του αντίστοιχου Γράφου Εξάρτησης Ενεργειών και τον αριθμό των αλληλουχιών από ατομικές ενέργειες που ικανοποιούν το σύστημα.

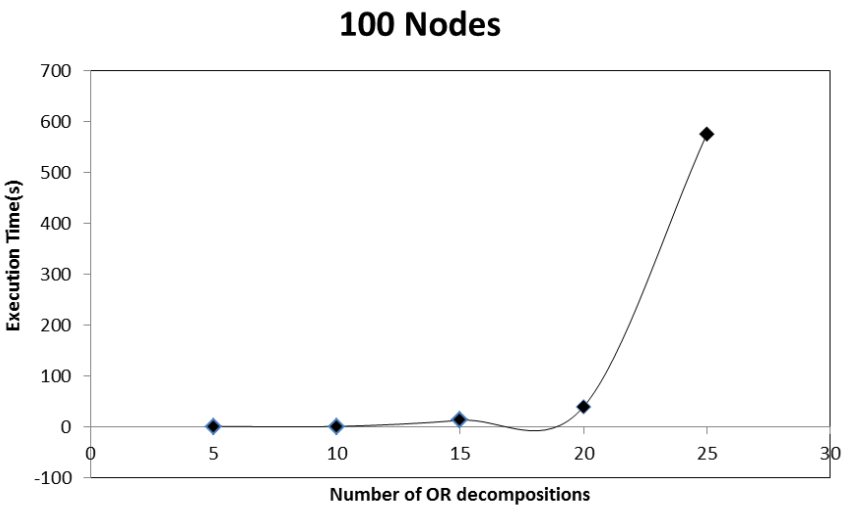
Για τις ανάγκες της αξιολόγησης της απόδοσης του συστήματος δημιουργήθηκαν διάφορα μεγέθη μοντέλων κρατώντας σταθερό τον αριθμό στα εξής χαρακτηριστικά: parallel nodes, parallel links, logicalprecondition links, temporalprecondition links, contribution links, resource dependencies, timeout links, timedifference links και parallel depth με τιμές 3,8,20,20,8,40,20,20,4 αντίστοιχα. Στη συνέχεια, εξετάστηκε η απόδοση του συστήματος σε χρόνο και μνήμη έχοντας ως σταθερές 150 κόμβους, 20 logicalprecondition links, 20 temporalprecondition links, 8 contribution links, 40 resource dependencies, 20 timeout links, 20 timedifference links και μεταβλητές το γινόμενο OR depth*Parallel depth, που όπως αναλύθηκε παραπάνω επιδρά σημαντικά στο σύστημα, και τον αριθμό των παράλληλων ομάδων και κόμβων.

6.1 Επίδραση Μεγέθους Μοντέλου Δέντρου Στόχων και OR αποσυνθέσεων στην Απόδοση του Συστήματος

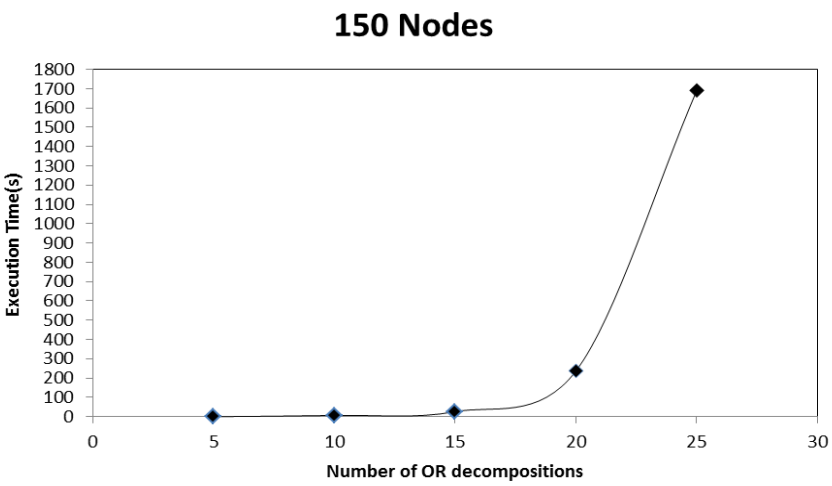
Σε αυτή την ενότητα θα παρουσιαστεί η απόδοση τους συστήματος σε χρόνο και μνήμη για διάφορα μεγέθη μοντέλων. Τα μεγέθη των μοντέλων που εξετάστηκαν είναι: 100, 150, 200, 250 και 300 κόμβοι. Παράλληλα, για κάθε μέγεθος εξετάζεται η επίδραση που έχει ο μέσος αριθμός των OR αποσυνθέσεων.

6.1.1 Χρονική απόδοση

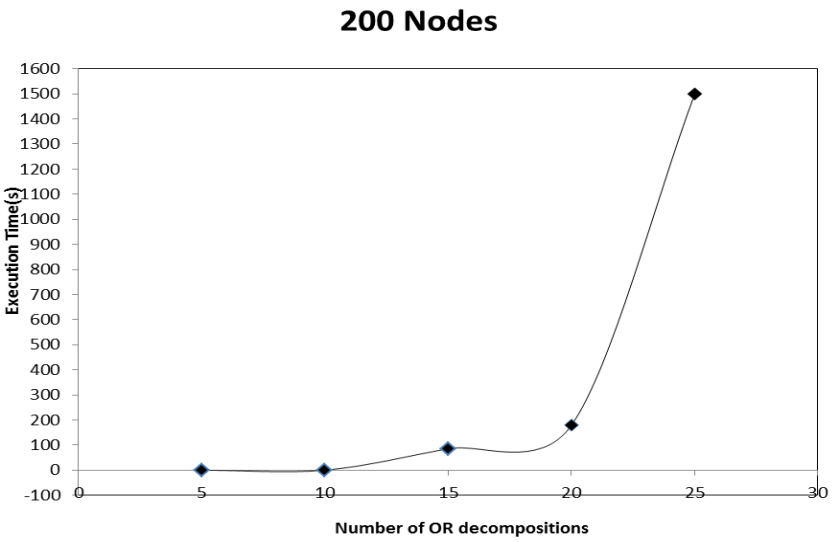
Στα παρακάτω διαγράμματα βλέπουμε το χρόνο εκτέλεσης του προγράμματος για διάφορες τιμές μεγεθών μοντέλων και για διάφορες τιμές OR αποσυνθέσεων.



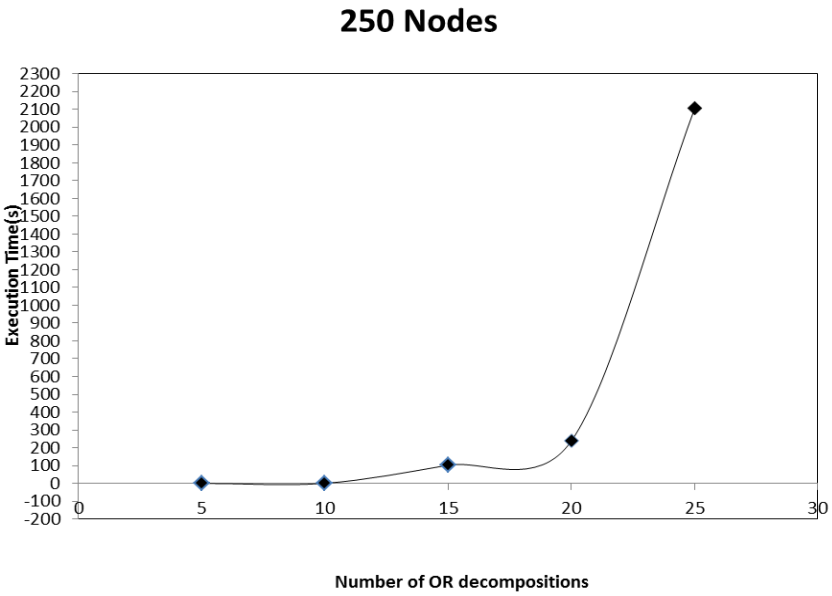
Διάγραμμα 6.1



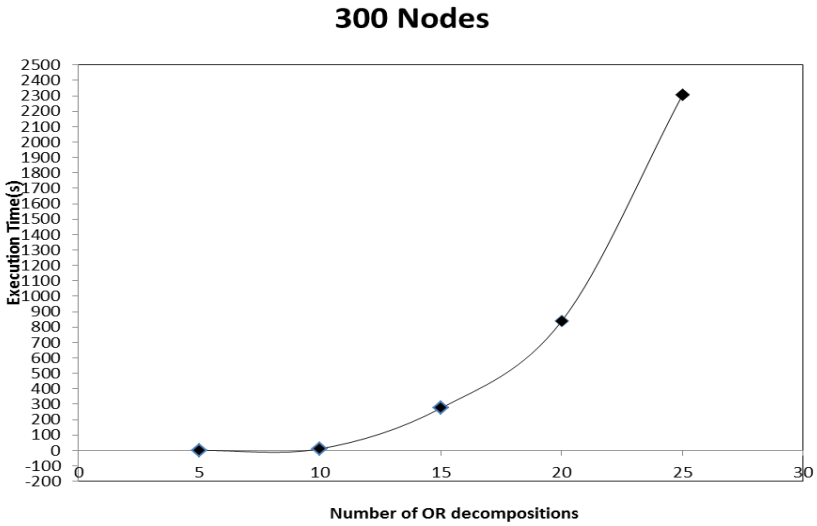
Διάγραμμα 6.2



Διάγραμμα 6.3



Διάγραμμα 6.4

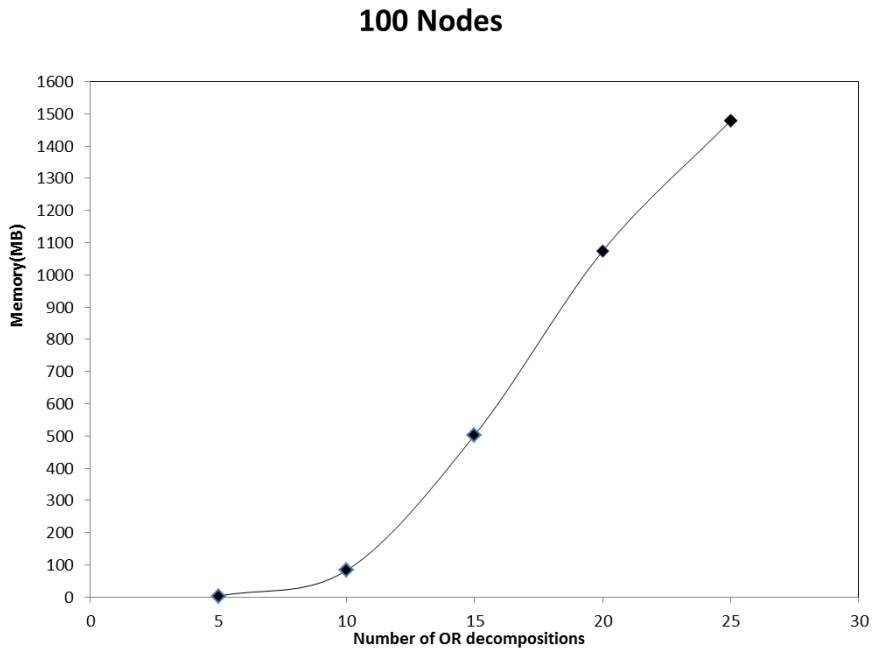


Διάγραμμα 6.5

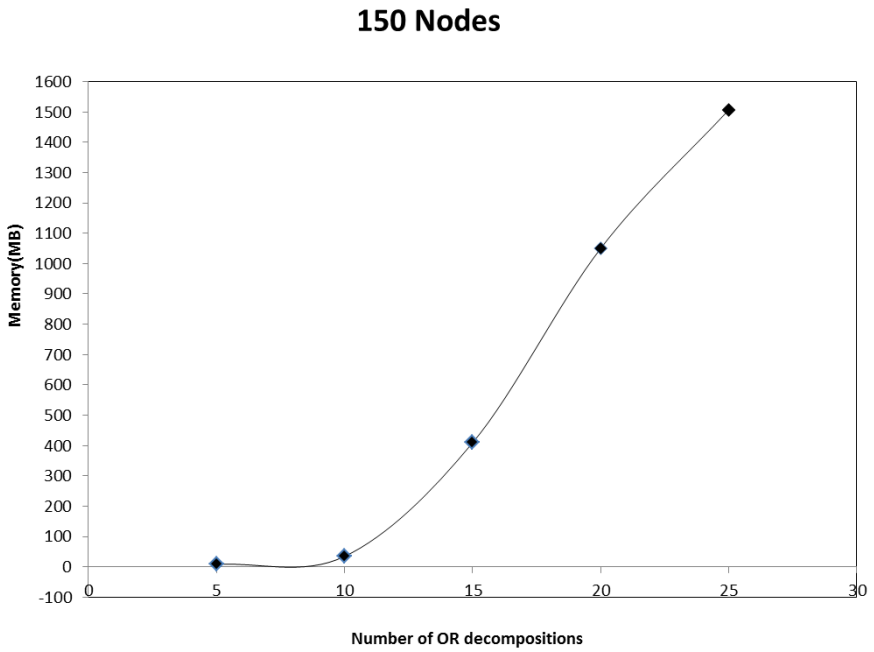
Από τα παραπάνω διαγράμματα φαίνεται μια αύξηση του χρόνου όσο μεγαλώνει το μέγεθος του μοντέλου - στόχων και μια εκθετική αύξηση του χρόνου με την αύξηση των OR αποσυνθέσεων. Για κάθε OR αποσύνθεση που εισάγεται στο σύστημα πρέπει να βρεθούν όλα εκείνα τα σύνολα ατομικών ενεργειών που ικανοποιούν τους κόμβους - παιδιά της αποσύνθεσης και να συνδυαστούν μεταξύ τους για κάθε πιθανό σενάριο που ικανοποιεί τον κόμβο - πατέρα. Οπότε η επιβάρυνση στο σύστημα είναι αρκετά μεγάλη με την εισαγωγή OR αποσυνθέσεων, το οποίο είναι και ένα μέτρο της ευελιξίας του συστήματος για την ικανοποίηση του κόμβου - ρίζα.

6.1.2 Απόδοση Μνήμης

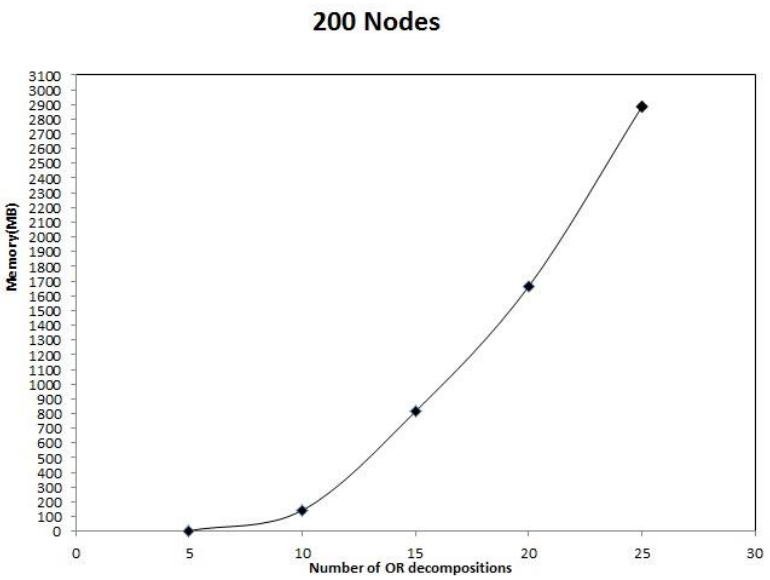
Τα διαγράμματα που ακολουθούν παρουσιάζουν τις απαιτήσεις σε μνήμη διάφορων μοντέλων σε διαφορετικά μεγέθη μεταξύ τους, σε συνδυασμό με το πλήθος των OR αποσυνθέσεων που υπάρχουν σε κάθε μοντέλο.



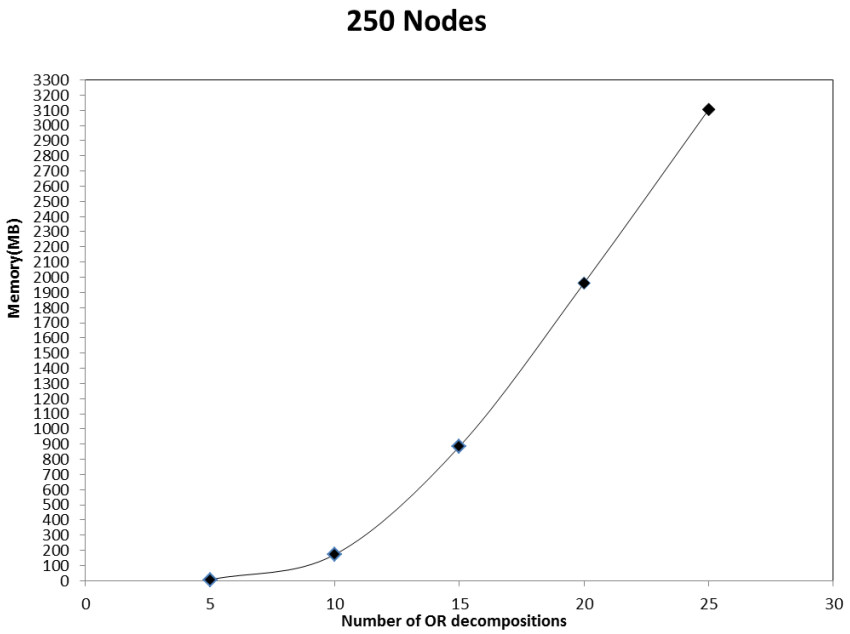
Διάγραμμα 6.6



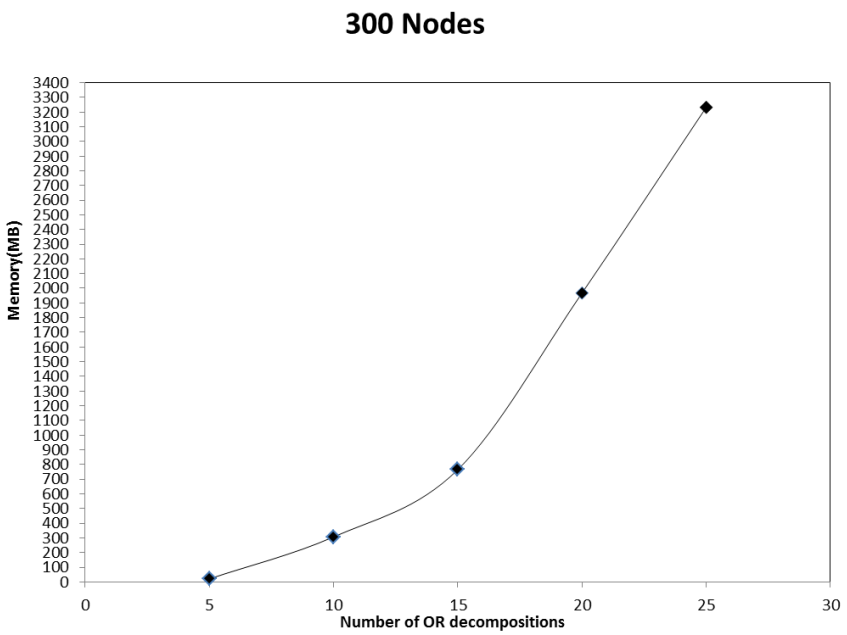
Διάγραμμα 6.7



Διάγραμμα 6.8



Διάγραμμα 6.9



Διάγραμμα 6.10

Όπως και στα αντίστοιχα διαγράμματα για το χρόνο εκτέλεσης παρατηρούμε ότι η κύρια επιβάρυνση για το σύστημα είναι η αύξηση των OR αποσυνθέσεων, όπου παρατηρείται εκθετική αύξηση σε σχέση με την αύξηση του αριθμού των OR αποσυνθέσεων.

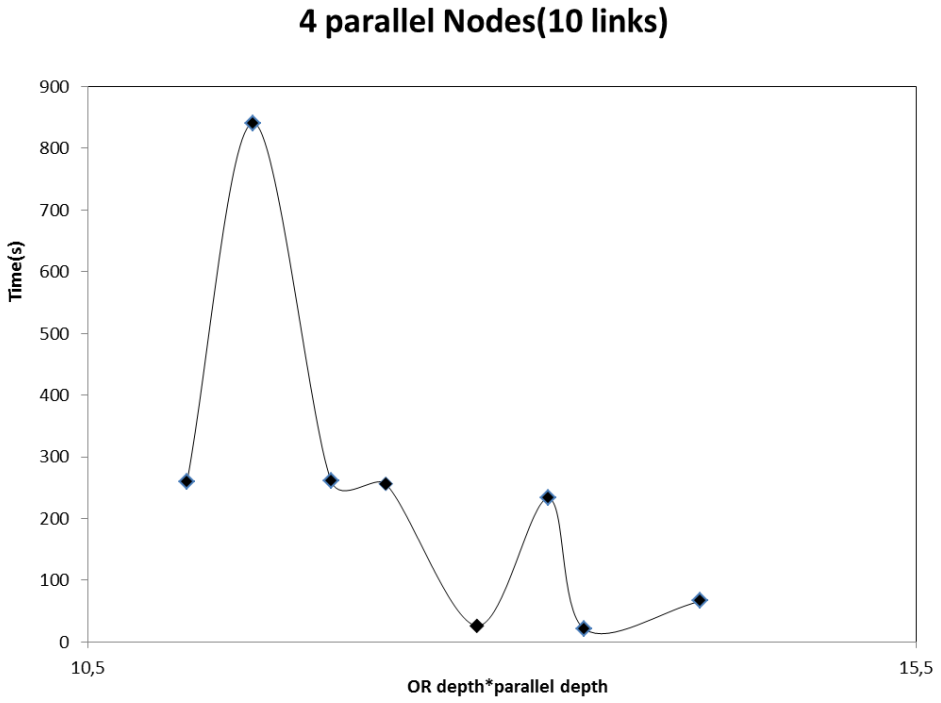
Καθοριστικός παράγοντας, λοιπόν, για την απόδοση του συστήματος αποτελεί σε μικρότερο βαθμό το μέγεθος του μοντέλου σε σχέση με τον αριθμό των OR αποσυνθέσεων, το οποίο αναπαριστά την ευελιξία του συστήματος για την ικανοποίηση του κόμβου-ρίζα και κατ επέκταση της ορθής λειτουργίας τους συστήματος. Η ευελιξία δηλαδή στην ικανοποίηση των στόχων αυξάνει την πολυπλοκότητα του συστήματος σε μνήμη και χρόνο.

6.2 Επίδραση Αριθμού Παραλληλιών και OR Αποσυνθέσεων στην Απόδοση του Συστήματος

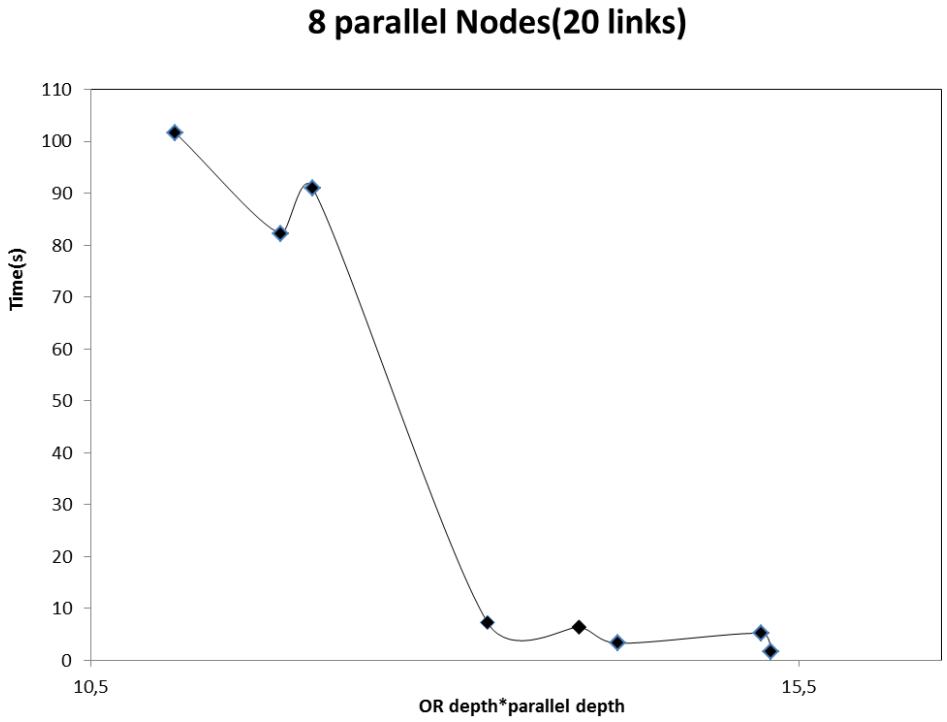
Στην παρούσα ενότητα θα παρουσιαστούν οι απαιτήσεις του συστήματος σε χρόνο και μνήμη σε σχέση με τις ομάδες παραλληλιών και τους παράλληλους κόμβους που συνδέονται σε αυτές. Επιπλέον, κρατώντας σταθερό τον αριθμό των παράλληλων ομάδων και των παράλληλων κόμβων μελετάται η επίδραση του γινομένου του μέσου βάθους των OR αποσυνθέσεων με το μέσο βάθος των παράλληλων κόμβων στην απόδοση του συστήματος.

Στα παρακάτω διαγράμματα εξετάζονται μοντέλα με αριθμό ομάδων 4,8,12 και παράλληλους κόμβους 10,20,28 αντίστοιχα κρατώντας σταθερά τον αριθμό των κόμβων, τα logicalprecondition links, τα temporalprecondition links, τα contribution links, τα resource dependencies, τα timedifference

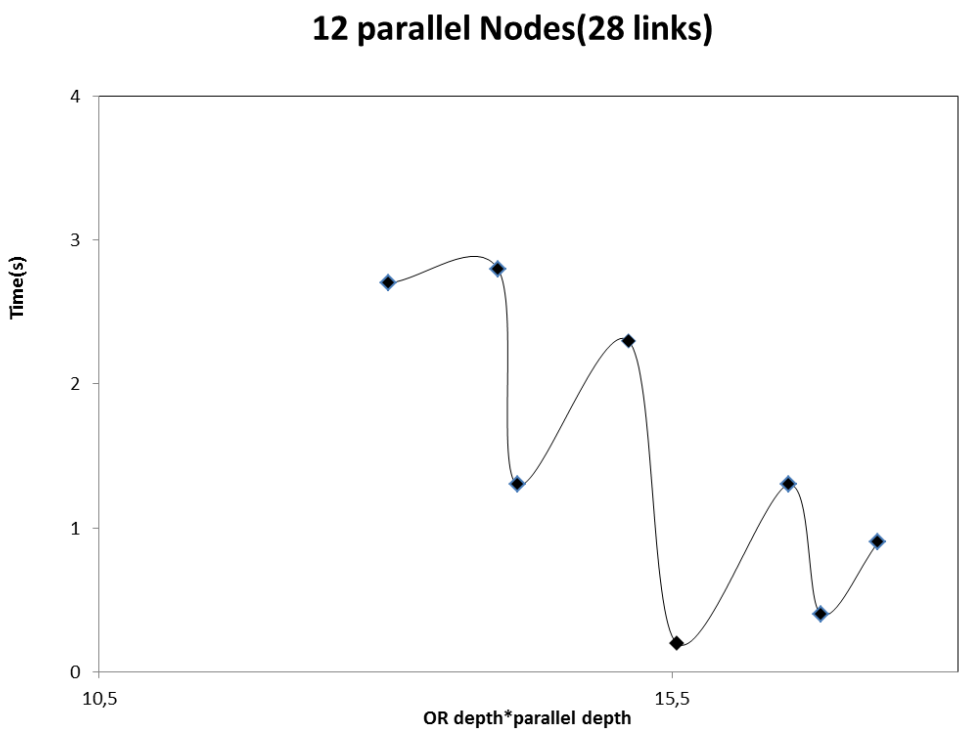
links, τα timeout links και τον αριθμό των OR αποσυνθέσεων με τιμές 150,20,20,8,40,20,20,15. Επιπλέον το γινόμενο OR depth*Parallel depth εκτείνεται από τιμές 11 έως 17.
Στη συνέχεια, παρουσιάζονται πρώτα τα διαγράμματα που αφορούν την απόδοση του συστήματος σε χρόνο και μετέπειτα αυτά που αφορούν τη μνήμη.



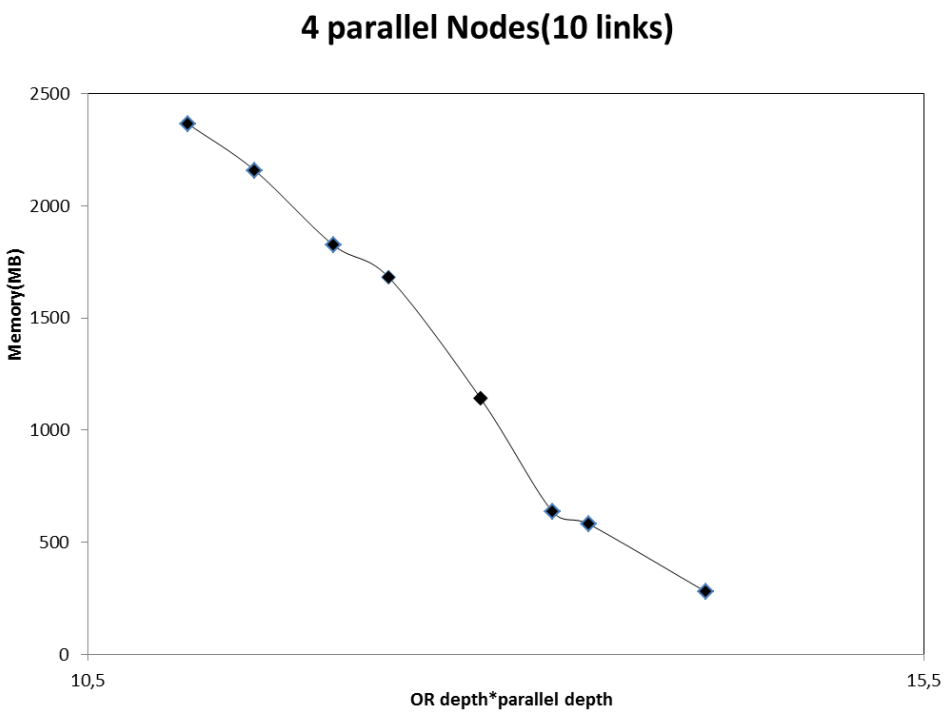
Διάγραμμα 6.11



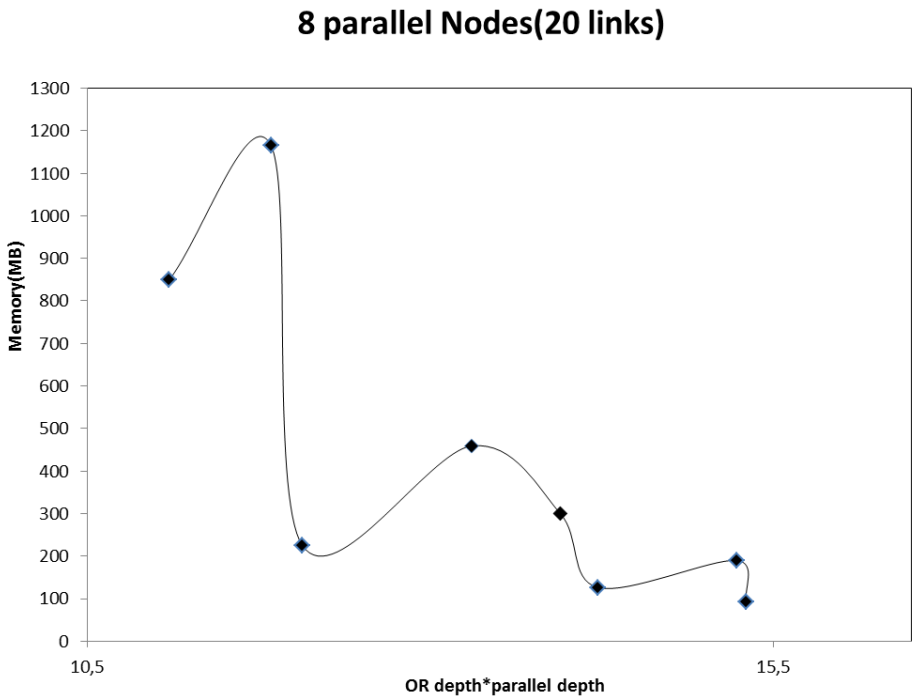
Διάγραμμα 6.12



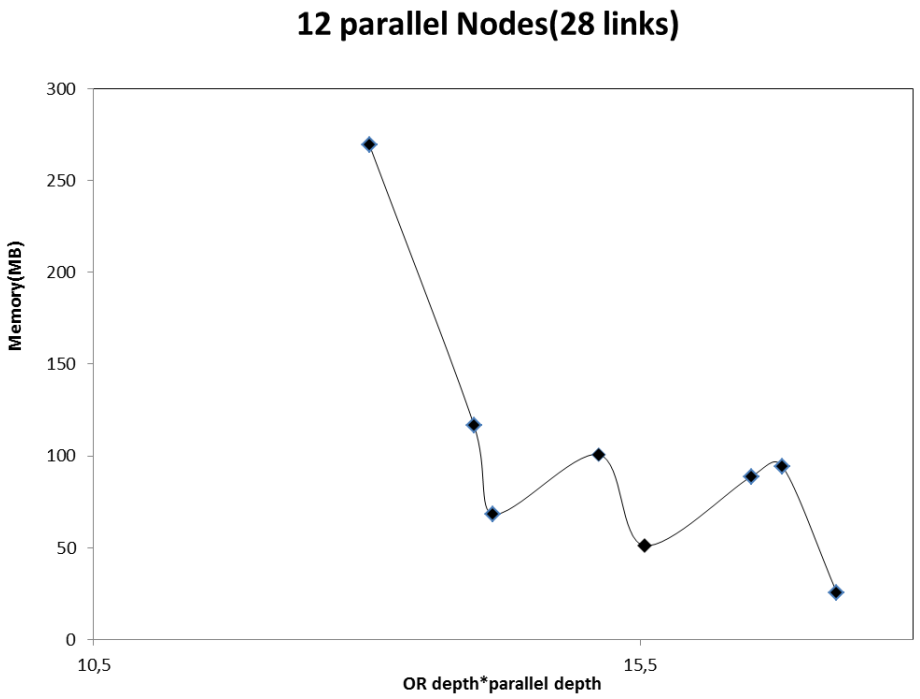
Διάγραμμα 6.13



Διάγραμμα 6.14



Διάγραμμα 6.15



Διάγραμμα 6.16

Παρατηρώντας τα παραπάνω διαγράμματα βλέπουμε ότι αυξάνοντας τις ομάδες παραλληλίας μειώνονται σε ένα βαθμό οι απαιτήσεις του συστήματος τόσο σε χρόνο όσο και σε μνήμη. Αυτό συμβαίνει καθώς για κάθε ομάδα παραλληλίας που υπάρχει στο μοντέλο εισάγεται ένας ψευδοκόμβος που αντιπροσωπεύει μια συλλογή από Γράφους Εξάρτησης Ενεργειών και περιλαμβάνει όλους τους σχετικούς, με αυτή την ομάδα, παράλληλους

κόμβους ενώ οι κανόνες μετασχηματισμού εφαρμόζονται στην κάθε συλλογή ξεχωριστά και ανεξάρτητα. Συνεπώς, η εύρεση των πλάνων ατομικών ενεργειών ανάγεται σε πρόβλημα διαίρει και βασίλευε, όπου αναζητούνται επαρκείς αλληλουχίες ατομικών ενεργειών σε κάθε συλλογή και στο τέλος συνδυάζονται για το τελικό αποτέλεσμα. Αντί η εύρεση των πλάνων να γίνει σε ολόκληρο το εύρος του δέντρου - στόχων με το αντίστοιχο κόστος που έχει αυτό για τον υπολογισμό όλων των συνδυασμών από ατομικές ενέργειες που ικανοποιεί κάποιον κόμβο, γίνεται στα διάφορα υποδέντρα που καθορίζονται από τις ομάδες παραλληλίας και τα οποία είναι σαφώς μικρότερα σε μέγεθος, το οποίο συνεπάγεται λιγότερο χρόνο και λιγότερη μνήμη.

Καθοριστικοί παράγοντες στην απόδοση τους συστήματος είναι το μέσο βάθος των OR αποσυνθέσεων και το μέσο βάθος των παράλληλων κόμβων.
Οι OR αποσυνθέσεις όταν βρίσκονται ψηλά στο δέντρο - στόχων έχουν ως αποτέλεσμα μεγάλο κόστος στον υπολογισμό όλων των δυνατών συνδυασμών που ικανοποιούν τον κόμβο - πατέρα εξαιτίας των πολυάριθμων αλληλουχιών από ατομικές ενέργειες που ικανοποιούν τους κόμβους που βρίσκονται χαμηλότερα στο νοητό υποδέντρο με κόμβο - ρίζα τον κόμβο - πατέρα της αποσύνθεσης. Εξάλλου, το μέσο βάθος των παράλληλων κόμβων καθορίζει το μέγεθος των συλλογών από γράφους που έχουν δημιουργηθεί για κάθε ομάδα παραλληλίας. Ένας παράλληλος κόμβος με χαμηλό βάθος στο δέντρο δεν «εισάγει» πολλούς κόμβους στους Γράφους Εξάρτησης Ενεργειών με αποτέλεσμα να μην είναι ιδιαίτερα πολύπλοκο και μεγάλο σε μέγεθος και να είναι ευκολότερη η διαχείριση του.

Βέβαια, το μοτίβο της μείωσης των απαιτήσεων σε χρόνο και μνήμη με την αύξηση του γινομένου βάθους δεν είναι καθολικό, όπως φαίνεται και από το διάγραμμα 6.14 στο δεύτερο σημείο με γινόμενο βάθους 11,84 και απαίτηση μνήμης 1165,9 MB. Σε αυτές τις περιπτώσεις, οι απαιτήσεις είναι μεγαλύτερες εξαιτίας της εφαρμογής των κανόνων μετασχηματισμού σε περισσότερους σύνθετους στόχους. Για παράδειγμα, μεγαλύτερο κόστος έχει για το σύστημα όταν η πλειοψηφία των logicalprecondition links αφορούν αποκλειστικά σύνθετους στόχους (Task) που περιλαμβάνουν πολυάριθμες ατομικές ενέργειες προς διαχείριση σε σχέση με τους απλούς στόχους που ικανοποιούνται από μόνο μια ατομική ενέργεια. Το ίδιο ισχύει και για κάθε σύνδεση που μπορεί να έχει ως άκρα σύνθετους στόχους.

Κεφάλαιο 7: Επίλογος

Στο κεφάλαιο αυτό γίνεται μια σύνοψη του συστήματος που εκπονήθηκε στην παρούσα διπλωματική και αποτιμώνται τα συμπεράσματα που εξήχθησαν ως προς την αρχιτεκτονική του συστήματος και τους αρχικούς στόχους που τέθηκαν. Επιπλέον, παρουσιάζονται προτάσεις για πιθανές μελλοντικές επεκτάσεις του θέματος που αναπτύχθηκε στον παρόντα τόμο.

7.1 Σύνοψη και Συμπεράσματα

Στα πλαίσια της παρούσας διπλωματικής εξετάστηκε το πρόβλημα της εύρεσης όλων των αλληλουχιών από ατομικές ενέργειες που εξασφαλίζουν την εύρυθμη λειτουργία ενός συστήματος του Διαδικτύου των Πραγμάτων. Παράλληλα, εξετάστηκε η δομή του Διαδικτύου των Πραγμάτων για διαφορετικού είδους εφαρμογές για την μοντελοποίηση της δομής του.

Για το σκοπό αυτό, το λογισμικό που αναπτύχθηκε δίνει το δυνατότητα στο χρήστη να απεικονίσει σε μοντέλα δέντρων - στόχων, όλους τους στόχους που είναι απαραίτητοι για την λειτουργία του συστήματος αλλά και να καθορίσει ποιοι κόμβοι είναι διαθέσιμοι ανάλογα με το περιβάλλον - πλαίσιο με το οποίο συνυπάρχει. Στη συνέχεια, υπολογίζει τα δυνατά πλάνα ατομικών ενεργειών που θα ικανοποιήσουν το στόχο - ρίζα του δέντρου και θα εξασφαλίζουν την εύρυθμη λειτουργία του συστήματος.

Στα προηγούμενα κεφάλαια παρουσιάστηκε η αρχιτεκτονική και το πλαίσιο λειτουργίας του λογισμικού που αναπτύχθηκε. Τα κυριότερα σημεία στα οποία εστιάσαμε ήταν:

- Αρχιτεκτονική του αναπτυχθέντος συστήματος: Η αρχιτεκτονική βασίζεται στον Fog Node, ο οποίος διαχειρίζεται τα υπόλοιπα στοιχεία του συστήματος. Με αυτόν τον τρόπο εξασφαλίζεται η ευελιξία και η επεκτασιμότητα του συστήματος.
- Μοντελοποίηση Διαδικτύου των Πραγμάτων: Η μοντελοποίηση της δομής του Διαδικτύου των Πραγμάτων δίνει μια σαφή κατεύθυνση για την υλοποίηση τέτοιων συστημάτων και των απαιτήσεων που αυτά έχουν.
- Μοντελοποίηση στόχων: Η μοντελοποίηση των στόχων που πρέπει να εκπληρώσει το σύστημα σε δενδρική μορφή καθιστά εύκολη τη διαχείρισή τους τόσο από το λογισμικό όσο και διαισθητικά από τους χρήστες. Η απλή αυτή μορφή αναπαράστασης έχει σαν αποτέλεσμα τη μείωση του χρόνου και της μνήμης που είναι απαραίτητα για την εύρεση των λύσεων.
- Ανάπτυξη αλγορίθμου: Σκοπός της παρούσας διπλωματικής είναι η εύρεση των πλάνων από ατομικές ενέργειες που ικανοποιούν τον κόμβο - ρίζα του δέντρου - στόχων. Σε αυτό το πλαίσιο αναλύθηκαν όλα τα βήματα του αλγορίθμου και δόθηκαν παραδείγματα δέντρων - στόχων συνοδευόμενα από τα πλάνα που εκπονήθηκαν.

- Αξιολόγηση λογισμικού: Τέλος, διεξήχθη αξιολόγηση του λογισμικού που αναπτύχθηκε ελέγχοντας την επίδοση σε σχέση με το χρόνο και τη μνήμη, μεταβάλλοντας κάθε φορά συγκεκριμένες μεταβλητές. Με αυτόν τον τρόπο εκπονήθηκαν συμπεράσματα για τις απαιτήσεις του σε χρόνο και μνήμη για διαφορετικά δέντρα - στόχων τόσο σε όγκο όσο και σε ποιοτικά χαρακτηριστικά.

7.2 Μελλοντικές επεκτάσεις

Όπως αναφέρθηκε και προηγουμένως, το λογισμικό που αναπτύχθηκε σχεδιάστηκε με τέτοιον τρόπο ώστε να διευκολύνονται πιθανές επεκτάσεις και βελτιστοποιήσεις.

Σχετικά, με τα πλάνα ατομικών ενεργειών μια βελτιστοποίηση θα ήταν η ενσωμάτωση βάρους σε κάθε κόμβο έτσι ώστε να επιλέγεται κάθε φορά το πλάνο ή τα πλάνα με το μικρότερο συνολικό βάρος. Με αυτό τον τρόπο, θα μειωθεί σημαντικά η πολυπλοκότητα του αλγορίθμου σε χρόνο και μνήμη, αφού θα περιορίζονται οι λύσεις στις καλύτερες. Στην παρούσα διπλωματική θεωρήθηκε το βάρος για όλους τους κόμβους ίσο.

Μια δεύτερη σημαντική βελτιστοποίηση θα ήταν η παραλληλοποίηση του κώδικα για την πλήρη εκμετάλλευση των πόρων του υπολογιστικού συστήματος που είναι επιφορτισμένο με την εύρεση των πλάνων. Έτσι, τα αποτελέσματα θα εξάγονται σε λιγότερο χρόνο, κάτι πολύ σημαντικό καθώς σε συστήματα Διαδικτύου των Πραγμάτων απαιτούνται αποφάσεις σε πραγματικό χρόνο.

Στην πρότυπη υλοποίηση θεωρήθηκε ότι οι στόχοι του συστήματος είναι απόλυτοι, δηλαδή ότι επιτυγχάνουν ή αποτυγχάνουν. Θα μπορούσε να χρησιμοποιηθεί ένα δέντρο - στόχων με ασαφή λογική, δηλαδή σε κάθε στόχο θα υπάρχει ένας βαθμός ικανοποίησης αυτού[47]. Αυτή η προσέγγιση είναι πιο αντιπροσωπευτική των πραγματικών συστημάτων καθώς πολλές φορές ενδιαφερόμαστε για την ικανοποίηση μερών του στόχου και όχι καθολικά ολόκληρου.

Μια χρήσιμη επέκταση του παρόντος λογισμικού θα ήταν η εφαρμογή μηχανικής μάθησης για τα διάφορα μεγέθη και χαρακτηριστικά των δέντρων - στόχων για να εξάγουμε άμεσα συμπέρασμα για τις απαιτήσεις σε χρόνο και σε μνήμη κατά την εκτέλεση. Η μάθηση σε μεγάλο αριθμό τέτοιων μοντέλων θα βοηθήσει στην διαχείριση των παραπάνω μοντέλων.

Τέλος, για την εύρεση του context που αφορά τον χρήστη μπορεί να αναπτυχθεί ένα σύστημα αντίστοιχο με το [48] που εκμεταλλεύεται τα κοινωνικά δίκτυα του χρήστη και με την χρήση διαδικτυακών οντολογιών, όπως για παράδειγμα OWL, να εξαχθούν συμπεράσματα για τις επιθυμίες του χρήστη σχετικά με το δέντρο - στόχων.

Κεφάλαιο 8: Βιβλιογραφία

[1] J. Bezivin, M. Barbero, and F. Jouault, ``On the applicability scope of model driven engineering,''
Model-Based Methodologies for Pervasive and
Embedded Software, 2007. MOMPES '07. Fourth International Workshop on, pp. 3--7, March 2007

[2] ObjectManagementGroup,Inc.,MDAGuideVersion1.0.1 https://www.omg.org/news/meetings/workshops/UML_2003_Manual/00-2_MDA_Guide_v1.0.1.pdf

[3] G. Chatzikonstantinou, M. Athanasopoulos and K. Kontogiannis, "Towards a Goal Driven Task Personalization Specification Framework," 2013 IEEE Ninth World Congress on Services, Santa Clara, CA, 2013, pp. 180-184.

[4] Bauer M. et al. (2013) IoT Reference Model. In: Bassi A. et al. (eds) Enabling Things to Talk. Springer, Berlin, Heidelberg

[5] Molano, J.I.R., Lovelle, J.M.C., Montenegro, C.E. et al. J Ambient Intell Human Comput (2018) 9: 709.

[6] Bonomi, Flavio & Milito, Rodolfo. (2012). Fog Computing and its Role in the Internet of Things. Proceedings of the MCC workshop on Mobile Cloud Computing. 10.1145/2342509.2342513.

[7] https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf

[8] Syed, Madiha & Fernández, Eduardo & Ilyas, Mohammad. (2016). A Pattern for Fog Computing. 1-10. 10.1145/3022636.3022649.

[9] Chatzikonstantinou G., Athanasopoulos M., Kontogiannis K. (2014) Task Specification and Reasoning in Dynamically Altered Contexts. In: Jarke M. et al. (eds) Advanced Information Systems Engineering. CAiSE 2014. Lecture Notes in Computer Science, vol 8484. Springer, Cham

[10] Chopra, A.K., Dalpiaz, F., Giorgini, P., Mylopoulos, J.: Reasoning about agents and protocols via goals and commitments. In: AAMAS. (2010) 457{464

[11] R. E. Tarjan, “Depth-first search and linear graph algorithms,”
SIAM J. Comput., vol. 1, no. 2, pp. 146–160, 1972.

[12] Flavio Bonomi, Rodolfo Milito, Jianz Zhi, Sateesh Addepalli, "Fog computing and its role in the internet of things",in Proceedings of the first edition of the MCC workshop on Mobile cloud computing, 2012.

[13] O.M.G., “Meta Object Facility (MOF) Core Specification, ”Management, 2006.

[14] A.C. Completion and C. Outline, “MOF Essentials,” Library

[15] Atkinson, C.; Kuhne, T., " Model-driven development: a metamodeling foundation", 2003

[16] Sridhar Srinivasa Iyengar," Metadata driven system for effecting extensible data interchange based on universal modeling language (UML), meta object facility (MOF) and extensible markup language (XML) standards", (2005)

[17] O.M.G., “OMG Systems Modeling Language (OMG SysML ™),” Source, 2010.

[18] John Mylopoulos,Lawrence Chung, Eric Yu, “From object-oriented to goal-oriented requirements analysis” Magazine Communications of the ACM CACM Homepage archive Volume 42 Issue 1, Jan. 1999 Pages 31-37

[19] E. Foundation. (2015, Sep.) Eclipse Modeling Framework. [Online]. http://www.eclipse.org/modeling/emf/

[20] Wikipedia. (2015, Sep.) Meta Object Facility -Wikipedia. [Online]. http://en.wikipedia.org/wiki/Meta-Object_Facility

[21] Iván García-Magariño, Rubén Fuentes-Fernández, Jorge J Gómez-Sanz,"Guideline for the definition of EMF metamodels using an Entity-Relationship approach", (2009)

[22] A. Lapouchnian. Goal-Oriented Requirements Engineering: An Overview of the Current Research. Depth Report, University of Toronto, 2005.

[23] Yu E, Mylopoulos J. Why goal-oriented requirements engineering. In: Dubois E,Opdahl AL, Pohl K (eds). Proceedings of the 4th international workshop on requirements engineering:foundations of software quality,Pisa, Italy, 8–9 June 1998,Pisa, Italy. Presses Universitaires de Namur, 1998, pp 15–22

[24] Dardenne, A., van Lamsweerde A. and Fickas, S., “Goal Directed RequirementsAcquisition”, Science of Computer Programming, Vol. 20, No.1-2, pp. 3–50, 1993

[25] ITU -Telecommunication Standardization Sector Draft Specification of the Goal-oriented Requirement Language (Z.151), September 2013.

[26] Anton, A.I., Goal Identification and Refinement in the Specification of Software-Based Information Systems, Ph.D. Dissertation, Georgia Institute of Technology,Atlanta GA, 1997.

[27] Mylopoulos, J., Kolp, M., and Castro, J. “UML for Agent-Oriented Software Development:The Tropos Proposal,” in Proceedings of UML 2001

[28] Rolland, C., Souveyet, C., and Ben Achour, C. “Guiding goal modeling using scenarios,” IEEE Trans. Software Eng., vol. 24, pp. 1055–1071, Dec. 1998.91.

[29] Yu, E.S.K. “Towards modelling and reasoning support for early-phase requirements engineering,” in Proceedings of the Third IEEE International Symposium on Requirements Engineering, Annapolis, Maryland, January 1997

[30] J. Mylopoulos, L. Chung, and E. Yu, "From object-oriented to goal-oriented requirements analysis," Commun. ACM, vol. 42, no. 1, pp. 31--37,1999.

[31] Amit K. Chopra, Fabiano Dalpiaz, Paolo Giorgini, and John Mylopoulos.Modeling and reasoning about service-oriented applications via goals and commitments. In Proceedings of the 22nd international conference on Advanced information systems engineering, CAiSE'10, pages 113—128.Springer-Verlag, 2010

[32] Amit K. Chopra, John Mylopoulos, Fabiano Dalpiaz, Paolo Giorgini, and unindar P. Singh. Requirements as Goals and Commitments too. In Intentional Perspectives on Information Systems Engineering, chapter 8,pages 137--153. Springer, 2010.

[33] Y. Wang, S. a McIlraith, Y. Yu, and J. Mylopoulos, “An automated approach to monitoring and diagnosing requirements,” Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering -ASE ’07,2007, p. 293.

[34] Lorena Castaneda, Norha M. Villegas, and Hausi A.Muller. Towards personalized web-tasking: Task simplification challenges. In Proceedings 1st Workshop on Personalized Web-Tasking (PWT 2013) at Ninth IEEE World Congress on Services (SERVICES 2013), pages 147–153. IEEE, 2013.

[35] Hausi A. Muller, Joanna Ng, Eleni Stroulia, Kostas Kontogiannis, Norha M. Villegas, and Diana Lau. Personalized Web-Tasking Workshop Organizers’ Message. In Proceedings 1st Workshop on Personalized Web-Tasking (PWT 2013) at Ninth IEEE World Congress on Services (SERVICES 2013), page xvii.IEEE, 2013.

[36] Betty H. Cheng, Rogerio Lemos, Holger Giese, Paola Inverardi, Jeff Magee, Jesper Andersson, Basil Becker, Nelly Bencomo, Yuriy Brun, Bojan Cukic, Giovanna Marzo Serugendo, Schahram Dustdar, Anthony Finkelstein, Cristina Gacek, Kurt Geihs, Vincenzo Grassi,Gabor Karsai, Holger M. Kienle, Jeff Kramer, Marin Litoiu,Sam Malek, Raffaela Mirandola, Hausi A.Muller, Sooyong Park, Mary Shaw,

Matthias Tichy, Massimo Tivoli, Danny Weyns, and Jon Whittle. Software engineering for self-adaptive systems: A research roadmap.Software engineering for self-adaptive systems,Lecture Notes in Computer Science 5525:1–26, 2009.

[37] Rogerio Lemos, Holger Giese, Hausi A.Muller, Mary Shaw, Jesper Andersson, Marin Litoiu, Bradley Schmerl, Gabriel Tamura,Norha M. Villegas, Thomas Vogel, Danny Weyns, Luciano Baresi, Basil Becker, Nelly Bencomo, Yuriy Brun, Bojan Cukic, Ron Desmarais, Schahram Dustdar, Gregor Engels, Kurt Geihs, Karl M. Goschka, Alessandra Gorla, Vincenzo Grassi, Paola Inverardi,Gabor Karsai, Jeff Kramer, Antonia Lopes,Jeff Magee, Sam Malek, Serge Mankovskii,Raffaela Mirandola, John Mylopoulos, Oscar Nierstrasz, Mauro Pezze, Christian Prehofer,Wilhelm Schafer, Rick Schlichting, Dennis B.Smith, Joao Pedro Sousa, Ladan Tahvildari,Kenny Wong, and Jochen Wuttke. Software engineering for self-adaptive systems: A second research roadmap. In Software Engineering for Self-Adaptive Systems II, volume 7475 of Lecture Notes in Computer Science, pages 1–32. Springer, 2013.

[38] Hausi A. Muller, Holger M. Kienle, and Ulrike Stege. Autonomic computing: Now you see it, now you don't. Design and evolution of autonomic software systems. Lecture Notes in Computer Science, 5413:32-54, 2009.

[39] Exploiting Social Context in Personalized Web-Tasking Applications

[40] Atlam, H.F.; Alenezi, A.; Walters, R.J.; Wills, G.B. An Overview of Risk Estimation Techniques in Risk-based Access Control for the Internet of Things. In Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security (IoTBDS 2017), Porto, Portugal, 24–26 April 2017; pp. 254–260.

[41] Giang, N.; Kim, S.; Kim, D.; Jung, M.; Kastner, W. Extending the EPCIS with Building Automation Systems: A New Information System for the Internet of Things. In Proceedings of the 2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, Birmingham, UK, 2–4 July 2014; pp. 364–369.

[42] Atlam, H.F.; Alenezi, A.; Hussein, R.K.; Wills, G.B. Validation of an Adaptive Risk-Based Access Control Model for the Internet of Things. Int. J. Comput. Netw. Inf. Secur. 2018,10, 26–35

[43] Atlam, H.F.; Alenezi, A.; Alharthi, A.; Walters, R.; Wills, G. Integration of cloud computing with internet of things: challenges and open issues. In Proceedings of the 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Exeter, UK, 21–23 June 2017; pp. 670–675.

[44] Ai, Y.; Peng, M.; Zhang, K. Edge cloud computing technologies for internet of things: A primer. Digit. Commun. Netw. 2017, in press.

[45] Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog computing and its role in the internet of things.
In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing-MCC '12, Helsinki,
Finland, 17 August 2012; pp. 13–15.

[46] Yi, S.; Hao, Z.; Qin, Z.; Li, Q. Fog computing: Platform and applications. In Proceedings of the 3rd Workshop
on Hot Topics in Web Systems and Technologies, HotWeb 2015, Washington, DC, USA, 24–25 October 2016;
pp. 73–78.