ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Περιβάλλον-Πλαίσιο Προσαρμοστικής Παρακολούθησης Συστημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Γεωργίου Παπαδόπουλου

Επιβλέπων : Κωνσταντίνος Κοντογιάννης
Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2014

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

Περιβάλλον-Πλαίσιο Προσαρμοστικής Παρακολούθησης Συστημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Γεωργίου Παπαδόπουλου

Επιβλέπων : Κωνσταντίνος Κοντογιάννης
Αναπληρωτής Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 1η Φεβρουαρίου 2014.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Κωνσταντίνος Κοντογιάννης	Ιωάννης Βασιλείου	Γιώργος Στάμου
Αναπλ. Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Λέκτορας Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2014

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

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα μου, Αναπληρωτή Καθηγητή Ε.Μ.Π. κύριο Κώστα Κοντογιάννη για το ενδιαφέρον και την υπομονή που επέδειξε κατά την εκπόνηση της διπλωματικής μου εργασίας. Η καθοδήγηση και η βοήθειά του υπήρξαν καθοριστικές και συνετέλεσαν ώστε να αποκομίσω τα μέγιστα από την παρούσα εργασία. Μου μετέδωσε πολλές χρήσιμες γνώσεις γύρω από τον τομέα της Τεχνολογίας Λογισμικού, έναν τομέα άγνωστο για μένα. Τον ευχαριστώ θερμά για την υπομονή του και την προσπάθεια του να με διδάξει και να μοιραστεί μαζί μου τις γνώσεις του.

Θα ήθελα επίσης να ευχαριστήσω τους ερευνητές του εργαστηρίου Τεχνολογίας Λογισμικού κ. Γιώργο Χατζηκωνσταντίνου και κ. Μιχαλη Αθανασόπουλο, καθώς επίσης και τα υπόλοιπα μέλη του εργαστηρίου για την συνεργασία τους, τις συμβουλές τους και τις ιδέες που μοιράστηκαν μαζί μου.

Σε αύτο το σημείο δεν θα μπορούσα να παραλείψω τους συμφοιτητές και τους φίλους μου. Θα ήθελα να τους ευχαριστήσω για τις όμορφες και δύσκολες στιγμές που περάσαμε μαζί. Δουλέψαμε, συνεργαστήκαμε και μάθαμε πολλά ο ένας από τον άλλον μέσα από τις συζητήσεις και αντιπαραθέσεις μας. Τους ευχαριστώ εκ βάθους καρδίας, καθώς ήταν αυτοί που συνέβαλαν στην διαμόρφωση μου ως επιστήμονα και την ολοκλήρωσή μου ως άνθρωπο.

Τέλος αν και είναι δύσκολο να αποτυπωθούν με λέξεις, θέλω να δώσω τις πιο θερμές ευχαριστίες μου στην οικογένεια μου, που όλα αυτά τα χρόνια με στήριξε, με συμβούλεψε και με βοήθησε ώστε να πετύχω τους στόχους μου. Οι πιο θερμές ευχαριστίες μου πηγαίνουν στα αδέρφια μου και τους γονείς μου Περικλή και Κλειώ για την αγάπη και την εμπιστοσύνη που μου δείχνουν όλα αυτά τα χρόνια.

Γιώργος

Περίληψη

Ο στόχος της παρούσας διπλωματικής εργασίας είναι ο σχεδιασμός και η υλοποίηση ενός περιβάλλοντος-πλαισίου που θα επιτρέψει την προσαρμοστική παρακολούθηση συστημάτων λογισμικού. Η αυξανόμενη πολυπλοκότητα των σημερινών συστημάτων λογισμικού έχει καταστήσει αναγκαία την αναπτυξη αυτο-διαχειριζόμενων συστημάτων λογισμικού. Η ανάπτυξη αυτο-διαχειριζόμενων συστημάτων λογισμικού θα διευκολύνει την διαχείρηση τους, κρύβοντας την πολύπλοκοτητα τους από τους διαχειρηστές τους. Από την άλλη κάτι τέτοιο θα απαιτούσε την συνεχή παρακολούθηση και ανάλυση της συμπεριφοράς τους έτσι ώστε να ανιχνευθουν και να επιλυθούν τυχών προβλημάτα. Η συνεχής παρακολούθηση ωστόσο συνεπάγεται υψηλό υπολογιστικό κόστος και ενδέχεται να επιρρεάσει την λειτουργία και επίδοση των συστημάτων. Για τη αντιμετώπηση αυτής της πρόκλησης προτείνουμε την ανάπτυξη συστημάτων παρακολούθησης τα οποία θα είναι σε θέση να καθορίζουν σε χρόνο εκτέλεσης το σύνολο των δεδομένων που συλλέγουν και επεξεργάζονται, έτσι ώστε να ελαχιστοποιηθεί κατά το μέτρον του δυνατού το κόστος παρακολούθησης. Η προσέγγιση μας στο πρόβλημα κάνει χρήση των μοντέλων δέντρων στόχων για την μοντελοποίηση υποθέσεων για το υπο παρακολούθηση σύστημα. Οι υποθέσεις αυτές ενεργοποιούνται με την ανακάλυψη μορφημάτων των γεγονότων παρακολούθησης, και εν συνεχεία επιχειρείται η επαλήθευση τους και η ανάλυση του βασικού αιτίου τους. Η επαλήθευση μιας υπόθεσης μπορεί να απαιτήσει την συλλογή δεδομενων που δεν συλλεγόντουσαν μέχρι πρότινος. Έτσι επιτυγχάνεται η προσαρμογή του επιπέδου παρακολούθησης. Με αυτόν τον τρόπο σε κάθε δεδομένη χρονική στιγμή παρακολουθείται μόνο εκείνο το σύνολο δεδομένων που είναι απαραίτητο για την ανακάλυψη προβλημάτων και των αιτιών τους. Αυτό συντελεί στην μείωση του υπολογιστικού κόστους παρακολούθησης και στον ταχύτερο προσδιορισμό προβλημάτων καθώς μειώνεται ο όγκος των δεδομένων που επεξερταζόμαστε. Στα πλαίσια της διπλωματικής ορίσαμε την αρχιτεκτονική του περιβάλλοντος-πλαισίου και υλοποιήσαμε ένα πρωτότυπο του. Η αρχιτεκτονική αυτή είναι βασισμένη στο αρχιτεκτονικό μόρφημα Μαυροπίνακα και υποστηρίζει έμμεση κλήση μεθόδων μέσω μιας υπηρεσίας publisher/subscriber. Επιπλέον ορίσαμε ένα μοντέλο για τις υποθέσεις, βασισμένο στα δέντρα στόχων, και παρέχουμε μια διεπαφή χρήστη για τον ορισμό τους. Τέλος μετασχηματίζουμε τα δέντρα που αντιστοιχούν σε κάθε υπόθεση, σε αυτές σε κανονική συζευκτική μορφή και προτείνουμε την χρήση έτοιμων(off-the-self) SAT-Solvers για την ανάλυση τους.

Λέξεις Κλειδιά: <<Δέντρα Στόχων, autonomic computing, παρακολούθηση λογισμικού, καταγραφή, προτασιακή ικανοποιησιμότητα, τεχνολογία λογισμικού, ανάλυση βασικού αιτίου>>

Abstract

The main goal of this diploma thesis is the design and implementation of a goal driven framework that allows for adaptive monitoring of software systems. The increasing complexity of today's software systems complexity has necessitated the development of adaptive and self-managed software systems. The development of self-managed systems will facilitate their management by hiding the systems' complexity from the administrators and engineers. However a self-managed system requires continious monitoring and analysis of the data collected in order to detect, verify and solve problems. Such a solution would impose a heavy load on system resources and influence the overall system's performance. Towards a solution to this problem we propose the development of adaptive monitoring systems, able to determine at the runtime, which data should be collected and analyzed, minimizing that way the monitoring cost. Our approach to this challenge utilizes goal trees to model and define hypotheses about the system's state. These hypotheses get activated upon detecting significant patterns of events in the event stream, and then their verification and root-cause analysis is attempted. Verifying a hypothesis may require collecting additional data that were not being collected before. This results in the adaptation of the monitoring level. This way only the set of data, necessary to detect a problem and determine it's root cause, is being monitored resulting in lower monitoring cost and faster problem determination since less data are being processed. For the purposes of this work, we defined the architecture of a framework that allows for adaptive monitoring and developed a prototype of that framework. The architecture is based on the Blackboard architectural pattern and supports indirect calls between its components by using a publisher/subscriber service. Furthermore we defined a model for the hypotheses, based on the goal trees model, and we provide a graphical user interface so as to define them. Finally we transform the goal trees in CNF and propose the use of SAT-Solvers to analyze them.

Keywords: <<goal tree, autonomic computing, software monitoring, logging, propositional satisfiability, software engineering, root-cause analysis>>

Πίνακας Περιεχομένων

Πίνακας Σχημάτων

Κεφάλαιο 1: Εισαγωγή

Η ακόλουθη εισαγωγή ξεκινά με το κίνητρο για την δημιουργία προσαρμοστικών συστήματων παρακολούθσης λογισμικού, δηλαδή λογισμικού που μπορεί να διαχειριστεί τον εαυτό του και να προσαρμόζει κατάλληλα το είδος των δεδομένων προς συλλογή και την λεπτομέρεια ανάλυσης τους ώστε να μειώσει το κόστος παρακολούθησης και την πετύχει καλύτερη απόδοση (κεφάλαιο 1.1). Οι προκλήσεις που παρουσιάζονται στην προσπάθεια σχεδιασης και ανάπτυξης προσαρμοστικών συστήματων παρακολούθησης συζητούνται στο (κεφάλαιο 1.2) και χρησιμοποιούνται για την εξαγωγή των ερευνητικών ερωτήσεων που μελετώνται στο πλαίσιο της παρούσας εργασίας. Στη συνέχεια, συνοψίζονται οι επιστημονικές συνεισφορές της παρούσας εργασίας (κεφάλαιο 1.3) και περιγράφεται η δομή της (κεφάλαιο 1.4).

1.1 Κίνητρο

Οι ηλεκτρονικές υπηρεσίες παίζουν ουσιαστικό ρόλο στις σύγχρονες κοινωνίες. Πολλές εργασίες στην καθημερινή μας ζωή απαιτούν την χρήση διαδικτυακών υπηρεσιών που προσφέρονται απο τράπεζες, επιχειρήσεις και οργανισμούς. Παραδείγματα τέτοιων υπηρεσιών αποτελούν το ηλεκτρονικό ταχυδρομείο, το ηλεκτρονικό εμπόριο, ηλεκτρονικές τραπεζικές υπηρεσίες, δημόσιες ηλεκτρονικές υπηρεσίες κ.α. Ομοίως πολλές επιχειρήσεις στηρίζονται στα υπολογιστικά τους συστήματα για να παράσχουν αυτές τις υπηρεσίες σε χρήστες και άλλες επιχειρήσεις ή οργανισμούς. Επιπλέον πολλές επιχειρήσεις σήμερα παρέχουν μόνο διαδικτυακές υπηρεσίες και έτσι εξαρτώνται πλήρως απο τα υπολογιστικά τους συστήματα. Όσο αυξάνεται ο αριθμός και η πολυπλοκότητα των ηλεκτρονικών υπηρεσιών τόσο αυξάνεται η εξάρτηση των επιχειρήσεων από τις υπηρεσίες αυτές και οι υπηρεσίες αυτές γίνονται όλο και πιο σημαντικές για την εξέλιξη των επιχειρήσεων. Τα συστήματα αυτά συνήθως υπόκεινται σε υψηλές απαιτήσεις αξιοπιστίας και επιδόσεων. Πρέπει να λειτουργούν σωστά όλο το εικοσιτετράωρο και με όσο το δυνατόν καλύτερη επίδοση.

Τα παραπάνω έχουν ως αποτέλεσμα την δημιουργία όλο και μεγαλύτερων και πιο περίπλοκων υπολογιστικών συστημάτων. Τα σύγχρονα συστήματα είναι συνήθως μεγάλα, περίπλοκα, κατανεμημένα και αποτελούνται από πολλά συστατικά στοιχεία, και στρώματα (layers). Η πολυπλοκότητα αυτή αυξάνει την δυσκολία διαχείρησης των συστημάτων λογισμικού. Ένας άλλος παράγοντας που συντελεί στην αύξηση της πολυπλοκότητας είναι ο ανταγωνισμός των επιχειρήσεων.

Καθώς τα συστήματα λογισμικού γίνονται όλο και πιο σηματνικά για τις επιχειρήσεις η ταχύτητα αναπτυξης και βελτίωσης των συστημάτων αυτών και των υπηρεσιών που αυτά παρέχουν είναι κριτικής σημασίας για την επιτυχία των επιχειρήσεων. Η ανάθεση μέρους της ανάπτυξης λογισμικού σε άλλες εταιρίες (outsourcing) είναι μια λύση που χρησιμοποιούν οι επιχειρήσεις προκειμένουν να μειώσουν το κόστος και τον χρόνο αναπτυξης του συστηματος. Μια άλλη λύση είναι η χρήση εμπορικών συστημάτων λογισμικού (Commercial-Off-The-Shelf (COTS)). Και οι δύο λύσεις αυξάνουν την πολυπλοκότητα του συστήματος μπορεί να οδηγήσουν σε συστηματα τα οποία είναι δύσκολο να γίνουν πλήρως κατανοητά ακόμα και από

τους προγραμματιστές που είναι υπεύθυνοι για ολόκληρο το σύστημα.

Για την εξασφάλιση της ορθής λειτουργίας των συστημάτων λογισμικού απαιτείται η συνεχής παρακολούθηση τους με σκοπό την εγκαιρη ανίχνευση βλαβών και παραβιάσεων. Σύμφωνα με το [1] οι πιο συχνοί τύποι αστοχιών είναι η προσωρινή πτώση του συστήματος, εξαιρέσεις και παραβιάσεις πρόσβασης, απώλεια ή φθορά δεδομένων και πτώση της επίδοση. Τέτοιες αστοχίες μπορεί να έχουν ώς συνέπεια γενικώς πολύ υψηλό κόστος, όπως για παράδειγμα απώλεια εσόδων, βλάβη της φήμης ή συγκεκριμένες ποινές στην περίπτωση που παραβιαστεί κάποιο σύμφωνο παροχής υπηρεσιών (Service-Level-Agreement (SLA)). Έτσι οι επιχειρήσεις ξοδεύουν μεγάλα χρηματικά ποσά για την παρακολούθηση και διαχείρηση της υποδομών των υπολογιστικών συστημάτων τους.

Παραδοσιακά οι οργανισμοί και οι επιχειρήσεις βασίζονταν σε εξειδικευεμένο προσωπικό που αναλάμβανε να επιβλέπει την υπολογιστική υποδομή, να αναγνωρίζει προβλήματα, να διαγιγνώσκει της αιτίες των προβλημάτων και να επιδιορθώνει τυχών προβλήματα. Αυτή η εξάρτηση από το ανθρώπινο δυναμικό για την διαχείρηση των υπολογιστικών συστημάτων είναι όμως προβληματική απο πολλές απόψεις.

- Πρώτον είναι ακριβή. Οι διαχειριστές των συστημάτων που έχουν την γνώση και τις ικανότητες να διαχειριστούν μεγάλα και πολύπλοκα συστήματα είναι λίγοι και ως εκ τούτου ακριβοί. Επιπλέον η αύξηση της πολυπλοκότητας των συστημάτων θα απαιτούσε περισσότερους και πιο καταρτισμένους διαχειριστες, πράγμα που συνεπάγεται υψηλό κόστος. Σύμφωνα με στατιστικές ο αριθμός αυτών των διαχειριστών στην Αμερική το 2004 υπολογίζεται σε 900.000 και αναμένεται να αυξηθεί κατά 30% μέχρι το 2014 [49].
- Δεύτερον δεν κλιμακώνει. Αυξηση του αριθμού των διαχειριστών δεν διευκολύνει την διαχείρηση του συστήματος. Απεναντίας, με περισσότερους ανθρώπους, ο σωστός συντονισμός, και επικοινωνία μεταξύ τους, δυσχεραινεται.
- Τρίτον δεν είναι αποτελεσματική. Ενώ η λύση αυτή μπορεί να παρέχει μια βραχυπρόθεσμη ανακούφηση, αποτυγχάνει να λύσει το μακροπρόθεσμο πρόβλημα του ότι η πολυπλοκότητα των συστημάτων λογισμικού φτάνεισε επίπεδα που ξεπερνά τις δυνατότητες των διαχειριστών [76]. Η διαχείρηση των συστημάτων αύτών γίνεται πολύ δύσκολη, καθώς γίνεται πιο δύσκολη η κατανόηση των συστημάτων αυτών και των συνεπειών μια διαχειριστική ενέργειας σε αυτά. Ακόμα και τώρα, η αποτελεσματικότητα των διαχειριστών είναι αμφισβητίσιμη. Ενα παράδειγμα αποτελεί η μελέτη [139], συμφωνα με την οποία το 40% των αστοχιών των συστημάτων αποδίδεται σε λάθος των διαχειριστών.
- Τέταρτον δεν είναι αποδοτική. Η ανίχνευση και η επιλυση λαθών και αστοχιών που γίνεται απο διαχειριστές είναι συνήθως αργή και χρονοβόρα. Εξαιτίας του υψηλού κόστους, δεν είναι δυνατόν για τους διαχειριστές να επιβλέπουν αδιάκοπα το σύστημα. Επιπλέον η παρακολούθηση του συστήματος απο διαχειριστές είναι χρονοβόρα για τον λόγο ότι οι διαχειριστές πρέπει να βρούν σχετικές πληροφρορίες μέσα από ένα μεγάλο σύνολο πολύπλοκων δεδομένων. Ως αποτέλεσμα, είναι σύνηθες πολλά λάθη και αστοχίες να ξεφεύγουν της προσοχής τους και να μην γίνονται αντιληπτά για μεγάλες περιόδους. [24], και συχνά να έρχονται στο φως μόνο μέσα από αγανακτησμένους χρήστες. Έτσι το αποτέλεσμα είναι μειωμένη διαθεσιμότητα του συστήματος, που με την σειρά της οδηγεί σε ανεπιθυμητες συνέπειες για τις επιχειρήσεις.

Έτσι είναι κριτικής σημασίας να αναπτυχθούν αυτοματοποιημένες προσεγγίσεις για την παρακολούθηση συστημάτων λογισμικού. Αυτές οι προσεγγίσεις οδηγούν στην μείωση του κόστους διαχείρησης των συστημάτων αυτών καθώς απαιτεί λιγοτερους διαχειριστές, και επιτρέπει την εξασφάλιση αποδοτικής διαχείρησης όλο και μεγαλύτερων συστημάτων.

Για την αντιμετώπιση αυτής της πρόκλησης, η ιδέα των αυτοδιαχείριζομενων συστημάτων έχει λάβει αρκετή προσοχή τόσο από την επιστημονική κοινότητα όσο και απο την βιομηχανία [18, 40, 54, 94]. Ο όρος *autonomic computing* [76] έχει εισαχθεί από την IBM για να αναφερόμαστε σε αυτο-διαχειριζόμενα συστήματα. Ο τελικός στόχος είναι η δημιουργία συστημάτων λογισμικού που θα έχουν την δυνατότητα να διαχειρίζονται τον εαυτό τους, μειώνοντας κατά το δυνατό την ανάγκη ανθρώπινης επέμβασης. Η διαχείριση των συστημάτων εκτείνεται σε ένα ευρύ φάσμα δραστηριοτήτων σχετικών με την λειτουργία του συστήματος, όπως πχ την διαμόρφωση του συστήματος, την επίδοση, performance (διασφάλιση του ότι οι στόχοι για την επίδοση του συστήματος πληρούνται), την ασφάλεια, και τον τον προσδιορισμό προβλήματων του συστήματος. Αν και η ιδέα των αυτο-διαχειριζόμενων συστημάτων μπορεί να εφαρμοστεί σε όλες τις παραπάνω δραστηριότητες, η παρούσα εργασία εστιάζει στην παρακολούθηση του συστήματος και στον προσδιορισμό προβλημάτων.

Η εφαρμογή της ιδέας των αυτο-διαχειριζόμενων συστημάτων στα συστήματα παρακολούθησης συνεπάγεται την δημιουργία συστημάτων παρακολούθησης που συλλέγουν δεδομένα απο την λειτουργία ενος υπολογιστικού συστήματος, τα αναλύουν με σκοπό την ανίχνευση αστοχιών και παραβιάσεων και διαμορφώνουν τον εαυτό τους σε χρόνο εκτέλεσης (πχ ορίζοντας το είδος των πληροφοριών που συλλέγονται, την συχνότητα συλλογής σε περίπτωση δειγματοληπτικού ελέγχου των δεδομένων παρακολούθησης κ.α.) με σκοπό τον προσδιορισμό των αιτιών των προβλημάτων αυτών. Αυτό οδηγεί στην μείωση της εξάρτησης των διαδικασιών που σχετίζονται με την παρακολούθηση του συστήματος και τον προσδιορισμό προβλημάτων από τον ανθρώπινο παράγοντα, το οποίο με την σειρά του μπορεί να συντελέσει στην μειωση σφαλμάτων οφειλόμενων στην υποκειμενική κρίση των διαχειριστών του συστήματος, στην μείωση του κόστους παρακολούθησης(τόσο σε χρήμα, όσο και σε υπολογιστικούς πόρους), στην ταχύτερη ανίχνευση αστοχιών, στον ταχύτερο προσδιορισμό των αιτιών των προβλημάτων, και στην ταχύτερη αντιμετώπιση προβλημάτων. Η μείωση του χρηματικού κόστους έρχεται ώς αποτέλεσμα της μείωση των φυσικών διαχειριστών του παρακολουθούμενου συστήματος ενώ η μείωση του υπολογιστικού κόστους σχετίζεται με την δυνατότητα των προσαρμοστικών συστημάτων παρακολούθησης να προσδιορίζουν σε χρόνο εκτέλεσης τα δεδομένα που πρέπει να συλλεγούν και να αναλυθούν.

Η παρακολούθηση ενός συστήματος λογισμικού απαιτεί υπολογιστικούς πόρους για την ανάλυση των δεδομένων παρακολούθησης και χώρο για την αποθήκευση τους. Όσο το σύστημα λειτουργεί σωστά η παρακολούθηση του συνεπάγεται απλά κατανάλωση των πόρων αυτών με πιθανές επιπτωσεις στην απόδοση του παρακολουθούμενου συστήματος. Έτσι σε αυτήν την περίπτωση είναι επιθυμητό να συλλέγονται και να αναλύονται μόνο δεδομένα που είναι είναι απαραίτητα για την ανίχνευση προβλημάτων. Ως εκ τούτου μειώνεται ο όγκος των δεδομένων αυτών, διευκολύνοντας έτσι την διαχείρηση τους και επιτυγχανοντας ταχύτερη ανίχνευση προβλημάτων, ταχύτερη επιδιόρθωση τους και περιορισμο των συνεπειών τους.

1.2 Περιγραφή Προβλήματος

Ο στόχος της εργασίας αυτής είναι η ανάπτυξη ενός περιβάλλοντος-πλαισίου που θα αποτελέσει την βάση για την ανάπτυξη προσαρμοστικών συστημάτων παρακολούθησης και τον προσδιορισμό προβλημάτων. Συγκεκριμένα το περιβάλλον-πλαίσιο θα παρέχει στα συστήματα παρακολούθησης, την υποδομή να ανιχνεύουν πιθανά προβλήματα και να προσαρμόζουν κατάλληλα το είδος των πληροφοριών που συλλέγουν και αναλύουν ώστε να μπορούν να αποφανθούν για την ύπαρξη προβλημάτων.

Το σύστημα θα συλλέγει ένα σύνολο πληροφοριών ικανών για το σύστημα να κάνει υποθέσεις για πιθανά προβλήματα. Στην συνέχεια οι υποθέσεις αυτές πρέπει να επαληθευτούν. Η

επαλήθευση τους θα γίνει με παρακολούθηση επιπλέον πληροφοριών σχετικών με την υπόθεση αυτή. Σε περίπτωση αποτυχίας επαλήθευσης μιας υπόθεσης οι επιπλέον πληροφορίες παύουν να συλλέγονται.

Με βάση τα παραπάνω για κάθε υπόθεση πρέπει να ορίσω δύο σύνολα πληροφοριών. Ένα σύνολο που θα μου επιτρέπει να ανιχνεύσω την πιθανότητα η υπόθεση αυτή να ισχύει και ένα δεύτερο σύνολο που θα μου επιτρέπει την απόδειξη μιας υπόθεσης. Έπιπλέον πρέπει να δοθεί προσοχή στην επιλογή της μεθόδου συλλογιστικής για την ενεργοποίηση και την απόδειξη μια υπόθεσης, και να οριστεί ένα σύνολο κανόνων για την προσαρμογή του επιπέδου παρακολούθησης.

Έτσι προκύπτουν τα εξής ερωτήματα που μας απασχόλησαν κατά την σχεδίαση του περιβάλλοντος-πλαισίου:

- Ποιο είναι το σύνολο των πληροφορίων που απαιτούνται για να κάνει το σύστημα παρακολούθησης μια υπόθεση για το σύστημα στόχο;

- Πως μπορούμε να μοντελοποιήσουμε τους κανόνες προσαρμογής του επιπέδου παρακολούθησης;

- Πως μπορεί να δωθεί το σύνολο των υποθέσεων σε ένα μοντέλο γενικό και επεκτάσιμο;

- Πως μπορούμε να αποφανθούμε για την ισχύ ή την άρνηση μιας υπόθεσης; Πιο συγκεκριμένα ποια τεχνική συλλογιστικής είναι η καταλληλότερη για την επαλήθευση μιας υπόθεσης;

- Σε περίπτωση αδυναμίας απόδειξης μια υποθέσεως μπορουμε με ασφάλεια να αποφανθούμε για την αρνηση της. Με άλλα λόγια μπορούμε να πούμε ότι μια υπόθεση ήταν λανθασμένη και να σταματήσουμε την συλλογή επιπλέον δεδομένων παρακολούθησης, όντας σίγουροι ότι η συνέχιση της συλλογής των δεδομένων αυτών δεν θα οδηγήσει στην επαλήθευση της υπόθεσης αυτής; Αν ναι πότε μπορούμε να αποφανθούμε και σε τί μπορούμε να βασίσουμε αυτήν την απόφαση;

1.3 Συνεισφορά της Διπλωματικής Εργασίας

Η παρούσα διπλωματική εργασία, παρουσιάζει τον σχεδιασμό και την υλοποίηση ενός περιβάλλοντος-πλαισίου λογισμικού, το οποίο καλείται να επιλύσει το πρόβλημα της προσαρμοστικής παρακολούθησης συστημάτων λογισμικού. Η συνεισφορά της παρούσας διπλωματικής εργασίας συνοψίζεται στα εξής σημεία:

- Τον ορισμό την υλοποιηση και την ποσοτική αξιολόγηση μιας αρχιτεκτονικής αναφοράς για το περιβάλλον-πλαίσιο, η οποία υλοποιεί το αρχιτεκτονικό μόρφημα μαυροπίνακα (blackboard architectural pattern), και επιτρέπει στα συστήματα παρακολούθησης που το επεκτείνουν να κάνουν υποθέσεις για το υπό παρακολούθηση σύστημα, και να τις επαληθεύουν βασιζόμενα σε πολιτικές που μπορούν να ορίσουν οι χρήστες, αλλαζοντας αν χρειαστεί το επιπέδο παρακολούθησης ανάλυσης των δεδομένων.

- Την επέκταση του Μοντέλου Δένδρων Στόχων (goal model), που αποτελεί εργαλείο της Μηχανικής Απαιτήσεων (Requirements Engineering), με επισημειώσεις και

alarms, για την μοντελοποίηση των υποθέσεων και των κανόνων επαλήθευσης τους. Τα alarms μοντελοποιούν οντότητες που σηματοδοτούν την εύρεση σημαντικών γεγονότων παρακολούθησης και εκκινούν την διαδικασία ενεργοποίησης και επαλήθευσης μιας υπόθεσης. Οι επισημειώσεις μοντελοποιούν ενέργειες που πρέπει να γίνουν και πληροφορίες που απαιτούνται για την επαλήθευση μιας υπόθεσης. Να τονίσουμε ότι το μετα-μοντέλο αυτό είναι γενικό με την έννοια ότι δεν κάνει υποθέσεις για το υπό παρακολούθηση σύστημα και έτσι μπορεί να χρησιμοποιηθεί για την μοντελοποίηση οποιουδήποτε συστήματος. Επιπλέον είναι επεκτάσιμο και μπορεί να επεκταθεί περιγράψει πιο σύνθετα μοντέλα. Τέλος είναι εύκολα κατανοήσιμο και διαχειρίσιμο από τους διαχειριστές του συστήματος και εύκολα διαχειρίσιμο προγραμματιστικά. Για την κατασκευή του μοντέλου δένδρων στόχων και τον ορισμό των δενδρων στόχων χρησιμοποιήθηκε το EMF, ένα εργαλείο της τεχνολογίας λογισμικού για την πλατφόρμα του Eclipse. Η χρήση του EMF βοήθησε στην επιτάχυνση της ανάπτυξης του περιβάλλοντος-πλαισίου και παρέχει μια γραφική διεπαφή για τον ορισμό και την εύκολη διαχείρηση των δένδρων στόχων.

- Την αναγωγή του προβλήματος της καταστρωσης πλάνου για την απόδειξη μιας υπόθεσης, και την προσαρμογή του επιπέδου παρακολούθησης (adaptation planning) στο πρόβλημα της ικανοποιησιμότητας λογικών εκφράσεων. Η επίλυση του προβλήματος στην συνέχεια γίνεται με χρήση κάποιου υπάρχοντος SAT-Solver, στην περίπτωσή μας του SAT4J. Το πρόβλημα της ικανοποιησιμότητας είναι ένα γνωστό και καλά θεμελιωμένο πρόβλημα με πλήθος αποδοτικών υλοποιήσεων και εφαρμογών. Η εφαρμογή του στην κατάστρωση πλάνων (planning) είναι γνωστή. Εδώ παρουσιάζουμε ένα παράδειγμα εφαρμογής αυτής της ιδέας.

1.4 Οργάνωση του Κειμένου

Το υπόλοιπο του κειμένου οργανώνεται ως εξής:

Στο 2ο κεφάλαιο παρουσιάζονται όλες οι σχετικές με την παρουσα εργασία, εργασίες και τεχνολογίες οι οποίες βοήθησαν με διάφορους τρόπους στην εκπόνηση της. Αναφέρονται οι έννοιες της μοντελοποίησης λογισμικού αλλά και η ιδέα των Δέντρων Στόχων στις οποίες βασίστηκε η μοντελοποιηση των υποθέσεων και των κανόνων επαλήθευσης τους και προσαρμογής του συστήματος παρακολούθησης. Παρουσιάζονται τα εργαλεία που βοήθησαν στην υλοποίηση του περιβάλλοντος-πλαισίου και στον ορισμό των Δέντρων Στόχων, με ποιο σημαντικά το EMF και την βιβλιοθήκη SAT4J. Τέλος δίνεται μια επισκόπηση εργασιών από το πεδίο του Autonomic Computing, της παρακολούθησης συστημάτων και της τήρησης αρχείων καταγραφής με ειδική μνεία στο Common Base Event.

Στο 3ο κεφάλαιο παρουσιάζονται τα αποτελέσματα της σχεδίασης του περιβάλλοντος-πλαισίου. Δίνεται η αρχιτεκτονική του περιβάλλοντος-πλαισίου και αναλύονται ένα προς ένα όλα τα δομοστοιχεία (components) που την απαρτίζουν. Ορίζεται επίσης η διαπροσωπία του κάθε δομοστοιχείου και οι αλληλεπίδρασεις μεταξύ τους.

Στο 4ο κεφάλαιο παρουσιάζεται ένα τροποποιημένο μοντέλο των δεντρων στόχων, το οποίο χρησιμοποιείται για την μοντελοποίηση των υποθέσεων και των κανόνων επαλήθευσης τους και προσαρμογής του συστήματος παρακολούθησης. Οι υποθέσεις αποτελούν τις ρίζες των δέντρων ενώ η δομή του δίνει τους τρόπους επαλήθευσης τους.Τα απαραίτητα δεδομένα προς παρακολούθηση με σκοπό την επαλήθευση μιας υπόθεσης, μοντελοποιούνται ως επισημειώσεις πάνω στα φύλλα των δέντρων.

Στο 5ο κεφάλαιο παρουσιάζουμε τους κανόνες που χρησιμοποιούνται για την οδήγηση του SAT-Solver και την μετατροπή των δέντρων στόχων στους κανόνες αυτούς. Στην συνέχεια

παρουσιάζουμε λεπτομέρειες του βρόχου ελέγχου που χρησιμποιεί το περιβάλλον-πλαίσιο και την αξιοποίηση του SAT-Solver. Τελειώνοντας το κεφάλαιο, δίνουμε ένα παράδειγμα λειτουργίας του περιβάλλοντος-πλαισίου.

Στο 6ο κεφάλαιο παρουσιάζεται η πειραματική αξιολόγηση του περιβάλλοντος-πλαισίου που αναπτύξαμε. Εξετάζονται θέματα απόδοσης της προτεινόμενης λύσης καθώς επίσης και το πως αυτή επιρεάζεται ως συνάρτηση του μεγέθους και της μορφής των μοντέλων δέντρων στόχων.

Στο 7ο κεφάλαιο ανακεφαλαιώνουμε βασικά σημεία της παρούσας εργασίας με σκοπό να εξάγουμε χρήσιμα συμπεράσματα και προτάσεις για μελλοντική έρευνα και επέκταση των ιδεών που εξετάσαμε.

Κεφάλαιο 2: Σχετικές εργασίες

2.1 Εισαγωγή

Το αντικείμενο αυτής της διπλωματικής εργασίας άπτεται σε μια σείρα επιστημονικών και ερευνητικών περιοχών. Πρώτον άπτεται της περιοχής της παρακολούθησης συστημάτων λογισμικού. Δεύτερον άπτεται της περιοχής των αυτόνομων συστημάτων και των συστημάτων παρακολούθησης λογισμικού. Τρίτον σχετίζεται με τον τομέα της συλλογιστικής και ιδιαίτερα της ασαφούς λογικής. Τέλος χρησιμοποιεί εργαλεία και τεχνικές μοντελοποίησης λογισμικου. Το παρόν κεφάλαιο αποτελεί μια επισκόπηση των περιοχών αυτών, εργασιών σχετικων με τις περιοχές αυτές καθώς επίσης και των εργαλείων και τεχνικών που χρησιμοποιήθηκαν στην ανάπτυξη του περιβάλλοντος-πλαισίου. 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 μονο στις βασικές έννοιες και στην παράθεση.

2.2 Μοντελοποίηση Συστημάτων Λογισμικού

Η Model Driven Architecture (MDA) αποτελεί την σημαντικότερη σύγχρονη μεντελοκεντρική προσέγγιση στο χώρο της τεχνολογίας λογισμικού. Είναι μια ενοποιητική προσέγγιση, η οποία έχει σαν τεχνολογική βάση μια σειρά από ήδη υπάρχουσες σχετικές τεχνολογίες, στις οποίες περιλαμβάνονται η Unified Modeling Language (UML), το Meta Object Facility (MOF), το XML Metadata Interchange (XMI) κτλ.

2.2.1 MOF

Το **Meta Object Facility (MOF)** [2][3] αποτελεί ένα πρότυπο του Object Management Group (OMG) και ένα απο τα θεμέλια της οδηγούμενης από μοντέλα μηχανικής (model-driven engineering). Το MOF προέκυψε απο την ανάγκη ορισμού της UML. Μια τέτοια γλώσσα μπορεί να δώσει την δυνατότητα να εξάγουμε μοντέλα από εφαρμογές και να τα εισάγουμε σε άλλες εφαρμογές, να τα αποθηκεύουμε και να τα φορτώνουμε να τα μεταφερουμε μέσω δικτύου και να παράγουμε κώδικα από αυτά. Το πρότυπο αυτό ορίζει μια ταξινόμηση των μοντέλων σε τέσσερα επίπεδα. Τα επίπεδά αυτά είναι τα εξής:

1. Το επίπεδο M3 ή επίπεδο μέτα-μεταμοντέλου (meta-metamodel layer). Το επίπεδο αυτό είναι το ανώτερο επίπεδο της ιεραρχίας και χρησιμοποιείται για να περιγράψει μοντέλα του επιπέδου M2 (πχ την ίδια την UML). Στο

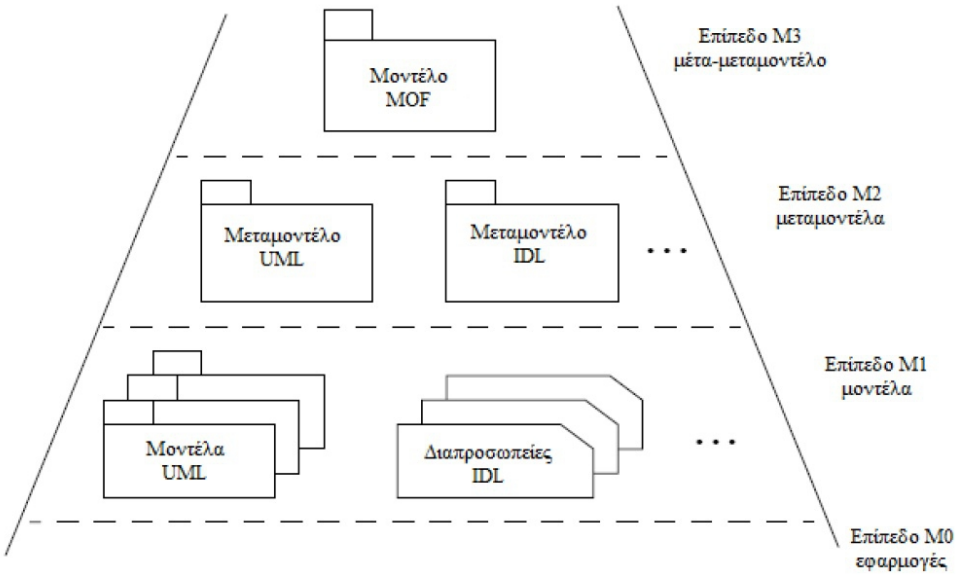
επίπεδο αυτό βρίσκονται τα μέτα-μεταμοντέλα, των οποίων ο ρόλος είναι το να ορίζουν γλώσσες με τις οποίες να είναι δυνατή η προδιαγραφή μεταμοντέλων. Σε αυτό το επίπεδο βρίσκονται MOF μοντέλα. Τέλος αξίζει να τονίσουμε ότι τα μέτα-μεταμοντέλα ορίζονται χρησιμοποιώντας μέτα-μεταμοντέλα, δηλαδή τα στοιχεία του επιπέδου M-3 ορίζονται αυτοαναφορικά (το MOF είναι ορισμένο με χρήση του ίδιου του MOF).

2. Το επίπεδο M2 ή επίπεδο μετα-μοντέλου (meta-model layer). Τα στοιχεία του επιπέδου M2 είναι τα μεταμοντέλα, δηλαδή οι γλώσσες μοντελοποίησης, τις οποίες χρησιμοποιούμε για να ορίσουμε μοντέλα. Χαρακτηριστικά παραδείγματα μεταμοντέλων είναι η UML και το CWM.

3. Το επίπεδο M1 ή επίπεδο μοντέλου (model layer). Σε αυτό το επίπεδο περιγράφονται τα μοντέλα του επιπέδου M0.

4. Το επίπεδο M0 ή επίπεδο πληροφορίας (information layer). Σε αυτό το επίπεδο βρίσκεται κάθε πληροφορία ή αντικείμενο που επιθυμούμε να μοντελοποιήσουμε. Τα αντικείμενα αυτού του επιπέδου είναι ένα στιγμιότυπου του μοντελου του επιπέδου M1.

Μια αξιοσημείωτη χρήση του επιπέδου M2 είναι ο ορισμός ενός μετα-μοντέλου που περιγράφει την ίδια την UML. Σε αυτήν την περίπτωση στο επίπεδο M2 θα βρισκόταν η ίδια η UML και στο επίπεδο M1 διαγράμματα UML που περιγράφουν ένα σύστημα λογισμικού. Στο επίπεδο M0 τέλος θα βρισκόταν η υλοποίηση αυτού του συστήματος. Στο παρακάτω σχήμα φαίνονται τα επίπεδα του MOF.



Σχήμα 2.2.1: MOF Levels

Κλείνοντας αυτήν την σύντομη περιγραφή του προτύπου MOF τονίζουμε οτι το MOF αποτελεί ένα χρήσιμο περιβάλλον για την ανάπτυξη μοντέλων καθώς καθώς προσφέρει ευελιξία στον προγραμματιστη επιτρέποντας του να ορίζει μοντέλα τα οπόια στην συνέχεια μπορεί να τα εξάγει απο μια εφαρμογή να τα εισάγει σε άλλη, να τα μεταφέρει μέσω δικτύου να τα αποθηκεύει και να τα ανακτά αργότερα καθώς και να τα μετασχηματίζει σε άλλες μορφές όπως το XML με την βοήθεια του XMI.

2.2.2 XMI

Το **XML Metadata Interchange** (XMI) είναι ένα πρότυπο της OMG που έχει πιστοποιηθεί κατά ISO/IEC 19503:2005 Information technology -- XML Metadata Interchange (XMI) για ανταλλαγή μέτα-δεδομένων μέσω της Extensible Markup Language (XML). Μπορεί να χρησιμοποιηθεί για κάθε μέτα-δεδομένο του οποίου το μεταμοντέλο μπορεί να αναπαρασταθεί στο Meta Object Facility (MOF). Η πιο κοινότυπη χρήση της XMI είναι σαν σχήμα ανταλλαγής για UML μοντέλα. Επίσης μπορεί να χρησιμοποιηθεί για την σειριοποίηση μοντέλων άλλων γλωσσών (μέτα-μοντέλα).

Σύμφωνα με το OMG, τα δεδομένα χωρίζονται σε αφηρημένα μοντέλα (abstract models) και σε συγκεκριμένα μοντέλα (concrete models). Τα αφηρημένα μοντέλα χρησιμοποιούνται για να περιγράψουν την σημασιολογία της πληροφορίας, ενώ τα συγκεκριμένα μοντέλα αναπαριστούν εποπτικά διαγράμματα. Αφηρημένα μοντέλα είναι στιγμιότυπα αυθαίρετων γλωσσών μοντελισμού που βασίζονται στο MOF όπως είναι η UML ή η SysML. Για τα διαγράμματα χρησιμοποιείται η τυποποίηση της ανταλλαξιμότητας διαγραμμάτων (Diagram Interchange) [XMI[DI]]. Μέχρι σήμερα υπάρχουν πολλές ασυμβατότητες μεταξύ διάφορων εργαλείων μοντελοποίησης που επιτρέπουν την εξαγωγή διαγραμμάτων σε XMI, ακόμη και μεταξύ ανταλλαγών αφηρημένων δεδομένων. Δυστυχώς, αυτό σημαίνει ότι η ανταλλαγή αρχείων μεταξύ εργαλείων χρησιμοποιώντας το XMI, σπάνια είναι δυνατή.

Ο οργανισμός OMG επιδιώκει, με τον ορισμό της XML Metadata Interchange (XMI), να υλοποιήσει την εύκολη ανταλλαγή μεταδεδομένων μεταξύ εργαλείων μοντελοποίησης που βασίζονται στην γλώσσα μοντελοποίησης UML και αποθηκών μεταδεδομένων που βασίζονται στο MOF σε κατανεμημένα ετερογενή περιβάλλοντα. Η XMI χρησιμοποιείται επίσης ως ο μεσολαβητής με τον οποίο μοντέλα διέρχονται από εργαλεία μοντελοποίησης σε εργαλεία παραγωγής λογισμικού ως μέρος της μοντελοκεντρικής μηχανικής.

2.2.3 Eclipse Modeling Framework

Το Eclipse Modeling Framework (EMF)[4] είναι ένα έργο ανοικτού κώδικα που χρηματοδοτείται και βρίσκεται υπό την επίβλεψη της IBM. Αποτελεί ίσως την καλύτερη προσέγγιση στον χώρο της τεχνολογίας λογισμικού με την οποία δημιουργούνται συστήματα λογισμικού που υπακούν στον ορισμό της Μοντελοκεντρικής Αρχιτεκτονικής.

Το EMF δίνει στον σχεδιαστή την δυνατότητα να σχεδιάσει μοντέλα που περιγράφουν ένα σύστημα λογισμικού. Δεδομένου ότι ο πηγαίος κώδικας είναι ένα μοντέλο περιγραφής του συστήματος, τα UML διαγράμματα είναι επίσης μοντέλα περιγραφής του συστήματος όπως και τα XML σχήματα, το EMF προσπαθεί να ενοποιήσει τις υπάρχουσες τεχνικές μοντελοποίησης και να δώσει στον μηχανικό λογισμικού την δυνατότητα να ορίσει το σύστημα στο μοντέλο με το οποίο αυτός είναι πιο εξοικειωμένος αλλά ταυτόχρονα να έχει και την περιγραφή των άλλων μοντέλων.

Το EMF, στηρίζεται σε δύο ευρέως διαδεδομένα πρότυπα του Object Management Group, MOF και το XMI τα οποία περιγράψαμε παραπάνω. Μέσω του μοντέλου MOF, με την αρωγή του EMF, μπορεί να περιγράψει κανείς ένα δομημένο και ιεραρχικό μοντέλο δεδομένων που σκοπό έχει να αποτελέσει την προδιαγραφή των δεδομένων που χρησιμοποιεί το σύστημα λογισμικού ή ακόμα και την προδιαγραφή του ίδιου του συστήματος λογισμικού. Έπειτα το EMF μπορεί να δημιουργήσει αυτόματα κώδικα, ο οποίος μπορεί να δημιουργήσει στιγμιότυπα των μοντέλων που ορίστηκαν σε MOF είτε με την βοήθεια του ίδιου του EMF είτε με κάποιο άλλο εργαλείο και εισήχθησαν στο EMF με την μορφή XMI.

Το EMF χρησιμοποιεί κατά κόρον το πρότυπο XMI ως μέσο ανταλλαγής MOF μοντέλων μεταξύ εφαρμογών και ως μέσο αρχικοποίησης και αποθήκευσης MOF μοντέλων στον κώδικα της εφαρμογής. Το EMF προσφέρει κλάσεις εκτενών λειτουργιών στο EMF API, την διαπροσωπία προγραμματισμού που προσφέρει το EMF στους χρήστες, για καλύτερη χρήση των προτύπων MOF και XMI.

Στα πλαίσια αυτής της διπλωματικής, το EMF χρησιμοποιήθηκε για την μοντελοποίηση του τροποποιημένου Μοντέλου Δένδρων Στόχων, που χρειάστηκε να ορισθεί για τις ανάγκες του περιβάλλοντος-πλαισίου. Αρχικά δημιουργήθηκε με βάση το MOF, το Μοντέλο του Δένδρου Στόχων, και στην συνέχεια χρησιμοποιήθηκε ο παραγόμενος κώδικα χειρισμού του μοντέλου δέντρου στόχων καθώς και το EMF API, ώστε να οριστούν τα δέντρα στόχων και να υλοποιηθούν οι αλγόριθμοι και δημιουργίας και διαχείρησης των δέντρων στόχων, για τις ανάγκες του περιβάλλοντος πλαισίου.

2.3 Δέντρα Στόχων (Goal Trees)

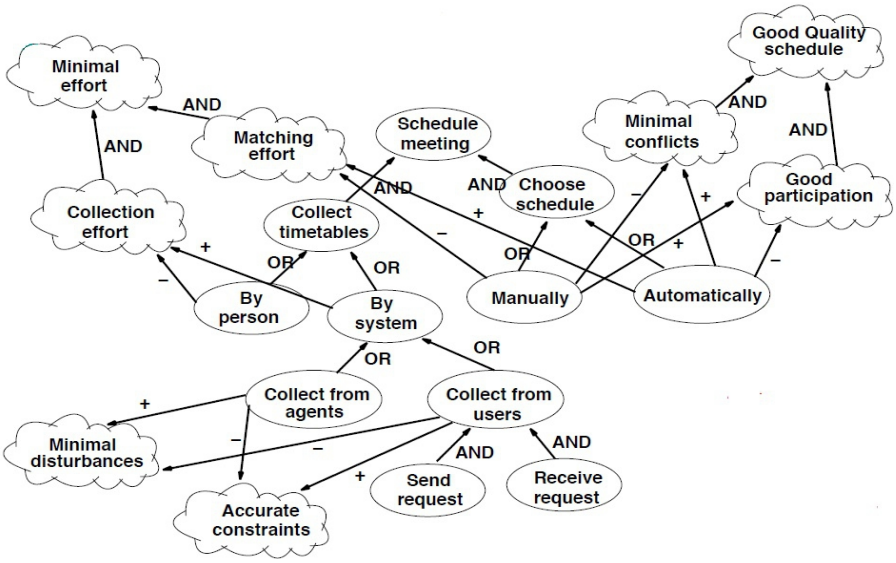
Ένας σημαντικός κλάδος της Τεχνολογίας Λογισμικού είναι η Μηχανική Απαιτήσεων. Ο κλάδος αυτός ασχολείται με την συλλογή και την ανάλυση απαιτήσεων. Η συλλογή και ανάλυση απαιτήσεων αποτελεί ένα σημαντικό στάδιο στην διαδικασία ανάπτυξης λογισμικού. Οι μηχανικοί λογισμικού πρέπει να προσδιορίσουν τις απαιτήσεις του συστήματος που φτιάχνουν έτσι ώστε να εξασφαλίσουν την καλύτερη δυνατή ποιότητα του. Σύμφωνα με το [5] ανεπαρκείς, ατελείς, διφορούμενες ή ασυνεπείς απαιτήσεις λογισμικού έχουν πολύ μεγάλη επίδραση στην ποιότητα του λογισμικού που παράγεται.

Μια προσέγγιση στην Μηχανική Απαιτήσεων είναι η προσανατολισμένη σε στόχους Μηχανική Απαιτήσεων. Στα [5] και [6] εξηγούνται τα πλεονεκτήματα αυτής της προσέγγισης. Ενδεικτικό της δημοφιλίας της την προσανατολισμένης σε στόχους Μηχανικής Απαιτήσεων αποτελεί η πληθώρα μεθοδολογιών που έχει προταθεί για την προσανατολισμένη σε στόχους Μηχανική Απαιτήσεων π.χ. τα KAOS [7], GRL [8], GBRAM [9], TROPOS [10], CREWS [11], i* [12] κτλ.

Ένα σηματικό μέρος της Μηχανικής Απαιτήσεων αποτελεί η μοντελοποίηση των απαιτήσεων. Για την μοντελοποίηση των λειτουργικών και μη απαιτήσεων ενός συστήματος λογισμικού έχει προταθεί πληθώρα μέθοδων, ενώ έχουν αναπτυχθεί και αρκετά εργαλεία που στηρίζονται στις μεθόδους αυτές. Γμρωτοεισήχθησαν στον χώρο της μηχανικής και μοντελοποίησης λογισμικού στο [13] και χρησιμοποιούνται για την μοντελοποίηση των στόχων στο TROPOS που αναφέρεται παραπάνω.

Τα Δέντρα Στόχων μοντελοποιούν τους στόχους σε μια δενδρική δομή αποδομώντας κάθε στόχο σε υποστόχους η επαλήθευση των οποίων οδηγεί στην επαλήθευση του αρχικού. Σύμφωνα με αυτό το μοντέλο ένας στόχος αναλύεται σε υποστόχους οι οποίοι αναπαρίστανται σαν παιδιά του πρώτου στην δενδρική δομή. Το ίδιο ισχύει και για κάθε έναν απο τους στόχους-παιδιά. Η αποσυθέση ενός στόχου μπορεί να είναι διαφόρων τύπων και ο κάθε τύπος υποδηλώνει τον τρόπο επαλήθευσης του στόχου-πατέρα από τους στόχους-παιδιά. Για τις ανάγκες της διπλωματικής περιοριζόμαστε μόνο σε αποσυνθέσεις τύπου ΚΑΙ ή Ή. Ο τύπος ΚΑΙ όπως υποδηλώνει και το όνομά του, συνδέει τους στόχους παιδιά με την λογίκή σύζευξη. Ώς εκ τούτου ο στόχος-πατέρας επαληθεύεται αν και μόνο αν απαληθεύονται όλοι οι στόχοι-παιδιά. Αντίστοιχα η αποσύνθεση Ή ενός στόχου συνδέει τους στόχους-παιδιά με την λογική διάζευξη. Δηλαδή επαληθεύει τον στόχο-πατέρα αν τουλάχιστον ένας από τους στόχους-παιδιά επαληθεύεται.

Τέτοιου είδους στόχοι αναφέρονται στην βιβλιογραφία ως **hard goals**. Οι στόχοι αυτοί ορίζουν ένα σύνολο υποστόχων, με βάση το οποίο υπάρχει μονοσήμαντη σημασιολογία επαλήθευσης του στόχου-πατέρα και του συνόλου στόχων-παιδιών. Υπάρχουν όμως και περιπτώσεις απαιτήσεων, στην μοντελοποίηση απαιτήσεων λογισμικού, όπου ένας στόχος μπορεί να μην επαληθεύεται μονοσήμαντα από ένα σύνολο υποστόχων ένα είδος θετικής ή αρνητικής συνεισφοράς στον στόχο αυτό από άλλους στόχους. Ανάλογα με το ποσοστό της θετική έναντι της αρνητικής συνεισφοράς, μπορεί να επαληθεύεται ή όχι ο στόχος. Τέτοιου είδους στόχοι αναφέρονται ώς **soft goals**. Παρακάτω φαίνεται ένα παράδειγμα ενός δέντρου στόχων όπου συνυπάρχουν τόσο hard goals και soft goals (σχ. 2.2).



Σχήμα 2.3.1: Παράδειγμα δέντρου στόχων με soft goals

Στο παραπάνω διάγραμμα τα hard goals παρουσιάζονται με τα οβάλ ενώ τα soft goals με τα σύννεφα. Φαινεται ξεκάθαρα η δενδρική δομή και οι αποσυνθέσεις των κόμβον καθώς επίσης και οι συνεισφορές των στόχων σε κάθε soft goal. Οι θετικές συνεισφορές παρουσιάζονται με το σύμβολο + (συν) ενώ οι αρνητικές με το σύμβολο - (μειον).

Τα ΚΑΙ/Η Δέντρα Στόχων προσφέρουν έναν εύκολο τρόπο αναπαράστασης των στόχων που ήταν πολύ χρήσιμος για την μοντελοποίηση δεδομένων που χρησιμοποιεί το περιβάλλον πλαίσιο που προτείνουμε σε αυτήν την διπλωματική. Μπορούν εύκολα να κατασκευαστούν και να διασχιστούν προγραμματιστικά. Επιπροσθέτως η άμεση σύνδεση τους με την άλγεβρα Boole και την λογική πρώτης τάξης, καθιστά τα δέντρα στόχων μια ελκυστική λύση καθώς μετατρέπονται εύκολα σε ένα σύνολο κανόνων λογικής πρώτης τάξης. Αυτοί οι κανόνες στην συνέχεια μπορούν να χρησιμοποιηθούν από έτοιμα εργαλεία που υλοποιούν αλγορίθμους λογικής πρώτης τάξης. Έτσι τα δέντρα στόχων έρχοναι με μια πληθώρα εργαλείων και αλγορίθμων που εφαρμόζονται στην αλγεβρα Boole και μας είναι άμεσα διαθέσιμα. Έπιπλέον τα δέντρα στόχων είναι εύκολα επεκτάσιμα καθώς επιτρέπουν την επισύναψη οποιασδήποτε πληροφορίας πάνω στους κόμβους-στόχους με την μορφή επισημειώσεων ή ακόμα και την επέκταση του μοντέλου δέντρων στόχων με επιπλέον σχέσεις μεταξύ των κόμβων στόχων. Στα [14], [15], [16] μπορούν να βρεθούν επεκτάσεις του μοντέλου στόχων που αφορούν αφορούν πράκτορες (Agents), ρόλους (Roles) και δεσμεύσεις (Commitments) μεταξύ των πρακτόρων, και συνεισφορές (Contirbution) μεταξύ κόμβων. Στα πλαίσια της διπλωματικής επεκτείναμε το μοντέλο στόχων ώστε να μπορεί να περιγράψει τις συνεισφορές μεταξύ των κόμβων.

Με τις συνεισφορές προσπαθούμε να μοντελοποιήσουμε περίπλοκες σχέσεις μεταξύ στόχων. Μια συνεισφορά μεταξύ δύο στόχων ορίζεται ως η δυνατότητα ενός στόχου να συνεισφέρει θετικά ή αρνητικά στην επιτυχία ή αποτυχία ενός άλλου στόχου. Έτσι, σύμφωνα και με το [15], υπάρχουν τέσσερα είδη συνεισφορών που συμβολίζονται με *++S (g, g′), --S (g, g′), ++D (g,*

g′) και *--D (g, g′)*.

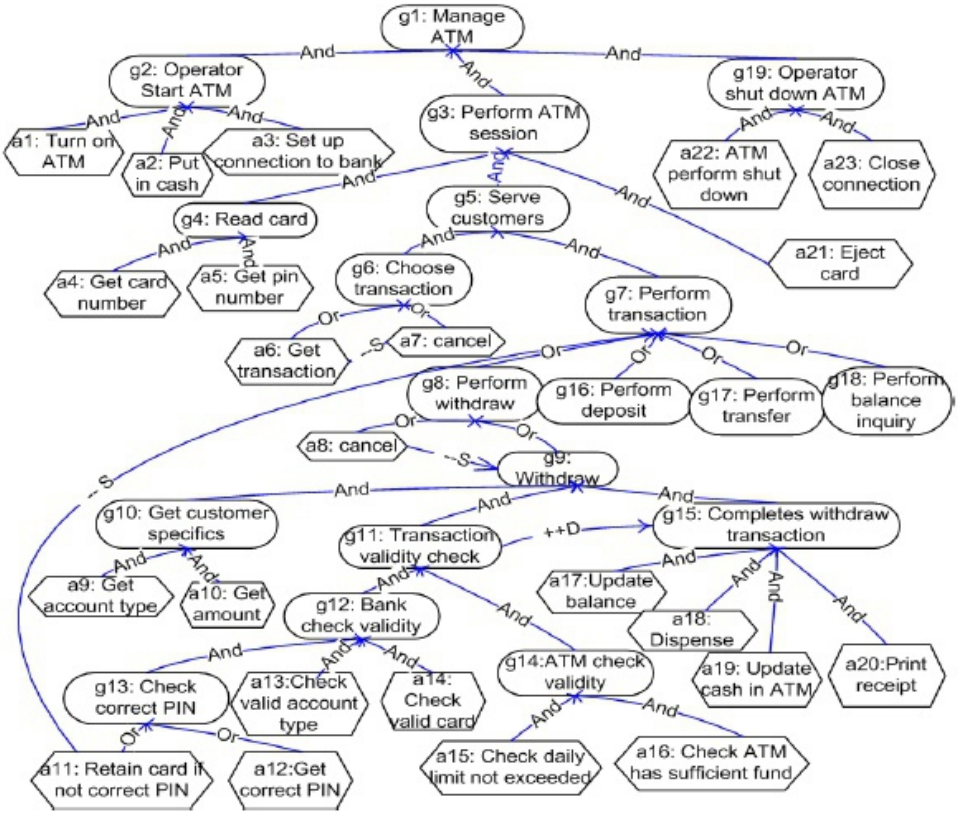
Η συνεισφορά *++S (g, g′)* εκφράζει ότι η επιτυχία του στόχου *g* συνεπάγεται την επιτυχία του στόχου *g′*.

Η συνεισφορά *--S (g, g′)* εκφράζει ότι η επιτυχία του στόχου *g* συνεπάγεται την αποτυχία του στόχου *g′*.

Η συνεισφορά *++ D (g, g′)* εκφράζει ότι η αποτυχία του στόχου *g* συνεπάγεται την επιτυχία του στόχου *g′*.

Και τέλος η συνεισφορά *-- D (g, g′)* εκφράζει ότι η αποτυχία του στόχου *g* συνεπάγεται την αποτυχία του στόχου *g′*.

Τέλος δίνουμε ένα διάγραμμα που βρίσκεται στο [17] και στο οποίο φαίνονται όλα τα είδη συνεισφορών μεταξύ των κόμβων στόχων που περιγράψαμε.



Σχήμα 2.3.2: Παράδειγμα δέντρου στόχων με soft goals και contributions

Για τις ανάγκες της διπλωματικής χρησιμοποιήσαμε τα δέντρα στόχων για να μοντελοποιήσουμε υποθέσεις για το υπό επιτήρηση σύστημα και να ορίσουμε τις πληροφορίες που πρέπει να συλλεγούν ώστε να αποδειχθεί μια υπόθεση. Αυτές οι επιπλέον πληροφορίες μοντελοποιήθηκαν ως επισημειώσεις πάνω στους κόμβους του δέντρου. Έτσι δεδομένης μιας υπόθεσης προς απόδειξη τα δέντρα στόχων που ορίσαμε μας δίνουν όλους τους δυνατούς τρόπους απόδειξης της υπόθεσης αυτής. Διαφορετικά μονοπάτια του δέντρου στόχων ορίζουν διαφορέτικούς τρόπους απόδειξης της υπόθεσης που σχετίζεται με το δέντρο καθώς επίσης και διαφορετικό σύνολο πληροφοριών προς συλλογή. Από αυτούς τους τρόπους επιλέγεται ο πιο συμφέρον σύμφωνα με κάποιο κριτήριο και έτσι γίνεται η προσαρμογή του επιπέδου

επιβλεψης του συστήματος. Περισσότερα λεπτομέρειες για την συνολική διαδικασία δίνονται στο κεφάλαιο 3 και λεπτομέρειες του εννοιολογικού μοντέλου των δέντρων στόχων που χρησιμοποιήσαμε στο κεφάλαιο 4.

2.4 Περιβάλλοντα Συλλογιστικής (Reasoning Frameworks)

Όπως αναφέρθηκε και στην προηγούμενη ενότητα ένας από τους βασικούς λόγους της χρήσης δένδρων στόχων για την μοντελοποίηση των υποθέσεων και των πληροφοριών προς συλλογη για την απόδειξη των υποθέσεων αυτών είναι το γεγονός ότι τα δέντρα στόχων μετασχηματίζονται εύκολα σε κανόνες λογικής πρώτης τάξης και έτσι επιτρέπουν την χρήση έτοιμων εργαλείων που υλοποιούν αλγορίθμους λογικής πρώτης τάξης. Σε αυτήν την ενότητα περιγράφουμε τα περιβάλλοντα συλλογιστικής που μελετήσαμε. Ο όρος περιβάλλοντα συλλογιστικής αναφέρεται σε όλες εκείνες τις θεωρίες, τις μεθοδολογίες και τα εργαλεία που παρέχουν δυνατότητες συμπερασμού βασισμένες σε λογικές επαγωγές. Στα πλαίσια αυτής της διπλωματικής μελετήσαμε τα λογικά δίκτυα Markov καθώς επίσης και το πρόβλημα της ικανοποιησιμότητας (Boolean satisfiability problem). Σε αυτό το σημείο ακολουθεί μια σύντομη ανασκόπηση της θεωρίας των μαρκοβιανών δικτύων και της λογικής πρώτης τάξης, που αποτελεί βάση της θεωρίας των μαρκοβιανών δικτύων. Τελος ακολουθεί μια ανασκόπηση του προβλήματος της ικανοποιησιμότητας, έτσι ώστε να γίνει ξεκάθαρη η σύνδεση του με προβλήματα που ανακύπτουν στα πλαίσια της διπλωματικής.

2.4.1 Λογική Πρώτης Τάξης

Η λογική πρώτης τάξης είναι ένας φορμαλιστικός τρόπος αναπαράστασης της γνώσης και εκτέλεσης συμπερασμων, ο οποίος προσεγγίζει τον ανθρώπινο τρόπο σκέψης. Χρησιμοποιείται κατά κόρον στα μαθηματικά, την φιλοσοφία, την γλωσσολογία και την επιστήμη των υπολογιστών[18]. Η λογική πρώτης τάξης χτίζει πάνω στην προτασιακή λογική είσαγοντας όρους (terms), κατηγορήματα (predicates) και ποσοδείκτες (quantifiers).

Μιά βάση γνώσης λογικής πρώτης τάξης είναι ένα σύνολο λογικων κανόνων πρώτης τάξης, οι οποιοι αποτελούνται από σταθερές, μεταβλητές, συναρτήσεις και κατηγορήματα. Οι σταθερές αντιπροσωπεύουν συγκεκριμένα αντικείμενα του κόσμου που περγράφουμε ενώ οι μεταβλητές αναπαριστούν οποιοδήποτε αντικείμενο του κόσμου. Οι συναρτήσεις από την άλλη αντιπροσωπεύουν αντιστοιχίσεις ενός πλήθους αντικειμένων του κόσμου σε άλλα αντικείμενα. Το πλήθος των ορισμάτων της συνάρτησης καθορίζει την τάξη της. Τέλος τα κατηγορήματα αντιπροσωπεύουν σχέσεις ανάμεσα στα αντικείμενα του κόσμου ή χαρακτηριστικά των αντικειμένων. Πέραν των παραπάνω, στην λογική πρώτης τάξης ορίζονται και τα ακόλουθα*:*

- *Ερμηνεία*: Είναι η συμβολική αναπαράσταση όλων των παραπάνω στοιχείων της λογικής πρώτης τάξης.

- *Όρος*: Είναι μία έκφραση λογικής πρώτης τάξης που αντιπροσωπεύει κάποιο αντικείμενο του κόσμου. Έτσι ένας όρος μπορεί να είναι μια σταθερά, μία μεταβλητή ή μία συνάρτηση.

- *Ατομικός τύπος/Άτομο:* είναι ένα κατηγόρημα σε κ σε πλήθος όρους.

- Ένας τύπος είναι είναι ένας σύνθετος τύπος που δημιουργείται αναδρομικά από ατομικούς τύπους, λογικούς συνδέσμους και ποσοδείκτες.

- *Βασικός όρος:* είναι ένας όρος που δεν περιέχει μεταβλητές.Ένα βασικό άτομο ή βασικό κατηγόρημα είναι ένας ατομικός τύπος που αποτελείται μόνο από βασικούς όρους.

Αφού δώσαμε μια σύντομη επισκόπηση της βασικής ορολογίας της λογικής πρώτης τάξης συνεχίζουμε με την ανασκόπηση των μαρκοβιανών λογικών δικτύων.

2.4.2 Markov Logic Networks

Η θεωρία των Μαρκοβιανών Λογικών Δικτύων (MLN) εισήχθη από τους Richardson and Domingos [19],[20] ως ένας τρόπος συνδυασμού των πλεονεκτημάτων της λογικής πρώτης τάξης και των πιθανοτικών γραφικών μοντέλων. Ένα MLN συνίσταται απο μια βάση γνώσης (KB) κατηγορημάτων και θεμελιωδών ατόμων. Η θεωρία των Μαρκοβιανών Λογικών Δικτύων συνδυάζει τα πλεονεκτήματα των Μαρκοβιανών Δικτύων με τα πλεονεκτήματα της λογικής πρώτης τάξης. Πιο συγκεκριμένα, ένα MLN είναι μια βάση γνώσης λογικής πρώτης τάξης στη οποία εφαρμόζεται πιθανοτικός συμπερασμός, σε αντίθεση με την αυστηρό συμπερασμό της λογικής πρώτης τάξης. Στην λογική πρώτης τάξης η βάση γκώσης αποτελείται από ένα σύνολο τύπων που αποτελούνται από λογικούς συνδέσμους, ποσοδείκτες και κατηγορήματα. Αναθέτοντας αληθοτιμές σε κάθε πιθανό κατηγόρημα, κατασκευάζουμε έναν πιθανό κόσμο στον οποίο η βάση γνώσης μπορεί να είναι αληθής ή ψευδής. Με άλλα λόγια, αν ο κόσμος παραβιάζει έστω και έναν λογικό κανόνα της βάσης γνώσης τότε έχει η πιθανότητα αυτός ο κόσμος να είναι μια ερμηνεία της βάσης γνώσης είναι μηδενική. Αντιθέτως, στα Μαρκοβιανά Λογικά Δίκτυα ο κόσμος μπορεί να είναι μια ερμηνεία της βάσης γνώσης ακόμα και αν παραβιάζονται καποιο κανόνες της βάσης. Το μέγεθος της πιθανότητας με την οποία ο κόσμος αποτελεί ερμηνεία της βάσης γνώσης, εξαρτάται αντίστροφα από τον αριθμό των κανόνων που παραβιάζονται, καθώς επίσης και από τους περιορισμους που εισάγονται από τους κανόνες που παραβιάζονται[21].

Όπως συζητήθηκε στο [19], ένα Μαρκοβιανό Δίκτυο είναι ένα μοντέλο της από κοινού κατανομής ενός συνόλου μεταβλητών X = (X1, X2, ...,Xn). Το μοντέλο αποτελείται απο έναν μή κατευθυνόμενο γράφο G και μία σειρά απο συναρτήσεις δυναμικού φ_k . Κάθε κόμβος του γράφου αντιστοιχεί σε μία μεταβλητή του συνόλου X και υπάρχει μία με συνάρτηση φ_k για κάθε κλίκα του γράφου. Οι συναρτήσεις δυναμικού έχουν μη αρνητικό πραγματικό πεδίο τιμών .και κάθε μία από αυτές αναπαριστά την κατάσταση της αντίστοιχης κλίκας. Λαμβάνοντας υπόψην τα παραπάνω, η από κοινού κατανομή πιθανότητας δίνεται από την ακόλουθη σχέση:

$$P(x)=\frac{1}{Z}\prod \varphi_k(x_k) \qquad (2.4.1)$$

όπου x_k είναι η κατάσταση των μεταβλητών που εμφανίζονται στην κλίκα k και Z είναι η συνάρτηση διαμέρισης που δίνεται από την εξής σχέση:

$$Z=\sum_x \prod_k \varphi_k(x_k) \qquad (2.4.2)$$

Τα Μαρκοβιανά Δίκτυα συχνά αναπαριστώνται ως λογαριθμογραμμικά μοντέλα (log - linear models). Σε αυτήν την περίπτωση οι συναρτήσεις δυναμικού αντικαθίστανται από εκθετικά αθροίσματα βαρών χαρακτηριστικών της κατάστασης του δικτύου. Ένα χαρακτηριστικό είναι μία συνάρτηση με πραγματικό πεδίο τιμών. Ως εκ τούτου, η από κοινού κατανομή πιθανότητας παίρνει την ακόλουθη μορφή :

$$P(X=x)=\frac{1}{Z}\exp(\sum_i w_i f_i(x)) \quad (2.4.3)$$

όπου με w_i αναπαριστώνται τα βάρη και με f_i τα χαρακτηριστικά για τα οποία θεωρούμε πως $f_i(x)\in\{0,1\}$.

Τα Μαρκοβιανά Λογικά δίκτυα (MLN) συνδυάζουν την ευκολία αναπαράστασης που παρέχει η λογική πρώτης τάξης με την ελαστικότητα των πιθανοτικών μεθόδων των μαρκοβιανών δικτύων. Πιο συγκεκριμένα, η βάση γνώσης της λογικής πρώτης τάξης μπορεί να ειδωθεί ως ένα σύνολο αυστηρών κανόνων (hard constraints) οι οποίοι ορίζουν έναν πιθανό κόσμο. Ως εκ τούτου, αν ο κόσμος δεν ικανοποιεί έστω και έναν από τους κανόνες τότε δεν υπάρχει πιθανότητα ο κόσμος να είναι έγκυρος. Τα μαρκοβιανά λογικά δίκτυα χαλαρώνουν την αυστηρότητα αυτών των κανόνων, χρησιμοποιώντας πιθανοτικές μεθόδους. Έτσι σε περίπτωση που παραβιαστούν κάποιοι κανόνες της βάσης γνώσης της λογικής πρώτης τάξης ο κόσμος έχει ακόμα πιθανότητα να είναι έγκυρος. Όσο περισσότεροι κανόνες παραβιάζονται τόσο μικρότερη είναι η πιθανότητα ο κόσμος αυτός να υπάρχει. Κάθε τύπος της βάσης γνώσης λογικής πρώτης τάξης έχει ένα βάρος που αντιπροσωπεύει το πόσο σημαντικός είναι ο κανόνας για το μαρκοβιανό δίκτυο ή με άλλα λόγια δείχνει πόσο επιρρεάζει η ισχύς ή όχι του κανόνα την πιθανότητα ύπαρξης ενός κόσμου.

Σε αυτό το σημείο μπορούμε να δώσουμε έναν πιο τυπικό όρισμό των Μαρκοβιανών Λογικών Δικτύων. Ένα Μαρκοβιανό Λογικό Δίκτυο είναι ένα σύηολο L από ζεύγη (F_i, w_i), όπου *Fi* είναι ένας τύπος λογικής πρώτης τάξης και *wi* ένας πραγματικός αριθμός που αντιπροσωπεύει το βάρος του συγκεκριμένου τύπου. Σε συνδυασμό με ένα πεπερασμένο σύνολο σταθερών $C=\{c_1, c_2, ..., c_{|c|}\}$, ορίζουν ένα Μαρκοβιανό Δίκτυο $M_{L,C}$ ως ακολούθως:

1. Το $M_{L,C}$ περιέχει έναν δυαδικό κόμβο για κάθε πιθανή θεμελίωση (grounding) κάθε κατηγορήματος που εμφανίζεται στο L. Η τιμή του κόμβου είναι 1 εάν το βασικό άτομο είναι αληθές, και μηδέν αν είναι ψευδές.

2. Το $M_{L,C}$ περιέχει ένα χαρακτηριστικό για κάθε πιθανή θεμελίωση κάθε τύπου F_i in *L*. Η τιμή του χαρακτηριστικού αυτού είναι 1 εάν ο ατομικός τύπος είναι αληθής, και 0 αν είναι ψευδής. Το βάρος κάθε χαρακτηριστικού είναι το w_i και σχετίζεται με το F_i στο *L*.

Ένα Μαρκοβιανό Λογικό Δίκτυο μπορεί να παράξει διαφορετικά δίκτυα για διαφορετικά σύνολα σταθερών. Τα δίκτυα αυτά ονομάζονται βασικά Μαρκοβιανά Δικτυα (ground Markov networks), και ενώ μπορεί να διαφέρουν πολύ σε έκταση, παρουσιάζουν ομοιότητες στην δομή και στις παραμέτρους τους. Ως εκ τούτου, η κατανομή πιθανότητας όλων των *x* που ορίζονται από το βασικό μαρκοβιανό δίκτυο παίρνει την ακόλουθη μορφή:

$$P(X=x)=\frac{1}{Z}\exp(\Sigma_i w_i n_i(x))=\frac{1}{Z}\prod_i \varphi_i(x_i)^{n_i(x_i)} \quad ,$$

Όπου $n_i(x)$ είναι ο αριθμός των αληθών θεμελιώσεων του F_i στο x, x_i είναι η κατάσταση των ατόμων που εμφανίζονται στο F_i , και $\varphi_i(x_i)=e^{w_i}$. Η θεωρία των Μαρκοβιανών Λογικών Δικτύων περιλαμβάνει δύο βασικές αλγοριθμικές διαδικασίες που μπορούν να εκτελεστούν πάνω σε ένα δίκτυο. Αυτές είναι οι εξής:

- **Μαθηση (Learning)** : Μέσω αυτής της διαδικασίας, μπορούμε να εξάγουμε τα βάρη

του Δικτύου από μία ή περισσότερες γνωσιακές βάσεις κάνοντας μια θεώρηση κλειστού κόσμου, δηλαδή για οποιοδήποτε βασικό άτομο δεν υπάρχει στην βάση γνώσης θεωρείται ψευδές.

- **Συμπερασμός (Inference)** : Μέσω της διαδικασίας αυτής εκτιμάτε η πιθανότητα να ισχύει ή όχι ένας ατομικός τύπος F_1 δεδομένης της ισχύος κάποιου άλλου τύπου F_2 .

Από σκοπιά πρακτικής σκοπιάς, ένα χρήσιμο εργαλείο που υλοποιεί αλγορίθμους στατιστικής μάθησης και συμπερασμού βασιζόμενο στην θεωρία των Μαρκοβιανών Λογικών Δικτύων είναι το εργαλείο Alchemy [22].

2.4.3 Δυαδική Ικανοποιησιμότητα (Sat Problem)

Σε αυτήν την ενότητα δίνουμε μια επισκόπηση του προβλήματος της ικανοποιησιμότητας (satisfiability problem (SAT)) και κάποιων βασικών αλγορίθμων επίλυσης του. Τέλος τονίζουμε την σημασία του προβλήματος δίνοντας παραδείγματα εφαρμογής του για την επίλυση άλλων προβλημάτων.

Το πρόβλημα της ικανοποιησιμότητας είναι ένα βασικό πρόβλημα της μαθηματικής λογικής και της θεωρίας υπολογισμού. Το πρόβλημα της ικανοποιησιμότητας είναι το προβλημα της απόφασης για το αν μια έκφραση δυαδικής λογικής μπορεί να ικανοποιηθεί η όχι, δηλαδή αν μπορεί να υπάρξει ανάθεση αληθοτιμών για κάθε μεταβλητή της έκφρασης τέτοια ώστε να κάνει την έκφραση αληθή. Το πρόβλημα της ικανοποιησιμότητας είναι το πρώτο πρόβλημα που αποδείχθηκε ότι ανήκει στην κλάση των NP-complete προβλημάτων. Πριν την απόδειξη του είχε εισαχθεί ο όρος NP-complete αλλά δεν υπήρχε απόδειξη ότι τέτοια προβλήματα όντως υπάρχουν.

Το πρόβλημα της ικανοποιησιμότητας είναι θεμελιώδες για την επίλυση προβλημάτων πολλών πρακτικών προβλημάτων. Οι πρακτικές αυτές εφαρμογές του προβλήματος της ικανοποιησιμότητας, είναι η κινητήριος δύναμη πίσω από την έρευνα για την βελτίωση των αλγορίθμων επίλυσης του. Πιο συγκεκριμένα το πρόβλημα της ικανοποιησιμότητας έχει άμεση εφαρμογή σε προβλήματα τεχνητής νοημοσύνης όπως το προβλήμα των N-Βασιλισσών, στην μηχανική όραση (ταίριασμα εικόνων), σε συστήματα βάσεων δεδομένων (βελτιστοποίηση ερωτημάτων, έλεγχος συγχρονικότητας), στην αυτοματοποίηση σχεδιασμού ολοκληρωμένων κυκλωμάτων (μοντελοποίηση κυκλωμάτων, λογική ελαχιστοποίηση, ελαχιστοποίηση καταστάσεων, έλεγχο και επαλήθευση, κ.α.), στην επαλήθευση υλικού και λογισμικού, στην βιοπληροφορική κ.α. Μια πληρέστερη λίστα και λεπτορμερέστερη ανάλυση των εφαρμογών του προβλήματος της ικανοποιησιμότητας μπορεί να βρεθεί στα [Algorithms-for-satisfiability] και [Practical Applications of Boolean Satisfiability].

Οσόν αφορά τους αλγορίθμους επίλυσης ο πρώτος και πιο διαδεδομένος αλγόριθμος επίλυσης του προβλήματος της ικανοποιησιμότητας είναι ο DPLL (Davis-Putnam-Logemann-Loveland) [23], ο οποίος χρησιμοποιεί αναζήτηση με οπισθοδρόμηση (backtracking search). Ένα βασικό χαρακτηριστικό του αλγορίθμου DPLL είναι το αποδοτικό κλάδεμα (pruning) του χώρου αναζήτησης ;όταν μια ανάθεση αποδεικνύεται λανθασμένη. Παρά το γεγονός ότι το πρόβλημα της ικανοποιησιμότητας είναι NP-complete και όλοι οι αλγόριθμοι απαιτούν στην χειρότερη περίπτωση εκθετικό χρόνο ως προς το μέγεθος του προβλήματος οι σύγχρονοι αλγόριθμοι επίλυσης του προβλήματος της ικανοποιησιμότητας είναι εξαιρετικά αποδοτικοί και ικανοί να λύσουν μεγάλα πολύπλοκα προβλήματα πραγματικών εφαρμογών. Μερικοί απο τους πιο αποδοτικούς αλγόριθμους είναι οι Chaff [24], BerkMin [25] και Siege [26].

Μια κατηγοριοποίηση που μπορεί να δώθεί στους αλγορίθμους επίλυσης του προβλήματος της ικανοποιησιμότητας βασίζεται στο κατά πόσο εγγυώνται την εύρεση λύσης ή όχι. Έτσι οι αλγόριμοι αυτοί μπορούν να κατηγοριοποιηθούν σε πλήρεις και μη. Ένας αλγόριθμος επίλυσης του προβλήματος της ικανοποιησιμότητας λέγεται πλήρης αν δεδομένης μιας λογικής έκφρασης εγγυάται την εύρεση μιας ανάθεση αληθοτιμών που ικανοποιούν την έκφραση σε περίπτωση που υπάρχει τετοια. Σε αντίθετη περίπτωση δηλώνει ότι η έκφραση δεν μπορεί να ικανοποιηθεί. Αντίστοιχα οι ατελείς αλγόριθμοι δεν εγγυώνται την εύρεση κάποιας ανάθεσης ακόμα κιαν αυτή υπάρχει. Αυτοί οι αλγόριθμοι συνήθως τρέχουν για καθορισμένο χρονικό διάστημα στο οποίο υπάρχει πιθανότητα να μην βρουν λύση. Αντίθετα με τους πλήρης αλγόριθμους, οι αλγόριθμοι αυτοί βασίζονται συνήθως σε στοχαστική τοπική αναζήτηση. Οι αλγόριθμοι αυτοί είναι ταχύτεροι απο τους πλήρεις και αποτελούν λύση σε περιπτώσεις που τα δεδομένα εισόδουν είναι τόσο μεγάλα που οι πλήρεις αλγόριθμοι δεν μπορούν να δώσουν απάντηση σε λογικό χρονικό διάστημα. Δύο βασικοί αλγόριθμοι αυτής της κατηγορίας είναι οι GSAT [27] and Walksat [28]. Μια πιο λεπτομερής μελέτη των αλγορίθμων επίλυσης του προβλήματος της ικανοποιησιμότητας μπορεί να βρεθεί στο[Algorithms-for-satisfiability].

Εκτός του βασικού προβλήματος της ικανοποιησιμότητας υπάρχουν ένα πλήθος επεκτάσεων και παραλλαγών του που επιτρέπει μεγαλύτερη ευελιξία μοντελοποίησης από απλό SAT. Τέτοια παραδείγματα περιλαμβάνουν τους ποσοτικούς δυαδικούς τύπους (Quantified Boolean Formulas ή QBF), ψευδο-δυαδική επίλυση και βελτιστοποίηση (Pseudo-Boolean (PB) solving and optimization), μέγιστη Ικανοποιησιμότητα (MaxSAT), ελάχιστη ικανοποιησιμότητα (MinSAT) και παραλλαγές. Η πιο αποτελεσματικές αλγοριθμικές τεχνικές που χρησιμοποιούνται στο SAT έχουν επίσης εφαρμοστεί στις περισσότερες επεκτάσεις του, επιτρέποντας έτσι σημαντικές πρακτικές εφαρμογές.

Οι ψευδο-δυαδικοί περιορισμοί (Pseudo-Boolean constraints) γενικεύουν το SAT θεωρώντας γραμμικές ανισότητες για τις δυαδικές μεταβλητές αντί για απλές προτάσεις (clauses). Το πρόβλημα της βελτιστοποίησης των ψευδο-δυαδικών περιορισμών είναι NP-Hard. Επιπλέον, όπως στην περίπτωση του SAT, έχει προταθεί μια σειρά από αποτελεσματικούς αλγορίθμους για το πρόβλημα της βελτιστοποίησης ψευδο-δυαδικών περιορισμών [29] [30], που ενσωματώνουν και επεκτείνουν τις πιο αποτελεσματικές τεχνικές SAT και έχει αναπτυχθεί μια σειρά από PB-Solvers από την επέκταση υπαρχόντων SAT-Solver (π.χ. PBS [31], PUEBLO [32],. κ.α.).

Το πρόβλημα της μέγιστης ικανοποιησιμότητας (MaxSAT) μπορεί να διατυπωθεί ως εξής. Δεδομένης μιας έκφρασης σε CNF ποια είναι εκείνη η ανάθεση αληθοτιμών των μεταβλητών που περιέχονται στην έκφραση έτσι ώστε να ικανοποιείται η εκφραση και το πλήθος των προτάσεων που συναποτελούν την έκφραση και είναι αλήθεις να μεγιστοποιείται. Παραλλαγές του προβλήματος MaxSAT περιλαμβάνουν το partial MaxSAT, το weighted MaxSAT και το weighted partial MaxSAT. Στο partial MaxSAT οι προτάσεις χωρίζονται σε ισχυρές και μή. Οι ισχυρές πρέπει να πληρούνται, ενώ άλλες δεν είναι απαραίτητο. Στο weighted MaxSAT, κάθε πρόταση έχει ένα συγκεκριμένο βάρος, και ο στόχος είναι να μεγιστοποιηθεί το άθροισμα των βαρών των προτάσεων που ικανοποιούνται. Τέλος, στο weighted partial MaxSAT, οι ισχυρές προτάσεις πρέπει να ικανοποιούνται, και ταυτόχρονα να μεγιστόποιείται το άθροισμα βαρών των προτάσεων που ικανοποιούνται. Το MaxSAT και παραλλαγές παρέχουν μια ευέλικτη λύση μοντελοποίησης και έναν αυξανόμενο αριθμό από πρακτικές εφαρμογές [33], [34], όπως είναι η ικανότητα να λύσει τα προβλήματα βελτιστοποίησης PB. Παρά όμως τις πιθανές εφαρμογές, οι πιο αποτελεσματικές τεχνικές επίλυσης του SAT δεν μπορούν να εφαρμοστούν απευθείας σε αλγόριθμους για το MaxSAT. Έτσι, οι καλύτεροι αλγόριθμοι για το MaxSAT χρησιμοποιούν αναζήτηση branch and bound [33][34]. Επιπλέον υπάρχουν μελέτες που δείχνουν πώς μπορεί να χρησιμοποιηθούν επαναληπτικά αλγόριθμοι για το SAT για την επίλυση MaxSAT [35],[36]. Τέλος αναφέρουμε ότι το MAX-SAT είναι NP-complete πρόβλημα. Ακόμα και το MAX-2SAT, που περιορίζει το πρόβλημα σε εκφράσεις στις οποίες κάθε πρόταση περιέχει το πολύ δυο μεταβλητές, είναι NP-complete.

Αναλογο με το πρόβλημα της μέγιστης ικανοποιησιμότητας είναι αυτό της ελάχιστης ικανοποιησιμότητας (minimum satisfiability problem (MinSAT)). Στο πρόβλημα αυτό, επιχειρείται να βρεθεί μια ανάθεση αληθοτιμών που ικανοποιεί την δοθείσα έκφραση CNF, με την επιπλέον ιδιότητα ότι ο αριθμός των αληθών προτάσεων είναι ο ελάχιστος δυνατός. Ομοίως με το MaxSAT και στην περίπτωση του MinSAT υπάρχουν οι παραλλαγές partial MinSAT weighted MinSAT και weighted partial MinSAT.

Για την παρούσα εργασία, χρησιμοποιήθηκε ο SAT4J [37], ένας αποδοτικός SAT-Solver για εφάρμογές JAVA, που κληρονομεί πολλά χαρακτηριστικά του αλγορίθμου Chaff που αναφέραμε παραπάνω. Η εφαρμογή του SAT4J για την παρούσα εργασία αναλύεται στο κεφάλαιο 5.

2.5 Παρακολούθηση Συστημάτων Λογισμικού

Η παρακολούθηση (monitoring) είναι η διαδικασία παρατήρησης της συμπεριφοράς ενός συστήματος κατά την διάρκεια της λειτουργίας του. Η ανάλυση των παρατηρούμενων ακολουθιών γεγονότων μπορεί να βοηθήσει στην κατανόηση και την αποσφαλμάτωση των συστημάτων, στην διασφάλιση της επίδοσης και της ποιότητας των συτημάτων, στον εντοπισμό προβλημάτων και την ειδοποίηση των διαχειριστών, στον εντοπισμό παραβιάσεων των απαιτήσεων, καθώς επίσης και στην ενίσχυση της ασφάλειας των εφαρμογών και των συστημάτων. Σε αυτήν την υποενότητα αναφέρουμε βασικές τεχνικές παρακολούθησης συστημάτων λογισμικού καθώς επίσης και κάποια παραδοσιακά περιβάλλοντα παρακολούθησης λογισμικού που έχουν αναπτυχθεί και παρουσιάζονται σε σχετικές εργασίες.

Για την δυναμική ανάλυση ενός συστήματος λογισμικού χρειάζεται ένας μηχανισμός εξόρυξης σημαντικών γεγονότων από την εκτέλεση ενός συστήματος. Οι πιο κοινές μέθοδοι είναι η ενορχήστρωση κώδικα και η ενορχήστωση διερμηνέα. Η ενορχήστρωση κώδικα είναι η εισαγωγή εντολών εξόρυξης πληροφοριών σχετικών με την λειτουργία ενός συστήματος απευθείας στον πηγαίο κώδικα. Η τεχνική αυτή ήταν και παραμένει μια από τις βασικές τεχνικές παρακολούθησης εφαρμογών παρά τους περιορισμούς της. Συγκεκριμένα η ενορχήστρωση κώδικα προϋποθέτει την πρόσβαση στον πηγαίο κώδικα και απαιτεί μεγάλη προσπάθεια από την πλευρά του προγραμματιστη καθώς επίσης και καλή γνώση του συστήματος που ενορχηστρώνεται. Η ενορχήστρωση διερμηνέα αντίστοιχα είναι η εισαγωγή εντολών παρακολούθησης στον διερμηνέα της γλώσσας προγραμματισμού. Έτσι η τεχνική αυτή μπορεί να εφαρμοστεί σε οποιοδήποτε προγραμμα εκτελείται από τον διερμηνέα και δεν απαιτεί την πρόσβαση στον πηγαίο κώδικα της εφαρμογής.

Η παρακολούθηση λογισμικού είναι ένας τομέας με πολύ έρευνα και έχουν αναπτυχθεί διαφορες τεχνικές. Μεγάλη έρευνα έχει διεξαχθει στα συστήματα παρακολούθησης CORBA. Το OrWell [38] είναι ένα περιβάλλον παρακολούθησης λογισμικού για την κατανεμημένες εφαρμογές CORBA. Το περιβάλλον αυτό χρησιμοποιεί μια ιεραρχία κλάσεων γεγονότων για να ειδοποιεί ένα πλήθος παρατηρητών (observers) για τις αλληλεπιδράσει στο σύστημα. Παρέχει λεπτομερή ανάλυσης του παρακολουθούμενου συστήματος. Δεν προσδιορίζεται παρόλα αυτά αν το σύστημα παρακολούθησης είναι φορητό μεταξύ διαφρόρετικών συστημάτων λογισμικού ή όχι. Το Wabash [39][40] είναι ένα εργαλείο για έλεγχο (testing) και παρακολούθηση κατανεμημένων συστημάτων CORBA. Χρησιμοποιεί τα λεγόμενα CORBA interceptors για την συλλογή πληροφοριών χρόνου εκτέλεσης του συστήματος και χρησιμοποιεί γεωγραφικές πληροφορίες για την ομαδοποίηση των συστατικών του συστήματος παρακολούθησης παρακολούθησης (monitoring components).

Το JEWEL [41] είναι ακόμα ένα περιβάλλον παρακολούθησης κατανεμημένων συστημάτων λογισμικού. Το βασικό πλεονέκτημα αυτού του συστήματος είναι ότι υπόστηρίζει την ανάλυση

μέγαλου όγκου δεδομένων. Ωστόσο η ανάλυση αυτή παραμένει στο επίπεδο του πρωτοκόλλου επικοινωνίας (communication protocol level) και έτσι αποτυγχάνει να δώσει αντικειμενοστρεφή εικόνα του συστήματος.

Στα [42] και [43], οι συγγραφείς προτείνουν την τεχνική Remote Reflection ως μια τεχνική γενικού σκοπού για την παρακολούθηση (monitoring) την αποσφαλμάτωσης (debugging) και την οπτικοποίησης (visualisation) κατανεμημένων συστημάτων γραμμένων σε Java. Με την τεχνική αυτή, η εποπτεία και διαχείρηση ενός κατανεμημενου συστήματος γίνεται κεντρικά. Οι αντανακλαστικές μέθοδοι (reflective techniques) αναφέρονται στην δυνατότητα των συστημάτων να ανακαλύπτουν στοιχεία της δομής τους κατα την διάρκεια εκτέλεσης τους και να προσαρμόζουν δυναμικά την συμπεριφορά τους.

Στο [44] προτείνεται ένα γενικό περιβάλλον-πλαισιο ελέγχου συμμόρφωσης για κατανεμημένα συστήματα. Το περιβάλλον-πλαισιο αυτό χρησιμοποιεί αισθητήρες που ενεργροποιούνται από απομαρκυσμένους ελεγτές, και ένα κατανεμημένο σύστημα δημοσίευσης γεγονότων που επιτρέπει την εγγραφή για συγκεκριμένους τύπους γεγονότων που συλλέγονται απο τους αισθητήρες.

2.6 Adaptive Monitoring

Πέραν των παραδοσιακών συστημάτων παρακολούθησης συστημάτων λογισμικού έχουν προταθεί πολλές τεχνικές προσαρμοστικής παρακολούθησης συστημάτων. Σκοπός τους είναι να μειωθεί το κόστος της παρακολούθησης ενώ ταυτόχρονα να επιτευχθεί αποδεκτή ακρίβεια επίβλεψης. Στο [45] οι Talwar και οι συνεργάτες του ανέπτυξαν έναν διαχειριστή του συστήματος παρακολούθησης καθοδηγούμενο από πολιτικές (policy-driven monitoring manager), ο οποίος αναλύει τα συλλεγόμενα γεγονότα, αναγνωρίζει αλλαγές στην υποδομή του συστήματος παρακολούθησης που πρεπει να εκτελεστούν, και τις εκτελεί.

Στο [46] παρουσιάζεται μια προσέγγιση για την προσαρμοστική παρακολούθηση εταιρικών συστημάτων λογισμικού, που χρησιμοποιεί στατιστικές τεχνικές για την αναγνώριση σχέσεων μεταξύ διαφόρων μετρικών του υπό παρακολούθηση συστήματος. Αρχικά αναγνωρίζονται όλα τα ζευγάρια μετρικών του συστήματος που σχετίζονται γραμμικά μεταξύ τους και εκτιμώνται οι τιμές των παραμέτρων παλινδρόμησης για κάθε ζευγάρι, οι οποίες πρεπει να ισχύουν κατά την ορθή λειτουργία του συστήματος. Έπειτα ορίζεται ένα σύνολο μετρικών χαμηλού κόστους συλλογής που εξυπηρετούν ως βασικοί δείκτες καλής λειτουργίας του συστήματος. Έτσι, η παρακολούθηση του συστήματος ελαχιστοποιείται μόνο στην συλλογή αυτών των βασικών μετρικών. Αφού διαγνωστεί κάποια ανωμαλία σε αυτούς του δείκτες, αυξάνεται το επίπεδο παρακολούθησης συλλέγοντας δεδομένα που αφορούν μετρικές σχετικές με αυτήν που παρουσίασε την ανωμαλία. Ενώ οι προσεγγίσεις αυτές διευκολύνουν το πρόβλημα της προσαρμοστικής παρακολούθησης, απαιτούν την γνώση από πριν των πιθανών γεγονότων και των αντίστοιχων διορθωτικών ενεργειών[45]. Επιπλέον όπως υποστηρίζεται στο [47], συλλέγοντας μόνο ζευγάρια μετρικών που προσδιορίζονται κατά την σχεδίαση του συστήματος παρακολούθησης, όπως παρουσιάζεται στο [46], πολλές φορές μπορεί να αποτυγχάνει να διαγνώσει ανωμαλίες σε άλλες μη συσχετιζόμενες μετρικές [46].

Στο [47] οι συγγραφείς επιχειρούν να ξεπεράσουν αυτό το πρόβλημα, παρουσιάζοντας μια προσέγγιση για την προσαρμοστική παρακολούθηση απαιτήσεων λογισμικού βασισμένη σε γενετικούς αλγορίθμους. Στην προσεγγιση τους, που ονομάζεται Plato-RE, δεν προδιαγράφουν αλλαγές του συστήματος παρακολούθησης αλλά ορίζουν τα επιθυμητά αποτελέσματα μιας διαμόρφωσης του συστήματος παρακολούθησης και ένας εξελικτικός αλγόριθμος αναζητά λύσεις που θα ικανοποιούν αυτές τις απαιτήσεις. Σε αντίθεση με άλλα προσεγίσεις που έχουν προταθεί για την προσαρμοστική παρακολούθηση λογισμικού, που εστιάζουν στην ελαχιστοποίηση του όγκου δεδομένων που πρέπει να επεξεργαστούν για να διαγνώσουν την

κατάσταση του συστήματος, το Plato-RE εστιάζει στην ελαχιστοποίηση του όγκου δεδομένων που συλλέγονται από τους αισθητήρες.

Στο [48] οι A.Mos και J.Murphy παρουσιάζουν το Compas, ένα περιβάλλον-πλαισιο για την προσαρμοστική παρακολούθηση component-based συστημάτων λογισμικού δομημένων στην πλατφόρμα J2EE. Η προσέγγιση που προτείνουν στηρίζεται στην αυτόματη μεταβαση ενός αισθητήρα από την ενεργή στην παθητική παρακολούθηση και αντίστροφα. Ως ενεργή παρακολούθηση ορίζεται αυτή κατά την οποία όλα τα συλλεγόμενα δεδομένα προωθούνται σε άλλες μοναδες του συστήματος παρακολούθησης για περεταίρω ανάλυση ενώ στην περίπτωση της παθητικής παρακολούθησης τα συλλεγόμενα δεδομένα αποθηκεύονται τοπικά και προωθούνται μόνο όταν διαγνωστεί κάποια ανωμαλία ή γεμίση η μνήμη προσωρινής αποθήκευσης τους.

Στο [49], οι συγγραφείς εστιάζουν σε μια υποδομή προσαρμοστικής παρακολούθησης σε υπολογιστικό περιβάλλον πλέγματος (grid-computing environment) που ονομάζεται JAMM Χρησιμοποιόντας απομακρυσμένη κλήση μεθόδων (RMI), εκτελεί προγράμματα παρακολούθησης όπως το netstat, το iostat και το vmstat έτσι ώστε να εξάγαγει in order to στατιστικές πληροφορίες για τους κόμβους του πλέγματος όπως είναι το φορτίο της CPU του δικτύου ή της μνήμης κάθε κόμβου. Η παρακολούθηση κάποιου κόμβου ξεκινάει με την ανίχνευση δραστηριότητας σε κάποιες θύρες των κόμβων, από πράκτορες παρακολούθησης των θυρών αυτών (port monitoring agent).

Στο [50] παρουσιάζεται ένα περιβάλλον-πλαισιο για την παρακολούθηση απαιτήσεων λογιμικού. Το περιβάλλον αυτό μπορεί να προσαρμόζει κατάλληλα την λεπτομέρεια (granularity) των συλλογής και επεξεργασίας δεδομένων καταγραφής (log data), μετατρέποντας τα σε μια συμπαγή κωδικοποιημένη προτασιακή μορφή που μπορεί να χρησιμοποιηθεί από εργαλεία που επιλύουν το πρόβλημα της ικανοποιησιμότητας (SAT-Solvers).

Στο [51] οι συγγραφείς παρουσιάζουν μία προσέγγιση για την διάγνωση ανωμαλιών, που βασίζεται στην ανάλυση δέντρων κλήσεων και στην προσαρμοστκή παρακολούθηση λογισμικού. Χρησιμοποιεί το Kieker[52], ενα περιβάλλον πλαίσιο για την παρακολούθηση της επίδοσης και την δυναμική ανάλυση λογισμικού. Για κάθε κόμβο του δέντρου κλήσεων, που αναπαριστά την εκτέλεση μιας μεθόδου σε ένα συγκεκριμένο πλαίσιο, υπολογίζεται ένας δείκτη ανωμαλίας συγκρίνοντας παρατηρούμενες τιμές με προσδοκόμενες βασισμένοι σε προηγούμενες παρατηρήσεις. Τελος χρησιμοποιεί την OCL [53] για να ορίσει τους κανόνες προσαρμογή της παρακολούθησης.

2.7 Logging

Η τήρηση καταγραφών (logging) είναι η διαδικασία κατα την οποία συλλέγονται και καταγράφονται γεγονότα κατά την λειτουργία και αλληλεπίδραση στοιχείων και χρηστών του συστήματος[54]. Τα συστήματα καταγραφής διαφέρουν μεταξύ τους ως προς τον τρόπο αναπαράστασης, συλλογής και οργάνωσης των γεγονότων. Ως αναπαράσταση των γεγονότων εννοούμε χαρακτηριστικά όπως το σχήμα που έχουν, το είδος των πληροφοριών που απαρτίζουν ένα γεγονός, το επίπεδο της αφαίρεσης των γεγονότων. Συνήθως καταγράφεται ο χρόνος και η προέλευση των γεγονότων. Οσον αφορά τον τρόπο συλλογής των γεγονότων αυτοί μπορεί να διαφοροποιούνται ανάλογα με τις τεχνικές ενορχήστρωσης probe που χρησιμοποιούνται. Η ενορχήστρωση μπορεί να απαιτεί παρέμβαση στον πηγαίο κώδικα (intrusive), παρέμβαση στον ενδιάμεσο κώδικα όπως για παράδειγμα σε εφαρμογες Java στο bytecode (Bytecode Instrumentation – BCI) ή χρήση κάποιας ανεξάρτητης εφαρμογής που συλλέγει δεδομένα αλληλεπίδρασης και επικοινωνίας μεταξύ των διαφόρων στοιχείων

λογισμικού (non-intrusive). Τέλος όσον αφορά την οργάνωση των γεγονότων καταγραφής, δύο τυπικοί τρόποι είναι η καταγραφή τους σε κάποια εξειδικευμένη βάση δεδομένων, ή σε αρχεία καταγραφής (log files).

Ένα σημαντικό πρόβλημα που αντιμετωπίζουν οι μηχανικοί λογισμικού και οι διαχειριστές συστημάτων ως προς την καταγραφή γεγονότων είναι η ετερογένεια των μορφοτύπων που περιγράφουν τα ατομικά γεγονότα. Πολλές φορές διαφέρει ακόμα και η μορφή των χρονοσφραγίδων (timestamps). Η ετερογένεια αυτή προκαλεί σύγχηση και δυσχαιρένει την ανάλυση και επεξεργασία των γεγονότων αυτών. Έτσι έχουν προταθεί γενικοί και συχνά προτυποποιημένοι μορφότυποι και έχουν αναπτυχθεί εργαλεία για τον μετασχηματισμό της αναπαράστασης των γεγονότων από μια μορφή σε άλλη.

Σχετικά με τους μορφότυπους των γεγονότων καταγραφής, η IBM έχει συμβάλει με το πρότυπο Common Base Event (CBE)[55]. Το Common Base Event είναι η υλοποίηση από μέρους της IBM του Web Event Format, μία φορμά που αφορά την απεικόνιση γεγονότων και που προδιαγράφεται από το Web Services Distributed Management, ένα πρότυπο εγκεκριμένο από τον OASIS (Organization for the Advancement of Structured InformationStandards) που αφορά στην διαχείριση και παρακολούθηση κατανεμημένων πόρων. Το CBE στοχεύει να διασφαλίσει την ακρίβεια και να βελτιώσει την λεπτομέρεια των γεγονότων καταγραφής ώστε να βοηθήσει στην ανάπτυξη εύρωστων, διαχειρήσιμων και ντετερμινιστικών συστημάτων να διευκόλυνει την αποτελεσματική επικοινωνία μεταξύ των διαφορετικών στοιχείων μιας επιχειρήσης που υποστηρίζουν καταγραφή και διαχείρηση γεγονότων, προσδιορισμό προβλημάτων και επισκευή τους. Ορίζει την μορφή και το περιεχόμενο που πρέπει να έχουν τα δεδομένα καταγραφής χωρίς να υποδεικνύει κάποιον τρόπο για την οργάνωση των καταγραφών. Η μορφή που προτείνει αποτελεί μια τετράδα που περιλαμβάνει το αναγνωριστικό του δομοστοιχείου που αναφέρει την κατάσταση καθώς και αυτού που επιρεάζεται απο την κατάσταση αυτή (ενδεχομένως το ίδιο με αυτό που αναφέρει την κατάσταση), την περιγραφή της κατάστασης και πληροφορίες που επιτρέπουν την συσχέτιση καταστάσεων. Το πρότυπο CBE βρίσκει εφαρμογή στα πλαίσια του *Autonomic Computing Toolkit* [56] της IBM, το οποίο είναι ένα σύνολο εργαλείων για την διευκόλυνση ανάπτυξης αυτοδιαχειριζόμενων συστημάτων λογισμικού.

Αρκετά διαδεδομένοι είναι και οι μορφότυποι *Common Log Format* θαη *Combined Log Format*[57] που χρησιμοποιουνται από τον εξυπηρετητή διαδικτύου της Apache (*Apache HTTP Server*). Οι μορφότυποι αυτοί χρησιμοποιούνται για την καταγραφή τον αιτήσεων που λαμβάνει ο εξυπηρετητής.

Νωρίτερα τονίσαμε την ανάγκη μετασχηματισμού των γεγονότων καταγραφής από έναν μορφότυπο σε άλλον. Το Generic Log Adaptor (GLA) [58] είναι ένα χρήσιμο εργαλείο που επιτρέπει τον μετασχηματισμό αρχείων καταγραφής (log files) σε μορφή CBE, μέσω κανόνων που διατυπώνονται σε Java ή σεναρίων που περιλαμβάνουν τυπικές εκφράσεις για την περιγραφή της αντιστοίχισης του περιεχομένου του αρχείου καταγραφής στα στοιχεία του CBE.

Η ανάγκη παρακολούθησης, καταγραφής και ανάλυσης γεγονότων έχει αποδόσει μια σειρά τόσο εμπορικών όσο και ανοιχτού κωδικα, και ερευνητικών πλαισίων (frameworks) και εργαλείων. Σε αυτό το σημείο παρουσιάζουμε μερικά από τα πλαίσια και τα εργαλεία αυτά.

Η Sun Microsystems έχει αναπτύξει το DTrace[59] ένα περιβάλλον πλαίσιο καταγραφής ιχνών (tracing framework) για το λειτουργικό σύστημα Solaris. Το DTrace παρέχει την υποδομή για την δυναμική καταγραφή αναλυτικών στοιχείων (traces) από την λειτουργία του λειτουργικού συστήματος με σκοπό να διευκολύνει τους τεχνολόγους λογισμικού και τους διαχειριστές του συστημάτος να επιτηρούν, να αποσφαλματώνουν και να βελτιστοποιούν την λειτουργία του συστήματος. Το DTrace παρέχει την δυνατότητα ορισμού probes, και ενεργειών που πυροδοτούνται υπό ορισμένες συνθηκές. Η Microsoft παρέχει το δικό της πλαίσιο

καταγραφής, το Microsoft Operations Manager (MOM) [60]. Παρακολουθεί πολλούς servers σε ένα εταιρικό περιβάλλον τοποθετόντας πράκτορες MOM (MOM agents) στον υπολογιστή που παρακολουθείται. Οι πράκτορες MOM συλλέγουν γεγονότα από διάφορες πηγές σε αυτόν τον υπολογιστή, όπως το αρχείο καταγραφής συμβάντων των Windows (Windows Event Log,), και τα προωθούν στο διακομιστή διαχείρισης MOM (MOM management server) όπου αποθηκεύονται στη βάση δεδομένων MOM για περαιτέρω επεξεργασία.

Τα πλαίσια που αναφέραμε παραπάνω είναι εμπορικά. Ωστόσο όπως αναφέραμε υπάρχουν αρκετά περβάλλοντα πλαίσια καταγραφής ανοιχτού κώδικα. Το εργαλείο Business Intelligence Reporting Tool (BIRT)[61] είναι ένα σύστημα αναφορών ανοιχτού κώδικα βασισμένο στο Eclipse, που προορίζονται κυρίως για διαδικτυακές εφαρμογές σε Java και J2EE. Το BIRT έχει δύο κύρια στοιχεία : έναν σχεδιαστή αναφορών που βασίζεται σε Eclipse, και ένα στοιχείο runtime που μπορεί να προστεθεί σε ένα διακομιστή εφαρμογών. Μια ενδιαφέρουσα εφαρμογή της BIRT είναι η δημιουργία των εκθέσεων παρακολούθησης από την απόκτηση δεδομένων από διαφορετικά συστήματα καταγραφής κάποιας εφαρμογής λογισμικού. Ένα άλλο πλαίσιο καταγραφής ανοιχτού κώδικα που σχετίζεται επίσης με Eclipse είναι το Eclipse Test and Performance Tools Platform (TPTP) [62]. Το TPTP περιέχει μια σειρά εργαλείων για tracing, monitoring και profiling συμπεριλαμβανομένου και του Log and Trace Analyzer (LTA) [63], το οποίο είναι ένα εργαλείο του Eclipse που μπορεί να κατασκευάσει καταλόγους συμπτωμάτων για μια εφαρμογή. Ο πληροφορίες που ορίζονται στους καταλόγους αυτούς χρησιμοποιούνται για να βοηθήσουν τους αναλυτές πρόβλημάτων στην διαδικασία της αποσφαλματωσης και επίλυσης των προβλημάτων που ανακύπτουν κατά τη διάρκεια της παράταξης (deployment) και λειτουργίας του συστήματος. Αυτό το πακέτο περιλαμβάνει επίσης και το Generic Log Adapter (GLA) που περιγράψαμε παραπάνω.

Αρκετά πλαίσια και πρωτόκολλα καταγραφής έχουν επίσης προταθεί από την ακαδημαϊκή κοινότητα.

Στον τομέα της καταγραφής γεγονότων για την επαλήθευση του συστήματος, οι Andrews και Zhang εκθέτουν στο [64] την εφαρμογή των τεχνικών ανάλυσης των log αρχείων για να ελέγξετε τα αποτελέσματα των δοκιμών για ένα ευρύ φάσμα ελέγχων, όπως ελέγχων σε επίπεδο μονάδας και σε επίπεδο συστήματος, ελέγχου έναντι των απαιτήσεων για τις κρίσιμα και μη συστήματα, και την χρήση των τεχνικών ανάλυσης log αρχείων με άλλες συμβατικές μεθόδους δοκιμών. Για να εκτελέσετε αυτά τα διαφορετικά καθήκοντα δοκιμών, μια καταγραφή πλαίσιο που μπορεί να εμμείνει σε μια σαφώς καθορισμένη πολιτική υλοτομία στο οποίο αναφέρονται λεπτομερώς τι το λογισμικό υπό δοκιμή θα πρέπει να συνδεθείτε υπό ποιες συγκεκριμένες προϋποθέσεις πρέπει να υπάρχουν. Αυτό μπορεί να γίνει είτε μέσω του προτύπου αναθεώρηση κώδικα και τις διαδικασίες ελέγχου ή μέσω αυτόματων οργάνων κώδικα, όπως η εργασία αυτή πρέπει να συζητήσουμε.

Σε μια σχετική περιοχή της ανάλυσης αρχείων καταγραφής, παρουσιάζεται το εργαλείο Simple Clustering Logfile Tool (SLCT) [65], που χρησιμοποιεί έναν αλγόριθμο ομαδοποίησης δεδομένων για σύνολα δεδομένων καταγραφής για να καθορίσει συχνά μοτίβα από τα αρχεία καταγραφής, να κατασκευάει προφίλ αρχείων καταγραφών και να εντοπίσει ανώμαλες γραμμές από αρχεία καταγραφής. Με τη σύγκριση του αναμενόμενου προφίλ μιας λειτουργίας, μπορούν να εντοπιστούν σφάλματα σε ένα σύστημα, καθώς θα εμφανίζονται ως ανωμαλίες στα αρχεία καταγραφής. Στο [66], παρουσιάζεται μία πρωτοποριακή προσέγγιση χρησιμοποιώντας τον αλγόριθμο Teiresias, έναν αλγόριθμο εμπνευσμένο απο την βιοπληροφορικής, για να ταξινομήσει αυτόματα τα μηνύματα καταγραφής του συστήματος. Τα στατιστικά εμφάνισης και τα αποτελέσματα που βρέθηκαν είναι συγκρίσιμα με εκείνα των SLCT Vaarandi που αναφέρεται παραπάνω.

Τέλος, υπάρχει μια σειρά από άλλες εφαρμογές της καταγραφής και ανάλυσης καταγραφής. Ένα κοινό σενάριο είναι η μελέτη των καταγραφών για να καθορίστούν μοτίβα χρήσης μιας ιστοσελίδας ή μιας δικτυακής υπηρεσίας. Οι Lin και Hadingham χρησιμοποιούν την ανάλυση

της καταγραφής για την παρακολούθηση των συχνά επισκέψιμων σημείων μιας ιστοσελίδας [67]. Ομοίως στο [68] παρουσιάζεται μια μελέτη περίπτωσης χρήσης τεχνικών ανάλυσης καταγραφής, για την βελτιώση μίας κατανεμημένης διαδικτυακής υπηρεσιας αναλύοντας μοτίβα χρήσης και δραστηριότητων, καθώς επίσης και άλλες παραμέτρους, όπως τα σημεία εισόδου και εξόδου, και σημεία αποτυχιών σε μια συνεδρία. Η καταγραφή και η ανάλυση καταγραφών είναι επίσης σημαντική σε μελέτες της συμπεριφοράς των χρηστών και της αλληλεπίδρασης ανθρώπου μηχανής. Στο [69] παρουσιάζεται μια εφαρμογή της ανάλυσης καταγραφών για την ανάλυση της ανθρώπινης συμπεριφοράς σε διαδραστικά μέσα ψυχαγωγίας. Μελετώντας τα αρχεία καταγραφής των χειρισμών των χρηστών και τα πρωτόκολλα που χρησιμοποιούνται κατά την διάρκεια χειρισμών διαδραστικών μέσων, μπορεί να επιτευχθεί ο καλύτερος σχεδιασμός διεπαφών χρηστών (user interfaces). Τέλος, οι τεχνικές ανάλυσης καταγραφών μπορούν επίσης να εφαρμοστούν στην ανάλυση των διαδικασιών ελέγχου για τα αρχεία καταγραφής πρόσβασης. Στο [70], οι συγγραφείς παρουσιάζουν μια μελέτη των απαιτήσεων ελέγχου πρόσβασης για συστήματα υγειονομικής περίθαλψης.

2.8 Autonomic Computing

Σε αυτήν την ενότητα δίνουμε μια επισκόπηση της αυτόνομη υπολογιστική (autonomic computing) καθώς επίσης και μια παρουσίαση σχετικών εργασιών. Η αυτόνομη υπολογιστική είναι μία προσέγγιση στην ανάπτυξη πολύπλοκων υπολογιστικών συστημάτων που στόχο έχει την ανάπτυξη υπολογιστικών συστημάτων με αυτοοδιαχειριστικές ιδιότητες (self-managing properties). Ο όρος autonomic computing εισήχθει από την IBM το 2001 στο [71] για να περιγράψει τέτοια συστήματα. Ο όρος είναι εμπνευσμένος από το νευρικό σύστημα του ανθρώπου. Με τον ίδιο τρόπο που ο ανθρώπινος οργανισμός ρυθμίζει την θερμοκρασία του σώματος, τους ρυθμούς της καρδιάς, την εφύδρωση καθώς και κάθε άλλη λειτουργία του οργανισμού που γίνεται ασυναίσθητα και αυτόματα, χωρίς να απαιτείται σκέψη και προσπάθεια από το άτομο, έτσι και ένα αυτόνομο υπολογιστικό σύστημα διαχειρίζεται τον εαυτό του χωρίς να απαιτείται ανθρώπινη παρέμβαση. Αυτό σύμφωνα με το [71] θα διευκολύνει την διαχείρηση των συστημάτων λογισμικού και θα επιτρέψει στους διαχειριστές των συστημάτων και τεχνολόγους λογισμικού να εστιάσουν σε άλλα προβλήματα.

Η IBM στο[72] παρατηρεί ότι το βασικό εμπόδιο για την περαιτέρω πρόοδο στον τομέα της πληροφορικής δεν προέρχεται από την επιβράδυνση στο νόμο του Moore αλλά στην πολυπλοκότητα των σύγχρονων συστημάτων λογισμικού. Είναι η εκμετάλλευση του των τεχνολογιών που έχουν αναπτυχθεί στον απόηχο του νόμου του Moore που μας οδήγησε στα πρόθυρα μιας κρίσης πολυπλοκότητας. Η βιομηχανία των υπολογιστών έχει πέρασει δεκαετίες δημιουργίας συστημάτων θαυμαστής και συνεχώς αυξανόμενης πολυπλοκότητας. Αλλά σήμερα, η πολυπλοκότητα αυτή καθεαυτή είναι το πρόβλημα. Η απότομη αύξηση του κόστους της διαχείρισης της αυξανόμενης πολυπλοκότητας των συστημάτων πληροφορικής απειλεί να γίνει ένας σημαντικός ανασταλτικός παράγοντας της μελλοντικής ανάπτυξης της τεχνολογίας των πληροφοριών. Με άλλα λόγια, πολύπολοκότητα των σύγχρονων συστημάτων έχει κάνει την διαχείρηση τουςπολύ δαπανηρή και επιρρεπή σε λάθη.

Ο Paul Horn στο [71] περιγράφει εκτός των άλλων τα εξής χαρακτηριστικά των αυτόνομων υπολογιστικών συστημάτων:

- Για να είναι ένα σύστημα αυτόνομο, πρέπει να «γνωρίζει τον εαυτό του» και να αποτελείται από συστατικά που έχουν επίσης μια ταυτότητα συστήματος.

- Ένα αυτόνομο σύστημα πρέπει να μπορεί να διαμορφώσει τον εαυτό του κάτω από ποικίλες και απρόβλεπτες συνθήκες.

- Ένα αυτόνομο σύστημα δεν επαναπαύεται αλλά ψάχνει πάντα τρόπους για τη βελτιστοποίηση της λειτουργίας.

- Ένα αυτόνομο σύστημα πρέπει να εκτελέσει κάτι σχετικά με τη θεραπεία του. Θα πρέπει να είναι σε θέση να ανακάμψει από έκτακτα γεγονότα που ενδέχεται να προκαλέσουν ορισμένα τμήματα να δυσλειτουργούν.

- Ένας εικονικός κόσμος δεν είναι λιγότερο επικίνδυνος από τον φυσικό. Έτσι ένα αυτόνομο υπολογιστικό σύστημα πρέπει να είναι ειδικό στην αυτο-προστασίας

- Ένα αυτόνομο υπολογιστικό σύστημα γνωρίζει το περιβάλλον του και το πλαίσιο γύρω από τη δραστηριότητά του, και ενεργεί ανάλογα.

- Ένα αυτόνομο σύστημα δεν μπορεί να υπάρξει σε ένα ερμητικό περιβάλλον (και πρέπει να συμμορφώνεται με ανοικτά πρότυπα(open standards)).

- Ίσως το πιο κρίσιμο για το χρήστη, είναι το ότι, ένα αυτόνομο υπολογιστικό σύστημα θα προβλέψει τη βέλτιστη χρήση των πόρων του συστήματος ώστε να ανταποκριθεί στις ανάγκες του χρήστη, κρύβοντας παράλληλα την πολυπλοκότητα του.

Για να ενσωματώσει τα χαρακτηριστικά αυτά σε ένα αυτοδιαχειριζόμενο συστήμα, τα μελλοντικά αυτόνομα συστήματα λογισμικού θα πρέπει να έχουν τέσσερα βασικά χαρακτηριστικά σύμφωνα με το [73]. Αυτά είναι τα εξής:

Α***υτο-διαμόρφωση***

Η εγκατάσταση, και διαμόρφωση μεγάλων, πολύπλοκων συστημάτων λογισμικού είναι δύσκολη, χρονοβόρα, και επιρρεπής σε λάθη, ακόμη και για τους ειδικούς. Οι περισσότερες μεγάλες ιστοσελίδες και τα περισσότερα μεγάλα εταιρικά κέντρα δεδομένων είναι τυχαίες συλλογές εξυπηρετητών, δρομολογητών, βάσεων δεδομένων και τεχνολογιών σε διαφορετικές πλατφόρμες και διαφορετικούς προμηθευτές. Μπορεί να πάρει στις ομάδες εξειδικευμένων προγραμματιστών μήνες η συγχώνευση δύο συστημάτων ή η εγκατασταση μιας βασικής εφαρμογής ηλεκτρονικού εμπορίους όπως αυτού της SAP[73]. Τα αυτόνομα συστήματα λογισμικού πρέπει να προσαρμόζονται αυτόματα σε δυναμικά μεταβαλλόμενα περιβάλλοντα. Λέμε ότι ένα σύστημα υλικού ή λογισμικού έχει δυνατότητα αυτοδιαμόρφωσης, όταν έχει τη δυνατότητα να καθορίσει το ίδιο την διαμόρφωση του κατά την διάρκεια της λειτουργίας του. Αυτή η πτυχή της αυτο-διαχείρισης δηλώνει ότι τα αυτόνομα συστήαματα μπορούν να αλλάξουν την υποδομή τους δυναμικά προσθετοντας νέα χαρακτηριστικά, αναβαθμίζοντας το λογισμικό τους, αλλά και προσθετοντας ή αφερόντας διακομιστές χωρίς διακοπή των υπηρεσιών που παρέχει το σύστημα. Τα συστήματα πρέπει να είναι σχεδιασμένα ώστε να παρέχουν δυνατότητες προσθήκης plug and play συσκευών, διαμόρφωσης με χρήση οδηγών δημιουργίας διαμόρφωσης (configuration setup wizards), καθώς και ασύρματης διαχείρισης των διακομιστών. Έτσι θα μπορεί να προστεθεί λειτουργικότητα δυναμικά στην υποδομής του συστήματος με ελάχιστη ανθρώπινη παρέμβαση.

Αυτο-ίασης

Η IBM και άλλοι προμηθευτές λογισμικού έχουν αφιερώσει πολύ προσωπικό στην αναγνώριση, τον εντοπισμό, και τον προσδιορισμό της βασικής αιτίας των αστοχιών σε πολύπλοκα υπολογιστικά συστήματα. Μπορεί να χρειαστεί σε ομάδες προγραμματιστών

αρκετές εβδομάδες για να διαγνώσουν και να διορθώσουν προβλήματα, και πολλές φορές το πρόβλημα εξαφανίζεται μυστηριωδώς χωρίς διάγνωση των αιτιών. Τα αυτόνομα συστήματα λογισμικού πρέπει να είναι σε θέση να εντοπίσουν, να διαγνώσουν και να επισκευάσουν προβλήματα που προκύπτουν από σφάλματα ή παραλείψεις στο λογισμικό και το υλικό. Τα αυτόνομο συστήματα θα πρέπει επίσης να προβλέπουν προβλήματα και να λαμβάνουν μέτρα για να αποτρέψουν μια αποτυχία να επηρεάσει την ορθή λειτουργία των εφαρμογών του συστήματος. Ο στόχος της αυτο-ίασης πρέπει να είναι η ελαχιστοποίηση των διακοπών της λειτουργίας του συστήματος, προκειμένου να κρατήσει τις εφαρμογές των επιχειρήσεων και διαθέσιμες ανά πάσα στιγμή.

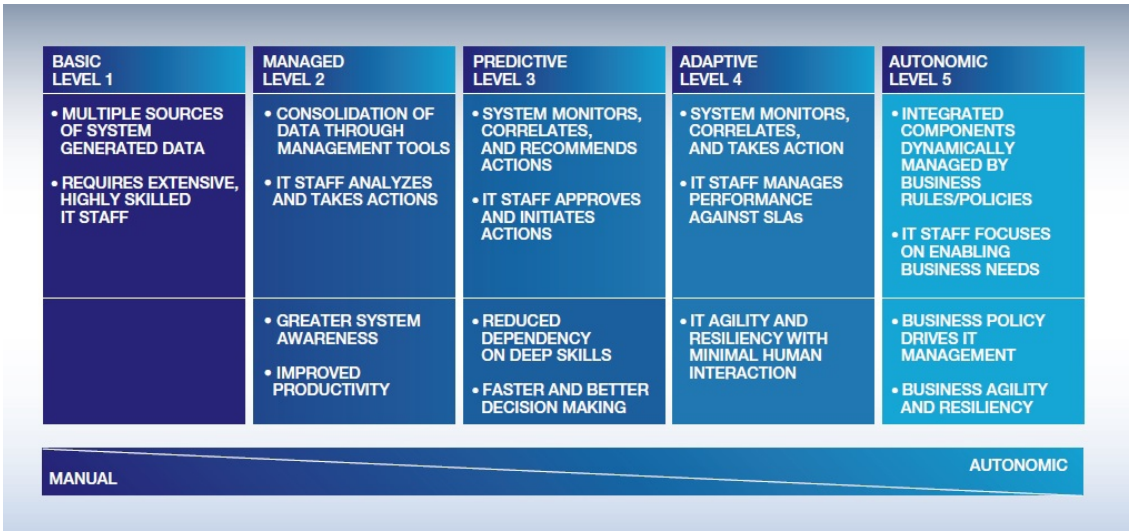
Αυτο-βελτιστοποίηση

Πολύπλοκα συστήματα ενδιάμεσου λογισμικού (middleware), όπως το WebSphere, ή συστήματα βάσεων δεδομένων, όπως η Oracle ή η DB2, μπορεί να έχουν εκατοντάδες ρυθμιζόμενες παραμέτρους που πρέπει να ρυθμιστούν σωστά ώστε το σύστημα να λειτουργεί βέλτιστα. Ωστόσο, λίγοι άνθρωποι γνωρίζουν καλά πως να συντονίσουν τις παραμέτρους αυτές. Τέτοια συστήματα συχνά συνδυάζονται με άλλα, εξίσου πολύπλοκα συστήματα. Κατά συνέπεια, οι η ρυθμιση τους για βελτιστοποίηση ενός μεγάλου υποσυστήματος μπορεί να έχει απρόβλεπτες συνέπειες για ολόκληρο το σύστημα. Τα αυτόνομα συστήματα λογισμικού πρέπει να ψάχνουν συνεχώς τρόπους για να βελτιωσουν την λειτουργίας τους, με σκοπό την πιο βελτιστοποίηση των επιδόσεων του και την ελαχιστοποίηση του κόστους λειτουργίας τους.

Αυτο-προστασίας

Τα αυτόνομα συστήματα λογισμικού πρέπει να είναι σε θέση να προβλέπουν, να ανιχνεύουν και να προστατεύουν τον εαυτό τους από επιθέσεις. Πρέπει να μπορούν να ορίζουν και να διαχειρίζονται την πρόσβαση των χρηστών σε όλους τους υπολογιστικούς πόρους στο πλαίσιο της επιχείρησης, για την προστασία από μη εξουσιοδοτημένη πρόσβαση, την ανίχνευση εισβολών, την αναφορά και παρεμπόδιση τέτοιων δραστηριότητων καθώς αυτές εκτελούνται, καθώς και την παροχή δυνατοτήτων επαναφοράς σε ασφαλείς καταστάσεις λειτουργίας. Τα συστήματα αυτά σύμφωνα με το[74] θα πρέπει να οικοδομηθούν πάνω σε βασικές τεχνολογίες ασφαλείας, που είναι διαθέσιμες σήμερα, όπως το πρωτόκολλο LDAP (Lightweight Directory Access Protocol), το πρωτόκολλο Kerberos, το πρωτόκολλο SSL (Secure Socket Layer).

Η IBM έχει ορίσει πέντε επίπεδα αυτονομίας [72][74][75][76]. Τα επίπεδα αυτά Όπως φαίνεται στο παραπάνω σχήμα είναι το βασικό (basic), το διαχειριζόμενο (managed), το προγνωστικό (predictive), το προσαρμοστικό (adaptive) και το αυτόνομο (autonomic).



Σχήμα 2.8.1: Επίπεδα εφαρμογής του Autonomic Computing (Autonomic Computing Adoption Model Levels) [74]

1. **Βασικό Επίπεδο (Basic Level):** Το βασικό επίπεδο αντιπροσωπεύει το επίπεδο, στο οποίο βρίσκονται πολλά συστήματα πληροφορικής σήμερα. Η διαχείρηση κάθε στοιχείου του συστήματος γίνεται από εξειδικευμένους διαχειριστές που στήνουν το σύστημα και το παρακολουθούν και αν χρειαστεί κάνουν μετατροπές ή αντικαθίστουν τμήματα του.

2. **Διαχειριζόμενο Επίπεδο (Managed Level):** Στο διαχειριζόμενο επίπεδο, χρησιμοποιούνται τεχνολογίες διαχείρισης συστημάτων για τη συλλογή πληροφοριών από ετερογενή συστήματα σε λιγότερους κονσόλες, μειώνοντας έτσι τον χρόνο που απαιτείται για το διαχειριστή για να συλλέξει και να συνθέσει τις πληροφορίες καθώς τα συστήματα γίνονται όλο και πιο σύνθετα.

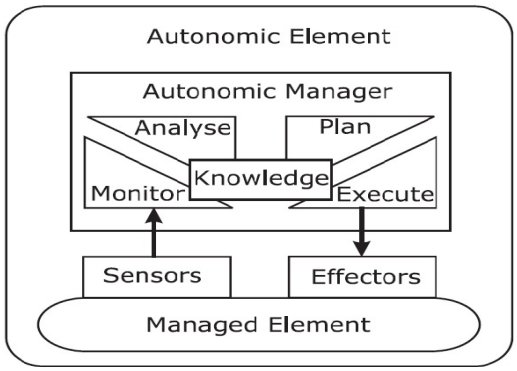
3. **Προγνωστική Επίπεδο (Predictive Level):** Στο προγνωστικό επίπεδο, καθώς εισάγονται νέες τεχνολογίες που παρέχουν συσχέτιση μεταξύ των διαφόρων στοιχείων του συστήματος, το ίδιο το σύστημα μπορεί να αρχίσει να αναγνωρίζει μοτίβα, να προβλέψει τη βέλτιστη διαμόρφωση, και να παρέχει συμβουλές σχετικά με το τι πορεία δράσης θα πρέπει να λάβει ο διαχειριστής.

4. **Προσαρμοστικό Επίπεδο (Adaptive Level):** Στο προσαρμοστικό επίπεδο δεδομένου ότι αυτές οι τεχνολογίες βελτιώνονται και καθώς οι άνθρωποι γίνονται αποκτούν εμπιστοσύνη στις συμβουλές και την προγνωστική ικανότητα των συστημάτων αυτών, επιτρέπεται στο ίδιο το σύστημα να λάβει αυτόματα τις σωστές ενέργειες βασιζόμενο στις πληροφορίες που έχει στη διάθεσή του και τη γνώση του τι συμβαίνει στα σύστημα.

5. **Αυτόνομο Επίπεδο (Autonomic Level):** Τέλος, στο πλήρως αυτόνομο επίπεδο, η λειτουργία του συστήματος διέπεται από επιχειρηματικές πολιτικές και στόχους. Οι χρήστες αλληλεπιδρούν με το σύστημα για την παρακολούθηση των επιχειρηματικών διαδικασιών ή για να τροποποιήσουν τους στόχους.

Για την επίτευξη του αυτόνομου υπολογιστικής, η IBM έχει προτείνει ένα μοντέλο αναφοράς για αυτόνομους βρόχους ελέγχου [72], η οποία ονομάζεται μερικές φορές βρόχος MAPE-K από τα αρχικά των λέξεων Monitor, Analyse, Plan, Execute, Knowledge βρόχο και απεικονίζεται στο Σχήμα 2.5. Οπως φαίνεται απο το σχήμα ο βασικός βρόχος αποτελείται από

την τέσσερα βασικά στάδια. Το στάδιο της παρακολούθησης όπου συλλέγονται τα δεδομένα παρακολούθησης, το στάδιο της ανάλυσης των δεδομένων με σκοπό να βρεθούν ενδιαφέρουσες καταστάσεις, το στάδιο της κατάστρωσης ενός πλάνου εκτέλεσης και το στάδιο της εκτέλεσης του πλάνου αυτού.



Σχήμα 2.8.2: MAPE-K

Στον αυτόνομο βρόχο MAPE-K, όπως φαίνεται στο σχήμα 2.5. το διαχειριζόμενο στοιχείο (managed elemenent) αντιπροσωπεύει οποιοδήποτε πόρο υλικού ή λογισμικού στο οποίο δίνεται αυτόνομη συμπεριφορά με την σύζευξη του με έναν αυτόνομο διαχειριστή. Ως εκ τούτου, ένα διαχειριζόμενο στοιχείο μπορεί είναι ένα διακομιστή ιστού (web server), μία βάση δεδομένων, ένα συγκεκριμένο συστατικό του λογισμικού μιας εφαρμογής (π.χ. ο βελτιστοποιητής ερωτημάτων μιας βάσης δεδομένων), το λειτουργικό σύστημα, ή ένα σύμπλεγμα μηχανών (cluster) σε ένα περιβάλλον πλέγματος (grid environment).

Οι αισθητήρες (sensors ή probes) συλλέγουν πληροφορίες σχετικά με το διαχειριζόμενο στοιχείο. Για έναν διακομιστή ιστού, αυτά θα μπορούσε να περιλαμβάνει το χρόνο απόκρισης στα αιτήματα των πελατών, την χρήση δικτύου και δίσκου και την χρήση της μνήμης και της κεντρικής μονάδας επεξεργασίας (CPU).

Οι ενεργοποιητές (Effectors) πραγματοποιούν αλλαγές στο διαχειριζόμενο στοιχείο. Η αλλαγή μπορεί να είναι λεπτομερής (fine-grained), π.χ. με αλλαγή των παραμέτρων διαμόρφωσης σε έναν διακομιστή ιστού [77][78] ή μη λεπτομερής (coarse-grained), π.χ. προσθέτοντας ή αφαιρώντας διακομιστές σε ένα σύμπλεγμα διακομιστών ιστού [79].

Ο αυτόνομος διαχειριστής (autonomic manager) είναι ένα στοιχείο λογισμικού που ιδανικά μπορεί να ρυθμιστεί από την φυσικούς διαχειριστές (human managers) με χρήση στόχων υψηλού επιπέδου και χρησιμοποιεί τις μετρήσεις από τους αισθητήρες και την εσωτερική γνώση του συστήματος για να σχεδιάσει και να εκτελέσει, βασιζόμενο σε αυτούς τους στόχους, ενέργειες χαμηλού επιπέδου που είναι αναγκαίες για την επίτευξη αυτών των στόχων. Η εσωτερική γνώση του συστήματος είναι συχνά ένα αρχιτεκτονικό μοντέλο του διαχειριζόμενου στοιχείου. Οι στόχοι συνήθως εκφράζονται με τη χρήση πολιτικών ECA (από τα αρχικά event, condition και action) και συναρτήσεις χρησιμότητας (utility function policies) [76],[80].

Τα αυτονόμα στοιχεία (autonomic elements) μπορούν να συνεργάζονται μεταξύ τους για την επίτευξη ενός κοινού στόχου [72], π.χ. οι διακομιστές σε ένα σύμπλεγμα συνεργάζονται για την βελτιστοποίηση της κατανομής των πόρων στις αιτήσεις των χρηστών προσπαθόντας παράλληλα να ελαχιστοποιήσουν του συνολικό χρόνο απόκρισης ή το χρόνο εκτέλεσης των εφαρμογών. Έτσι, οι αυτόνομοι διαχειριστές μπορεί να χρειαστεί να γνωρίζουν όχι μόνο την κατάσταση του στοιχείου που διαχειρίζονται, αλλά επίσης και το περιβάλλον τους. Αυτή η ιδέα της συνεργασίας των επιμέρους αυτόνομων στοιχείων για την επίτευξη ενός κοινού στόχου αποτελεί θεμελιώδη πτυχή των ερευνητικών συστημάτων πολλαπλών πρακτόρων (multi-agent

systems), και ως εκ τούτου, δεν προκαλεί έκπληξη το γεγονός ότι γίνεται σημαντική έρευνα για την υλοποιηση αυτόνομων στοιχείων με χρήση συνεργαζόμενων πρακτόρων.[73][81][82] [83]. Μια εναλλακτική λύση για τη συνεργασία πολλαπλών παραγόντων είναι μια ιεραρχική δόμηση των αυτόνομων στοιχείων[84].

Η IBM έχει αναπτύξει μια πρότυπη υλοποίηση του βρόχου MAPE-K που ονομάζεται Autonomic Management Engine, ως μέρος του Autonomic Computing Toolkit (ACT)[56]. Το ACT παρέχει ένα πρακτικό πλαίσιο και μία υλοποίηση αναφοράς για την ενσωμάτωση του δυνατοτήτων αυτονομίας σε συστήματα λογισμικού[85]. Έτσι, δεν είναι ένα πλήρης αυτόνομος διαχειριστής, αλλά παρέχει τα θεμέλια για την οικοδόμηση αυτόνομων διαχειριστών. Είναι υλοποιημένο σε Java, αλλά μπορεί να επικοινωνεί με άλλες εφαρμογές μέσω μηνυμάτων XML, π.χ. για την ανίχνευση αναλύοντας τα αρχεία καταγραφής μιας διαχειριζόμενης εφαρμογής.

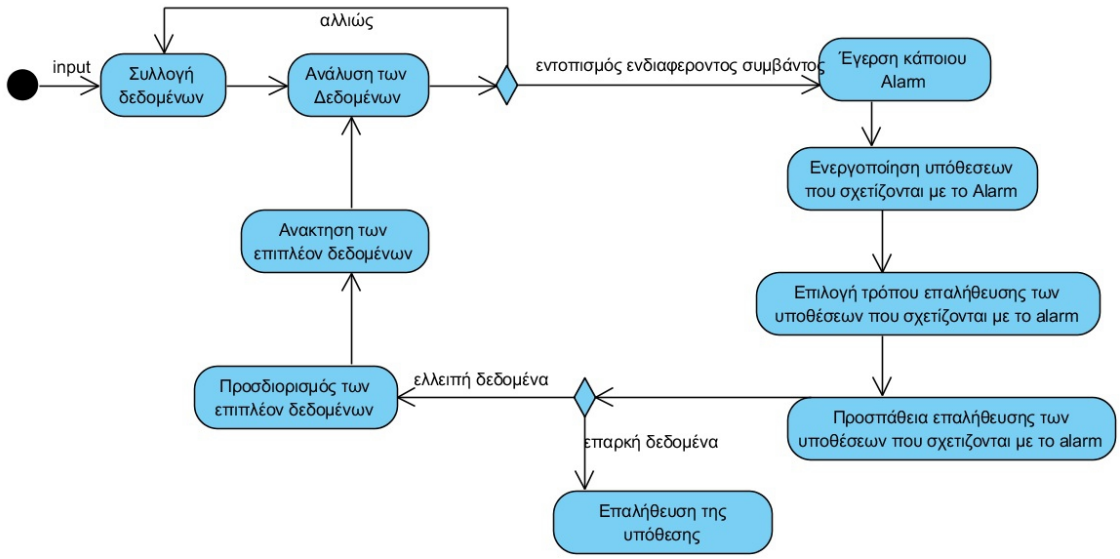
Κεφάλαιο 3: Αρχιτεκτονική και λειτουργία του συστήματος

3.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζουμε την δομή της αρχιτεκτονικής του προτεινόμενου συστήματος, το οποίο καλείται να λύσει το πρόβλημα της προσαρμοστικής επίβλεψης συστημάτων λογισμικού.

Ένα από τα βασικά ζητήματα κατά την σχεδίαση ενός συστήματος λογισμικού είναι η επεκτασιμότητα και η ευκολία προσαρμογής σε νέες ανάγκες πιθανόν διαφορετικές από αυτές που είχε στο μυαλό του ο σχεδιαστής. Στην διάρκεια ζωής ενός συστήματος λογισμικού οι απαιτήσεις συνεχώς αλλάζουν και εμπλουτίζονται, προστίθεται νέα λειτουργικότητα. Η αρχιτεκτονική του περιβάλλοντος-πλαισίου πρεπει να είναι τέτοια ώστε να υποστηρίζει μελλοντικές αλλαγές και επεκτάσεις. Έτσι το πρώτο βήμα κατά την σχεδίαση του συστήματος είναι η αναγνώριση των υπο-λειτουργιών που απαιτούνται για την επίλυση του προβλήματος που εξετάζουμε και η ανάθεση τους σε διαφορετικά components. Η λειτουργία κάθε component πρέπει να είναι ανεξάρτητη από τις λειτουργίες των υπολοίπων και να εξαρτάται μόνο από διαπροσωπία που αυτά παρέχουν. Η σχεδίαση ενός τέτοιου συστήματος πρέπει να περιλαμβάνει όλα τα components που είναι απαραίτητα ώστε να εκτελεστούν οι λειτουργίες που απαιτούνται για την επίλυση του προβλήματος της προσαρμοστικής επίβλεψης λογισμικού.

Κεντρική ένοια για την περιγραφή της λειτουργίας του συστήματος είναι αυτή του συναγερμού (alarm) και της υπόθεσης (hypothesis). Τα Hypotheses μοντελοποιούν υποθέσεις για το υπο παρακολούθηση σύστημα και μοντελοποιούνται ως δέντρα στόχων ενώ τα Alarms μοντελοποιούν οντότητες που ενεργοποιουν υποθέσεις βασιζόμενα σε συγκεκριμενα μοτιβα γεγονοτων. Όταν ενεργοποιηθεί μια υπόθεση στην συνέχεια πρέπει να ελεγχθεί και να επιβεβαιωθεί ή να απορριφθέι. Για να το κάνει αυτό απαιτείται να βρεθεί ένα κατάλληλο πλάνο για την επαλήθευση της υπόθεσης. Αυτό γίνεται με την βοήθεια του SAT4J και η διαδικασία περιγράφεται με λεπτομέρεια στο κεφάλαιο 5. Για την επαλήθευση μιας υπόθεσης μπορεί να χρειαστεί η συλλογή επιπλέον πληροφοριών. Το παρακάτω διάγραμμα δείχνει τις βασικές λειτουργίες του βρόχου ελέγχου του περιβάλλοντος-πλαισίου.



Σχήμα 3.1.1: Διάγραμμα Δραστηριότητας Βασικού Βρόχου Εκτέλεσης

Έτσι με μια πρώτη ματιά εντοπίζουμε την ανάγκη ύπαρξης των εξής components:
Ένα component υπεύθυνο για την συλλογή των δεδομένων που θα έχει την δυνατότητα επαναπροσδιοσμου των δεδομένων που συλλέγονται,

- Ένα component υπεύθυνο για την αλλαγή της σταθμης παρακολούθησης και προσδιορισμό των δεδομένων προς συλλογή.
- Ένα component υπεύθυνο για την πυροδότηση των alarms και την ενεργοποιηση και απενεργοποίηση των υποθέσεων. Το component αυτό πρέπει να έλεγχει των γεγονότα παρακολούθησης με σκοπό των εντοπισμό κατάλληλων μοτίβων γεγονότων ικανών να πυροδοτήσουν κάποιο alarm.
- Ενα component υπεύθυνο για την κατάστρωση ενός πλάνου για την απόδειξη μιας υπόθεσης στηριζόμενο στο μοντέλο δένδρων στόχων και σε πολιτικές ορισμένες από τον χρήστη, και
- Ένα component υπεύθυνο για την επαλήθευση των υποθέσεων και την αίτηση επιπλέον πληροφοριών σε περίπωση που οι διαθέσιμες δεν είναι επαρκείς.

Επιπλέον αναλύοντας περαιτέρω τις απαιτήσεις που πρέπει να πληρεί το υπό σχεδίαση συστήμα αναγνωρίζουμε τις εξής:

- Την δυνατότητα ορισμού νέων υποθέσων και alarm.
- Την δυνατότητα ορισμού κανόνων προσαρμογής του συστήματος παρακολούθησης και του τρόπου αποδειξης ενός κόμβου στόχου.
- Την δυνατότητα συλλογής δεδομένων από ποικίλες πηγές και ενεργοποίησης και απενεργοποίησης των πηγών αυτών ανάλογα με τις ανάγκες παρακολούθησης.
- Τέλος θέλουμε να υπάρχει η δυνατότητα ορισμού και διαφορετικών στρατηγικών για την επαλήθευση και την παρακολούθηση των υποθέσεων. Με βάση αυτές τις στραγητικές θα καθορίζεται ποιες υποθέσεις θα παρακολουθούνται, ποιες υποθέσεις θα επαληθευτούν η σειρά και ο τρόπος επαλήθευσης τους.

Για την εκπλήρωση των δύο πρώτων απαιτήσεων όπως έχουμε ήδη αναφέρει στην εισαγωγή μοντελοποιούμε τις υποθέσεις ως δέντρα στόχων και ορίζουμε alarms για τις υποθέσεις αυτές. Τα δέντρα αυτά αναπαριστούν υποθέσεις για το υπο παρακολουθηση σύστημα και μέσω της δομής τους δείχνουν τους τρόπους με τους οποίους αυτές μπορούν να επαληθευτούν. Οπως έχουμε ήδη αναφέρει οι κόμβοι των δέντρων έχουν επισημειώσεις που όριζουν τα δεδομένα που χρειάζονται για την επαλήθευση των κόμβων αυτών αλλά και τις προυποθέσεις που πρέπει

να πληρούν τα δεδομένα αυτά ώστε να αποδειχθεί η ισχύς του αντίστοιχου στόχου. Συγκεκριμένα για την επαλήθευση μιας υπόθεσης επιλέγεται ένα μονοπάτι του δέντρου και αποδεικνύεται κάθε κόμβος του μονοπατιού αυτού. Οι επισημειώσεις στους κόμβους του μονοπατιού δίνουν τα επιπλέον δεδομένα που πρέπει να συλλεγούν και σε συνδυασμό με την δομή του δέντρου ένα σύνολο κανόνων για την προσαρμογή του συστήματος παρακολούθησης και την επαλήθευση μιας υπόθεσης. Τα alarms αναπαρισταναι ως ξεχωριστές οντότητες που συσχετίζονται με τις υποθέσεις και ενεργοποιούνται με την παρατήρηση συγκεκριμένων μοτίβων δεδομένων. Έτσι χρειαζόμαστε ένα component που θα αναπαριστά τα δέντρα στόχων, ένα component για την αναπαράσταση των alarms και ένα component που θα μας δίνει την δυνατότητα να τα διαχειριζόμαστε μέσω κάποιας διεπαφής χρήστη. Τα δεντρα στόχων οι επισημειώσεις και τα alarms παρουσιάζονται αναλυτικά στο κεφάλαιο 4.

Για την υποστήριξη της δυνατότητας συλλογής δεδομένων από ποικίλες πηγές χρειαζόμαστε άλλο ένα component που θα αναλαμβάνει να φέρει τα συλλεγόμενα δεδομένα σε μια πρότυπη μορφή ώστε να είναι δυνατή η επεξεργασία τους από τα υπόλοιπα components του συστήματος. Θεωρούμε δηλαδή δεδομένα από διαφορετικές πηγές είναι πιθανό να ακολουθούν διαφορετικά πρότυπα μεταξύ τους και διαφορετικά από αυτά "καταλαβαίνουν" τα components του framework που αναπτύσουμε.

Όσον αφορά την τελευτάια απαίτηση χρειάζεται ένα component που θα μοντελοποιεί αυτές τις στρατητικές και ένα component που θα παίρνει αποφάσεις εφαρμόζοντας αυτές τις στρατηγικές.

Το κεφάλαιο αυτό οργανώνεται ως εξής. Στην επόμενη παράγραφο παρουσιάζουμε το ψηφιδικό διάγραμμα της αρχιτεκτονικής στο οποίο φαίνονται όλα τα components που προέκυψαν από την παραπάνω ανάλυση και δίνεται μια σύντομη περιγραφή του κάθε component. Στην συνέχεια ακολουθεί η αναλυτική περιγραφή του κάθε component. Για κάθε component επεξηγούμε τις διαπροσωπίες που υλοποιεί και παρουσιάζουμε τα συστατικά του components. Τελειώνουμε το κεφάλαιο αυτό δίνοντας ένα ακολουθιακά διάγραμματα για κάθε μία από τις λειτουργίες του βασικού βρόχου λειτουργίας του συστήματος.

3.2 Επισκόπηση της αρχιτεκτονικής

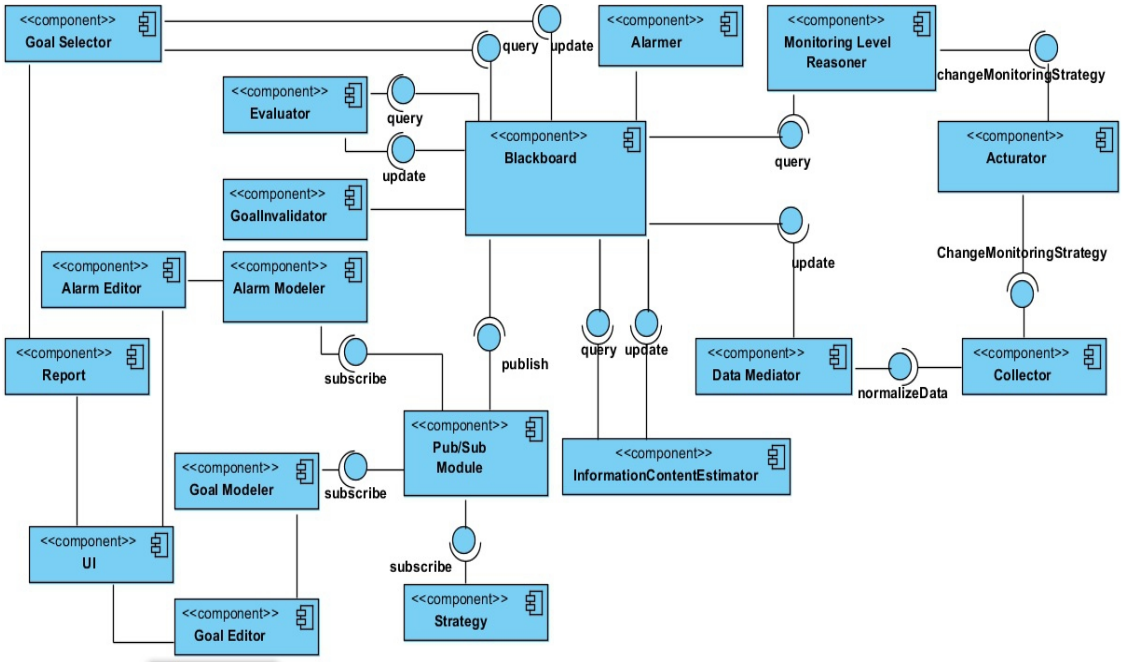
Σε αυτήν την ενότητα παρουσιάζουμε την αρχιτεκτονική του framework που προτείνουμε για το πρόβλημα της προσαρμοστικής παρακολούθησης λογισμικού. Με τον όρο αρχιτεκτονική του framework εννοούμε το σύνολο των components που το απαρτίζουν και πρέπει να υλοποιεί είτε το ίδιο το framework είτε το χρησιμοποιούν και το επεκτείνουν. Επίσης στην αρχιτεκτονική ενός framework παρουσιάζονται και οι αλληλεπιδράσεις μεταξύ των components που ορίζονται.

Η αρχιτεκτονική που προτείνουμε βασίζεται στο αρχιτεκτονικό μοτίβο μαυροπίνακα. Το μοτίβο αυτό παρουσιάζεται στο [86] και είναι χρήσιμο σε προβλήματα στα οποία δεν είναι γνωστή κάποια ντετερμινιστική λύση. Σύμφωνα με αυτό το μοτίβο υπάρχει ένα κεντρικό component, το οποίο στο εξής θα ονομάζουμε Blackboard, στο οποίο βρίσκεται όλη η γνώση για την επίλυση του προβλήματος και τα υπόλοιπα components χρησιμοποιούν αυτήν την γνώση για να επιλύσουν το πρόβλημα μερικώς και να αποθηκεύσουν τα αποτελέσματα τους πίσω στο blackboard, έτσι ώστε η νέα γνώση να είναι διαθέσιμη στα υπόλοιπα. Θεωρούμε λοιπόν το κάθε component σαν έναν αυτόνομο πράκτορα που χρησιμοποιεί την ήδη υπάρχουσα γνώση για να παράξει νέα.

Το πότε θα "τρέξει" ένα component εξαρτάται από την διαθεσιμότητα των δεδομένων που χρειάζεται το component στο Blackoard. Μια ιδέα είναι κάθε component να ζητά περιοδικά

(polling) την γνώση που είναι αποθηκευμένη στο Blackboard και να ελέγχει την νέα γνώση που παρήχθει. Επειδή οι τεχνική αυτή θα συνεπαγόταν μεγάλο κόστος σε υπολογιστικούς πόρους και συνεπώς χαμηλή απόδοση προτείνεται η χρήση του σχεδιαστικού μοτίβου Publisher/Subscriber σύμφωνα με το οποίο υπάρχει ένα κεντρικό component το οποίο ενημερώνεται για σημαντικά γεγονότα του συστήματος και στην συνέχεια ενημερώνει όλους τους ενδιαφερόμενους. Έτσι υπάρχουν components (subscribers) που εκδηλώνουν ενδιαφέρον για συγκεκριμένο είδος γεγονότων και components (publishers) που παράγουν αυτά τα γεγονότα και ενημερώνουν το κεντρικό αυτό component. Με αυτόν τον τρόπο ένας publisher δεν χρειάζεται να ξέρει ποια components πρέπει να ενημερώσει και έτσι πετυχαίνεται αποσύνδεση (decoupling) του καλούντος από τον καλούμενο. Επιπλέον τονίζεται ότι ένα component μπορεί να είναι ταυτόχρονα publisher και subscriber για κάποιο άλλο είδος δεδομένων.

Ακολουθεί το ψηφιδικό διάγραμμα της αρχιτεκτονικής του συστήματος στο οποίο φαίνονται όλα όσα αναπτύξαμε παραπάνω.



Σχήμα 3.2.1: Ψηφιδικό Διάγραμμα Αρχιτεκτονικής Περιβάλλοντος-Πλαισίου

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

- **Blackboard**: Αυτό είναι το βασικό component του framework και χρησιμοποιείται για να "κρατάει" την κατάσταση του framework καθώς επίσης και όλα τα events που δημιουργούνται απο καθένα από τα components του. Λειτουργεί ως ένα κεντρικό σημείο αποθήκευσης της κατάστασης του συστήματος παρακολούθησης και του συστήματος υπο παρακολούθηση και όλων των γεγονότων και ενδιάμεσων αποτελεσμάτων τα οποία παράγουν τα components που απαρτίζουν το περιβάλλον πλαίσιο.

- **Pub/Sub Module:** Αυτο το component αναλαμβάνει την ενημέρωση των διαφορων components για τα events του συστήματος για τα οποια έχουν εκδηλώσει ενδιαφέρον. Ο σκοπός του είναι να διευκολύνει την επικοινωνία και ενημέρωση των διαφορων components προσφέροντας στα διαφορα components την δυνατότητα αποστολής

ενημερώσεων χωρίς γνώση των παραλήπτων. Τα διαφορα components απλά παράγουν events ενήμερώνουν το Pub/Sub Module για τα events που αυτά και αυτό με την σειρά του ενημερώνει όλα τα components που έχουν εκδηλώσει ενδιαφέρον για τον συγκεκριμένο τύπο γεγονότων. Για να το κάνει αυτό προσφέρει στα components την δυνατότητα εγγραφής τους για συγκεκριμένους τύπους από events και κρατά λίστες με τα εγγεγραμένα για καθε event components.

- **Goal Modeler:** Αυτό το component μοντελοποιεί τα δέντρα στόχων και χρησιμοποιείται για να αναπαραστήσει το μεταμοντέλο στο οποίο πρέπει να υπακούουν τα μοντέλα δέντρων στόχων.

- **Goal Editor:** Αυτό το component παρέχει τις συναρτήσεις μέσω των οποίων μπορεί να οριστούν νέα δεντρα στόχων και να διαγραφούν ή να τροποποιηθούν ήδη υπάρχοντα. Παρέχει επίσης δυνατότητες φόρτωσης και μόνιμης αποθήκευσης των μοντέλων δενδρων στόχων καθώς επίσης και έλεγχο και επαλήθευση της ορθότητας των μοντέλων (δηλαδή του αν υπακούουν στο μεταμοντέλο που τα προδιαγραφει).

- **Alarm Modeler:** Αυτό το component μοντελοποιεί τα alarms και χρησιμοποιείται για να αναπαραστήσει το μεταμοντέλο στο οποίο πρέπει να υπακούουν τα alarms.

- **Alarm Editor:** Αυτό το component παρέχει τις συναρτήσεις μέσω των οποίων μπορεί να οριστούν νέα alarms και να διαγραφούν ή να τροποποιηθούν ήδη υπάρχοντα. Παρέχει επίσης δυνατότητες φόρτωσης και μόνιμης αποθήκευσης των alarms αυτών καθώς επίσης και έλεγχο και επαλήθευση της ορθότητας τους σύμφωνα με το μεταμοντέλο που έχει οριστεί για αυτά.

- **Report:** Αυτό το component χρησιμοποιείται για την ενημέρωση του χρήστη για την κατάσταση του συστήματος. Ενημερώνει τον χρήστη για την ενεργοποίηση κάποιου alarm και για την εξέλιξη της προσπάθειας απόδειξης μιας υπόθεσης τις αλλαγές στο επίπεδο παρακολουθησης κ.ο.κ.

- **UI**: Αυτό το component χρησιμοποιείται για την επικοινωνία του framework με τον χρήστη. Μέσω αυτού του component ο χρήστης μπορεί να ορίσει νέα goal models και alarms, να τα αποθηκεύσει καθώς επίσης και να τροποποιήσει κατάλληλα τα ήδη υπάρχοντα goal models και alarms. Επιπλέον μέσω αυτού του component λαμβάνει ενημερώσεις σχετικά με την ενεργοποίηση κάποιου alarm καθώς επίσης και με την στρατηγική που θα ακολουθηθεί για την επαλήθευση μιας υπόθεσης που σχετίζεται με κάποιο ενεργό alarm.

- **Strategy:** Μοντελοποιεί τις διάφορες στατηγικές βάση των οποίων αποφασίζεται ποιες υποθέσεις παρακολουθούνται, ποιες υποθέσεις θα επαληθευτούν και η σειρά και ο τρόπος επαλήθευσης τους. Οι στρατηγικές αυτές μπορεί να δίνουν προτεραιότητα σε κάποιες υποθέσεις έναντι άλλων, να καθορίζουν αν οι υποθέσεις πρέπει να εξεταστούν παράλληλα ή σειριακά κ.ο.κ. Η απόφαση αυτή μπορεί να βασιζεται σε εξαρτήσεις αναμεσα στις υποθέσεις, στην σημαντικότητα των υποθέσεων, στην πιθανότητα ισχύς τους, στον εκτιμώμενο χρονο που απαιτείται για την επαλήθευση τους κ.ο.κ.

- **Goal Selector:** Ο Goal Selector χρησιμοποιεί τις στρατηγικές που ορίζονται στο component Strategy και αποφασίζει ποιες υποθέσεις πρέπει να παρακολουθούνται, ποιες πρέπει να επαληθευτούν, με ποια σειρα και με ποιο τρόπο πρέπει να γίνει η επαλήθευση τους.

- **Evaluator:** Αυτό το component προσπάθεί να επαληθεύσει ή να αρνηθεί μια υπόθεση. Για να το κάνει αυτό ενδεχεται να προσπαθήσει να αποδείξει άλλα goals και να ζητήσει την συλλογή επιπλέον δεδομένων δηλώνοντας στο Blackboard την ανάγκη για δεδομένα αυτά. Αποδεικνύει στόχους του μοντέλου δένδρων στόχων είτε μέσω της δομή του δέντρου αυτής καθαυτής είτε χρησιμοποιώντας τα γεγονότα παρακολούθησης αξιοποιώντας τις επισημειώσεις των κόμβων-στόχων.

- **Monitoring Level Reasoner:** Αυτό το component λειτουργεί ως συντονιστής της διαδικασίας αλλαγής του επιπέδου επίβλεψης του συστήματος. Γνωρίζει ανά πάσα στιγμή τα δεδομένα που συλλέγονται, αποφασίζει ποια δεδομένα χρειάζεται να συλλεγούν και αναθέτει στον Actuator την συλλογή αυτών τον δεδομένων.

- **InformationContentEstimator:** Αυτό το component αξιολογεί το περιεχόμενο της πληροφορίας των δεδομένων παρακολούθησης. Τα χαρακτηριστικά (attributes) των δεδομένων παρακολούθησης διαφέρουν ως προς την σημαντικότητα της πληροφορίας που περιέχουν. Για παράδειγμα αν η τιμή ενός attribute είναι ίδια σε όλα τα γεγονότα που συλλέγονται, τότε το attribute αυτό δεν έχει μεγάλη αξία. Ο InformationContentEstimator χρησιμοποιει τα δεδομένα παρακολούθησης και κάνει μια αξιολογηση του περιεχομένου τους. Τα αποτελέσματα που παράγει μπορούν να αξιοποιηθούν έτσι ώστε να απόφασιστεί ποια δεδομένα πρέπει να συλλεγούν κάθε δεδομένη στιγμή.

- **Actuator:** Ο Actuator είναι υπεύθυνος για την αλλαγή του επιπέδου επίβλεψης. Το κάνει αυτό με δύο τρόπους:
    1. Με την ενεργοποίηση ή απενεργοποίηση κάποιων από τους Collectors.
    2. Με την ανάθεση σε κάποιον Collector της συλλογής περισσότερων ή λιγότερων δεδομένων. Για παράδειγμα κάποια attributes των events μπορεί να μην τόσο σημαντικά όπως για παραδειγμα το timestamp. Έτσι μπορεί να αλλάζει το επίπεδο παρακολουθησης ζητώντας μόνο μερικά από τα χαρακτηριστικά των events.

- **Collector**: Αλληλεπιδρά με το monitored system, συλλέγει οτι δεδομένα χρειάζονται για να αποδειχθεί η ισχύς ενός στόχου. Περιλαμβάνει όλα τα components και τις τεχνικές που απαιτούνται για την λήψη γεγονότων απο το υπο έλεγχο σύστημα. Οι πηγές των δεδομένων μπορεί να είναι οποιεσδήποτε. Μπορεί να είναι log files, βάσεις δεδομένων ή data streams από κάποιον κόμβο δικτύου. Το σύστημα λογισμικού που θα επεκτείνει το framework θα μπορεί να ορίσει τους δικούς του Collectors υλοποιόντας τις μεθόδους που ορίζονται στο interface του Collector. Ο Collector πρέπει να προσφέρει την δυνατότητα ορισμού των δεδομένων που πρέπει να συλλέξει. Τέλος για την επαλήθευση ενός στόχου μπορεί να απαιτούνται δεδομένα που προσπεράστηκαν. Σε αυτήν την περίπτωση απαιτείται αποθήκευση αυτών των δεδομένων για κάποιο χρονικό διάστημα ωστε να καθίσταται δυνατή συλλογή τους εκ των υστέρων.

- **Data Mediator:** Αυτο το component δεχεται δεδομένα από τον Collector και τα φέρνει σε μια ομογενοποιημένη (normilized) μορφή έτσι ώστε να είναι δυνατή η περαιτέρω επεξεργασία τους απο τα υπόλοιπα componets του framework. Τα δεδομένα που συλλέγονται απο τους Collector μπορεί να προέρχονται από διαφορες πηγές και έτσι μπορεί να ακολουθούν διαφορετικά πρότυπα. Έτσι υπάρχει η ανάκη ομογενοποίησης τους. Ο χρήστες του framework μπορούν να ορίζουν τους δικούς τους Mediators για τους δικούς τους Collectors. Τέλος ο Data Mediator ενημερώνει το Blackboard με τα νέα δεδομένα.

- **Alarmer:** Αυτό το component είναι υπεύθυνο για την ενεεργοποίηση και απενεργοποίηση των alarms. Ελέγχει τα δεδομένων που συλλέγονται και αποθηκεύονται στο Blackboard με σκοπό την ανακάλυψη ενδιαφερόντων γεγονότων. Οταν συμβεί αυτό πυροδοτεί το κατάλληλο alarm και εκκινεί την διαδικασία απόδειξης μιας των συσχετιζόμενων υποθέσεων.

Σε επίπεδο υλοποίησης το EMF, το οποίο περιγράφηκε στο προηγούμενο κεφάλαιο, αποτελεί μια καλή λύση για τον ορισμό των components Goal Modeler, Goal Editor, Alarm Modeler, Alarm Editor και UI και αυτό χρησιμοποιήσαμε για την πρωτότυπη υλοποίηση του περιβάλλοντος-πλαισίου.

3.3 Αναλυτική Περιγραφή των Components

Σε αυτό το κεφάλαιο παρουσιάζουμε αναλυτικά ένα ένα τα components του συστήματος που παρουσιάστηκε στην προηγούμενη ενότητα. Εξηγούμε τις διεπαφές κάθε component και αναλύουμε την εσωτερική δομή καθενός παραθέτοντας όπου είναι απαραίτητο επιπλέον uml διαγράμματα που δείχουν τα συστατικά τους components (sub-components) και τις αλληλεπιδράσεις μεταξύ τους.

Όπως φαίνεται και από το ψηφιδικό διάγραμμα της προηγούμενης παραγράφου έχουμε απλοποιήσει αρκετά τις διεπαφές των components που απαρτήζουν το σύστημα. Έχουμε θεωρήσει τα components που χρησιμοποιούν την γνώση του blackboard ως αυτόνομους πράκτορες που κάνουν subscribe στο Pub/Sub Module και "ενεργοποιούνται" μόνο όταν οι λάβουν κάποια ενημέρωση απο το Pub/Sub Module για κάποιο γεγονός που τους ενδιαφέρει. Έτσι η μόνη απαίτηση για αυτά τα components είναι να υλοποιούν μια μέθοδο που πρέπει να υλοποιούν όλοι οι subscribers έτσι ώστε να μπορούν να κληθούν από το Pub/Sub Module. Αυτή η μέθοδος όπως φαίνεται και απο το ψηφιδικό διάγραμμα είναι η μέθοδος notifyNNewEvent και την υλοποιούν ο Evaluator ο Goal Selector ο Monitoring Level Reasoner και ο Alarmer.

3.3.1 Goal Selector και Strategy

Όπως αναφέραμε και παραπάνω ο GoalSelector αναλαμβάνει να αποφασίσει τον τρόπο με τον οποίο θα επαληθευτεί επαληθευτούν οι υποθέσεις που σχετίζονται με κάποιο ενεργό alarm. Αυτό το κάνει χρησιμοποιώντας κάποιον sat solver. Στην περίπωση μας χρησιμοποιήσαμε τον Sat4j τον οποίο περιγράψαμε στο προηγούμενο κεφάλαιο. Ο Sat4j υπολογίζει όλους τους δυνατόυς τρόπους με τους οποίους μπορεί να ικανοποιηθεί μια φόρμουλα CNF. Έτσι λοιπόν ώς πρώτο βήμα απαιτείται η μετατροπή του μοντέλου στόχων (goal model) σε μορφή CNF. Αυτό το αναλαμβάνει το component GoalModelToCNFConverter. Οι δυνατοί τρόποι με τους οποίους μπορεί να ικανοποιηθεί η φόρμουλα CNF που παρήχθη με βάση το μοντέλο στόχων στην ουσία αντιστοιχεί σε όλους τους δυνατούς τρόπους με τους οποίους μπορεί να αποδειχθεί η ισχύς των υποθέσεων που θέλουμε να αποδείξουμε. Από όλους τους δυνατούς τρόπους απόδειξης των ενεργών υποθέσεων πρέπει να επιλεγεί η καταλληλότερη. Έτσι απαιτείται ταξινόμηση τους και ο έλεγχος τους μία προς μία μέχρι να αποδειχθεί ή να ισχύς ή η άρνηση του στόχου. Αυτήν την εργασία αναλαμβάνει ο SatSolutionSortingStrategy του component Strategy. Αυτή η λίστα με τους προς επαλήθευση στόχους πρέπει να ταξινομηθεί έτσι ώστε η επαλήθευση των στόχων να γίνει με έναν εύλογο και αποδοτικό τρόπο. Την ταξινόμηση της λίστας αυτης αναλαμβάνει το υπο-component του Strategy, GoalSortingStrategy. Το component Strategy περιλαμβάνει επίσης τα υπο-componens HypothesisEvaluationStrategy, το οποίο προσδιορίζει την σειρά και τον τρόπο με τον οποίο θα επαληθευτούν οι υποθέσεις, και HypothesisMonitoringStrategy που ορίζει ποιες υποθέσεις πρέπει να παρακολουθηθούν. Για

παράδειγμα μπορεί να οριστεί ένας παράγοντας σημαντικότητας για κάθε υπόθεση και όταν στο σύστημα όλα βαίνουν καλώς να παρακολουθούνται μόνο οι υποθέσεις που θεωρούνται πιο σημαντικές.

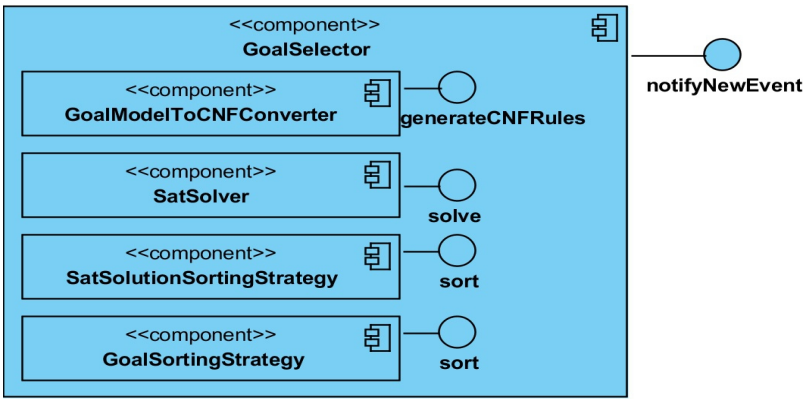
Συνοπτικά λοιπόν τα συστατικά components του GoalSelector είναι τα εξής:

- **GoalModelToCNFConverter**: Παράγει μια φόρμουλα CNF βασιζόμενο στο μοντέλο στόχων στις αληθοτιμές των υποθέσεων που έχουν επαληθευτεί και στις υποθέσεις που θέλουμε να αποδείξουμε.

- **SatSolver**: Υπολογίζει όλους τους δυνατούς τρόπους επαλήθευσης των ενεργών υποθέσεων.

- **SatSolutionSortingStrategy**: Ταξινομεί το σύνολο των λύσεων που επιστρέφει ο Sat Solver με σκοπό την χρήση της καταλληλότερης πρώτα και διαδοχικά τις υπόλοιπες σε περίπτωση αποτυχίας.

- **GoalSortingStrategy**: Ταξινομεί τους στόχους κάθε λύσης που επιστρέφει ο Sat Solver με σκοπό την επαλήθευση τους με έναν ευλογο και αποδοτικό τρόπο.



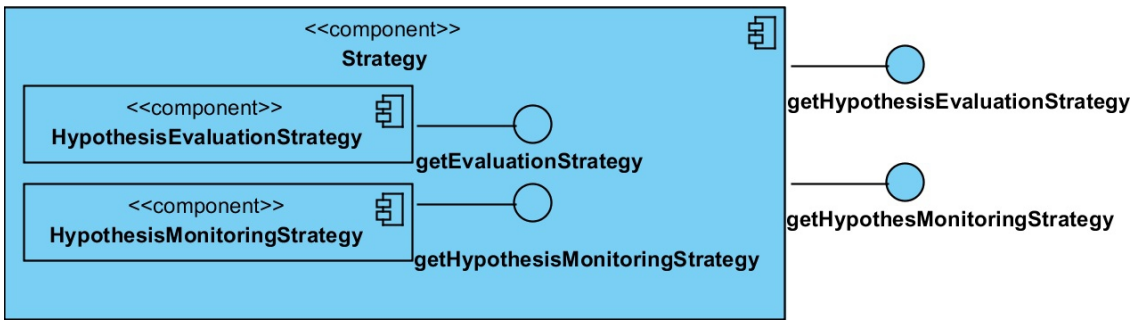
Σχήμα 3.3.1: Ψηφιδικό Διάγραμμα του Goal Selector

Όσον αφορά τις διεπαφές που πρέπει να υλοποιεί ο GoalSelector δεν είναι άλλη απο την notifyNewEvent. Μέσω αυτής της μεθόδου ο GoalSelector λαμβάνει ενημερώσεις σχετικά με τα alarms που ενεργοποιούνται και τις στόχων που αποδεικνύονται. Σε κάθε περίπτωση πρέπει να υπολογιστεί εκ νέου κάποια στρατηγική για την επαλήθευση των υποθέσεων. Η απόδειξη ενός στόχου μπορεί να καθιστά μια νέα στρατηγική για την απόδειξη των υποθέσεων καταλληλότερη από την από την ακολουθούμενη στρατηγική.

Τα συστατικά components του GoalSelector είναι τα εξής:

- **HypothesisEvaluationStrategy:** Επιλέγει την σειρά και τον τρόπο επαλήθευσης των ενεργών υποθέσεων.

- **HypothesisMonitoringStrategy:** Αυτό το component αποφασίζει ποιες από τις υποθέσεις παρακολουθουμε με σκοπό να εγείρουμε κάποιο alarm σε περίπτωση που χρειαστεί.



Σχήμα 3.3.2: Ψηφιδικό Διάγραμμα του Strategy

Η διεπαφή του Strategy ορίζει τις εξής συναρτήσεις:

- **getHypothesisEvaluationStrategy:** Επιστρέφει την στρατηγική επαλήθευσης των υποθέσεων.
- **GetHypothesisMonitoringStrategy:** Επιστρέφει την στρατηγική παρακολούθησης των υποθεσεων.

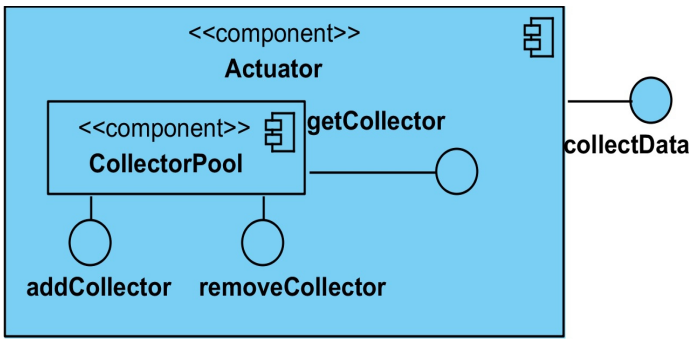
3.3.2 Actuator

Ο Actuator όπως είπαμε και παραπάνω είναι υπεύθυνος για την αλλαγή του επιπέδου παρακολούθησης. Το κάνει αυτό είτε ενεργοποιώντας ή απενεργοποιώντας καποιον Collector είτε μειώνοντας ή αυξάνοντας την λεπτομέρεια των πληροφοριών ποιυ συλλέγονται. Για να το κάνει αυτό χρειάζεται να γνωρίζει τους διαθέσιμους Collectors. Αυτούς τους κρατάει στο component CollectorPool το οποίο παρέχει δυνατότητες ενεργοποιησης και απενεργοποιησης των Collectors.

Η διέπαφή του Actuator έχει την συνάρτηση collectData η οποια καλείται για να αυξήσει ή να μειώσει το επίπεδο παρακολούθησης. Ως ορίσματα δέχεται την πηγή των δεδομένων, το είδος των γεγονότων και μια boolean τιμή η οποία καθορίζει αν θα το συγκεκριμένο είδος γεγονότων πρέπει να συλλεγεται ή όχι.

Η διεπαφή του CollectorPool περιλαμβάνει τις εξής συναρτήσεις:

- **getCollector**: Επιστρέφει τον Collector που του ζητάται από το CollectorPool, έτσι ώστε να ενεργοποιηθεί ή να απενεργοποιηθεί ή να του ανατεθεί η συλλογή περισσότερων ή λιγότερων δεδομένων.
- **addCollector:** Προσθέτει έναν νέο Collector στο CollectorPool.
- **removeCollector:** Αφαιρεί έναν Collector από το CollectorPool.



Σχήμα 3.3.3: Ψηφιδικό Διάγραμμα του Actuator

3.3.3 Collector

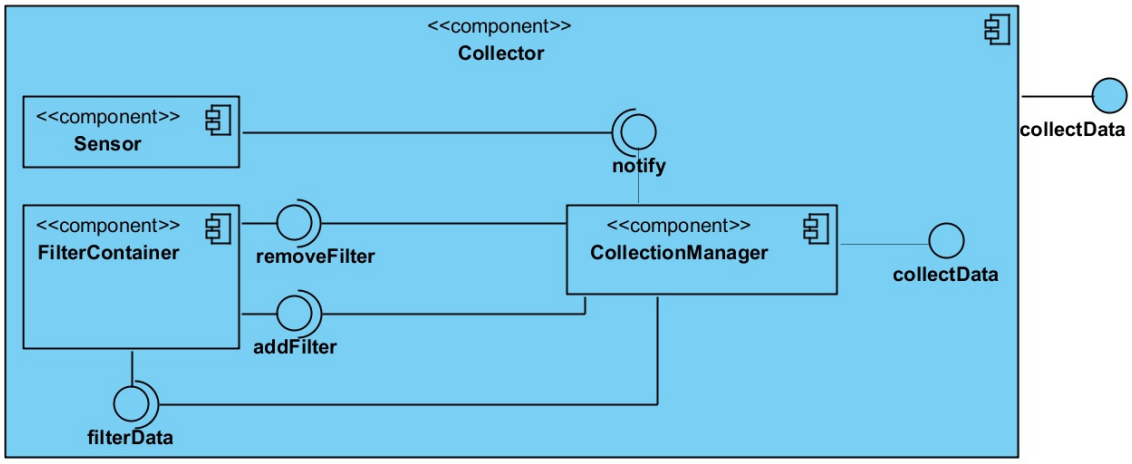
Ο Collector είναι υπεύθυνος για την συλλογή και προωθηση των γεγονότων παρακολούθησης προς το Blackboard. Περιλαμβάνει τα εξής components:

- **Sensor:** Το component αυτό είναι αυτό που κάνει την συλλογή των δεδομένων. Μπορεί να είναι κάποιο διαβάζει δεδομένα από μία βάση δεδομένων από ένα αρχείο καταγραφής ή οποιαδήποτε άλλη πηγή δεδομένων. Ότι δεδομένα διαβάζει τα προωθεί στον CollectorManager που σύνδέεται μαζί του.

- **FilterContainer**: Ο FilterContainer περιέχει όλα τα ενεργά φίλτρα για τα γεγονότα παρακολούθησης. Χρησιμοποιείται για ένα πρώτο φιλτράρισμα των δεδομένων όπως για παράδειγμα για την επιλογή μόνο εκείνων των attributes των δεδομένων που μας ενδιαφέρουν. Η διεπαφή του όρίζει τρείς συναρτήσεις. Η δύο, addFilter και removeFilter, χρησιμοποιούνται για την προσθήκη ή αφαίρεση κάποιου φίλτρου για τα δεδομένα, και βοηθούν στην αύξηση και μείωση του επιπέδου παρακολούθησης. Η τρίτη (filterData) χρησιμοποιείται για το φιτράρισμα αυτό καθαυτό.

- **EventDB**: Το component αυτό αποθηκεύει προσωρινά τα δεδομένα παρακολούθησης που δεν στέλνονται προς το Blackboard. Θεωρούμε ότι μπορεί να υπάρξουν σενάρια λειτουργίας του συστήματος κατά τα οποία η απόδειξη μιας υπόθεσης μπορεί να απαιτήσει προγενέστερα δεδομένα. Σε αυτήν την περίπτωση χρειάζεται κάποιο είδος προσωρινής αποθήκευσης αυτών των δεδομένων και αυτό είναι και το νόημα αυτού του component.

- **CollectionManager**: Ο CollectionManager διαχειρίζεται τον Collector και συντονίζει την διαδικασία συλλογής των δεδομένων παρακολούθησης. Προσθέτει και αφαιρεί φίλτρα αναλογα με τις ανάγκες παρακολούθησης, ενημερώνεται για τα νέα γεγονότα και τα προωθεί προς το blackboard αν χρειάζεται. Τα δεδομένα που δεν στέλνονται προς το Blackboard αποθηκεύονται προσωρινά για κάποιο μικρό χρονικό διάστημα σε περίπτωση που χρειαστούν για την επαλήθευση κάποιας υπόθεσης.



Σχήμα 3.3.4: Ψηφιδικό Διάγραμμα του Collector

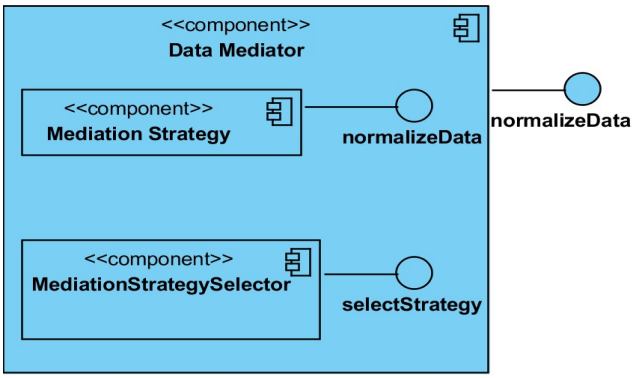
3.3.4 Data Mediator

Οπως έχουμε αναφέρει και στην προηγούμενη παράγραφο ο DataMediator χρησιμοποιείται για να φέρει τα δεδομένα παρακολούθησης σε μία πρότυπη μορφή, κατανοητή από τα υπόλοιπα components του περιβάλλοντος-πλαισίου. Το component αυτό είναι βασικό για την υποστήριξη της συλλογής δεδομένων από διάφορες πηγές. Κάθε πηγή μπορεί να ακολουθεί διαφορετικά πρότυπα για την αναπαράσταση των δεδομένων της και έτσι απαιτούνται διαφορετικές στρατηγικές για την προτυποποίηση δεδομένων από διαφορετικές πηγές. Έτσι ο DataMediator αποτελείται από τα εξής components:

- **MediationStrategy:** Το component αυτό αποτελεί την στρατηγική προτυποποίησης των δεδομένων παρακολούθησης. Η στρατηγική αυτή μπορεί να εξαρτάται από την πηγή των δεδομένων το είδος τους κ.τ.λ. Η διεπαφή του ορίζει την συνάρτηση normalizeData που καλείται για κόποιο γεγονός του συστήματος υπο παρακολούθηση και επιστρέφει μία πρότυπη μορφή αυτού του γεγονοτος.

- **MediationStrategySelector:** Το component αυτό αποφασίζει για την στρατηγική που πρέπει να ακολουθηθεί για την προτυποποίηση των συλλεγόμενων δεδομένων. Η απόφαση αυτή μπορεί να στηρίζεται στην πηγή των δεδομένων το είδος τους κ.τ.λ. Η διεπαφή του ορίζει την συνάρτηση selectStrategy η οποία καλείται για κόποιο γεγονός του συστήματος υπο παρακολούθηση και επιστρέφει την κατάλληλη στρατηγική για την προτυποποίηση του.

Όσον αφορά την διεπαφή του DataMediator αυτή ορίζει μόνο την συνάρτηση normalizeData η οποία προτυποποιεί τα δεδομένα παρακολούθησης και προωθεί τα προτυποποιημένα δεδομένα στο Blackboard.



Σχήμα 3.3.5: Ψηφιδικό Διάγραμμα του Data Mediator

3.3.5 Pub/Sub Module

Ο Pub/Sub αναλαμβάνει την ενημέρωση των components του συστήματος για events για τα οποια έχουν εκδηλώσει ενδιαφέρον. Ο σκοπός του είναι να διευκολύνει την επικοινωνία και ενημέρωση των components του περιβάλλοντος-πλαισίου προσφέροντας τους την δυνατότητα εκδήλωσης ενδιαφέροντος για συγκεκριμένα γεγονότα και λήψης ενημερώσεων για αυτά τα γεγονότα. Για να το κάνει αυτό προσφέρει στα components την δυνατότητα εγγραφής τους για συγκεκριμένους τύπους από events και για καθε event κρατά λίστες με τα εγγεγραμένα για αυτά components. Μέσω του Pub/Sub Module επιτυγχάνεται αποσυνδεδεμένη (decoupled) επικοινωνία μεταξύ των components του συστήματος. Τα components που δηλώνουν ενδιαφέρον για γεγονότα του συστήματος λαμβάνουν ενημερώσεις από το Pub/Sub Module χωρίς γνώση εκ των προτέρων των components που δημιουργούν τα events αυτά, ενώ αντίστροφα τα components που δημιουργουν κάποιο event το δηλώνουν στο Pub/Sub Module δίχως να γνωρίζουν τους τελικούς παραλήπτες.

Με βάση τα παραπάνω η διεπαφή του Pub/Sub Module ορίζει τις εξής συναρτήσεις:

- **subscribe:** Μέσω της συνάρτησης αυτής τα components του περιβάλλοντος-πλαισίου εκδηλώνουν ενδιαφέρον για συγκεκριμένα events του συστήματος, έτσι ώστε να λαμβάνουν αυτόματα ενημερώσεις κάθε φορά που καταγράφεται ένα τέτοιο event.

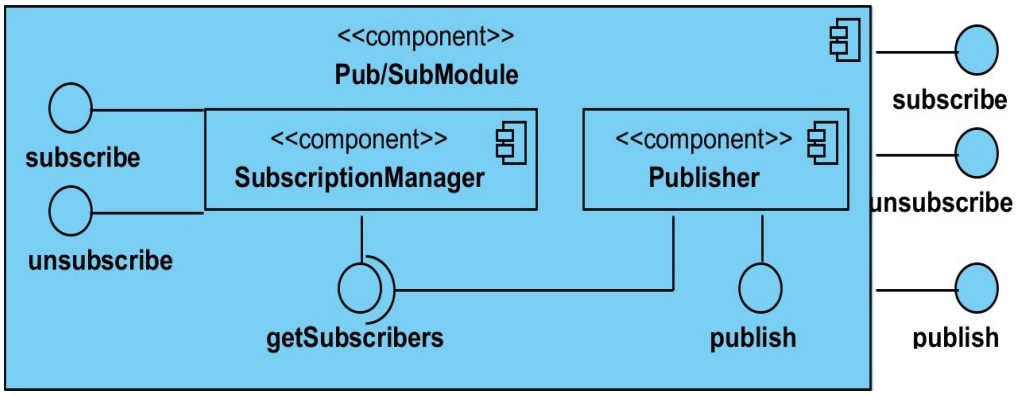
- **unsubscribe:** Μέσω της συνάρτησης αυτής τα components του περιβάλλοντος-πλαισίου δηλώνουν ότι θέλουν να πάψουν να λαμβάνουν ενημερώσεις για events για τα οποία είχαν δηλώσει ενδιαφέρον νωρίτερα.

- **publish:** Μέσω αυτής της συνάρτησης τα components του περιβάλλοντος-πλαισίου δηλώνουν κάποιο event του συστήματος, το οποίο στην συνέχεια αποστέλεται σε όλα τα ενδιαφερόμενα components.

Για να υλοποιήσει τις παραπάνω λειτουργίες περιέχει τα εξής sub-components:

- **SubscriptionManager:** Αυτό το component διατηρεί λίστες με τα components και τα events για τα οποία έχουν εκδηλώσει ενδιαφέρον. Παρέχει δυνατότητες εγγραφής και διαγραφής των components για συγκεκριμένα events. Είναι υπεύθυνο για την οργάνωση των συνδρομών (subscriptions) έτσι ώστε να γίνεται αποδοτικά η εύρεση των παραληπτών των γεγονότων του συστήματος. Η διεπαφή του περιλαμβάνει τις μεθόδους subscribe, unsubscribe και getSubscribers.

- **Publisher:** Ο Publisher ειναι υπεύθυνος για την ενημέρωση των ενδιαφερόμενων component. Λαμβάνει έναν νέο event μέσω της διεπαφής publish, βρίσκει τους παραλήπτες με την βοήθεια του SubscriptionManager και αποστέλει τα νέα events στους παραλήπτες αυτούς.



Σχήμα 3.3.6: Ψηφιδικό Διάγραμμα του Pub/Sub Module

3.3.6 Evaluator

Το component αυτό είναι υπεύθυνο για την επαλήθευση ή την άρνηση των υποθέσεων και των υποστόχων τους. Λαμβάνει ενημερώσεις από το Pub/Sub Module σχετικά με νέες υποθέσεις προς επαλήθευση καθώς και για την στρατηγική που πρέπει να ακολουθήσει για την επαλήθευση τους. Για να επαληθεύσει ή να αρνηθεί μια υπόθεση ή έναν στόχο χρησιμοποιεί τα γεγονότα παρακολούθησης και την δομή του δέντρου. Όπως έχουμε ήδη αναφέρει σε προηγούμενο κεφάλαιο και παρουσιάζουμε αναλυτικά στο επόμενο κάθε κόμβος μπορεί να περιέχει επισημειώσεις που ορίζουν ποια δεδομένα χρειάζεται να συλλεγούν για την απόδειξη ή την άρνηση του καθώς επίσης και τα μοτίβα (patterns) που πρέπει να ακολουθούν αυτά τα δεδομένα ώστε να καταστήσουν τον κόμβο αληθή ή ψευδή. Στην περίπτωση που ο κόμβος-στόχος που θέλουμε να αποδείξουμε περιέχει επισημειώσεις, μπορούμε να της χρησιμοποιήσουμε ώστε να ελέγχθούν τα δεδομένα παρακολούθησης με σκοπό την ανακάλυψη αυτών των μοτίβων. Στην περίπτωση χρήσης των επισημειώσεων για την απόδειξη ενός στόχου τα δεδομένα που χρειάζονται για την απόδειξή του μπορεί να μην είναι διαθέσιμα στο Blackboard, ο Evaluator ζητά την συλλογή τους. Αν ο κόμβος που θέλουμε να αποδείξουμε δεν περιέχει επισημειώσεις ο μόνος τρόπος απόδειξης του είναι είτε μέσω των συνεισφορών στον κόμβο αυτόν είτε μέσω της αποσύνθεσης του σε υποστόχους.

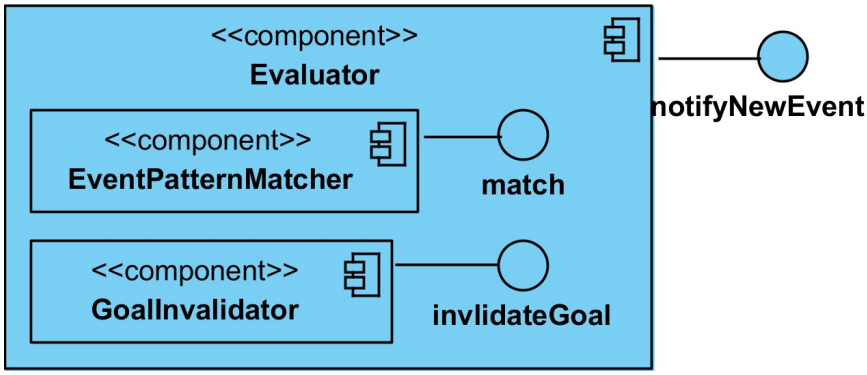
Σε αυτό το σημείο τίθεται το ερώτημα, τί γίνεται στην περίπτωση που αποδειχθεί ένας στόχος αληθής ή ψευδής. Για πόσο θεωρείται γνωστή η τιμή του και με ποια λογική μπορεί να ακυρωθεί.

Από τα παραπάνω προκύπτουν λογικά τα εξής sub-components για τον Evaluator:

- **EventPatternMatcher:** Ο EventPatternMatcher χρησιμοποιείται για να αποδείξει έναν στόχο με χρήση των επισημειώσεων των κόμβων και των γεγονότων παρακολούθησης. Η διεπαφή match προσπαθεί να ταιριάξει τα γεγονότα παρακολούθησης με ένα μόρφηματα (patterns) γεγονότων που ορίζονται στις επισημειώσεις των κόμβων.

- **GoalInvalidator:** Ο GoalInvalidator χρησιμοποιείται για να ακυρώσει την αληθότιμή ενός στόχου που έχουμε αποδείξει. Οταν αποδεικνύεται ότι ένας στόχος ή μια υπόθεση

είναι αληθής ή ψευδής η αληθοτιμή αυτή δεν μπορεί να ισχύει επ'απειρον. Δεδομένου ότι ο στόχος της διπλωματικής είναι η προσαρμοστική παρακολούθηση συστημάτων, δεν γίνεται παρακολούθηση όλων των γεγονότων του συστήματος. Ως εκ τούτου δεν είναι δυνατόν για το σύστημα να έχει πλήρη εικόνα για κάθε στόχο των δέντρων στόχων. Έτσι πρέπει να υπάρχει κάποια στρατηγική σύμφωνα με την οποία να ακυρώνεται η αληθοτιμή η οποία αποδείχθηκε για τον εκάστοτε στόχο. Ο GoalInvalidator υλοποιεί αυτήν την στρατηγική και μέσω της διεπαφής invalidateGoal αποφασίζει αν ένας στόχος πρέπει να ακυρωθεί ή όχι και τον ακυρώνει.

- 

Σχήμα 3.3.7: Ψηφιδικό Διάγραμμα του Evaluator

3.3.7 Alarmer

Ο Alarmer είναι υπεύθυνος για την παρακολούθηση των alarms και την ενεργοποίηση και απενεργοποίηση τους. Ενήμερώνεται μέσω του Pub/Sub Module για τα γεγονότα παρακολούθησης και για τα alarms που πρέπει να παρακολουθεί μέσω της διεπαφής notifyNewEvent. Τα components που χρησιμοποιεί είναι τα εξής:

- **EventPatternMatcher:** Ο EventPatternMatcher ελέγχει τα γεγονότα παρακολούθησης με σκοπό να ανακαλύψει συγκεκριμένα μορφήματα (patterns) των γεγονότων αυτών ικανά να ενεργοποιήσουν alarms του συστήματος.

- **AlarmManager:** Ο AlarmManager διαχειρίζεται όλη την διαδικασία παρακολούθησης των alarms ενεργοποίησης και απενεργοποίησης τους. Η διεπαφή του ορίζει τις συνάρτησεις addAlarmToMonitor και removeAlarm οι οποίες προσθέτουν και αφαιρούν ένα επιπλέον alarm προς παρακολούθηση. Για την ενεργοποιηση των alarms χρησιμοποιεί τον EventPatternMatcher ενώ για την απενεργοποίηση τους αξιοποιεί την στρατηγική που ορίζεται απο το AlarmDeactivationStrategy.

- **AlarmDeactivationStrategy:** Το component αυτό ορίζει την στρατηγική που εφαρμόζεται για την απενεργοποίηση των alarms.

3.4 Επικοινωνία μεταξύ των components του συστήματος

Σε αυτήν την ενότητα παρουσιάζουμε με την βοήθεια ακολουθιακών διαγραμμάτων την λειτουργία του συστήματος και τις αλληλεπιδράσεις μεταξύ των διάφορων components που το απαρτίζουν.

Όπως έχουμε ήδη αναφέρει η επικοινωνία μεταξύ των components που απαρτίζουν το σύστημα πραγματοποιείται μέσω του Pub/Sub Module. Τα components εκδηλώνουν ενδιαφέρον για συγκεκριμένα γεγονότα του συστήματος και ενεργοποιούνται μόνο όταν λάβουν ενημέρωση για κάποιο από τα γεγονότα που τα ενδιαφέρουν. Τα components αυτά επιτελούν τις βασικές υπο-λειτουργίες του συστήματος. Οι λειτουργίες αυτές όπως φαίνεται και απο το διάγραμμα δραστηριότητας της εισαγωγής είναι η συλλογή δεδομένων, η ανάλυση των δεδομένων, η επιλογή κάποιας στρατιγικής για την επαλήθευση μιας υπόθεσης η επαλήθευση μιας υπόθεσης και η αλλαγή του επιπέδου επίβλεψης σε περίπτωση που χρειάζεται. Κάθε μια από αυτές τις λειτουργίες πρέπει να εκτελείται παράλληλα με τις υπόλοιπες (εφόσον είναι διαθέσιμα όλα τα δεδομένα που μπορεί να χρειάζεται). Για παράδειγμα η συλλογή των δεδομένων είναι μια διαδικασία που πρέπει να λαμβάνει χώρα αδιάκοπα και να ενημερώνει τον Blackboard για τα νέα δεδομένα. Όσον αφορά τον Evaluator όπως αναφέραμε και στην προηγούμενη ενότητα στην προσπάθεια του να αποδείξει μια υπόθεση μπορεί να ζητήσει την συλλογή επιπλέον δεδομένων και να "παγώσει" μέχρι τα δεδομένα που ζήτησε να είναι διαθέσιμα. Έτσι λοιπόν και ο Evaluator πρέπει να "τρέχει" σε ξεχωριστό thread από το βασικό σύστημα γιατί αλλιώς θα σήμαινε το "πάγωμα" όλου του συστήματος μέχρι να γίνουν διαθέσιμα τα δεδομένα. Παράλληλα με τα υπόλοιπα components πρέπει να τρέχει και ο Alarmer καθώς η λειτουργία του είναι τελείως ανεξάρτητη από των υπόλοιπων components και μπορεί για παράδειγμα να ελέγχει για νέα alarms όσο ο Evaluator θα προσπαθεί να αποδείξει την ισχύ μιας υπόθεσης. Τα ίδια ισχύουν και για τον GoalSelector και για τον Monitoring Level Reasoner.

Με βάση λοιπόν τα παραπάνω κάθε ένα απο τα components του περιβάλλοντος πλαισίου τρέχει σε ξεχωριστό thread. Γιαυτό το λόγο παραθέτουμε ακολουθιακά διαγράμματα για κάθε μια από τις βασικές λειτουργίες του συστήματος που φαίνονται στο διάγραμμα δραστηριότητας της πρώτης ενότητας. Αυτό διευκολύνει την περιγραφή των επιμέρους λειτουργιών.

3.4.1 Ακολουθιακό Διάγραμμα Συλλογής Δεδομένων

Στο διάγραμμα που ακολουθεί περιγράφεται η διαδικασία συλλογής των δεδομένων. Περιλαμβάνει την συλλογή των γεγονότων παρακολούθησης, την κανονικοποιηση τους και την ενημέρωση του Blackboard και των component του περιβάλλοντος-πλαισίου για τα νέα δεδομένα. Αναλυτικά τα βήματα είναι τα εξής:

1. **newData**: Ο Collector λαμβάνει νέα δεδομένα από τον πόρο που παρακολουθεί.

2. **filterData**: Ο Collector φιλτράρει τα δεδομένα αυτά με βάση τις ανάγκες παρακολούθησης έτσι ώστε να στείλει προς το Blackboard μόνο εκείνα που χρείαζονται.

   Σε περίπτωση που τα δεδομένα χρειάζονται στα components του περιβάλλοντος-πλαισίου ακολουθούν οι εξής ενέργειες

3. **normalize**: Ο Collector καλεί τον DataMediator για να κανονικοποιήσει τα δεδομένα και να τα αποθηκεύσει στον Blackboard.

4. **update**: Ο DataMediator αποθηκεύει τα νέα δεδομένα στον Blackboard.

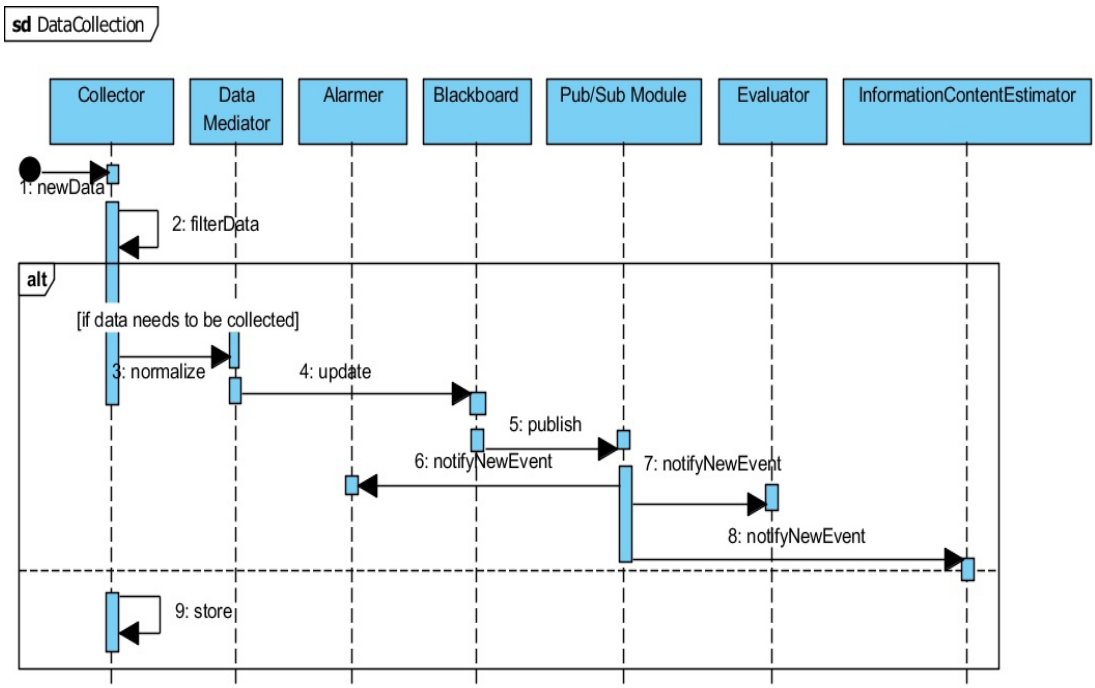
5. **publish**: Ο Blackboard στέλνει στο Pub/Sub Module τα νέα δεδομένα που μόλις έλαβε ωστε αυτό με την σειρά του να ενημερώσει όλους τους ενδιαφερόμενους για το νέο event.

6. **notifyNewEvent**: Το Pub/Sub Module ενημερώνει τον Alarmer για τα νέα δεδομένα παρακολούθησης.

7. **notifyNewEvent**: Το Pub/Sub Module ενημερώνει τον Evaluateor για τα νέα δεδομένα παρακολούθησης.

8. **notifyNewEvent**: Το Pub/Sub Module ενημερώνει τον InformationContentEstimator για τα νέα δεδομένα παρακολούθησης.

9. **store:** Στην περίπτωση που τα δεδομένα που συλλέχθηκαν δεν χρειάζονται στα components του περιβάλλοντος-πλαισίου για την επαλήθευση κάποιας υπόθεσης αποθηκεύονται προσωρινά στον Collector σε περίπτωση που χρειαστούν αργότερα. Αν δεν χρειαστούν τότε διαγράφονται.



Σχήμα 3.4.1: Ακολουθιακό Διάγραμμα για τηνΣυλλογή Δεδομένων Παρακολούθησης

3.4.2 Ακολουθιακό Διάγραμμα Ενεργοποίησης Alarm

Στο διάγραμμα που ακολουθεί περιγράφεται η διαδικασία ενεργοποίησης των Alarms. Αναλυτικά τα βήματα είναι τα εξής:

1. **notifyNewEvent**: Ο Alarmer ενημερώνεται για τα νέα δεδομένα που συλλέχθηκαν.

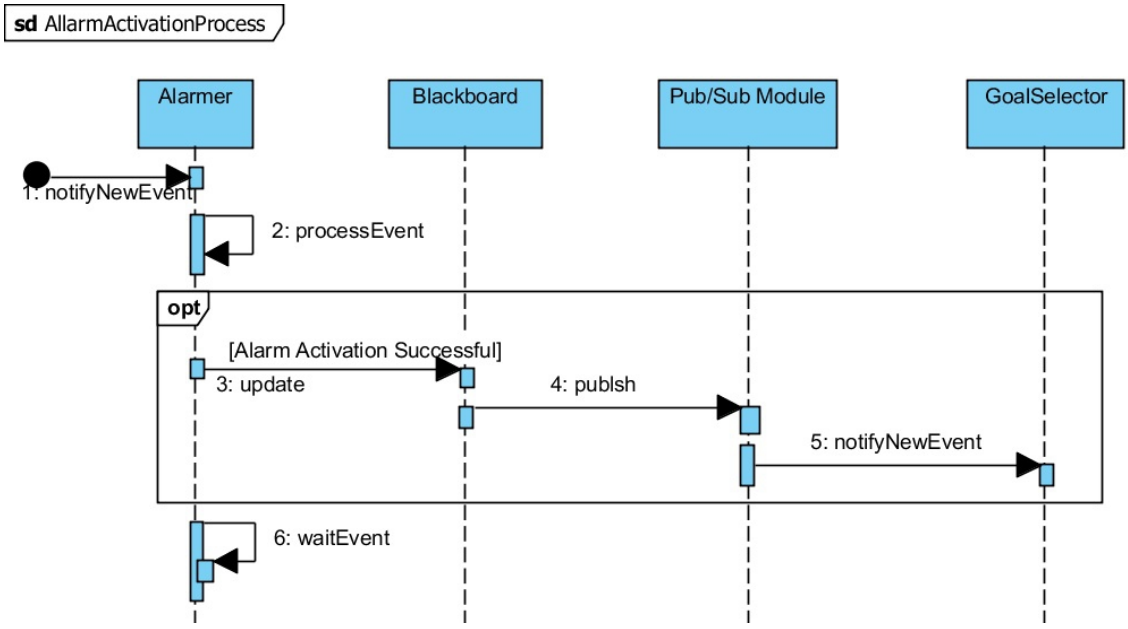
2. **processEvent**: Ο Alarmer επεξεργάζεται τα δεδομένα που μόλις έλαβε και τα συνδυάζει με προγενέστερα δεδομένα με σκοπό να ανακαλύψει μοτίβα που ενεργοποιούν κάποιο alarm.

   Στην περίπτωση που δεν βρεί κάποιο τέτοιο μοτίβο περιμένει μέχρι να του έρθουν νέα δεδομένα (ενέργεια 6). Στην περίπτωση που βρέθει κάποιο τέτοιο μοτίβο εκτελεί τις εξής ενέργειες.

3. **update**: Ο Alarmer ενεργοποιεί ένα νέο Alarm και ενημερώνει τον Blackboard για την ενεργοποίηση του Alarm αυτού.

4. **publish**: Ο Blackboard στέλνει στο Pub/Sub Module ένα μήνυμα με το Alarm που μόλις ενεργοποιήθηκε ωστε αυτό με την σειρά του να ενημερώσει όλα τα ενδιαφερόμενα components για την ενεργοποίηση του Alarm αυτού.

5. **notifyNewEvent**: Το Pub/Sub Module ενημερώνει τον GoalSelector για την ενεργοποίηση του νέου Alarm.



Σχήμα 3.4.2: Ακολουθιακό Διάγραμμα για την Ενεργοποίηση κάποιου Alarm

3.4.3 Ακολουθιακό Διάγραμμα σχεδιασμού πλάνου επαλήθευσης υποθέσεων

Στο διάγραμμα που ακολουθεί περιγράφεται η διαδικασία εύρεσης ενός πλάνου για την επαλήθευση των ενεργών υποθέσεων. Την διαδικασία αυτή αναλαμβάνει ο GoalSelector με την βοήθεια των component Strategy και του SatSolver. Η διαδικασία ξεκινά με την ενεργοποίηση καποιου Alarm. Αναλυτικά τα βήματα είναι τα εξής:

1. **notifyNewEvent**: Ο GoalSelector ενημερώνεται για την ενεργοποίηση κάποιου alarm. Υπολογίζει την στρατηγική για την επαλήθευσητων υποθέσεων που σχετίζονται με τα

ενεργά alarms. Για να το κάνει αυτό χρειάζεται να γνωρίζει τις αληθοτιμές των υποθέσεων που έχουν αποδειχθεί καθώς επίσης και το σύνολο των ενεργών alarms.

2. **query**: Ο GoalSelector ζητά την κατάσταση του Blackboard απο την οποία θα χρειαστεί τις αληθοτιμές των υποθέσεων που έχουν αποδειχθεί καθώς επίσης και το σύνολο των ενεργών alarms.

3. **generateCNFRules:** Ο GoalSelector μετατρέπει τα δέντρα στόχων μαζί με τις αληθοτιμές των κόμβων που είναι ήδη γνωστές με την βοήθεια του GoalModelToCNFConverter**.** Η έκφραση CNF που προκύπτει χρησιμοποιείται ως είσοδος στο Sat Solver.

4. **solve**: Ο GoalSelector ζητά απο τον SatSolver όλους τους δυνατούς τρόπους με τους οποιους είναι μπορεί να αποδειχθεί η ισχύς των υποθέσεων που σχετίζονται με τα ενεργά alarms. Οι αληθοτιμές των υποθέσεων που έχουν ήδη αποδειχθεί χρησιμοποιούνται ως βάση γνώσης για την εύρεση των τρόπων απόδειξης των υποθέσεων που μένει να αποδειχθούν.

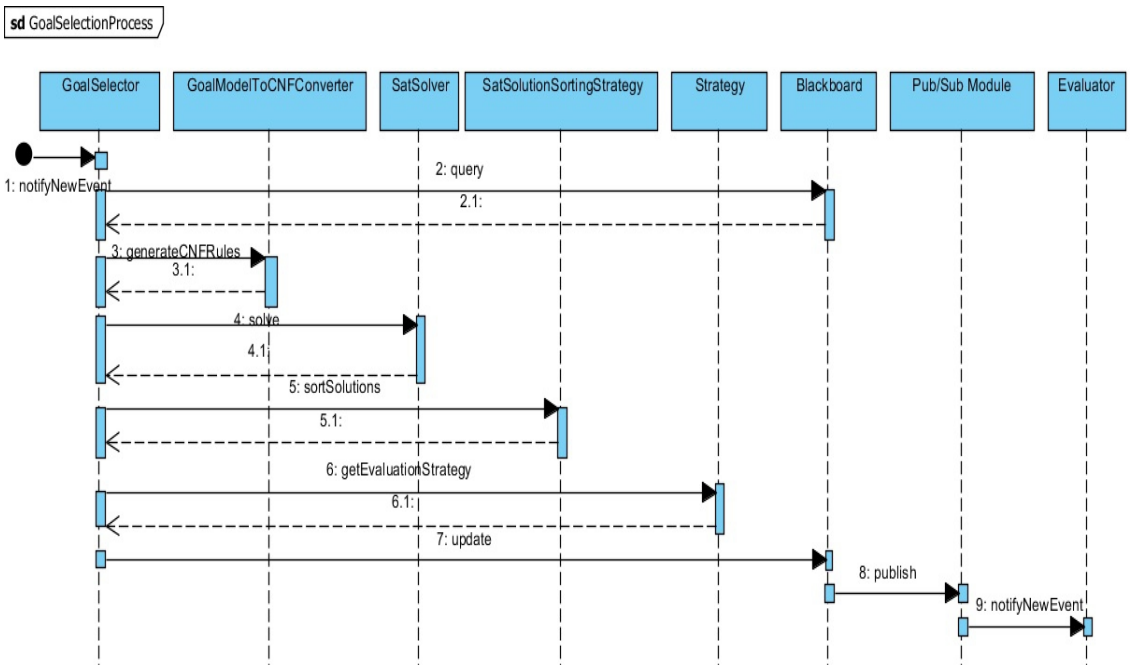
5. **sortSolutions**: Απο το προηγούμενο βήμα είναι γνωστοί οι όλοι οι δυνατοί τρόποι με τους οποιους μπορεί να αποδειχθεί η ισχύς των υποθέσεων που σχετίζονται με τα ενεργά alarms. Σε αυτό το βήμα ο GoalSelector ζητά απο τον SatSolutionSortingStrategy να ταξινομήσει τους τρόπους αυτούς ώστε χρησιμοποιηθεί η καταλληλότερη λύση και εαν αποτύχει να χρησιμοποιηθε η επόμενη κ.ο.κ.

6. **getEvaluationStrategy**: Όπως έχουμε ήδη αναφέρει ενδέχεται να υπάρχουν πολλές υποθέσεις προς επαλήθευση. Σε αυτήν την περίπτωση χρειάζεται ένα πλάνο για την επαλήθευση τους. Το πλάνο αυτό μπορεί να ορίζει την σειρά που θα αποδειχθούν οι υποθέσεις, το αν θα επαληθευτούν παράλληλα ή σειριακά κ.ο.κ. Ο GoalSelector ζητά απο το component Strategy να αποφασίσει τον τρόπο με τον οποίο θα επαληθευτούν οι ενεργές υποθέσεις.

7. **update**: Ο GoalSelector ενημερώνει τον Blackboard για τις νέες υποθέσεις προς επαλήθευση καθώς επίσης και για τον τρόπο επαλήθευσης τους.

8. **publish**: Ο Blackboard στέλνει στο Pub/Sub Module ένα μήνυμα με τα νέα δεδομένα που μόλις έλαβε ωστε αυτό με την σειρά του να ενημερώσει όλους τους ενδιαφερόμενα για τον συγκεκριμένο τύπο event, components.

9. **notifyNewEvent**: Το Pub/Sub Module ενημερώνει τον Evaluator με την ενημερωμένη λίστα με τις υποθέσεις προς επαλήθευση, καθώς επίσης και τον τρόπο επαλήθευσης τους.



Σχήμα 3.4.3: Ακολουθιακό Διάγραμμα για την Εύρεση Πλάνου Επαλήθευσης των Υποθέσεων

3.4.4 Ακολουθιακό Διάγραμμα Επαλήθευσης Υποθέσης

Στο διάγραμμα που ακολουθεί περιγράφεται η διαδικασία επαλήθευση μιας υπόθεσης. Την διαδικασία αυτή αναλαμβάνει ο Evaluator. Αναλυτικά τα βήματα είναι τα εξής:

1. **notifyNewEvent**: Ο Evaluator ενημερώνεται για μια νέα υπόθεση προς επαλήθευση καθώς και για το πλάνο που πρέπει να ακολουθηθεί για την επαλήθευση της.

2. **proveHypothesis:** Ο Evaluator προσπαθεί να επαληθεύσει την υπόθεση αυτή είτε μέσω της δομή του δέντρου είτε μέσα από τα γεγονότα παρακολούθησης. Αυτή η προσπάθεια έχει τα εξής πιθανά αποτελέσματα:
    - Την επαλήθευση ή άρνηση της υπόθεσης.
    - Την απόδειξης άλλων στόχων που χρειάζονται για την επαλήθευση της υπόθεσης και κατόπιν την επαληθευση της υπόθεσης.
    - Την αλλαγή του επιπέδου παρακολούθησης σε περίπτωση που τα δεδομένα δεν επαρκούν για την επαλήθευση της υπόθεσης.

    Στην πρώτη περίπτωση τα βήματα που ακόλουθούνται είναι τα εξής:

    3. **update:** Ο Evaluator ενημερώνει τον Blackboard για την επαλήθευση και ή την άρνηση μιας υπόθεσης.

    4. **publish:** Ο Blackboard δημοσιεύει το γεγονός της επαλήθευσης ή της άρνησης της υπόθεσης αυτής έτσι ώστε να ενημερωθούν αυτόματα όλα τα ενδιαφερόμενα components.

    Στην δεύτερη περίπτωση τα βήματα που ακόλουθούνται είναι τα εξής:

    5. **update:** Ο Evaluator ενεργοποιεί ένα νέο ψευδο-alarm για το κόμβο-στόχο που

θέλει να αποδείξει και ενημερώνει τον Blackboard έτσι ώστε ο GoalSelector να του δώσει ένα πλάνο για την απόδειξη του στόχου αυτού.

6. **publish:** Ο Blackboard δημοσιεύει το νέο alarm στο Pub/Sub Module ώστε να ενημερωθούν όλα τα ενδιαφερόμενα components.

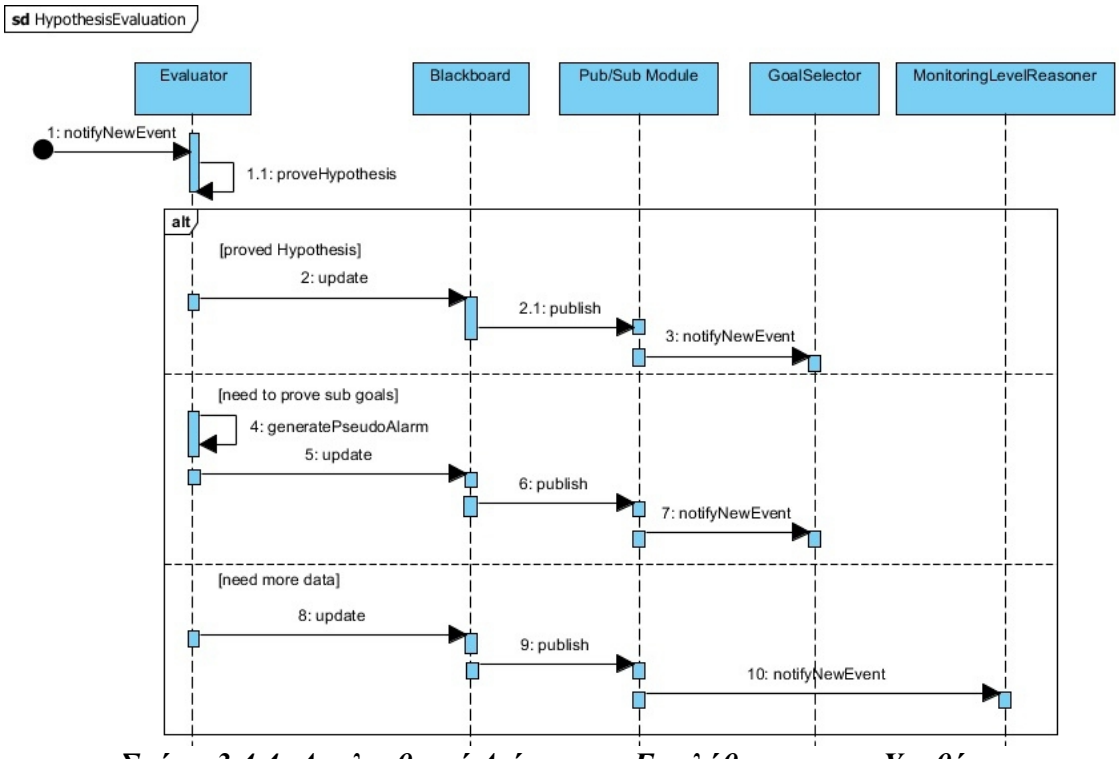
7. **notifyNewEvent:** Το Pub/Sub Module ενημερώνει τον GoalSelector για το νέο alarm.

Στην τρίτη περίπτωση τα βήματα που ακόλουθούνται είναι τα εξής:

8. **update:** Ο Evaluator ενημερώνει τον Blackboard για την ανάγκη αλλαγής του επιπέδου παρακολούθησης.

9. **publish:** Ο Blackboard δημοσιεύει την αναγκη για αλλαγή του επιπέδου παρακολούθησης στο Pub/Sub Module έτσι ώστε να ενημερωθούν όλα τα ενδαφερόμενα components.

10. **notifyNewEvent:** Το Pub/Sub Module ενημερώνει τον MonitoringLevelReasoner για την ανάγκη αλλαγής του επιπέδους παρακολούθησης.



Σχήμα 3.4.4: Ακολουθιακό Διάγραμμα Επαλήθευσης των Υποθέσεων

3.4.5 Ακολουθιακό Διάγραμμα Αυξησης Επιπέδου Παρακολούθησης

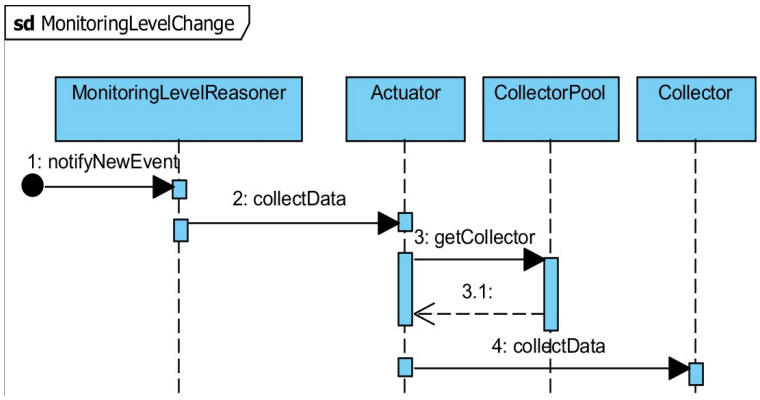
Στο διάγραμμα που ακολουθεί περιγράφεται η διαδικασία αλλαγής του επιπέδου παρακολούθησης. Την διαδικασία αυτή αναλαμβάνει ο MonitoringLevelReasoner και εκκινεί ύστερα από αιτημα του Evaluator για δεδομένα που δεν συλλέγονται. Αναλυτικά τα βήματα είναι τα εξής:

1. **notifyNewEvent**: Ο MonitoringLevelReasoner ενημερώνεται για την ανάγκη συλλογής περισσότερων δεδομένων. Όπως είδαμε είδαμε παραπάνω τέτοιου είδους μηνύματα παράγονται απο τον Evaluator στην προσπάθειά του να αποδείξει την ισχύ μιας υπόθεσης.

2. **collectData**: Ο MonitoringLevelReasoner αναθέτει στον Actuator την συλλογή των νέων δεδομένων.

3. **getCollector**: Ο Actuator ψάχνει στην δεξαμενή των Collectors να βρει τον κατάλληλο Collector για την συλλογή των νέων δεδομένων.

4. **collectData**: Ο Actuator αναθέτει στον κατάλληλο Collector την συλλογή των νέων δεδομένων.



Σχήμα 3.4.5: Ακολουθιακό Διάγραμμα για την Αλλαγή του Επιπέδου Παρακολούθησης

Κεφάλαιο 4: Domain Model

4.1 Εισαγωγή

Στο κεφάλαιο αυτό περιγράφουμε το εννοιολογικό μοντέλο (domain model) των δέντρων στόχων του προτεινόμενου περιβάλλοντος-πλαισίου. Το μοντέλο περιγράφει τις οντότητες που συνθέτουν τα δέντρα στόχων καθώς επίσης και τις σχέσεις μεταξύ τους. Το μοντέλο που ακόλουθεί, ακολουθεί την προδιαγραφή MOF και αποτέλεσε την βάση για την παραγωγή του κώδικα του περιβάλλοντος-πλαισίου με την βοήθεια του EMF.

4.2 Εννοιολογικό Μοντέλο Δέντρων Στόχων

Το εννοιολογικό μοντέλο των δέντρων στόχων κατασκευάστηκε για να καλύψει τις απαιτήσεις του παρόντος περιβάλλοντος-πλαισίου, ορίζει τις διάφορες υποθέσεις με τη μορφή Ή / ΚΑΙ Δεντρων Στόχων και επεκτείνειται με τις έννοιες των συνεισφορών και των επισημειώσεων, όπως συζητήθηκε στο κεφάλαιο 2. Αυτή η δομή μοντελοποίηση επιλέχθηκε λόγω των χαρακτηριστικών της και των πλεονεκτημάτων που έχει. Πιο συγκεκριμένα, παρέχει ευελιξία όσον αφορά τα μέσα μοντελοποίησης κάθε υπόθεσης, ευκολία ορισμού νεων υποθέσεων, και ευκολία στην ανάλυση των υποθέσεων και τον συμπερασμό γι αυτές. Επιπλέον, τόσο οι αποσυνθέσεις Ή/ΚΑΙ που μοιράζονται όλοι οι κόμβοι-στόχοι του δέντρου εξ'ορισμού όσο και οι συνεισφορές μεταξύ των κόμβων, επιτρέπουν άμεσο και εύκολο μετασχηματισμό των δέντρων στόχων σε εκφράσεις CNF, οι οποίες απαιτούνται από τον SAT4j για να παράξει τους δυνατούς τρόπους επαλήθευσης των υποθέσεων.

Στο σημείο αυτό περιγράφουμε μία προς μία τις κλάσεις του εννοιολογικού μοντέλου των δέντρων στόχων, ενώ στο τέλος του κεφαλαίου δίνεται το διάγραμμα κλάσεων του μοντέλου αυτού.

4.2.1 Κλάση HypoType

Η κλάση *HypoType* αντιπροσωπεύει τον τύπο στον οποίο ανοίκει μια υπόθεση. Η ιδιότητα Name τύπου String που ανήκει στην κλάση *HypoType* χρησιμοποιείται για να διαχωρήσει τα διάφορα στιγμιότυπα της κλάσης και δείχνει τον τύπο της υπόθεσης για τον οποίο εκτελείται η ανάλυση. Η κλάση *HypoType* αποτελείται από πολλές υποθέσεις όπως φαίνεται και από την συσχέτιση με την κλάση *Hypothesis* η οποία έχει πολλαπλότητα ένα προς πολλά. Από αυτό προκύπτει η ύπαρξη της ιδιότητας *+hypothesis* τύπου *Hypothesis* και πολλαπλότητας 1..*.

4.2.2 Κλάση Hypothesis

Κάθε στιγμιότυπο της κλάσης Hypothesis αντιπροσωπεύει κάθε υπόθεση που περιλαμβάνεται σε έναν τύπο υποθέσεων. Ξεχωρίζει απο τις υπόλοιπες υποθέσεις μέσω του πεδίου *Name* τύπου και παίζει το ρόλο του πρωτεύοντος κλειδιού της κλάσης. Λόγω της συσχέτισης με την κλάση HypoType, που περιγράφηκε πιο πάνω, περιλαμβάνει το επιπλέον πεδίο *+type* τύπου *HypoType* και πολλαπλότητας 1 το οποίο δείχνει τον τύπον στον οποίο αντιστοιχεί κάθε αντικείμενο της κλάσης Hypothesis. Κάθε στιγμιότυπο της κλάσης *HypoThesis* περιλαμβάνει μόνο ένα δέντρο, όπως φαίνεται απο την σχέση σύνθεσης με την κλάση *GoalTree*, μία συσχέτιση η οποία προκύπτει άμεσα ως αποτέλεσμα του γεγονότος ότι το σύστημα που σχεδιάζουμε στοχεύει στην επαλήθευση ή την άρνηση μιας υπόθεσης μέσω της αναπαράστασης της με δέντρα στόχων. Από αυτό είναι ξεκάθαρο ότι μόνο ένα δέντρο στόχων αντιστοιχεί σε μία υπόθεση. Η κλάση επίσης συσχετίζεται με την κλάση Verifier με μία σχέση πολλά προς ένα. Απο την σχέση αυτή η κλάση Hypothesis λαμβάνει την ιδιότητα +verifier πολλαπλότητας 1. Τέλος η κλάση *Hypothesis* συσχετίζεται με την κλάση *Alarm* από την οποία λαμβάνει την ιδιότητα +alarms πολλαπλότητας 1.

4.2.3 Κλάση GoalTree

Η κλάση GoalTree αντιπροσωπεύει την δομή ενός δέντρου στόχων, το οποίο είναι η βασική δομή που χρησιμοποιείται για να μοντελοποιήσει μια υπόθεση με ευέλικτο τρόπο. Αυτή η κλάση έχει την ιδιότητα *idname* τύπου String η οποία λειτουργεί σαν πρωτεύον κλειδί της της κλάσης για να διαχωρίσει τα διαφορα στιγμιότυπα της κλάσης. Η κλάση GoalTree, επιπλέον, έχει μία συσχέτιση σύνθεσης με την κλάση Hypothesis απο την οποία λαμβάνει την ιδιότητα *+hypothesis* πολλαπλότητας 1.

4.2.4 Κλάση GoalNode

Η κλάση GoalNode αντιπροσωπεύει τους κόμβους του δενδρικού γράφου του μοντέλου στόχων. Όπως σε κάθε δέντρο, έτσι και εδώ, κάθε κόμβος του δέντρου, εκτός από την ρίζα του του, έχει ένα κόμβο πατέρα και όλοι οι κόμβοι εκτός από το φύλλα έχουν κόμβους παιδιά. Οι σχέσεις αυτές περιγράφονται από την συσχέτιση με την κλάση *DecompositionGoal* από την οποία λαμβάνει *+parent* με πολλαπλότητα 1. Η κλάση *GoalNode* έχει επίσης τις ιδιότητες *Name* τύπου String και *ID* τύπου Integer. Η ιδιότητα Name χρησιμοποιείται για την κατασκευή των κατηγορημάτων (predicates) ενώ το ID χρησιμποιείται ως ένα ασφαλές πρωτεύον κλειδί για να διαχωρίσει τους διάφορους κόμβους και να αποτρέψει πιθανά ψευδώνυμα (name aliases) μεταξύ διαφορετικών δέντρων στόχων. Η κλάση *GoalNode* συσχετίζεται με την κλάση Contribution με τις συσχετίσεις from και to οι οποίες είναι και οι δύο πολλαπλότητας ένα προς πολλά και από τις οποίες λαμβάνει τις ιδιότητες *+inContributions* και *+outContributions* οι οποίες δηλώνουν αντίστοιχα τις συνεισφορές που συνεισφέρουν στον κόμβο αυτό και τις συνεισφορές στις οποιες συνεισφέρει ο κόμβος αυτός. Η κλάση *GoalNode* παρουσιάζει μία σχέση σύνθεσης με την κλάση *GoalTree* απο την οποία λαμβάνει από την ιδιότητα *+tree* τύπου *GoalTree* με πολλαπλότητα 1, η οποια ορίζει κάθε κόμβος-στόχος σε ποιο δέντρο στόχων ανοίκει. Τέλος η κλάση *GoalNode* παρουσιάζει μία συσχέτιση σύνθεσης με την κλάση *AnnotationContainer* από την οποία λαμβάνει την ιδιότητα *+annotContainer* τύπου πολλαπλότητας 1 καθώς κάθε κόμβος-στόχος έχει ένα *AnnotationContainer* για να αποθηκεύει απαραιτητες πληροφορίες για την απόδειξη του. Σημειώνεται ότι, όπως φαίνεται και στο σχήμα 4.1, η κλάση *GoalNode* εμφανίζει μια σχέση κληρονομικότητας με τις κλάσεις *AtomicGoal* και *DecompositionGoal* οι οποίες εξηγούνται παρακάτω. Η κλάση *GoalNode*,

τέλος, έχει ένα πεδίο +*goalVerifier* τύπου G*oalVerifier* το οποίο λαβμάνει απο την συσχέτιση του με την κλάση *GoalVerifier* η οπόια χρησιμοποιείται για τον χειρισμό των επισημειώσεων και την απόδειξη του κόμβου-στόχου. Συσχετίζεται επίσης με την κλάση *Alarm* πολλαπλότητας ένα προς πολλά. Μέσω της σχέσης του με την κλάση *GoalState* λαμβάνει την ιδιότητα +*goalState* πολλαπλότητας 1. Τέλος συσχετίζεται με την κλάση *CNFExpression* από την οποία λαμβάνει την ιδιότητα +*cnf* πολλαπλότητα 1 και η οποία αντιστοιχεί στην cnf εκφραση που αντιστοιχεί στον κόμβο αυτόν.

4.2.5 Κλάση AtomicGoal

Η κλάση AtomicGoal είναι μια υποκλάση της κλάσης GoalNode και ως εκ τούτου, κληρονομεί τις ιδιότητες και της λειτουργίες της υπερκλάσης GoalNode. Η κλάση AtomicGoal αντιπροσωπεύει τους στόχους που είναι φύλλα στο δέντρο στόχων, δηλαδή τους στόχους που δεν μπορούν να αναλυθούν περαιτέρω σε υποστόχους.

4.2.6 Κλάση GoalDecomposition

Η κλάση DecompositionGoal είναι η δεύτερη υποκλάση της κλάσης *GoalNode* και κληρονομεί και αυτή όλες τις ιδιότητες και λειτουργίες της. Η κλάση *DecompositionGoal* αντιπροσωπεύει όλα τα στιγμιότυπα της κλάσης *GoalNode* τα οποία είναι σύνθετοι στόχοι, δηλαδή στόχοι που έχουν παιδιά-στόχους τα οποία παρουσιάζουν μεταξύ τους σχέσεις And ή Or. Για αυτόν τον λόγο η κλάση DecompositionGoal έχει δύο υποκλάσεις, την κλάση *ANDDecomposition και την κλάση ORDecomposition* έτσι ώστε να διαχωρήσει τις δύο περιπτώσεις σύνθετων κόμβων-στόχων και ως εκ τουτου να επιτρέψει την ξεχωριστή διαχείριση κάθε τύπου κόμβου-στόχου. Η κλάση *GoalDecomposition* συσχετίζεται με την κλάση *GoalNode* μέσω μίας σχέσης σύνθεσης. Από την συσχέτιση αυτή φαίνεται και τρόπος ορισμού των δέντρων στόχων. Δηλαδή για την δημιουργία ενός σύνθετου κόμβου απαιτείται να έχουν δημιουργηθεί πρώτα τα παιδιά του. Από την συσχέτιση του *GoalDecomposition* με την κλάση *GoalNode* η κλάση *GoalDecomposition* λαμβάνει την ιδιότητα +*children* η οποία έχει πολλαπλότητα 2..* και ορίζει τους κόμβους-παιδιά στους οποίους αποσυντίθεται ένα στιγμιότυπο της κλάσης *GoalDecomposition*.

4.2.7 Κλάση GoalState

Η κλάση *GoalState* ορίζει τις καταστάσεις στις οποίες μπορεί να βρεθεί ένας κόμβος-στόχος. Ορίζεται ως μία απαρίθμηση με δυνατές τιμές *TRUE, FALSE* και *UNKNOWN*. Η κατάσταση *GoalState.TRUE* αντιστοιχεί στην περίπτωση που έχει αποδειχθεί ο κόμβος-στόχος, η κατάσταση *GoalState.FALSE* στην περίπτωση που έχει αποδειχθεί η άρνηση του κόμβου-στόχου και η κατάσταση *GoalState.UNKNOWN* στην περίπτωση που δεν γνωρίζουμε αν ο κόμβος-στόχος είναι αληθής ή ψευδής.

4.2.8 Κλάση Contribution

Η κλάση *Contribution* αναπαριστά τις συνεισφορές μεταξύ των στόχων του μοντέλου δέντρων στόχων. Οι συνεισφορές αυτές έχουν έναν τύπο ο οποίος δίνεται μέσα απο την συσχέτιση τη κλάσης *Contribution* με την κλάση *ContributionType*. Από την συσχέτιση αυτή λαμβάνει την

ιδιότητα *+contrType* τύπου *ContributionType* με πολλαπλότητα 1. Τέλος κάθε στιγμιότυπο της κλάσης *Contribution* συσχετίζεται με δύο στιγμιότυπα της κλάσης *GoalNode*. Από αυτές τις συσχετίσεις λαμβάνει τις ιδιότητες *+from* και *+to* οι οποίες αντιστοιχούν στον κόμβο που συνεισφέρει και στον κόμβο στον οποίο γίνεται η συνεισφορά αντίστοιχα.

4.2.9 Κλάση ContributionType

Η κλάση *ContributionType* ορίζει τους τύπους συνεισφορών που χρησιμοποιούνται μεταξύ των κόμβων των δέντρων στόχων. Η κλάση αυτή ορίζεται ως μία απαρίθμηση με τέσσερις δυνατές τιμές. Έτσι μια συνεισφορά μπορεί να έχει έναν απο τους εξής τύπους:

- PPS
- MMS
- PPD
- MMD

4.2.10 Κλάση GoalVerifier

Η κλάση Goal*Verifier* χρησιμοποιείται για την επαλήθευση ενός στόχου που περιέχει επισημειώσεις. Ελέγχει τις επισημειώσεις του κόμβου-στόχου και αποφαίνεται για την ισχύ του στόχου αυτού. Χρησιμοποιεί την κλάση VerificationStrategy για την επαλήθευση του στόχου έτσι ώστε να επιτραπεί η χρήση διαφορετικών στρατηγικών για διαφορετικούς τύπους επισημειώσεων. Τέλος ορίζει την συνάρτηση verify() η οποία επαληθεύει τον στόχο.

4.2.11 Κλάση Verifier

Η κλάση *Verifier* χρησιμοποιείται για την επαλήθευση των υποθέσεων. Χρησιμοποιεί την κλάση Reasoner για την επαλήθευση των υποθέσεων. Συσχετίζεται με την κλάση Reasoner απο την οποία λαμβάνει την ιδιότητα *+reasoner* πολλαπλότητας 1 και ορίζει την συνάρτηση *verifyHyp()*, η οποία επαληθεύει μια υπόθεση.

4.2.12 Κλάση Alarm

Η κλάση *Alarm* αντιπροσωπεύει τα alarms που ενεργοποιούν μια υπόθεση. Ένα alarm ορίζει ένα σύνολο δεδομένων που είναι απαραίτητα για την ενεργοποίηση του alarm αυτού και ένα σύνολο μορφημάτων (pattern) των δεδομένων αυτών. Τα μορφήματα αυτά ενεργοποιούν το alarm αυτό. Το σύνολο των δεδομένων που χρειάζονται για την ενεργοποίηση του alarm όριζουν στην ουσία το σύνολο των γεγονότων που πρέπει να παρακολουθούνται για να είναι ικανή η ενεργοποιηση του alarm αυτού. Η παρακολούθηση των γεγονότων αυτών δεν οδηγεί απαραίτητα στην ενεργοποιηση του alarm. Για να γίνει αυτό πρέπει να ανακαλυφθούν κάποιο pattern των δεδομένων αυτών που ενεργοποιεί το alarm. Έτσι προκύπτουν οι εξής συσχετίσεις για την κλάση *Alarm*:

- Μία συσχέτιση με την κλάση *ActivationPattern* πολλαπλότητας ένα προς πολλά απο την οποία λαμβάνει την ιδιότητα *+pattern* πολλαπλότητας 1..*. Έτσι μπορεί να υπάρχουν πολλά patterns που ενεργοποιούν ένα alarm.

- Μία συσχέτιση με την κλάση *AlarmActivator* από την οποία λαμβάνει την ιδιότητα *+alarmActivator* πολλαπλότητας 1.
- Μία συσχέτιση με την κλάση NeededData από την οποία λαμβάνει την ιδιότητα *+neededData* πολλαπλότητας 1..*.
- Μία συσχέτιση με την κλάση Hypothesis από την οποία λαμβάνει την ιδιότητα *+hypothesis* πολλαπλότητας 1..*.

4.2.13 Κλάση ActivationPattern

Η κλάση ActivationPattern αντιπροσωπεύει μορφήματα των γεγονότων παρακολούθησης που σηματοδοτούν την ενεργοποίηση ενός Alarm.

4.2.14 Κλάση AlarmActivator

Η κλάση AlarmActivator κάνει ένα ταίριασμα των γεγονότων παρακολούθησης με τα ActivationPattern έτσι ώστε να ανακαλύψει μορφήματα μέσα στα γεγονότα παρακολούθησης και να ενεργοποιήσει τα κατάλληλα alarms. Η κλάση αυτή αντιπροσωπεύει το component Alarmer.

4.2.15 Κλάση NeededData

Η κλάση *NeededData* αντιπροσωπεύει το σύνολο των δεδομένων που είναι απαραίτητα για την ενεργοποίηση ενός alarm. Η κλάση αυτή έχει ως πεδία το *eventSource* τύπου Integer που αποτελεί το διαχωριστικό γνώρισμα των διαφόρων πηγών συλλογής δεδομένων, το eventType τύπου EventType που ορίζει τον τύπο των γεγονότων που πρέπει να συλλεγού, και το attributes τύπου List<String> που ορίζει τα attributes των γεγονότων παρακολούθησης που ενδιαφέρουν (πχ τον τύπο του γεγονότος και την χρονοσφραγίδα του).

4.2.16 Κλάση Reasoner

Η κλάση Reasoner αντιπροσωπεύει αντικείμενα που αναλαμβάνουν την συλλογιστική διαδικασία που είναι απαραίτητη για την επαλήθευση μιας υπόθεσης. Στην περίπτωση μας την συλλογιστική διαδικασία αναλαμβάνει ο SAT4j που θεωρείται υποκλάση της κλάσης SATSolver που είναι υποκλάση της κλάσης Reasoner.

4.1.17 Κλάση CNFExpr

Η κλάση *CNFExpr* αντιπροσωπεύει την συζευκτική κανονική μορφή, στην οποία η μετατρέπει ένας κόμβος-στόχος προκειμένου για τον έλεγχο και την συλλογιστική που θα πραγματοποιηθεί για την απόδειξη του στόχου αυτού. Αυτός είναι ο λόγο για τη συσχέτιση του με τις κλάσεις *SATSolver* και *GoalNode* οι οποίες είναι και οι δύο ένα προς ένα.

4.1.18 Κλάση GoalToCNFConverter

Η κλάση GoalToCNFConverter χρησιμοποιείται για την μετατροπή των δέντρων στόχων σε κανονική συζευκτική μορφή. Το κάνει αυτό μέσω της συνάρτησης converToCNF() που ορίζει. Συνδέεται με τον GoalNode μεσω μίας συσχέτισης πολλαπλότητας ένα προς ένα.

4.2.19 ΚλάσηAnnotationContainer

Η κλάση AnnotationContainer αντιπροσωπεύει μια οντότητα η οποία ουσιαστικά είναι η αποθηκευτική δομή των επισημιώσεων που ορίζονται από τον χρήστη του περιβάλλοντος-πλαισίου. Έπομένως συνδέεται με μία σχέση σύνθεσης με την κλάση Annotation από την οποία λαμβάνει την επιπλέον ιδιότητα +*annotation* τύπου *Annotation* σπολλαπλότητας 0..*.

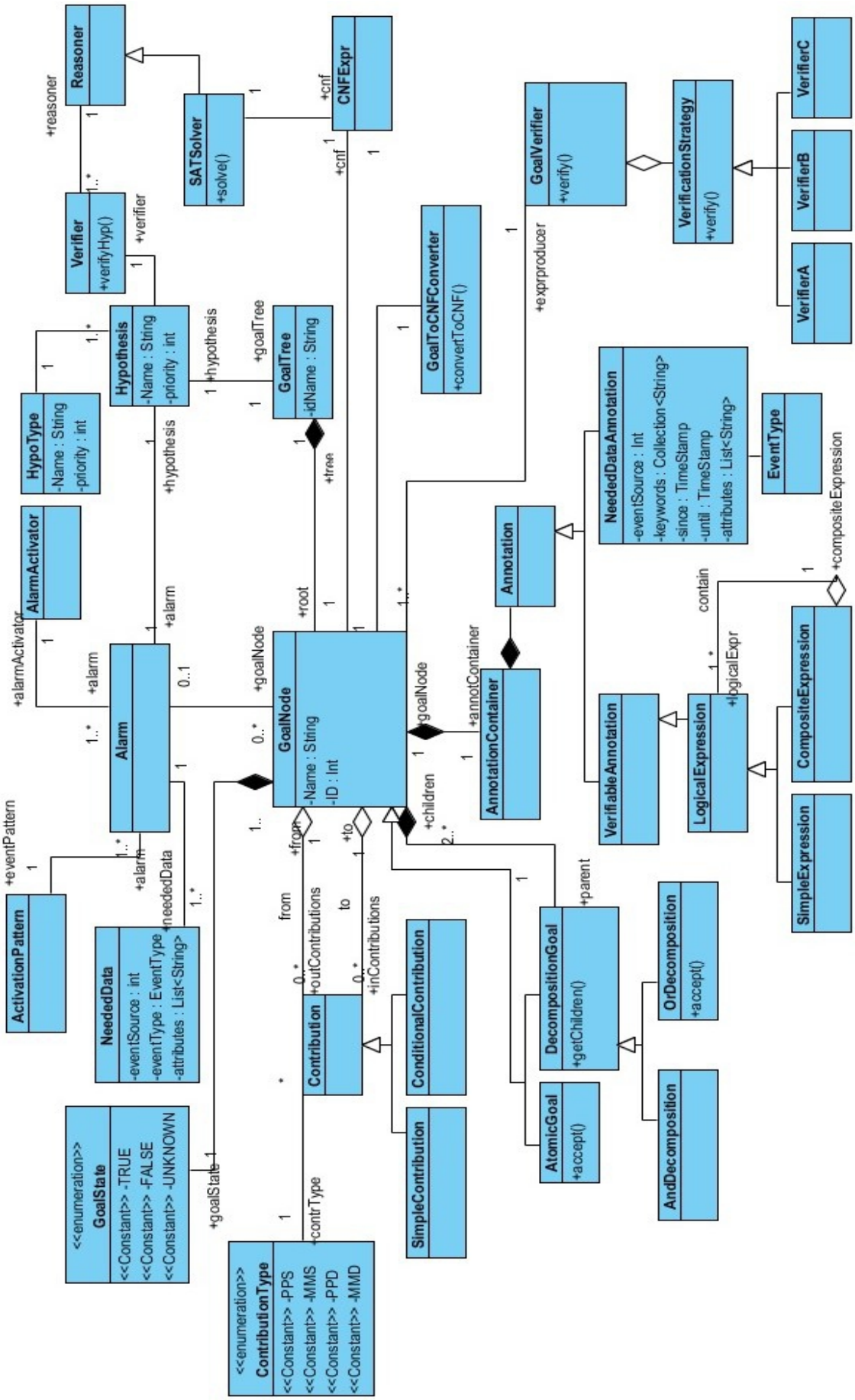
4.2.20 Κλάση Annotation

Τα στιγμιότυπα της κλάσης Annotation συνθέτουν την οντότητα AnnotationContainer που περιγράψαμε προηγουμένως και ορίζουν όλες τις επισημιώσεις που ορίζονται από τον χρήστη και συνδέονται στο μοντέλο στόχων. Η κλάση Annotation αντιπροσωπεύει τα γενικά μέσα επισύναψης των απαραίτητων πληροφοριών για την κατασκευή των κατηγορημάτων για κάθε κόμβο. Τα στιγμιότυπα αυτής της κλάσης περιέχονται σε ένα στιγμιότυπο της κλάσης AnnotationContainer και ως εκ τούτου, η κλάση αυτή έχει μια επιπλέον ιδιότητα +*annotcontainer* τύπου *AnnotationContainer* και πολλαπλότητας 1. Τα annotations μπορεί να είναι διαφόρων τύπων, αλλά στα πλαίσια της παρούσας διπλωματικής εξετάζουμε μόνο annotations που βοηθούν την διαδικασία επαλήθευσης ενός στόχου. Επομένως η κλάση Annotation έχει ως υποκλάσεις την κλάση *NeededDataAnnotation* η οποία ορίζει τα δεδομένα παρακολούθησης που χρειάζονται ώστε να αποδειχθεί η ισχύς του στόχου και την κλάση *VerifiableAnnotation* η οποία περιγράφει ορίζει τις συνθήκες που πρέπει να ικανοποιούν τα δεδομένα παρακολούθησης ώστε να καταστήσουν τον στόχο αληθή ή ψευδή. Μία υποκλάση της VerifiableAnnotation είναι η κλάση *LogicalExpression*. Τα στιγμιότυπα αυτής της κλάσης είναι λογικές εκφράσεις, όπως πχ εκφράσεις λογικής πρώτης τάξης. Η κλάση αυτή έχει την ιδιότητα +*predicate* τύπου String η οποία αναπαριστά το κατηγόρημα της λογικής έκφρασης. Η κλάση LogicalExpression έχει δύο υποκλάσεις οι οποίες κληρονομούν απο αυτήν, την κλάση *SimpleExpression* και *CompositeExpression*. Η κλάση *CompositeExpression* έχει μία σχέση συνάθροισης με την κλάση *LogicalExpression* καθώς μία σύνθετη λογική έκφραση αποτελείται απο πολλές λογικές εκφράσεις, απλές και σύνθετες. Ως εκ τούτου η κλάση CompositeExpression έχει την ιδιότητα +*logicalExpr* τύπου *LogicalExpression* και πολλαπλότητας 1..*.

4.2.21 Κλάση NeededDataAnnotation

Η κλάση *NeededDataAnnotation* είναι η κεντρική κλάση στην διαδικασία προσαρμογής της στάθμης επίβλεψης. Μέσω αυτής ορίζονται τα επιπλέον γεγονότα που πρέπει να συλλεγούν έτσι ώστε να αποδειχθεί η ισχύς ενός στόχου. Η κλάση αυτή έχει ως πεδία το *eventSource* τύπου Integer που αποτελεί το διαχωριστικό γνώρισμα των διαφόρων πηγών συλλογής

δεδομένων, τα *since* και *until* τύπου TimeStamp τα οποία ορίζουν το χρονικό διάστημα ενδιαφέροντος, το *keywords* τύπου Collection<String> που ορίζει τις λέξεις κλειδιά που πρέπει να περιέχονται στα γεγονότα ώστε να συλλεγούν, και το attributes τύπου List<String> που ορίζει τα attributes των γεγονότων παρακολούθησης που ενδιαφέρουν(πχ τον τύπο του γεγονότος και την χρονοσφραγίδα του). Τέλος λαμβάνει το πεδίο *+eventType* τύπου *EventType* και πολλαπλότητας 1 από την συσχέτιση του με την κλάση *EventType*. Έτσι γίνεται δυνατό το φιλτράρισμα τον συλλεγόμενων δεδομένων με βάση τον τύπο τους την πηγή απο την οποία προέρχονται και τις λέξεις κλειδιά που περιέχουν.



Σχήμα 4.1: Εννοιολογικό Μοντέλο Δέντρων Στόχων

Κεφάλαιο 5: Συλλογιστική βασισμένη στα δέντρα στόχων

Στο κεφάλαιο αυτό παρουσιάζουμε την συλλογιστική των μοντέλων δέντρων στόχων. Στην αρχή παρουσιάζουμε την μετατροπή των μοντέλων δέντρων στόχων σε συζευκτική κανονική μορφή (CNF), την αναγωγή του προβλήματος ανάλυσης βασικού αιτίου στο πρόβλημα ικανοποιησιμότητας λογικών εκφράσεων και την εφαρμογή του SAT4j για την επίλυση του. Στην συνέχεια παρουσιάζεται ο αλγόριθμος επαναφοράς του επιπέδου παρακολούθησης. Τέλος δίνεται μια συνολική επισκόπηση του βασικού βρόχου ελέγχου του περιβάλλοντος-πλαισίου και μία μελέτη περίπτωσης που επιδεικνύει την λειτουργία του περιβάλλοντος και τα όσα συζητούνται σε αυτό και στα προηγούμενα κεφάλαια.

5.1 Μετασχηματισμός Δεντρων Στόχων σε Κανονική Συζευκτική Μορφή

Στο κεφάλαιο αυτό παρουσιάζεται ο μετασχηματισμός των δέντρων στόχων σε λογικές εκφράσεις σε συζευκτική κανονική μορφή. Αρχικά παρουσιάζεται ο μετασχηματισμός των αποσυνθέσεων ΚΑΙ και Ή. Στην συνέχεια παρουσιάζεται με φορμαλιστικό τρόπο ο μετασχηματισμός των δέντρων στόχων σε CNF. Η ενότητα κλείνει με ένα παράδειγμα μετασχηματισμού ενός απλού μοντέλου δέντρων στόχων σε CNF.

5.1.1 Μετασχηματισμός Αποσυνθέσεων και Συνεισφορών

Ένα σημαντικό μέρος της ανάλυσης των δέντρων στόχων και της συλλογιστικής με βάση αυτά, είναι η παραγωγή της βάσης των κανόνων, δηλαδή ο μετασχηματισμός του μοντέλου δέντρων στόχων σε συζευκτική κανονική μορφή (Conjunctive Normal Form - CNF). Προκειμένου να διεξαχθεί η ανάλυση με συνοπτικό και συγκεκριμένο τρόπο, είναι ζωτικής σημασίας να διατηρήσουμε ένα επίπεδο φορμαλισμού ως προς τον αλγόριθμο μετασχηματισμού, κατά τη μετατροπή οποιαδήποτε στιγμιοτύπου του μοντέλου δέντρων στόχων στο αντίστοιχο σύνολο κανόνων CNF. Η συνάρτηση μετασχηματισμού θα πρέπει να επιστρέψει μια σειρά κανόνων προτασιακής λογικής, οι οποίοι θα μπορούν να εκφράσουν όλους τους κανόνες που μοντελοποιούνται στο μοντέλο δέντρων στόχων και τις σχέσει μεταξύ των στόχων συμπεριλαμβανομένων και των AND/OR αποσυνθέσεις και τις συνεισφορές. Για το σκοπό αυτό, είναι σημαντικό να καθορίσουμε την σημασιολογία που χρησιμοποιείται για να εκτελέσει τη διαδικασία συλλογισμού. Σε αυτό το υποκεφάλαιο δίνουμε την αντιστοίχιση των μοντέλων δέντρων στόχων σε κανόνες CNF.

Δέντρα στόχων AND/OR: Το Μοντέλο Δέντρων στόχων βασίζεται στην έννοια της από πάνω προς τα κάτω αποσύνθεσης των στόχων σε υποστόχους και έχει χρησιμοποιηθεί με επιτυχία

για τον καθορισμό λειτουργικών και μη-λειτουργικών απαιτήσεων συστημάτων λογισμικού [5]. Πιο συγκεκριμένα, ένας στόχος μπορεί να χωριστεί σε επιμέρους στόχους οι οποίοι αναπαριστώνται ως παιδιά του. Δανειζόμενοι την σημειογραφία που χρησιμοποιείται στο [5], συμβολίζουμε μία αποσύνθεση AND του στόχου G σε ένα σύνολο των επιμέρους στόχων ως εξής:

$$G \overset{\text{AND}}{\rightarrow} a \qquad (1)$$

Εναλλακτικά, ένας στόχος μπορεί να είναι τύπου OR. Παρομοίως με την περίπτωση της αποσύνθεσης AND, συμβολίζουμε την αποσύνθεση Or ως εξής:

$$G \overset{\text{OR}}{\rightarrow} a \qquad (2)$$

Επιπλέον με την έννοια της αποσύνθεσης έχουμε ορίσει και την έννοια της συνεισφοράς (contribution). Πιο συγκεκριμένα ένας στόχος μπορεί να συνεισφέρει σε άλλους στόχους με τέσσερις πιθανούς τρόπους[6], δηλαδή, $++S(g,g')$ και $--S(g,g')$ που σημαίνουν ότι ο στόχος g συνεισφέρει θετικά, ή αντιστοιχα αρνητικά, στον στόχο g' και $++D(g,g')$ και $--D(g,g')$ που σημαίνουν ότι η άρνηση του στόχου g συνεισφέρει θετικά ή αρνητικά στην άρνηση του στόχου g'. Αυτοί οι τέσσερεις τύποι συνεισφορών συμβολίζονται σε μαθηματική λογική ως εξής:

$++S(g,g')$ είναι ανάλογο προς $g \rightarrow g'$ (3)

$--S(g,g')$ είναι ανάλογο προς $g \rightarrow \neg g'$ (4)

$++D(g,g')$ είναι ανάλογο προς $\neg g \rightarrow \neg g'$ (5)

$--D(g,g')$ είναι ανάλογο προς $\neg g \rightarrow g'$ (6)

Οι συνεισφορές μπορεί να ισχύουν υπο ορισμένες συνθήκες. Έτσι πέραν των συνεισφορών που περιγράψαμε παραπάνω έχουμε και τις συνεισφορές υπό συνθήκη (conditional contributions).

Ακολουθούν οι κανόνες μετασχηματισμού των δέντρων στόχων σε CNF.

Μετασχηματισμός αποσυνθέσεων

- Αποσύνθεση AND

$$G \overset{\text{AND}}{\rightarrow} a \Rightarrow g_1 \wedge g_2 \wedge \ldots \wedge g_n \leftrightarrow a \Rightarrow$$

$$\Rightarrow (g_1 \wedge g_2 \wedge \ldots \wedge g_n \rightarrow a) \wedge (a \rightarrow g_1 \wedge g_2 \wedge \ldots \wedge g_n) \Rightarrow$$
$$\Rightarrow (\neg(g_1 \wedge g_2 \wedge \ldots \wedge g_n) \vee a) \wedge (\neg a \vee (g_1 \wedge g_2 \wedge \ldots \wedge g_n)) \Rightarrow$$
$$\Rightarrow (\neg g_1 \vee \neg g_2 \vee \ldots \vee \neg g_n \vee a) \wedge (\neg a \vee g_1) \wedge (\neg a \vee g_2) \wedge \ldots \wedge (\neg a \vee g_n) \qquad (7)$$

- Αποσύνθεση OR

$$G \overset{\text{OR}}{\rightarrow} a \Rightarrow g_1 \vee g_2 \vee \ldots \vee g_n \leftrightarrow a \Rightarrow$$
$$\Rightarrow (g_1 \vee g_2 \vee \ldots \vee g_n \rightarrow a) \wedge (a \rightarrow g_1 \vee g_2 \vee \ldots \vee g_n) \Rightarrow$$
$$\Rightarrow (\neg(g_1 \vee g_2 \vee \ldots \vee g_n) \vee a) \wedge (\neg a \vee (g_1 \vee g_2 \vee \ldots \vee g_n)) \Rightarrow$$
$$\Rightarrow ((\neg g_1 \wedge \neg g_2 \wedge \ldots \wedge \neg g_n) \vee a) \wedge (\neg a \vee g_1 \vee g_2 \vee \ldots \vee g_n) \Rightarrow$$

$$\Rightarrow\left(\left(\neg g_1\vee a\right)\wedge\left(\neg g_2\vee a\right)\wedge\ldots\wedge\left(\neg g_n\vee a\right)\right)\wedge\left(\neg a\vee g_1\vee g_2\vee\ldots\vee g_n\right)\Rightarrow$$
$$\Rightarrow\left(\neg g_1\vee a\right)\wedge\left(\neg g_2\vee a\right)\wedge\ldots\wedge\left(\neg g_n\vee a\right)\wedge\left(\neg a\vee g_1\vee g_2\vee\ldots\vee g_n\right) \qquad (8)$$

Μετασχηματισμός Συνεισφορών

$$g_1\overset{++S}{\rightarrow}g_2\Rightarrow g_1\rightarrow g_2\Rightarrow\neg g_1\vee g_2 \qquad (9)$$

$$g_1\overset{--S}{\rightarrow}g_2\Rightarrow g_1\rightarrow\neg g_2\Rightarrow\neg g_1\vee\neg g_2 \qquad (10)$$

$$g_1\overset{++D}{\rightarrow}g_2\Rightarrow\neg g_1\rightarrow\neg g_2\Rightarrow g_1\vee\neg g_2 \qquad (11)$$

$$g_1\overset{--D}{\rightarrow}g_2\Rightarrow\neg g_1\rightarrow g_2\Rightarrow g_1\vee g_2 \qquad (12)$$

5.1.2 Δημιουργία τελικής έκφρασης CNF

Για να δείξουμε την διαδικασία παραγωγής της έκφρασης CNF θεωρούμε την ενδιάμεση έκφραση f η οποία βρίσκεται σε CNF και αντιστοιχεί σε μέρος του μοντέλου δένδρων στόχων. Οι αποσυνθέσεις AND και OR είναι ήδη σε CNF και προστίθενται στην f με το λογικό ΚΑΙ. Οπότε η έκφραση που προκύπει είναι επίσης σε CNF. Οι συνεισφορές προστίθενται στην έκφραση f με το λογικό Ή. Η νέα έκφραση δεν είναι πλέον σε CNF και για να την μετατρέψουμε σε CNF κάνουμε χρήση της επιμεριστικής ιδιότητας όπως φαίνεται παρακάτω.

Έστω $f=\left(v_1\vee v_2\vee\ldots\vee v_n\right)\wedge\left(v_1^{(2)}\vee v_2^{(2)}\vee\ldots\vee v_n^{(2)}\right)\wedge\ldots\wedge\left(v_1^{(m)}\vee v_2^{(m)}\vee\ldots\vee v_n^{(m)}\right)$ η ενδιάμεση έκφραση CNF που έχει προκύψει από τον μετασχηματισμό κάθε δέντρου σε CNF, όπου $v_1, v_2, \ldots, v_n, v_1^{(2)}, v_2^{(2)}, \ldots, v_n^{(2)}, \ldots v_1^{(m)}, v_2^{(m)}, \ldots, v_n^{(m)}$ ένας στόχος ή η άρνηση ενός στόχου.

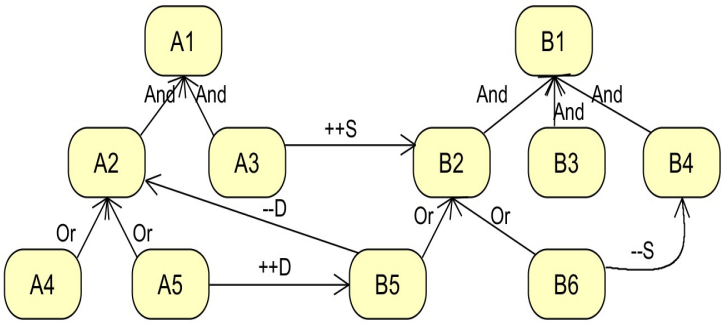
Έστω μία συνεισφορά τύπου ++S από τον κόμβο g_1 στον κόμβο g_2

Η έκφραση που προκύπτει από την λογική διάζευξη της συνεισφοράς και της CNF έκφρασης f είναι η εξής:

$$f\vee\left(\neg g_1\vee g_2\right)\Rightarrow$$
$$\Rightarrow\left(\left(v_1\vee v_2\vee\ldots\vee v_n\right)\wedge\left(v_1^{(2)}\vee v_2^{(2)}\vee\ldots\vee v_n^{(2)}\right)\wedge\ldots\wedge\left(v_1^{(m)}\vee v_2^{(m)}\vee\ldots\vee v_n^{(m)}\right)\right)\vee\left(\neg g_1\vee g_2\right)\overset{\text{επιμεριστική ιδιοτητα}}{\Rightarrow}$$
$$\Rightarrow\left(\left(v_1\vee v_2\vee\ldots\vee v_n\right)\vee\left(\neg g_1\vee g_2\right)\right)\wedge\left(\left(v_1^{(2)}\vee v_2^{(2)}\vee\ldots\vee v_n^{(2)}\right)\vee\left(\neg g_1\vee g_2\right)\right)\wedge\ldots$$
$$\ldots\wedge\left(\left(v_1^{(m)}\vee v_2^{(m)}\vee\ldots\vee v_n^{(m)}\right)\vee\left(\neg g_1\vee g_2\right)\right)$$

Η παραπάνω έκφραση βρίσκεται σε CNF. Ομοίως αντιμετωπίζονται και οι υπόλοιπες συνεισφορές και δίνουν τα αντίστοιχα αποτελέσματα.

Για να γίνουν σαφή τα παραπάνω δίνουμε ως παράδειγμα την CNF έκφραση που προκύπτει από τον μετασχηματισμό του μοντέλο δέντρων στόχων του σχήματος που ακολουθεί.



Σχήμα 5.1: Παράδειγμα Μοντέλου Δεντρων Στόχων

$$((A_2 \wedge A_3 \leftrightarrow A_1) \wedge (A_4 \vee A_5 \leftrightarrow A_2) \wedge (B_2 \wedge B_3 \wedge B_4 \leftrightarrow B_1) \wedge (B_5 \vee B_6 \leftrightarrow B_2)) \vee \ldots$$
$$\ldots \vee (A_3 \overset{++S}{\rightarrow} B_2) \vee (B_5 \overset{--D}{\rightarrow} A_2) \vee (A_5 \overset{++D}{\rightarrow} B_5) \vee (B_6 \overset{--S}{\rightarrow} B_4) \Rightarrow$$

$$\Rightarrow ((\neg A_2 \vee \neg A_3 \vee A_1) \wedge (A_2 \vee \neg A_1) \wedge (A_3 \vee \neg A_1) \wedge \ldots$$
$$\ldots \wedge (A_4 \vee A_5 \vee \neg A_2) \wedge (\neg A_4 \vee A_2) \wedge (\neg A_4 \vee A_2) \wedge \ldots$$
$$\ldots \wedge (\neg B_2 \vee \neg B_3 \vee B_4 \vee B_1) \wedge (B_2 \vee \neg B_1) \wedge (B_3 \vee \neg B_1) \wedge (B_4 \vee \neg B_1) \wedge \ldots$$
$$\ldots \wedge (B_5 \vee B_6 \vee \neg B_2) \wedge (\neg B_5 \vee B_2) \wedge (\neg B_6 \vee B_2)) \vee \ldots$$
$$\ldots \vee ((\neg A_3 \vee B_2) \vee (B_5 \vee A_2) \vee (A_5 \vee \neg B_5) \vee (\neg B_6 \vee B_4)) \Rightarrow$$

$$\Rightarrow (\neg A_2 \vee \neg A_3 \vee A_1 \vee \neg A_3 \vee B_2 \vee B_5 \vee A_2 \vee A_5 \vee \neg B_5 \vee \neg B_6 \vee B_4) \wedge \ldots$$
$$\ldots \wedge (A_2 \vee \neg A_1 \vee \neg A_3 \vee B_2 \vee B_5 \vee A_2 \vee A_5 \vee \neg B_5 \vee \neg B_6 \vee B_4) \wedge \ldots$$
$$\ldots \wedge (A_3 \vee \neg A_1 \vee \neg A_3 \vee B_2 \vee B_5 \vee A_2 \vee A_5 \vee \neg B_5 \vee \neg B_6 \vee B_4) \wedge \ldots$$
$$\ldots \wedge (A_4 \vee A_5 \vee \neg A_2 \vee \neg A_3 \vee B_2 \vee B_5 \vee A_2 \vee A_5 \vee \neg B_5 \vee \neg B_6 \vee B_4) \wedge \ldots$$
$$\ldots \wedge (\neg A_4 \vee A_2 \vee \neg A_3 \vee B_2 \vee B_5 \vee A_2 \vee A_5 \vee \neg B_5 \vee \neg B_6 \vee B_4) \wedge \ldots$$
$$\ldots \wedge (\neg A_4 \vee A_2 \vee \neg A_3 \vee B_2 \vee B_5 \vee A_2 \vee A_5 \vee \neg B_5 \vee \neg B_6 \vee B_4) \wedge \ldots$$
$$\ldots \wedge (\neg B_2 \vee \neg B_3 \vee B_4 \vee B_1 \vee \neg A_3 \vee B_2 \vee B_5 \vee A_2 \vee A_5 \vee \neg B_5 \vee \neg B_6 \vee B_4) \wedge \ldots$$
$$\ldots \wedge (B_2 \vee \neg B_1 \vee \neg A_3 \vee B_2 \vee B_5 \vee A_2 \vee A_5 \vee \neg B_5 \vee \neg B_6 \vee B_4) \wedge \ldots$$
$$\ldots \wedge (B_3 \vee \neg B_1 \vee \neg A_3 \vee B_2 \vee B_5 \vee A_2 \vee A_5 \vee \neg B_5 \vee \neg B_6 \vee B_4) \wedge \ldots$$
$$\ldots \wedge (B_4 \vee \neg B_1 \vee \neg A_3 \vee B_2 \vee B_5 \vee A_2 \vee A_5 \vee \neg B_5 \vee \neg B_6 \vee B_4) \wedge \ldots$$
$$\ldots \wedge (B_5 \vee B_6 \vee \neg B_2 \vee \neg A_3 \vee B_2 \vee B_5 \vee A_2 \vee A_5 \vee \neg B_5 \vee \neg B_6 \vee B_4) \wedge \ldots$$
$$\ldots \wedge (\neg B_5 \vee B_2 \vee \neg A_3 \vee B_2 \vee B_5 \vee A_2 \vee A_5 \vee \neg B_5 \vee \neg B_6 \vee B_4) \wedge \ldots$$
$$\ldots \wedge (\neg B_6 \vee B_2 \vee \neg A_3 \vee B_2 \vee B_5 \vee A_2 \vee A_5 \vee \neg B_5 \vee \neg B_6 \vee B_4)$$

5.2 Αναγωγή στο πρόβλημα της ικανοποιησιμότητας

Στην ενότητα αυτήν παρουσιάζουμε την αναγωγή του προβλήματος ανάλυσης βασικού αιτίου στο πρόβλημα της ικανοποιησιμότητας και την εφαρμογή του SAT4j. Παρουσιάζονται οι ενέργειες που εκτελούνται πριν χρήση του SAT4j και αυτές που εκτελούνται μετά την λήψη των αποτελεσμάτων τους.

5.2.1 Ενέργειες πριν την χρήση του SAT4j

Οπως αναφέραμε και στο 2ο κεφάλαιο ο SAT4J είναι μια βιβλιοθήκη της γλώσσας JAVA που χρησιμοποιείται για την επίλυση του προβλήματος της ικανοποιησιμότητας. Υπάρχουν πολλές εκδόσεις του SAT4J για παραλλαγές του προβλήματος της ικανοποιησιμότητας όπως το MaxSat. Η έκδοση του SAT4J που χρησιμοποιούμε επιστρέφει ως αποτέλεσμα όλες τους

δυνατούς τρόπους ικανοποίησης μιας εκφρασης προτασιακής λογικής. Δηλαδή επιστρέφει ένα σύνολο λύσεων για την λογική έκφραση, όπου κάθε λύση είναι ένα σύνολο αληθοτιμών για κάθε λογική μεταβλητή που περιλαμβάνεται στην έκφραση CNF.

Η έκφραση που χρησιμοποιείται ως είσοδος στον SAT4J για την εξαγωγή των μοντέλων που την ικανοποιούν, σχηματίζεται τόσο από τον μετασχηματισμό του μοντέλου δέντρων στόχων σε CNF (με βάση τους κανόνες που δόθηκαν παραπάνω) όσο και από την βάση γνώσης που αποτελείται από στόχους για τους οποίους γνωρίζουμε την αληθοτιμή τους, αλλά και από τους κόμβους για τους οποίους δεν γνωρίζουμε την αληθοτιμή τους και θέλουμε να επιβεβαιώσουμε υποθέσεις για αυτούς. Για να δώσουμε μια καλύτερη εικόνα της έκφρασης που δημιουργείται αν f είναι η έκφραση που προκύπτει από τον μετασχηματισμό των δέντρων στόχων, G1 ένας κόμβος για τον οποίο έχουμε αποδείξει την ισχύ του G2 ένας κόμβος για τον οποίο έχουμε αποδείξει ότι είναι ψευδής και G3 ένας κόμβος για τον οποίο δεν γνωρίζουμε την αληθοτιμή του και θέλουμε να αποδείξουμε την ισχύ του, τότε η έκφραση που προκύπτει είναι $f \wedge G1 \wedge \neg G2 \wedge G3$.

Έτσι το πρώτο βήμα για την αξιοποίηση του SAT4J είναι ο μετασχηματισμός του μοντέλου δέντρων στόχων σε CNF. Η επιλογή του υποσυνόλου του μοντέλου δενδρων στόχων που θα σχηματίσει την CNF έκφραση είναι καίριας σημασίας για την επίδοση του αλγορίθμου συλλογιστικής και επιβεβαίωσης μιας υπόθεσης. Έτσι δεδομένης μιας υπόθεσης προς απόδειξη σχηματίζουμε μια CNF έκφραση που περιλαμβάνει την προς απόδειξη υπόθεση, όλους τους κόμβους-στόχους που αποτελούν το δέντρο με ρίζα την υπόθεση αυτή και όλους τους κόμβους-στόχους που συνεισφέρουν στην υπόθεση αυτή ή στο δέντρο με ρίζα την υπόθεση αυτήν. Οπως είπαμε παραπάνω ο Sat4J επιστρέφει ως αποτέλεσμα όλους τους τρόπους να ικανοποιηθεί η έκφραση. Στον κόσμο των δέντρων στόχων αυτό ισοδυναμεί με όλους τους τρόπους να αποδειχθεί μία υπόθεση. Πιο συγκεκριμένα επιστρέφει ένα σύνολο λύσεων με κάθε λύση να περιλαμβάνει μια αληθοτιμή για κάθε κομβο στόχο που περιέχεται στην έκφραση CNF τέτοιες ώστε να ικανοποιούν την έκφραση. Η αληθοτιμή αυτή είναι είτε αυτή που έχει ήδη αποδειχθεί για τον κόμβο είτε η επιθυμητή ώστε να ικανοποιηθεί η έκφραση και να επιβεβαιωθεί η υπόθεση. Οι αληθοτιμές αυτές πρέπει να επιβεβαιωθούν με την σειρά μία προς μία. Έτσι κάθε λύση μπορεί να ειδωθεί ως ένα ένα πλάνο για την επαλήθευση μιας υπόθεσης.

Μιά εναλλακτική για την κατασκευή της CNF έκφασης, θα ήταν να φτιάχναμε μία έκφραση CNF για όλο το μοντέλο δέντρων στόχων. Σε αυτήν την περίπτωση η χρήση του Sat Solver δεν θα ήταν αποδοτική καθώς θα μεγάλωνε πολύ η έκφραση CNF και θα καθυστερούσε η επίλυσή του, δίχως αυτό να είναι απαραίτητο, και ο SAT4J θα παρήγαγε ένα σύνολο με λύσεων με κάθε λύση να περιέχει αληθοτιμές για όλους τους κόμβους του μοντέλου. Αυτό δεν είναι επιθυμητό γιατί όπως αναφέραμε λιγο πιο πάνω και παρουσιάζουμε αναλυτικά παρακάτω μετά την λήψη των αποτελεσμάτων του SAT Solver θα προσπαθήσουμε εφαρμόσουμε κάθε λύση με την σειρά και για κάθε λύση μέχρι να βρεθεί η κατάλληλη και για κάθε κόμβο που περιλαμβάνεται στην λύση θα προσπαθήσουμε να επαληθεύσουμε αληθοτιμή του. Αυτό πρακτικά σημαίνει ότι θα επιχειρούσαμε την επαλήθευση όλου του μοντέλου.

Σχετικά με τις συνεισφορές όπως είπαμε και παραπάνω επιλέξαμε να συμπεριλάβουμε μόνο τους κόμβους-στόχους που συνεισφέρουν σε έναν από τους κόμβους-στοχους του δέντρου με ρίζα την υπόθεση προς απόδειξη. Μια αλλη δυνατότητα θα ήταν πέραν των κόμβων που συνεισφέρουν σε κάποιο κόμβο του δέντρου που έχει ρίζα την υπόθεση προς απόδειξη να συμπεριλάβουμε και τους τους κόμβους του δέντρου που έχουν ρίζα τον κόμβο που συνεισφέρει στο αρχικό δέντρο, αλλά και αναδρομικά τους κόμβους κόμβους που συνεισφέρουν στο δεύτερο δέντρο με τα αντιστοιχα υποδέντρα τους κ.ο.κ. Κάτι τέτοιο όμως πρώτον θα απαιτούσε πολύ προσοχή για την αποφυγή κύκλων και θα δεύτερον πιθανόν η έκφραση που θα προέκυπτε να ήταν πολύ μεγάλη και να αυξανε σημαντικά τον χρόνο εκτέλεσης του SAT4J. Σε περίπτωση που χρησιμοποιήσουμε καποια λύση από αυτες που επέστρέψε ο SAT4J απαιτεί την απόδειξη ενός κόμβου-στοχου που προέρχεται απο

συνεισφορά τρέχουμε τότε τον SAT4J για τον κόμβο αυτό (και το δέντρο κάτω απο αυτόν και τις συνεισφορές στο δέντρο αυτό). Παρουσιάζουμε αυτήν την διαδικασία αναλυτικά σε επόμενο κεφάλαιο. Εδώ θέλαμε απλά αιτιολογήσουμε τις επιλογές μας για την δημιουργία της CNF έκφρασης. Ακολουθεί ένα παράδειγμα που επιδεικνύει τον τρόπο κατασκευής της έκφρασης CNF.

Έστω το μοντέλο δένδρων στόχων που δίνεται στην προηγούμενη σελίδα, και έστω ότι θέλουμε να αποδείξουμε την ισχύ του στόχου A_1 και ότι έχει ήδη αποδειχθεί η ισχύς του A_3 και η άρνηση του B_3 . Η έκφραση CNF που παράγεται για την επαλήθευση του A_1 περιλαβμάνει όλο το δέντρο κάτω από τον κόμβο A_1 την συνεισφορά --D την αληθοτιμή του κόμβου A_3 που είναι ήδη γνωστή και την αληθοτιμή που θέλουμε να επαληθεύσουμε για τον A_1 . Η αλληθοτιμή του B_3 μας είναι αδιάφορη καθώς ανήκει σε άλλο δέντρο και δεν συνεισφέρει στο δέντρο με κάποιο τρόπο στο υπό εξέταση δέντρο. Έπισης είναι αδιαφορες οι συνεισφορές ++S, ++D και --S καθώς ούτε αυτές συνεισφέρουν στο υποεξέταση δέντρο.

Έτσι η έκφραση που προκύπτει είναι η εξής:

$$((A_2 \wedge A_3 \leftrightarrow A_1) \wedge (A_4 \vee A_5 \leftrightarrow A_2)) \vee (B_5 \overset{--D}{\rightarrow} A_2) \wedge A_3 \wedge A_1$$

ή σε CNF

$$(\neg A_2 \vee \neg A_3 \vee A_1 \vee B_5 \vee A_2) \wedge (A_2 \vee \neg A_1 \vee B_5 \vee A_2) \wedge (A_3 \vee \neg A_1 \vee B_5 \vee A_2) \wedge \ldots$$
$$\ldots \wedge (A_4 \vee A_5 \vee \neg A_2 \vee B_5 \vee A_2) \wedge (\neg A_4 \vee A_2 \vee B_5 \vee A_2) \wedge (\neg A_4 \vee A_2 \vee B_5 \vee A_2) \wedge \ldots$$
$$\ldots \wedge A_3 \wedge A_1$$

5.2.2 Εφαρμογή του SAT4j

Στην ενότητα αυτήν παρουσιάζουμε λεπτομέρειες της χρήσης του SAT4J και των ενεργειών που εκετελούνται μετά την λήψη των αποτελεσμάτων του.

Για να ξεκινήσουμε την ανάλυση αυτού του κεφαλαίου θεωρούμε ότι έχουμε ένα σύνολο υποθέσεων προς απόδειξη. Το πρώτο βήμα για την επαλήθευση των υποθέσεων αυτών είναι η ταξινόμηση τους έτσι ώστε η σειρά με την οποία επαληθεύονται να μην είναι τυχαία αλλά να βασίζεται σε κάποια λογική που μπορεί να οριστεί απο τον χρήστη. Στην πρότυπη υλοποίηση ταξινομούμε την λίστα των υποθέσεων με σειρά σημαντικότητας βάσει του πεδίου Priority της κλάσης Hypothesis. Το πρωτότυπο του περιβάλλοντος-πλαισίου που σχεδιάσαμε και υλοποιήσαμε κάνει χρήση των σχεδιαστικών μορφημάτων Strategy και Factory Method έτσι ώστε να είναι δυνατός ο ορισμός και η εφαρμογή νέων στρατηγικών ταξινόμησης των υποθέσεων χωρίς επίδραση στον υπόλοιπο κώδικα του περιβάλλοντος-πλαισίου. Έτσι η στρατηγική ταξινόμησης μπορεί εύκολα να επεκταθεί για να υποστηρίξει πιο σύνθετες στρατηγικές όπως για παράδειγμα να μην βασιστεί μόνο στην σηματικότητα των υποθέσεων αλλά και σε εκτίμησεις για την πιθανότητα και τον χρόνο που απαιτείται για την επαλήθευση τους.

Αφού έχουμε αποφασίσει την σειρά με την οποία θα επιχειρηθεί η επαλήθευση των υποθέσεων σειρά έχει η εύρεση ενός πλάνου για την επαλήθευση κάθε μίας υπόθεσης και η εκτέλεση του πλάνου αυτού. Το πλάνο αυτό όπως είπαμε και παραπάνω είναι στην ουσία μία απο τις λύσεις που επιστρέφει ο SAT4J. Έτσι εκτελείται ο SAT4J για κάθε μία από τις υποθέσεις αυτές. Ο SAT4J επιστρέφει όλους τους δυνατούς τρόπους να επαληθεύσουμε μια υπόθεση, έναν προς έναν ύστερα από αίτηση. Δηλαδή επιστρέφει έναν πιθανο τρόπο και διαδοχικά τους υπόλοιπους όσο συνεχίζουμε να ζητάμε κι άλλους μέχρι να επιστραφούν όλες οι λύσεις. Η διαδικασία αυτή μπορεί να είναι πολύ χρονοβόρα και ο χρόνος εξαρτάται εκτός των αλλων και

από το πλήθος τον λύσεων που επιστρέφονται. Έτσι αντί να ζητάμε όλους τους δυνατούς τρόπους επαλήθευσης μιας υπόθεσης περιορίζομαστε στους Ν πρώτους, τους ταξινομούμε και επιχειρούμε την εφαρμογή καθενός με την σειρά μέχρι να βρεθεί κάποιο που οδηγεί στην επιβεβαίωση ή την άρνηση της υπόθεσης. Αν και μπορεί στις Ν πρώτες λύσεις που επιστρέφει ο SAT4J μπορεί να μην βρεθεί κάποια που να οδηγεί στην επαλήθευση της υπόθεσης η προσπάθεια εφαρμογής των λύσεων αυτών θα αποδόσει αληθοτιμές σε κάποιους από τους κόμβους-στόχους του δέντρου που μοντελοποιεί την υπόθεση αυτή. Έτσι αν δεν βρεθεί μία λύση στις πρώτες Ν αντί να συνεχίσουμε με τις επόμενες Ν εκτελούμε ξανά τον SAT4J για αυτήν την υπόθεση με την νέα βάση γνώσης. Η απόφαση αυτή βασίστηκε στο γεγονός ότι καθώς αλλάζει η βάση γνώσης με την απόδειξη νέων στόχων πολλές απο τις λύσεις που επιστρέφει ο SAT4J θα είναι άκυρες καθώς δεν θα συμφωνούν με την νέα βάση γνώσης. Έτσι απαιτείται η εκτέλεση του SAT4J με την νέα βάση γνώσης. Η γνώση για τις αληθοτιμές των κόμβων στόχων θέτει επιπλέον περιορισμούς στον SAT4J και έτσι μειώνεται το πλήθος των λύσεων που επιστρεφονται. Οι λύσεις αυτές συμφωνούν με την νέα βάση γνώσης και έτσι αποφεύγεται η εφαρμογή άκυρων λύσεων.

5.2.3 Επαλήθευση στόχων και ενέργειες που εκτελούνται μετά την λήψη των αποτελεσμάτων του SAT4J

Παραπάνω περιγράψαμε σε γενικές γραμμές τον τρόπο χρήσης του SAT4J. Εδώ περιγράφουμε τις ενέργειες που εκτελούνται μετά την λήψη των αποτελεσμάτων του SAT4J για την επαλήθευση ενός στόχου.

Θεωρούμε μια υπόθεση προς απόδειξη. Για την υπόθεση αυτήν εκτελούμε τον SAT4J και παίρνουμε ως αποτέλεσμα ένα σύνολο λύσεων. Κάθε λύση αποτελείται από ένα σύνολο κόμβων (που είναι είτε στο δέντρο που έχει ρίζα την υπόθεση αυτήν είτε συνεισφέρει στο δέντρο αυτό), μαζί με μία αληθοτιμή για κάθε κόμβο. Η αληθοτιμή αυτή είναι είτε αυτή που έχει ο κόμβος-στόχος ή η επιθυμητή, δηλαδή αυτή που θέλουμε να επιβεβαιώσουμε. Το σύνολο των λύσεων αυτών ταξινομούνται με βάση κάποια στρατηγική. Όπως έχουμε ήδη αναφέρει κάθε λύση του SAT4J ισοδυναμεί με έναν τρόπο απόδειξης μιας υπόθεσης. Κάθε τρόπος απόδειξης μπορεί να απαιτεί την συλλογή διαφορετικού είδους πληροφοριών, από διαφορετικές πηγές και αρχεία καταγραφής. Έτσι διαφορετικοί τρόποι επαλήθευσης μιας υπόθεσης μπορεί να συνεπάγονται διαφορετικό κόστος. Η στρατηγική ταξινόμησης των λύσεων του SAT4J λοιπόν μπορεί να βασίζεται σε εκτιμήσεις του κόστους επαλήθευσης της απόδειξης (cost-aware). Στην πρότυπη υλοποίηση δίνεται η δυνατότητα εφαρμογής διαφορετικών στρατηγικών μέσω της χρήσης των σχεδιαστικών μορφημάτων Strategy και Factory Method. Στην πρωτότυπη υλοποίηση χρησιμοποιούμε έναν απλοϊκό τρόπο για την αξιολόγηση του κόστους κάθε λύσης. Για την αξιολογηση και ταξινόμηση των λύσεων δεν χρησιμοποιούμε καποια εξιδεικευμένη τεχνική. Ταξινομούμε τις λύσεις με έναν απλοικό τρόπο βασιζόμενοι μόνο στο πλήθος των στόχων που πρέπει να επαληθευθούν ωστε να επαληθευτεί η υπόθεση. Όσοι επεκτείνουν το περιβάλλον-πλαισιο μπορούν να ορίσουν δικές τους στρατηγικές για την ταξινόμηση των λύσεων.

Πλέον έχουμε ένα ταξινομημένο σύνολο υποθέσεων προς επαλήθευση και για κάθε υπόθεση ένα σύνολο τρόπων για την επαλήθευση της. Κάθε τέτοιος τρόπος απότελειται από ένα σύνολο κόμβων-στόχων μαζί με την αληθοτιμή που πρέπει να ισχύει για τον κόμβο αυτόν. Οπως περιγράψαμε παραπάνω ελέγχουμε κάθε λύση μία προς μία μέχρι να βρεθεί κάποια που οδηγεί στην επαλήθευση ή την άρνηση της υπόθεσης. Ο έλεγχος κάθε λύσης απαιτεί την επαλήθευση της αληθοτιμής κάθε κόμβου-στόχου στην λύση αυτή. Η σειρά ελέγχου αυτών των κόμβων έχει σημασία και γίνεται από τα φύλλα προς την ρίζα. Έτσι το τελευταίο βήμα για την αξιοποίηση των αποτελεσμάτων το SAT Solver είναι η ταξινόμηση των κόμβων-στόχων σε κάθε λύση έτσι ώστε να επαληθεύονται πρώτα τα φύλλα.

5.3 Επαναφορά Επιπέδου Παρακολούθησης

Σε αυτήν την ενότητα περιγράφουμε την διαδικασία μείωσης του επιπέδου παρακολούθησης. Για την μείωση του επιπέδου επίβλεψης χρησιμοποιούμε τον αλγόριθμο προσομοιωμενης ανόπτησης (simulated annealing). Στην συνέχεια του κεφαλαίου περιγράφουμε τον αλγόριθμο προσομοιωμενης ανόπτησης και εξηγούμε την εφαρμογή του στην περίπτωση της μείωσης του επιπέδου παρακολούθησης.

5.3.1 Επισκόπηση του αλγορίθμου προσομοιωμένης ανόπτησης

Ο αλγόριθμος προσομοιωμένης ανόπτησης (SA) είναι ένας γενικός πιθανοτικός μεταευριστικός αλγόριθμος για προβλήματα βελτιστοποίησης με μεγάλο χώρο αναζήτησης στα οποία δεν ενδιαφέρει απαραίτητα η βέλτιστη λύση αλλά μια καλή προσέγγιση της. Δεδομένου ενός προβλήματος βελτιστοποίησης ο αλγόριθμος επιτρέπει την αποδοχή, με κάποια πιθανότητα, λύσεων χειρότερων της τρέχουσας λύσης, με σκοπό την αποφυγή τοπικών βέλτιστων λύσεων. Η αποδοχή "κακών" λύσεων αποτελεί θεμελιώδη αρχή των μεταευριστικών αλγορίθμων καθώς επιτρέπει πιο εκτενή αναζήτηση της βέλτιστης λύσης. Οι μεταευριστικοί αλγόριθμοι χρησιμοποιούν ευριστικές μεγαλύτερης τάξης έτσι ώστε να μπορούν να επιλέξουν ευριστικές χαμηλότερης τάξης, κατάλληλες να αποδόσουν μια καλή λύση σε προβλήματα βελτιστοποίησης.

Το όνομα και η έμπνευση για τον αλγόριθμο αυτόν προέρχονται από την ανόπτηση στη μεταλλουργία, μια τεχνική που περιλαμβάνει την θέρμανση και ελεγχόμενη ψύξη ενός υλικού για την αύξηση του μεγέθους των κρυστάλλων τους. Αν θερμάνθει ένα στερεό πέραν του σημείου τήξης και στη συνέχεια ψύχθεί, οι δομικές ιδιότητες του στερεού εξαρτώνται από το ρυθμό ψύξης. Εάν το υγρό ψύχθεί αρκετά αργά, θα σχηματιστούν μεγάλοι κρύσταλλοι. Ωστόσο, εάν το υγρό ψύχθεί γρήγορα οι κρύσταλλοι θα περιέχουν ατέλειες.

Η έννοια του ρυθμού ψύξης του υλικού στην περίπτωση της μεταλλουργίας αντιστοιχεί στη περίπτωση του αλγορίθμου προσομοιωμένης ανόπτησης στον ρυθμό μείωσης της πιθανότητας αποδοχής χειρότερων λύσεις της τρέχουσας.

Για να γίνει κατανοητή η ιδέα της προσομοιωμένης ανόπτησης και να φανεί η χρησιμότητα της, παρουσιάζουμε την εφαρμογή της στο γενικό πρόβλημα βελτιστοποίησης. Στην γενική περίπτωση του προβλήματος βελτιστοποίησης θεωρούμε μία αρχική κατάσταση s ένα σύνολο τελικών κατάστασεων στο οποία ανοίκει και η βέλτιστη λύση, ένα σύνολο τελεστών που δίνουν τις πιθανές επόμενες καταστάσης μιας κατάστασης και μία ευριστική συνάρτηση που χρησιμοποιείται για την αξιολόγηση επόμενων καταστάσεων. Η ιδέα την προσομοιωμένης ανόπτησης επιτρέπει σε έναν αλγόριθμο αναζήτησης να μεταβεί σε καταστάσεις χειρότερες της τρέχουσας αν δεν υπάρχει κάποια καλύτερη επόμενη κατάσταση με την ελπίδα ότι θα βγεί από κάποιο πιθανό τοπικό βέλτιστο. Η μετάβαση γίνεται με πιθανότητα P η οποία είναι συνάρτηση αυτή δίνεται από μία συνάρτηση τριών παραμέτρων. Είναι συνάρτηση των τιμών της ευριστικής της τρέχουσας και της πιθανής επόμενης κατάστασης και της παραμέτρου T η οποία στην αναλογία με την μεταλουργία αντιστοιχεί στην θερμοκρασία. Όσο μικρότερη είναι η διαφορά των τιμών των ευριστικών των δύο καταστάσεων και όσο μεγαλύτερη είναι η θερμοκρασία τόσο μεγαλύτερη είναι η πιθανότητα να γίνει αποδεκτή μία χειρότερη κατάσταση. Έτσι γίνονται δεκτές καταστάσεις που αν και χειρότερες τις παρούσας είναι καλύτερες από τις υπόλοιπες. Όταν η θερμοκρασία τείνει στο μηδέν γίνονται δεκτές μόνο καλύτερες καταστάσεις και ο αλγόριθμος δεν διαφέρει από έναν απλό αλγόριθμο αναζήτησης.

Τα παραπάνω φαίνονται και από την σχέση που δίνει την πιθανότητα αποδοχής "κακών" κινήσεων. Η πιθανότητα αποδοχής κάποιας χειρότερης κατάστασης δίνεται από την ακόλουθη σχέση:

$$P(e, e', T) = \frac{1}{e^{\frac{|e-e'|}{T}}} \quad ,$$

όπου e και e' οι τιμές των ευριστικών της παρούσας και επόμενης κατάστασης και T η θερμοκρασία.

Η θερμοκρασία πέφτει κάθε φορά με βάση κάποιο πλάνο μείωσης της θερμοκρασίας. Όταν η θερμοκρασία φτάσει στο μηδέν η πιθανότητα αποδοχής "κακών" κινήσεων φτάνει επίσης στο μηδέν. Με βάση τα παραπάνω ο αλγόριθμος προσομοιωμένης ανόπτησης εφαρμόζεται σε προβλήματα βελτιστοποίησης και δίνει τον αριθμό "κακών" κινήσεων που μπορούμε να κάνουμε έτσι ώστε να ξεφύγουμε από τοπικά βέλτιστες λύσεις. Ο αριθμός αυτός εξαρτάται άμεσα από την αρχική τιμή της θερμοκρασίας και την ταχύτητα μείωσης της. Έτσι οι παράμετροι αυτοί πρέπει να επιλεγούν με προσοχή ανάλογα με τον αριθμό των "κακών" κινήσεων που επιτρέπουμε στον αλγόριθμο να λάβει.

Στο σημείο αυτό παρουσιάζουμε τα βήματα του αλγορίθμου σε ψευδοκώδικα:

```
Function SIMULATED-ANNEALING(problem, schedule) returns state
Input: problem: a problem
       schedule: a mapping from time to "temperature"
Local Variables: current: a node, next: a node,
       T: a "temperature" controlling the probability of a bad move
current ← Initial State
for t ← 1 to ∞ do
    T ← schedule[t]
    if T == 0 then return current
    next ← a randomly selected successor of current
    ΔE ← VALUE[next] – VALUE[current]
    if ΔE > 0 then current ← next
      else current ← next only with probability  e^(ΔE/T)
```

Όπως φαίνεται και στον παραπάνω αλγόριθμο το πρώτο βήμα σε κάθε επανάληψη του βρόχου είναι η επιλογή της θερμοκρασίας. Στην σύνέχεια επιλέγεται μια κατάσταση από τις πιθανές επόμενες καταστάσεις επιλέγεται μία με τυχαίο τρόπο και αν είναι καλύτερη από την παρούσα γίνεται αποδεκτή. Αν είναι χειρότερη γίνεται αποδεκτή με μία πιθανότητα που εξαρτάται από το πόσο κόντα είμαστε στο τέλος της διαδικασίας (θερμοκρασία T) και από το πόσο χειρότερη είναι από την παρούσα κατάσταση (ΔE).

5.3.2 Εφαρμογή του αλγορίθμου προσομοιωμένης ανόπτησης στην μείωση του επιπέδου παρακολούθησης

Η ιδέα για την εφαρμογή του αλγορίθμου προσομοιωμένης ανόπτησης στο πρόβλημα της μείωσης του επιπέδου παρακολούθησης προκύπει από την ανάγκη η μείωση να γίνεται σταδιακά και όχι μονομιάς. Στην περίπτωση αδυναμίας απόδειξης ενός στόχου πως μπορούμε να γνωρίζουμε εκ του ασφαλούς ότι το χρονικό διάστημα παρακολούθησης των επιπλέον δεδομένων ήταν αρκετό; Η απάντηση σε αυτό το ερώτημα είναι ότι δεν μπορούμε να γνωρίζουμε. Παρ'όλα αυτά θέλουμε να μειώνουμε το επιπεδο παρακολούθησης μετά την

απόδειξη ενός στόχου ή ύστερα από αδυναμία απόδειξης ενός στόχου.

Μια προσέγγιση θα ήταν να σταματάμε την παρακολούθηση των επιπλέον δεδομένων που χρειάζονται για την απόδειξη του στόχου, μετά από ένα ορισμένο χρονικό διάστημα. Αντ'αυτού κρατάμε μία πιο συντηριτική αλλά και αισιόδοξη στάση όσον αφορά την μείωση του επιπέδου παρακολούθησης. Αντί να σταματάμε την παρακολούθηση των επιπλέον δεδομένων μονομιάς κάνουμε μια αξιολόγηση του περιεχομένου τους και κρατάμε μόνο αυτά που περιέχουν πληροφορία μεγαλης αξίας, με την ελπίδα να καταφέρουμε να αποδείξουμε τον στόχο χρησιμοποιώντας μόνο αυτό το μειωμένο σύνολο δεδομένων παρακολούθησης. Πιο συγκεκριμένα ορίζουμε κάποιο χρονικό διάστημα μέσα στο οποίο ευελπιστούμε να έχουμε αποδείξει τον στόχο για τον οποίο πραγματοποιείται η ανάλυση. Αν μέσα σε αυτό το διάστημα δεν έχει επιτευχθεί η απόδειξη του στόχου αυτού αρχίζουμε να μειώνουμε λίγο λίγο το πλήθος των δεδομένων παρακολούθησης που συλλέγουμε και επεξεργαζόματε κατά ένα ποσοστό μέχρι τελικά να σταματήσουμε την συλλογή όλων των επιπλέον δεδομένων. Σε αυτό το σημείο ανακύπτουν τα εξής ερωτήματα για την παραπάνω διαδικασία:

- Ποιο είναι χρονικό διάστημα μέχρι την έναρξη της διαδικασίας μείωσης του επιπέδου παρακολούθησης;
- Τι σημαίνει μείωση και αύξηση του επιπέδου παρακολούθησης;
- Ποιο είναι το ποσοστό μείωσης των επιπλέον δεδομένων παρακολουθησης;

Το πρώτο ερώτημα είναι δύσκολο να απαντηθεί. Στην γενική περίπτωση εξαρτάται από το υπο παρακολούθηση σύστημα, τον στόχο που προσπαθούμε να αποδείξουμε, την τεχνική που θα χρησιμοποιηθεί για την απόδειξη του, την λεπτομέρεια (granularity) των δεδομένων που χρειάζονται για την απόδειξη του και τις συνθήκες που πρέπει να ικανοποιούν. Το μέγεθος του υπο παρακολούθηση συστήματος έχει επίδραση στον όγκο των δεδομένων που παράγονται. Έτσι για μεγαλύτερα συστηματα απαιτείται συνήθως περισσότερος χρόνος για την απόδειξη στόχων ένος στόχου για αυτά. Ο χρόνος που απαιτείται για την απόδειξη ένος στόχου αυξάνεται επίσης όσο αυξάνεται η λεπτομέρεια των δεδομένων που χρησιμοποιουνται για την απόδειξη του, και αυξάνεται όσο αυξάνεται η πολυπλοκότητα των συνθηκών που πρέπει να ικανοποιούνται. Με βάση τα παραπάνω το χρονικό διάστημα μέχρι την έναρξη της διαδίκασίας μείωσης του επιπέδου παρακολούθησης είναι μεταβλητό και στο πρωτότυπο του περιβάλλοντος που υλοποιήσαμε έχουμε λάβει μέριμνα έτσι ώστε όσοι το χρησιμοποιούν να μπορούν να θέσουν την τιμή του διαστήματος αυτού.

Όσον αφορά το δεύτερο ερώτημα η αλλαγή του επιπέδου παρακολούθησης μπορεί να είναι coarse-grained ή fine-grained. Πιο συγκεκριμένα μια αλλαγή στο επίπεδο παρακολούθησης μπορεί να αποτελεί την έναρξη ή παύση της συλλογής δεδομένων από μια πηγή συλλογής δεδομένων (πχ ένα αρχείο καταγραφής), την έναρξη ή παυση της συλλογής μόνο μέρους των χαρακτηριστικών (attributes) των δεδομένων που παρακολουθούνται, ή ακόμα και την ενεργοποίηση και απενεργοποίηση φίλτρων που θα επιτρέπουν την προώθηση προς τον Blackboard δεδομένων με συγκεκριμένα χαρακτηριστικά.

Όσον αφορά το τρίτο ερώτημα, εδώ είναι που έρχεται ο αλγόριθμος της προσομοιωμένης ανόπτησης (simulated annealing). Το ποσοστό μείωσης των δεδομένων παρακολούθησης αντιστοιχεί στην πιθανότητα αποδοχής "κακών" κινήσεων που είδαμε παραπάνω. Πιο συγκεκριμένα ορίζουμε κάποια αρχική τιμή για την θερμοκρασία και μία στρατηγική για την μείωση της. Σε κάθε βήμα του αλγορίθμου εκτελούμε τις εξής ενέργειες:

- υπολογίζουμε την τιμή της "θερμοκρασίας" μεσω της σχέση $T(k)=\frac{T}{k}$,
- υπολογίζουμε το ποσοστό των δεδομένων που συνεχίζουμε να παρακολουθούμε με βάση την σχέση $P(T)=\mathrm{e}^{-|\frac{1}{T}|}$,
- υπολογίζουμε την νεα τιμή του k έτσι ώστε να μειώνεται σταδιακά η "θερμοκρασία"

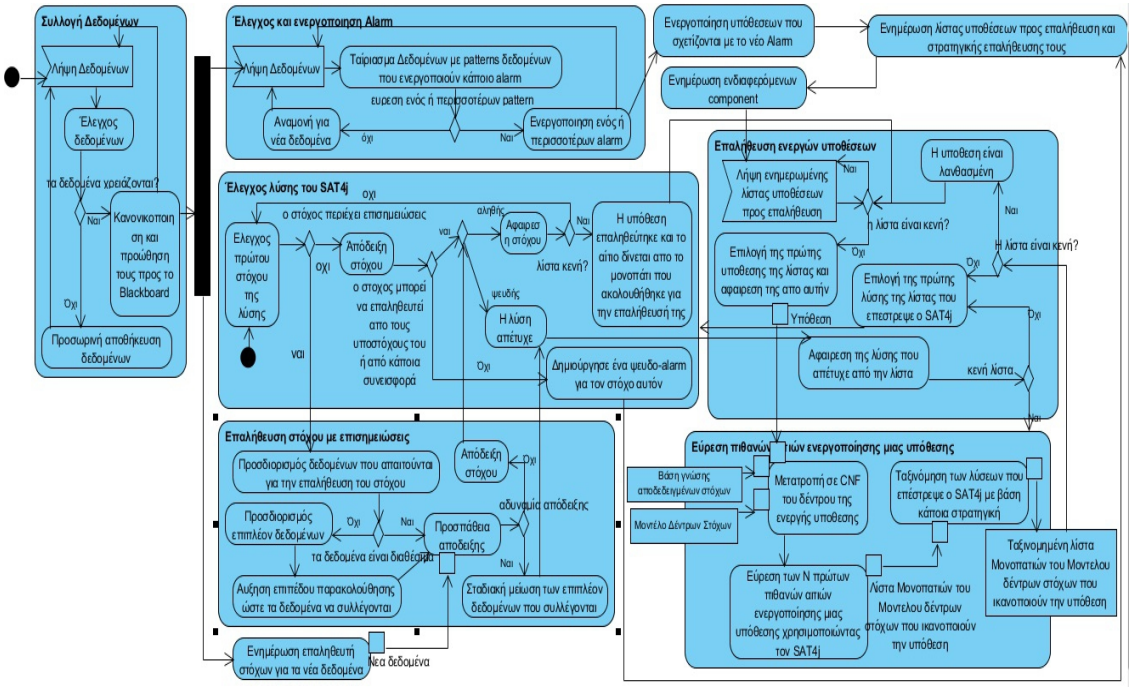
και κατα συνέπεια το ποσοστό των δεδομένων που συνεχίζουμε να συλλέγουμε.

Στην πρωτότυπη υλοποίηση του περιβάλλοντος-πλαισίου δίνεται η δυνατότητα ορισμού της αρχικής τιμή της θερμοκρασίας T και της στρατηγικής αυξησης του k. Η παραπάνω διαδικασία σταματάει όταν το ποσοστό των επιπλέον δεδομένων που παρακολουθούμε πέσει στο μηδέν.

Η επιλογή του αλγορίθμου προσομοιωμένης ανόπτησης για την μείωση του επιπέδου παρακολούθησης έχει το εξής πλεονέκτημα. Ο ρυθμός μείωσης του επιπέδου παρακολούθησης στα πρώτα βήματα εκτέλεσης του αλγορίθμου είναι αργός και αυξάνεται εκθετικά όσο περνά η ώρα. Αυτό συμβαίνει λόγο του αρνητικού εκθετικού όρου στην σχέση που δίνει το ποσοστό των επιπλέον δεδομένων παρακολούθησης που είδαμε παραπάνω. Έτσι αυτό που πετυχαίνουμε είναι στα πρώτα βήματα του αλγορίθμου να είμαστε συντηρητικοί και να δίνουμε μεγάλη πιθανότητα στο ενδεχόμενο να μην ήταν αρκετός ο χρόνος παρακολούθησης των επιπλέον δεδομένων, ενώ όσο περνά η ώρα να είμαστε πιο απότομοι όσον αφορά την μείωση του επιπέδου παρακολούθησης, θεωρώντας ότι η πιθανότητα αυτή μειώνεται εκθετικά. Τέλος η παραμετροποίηση της διαδικασίας μείωσης του επιπέδου παρακολούθησης μέσω της εφαρμογής του αλγορίθμου προσομοιωμένης ανόπτησης μας δίνει την δυνατότητα πειραματιστούμε με διάφορες τιμες για την τιμή του T και της στρατηγικής αυξησης της τιμής του k, με σκοπό να βρεθούν οι κατάλληλες τιμές για κάθε κόμβο του δέντρου.

5.4 Ανάλυση Βασικού Βρόχου Λειτουργίας

Αφού περιγράψαμε την εφαρμογή του SAT Solver και τις ενέργειες που εκτελούνται μετά την λήψη των αποτελεσμάτων του, παρουσιάζουμε πως ενσωματώνονται τα παραπάνω στον βασικό βρόχου ελέγχου του περιβάλλοντος πλαισίου. Ακολουθεί το διάγραμμα δραστηριότητας του βρόχου αυτού:



Σχήμα 5.2: Λεπτομερές Διαγραμμα Δραστηριότητας Βασικού Βρόχου Λειτουργίας

Όπως φαίνεται και από το παραπάνω διάγραμμα η διαδικασία ξεκινά με την λήψη δεδομένων από το υπο παρακολούθηση σύστημα. Στην συνέχεια τα δεδομένα αυτά φιλτράρονται και στέλντονται προς το Blackboard μόνο αν χρειάζονται σε κάποιο component του περιβάλλοντος-πλαισίου. Αλλιώς αποθηκεύονται προσωρινά σε κάποια βάση δεδομένων σε περίπτωση που χρειαστούν για την επαλήθευση κάποιας υπόθεσης που ενεργοποιηθεί στο άμεσο μέλλον.

Τα δεδομένα που στέλνονται προς το Blackboard στην αρχή υποκεινται σε κάποια προεπεξεργασία ώστε να έρθουν σε κάποια πρότυπη μορφή που "καταλαβαίνουν" τα components του περιβάλλοντος. Με αυτόν τον τρόπο παρέχεται η δυνατότητα συλλογής δεδομένων από ποικίλες πηγές. Στην συνέχεια τα δεδομένα αναλύονται για την απόδειξη υποθέσεων και την ενεργοποίηση Alarm.

Αν η παραπάνω διαδικασία δεν οδηγήσει στην ενεργοποίηση κάποιου Alarm τότε το σύστημα περιμένει την συλλογή αλλων δεδομένων. Αν οδηγήσει στην ενεργοποίηση κάποιου Alarm ενεργοποιούνται οι σχετικές με το Alarm υποθέσεις, ενημερώνεται η λίστα προτεραιότήτας επαλήθευσης των υποθέσεων και η στρατηγική επαλήθευσης του. Στην συνέχεια ενημερώνονται τα ενδιαφερόμενα components για την αλλαγή της λιστας προτεραιότητας των υποθέσεων και επιχειρείται η επαλήθευση των υποθέσεων μία προς μία.

Για κάθε ενεργή υπόθεση παράγεται μια CNF έκφραση με την οποία τροφοδοτείται ο SAT4j για να δώσει τα μονοπάτια του μοντέλου δέντρων στόχων που ικανοποιούν την έκφραση αυτή. Για την παραγωγή της έκφρασης αυτής λαμβάνεται υπόψην η βάση γνώσης για τους στόχους που έχουν ήδη αποδειχθεί. Όπως περιγράψαμε και παραπάνω ζητάμε από τον SAT4j μόνο N λύσεις έτσι ώστε να περιοριστεί ο χρόνος εκτέλεσης του SAT4j και να έχουμε μια καλή λύση σε μικρό χρονικό διάστημα. Στην συνέχεια οι λύσεις που επιστρέφονται από τον SAT4j ταξινομούνται με βάση τον αριθμό των στόχων που πρέπει να αποδειχθούν ώστε να επαληθευτεί η υπόθεση. Αν η λίστα τον λύσεων που επέστρεψε ο SAT4j είναι κενή αυτό σημαίνει ότι δεν μπορεί να ικανοποιηθεί η έκφραση CNF που έχει προκύψει γιαυτήν οπότε η υπόθεση αποδεικνύεται ψευδής. Όσο η λίστα δεν είναι κενή και δεν έχει αποδειχθεί η υπόθεση επιχειρείται η εφαρμογή της πρώτης λύσης και αν αποτείχει αφαιρείται από την λίστα και συνεχίζουμε με την επόμενη που λύση που είναι πλέον στην αρχή της λίστας. Αν η λίστα των λύσεων του SAT4j εξαντληθεί σημαίνει ότι και N λύσεις που επεστρεψε ο SAT4j αποτύχαν. Έτσι πρέπει να ξανατρέξουμε τον SAT4j με την νέα βάση γνώσης για να μας δωσει ένα νέο σύνολο λύσεων. Η διαδικασία σταματάει αν επιτευχθεί η επαλήθευση της υπόθεσης ή αν δεν επιστραφεί λύση από την SAT4j. Αν επιτύχει κάποια λύση, επαληθεύεται η υπόθεση και αφαιρείται από την λίστα προτεραιότητας των ενεργών υποθέσεων. Το μονοπάτι του δέντρου στόχων που ακολουθήθηκε για να αποδειχθεί η υπόθεση αποτελεί την διάγνωση της αιτίας που οδήγησε στην ενεργοποίηση της υπόθεσης αυτής.

Ο έλεγχος κάθε λύσης που επέστρεψε ο SAT4j απαιτεί την επαλήθευση των αληθοτιμών όλων των υποστόχων που αποτελούν την λύση. Αν επαληθευτούν όλοι οι υποστόχοι τότε επαληθεύεται η υποθεση. Αν αποτύχει η επαλήθευση έστω και ενός υποστόχου τότε απορρίπτεται η λύση και συνεχίζουμε με την επόμενη. Η προσπάθεια επαλήθευσης ενός στόχου μπορεί να έχει τρία πιθανά αποτελέσματα.

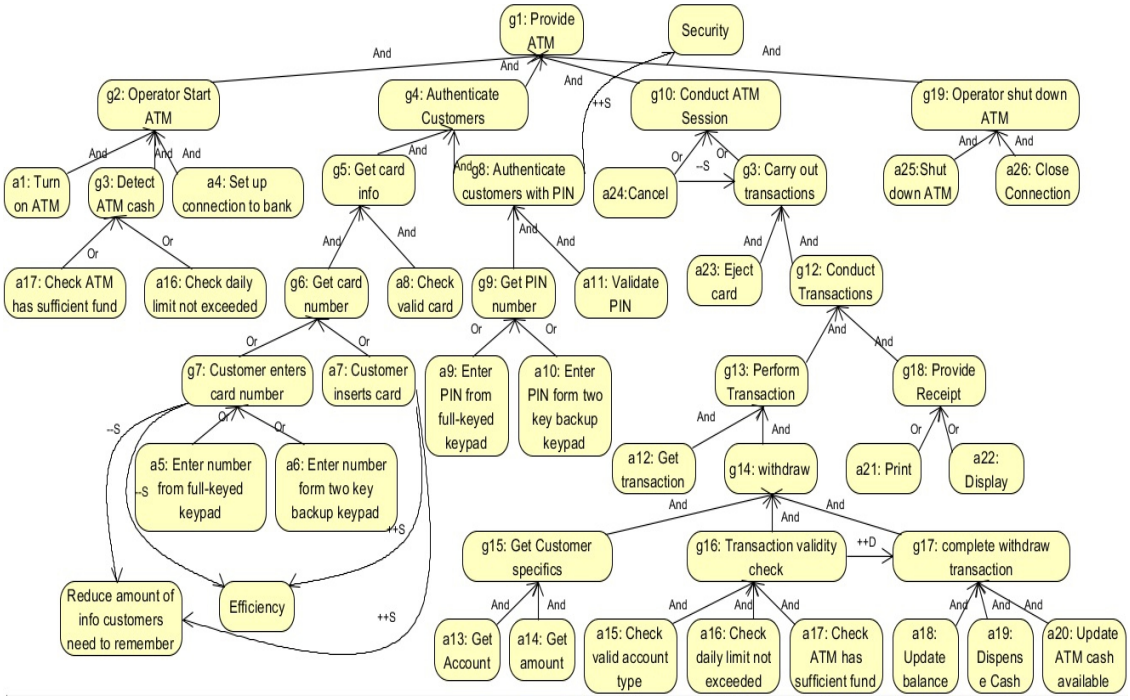
1. Ο στόχος μπορεί να επαληθευτεί απευθείας από τους υποστόχους του.

2. Ο στόχος είναι ενδιάμεσος κόμβος αλλά οι αληθοτιμές των υποστόχων του δεν είναι γνωστές. Σε αυτήν την περίπτωση θέλουμε να επωφεληθούμε της ήδη υπάρχουσας υποδομής και έτσι χειριζόμαστε τον στόχο αυτόν σαν μια υπόθεση αφήνοντας τον SAT4j να βρεί έναν τρόπο για την επαλήθευση του στοχου αυτού. Για να το κάνουμε αυτό δημιουργούμε ένα ψευδο-alarm για τον στόχο αυτόν και το προσθέτουμε στην λίστα προτεραιότητας των ενεργών υποθέσεων. Η διαδικασία συνεχίζει όπως περιγράφηκε παραπάνω.

3. Ο στόχος είναι φύλλο ενός δέντρου. Στην περίπτωση αυτήν ο στόχος μπορεί να επαληθευτεί μόνο με αξιοποίηση των επισημειώσεών του. Οι επισημειώσεις δίνουν τα επιπλεον δεδομένα που πρέπει να συλλεγούν καθώς επίσης και τις συνθήκες που πρέπει να πληρούν ώστε να αποδείξουν τον αντίστοιχο στόχο αληθή ή ψευδή. Επόμενο λοιπόν βήμα είναι ο προσδιορισμός των επιπλέον δεδομένων. Αν αυτά τα δεδομένα είναι διαθέσιμα τότε επιχειρείται η απόδειξη του στόχου. Αν όχι αυξάνεται το επίπεδο παρακολούθησης και ανακτώνται τα επιπλέον δεδομένα. Σε κάθε περίπτωση επιχειρείται η απόδειξη του στόχου με δύο πιθανά αποτελέσματα. Ο στόχος είτε αποδεικνύεται αληθής ή ψευδής με χρήση των δεδομένων παρακολούθησης και των επισημειώσεων είτε δεν μπορει να αποδειχθεί. Στην πρώτη περίπτωση ο στόχος αποδεικνύεται και συνεχίζουμε με τον επομενο στόχο της λυσης. Στην δευτερη περίπτωση μειώνουμε σταδιακά το επίπεδο παρακολούθησης σταδιακά όπως περιγραψαμε στην προηγούμενα ενότητα.

5.5 Μελέτη περίπτωσης

Σε αυτήν την ενότητα δίνουμε ένα παράδειγμα χρήσης του προτεινόμενου περιβάλλοντος-πλαισίου για να φανεί η λειτουργία του. Ως παράδειγμα επιλέξαμε την περίπτωση του ATM. Το μοντέλο δέντρων στόχων που χρησιμοποιήσαμε βασίζεται στο μοντέλο που παρουσιάζεται στο [87].



Σχήμα 5.3: Μοντέλο Δεντρων Στόχων Λειτουργίας του ATM[87]

Για τις ανάγκες τις παρουσίασης της λειτουργίας του περιβάλλοντος-πλαισίου δόθηκε το παραπάνω μοντέλο δέντρων στόχων ως είσοδος στο περιβάλλον και ενεργοποιήθηκε ένα alarm για την ρίζα του δέντρου ώστε να επιχειρηθεί η επαλήθευση της υποθεσης ότι υπάρχει βλάβη στο ATM και να γίνει ανάλυση του αιτίου της. Δεδομένου ότι το περιβάλλον-πλαισιο δεν έχει ολοκληρωθεί με κάποιο σύστημα προς παρακολούθηση δεν υπάρχουν δεδομένα για την απόδειξη των στόχων-φύλλων του δέντρου. Για τον λόγο αυτό έχουμε αποφασίσει από πριν τις

αληθοτιμές των φύλλων τις οποίες το σύστημα λαμβάνει από ένα αρχείο διαμόρφωσης (configuration file). Οι αληθοτιμή έχει τεθεί αληθής για όλα τα φύλλα εκτός από τα φύλλα a3, a6, a10, a11, a22 και a24. Έτσι η υπόθεση είναι αληθής ή αντίστοιχα η ρίζα του δέντρου (g1) πρέπει να αποδειχθεί ψευδής όπως και συμβαίνει (*Event67*).

Για την παρουσίαση της λειτουργία του περιβάλλοντος-πλαισίου παρέχουμε το trace που παράγεται κατα την εκτέλεση του με το μοντέλο που παρουσιάστηκε παραπάνω. Εχουμε επισημειώσει κάθε βήμα της διαδικασίας με το όνομα "Event" ακολουθούμενο από έναν αριθμό έτσι ώστε να μπορούμε να αναφερόμαστε σε μεμονωμένα βήματα του διαδικασίας. Ο αναγνωστης μπορεί να ρίξει μια ματιά στο trace της εκτέλεσης που δίνεται παρακάτω, θα θέλαμε ωστόσο να σχολιάσουμε τα βασικά του σημεία.

Όπως φαίνεται από το trace η διαδικασία ξεκινά με την ενεργοποίηση του alarm και των συσχετιζόμενων στόχων του μοντέλου (σε αυτήν την περίπτωση μόνο του g1) (Event1). Στην συνέχεια καλείται ο SAT4j για τον g1 και ενημερώνεται η λίστα των στόχων προς επαλήθευση (Event2). Η πρώτη λύση που επιχειρείται, απαιτεί την επαλήθευση των στόχων a8, a25, a7, a26, a3, a10, a24, a4, a11, a1, g3, g2, g6, g5, g19, g9, g8, g10 και g4. Η λύση αυτή θα αποτύχει καθώς όπως είπαμε ο a3 είναι ψευδής (Event32). Σε αυτό το σημείο τονίζουμε ότι η λίστα των στόχων είναι ταξινομημένη έτσι ώστε να επαληθευτούν πρώτα τα φύλλα και εν συνεχεία οι ενδιάμεσοι κόμβοι. Επιχειρειται η επαλήθευση του κόμβου a8 και αυξάνεται το επίπεδο παρακολούθησης για την επαλήθευση του (Event). Κατα την προσπάθεια επαλήθευσης του g1 με την χρήση της παραπάνω λύσης αποδεινκύονται οι στόχοι a8, a25, a7, a26 και a3 (Event6, Event13, Event17, Event29 και Event32). Για καθέ εναν απο τους παραπάνω στόχους αυξάνεται το επίπεδο παρακολούθησης για την επαλήθευση του και μειώνεται σταδιακά όπως περιγράψαμε παραπάνω. Μόλις αποδειχθεί η αρνηση του στόχου a3 η λύση απορρίπτεται και δεν επιχειρείται η επαλήθευση των επόμενων στόχων.

Η λύση αποτυγχάνει και επιχειρείται η επαλήθευση του στόχου g1 με την επαλήθευση των στόχων a8, a25, a2, a26, a7, a10, a24, a4, a11, a1, g3, g2, g19, g5, g9, g8, g6, g10 και g4. Η λύση αυτή επίσης θα αποτύχει λόγω του στόχου a10 που όπως αναφέραμε είναι ψευδής (Event57). Ως αποτέλεσμα της εφαρμογής αυτής της λύσης θα αποδειχθούν οι κόμβοι a2 και a10 (Event53 και Event57). Οι επόμενοι στόχοι της λίστας δεν ελέγχονται ομοίως με την προηγούμενη περίπτωση.

Τέλος επιχειρείται η επαλήθευση του g1 με την επαλήθευση των στόχων a8, a2, a7, a26, a11, a1, a25, a24, a9, a4, g3, g2, g6, g5, g10, g9, g8, g19 και g4 η οποία επιτυγχάνει. Τέλος επισημαινουμε ότι για κάθε στοχο φύλλο του δέντρου που επιχειρούμε να επαληθεύσουμε αυξάνεται το επίπεδο παρακολούθησης ώστε να συλλεγούν τα απαραίτητα δεδομένα.

Ως τελευταίο σχόλιο αναφέρουμε ότι η χρήση του αλγορίθμου προσομοιωμένης ανόπτησης για την επαναφορά του επιπέδου παρακολούθησης είναι εμφανής αν κανείς παρατηρήσει ότι η μείωση είναι αργή στην αρχή και αυξάνεται απότομα καθώς ο χρόνος περνά. Για παράδειγμα στην περίπτωση του κόμβου a8 παρατηρούμε ότι το επίπεδο παρακολουθησης πέφτει από 100% σε 90% μέσα σε 55 δευτερόλεπτα, στην συνέχεια από 90% σε 70% σε 15 δευτερόλεπτα, από 70% σε 40% σε 10 δευτερόλεπτα, απο 40% σε 20 σε 5 δευτερόλεπτα και από 20% σε 0% σε 5 δευτερόλεπτα. Η αυξηση του ρυθμού μείωσης του επιπέδου παρακολούθησης όσο περνά ο χρόνος ήταν ενα από τα βασικά προτερήματα της χρήσης του αλγορίθμου προσομοιωμένης ανόπτησης.

Event1: Wed Jan 29 16:52:01 EET 2014: <<Alarmer>> Alarm Manage ATM Problem was activated - (Associated Goals: g1: Manage ATM)
Event2: Wed Jan 29 16:52:01 EET 2014: <<GoalSelector>> Generates solutions for goal g1: Manage ATM using sat solver and updates evaluation queue
Event3: Wed Jan 29 16:52:01 EET 2014: <<Evaluator>> tries to prove goal g1: Manage ATM by proving the following goals:
a8: Check valid card
a25: Shut down ATM
a7: Customer inserts card
a26: Close connection
a3: Operator enters cash available
a10: Enter PIN from two key keypad
a24: cancel
a4: Set up connection to bank
a11: Validate PIN
a1: Turn on ATM
g3: Detect ATM cash
g2: Start ATM
g6: Get card number
g5: Get card info
g19: Shutdown ATM
g9: Get PIN number
g8: Authenticate Customer with pin
g10: Conduct ATM session
g4: Authenticate Customer
Event4: Wed Jan 29 16:52:01 EET 2014: <<Evaluator>> requests additional data for a8: Check valid card
Event5: Wed Jan 29 16:52:01 EET 2014: <<MonitoringLevelReasoner>> increasing monitoring. Fetching additional data for goal a8: Check valid card
Event6: Wed Jan 29 16:52:16 EET 2014: <<Evaluator>> to prove goal g1: Manage ATM proves goal a8: Check valid card true
Event7: Wed Jan 29 16:52:16 EET 2014: <<Evaluator>> requests additional data for a25: Shut down ATM
Event8: Wed Jan 29 16:52:16 EET 2014: <<MonitoringLevelReasoner>> increasing monitoring. Fetching additional data for goal a25: Shut down ATM
Event9: Wed Jan 29 16:52:56 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a8: Check valid card to 90.0%
Event10: Wed Jan 29 16:53:06 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a25: Shut down ATM to 90.0%
Event11: Wed Jan 29 16:53:11 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a8: Check valid card to 70.0%
Event12: Wed Jan 29 16:53:11 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a25: Shut down ATM to 80.0%
Event13: Wed Jan 29 16:53:16 EET 2014: <<Evaluator>> to prove goal g1: Manage ATM proves goal a25: Shut down ATM true
Event14: Wed Jan 29 16:53:16 EET 2014: <<Evaluator>> requests additional data for a7: Customer inserts card
Event15: Wed Jan 29 16:53:16 EET 2014: <<MonitoringLevelReasoner>> increasing monitoring. Fetching additional data for goal a7: Customer inserts card
Event16: Wed Jan 29 16:53:16 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a25: Shut down ATM to 70.0%
Event17: Wed Jan 29 16:53:18 EET 2014: <<Evaluator>> to prove goal g1: Manage ATM proves goal a7: Customer inserts card true
Event18: Wed Jan 29 16:53:18 EET 2014: <<Evaluator>> requests additional data for a26: Close connection
Event19: Wed Jan 29 16:53:18 EET 2014: <<MonitoringLevelReasoner>> increasing monitoring. Fetching additional data for goal a26: Close connection
Event20: Wed Jan 29 16:53:21 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a8: Check valid card to 40.0%
Event21: Wed Jan 29 16:53:21 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a25: Shut down ATM to 50.0%
Event22: Wed Jan 29 16:53:26 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a8: Check valid card to 20.0%
Event23: Wed Jan 29 16:53:26 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a25: Shut down ATM to 20.0%
Event24: Wed Jan 29 16:53:31 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a8: Check valid card to 0.0%
Event25: Wed Jan 29 16:53:36 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a25: Shut down ATM to 0.0%
Event26: Wed Jan 29 16:53:56 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a7: Customer inserts card to 90.0%
Event27: Wed Jan 29 16:54:11 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a7: Customer inserts card to 80.0%
Event28: Wed Jan 29 16:54:16 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a7: Customer inserts card to 70.0%
Event29: Wed Jan 29 16:54:18 EET 2014: <<Evaluator>> to prove goal g1: Manage ATM proves goal a26: Close connection true
Event30: Wed Jan 29 16:54:18 EET 2014: <<Evaluator>> requests additional data for a3: Operator enters cash available
Event31: Wed Jan 29 16:54:18 EET 2014: <<MonitoringLevelReasoner>> increasing monitoring. Fetching additional data for goal a3: Operator enters cash available
Event32: Wed Jan 29 16:54:21 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a7: Customer

inserts card to 30.0%
Event33: Wed Jan 29 16:54:26 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a7: Customer inserts card to 0.0%
Event34: Wed Jan 29 16:54:28 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a26: Close connection to 90.0%
Event35: Wed Jan 29 16:54:38 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a26: Close connection to 70.0%
Event36: Wed Jan 29 16:54:43 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a26: Close connection to 40.0%
Event37: Wed Jan 29 16:54:48 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a26: Close connection to 10.0%
Event38: Wed Jan 29 16:54:53 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a26: Close connection to 0.0%
Event39: Wed Jan 29 16:55:18 EET 2014: <<Evaluator>> to prove goal g1: Manage ATM proves goal a3: Operator enters cash available false. Evaluation of g1: Manage ATM fails. Evaluator will check another solution
Event40: Wed Jan 29 16:55:18 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a3: Operator enters cash available to 90.0%
Event41: Wed Jan 29 16:55:18 EET 2014: <<Evaluator>> tries to prove goal g1: Manage ATM by proving the following goals:
a8: Check valid card
a25: Shut down ATM
a2: Sensor senses cash available
a26: Close connection
a7: Customer inserts card
a10: Enter PIN from two key keypad
a24: cancel
a4: Set up connection to bank
a11: Validate PIN
a1: Turn on ATM
g3: Detect ATM cash
g2: Start ATM
g19: Shutdown ATM
g5: Get card info
g9: Get PIN number
g8: Authenticate Customer with pin
g6: Get card number
g10: Conduct ATM session
g4: Authenticate Customer
Event42: Wed Jan 29 16:55:18 EET 2014: <<Evaluator>> requests additional data for a2: Sensor senses cash available
Event43: Wed Jan 29 16:55:18 EET 2014: <<MonitoringLevelReasoner>> increasing monitoring. Fetching additional data for goal a2: Sensor senses cash available
Event44: Wed Jan 29 16:55:28 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a3: Operator enters cash available to 80.0%
Event45: Wed Jan 29 16:55:38 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a3: Operator enters cash available to 60.0%
Event46: Wed Jan 29 16:55:48 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a3: Operator enters cash available to 20.0%
Event47: Wed Jan 29 16:55:48 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a2: Sensor senses cash available to 90.0%
Event48: Wed Jan 29 16:55:53 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a3: Operator enters cash available to 0.0%
Event49: Wed Jan 29 16:55:53 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a2: Sensor senses cash available to 80.0%
Event50: Wed Jan 29 16:56:03 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a2: Sensor senses cash available to 60.0%
Event51: Wed Jan 29 16:56:08 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a2: Sensor senses cash available to 20.0%
Event52: Wed Jan 29 16:56:13 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a2: Sensor senses cash available to 0.0%
Event53: Wed Jan 29 16:56:18 EET 2014: <<Evaluator>> to prove goal g1: Manage ATM proves goal a2: Sensor senses cash available true
Event54: Wed Jan 29 16:56:18 EET 2014: <<Evaluator>> requests additional data for a10: Enter PIN from two key keypad
Event55: Wed Jan 29 16:56:18 EET 2014: <<MonitoringLevelReasoner>> increasing monitoring. Fetching additional data for goal a10: Enter PIN from two key keypad
Event56: Wed Jan 29 16:56:58 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a10: Enter PIN from two key keypad to 90.0%
Event57: Wed Jan 29 16:56:59 EET 2014: <<Evaluator>> to prove goal g1: Manage ATM proves goal a10: Enter PIN from two key keypad false. Evaluation of g1: Manage ATM fails. Evaluator will check another solution
Event58: Wed Jan 29 16:56:59 EET 2014: <<Evaluator>> tries to prove goal g1: Manage ATM by proving the following goals:
a8: Check valid card
a2: Sensor senses cash available
a7: Customer inserts card
a26: Close connection
a11: Validate PIN
a1: Turn on ATM
a25: Shut down ATM

a24: cancel
a9: Enter PIN from full-keyed keypad
a4: Set up connection to bank
g3: Detect ATM cash
g2: Start ATM
g6: Get card number
g5: Get card info
g10: Conduct ATM session
g9: Get PIN number
g8: Authenticate Customer with pin
g19: Shutdown ATM
g4: Authenticate Customer

Event59: Wed Jan 29 16:56:59 EET 2014: <<Evaluator>> requests additional data for a11: Validate PIN
Event60: Wed Jan 29 16:56:59 EET 2014: <<MonitoringLevelReasoner>> increasing monitoring. Fetching additional data for goal a11: Validate PIN
Event61: Wed Jan 29 16:57:08 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a10: Enter PIN from two key keypad to 80.0%
Event62: Wed Jan 29 16:57:18 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a10: Enter PIN from two key keypad to 50.0%
Event63: Wed Jan 29 16:57:23 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a10: Enter PIN from two key keypad to 20.0%
Event64: Wed Jan 29 16:57:33 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a10: Enter PIN from two key keypad to 0.0%
Event65: Wed Jan 29 16:57:34 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a11: Validate PIN to 90.0%
Event66: Wed Jan 29 16:57:40 EET 2014: <<Evaluator>> to prove goal g1: Manage ATM proves goal a11: Validate PIN false. Evaluation of g1: Manage ATM fails. Evaluator will check another solution
Event67: Wed Jan 29 16:57:40 EET 2014: <<Evaluator>> all solutions for g1: Manage ATM failed. g1: Manage ATM is false.
Event68: Wed Jan 29 16:57:44 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a11: Validate PIN to 80.0%
Event69: Wed Jan 29 16:57:49 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a11: Validate PIN to 50.0%
Event70: Wed Jan 29 16:57:59 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a11: Validate PIN to 20.0%
Event71: Wed Jan 29 16:58:04 EET 2014: <<MonitoringLevelReasoner>> monitoring of additional data for goal a11: Validate PIN to 0.0%

Κεφάλαιο 6: Αξιολόγηση του περιβάλλοντος-πλαισίου

Στο κεφάλαιο αυτό εξετάζουμε την επίδοση του περιβάλλοντος-πλαισίου που αναπτύξαμε. Μετρήσαμε τον χρόνο εκτέλεση του και τις απαιτήσεις σε μνήμη για διαφορέτικά μεγέθη μοντέλων. Επιπλέον εξετάζουμε την επίδραση των συνεισφορών στον χρόνο εκτέλεσης.

Για τις ανάγκες της αξιολόγησης του περιβάλλοντος-πλαισίου δημιουργήσαμε τυχαία μοντέλα δέντρων στόχων μεγέθους διαφορέτικού πλήθους δέντρων κρατώντας σταθερό τον αριθμό των κόμβων κάθε δέντρου σε 20 κόμβους. Στην συνέχεια δημιουργήσαμε τυχαια συσχετίσεις μεταξύ των κόμβων των δέντρων.

Δεδομένου ότι το περιβάλλον-πλαίσιο που αναπτύξαμε δεν ολοκληρώθηκε με κάποιο σύστημα υπο παρακολούθηση, δεν υπάρχει συλλογή και ανάλυση δεδομένων. Ως εκ τούτου η απόδειξη των ατομικών στόχων έγινε με χρήση αρχείων διαμόρφωσης (configuration files) τα οποία παρείχαν τις αληθοτιμές καθε ατομικού στόχου. Δημιουργήσαμε ένα alarm για κάθε ρίζα δέντρου και τα ενεργοποιήσαμε όλα ένα προς ένα έτσι ώστε να επαληθευτούν όλα τα δέντρα. Ο χρόνος που μετράμε ξεκινά με την ενεργοποιηση του πρώτου alarm και τελειώνει με την απόδειξη της τελευταίας υπόθεσης.

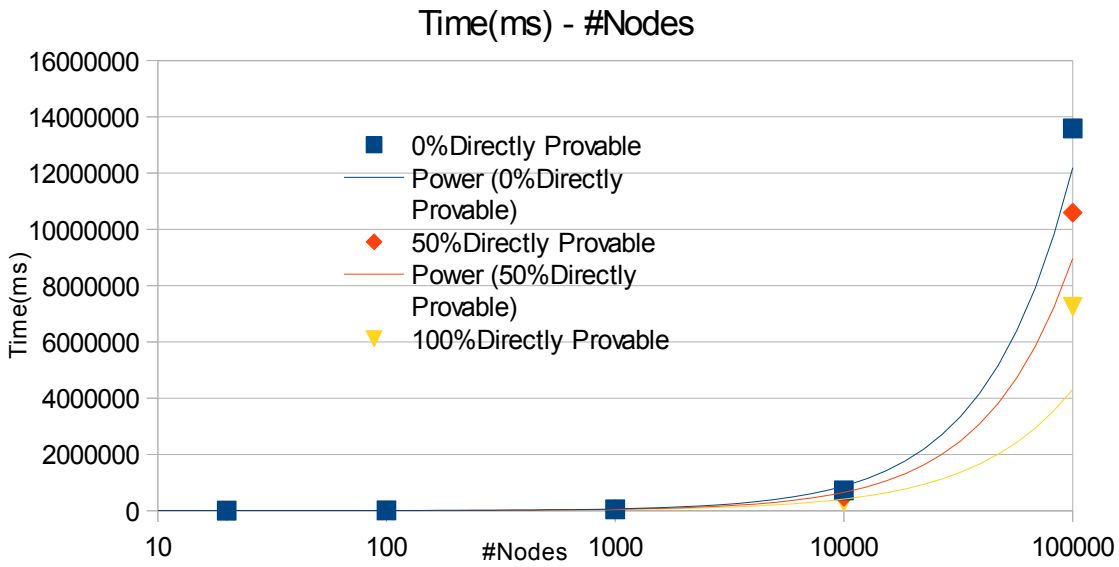
Όσον αφορά τον χρόνο απόδειξης ενός ατομικού στόχου αυτός προκύπτει από τον χρόνο που απαιτεί ο έλεγχος τον δεδομένων παρακολούθησης με τους κανόνες που πρέπει να ικανοποιούνται για την απόδειξη του στόχου και από τον χρόνο που απαιτείται για την ανάκτηση των απαιτούμενων δεδομένων. Αυτός ο χρόνος προσομοιώθηκε βάζοντας το thread που επιχειρεί την επαλήθευση του στόχου, σε αναμονή (sleep) για κάποιον ορισμένο χρόνο. Ο χρόνος αυτός ήταν 100ms στην περίπτωση που τα δεδομένα ήταν διαθέσιμα και 300ms στην αντίθετη περίπτωση. Αυτή η διαδικασία μας έδωσε μια καλή εικόνα της επίδοσης της συλλογιστικής διαδικασίας. Ωστόσο οι χρόνοι αυτοί δεν προσεγγίζουν τους πραγματικούς καθώς δεν λαμβάνουν υπόψην τον όγκο των συλλεγόμενων δεδομένων. Για τον λόγο αυτόν προσεγγίσαμε των πραγματικό χρόνο επαλήθευσης ενός στόχου μέσω της σχέσης $T=(N*M)/1000+c$, όπου N είναι το πλήθος των δεδομένων παρακολούθησης που επεξεργαζόμαστε, M είναι το μέγεθος των μορφημάτων γεγονότων που αποδεικνύουν τον στόχο και c μια σταθερά. Το N παίρνει τιμές από 10.000 έως 10.000.000, το M από 5 έως 10 και το c από 1 έως 100. Οι τιμές των N, M και c για κάθε κόμβο παρέχονται από κάποιο αρχείο διαμόρφωσης.

6.1 Επίδραση Μεγέθους Μοντέλου Δέντρων Στοχων στην επίδοση του περιβάλλοντος-πλαισίου

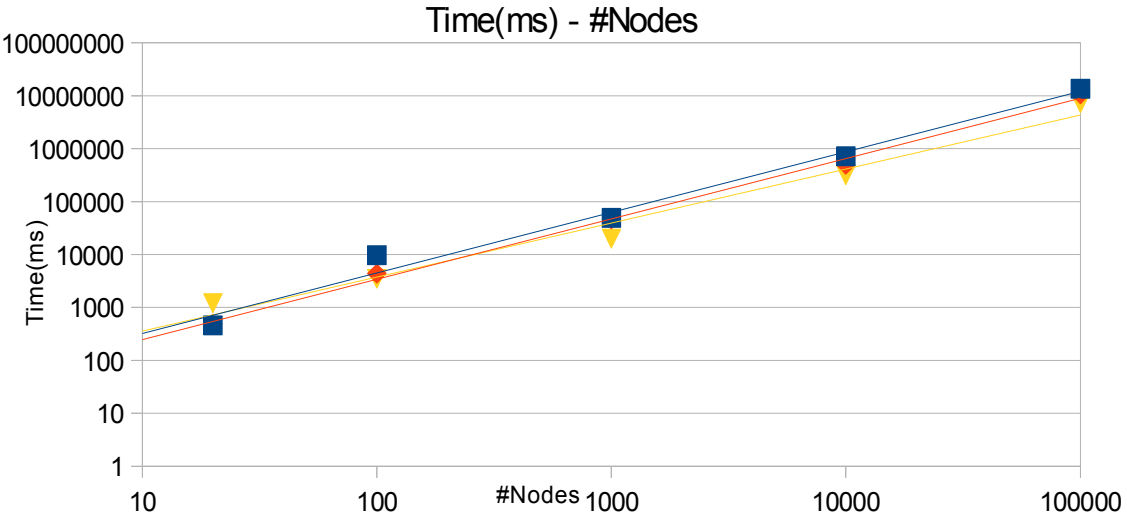
Σε αυτήν την ενότητα βλέπουμε την επίδραση του μεγέθους του μοντέλου δέντρων στόχων στην επίδοση του περιβάλλοντος-πλαισίου. Τα μοντέλα που εξετάσαμε έχουν μέγεθος 20, 100,

1000, 10000, και 100000 κόμβων.

Στο παρακάτω διάγραμμα βλέπουμε την σχέση του χρόνου εκτέλεσης ως προς τον αριθμό των κόμβων του μοντέλου. Οι τρεις γραφικές παραστάσεις που προκύπτουν, αναφέρονται στις περιπτώσεις όλα τα επιπλεόν δεδομένα για την απόδειξη των ατομικών φύλλοων να είναι διαθέσιμα εκ των προτέρων (κίτρινη γραμμή), 50% των δεδομένων να είναι διαθέσιμα εκ των προτέρων (πορτοκαλί γραμμή) και κανένα από τα επιπλέων δεδομένα να μην είναι διαθέσιμα (μπλε γραμμή). Όπως παρατηρούμε από το παρακάτω διάγραμμα η αυξηση του χρόνου είναι γραμμική ως προς την αύξηση του μεγέθους των δέντρων στόχων. Για να γίνει αυτό εμφανές παρουσιάζεται το ίδιο διάγραμμα σε λογαριθμική κλίμακα. Από το πρώτο διάγραμμα γίνεται εμφανής η επίδραση της διαθεσιμότητας τον δεδομένων, στον χρόνο εκτέλεσης ενώ από το δεύτερο διάγραμμα φαίνεται η γραμμική συσχέτιση των δύο μεγεθών.

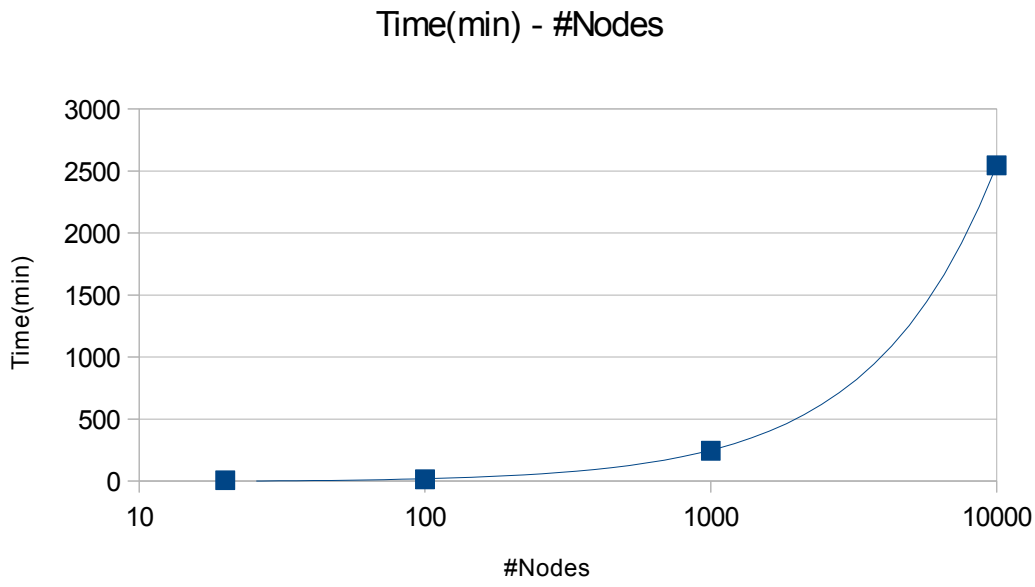


Σχήμα 6.1.1: Διάγραμμα Χρόνου ως προς τον Αριθμό των κόμβων του μοντέλου



Σχήμα 6.1.2: Διάγραμμα Χρόνου ως προς τον Αριθμό των κόμβων του μοντέλου (Λογαριθμική Κλίμακα)

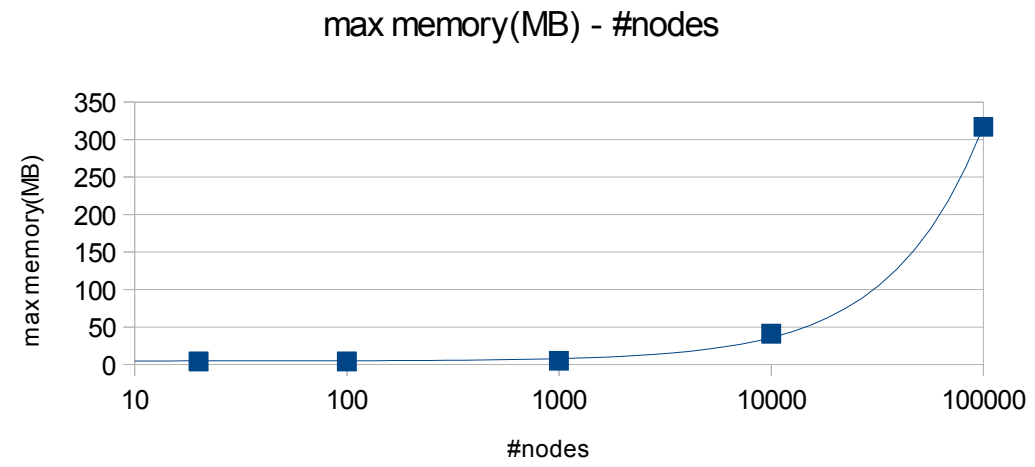
Το ίδιο διάγραμμα δίνεται για χρόνο αναμονής που δίνεται από την σχέση T=(N*M)/1000+c.



Σχήμα 6.1.3: Διάγραμμα Χρόνου ως προς τον Αριθμό των κόμβων του μοντέλου (Προσέγγιση χρόνου ανάλυσης μέσω της σχέση T=(N*M)/1000+c)

Από αυτό το διάγραμμα φαίνεται ότι για ένα μοντέλο 1000 κόμβων αναμένεται περίπου 200 λέπτα μέχρι την απόδειξη όλων των υποθέσεων, ενώ για μοντέλα 10000 κόμβων η τεχνική είναι ασύμφορη.

Το επόμενο διάγραμμα παρουσιάζει τις απαιτήσεις σε μνήμη για διαφορετικά μεγέθη μοντέλων δέντρων στόχων. Η μνήμη που καταναλώνεται όπως βλέπουμε είναι σχεδόν γραμμική και είναι σχετικά χαμηλή ακόμα και για μοντέλα 100.000 κόμβων.

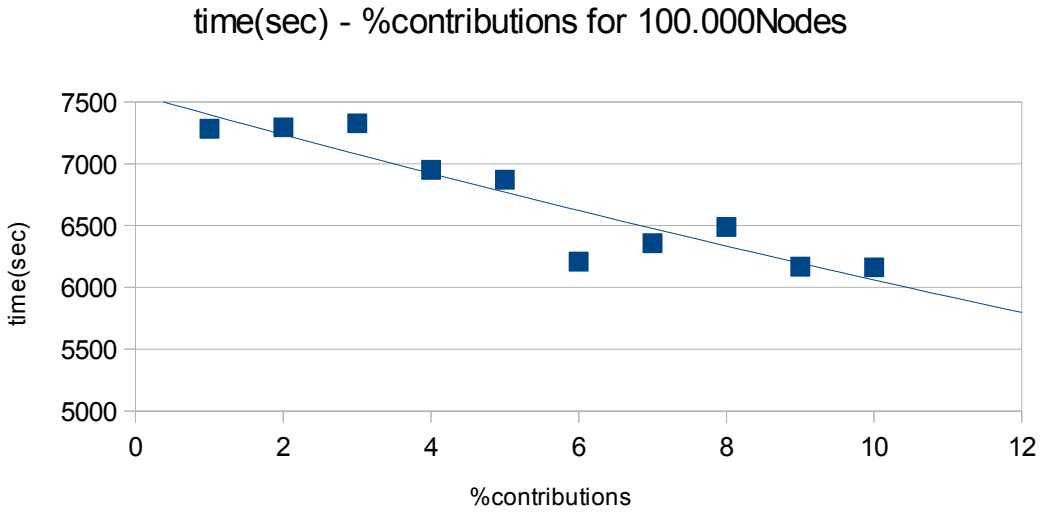


Σχήμα 6.1.4: Διάγραμμα Χρήσης RAM ως προς τον Αριθμό των κόμβων του μοντέλου

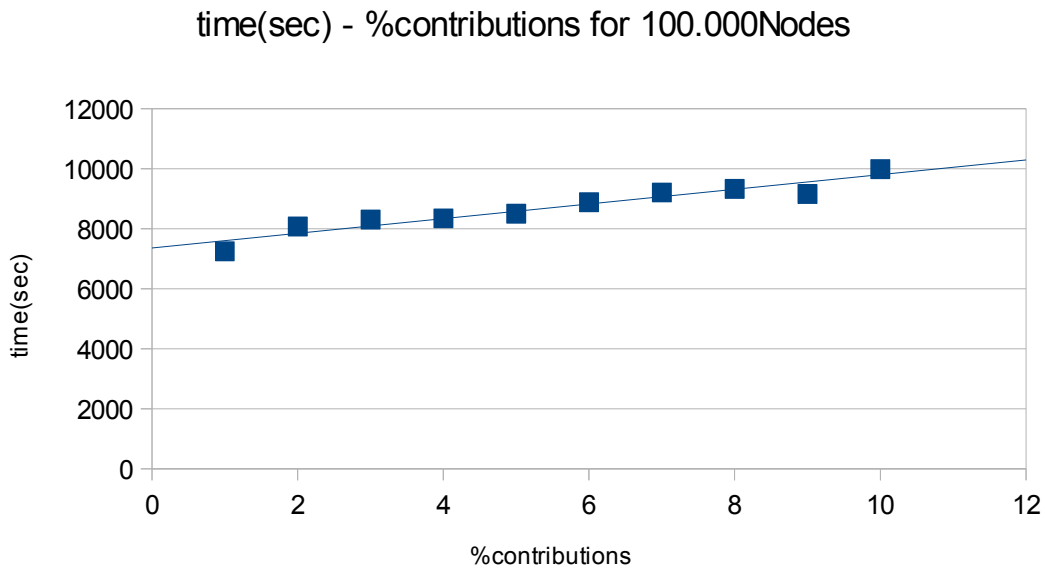
6.2 Επίδραση των συνεισφορών στην επίδοση του περιβάλλοντος-πλαισίου

Σε αυτήν την ενότητα παρουσιάζουμε την επίδραση των συνεισφορών στην επίδοση του περιβάλλοντος-πλαισίου.

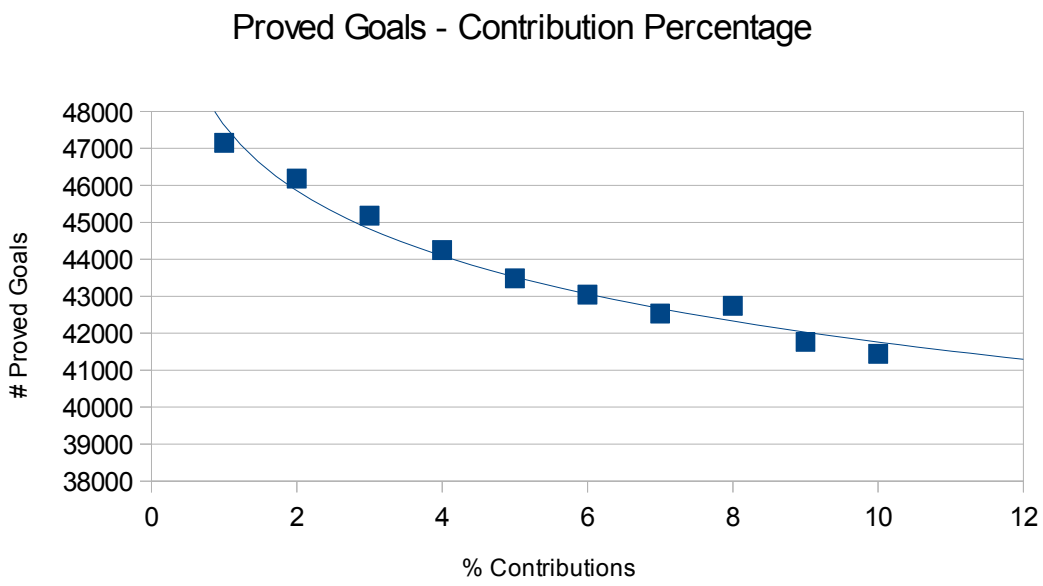
Το πρώτο διάγραμμα που παρουσιάζουμε δείχνει επίδραση συνεισφορών για μοντέλα 100.000 κόμβων με 1%, 2%, 3% έως 10% συνεισφορές. Στο παράδειγμα αυτό ότι κόμβος αποδεικνύεται θεωρείται γνωστή η τιμή του μέχρι το τέλος της διαδικασίας. Όπως φαίνεται από το παρακάτω διάγραμμα όσο αυξάνεται το ποσοστό των συνεισφορών τόσο μειώνεται ο χρόνος που απαιτείται για την επαλήθευση των υποθέσεων. Αυτό μπορεί να μοιάζει παράξενο με την πρώτη ματιά καθώς ίσως να ανέμενε κανείς αύξηση του χρόνου εκτέλεσης του SAT4j καθώς οι παραπάνω συνεισφορές θα εισάγουν παραπάνω κανόνες στην έκφραση CNF. Αυτό είναι όντως συμβαίνει και φαίνεται από το μεθεπόμενο διάγραμμο όπου κάθε κόμβος παραμένει αληθής μόνο για 10 δευτερόλεπτα. Όπως φαίνεται από αυτό διάγραμμα όσο αυξάνεται το ποσοστό των συνεισφορών τόσο αυξάνεται ο χρόνος για την απόδειξη όλων των υποθέσεων. Ωστόσο η μείωση του χρόνου επαλήθευσης των υποθέσεων, στην πρώτη περίπτωση, είναι απόλυτα λογικό. Αυτό συμβαίνει διότι πολλές φορές παρακάμπτεται η απόδειξη κόμβων του μοντέλου από τα υποδέντρα τους και χρησιμοποιείται η τιμή κάποιου ήδη αποδεδειγμένου στόχου. Έτσι μειώνεται ο αριθμός των κόμβων που πρέπει να αποδειχθουν τελικά. Αυτό φαίνεται από το διάγραμμα 6.6 στο οποίο παρουσιάζεται η συσχέτιση μεταξύ των κομβων που αποδεικνύονται και του αριθμού των συνεισφορών. Από το διάγραμμα αυτό φαίνεται ότι όσο αυξανεται ο αριθμός των συνεισφορών τόσο μειώνεται ο αριθμός των κόμβων που αποδεικνύονται για την επαλήθευση όλων των υποθέσεων του μοντέλου. Τέλος στο διάγραμμα 6.7 παρουσιάζεται η συσχέτιση του αριθμού των στόχων που αποδεικνύονται για την επαλήθευση όλων των υποθέσεων και του ποσοστού των συνεισφορών στην περίπτωση που η αληθοτιμή καθε στόχου παραμένει γνωστή μόνο για 10 δευτερόλεπτα. Όπως φαίνετα από αυτό το διάγραμμα όσο αυξάνεται ο αριθμός των συνεισφορών τόσο αυξανεται ο αριθμός των στοχων που αποδεικνύονται. Αυτό συμβαίνει γιατί καποιος στοχος μπορεί να αποδεικνύεται παραπάνω από μία φορές. Μπορεί να αποδεικνύεται για την επαλήθευση της υπόθεσης με ρίζα το δέντρο στο οποίο ανήκει ο στόχος αυτός αλλά και για την απόδειξη των στόχων στους οποίους συνεισφέρει.



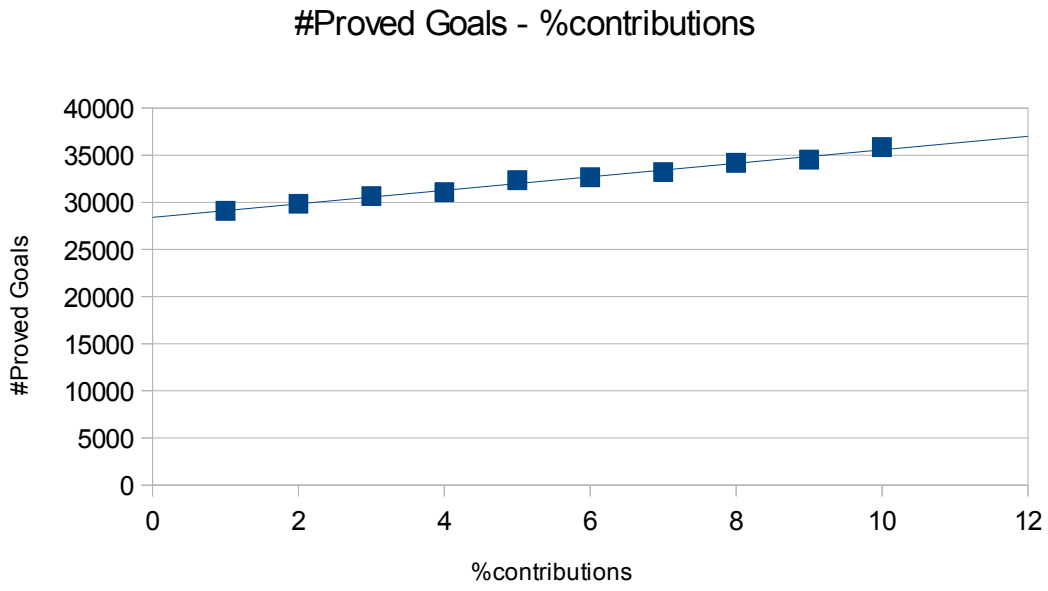
Σχήμα 6.2.1: Διάγραμμα Χρόνου ως προς το Ποσοστό των Συνεισφορών του μοντέλου (Χωρίς Timeout)



Σχήμα 6.2.2: Διάγραμμα Χρόνου ως προς το Ποσοστό των Συνεισφορών του μοντέλου (Timeout 10sec)



Σχήμα 6.2.3: Διάγραμμα Αποδειχθέντων Στοχων ως προς το Ποσοστό των Συνεισφορών του μοντέλου (Χωρίς Timeout)



Σχήμα 6.2.4: Διάγραμμα Χρόνου ως προς το Ποσοστό των Συνεισφορών του μοντέλου (Timeout 10sec)

Κεφάλαιο 7: Επίλογος

Στο κεφάλαιο αυτό γίνεται μία σύνοψη του περιβάλλοντος-πλαισίου που παρουσιάστηκε στον παρόντα τόμο και συζητούνται τα συμπεράσματα που εξήχθησαν ως προς την κάλυψη των τεθέντων στόχων. Το κεφάλαιο κλείνει με μια αναφορά σε δυνατές μελλοντικές επεκτάσεις του πλαισίου.

7.1. Σύνοψη και συμπεράσματα

Στην παρούσα εργασία εξετάσαμε το πρόβλημα της προσαρμοστικής παρακολούθησης, καταγραφής και ανάλυσης της λειτουργίας συστημάτων λογισμικού και προτείναμε μία λύση για αυτό. Μια καλή λύση στο πρόβλημα αυτό θα επιτρέψει την πιο αποδοτική παρακολούθηση συστημάτων λογισμικού, με αποτελεσμα την μείωση υπολογιστικού και χρηματικού κόστους παρακολούθησης και διαχειρησης των συστημάτων λογισμικού καθώς επίσης και την ταχύτερη ανακάλυψη σφαλμάτων και ανάλυση των αιτιών τους.

Σκοπός της διπλωματικής εργασιάς ήταν ο σχεδιασμός και η υλοποίηση ενός περιβάλλοντος-πλαισίου που θα επιτρέψει την προσαρμοστική παρακολούθηση συστημάτων λογισμικού. Μέσω του περιβάλλοντος-πλαισίου δίνεται δυνατότητα στους χρήστες του, να ορίσουν υποθέσεις για το υπο παρακολούθηση σύστημα ως ΚΑΙ/Ή δεντρα στόχων, και να όρισουν τον τρόπο επαλήθευση τους μέσω επισημειώσεων στους κόμβους των δέντρων. Το επίπεδο παρακολούθησης είναι τέτοιο ώστε να είναι ικανή η ενεργοποίηση υποθέσεων αλλά δεν αρκεί πάντα για την επαλήθευση τους και την ανάλυση των αιτίων ενεργοποίησης τους. Η αλλαγή του επιπέδου παρακολούθησης προκύπτει ως αποτέλεσμα της προσπάθειας επαλήθευσης των υποθεσεων αυτών.

Στα παραπάνω κεφάλαια παρουσιάστηκε η αρχιτεκτονική και η λειτουργία του περιβάλλοντος-πλαισίου που υλοποιήσαμε. Τα σημεία στα οποία εστιάσαμε κατά την εκπόνηση αυτής της εργασίας ήταν τα εξής:

- Αρχιτεκτονική του περιβάλλοντος-πλαισίου: Η ανάπτυξη του πλαισίου βασίστηκε στα αρχιτεκτονικά μορφήματα Μαυροπίνακα και Publisher-Subscriber. Η επιλογή της αρχιτεκτονικής εγινε με στόχο την ευελιξία και την επεκτασιμότητα του συστήματος.

- Μοντελοποίηση υποθέσεων: Η μοντελοποίηση των υποθέσων για το υπό παρακολούθηση σύστημα ως δέντρα στόχων ΚΑΙ/Η, παρέχει έναν εύκολο τρόπο ορισμού υποθέσεων αναλύοντας τες σε υποστόχους, ενώ με την χρήση συνεισφορών μπορουν να οριστούν συσχετίσεις μεταξύ των υποστόχων των υποθέσεων. Το όφελος αυτής της δομής είναι η ευκολία αναπαράστασης του κάθε στόχου σε προτασιακή λογική, που μπορεί να μετατραπεί σε συζευκτική κανονική μορφή (CNF) με έναν απλό τρόπο. Το προκύπτον σύνολο εκφράσεων CNF μπορεί να χρησιμοποιηθεί ως είσοδος

σε περιβάλλοντα συλλογιστικής όπως το εργαλείο Alchemy ή η βιβλιοθήκη SAT4j που περιγράψαμε. Οι επεκτάσεις που έγιναν προκειμένου να ενσωματώσει την έννοια των εισφορών μεταξύ των στόχους και επιμέρους στόχους, έπαιξε επίσης έναν πολύ σημαντικό ρόλο, δεδομένου ότι αντανακλά την πολυπλοκότητα των εξαρτήσεων που υπάρχουν μεταξύ των διαφόρων στόχων, οι οποίες δεν μπορούν πάντα να αναπαρασταθούν από τις απλές αποσυνθέσεις ΚΑΙ και Ή. Τέλος με την χρήση επισημειώσεων μπορουν να οριστούν τα απαραίτητα δεδομένα για την επαλήθευση μιας υπόθεσης. Οι επισημειώσεις αυτές αποτελούν την βάση για την αλλαγή του επιπέδου παρακολούθησης.

- Συλλογιστική και ανάλυση βασισμένη στα δέντρα στόχων: Σκοπός του περιβάλλοντος-πλαισίου που αναπτύξαμε ήταν η ανάλυση βασικού αιτίου. Για την ανάλυση αυτή προτάθηκε η χρήση του SAT4j. Ο SAT4j παρέχει μια καλή λύση στο πρόβλημα της ανακάλυψης των αιτιών ενεργοποίησης μιας υπόθεσης καθώς δεδομένης μιας ενεργής υπόθεσης μπορεί να βρεί όλα τα πιθανά αίτια ενεργοποίησης της. Η ιδέα της αναγωγής του προβλήματος ανάλυσης βασικού αιτίου στο πρόβλημα της ικανοποιησιμότητας παρουσιάστηκε στο [87].

- Αξιολόγηση του περιβάλλοντος-πλαισίου: Τέλος διεξάγαμε μια αξιολόγηση του προτεινόμενου περιβάλλοντος-πλαισίου ελέγχοντας επίδραση των συσνεισφροών καθώς επίσης και του μεγέθους των μόντελων δέντρων στόχων στην επίδοση του συστήματος. Η αξιολόγηση αυτή έγινε σε τυχαία μοντέλα που παρήχθησαν αυτόματα. Η ανάλυση των δεδομένων που προεκυψε απο την παραπανω διαδικασία έδειξε ότι το προτεινόμενο περιβάλλον αποδίδει καλά παρέχοντας μια καλή λύση για μοντέλα μεγέθους μέχρι και 10000 κόμβων.

7.2. Μελλοντικές Επεκτάσεις

Όπως αναφέρθηκε και παραπάνω το περιβάλλον-πλαίσιο που αναπτύξαμε σχεδιάστηκε έτσι ώστε να είναι ευέλικτο και επεκτάσιμο. Ως εκ τούτου προσφέρεται για μελλοντικές αλλαγές, επεκτάσεις και βελτιστοποιήσεις.

Όσον αφορά τον ορισμό των Alarm μία βελτιστοποίηση που θα μπορούσε να επιχειρηθεί είναι η εφαρμογή αλγορίθμων μηχανικής μάθησης έτσι ώστε να μπορεί το περιβάλλον πλαίσιο να ανακαλύψει μόνο του μορφήματα γεγονότων που οδηγούν στην ενεργοποίηση των Alarms. Το περιβάλλον-πλαίσιο απαιτεί από τους χρήστες του να δηλώσουν ρητά τα μορφήματα αυτά. Η διαδικασία αυτή είναι επιρρεπής σε λάθη και απαιτεί εξειδικευμένο προσωπικό. Μία λύση θα ήταν το περιβάλλον-πλαίσιο να μπορεί να ανακαλύψει μόνο του μορφήματα που ενεργοποιούν τα Alarms και να εμπλουτίσει τα μορφήματα που ορίζουν οι χρήστες.

Επίσης ως μελλοντική επέκταση προτείνουμε την ενεργοποίηση υποθέσεων με κάποια πιθανότητα. Πιο συγκεκριμένα προτεινουμε να οριστεί κάποια πιθανότητα για ζευγάρια υποθέσεων και μορφημάτων γεγονότων έτσι ώστε κάθε φορά που παρατηρείται ένα μόρφημα στα γεγονότα παρακολούθησης να αξιολογείται η πιθανότητα να ισχύει κάποια υπόθεση. Αυτή η γνώση θα μπορεί να χρησιμοποιηθεί για την ταξινόμηση των υποθέσεων προς επαλήθευση. Ετσι θα μπορεί να δωθεί προτεραιότητα στην επαλήθευση των πιο πιθανών υποθέσεων πρώτα.

Στην πρότυπη υλοποίηση η στρατηγική επιλογής κάποιας λύσης από αυτές που επιστρέφει ο SAT4j δεν ήταν τίποτε άλλο από την επιλογή της λύσης με τους λιγότερους υποστόχους προς απόδειξη. Το πλήθος των στόχων προς απόδειξη δεν αποτελεί καλή εκτίμηση του υπολογιστικού κόστους των λύσεων που επιστρέφονται από τον SAT4j. Μια μελλοντική

επέκταση θα ήταν η υλοποίηση μεθόδων και αλγορίθμων για την εκτίμηση του κόστους απόδειξης κάθε στόχου.

Όσον αφορά την επαλήθευση μιας λίστας υποθέσεων μια προφανής βελτιστοποίηση που θα θέλαμε να υλοποιήσουμε είναι η παραλληλοποίηση της επαλήθευσης τους. Κάτι τέτοιο απαιτεί ιδιαίτερη προσοχή καθώς τίθενται θέματα συγχρονισμού. Ωστόσο η παραλληλοποίηση της επαλήθευσης πολλών υποθέσεων αναμένεται να βελτιώσει σημαντικά την επίδοση του περιβάλλοντος-πλαισίου.

Τέλος θα ήταν πολύ χρήσιμη η ολοκλήρωση του περιβάλλοντος-πλαισίου με εργαλεία συλλογής δεδομένων έτσι ώστε να μπορεί να χρησιμοποιηθεί για την παρακολούθηση πραγματικών συστημάτων. Κάτι τέτοιο θα επιτρέψει πιο λεπτομερή αξιολογηση του αλλά και σύγκριση του με άλλα περιβάλλοντα παρακολούθησης.

Κεφάλαιο 8: Βιβλιογραφία

[1]Soila Pertet and Priya Narasimhan. Causes of failure in web applications. Technical Report CMU-PDL-05-109, Carnegie Mellon University Parallel Data Lab, December 2005. 1

[2] O.M.G., "Meta Object Facility (MOF) Core Specification," *Management*, 2006.

[3] A.C. Completion and C. Outline, "MOF Essentials," *Library*

[4] E. Foundation. (2009, Oct.) Eclipse Modeling Framework. [Online]. http://www.eclipse.org/modeling/emf/

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

[6]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

[7] Dardenne, A., van Lamsweerde A. and Fickas, S., "Goal Directed Requirements Acquisition", *Science of Computer Programming*, Vol. 20, No. 1-2, pp. 3–50, 1993

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

[9] 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.

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

[11] Rolland, C., Souveyet, C., and Ben Achour, C. "Guiding goal modeling using scenarios," *IEEE Trans. Software Eng*., vol. 24, pp. 1055–1071, Dec. 1998.

[12] 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

[13] 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.

[14] 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

[15] Amit K. Chopra, Fabiano Dalpiaz, Paolo Giorgini, and John Mylopoulos. Reasoning about agents and protocols via goals and commitments. In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1, AAMAS '10, pages 457--464, 2010.

[16] 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.

[17] 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.

[18]http://en.wikipedia.org/wiki/First-order_logic

[19] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107--136, 2006.

[20] D. Pedro and D. Lowd, *Markov Logic: An Interface Layer for Artificial Intelligence*, Morgan & Claypool, 2009.

[21] Tivadar Papai, Parag Singla, and Henry Kautz. Constraint propagation for efficient inference in markov logic. In *Proceedings of 17th International Conference on Principles and Practice of Constraint Programming (CP 2011)*, number 6876 in Lecture Notes in Computer Science (LNCS), pages 691—705, 2011.

[22] Pedro Domingos Marc Sumner. The alchemy tutorial [online]. http://alchemy.cs.washington.edu/tutorial/tutorial.pdf.

[23] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. Journal of ACM, 5:394–397, 1962

[24] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: engineering an efficient sat solver. In Design automation, pages 530–535. ACM Press New York, NY, USA, 2001.

[25] E. Goldberg and Y. Novikov. Berkmin: A fast and robust sat-solver. In DATE,

pages 142–149, 2002.

[26] L. Ryan. Efficient algorithms for clause-learning SAT solvers. Master's thesis, Simon Fraser University, 2004.

[27] J. Gu. Global optimization for satisfiability (SAT) problem. *IEEE Trans, on Knowledge and Data Engineering,* 6(3):361-381, Jun. 1994, and 7(1):192, Feb. 1995.

[28] *J. Gu. The UniSAT problem models (appendix). IEEE Trans, on Pattern Analysis and Machine Intelligence,* 14(8):865, Aug. 1992.

[29] V. Manquinho and J. Marques-Silva, "Search pruning conditions for Boolean optimization," in *European Conference on Artificial Intelligence*, August 2000, pp. 130–107

[30] ——, "Translating pseudo-Boolean constraints into SAT," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 2, March 2006. [31] F. Aloul, A. Ramani, I. Markov, and K. Sakallah. PBS: A Backtrack Search Pseudo-Boolean Solver. In Symposium on the Theory and Applications of Satisability Testing (SAT'2002), 2002.

[32] Hossein M. Sheini and Karem A. Sakallah. Pueblo: A Hybrid Pseudo-Boolean SAT Solver. Journal on Satisability, Boolean Modeling and Computation (JSAT), 2:163{ 187, 2006.

[33] F. Heras, J. Larrosa, and A. Oliveras, "MiniMaxSat: a new weighted Max-SAT solver," in *International Conference on Theory and Applications of Satisfiability Testing*, May 2007, pp. 41–55.

[34] C. M. Li, F. Many`a, and J. Planes, "New inference rules for Max-SAT," *Journal of Artificial Intelligence Research*, vol. 30, pp. 321–359, 2007.

[35] Z. Fu and S. Malik, "On solving the partial MAX-SAT problem," in *International Conference on Theory and Applications of Satisfiability Testing*, August 2006, pp. 252–65.

[36] J. Marques-Silva and J. Planes, "Algorithms for maximum satisfiability using unsatisfiable cores," in *Design, Automation and Testing in Europe Conference*, March 2008.

[37] D. Le Berre. A satisfiability library for java. http://www.sat4j.org/.

[38] R. Weinreich, W. Kurschl, *Dynamic Analysis of Distributed Object-Oriented Applications,* Proc. Hawaii International Conference On System Sciences, Kona, Hawaii, January 6-9, 1997

[39] B. Sridharan, S. Mundkur, A.P. Mathur, *Non-intrusive Testing, Monitoring and Control of Distributed CORBA Objects,* TOOLS Europe 2000, St. Malo, France, June 2000

[40] B. Sridharan, B. Dasarathy and A. P. Mathur, *On Building Non-intrusive Performance Instrumentation Blocks for CORBA-based Distributed Systems,* 4th IEEE International Computer Performance and Dependability Symposium, Chicago March 2000.

[41] F. Lange, R. Kroeger, M. Gergeleit, *JEWEL: Design and Measurement of a Distributed Measurement System,* IEEE Transactions on Parallel and Distributed Systems, November 1992

[42] M. Richmond, *Flexible Migration Support for Component Frameworks,* Doctoral Dissertation, Department of Computing, Macquarie University, January, 2003

[43] M. Richmond, J. Noble, *Reflections on Remote Reflection,* Australasian Computer Science Conference (ACSC) 2001, Brisbane, Jan 2001

[44] P. H. Deussen, G. Din, I. Schieferdecker, An On-line Test Platform for Component-based Systems, Proceedings of 27th IEEE / NASA Goddard Software Engineering Workshop (SEW-27'02)

[45] V. Talwar, C. Shankar, S. Rafaeli, D. Milojicic, S. Iyer, K. Farkas, and Y. Chen, "Adaptive monitoring automated change management for monitoring systems," in Proceedings of the 13th Workshop of the HP OpenView University Association, 2006, pp. 21–24.

[46] M. A. Munawar and P. A. Ward, "Adaptive monitoring in enterprise software systems," SysML, June 2006.

[47] Andres J. Ramirez, Betty H.C. Cheng, and Philip K. McKinley, "Adaptive Monitoring of Software Requirements", in Proceedings of the First International Workshop on Requirements at Run Time, Sydney, Australia, October 2010.

[48] A. Mos and J. Murphy, "Compas: Adaptive performance monitoring of component-based systems," in In the Proceedings of the Second International Workshop on Remote Analysis and Measurement of Software Systems (RAMSS 04). Edinburgh, Scotland, UK: ACM, May 2004, pp. 35–40.

[49] B. Tierney et ai., *A Monitoring Sensor Management System for Grid Environments,* In Proceedings of Ninth IEEE International Symposium on High Performance Distributed Computing (HPDC'OO), August 01 - 04, 2000, Pittsburgh, Pennsylvania

[50] Y. Wang, S. A. McIlraith, Y. Yu, and J. Mylopoulos, "An automated approach to monitoring and diagnosing requirements," in ASE '07: Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering. Atlanta, Georgia, USA: ACM, November 2007, pp. 293–302.

[51] Ehlers, J., van Hoorn, A., Waller, J., Hasselbring, W.: Self-Adaptive Software System Monitoring for Performance Anomaly Localization. In Proceedings of the 8th IEEE/ACM International Conference on Autonomic Computing (ICAC 2011). ACM, Karlsruhe, Germany. 197-200. (2011)

[52]. Hoorn, A. v., Hasselbring, W., Waller, J.: Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis. Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE 2012). ACM, Boston, Massachusetts, USA. To appear. (2012)

[53] Object Constraint Language (OCL) 2.0. OMG. [Online] Available: http://www.omg.org/spec/MOF/2.0 (September 2011)

[54] K. Kent and M. Souppaya, "Guide to computer security log management," Tech. Rep. 800-92, Nataional Institute of Standards and Technology, September 2006.

[55] David Ogle et al. Canonical situation data format: The common base event v1.0.1, November 2003. Copyright IBM, http://www.eclipse.org/tptp/platform/documents/resources/cbe101spec/CommonBaseEvent_SituationData_V1.0.1.pdf

[56] B. Jacob, R. Lanyon-Hogg, D. K. Nadgir, and A. F. Yassin. A Practical Guide to the IBM Autonomic Computing Toolkit. IBM Redbooks, 2004

[57] Apache HTTP Server Log Formats http://httpd.apache.org/docs/2.0/logs.html

[58] Generic Log Adapter http://help.eclipse.org/help31/index.jsp?topic=/org.eclipse.tptp.monitoring.doc.user/concepts/cintro.html

[59] "Dtrace tool." http://docs.oracle.com/cd/E19253-01/819-5488/

[60] Microsoft Corporation, "Microsoft operations manager."http://msdn.microsoft.com/en-us/library/aa505337.aspx

[61] Eclipse Foundation Inc., "Eclipse birt." http://www.eclipse.org/birt/phoenix/.

[62] Eclipse Hyades Project, "Eclipse test and performance tools platform." http://www.eclipse.org/tptp/index.html.

[63] IBM Rational Software Architect, "Determining problems in distributed applications."http://publib.boulder.ibm.com/infocenter/rtnlhelp/v6r0m0/index.jsp?topic=/org.eclipse.hyades.log.ui.doc.user/concepts/ceautcom.htm.

[64] J. H. Andrews and Y. Zhang, "Broad-spectrum studies of log file analysis," in *Proceedings of the 2000 International Conference on Software Engineering*, pp. 105–114, ICSE, 4–11 June 2000.

[65] R. Vaarandi, "A data clustering algorithm for mining patterns from event logs," in *Proceedings of the Third IEEE Workshop on IP Operations and Management*, pp. 119–126, IPOM, 1–3 October 2003.

[66] J. Stearley, "Towards informatic analysis of syslogs," in *Proceedings of the 2004 IEEE International Conference on Cluster Computing*, pp. 309–318, 20–23 September 2004.

[67] Y.-C. Lin and P. Hadingham, “Tracking frequent traversal areas in a web site via log analysis,” in *Seventh International Conference on Parallel and Distributed Systems: Workshops*, pp. 321–325, PADSW, 4–7 July 2000.

[68] T. Koch, A. Ard¨o, and K. Golub, “Browsing and searching behaviour in the renardus web service: A study based on log analysis,” in Proceedings of the 2004 Joint ACM/IEEE Conference on Digital Libraries, p. 378, 7–11 June 2004.

[69] A. Maeda, K. Sugiyama, and K. Mase, “Log analyses of human behaviour on interactive amusement media,” in Proceedings of the Fourth International Conference on Knowledge- ased Intelligent Engineering Systems and Allied Technologies, vol. 1, pp. 229–232, KES, 30 Aug – 1 Sept 2000.

[70] L. Rostad and O. Edsberg, “A study of access control requirements for healthcare systems based on audit trails from access logs,” in Proceedings of the 22nd Annual Computer Security Applications Conference, pp. 175–186, ACSAC, December 2006.

[71] P. Horn, “Autonomic Computing: IBM's Perspective on the State of Information Technology,” IBM Corp. (October 2001), http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf

[72]IBM. An Architectural Blueprint for Autonomic Computing. 2006.

[73] J.Kephart, D.Chess, The vision of autonomic computing, IEEE Computer 36 (1) (2003) 41–50.

[74]A.G. Ganek and T. A. Corbi, The dawning of the autonomic computing era, IBM Systems Journal, Vol 42, No 1, 2003, pp. 5-18

[75] LALANDA, P., MCCANN, J., AND DIACONESCU, A. Autonomic Computing: Principles, Design and Implementation. Undergraduate Topics in Computer Science Series. Springer London, Limited, 2013.

[76] Huebscher, M. C. and McCann, J. A. 2008. A survey of autonomic computing degrees, models, and applications. *ACM Comput. Surv. 40,* 3, 1-28.

[77] Sterritt, R., Smyth, B., and Bradley, M. 2005. PACT: Personal autonomic computing tools. In Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS). IEEE Computer Society, Washington, DC, USA, 519– 527.

[78] Bigus, J. P., Schlosnagle, D. A., Pilgrim, J. R., III, W. N. M., and Diao, Y. 2002. ABLE: A toolkit for building multiagent autonomic systems. IBM Systems Journal 41, 3, 350–371. Bougaev, A. A. 2005. Pattern recognition based tools enabling autonomic computing. In Proceedings of 2nd IEEE International Conference on Autonomic Computing (ICAC). 313– 314.

[79] Garlan, D. and Schmerl, B. 2002a. Exploiting architectural design knowledge to support selfrepairing systems. In Proceedings of the 14th international conference on

Software engineering and knowledge engineering.

[80]Kephart, J. O. and Walsh, W. E. 2004. An artificial intelligence perspective on autonomic computing policies. In Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks. 3–12.

[81]Jennings, N. R. 2000. On agent-based software engineering. Artificial Intelligence 117, 2 (Mar.), 277–296.

[82]Gleizes, M.-P., Link-Pezet, J., and Glize, P. 2000. An adaptive multi-agent tool for electronic commerce. In Proeedings of the IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE).

[83]Kumar, S. and Cohen, P. R. 2000. Towards a fault-tolerant multi-agent system architecture. In Proceedings of the fourth international conference on Autonomous agents.

[84]Wise, A., Cass, A. G., Lerner, B. S., Call, E. K. M., Osterweil, L. J., and Jr., S. M. S. 2000. Using Little-JIL to coordinate agents in software engineering. In Automated Software Engineering Conference (ASE 2000).

[85]Melcher, B. and Mitchell, B. 2004. Towards an autonomic framework: Self-configuring network services and developing autonomic applications. Intel Technology Journal 8, 4 (Nov.), 279–290.

[86] F. Buschman, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal; *Pattern-oriented Software Architecture – A System of Patterns*, Wiley, Chichester, 1996.

[87] Y. Wang, Monitoring and Diagnosis for Autonomic Systems: A Requirement Engineering Approach, PhD thesis, Univeristy of Toronto, 2010.