ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Περιβάλλον-Πλαίσιο Ανάλυσης Δομικής και Λειτουργικής Διαμόρφωσης Υπηρεσιοκεντρικών Συστημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΝΑΣΤΑΣΙΟΣ – ΠΑΝΑΓΙΩΤΗΣ Δ. ΛΙΒΟΓΙΑΝΝΗΣ

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

Αθήνα, Οκτώβριος 2009

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

Περιβάλλον-Πλαίσιο Ανάλυσης Δομικής και Λειτουργικής Διαμόρφωσης Υπηρεσιοκεντρικών Συστημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΝΑΣΤΑΣΙΟΣ – ΠΑΝΑΓΙΩΤΗΣ Δ. ΛΙΒΟΓΙΑΝΝΗΣ

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

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 26η Οκτωβρίου 2009.

.....	.....	.....
Κωνσταντίνος Κοντογιάννης Αναπλ. Καθηγητής Ε.Μ.Π.	Ιωάννης Βασιλείου Καθηγητής Ε.Μ.Π.	Τιμολέων Σελλής Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2009

.....

Αναστάσιος – Παναγιώτης Δ. Λιβογιάννης

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

Περίληψη

Σκοπός της παρούσας διπλωματικής εργασίας είναι ο σχεδιασμός και η πρωτότυπη υλοποίηση ενός περιβάλλοντος-πλαισίου που θα πραγματοποιεί πριν την χρήση έλεγχο σε υπηρεσιοκεντρικά συστήματα λογισμικού. Τα υπηρεσιοκεντρικά συστήματα λογισμικού, αποκτούν όλο και μεγαλύτερο ενδιαφέρον τόσο στην επιστημονική όσο και στην επιχειρηματική κοινότητα. Για την επιχειρηματική κοινότητα, τα υπηρεσιοκεντρικά συστήματα προσφέρουν νέες τεχνικές, αντιλήψεις και μεθόδους για τα στάδια της ανάπτυξη και ολοκλήρωσης σύγχρονων δικτυακών, WEB 2.0 τύπου υπηρεσιών. Για την επιστημονική κοινότητα, τα υπηρεσιοκεντρικά συστήματα λογισμικού, αποτελούν ένα απαιτητικό πεδίο έρευνας, το οποίο στηρίζεται σε βασικές αρχές της Τεχνολογίας Λογισμικού, αλλά επιζητά την επίλυση νέων, ενδιαφερόντων προβλημάτων που θα μπορούσαν να βοηθήσουν συνολικά στις μεθόδους σχεδιασμού και ανάπτυξης λογισμικού. Τα υπηρεσιοκεντρικά συστήματα λογισμικού, αποκτούν ολοένα και περισσότερες εφαρμογές, ιδιαίτερα στις ιστοσελίδες – δικτυακές υπηρεσίες μεγάλων οργανισμών και επιχειρήσεων, καθώς και στην αντικατάσταση παλαιότερων συστημάτων λογισμικού, τύπου WEB 2.0, που είναι εξαρτώμενες από το περιβάλλον. Η ραγδαία ανάπτυξη αυτού του είδους των συστημάτων λογισμικού, καθώς και η απαίτηση για συνεχή λειτουργία αυτών, συνεπάγονται δυσκολία στην συντήρησή και έλεγχό τους. Η παρούσα διπλωματική εργασία, προτείνει την αρχιτεκτονική ενός περιβάλλοντος-πλαισίου, το οποίο θα μπορεί πριν την χρήση υπηρεσιοκεντρικών συστημάτων λογισμικού, να ελέγχει από τα αρχεία διαμόρφωσής τους, αν μπορούν να επιτευχθούν κάποιοι στόχοι που έχουν τεθεί για την λειτουργία τους. Οι στόχοι αυτοί, μοντελοποιούνται μέσω ενός μοντέλου δέντρων στόχων, που επίσης αποτελεί αντικείμενο της παρούσας διπλωματικής. Στο τέλος, το προσανατολισμένο σε στόχους περιβάλλον-πλαίσιο που σχεδιάζεται στην παρούσα διπλωματική, ελέγχεται για την συνεισφορά του στην συντήρηση υπηρεσιοκεντρικών συστημάτων λογισμικού, καθώς και δίνονται συμβουλές για την επέκταση και χρήση του από κάθε ενδιαφερόμενο.

Λέξεις Κλειδιά: <<υπηρεσιοκεντρικά συστήματα λογισμικού, δέντρο στόχων, αυτόνομα συστήματα λογισμικού, πριν την χρήση έλεγχος, συντήρηση συστημάτων λογισμικού, τεχνολογία λογισμικού>>

Abstract

The main goal of this diploma thesis is the design and implementation of a goal driven framework for preflight checks that are required when updating components in Service Oriented Architecture (SOA) software systems. Over the past few years, Service Oriented Architecture has received significant attention both from the business and the academic community, as it provides the necessary means and protocols for the flexible and customizable integration of distributed software components to form complex service providing applications. Form one hand, the business community, relies on SOA systems to provide new techniques and concepts for developing and integrating contemporary, WEB 2.0 services while on the other hand, the academic community considers SOA systems an interesting research field both from the programming model perspective and from the software architecture perspective. As these SOA systems become more complex, the need to develop elaborate frameworks to support their maintenance also increases. In this diploma thesis a reference architecture and a supporting software framework to perform preflight checks during the maintenance of SOA software systems, are proposed. The proposed architecture is based on the blackboard architecture style and allows for the flexible management of preflight policies to be evaluated. Similarly, the proposed framework is based on goal models that are automatically built by analyzing the SCA specification of the system being checked. Goal models are then verified by specialized components that are triggered according to the specific goals set.

Keywords: <<service oriented software systems, goal tree, autonomic software systems, preflight check of SOA Systems, software maintenance, software engineering>>

Ευχαριστίες

Θέλω να ευχαριστήσω θερμά τον επιβλέποντα της παρούσας διπλωματικής εργασίας, Αναπληρωτή Καθηγητή του Ε.Μ.Π., κύριο Κώστα Κοντογιάννη για την στήριξη, την υπομονή του και την βοήθειά του στην εκπόνησή της. Υπήρξε ο άνθρωπος ο οποίος με ενέπνευσε να ασχοληθώ με ερευνητικά θέματα της Τεχνολογίας Λογισμικού και ο οποίος ήταν δίπλα σε κάθε βήμα μου κατά την εξερεύνηση αυτού του δύσκολου και άγνωστου κόσμου για μένα. Οι συμβουλές του, είναι αυτές που άλλαξαν τον τρόπο σκέψης μου, μυώντας με στον κόσμο της έρευνας στον τομέα της Τεχνολογίας Λογισμικού.

Μεγάλη συμβολή στην πενταετή πορεία μου στην σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ είχαν και οι συμφοιτητές μου. Θέλω εκ βάθους καρδίας να ευχαριστήσω όλους τους συναδέλφους μου με τους οποίους δουλέψαμε, συνεργαστήκαμε, αντιδικήσαμε μαζί στην διάρκεια των σπουδών μας στην σχολή, καθώς αυτοί είναι που συνέβαλαν στην διαμόρφωση του χαρακτήρα μου ως επιστήμονα. Ιδιαίτερη μνεία αξίζει στους φίλους μου, εντός και εκτός σχολής όλο αυτό το διάστημα, οι οποίοι συνέβαλαν στην ολοκλήρωσή μου ως άνθρωπο.

Τέλος, αν και είναι δύσκολο να αποτυπωθούν με λέξεις, θέλω να δώσω τις πιο θερμές ευχαριστίες μου στην οικογένειά μου, που όλα αυτά τα χρόνια με στήριξε, με καθοδήγησε να θέτω και να πετυχαίνω τους στόχους μου και με μεγάλωσε με κλίμα διεύρυνσης πνεύματος και αποφυγής προκαταλήψεων. Οι πιο θερμές μου ευχαριστίες, πηγαίνουν στην αδερφή μου Κωνσταντίνα και τους γονείς μου, Δημήτρη και Βάσω, για την αγάπη, εμπιστοσύνη, ειλικρίνεια και ομοψυχία που μου έδειξαν όλα αυτά τα χρόνια.

Τάσος

Πίνακας Περιεχομένων

Πίνακας Σχημάτων

Κεφάλαιο 1: Εισαγωγή

1.1 Λογισμικό Προσανατολισμένο σε Υπηρεσίες

Στην σύγχρονη εποχή, η επιστήμη της Τεχνολογίας Λογισμικού έχει επωμιστεί τον ρόλο της σχεδίασης, επαλήθευσης και υλοποίησης νέων αρχιτεκτονικών τεχνοτροπιών λογισμικού αλλά και μεθόδων γραφής κώδικα, με στόχο το λογισμικό να παράγεται γρηγορότερα, ευκολότερα, να είναι αξιόπιστο και επαναχρησιμοποιήσιμο.

Από τις πρώτες, χρονικά, προσπάθειες προς αυτήν την κατεύθυνση ήταν η αντικειμενοστραφής σχεδίαση λογισμικού και το λεγόμενο αντικειμενοστραφές μοντέλο προγραμματισμού. Το μοντέλο αυτό από το 1990 ως και σήμερα, είναι αυτό που χρησιμοποιείται κατά κόρον στα συστήματα λογισμικού και μεγάλος αριθμός γλωσσών προγραμματισμού έχει δημιουργηθεί/τροποποιηθεί ώστε να διευκολύνει τον προγραμματισμό με βάση αυτό το μοντέλο. Χαρακτηριστικό παράδειγμα γλώσσας προγραμματισμού που δημιουργήθηκε με σκοπό να βοηθήσει στον αντικειμενοστραφή προγραμματισμό ήταν η Java. Αντίστοιχα, η Objective-C αποτελεί χαρακτηριστικό παράδειγμα διαλέκτου μιας ευρέως διαδεδομένης γλώσσας προγραμματισμού, που δημιουργήθηκε με σκοπό να υποστηρίξει τον αντικειμενοστραφή προγραμματισμό.

Σχεδόν ταυτόχρονα με την πληροφορική και την ύπαρξη υπολογιστών, έχουμε την εμφάνιση των δικτύων υπολογιστών και την δυνατότητα δυο ξεχωριστοί ηλεκτρονικοί υπολογιστές να επικοινωνούν μέσω δικτύου και να ανταλλάσσουν δεδομένα κατά την εκτέλεση των προγραμμάτων τους. Από την αρχή της εμφάνισης δικτύων υπολογιστών ως και τα πρώτα χρόνια ύπαρξης του διαδικτύου, το βασικό μοντέλο προγραμματισμού για εφαρμογές που περιλάμβαναν ανταλλαγή δεδομένων μεταξύ υπολογιστών στο δίκτυο, ήταν το μοντέλο πελάτη-εξυπηρετητή (client-server model). Με την περαιτέρω ανάπτυξη του διαδικτύου και με την ολοένα και αυξανόμενη είσοδό του στην καθημερινή ζωή των ανθρώπων, έπρεπε να παρουσιαστεί ένα διαφορετικό μοντέλο, το οποίο θα επέτρεπε χαλαρότερη σύνδεση μεταξύ των υπολογιστών που έπαιζαν το ρόλο είτε του πελάτη είτε του εξυπηρετητή. Ένα μοντέλο που θα συνδύαζε τα καλά χαρακτηριστικά του αντικειμενοστραφούς μοντέλου με έναν διαδικτυακό προσανατολισμό.

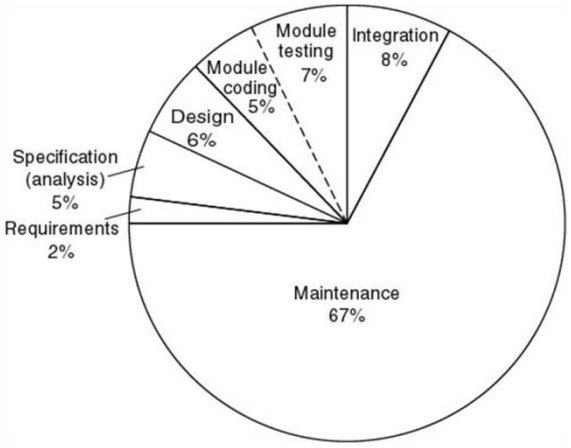
Το μοντέλο αυτό βρέθηκε στο μοντέλο της αρχιτεκτονικής λογισμικού προσανατολισμένης σε υπηρεσίες. Τα συστήματα λογισμικού που ακολουθούν την τεχνοτροπία της αρχιτεκτονικής προσανατολισμένης σε υπηρεσίες, από εδώ και στο εξής υπηρεσιοκεντρικά συστήματα λογισμικού, δομούν την λειτουργικότητά τους σε υπηρεσίες. Τα δομοστοιχεία του συστήματος λογισμικού χαρακτηρίζονται ως παραγωγοί υπηρεσιών (προσφέρουν υπηρεσίες στο περιβάλλον) και ως καταναλωτές υπηρεσιών (καταναλώνουν υπηρεσίες που προσφέρουν άλλα δομοστοιχεία). Ένα δομοστοιχείο μπορεί να είναι ταυτόχρονα και παραγωγός και καταναλωτής, διαφορετικών υπηρεσιών. Τα υπηρεσιοκεντρικό πρότυπο συστημάτων λογισμικού παρέχει χαμηλή σύζευξη και διευκολύνει την ανεξάρτητη υλοποίηση λειτουργικότητας σε διαφορετικές υπηρεσίες.

1.2 Αυτόνομα Συστήματα Λογισμικού

Στην σύγχρονη εποχή, οι απαιτήσεις των χρηστών λογισμικού έχουν αυξηθεί κατακόρυφα και η Τεχνολογία Λογισμικού έχει αναπτύξει εργαλεία και τεχνικές ώστε να ικανοποιούνται αυτές οι απαιτήσεις, με παράλληλη αύξηση, όμως, της πολυπλοκότητας του σύγχρονου λογισμικού. Αυτή η αύξηση, εντούτοις, δεν μπορεί να είναι δικαιολογία για τους αρχιτέκτονες και μηχανικούς λογισμικού, οι οποίοι πρέπει να διασφαλίσουν την αξιοπιστία του λογισμικού και την ευκολία συντήρησής του.

Η συντήρηση, είναι το στάδιο του κύκλου ζωής ενός συστήματος λογισμικού, το οποίο αποτελεί την μεγαλύτερη πρόκληση για την Τεχνολογία Λογισμικού, καθώς είναι το μεγαλύτερο σε διάρκεια στον κύκλο ζωής του συστήματος, συνεπώς και αυτό με το περισσότερο κόστος από όλα. Επίσης, η δυσκολία συντήρησης του λογισμικού είναι συνάρτηση της πολυπλοκότητάς του και παρά τις προσπάθειες της Τεχνολογίας Λογισμικού να προτείνει αρχιτεκτονικές τεχνοτροπίες εύκολες στην συντήρηση, η πολυπλοκότητα του σύγχρονου λογισμικού εξακολουθεί να αυξάνει, ανάλογα με την αύξηση των απαιτήσεων των χρηστών. Ενδεικτικό των παραπάνω είναι το παρακάτω σχήμα στην Εικόνα 1-Ι δείχνει πόσο μεγάλο ποσοστό του κόστους ενός συστήματος λογισμικού καταλαμβάνει το στάδιο της συντήρησης. Το σχήμα αυτό, αναφέρεται στο βιβλίο [1].

Για το λόγο αυτό, ιδιαίτερο ρόλο στην Τεχνολογία Λογισμικού παίζει η ανάπτυξη των Αυτόνομων Συστημάτων. Ως Αυτόνομα Συστήματα νοούνται εκείνα τα οποία μπορούν και παρουσιάζουν προσαρμοστικότητα στις αλλαγές κατάστασης του περιβάλλοντός τους. Συνήθως αυτό συμβαίνει μέσω ενός βρόχου ανατροφοδότησης ο οποίος φροντίζει να ενημερώνει το Αυτόνομο Σύστημα για τις αλλαγές στην κατάσταση του περιβάλλοντος. Τα Αυτόνομα Συστήματα λόγω αυτής της προσαρμοστικότητας – νοημοσύνης που παρουσιάζουν, αποτελούν ελπίδα για την ανάπτυξη μεθόδων αυτοματοποιημένης συντήρησης λογισμικού και βελτίωσή της, με παράλληλη μείωση του κόστους ανάπτυξης λογισμικού.



Εικόνα 1-Ι:Ποσοστιαία κατανομή κόστους ανά στάδιο του κύκλου ζωής λογισμικού

1.3 Κίνητρο

Όπως αναφέρθηκε προηγουμένως, το μεγαλύτερο κόστος στην ζωή ενός συστήματος λογισμικού, το κατέχει το στάδιο της συντήρησής του. Το στάδιο αυτό, περιλαμβάνει τις εξής 6 βασικές λειτουργίες, κατά το πρότυπο **ISO/IEC 14764:2006**

1. **Προετοιμασία** – Αφορά στον σχεδιασμό κώδικα ο οποίος θα λύνει προβλήματα που προέκυψαν στο σύστημα λογισμικού. Επίσης περιλαμβάνει και όλες τις κατάλληλες ενέργειες μετάβασης από το αρχικό, με σφάλματα λογισμικό, στην έκδοση στην οποία τα σφάλματα που παρουσιάστηκαν έχουν λυθεί.

2. **Ανάλυση προβλημάτων και τροποποιήσεων** – Αναφέρεται στην περίπτωση όπου εντοπισθεί κάποιο πρόβλημα ή υπάρξει ανάγκη για τροποποίηση. Περιλαμβάνει τις ενέργειες που απαιτούνται ώστε να επαληθευτεί η ύπαρξη ή μη του προβλήματος, να επαληθευτεί αν η τροποποίηση είναι εφικτή και σύμφωνη με τα υπόλοιπα λειτουργικά χαρακτηριστικά του προγράμματος και τέλος η τεκμηρίωση και ενσωμάτωση στο αρχικό σύστημα της προτεινόμενης λύσης/τροποποίησης.

3. **Υλοποίηση** – Περιλαμβάνει την πραγματική υλοποίηση μιας τροποποίησης, είτε αυτή προέρχεται από αίτηση ικανοποίησης μιας επιπλέον απαίτησης, είτε από τον εντοπισμό ενός σφάλματος στο σύστημα. Περιλαμβάνει δηλαδή, το κομμάτι της συγγραφής του κώδικα που χρειάζεται η λύση/τροποποίηση.

4. **Αποδοχή –** Εδώ, έχουμε την περίπτωση όπου η ομάδα που έχει αναλάβει την συντήρηση του λογισμικού, σε συνεργασία με την ομάδα/άτομο που ζήτησε την τροποποίηση ή εντόπισε το σφάλμα, ελέγχουν την τροποποίηση/διόρθωση ώστε να επαληθευτεί ότι ικανοποιεί την αίτηση.

5. **Μεταφορά πλατφόρμας –** Μια λειτουργία που δεν συμβαίνει συχνά στον κύκλο ζωής ενός συστήματος λογισμικού, καθώς η πλατφόρμα στην οποία θα αναπτυχθεί είναι στις περισσότερες περιπτώσεις σαφώς καθορισμένη στα αρχικά στάδια σχεδίασης του συστήματος. Εντούτοις, αν υπάρξει ανάγκη μεταφοράς του συστήματος λογισμικού σε άλλη πλατφόρμα από την αρχική, για παράδειγμα σε άλλο λειτουργικό σύστημα ή σε άλλο εξυπηρετητή, η εργασία αυτή ανατίθεται στην ομάδα συντήρησης του λειτουργικού συστήματος.

6. **Απόσυρση –** Επίσης μια λειτουργία που δεν συμβαίνει σε τακτά χρονικά διαστήματα στην διάρκεια ζωής ενός συστήματος λογισμικού, είναι η απόσυρση μέρους του είτε λόγω παλαιότητας, επειδή για παράδειγμα υποστήριζε παρωχημένες αρχιτεκτονικές που δεν χρησιμοποιούνται πλέον, είτε λόγω αντικατάστασής του.

Από τα παραπάνω, γίνεται προφανές ότι η μισή και πλέον αρμοδιότητα που επωμίζεται η ομάδα συντήρησης ενός συστήματος λογισμικού στηρίζεται στην διόρθωση λαθών. Αυτό, προϋποθέτει τον σωστό εντοπισμό λαθών τόσο σε επίπεδο λειτουργικών απαιτήσεων του συστήματος, όσο και σε επίπεδο κώδικα, του ακριβούς τμήματος δηλαδή που σχετίζεται με το λάθος και πρέπει να αλλαχτεί. Επομένως, ο εντοπισμός λαθών είναι ένα κρίσιμο ζήτημα κατά την συντήρηση του λογισμικού. Επιπλέον, τα σύγχρονα συστήματα λογισμικού, που είναι κατά κόρον διαδικτυακά και κατανεμημένα, έχουν υψηλότερες απαιτήσεις όσον αφορά την αξιοπιστία τους. Αυτό δυσχεραίνει ακόμα περισσότερο την συντήρηση του συστήματος λογισμικού, καθώς στην περίπτωση σοβαρού λάθους, πολλές φορές ένα μέρος του συστήματος πρέπει να βγει εκτός σύνδεσης, μέχρι να διορθωθεί το λάθος.

Από τα παραπάνω συμπεραίνουμε ότι ο έλεγχος της σωστής λειτουργίας ενός συστήματος λογισμικού είναι πολύ σημαντικός και μπορεί να βοηθήσει σημαντικά στην βελτίωση του κόστους του λογισμικού, στην αξιοπιστία του και την χρηστικότητά του. Παράλληλα με τα παραπάνω, η δυνατότητα ελέγχου της καλής λειτουργίας του λογισμικού, πριν αυτό χρειαστεί να επιτελέσει την λειτουργία του, θα βελτίωνε περαιτέρω την αξιοπιστία του, ενώ θα ήταν χρήσιμο και στον εντοπισμό λαθών πριν αυτά συμβούν. Ιδιαιτέρως χρήσιμο κάτι τέτοιο, θα ήταν σε συστήματα λογισμικού μεγάλης κλίμακας, που χρησιμοποιούνται από μεγάλους οργανισμούς και τα οποία εκτός από την εγγενή πολυπλοκότητα που κάνει δύσκολη την συντήρησή τους, υπάρχει και η

απαίτηση να είναι 100% διαθέσιμα από την στιγμή που εγκατασταθούν, ως την στιγμή που θα χρειαστεί να αποσυρθούν. Όλα τα παραπάνω που περιλαμβάνονται στον τομέα συντήρησης συστημάτων λογισμικού, αποτελούν το κίνητρο της παρούσας διπλωματικής εργασίας.

1.4 Περιγραφή του Προβλήματος

Με βάση τα παραπάνω, η Τεχνολογία Λογισμικού, ιδιαίτερα όσοι ασχολούνται με συστήματα μεγάλης κλίμακας που λειτουργούν σε κατανεμημένα περιβάλλοντα, καλούνται να απαντήσουν στα εξής ζητήματα:

1. Δεδομένου των λειτουργικών απαιτήσεων ενός λογισμικού, υπάρχει η δυνατότητα να γνωρίζω κατά την παράταξή του στο λειτουργικό σύστημα ή στον εξυπηρετητή αν μπορεί να λειτουργήσει σωστά, τουλάχιστο όσον αφορά τις βασικές πιο πάνω απαιτήσεις;

2. Είναι εφικτό σε ένα σύστημα που αποτελείται από διαφορετικές υπηρεσίες, που καθεμιά στηρίζεται στην άλλη, να βρεθούν ποιες είναι οι εξαρτήσεις των υπηρεσιών μεταξύ τους, αλλά και τις εξαρτήσεις των δομοστοιχείων μεταξύ τους, κάτι που θα βοηθήσει να απαντηθεί το προηγούμενο ερώτημα;

3. Δεδομένων των ρυθμίσεων που δηλώνουν τα δομικά και λειτουργικά χαρακτηριστικά κάθε δομοστοιχείου ή υπηρεσίας ενός συστήματος λογισμικού, είναι δυνατόν να βρούμε τις απλές εκείνες ενέργειες που χρειάζεται να εκτελέσει κάθε δομοστοιχείο ή υπηρεσία πριν και μετά την φόρτωσή του στο σύστημα, ώστε να αρχικοποιηθεί σωστά;

4. Δεδομένων των απαντήσεων στα παραπάνω ερωτήματα, υπάρχει η δυνατότητα να βρεθεί ένα σύνολο από στόχους που να περιγράφουν λειτουργικές απαιτήσεις του συστήματος λογισμικού που ελέγχεται, ώστε να ελέγξω αν επαληθεύονται;

5. Η επαλήθευση του παραπάνω συνόλου στόχων που τίθεται, αποτελεί ικανή και αναγκαία συνθήκη για τον έλεγχο της ορθής λειτουργίας του συστήματος λογισμικού, έστω και περιορισμένης μόνο στο γενικό μοντέλο στόχων που έχει δοθεί;

Στην έγερση των παραπάνω ερωτημάτων, προσπαθεί να δώσει τις απαντήσεις η παρούσα διπλωματική εργασία. Το περιβάλλον-πλαίσιο που παρουσιάζεται, έχει ως στόχο την υλοποίηση των παραπάνω ερωτημάτων, ή το μέρος αυτών που είναι εφικτό.

Όσον αφορά τον πριν την χρήση έλεγχο λογισμικού, ενδιαφέρον παρουσιάζουν οι εξής πολύ απλές, αλλά σημαντικές καταστάσεις στις οποίες μπορεί να βρεθεί ένα σύστημα λογισμικού μεγάλης κλίμακας και οι οποίες μπορεί να υπονομεύσουν την εκτέλεσή του.

1. Κατά την ομαλή λειτουργία του συστήματος, την χρονική στιγμή t, ένα υποσύστημα παρουσιάζει ελαττωματική λειτουργία και τελικά βγαίνει εκτός λειτουργίας. Ποια υποσυστήματα επηρεάζονται και τι αντίκτυπο έχει η κατάσταση αυτή στην λειτουργία του συστήματος;

2. Κατά την ομαλή λειτουργία του συστήματος, την χρονική στιγμή t, ένα υποσύστημα αποσύρεται για να δώσει την θέση του σε ένα άλλο, για λόγους είτε διόρθωσης λαθών, είτε απλής αναβάθμισής του. Μετά από αυτήν την διαδικασία, πληρούνται οι λειτουργικές απαιτήσεις του συστήματος, όπως ορίσθηκαν από τον αρχιτέκτονά του και το σύστημα θα συνεχίσει να λειτουργεί ομαλά όπως πριν;

3. Κατά την ομαλή λειτουργία του συστήματος, την χρονική στιγμή t, ένα υποσύστημα προστίθεται στο σύστημα. Στην περίπτωση αυτή α) το σύστημα ικανοποιεί τις λειτουργικές απαιτήσεις, όπως τις ικανοποιούσε προηγουμένως και β) στην περίπτωση που το νέο υποσύστημα περιλαμβάνει νέες λειτουργικές απαιτήσεις, μπορεί να βρεθεί αν το συνολικό σύστημα τις επαληθεύει και αυτές;

1.4.1 Συνεισφορά της διπλωματικής εργασίας

Η παρούσα διπλωματική εργασία, παρουσιάζει τον σχεδιασμό και την υλοποίηση ενός περιβάλλοντος-πλαισίου λογισμικού, το οποίο καλείται να λύσει το πρόβλημα του πριν την χρήση ελέγχου λογισμικού. Στα πλαίσια αυτού του σκοπού έγιναν τα εξής:

- Ορισμός μιας αρχιτεκτονικής αναφοράς η οποία δανειζόμενη χαρακτηριστικά από αυτόνομα συστήματα, έχει την δυνατότητα μέσω αρθρωμάτων πολιτικών να επαληθεύει την διαμόρφωση του υπό έλεγχο συστήματος λογισμικού, κατά τις αλλαγές κατάστασής του.

- Επέκταση του Μοντέλου Δέντρου Στόχων, που αποτελεί εργαλείο της Μηχανικής Απαιτήσεων (Requirements Engineering), με επισημειώσεις που συνεπικουρούν τις πολιτικές που αναφέρθηκαν προηγουμένως. Το μοντέλο αυτό, αποτελεί αναφορά για την μοντελοποίηση υπηρεσιοκεντρικών συστημάτων λογισμικού, με υποστήριξη αρθρωμάτων πολιτικών επαλήθευσης.

- Παρουσίαση αλγορίθμων που ενορχηστρώνουν τα παραπάνω, στο σκοπό του πριν την χρήση έλεγχο της διαμόρφωσης υπηρεσιοκεντρικών συστημάτων, σε περίπτωση ενημέρωσης ενός δομοστοιχείου.

1.5 Οργάνωση του Κειμένου

Στο 2^{ο} κεφάλαιο, παρουσιάζονται όλες οι σχετικές εργασίες πριν την παρούσα, οι οποίες βοήθησαν με διάφορους τρόπους στην εκπόνηση της παρούσας διπλωματικής. Αναφέρονται οι έννοιες της μοντελοποίησης λογισμικού αλλά και η ιδέα των Δέντρων Στόχων στις οποίες βασίστηκε το στάδιο σχεδιασμού του υπό έλεγχο συστήματος. Παρουσιάζονται, επίσης όλα τα εργαλεία που βοήθησαν στον σχεδιασμό και την υλοποίηση του περιβάλλοντος-πλαισίου, με πιο σημαντικά το Ενοποιημένο Περιβάλλον Ανάπτυξης Λογισμικού Eclipse και ιδιαιτέρως το Eclipse Modeling Framework. Ιδιαίτερη μνεία γίνεται και στο πρότυπο SCA, το οποίο χρησιμοποιήθηκε ως προγραμματιστικό μοντέλο, για υπηρεσιοκεντρικά συστήματα λογισμικού.

Στο 3^{ο} κεφάλαιο, παρουσιάζονται τα αποτελέσματα της σχεδίασης του περιβάλλοντος-πλαισίου. Δίνεται η αρχιτεκτονική του περιβάλλοντος-πλαισίου και αναλύονται δομοστοιχείο προς δομοστοιχείο, όλα τα χαρακτηριστικά που την απαρτίζουν. Ορίζεται επίσης, η διαπροσωπία του περιβάλλοντος-πλαισίου και των δομοστοιχείων που το αποτελούν.

Στο 4^{ο} κεφάλαιο, παρουσιάζεται, ένα τροποποιημένο μοντέλο δέντρου στόχων, το οποίο χρησιμοποιείται για την μοντελοποίηση του υπό έλεγχο συστήματος. Το μοντέλο αυτό, έχει ως σκοπό την μοντελοποίηση των λειτουργικών και δομικών απαιτήσεων του υπό έλεγχο συστήματος λογισμικού, ώστε αυτή η πληροφορία να είναι δομημένη και εύκολη στην χρήση της προγραμματιστικά, από το περιβάλλον-πλαίσιο. Οι δομικές απαιτήσεις που χρειάζονται, μοντελοποιούνται απευθείας από τη δομή του Δέντρου Στόχων, ενώ οι λειτουργικές απαιτήσεις μοντελοποιούνται (στο μέτρο που είναι απαραίτητες για το περιβάλλον-πλαίσιο), μέσω ειδικών επισημειώσεων που επισυνάπτονται σε κάθε στόχο.

Στο 5^{ο} κεφάλαιο, παρουσιάζονται οι στρατηγικές επαλήθευσης που μπορεί να ακολουθήσει το περιβάλλον-πλαίσιο καθώς και οι βασικοί αλγόριθμοι που χρησιμοποιούνται από το περιβάλλον-πλαίσιο για τον καθορισμό των κυρίων στόχων και την επαλήθευσή τους.

Στο 6^{ο} κεφάλαιο παρουσιάζεται ένα ολοκληρωμένο παράδειγμα χρήσης του περιβάλλοντος-πλαισίου, για τον πριν την χρήση έλεγχο ενός συστήματος που ακολουθεί την αρχιτεκτονική SCA (Service Component Architecture).

Στο τελευταίο, 7^{ο} κεφάλαιο, συνοψίζονται τα αποτελέσματα της διπλωματικής αυτής εργασίας. Αναφέρονται ποια ερωτήματα λύθηκαν με την εκπόνησή της, καθώς και ποια μείνανε ανοιχτά. Τέλος, σκιαγραφείται προτεινόμενη μελλοντική δουλειά, που ενδεχομένως κληθεί να λύσει τα ανοιχτά ζητήματα.

Κεφάλαιο 2: Σχετικές Εργασίες

2.1 Εισαγωγή

Στο παρόν κεφάλαιο, γίνεται μια ανασκόπηση όλων εκείνων των τεχνολογιών που χρησιμοποιήθηκαν στα στάδια του σχεδιασμού, υλοποίησης και ελέγχου του περιβάλλοντος-πλαισίου το οποίο αποτέλεσε το αντικείμενο της εκπόνησης της παρούσας διπλωματικής εργασίας.

Το πρώτο και το πιο βασικό στάδιο κατά την ανάπτυξη σύγχρονου λογισμικού είναι το στάδιο του σχεδιασμού. Μια σωστή σχεδίαση, είναι απαραίτητη ώστε το σύστημα λογισμικού να μπορεί να πληροί τις βασικές προϋποθέσεις για μακροβιότητα, επαλήθευση και αξιοπιστία. Ιδιαίτερα στην περίπτωση ενός περιβάλλοντος-πλαισίου, το οποίο δεν προβλέπεται μονάχα να χρησιμοποιείται αυτούσιο από τους χρήστες, αλλά να επεκτείνεται και να τροποποιείται από αυτούς, η σχεδίαση είναι κρίσιμη και να περιλαμβάνει τον πλήρη ορισμό της διαπροσωπίας του περιβάλλοντος-πλαισίου. Η διαπροσωπία δίνει την δυνατότητα επαναχρησιμοποίησης του περιβάλλοντος-πλαισίου και δίνει την δυνατότητα στους χρήστες του να έχουν μια γρήγορη και πλήρη εικόνα για την λειτουργία του χωρίς να αναγκαστούν να μπουν σε λεπτομέρειες του κώδικα.

Το περιβάλλον-πλαίσιο που αναπτύξαμε, ακολούθησε το αντικειμενοστραφές μοντέλο. Για την μοντελοποίηση συστημάτων λογισμικού με χρήση αντικειμενοστρέφειας χρησιμοποιούνται κατά κόρον εργαλεία όπως η UML. Βασισμένο στην UML και ένα πολύ χρήσιμο εργαλείο για την μοντελοποίηση αντικειμενοστραφών συστημάτων είναι το EMOF (Extended Meta-Object Facility) του OMG. Τα παραπάνω αποτελούν την βάση για το ευρέως διαδεδομένο εργαλείο μοντελοποίησης συστημάτων λογισμικού EMF (Eclipse Modeling Framework) το οποίο χρησιμοποιήθηκε κατά κόρον κατά την μοντελοποίηση του περιβάλλοντος-πλαισίου που αναπτύχθηκε.

Όπως έχει αναφερθεί και στο κεφάλαιο της εισαγωγής, το περιβάλλον-πλαίσιο που παρουσιάζεται στην παρούσα διπλωματική, αναλαμβάνει να ελέγξει πριν την χρήση ένα υπάρχον σύστημα λογισμικού. Για να γίνει αυτό, θα πρέπει με κάποιον τρόπο, το σύστημα λογισμικού που θα ελεγχθεί

να μοντελοποιηθεί με βάση τις λειτουργικές απαιτήσεις με τις οποίες σχεδιάστηκε. Η Τεχνολογία Λογισμικού βρίθει από τρόπους και εργαλεία για μοντελοποίηση των λειτουργικών απαιτήσεων ενός συστήματος λογισμικού. Για λόγους που θα περιγράψουμε παρακάτω στο κεφάλαιο αυτό, ως καταλληλότερος τρόπος μοντελοποίησης των λειτουργικών απαιτήσεων του λογισμικού που θέλουμε να ελέγξουμε, θα χρησιμοποιήσουμε τα Δέντρα Στόχων, τα οποία εισήχθησαν στο άρθρο [2].

Στο παρόν κεφάλαιο, παρουσιάζονται όλα τα παραπάνω και εξηγείται ποια χαρακτηριστικά τους βοήθησαν στον σχεδιασμό και την υλοποίηση του περιβάλλοντος-πλαισίου που παρουσιάζουμε, καθώς και τα εργαλεία που συνέβαλαν κατά τον σχεδιασμό και υλοποίηση του περιβάλλοντος-πλαισίου.

2.2 Μοντελοποίηση Συστημάτων Λογισμικού

Σημαντικό κομμάτι της Τεχνολογίας Λογισμικού είναι η σχεδίαση. Η σύγχρονη τάση στην σχεδίαση λογισμικού, επιβάλλει αντικειμενοστραφές μοντέλο.

Το μοντέλο αυτό δανείζεται για την σχεδίαση συστημάτων, το μοντέλο του κόσμου που αποτελείται από διαφορετικά αντικείμενα τα οποία αλληλεπιδρούν μεταξύ τους. Τα *αντικείμενα* αυτά μπορούν να κατηγοριοποιηθούν χρησιμοποιώντας *κλάσεις* οι οποίες είναι συλλογές αντικειμένων που παρουσιάζουν παρόμοια δομικά και λειτουργικά χαρακτηριστικά. Τα αντικείμενα που ανήκουν στην ίδια κλάση, μοιράζονται την κοινή δομή ενώ μπορούν να επιτελούν τις ίδιες λειτουργίες. Οι κλάσεις επίσης μπορούν να χρησιμοποιούν το χαρακτηριστικό της *κληρονομικότητας*. Μια κλάση μπορεί να *κληρονομεί* μια άλλη κλάση. Έτσι, μοιράζεται τα περισσότερα στοιχεία της, ενώ μπορεί ορίζοντας επιπλέον λειτουργίες να επεκτείνει την αρχική λειτουργικότητα της κλάσης. Τα διάφορα αντικείμενα, μπορούν να ανταλλάσσουν *μηνύματα* μεταξύ τους. Τα μηνύματα που μπορούν να ανταλλάσσουν μεταξύ τους είναι προκαθορισμένα και συνήθως ορίζονται στις κλάσεις από τις οποίες απορρέουν τα αντικείμενα. Το σύνολο των μηνυμάτων που μπορεί ένα αντικείμενο να δέχεται ονομάζεται διαπροσωπία του αντικειμένου. Τα μηνύματα που ανταλλάσσονται, στις περισσότερες γλώσσες προγραμματισμού αντιστοιχούν με τις μεθόδους που μια κλάση περιλαμβάνει.

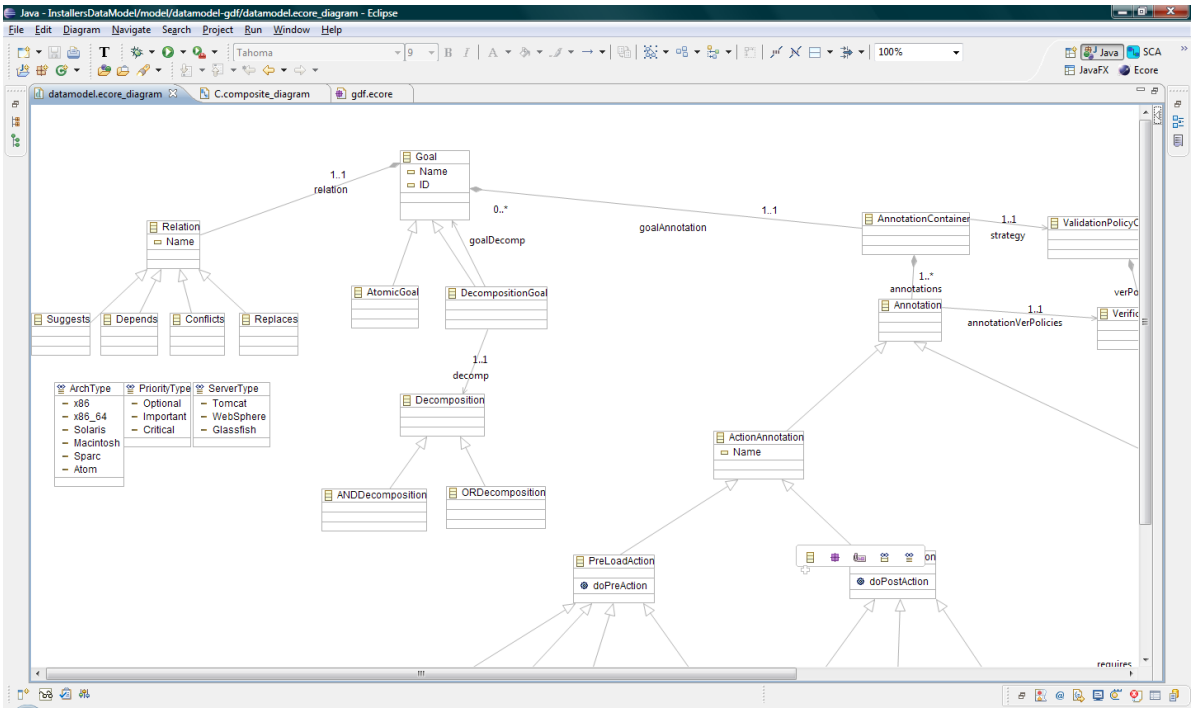
Η παραπάνω παράγραφος, περιέγραψε ακροθιγώς το αντικειμενοστραφές μοντέλο προγραμματισμού, που αποτελεί το σύνηθες πρότυπο που ακολουθούν τα σύγχρονα συστήματα λογισμικού. Πληθώρα εργαλείων έχει δημιουργηθεί, ώστε να βοηθήσει τους παραγωγούς λογισμικού στην αντικειμενοστραφή μοντελοποίηση. Παρακάτω, παρουσιάζεται ένα από τα πλέον διαδεδομένα τέτοια εργαλεία, το Eclipse Modeling Framework.

2.2.1 Eclipse Modeling Framework

Ένα από τα πιο διαδεδομένα και πλούσια σε δυνατότητες εργαλεία που βοηθούν την μοντελοποίηση συστημάτων λογισμικού, είναι το Eclipse Modeling Framework [3]. Όπως αναφέρεται στην αντίστοιχη παραπομπή [3], αυτό το προσάρτημα λογισμικού (plug-in) στο Ενοποιημένο Περιβάλλον Ανάπτυξης Λογισμικού Eclipse ([4]), αποτελεί (όπως αναφέρεται και στο site) *ένα περιβάλλον-πλαίσιο μοντελοποίησης και αυτόματης παραγωγής κώδικα για την δημιουργία εργαλείων και εφαρμογών που βασίζονται σε δομημένα και ιεραρχικά μοντέλα δεδομένων*.

Το EMF, στηρίζεται σε δυο ευρέως διαδεδομένα πρότυπα του Object Management Group, το Meta-Object Facility (MOF, [5] και [6]) και το XMI (XML Metadata Interchange, [7]). Οι προδιαγραφές αυτών των προτύπων βρίσκονται στην ιστοσελίδα [8].

Το πρότυπο MOF, αποτελεί ένα μεταμοντέλο που δημιουργήθηκε ώστε να περιλαμβάνει τα πιθανά μοντέλα αντικειμενοστραφούς σχεδιασμού συστημάτων λογισμικού. Βασίζεται στην UML ([9]), η οποία αποτελεί την πρώτη ευρέως διαδεδομένη γλώσσα περιγραφής αντικειμενοστραφών συστημάτων. Μέσω του μοντέλου MOF, με την αρωγή του EMF, μπορεί κανείς να περιγράψει ένα δομημένο και ιεραρχικό μοντέλο δεδομένων, που σκοπό έχει να αποτελέσει την προδιαγραφή των δεδομένων που θα χρησιμοποιεί το σύστημα λογισμικού, ή και την προδιαγραφή του ίδιου του συστήματος λογισμικού. Έπειτα, το EMF μπορεί να δημιουργήσει αυτόματα κώδικα, ο οποίος να μπορεί να δημιουργεί στιγμιότυπα των μοντέλων που έχει περιγράψει. Ένα στιγμιότυπο λειτουργίας του EMF φαίνεται στην παρακάτω εικόνα, όπου φαίνεται και τμήμα του Μοντέλου Δέντρου Στόχων που θα παρουσιαστεί στο κεφάλαιο 4.



Εικόνα 2-I: Στιγμιότυπο λειτουργίας Eclipse Modeling Framework

Το πρότυπο XMI, αποτελεί ένα πρότυπο που στηρίζεται στην γλώσσα σήμανσης XML ([10]) και επιτρέπει την ανταλλαγή μοντέλων που έχουν περιγραφεί κατά MOF μεταξύ εφαρμογών, αλλά και να χρησιμοποιούνται από εφαρμογές. Το EMF, χρησιμοποιεί κατά κόρον το XMI και ως μέσο ανταλλαγής MOF μοντέλων μεταξύ εφαρμογών και ως μέσο αρχικοποίησης/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 του, για καλύτερη χρήση των προτύπων MOF και XMI.

Το EMF χρησιμοποιήθηκε στην παρούσα διπλωματική, κυρίως στο στάδιο της μοντελοποίησης του τροποποιημένου Μοντέλου Δέντρου Στόχων, που χρειάστηκε να ορισθεί για τις ανάγκες του περιβάλλοντος-πλαισίου. Αρχικά, δημιουργήθηκε με βάση το MOF, το μοντέλο του τροποποιημένου Δέντρου Στόχων και έπειτα χρησιμοποιήθηκε ο παραγόμενος κώδικας χειρισμού του μοντέλου δέντρου στόχων καθώς και το EMF API, ώστε να υλοποιηθούν οι αλγόριθμοι δημιουργίας και διαχείρισης του Δέντρου Στόχων, για τις ανάγκες του περιβάλλοντος-πλαισίου.

2.3 Δέντρα Στόχων (Goal Trees)

Ένα από τα μεγαλύτερα κεφάλαια στην Τεχνολογία Λογισμικού είναι η μοντελοποίηση των λειτουργικών απαιτήσεων ενός συστήματος. Με τον όρο λειτουργικές απαιτήσεις ενός συστήματος λογισμικού, εννοούμε τις ενέργειες που πρέπει αυτό να επιτελεί, δηλαδή δεδομένης της εισόδου στο σύστημα λογισμικού, τι έξοδο περιμένουμε από αυτό. Για παράδειγμα, σε έναν αθροιστή ακεραίων,

μια λειτουργική απαίτηση είναι δεδομένων δυο ακεραίων α και β, να δίνει ως έξοδο το άθροισμά τους α+β.

Για την μοντελοποίηση των λειτουργικών απαιτήσεων ενός συστήματος λογισμικού έχει προταθεί πληθώρα μεθόδων, ενώ έχουν υλοποιηθεί αρκετά εργαλεία που στηρίζονται σε αυτές τις μεθόδους για την βοήθεια του αρχιτέκτονα λογισμικού. Για τις ανάγκες της παρούσας διπλωματικής εργασίας, η προσοχή στράφηκε στα δέντρα στόχων, τα οποία πρωτοεισήγαγε στο χώρο της μηχάνευσης και μοντελοποίησης απαιτήσεων λογισμικού (Requirements Engineering), το [2].

Το έργο αυτό, υπήρξε μια μεταφορά της σκέψης της μοντελοποίησης απαιτήσεων λογισμικού, από αντικειμενοστραφή πρότυπα, σε πρότυπα οδηγούμενα από στόχους. Βασική ιδέα του [2], είναι πως σε ένα σύστημα λογισμικού, οι λειτουργικές του απαιτήσεις, μπορεί να μοντελοποιηθούν μέσω στόχων, οι οποίοι δομούνται σε υποστόχους, με βάση μια δενδρικού τύπου ιεραρχία. Κάθε στόχος, χωρίζεται σε υποστόχους, δηλώνοντας τον τύπο της εξάρτησης που μπορεί να έχει από αυτούς. Οι εξαρτήσεις μπορεί να είναι δυο ειδών, ΚΑΙ ή Ή και καθένας τύπος δηλώνει και διαφορετικό τρόπο επαλήθευσης του στόχου – πατέρα από τους στόχους – παιδιά του. Το είδος ΚΑΙ, δηλώνει πως ο στόχος – πατέρας επαληθεύεται όταν επαληθευτούν όλοι οι στόχοι – παιδιά του, ενώ το είδος Ή δηλώνει πως ο στόχος – πατέρας επαληθεύεται όταν επαληθευτεί τουλάχιστο ένας από τους στόχους.

Τέτοιου είδους στόχοι, στο [2], αναφέρονται ως **hard goals**. Αυτοί οι στόχοι, ορίζουν ένα σύνολο υποστόχων, με βάση το οποίο υπάρχει μονοσήμαντη σημασιολογία επαλήθευσης του στόχου – πατέρα και του συνόλου στόχων – παιδιών. Υπάρχουν και περιπτώσεις λειτουργικών απαιτήσεων στην μοντελοποίηση απαιτήσεων λογισμικού, όπου ένας στόχος μπορεί να μην επαληθεύεται μονοσήμαντα από ένα σύνολο υποστόχων, αλλά να υπάρχει ένα είδος συνεισφοράς στον στόχο – πατέρα, από τους στόχους – παιδιά, θετικής ή αρνητικής. Ανάλογα με το ποσοστό της θετικής έναντι της αρνητικής συνεισφοράς, να επαληθεύεται ή όχι ο στόχος. Αυτή η έννοια αναφέρεται στο [2], ως **soft goal**. Σε αυτήν την περίπτωση, για την επαλήθευση ενός στόχου του υπό έλεγχο συστήματος λογισμικού, μετράται η διαφορά της αρνητικής από την θετική συνεισφορά στο στόχο και ανάλογα ο στόχος αναφέρεται ότι επαληθεύεται ή όχι.

Οι ιδέες που αναφέρονται στο [2] επίδρασαν καταλυτικά στην δομή και τις έννοιες που ενσωματώνει το Μοντέλο Δέντρου Στόχων που παρουσιάζεται στην παρούσα διπλωματική και που χρησιμοποιείται ως μοντέλο για το υπό έλεγχο σύστημα λογισμικού, από το περιβάλλον-πλαίσιο. Οι λόγοι που επιλέχθηκε το δέντρο στόχων, ως μέσο μοντελοποίησης των λειτουργικών απαιτήσεων του υπό έλεγχο συστήματος, έναντι κάποιας αντικειμενοστραφούς μεθόδου, είναι οι εξής:

- **Ευκολία στην αυτόματη κατασκευή**. Το δέντρο στόχων είναι δομικά λιγότερο πολύπλοκο από τα περισσότερα αντικειμενοστραφή μοντέλα που χρησιμοποιούνται για την μοντελοποίηση απαιτήσεων λογισμικού. Αυτό συνεπάγεται ευκολία και ταχύτητα στην παραγωγή και την διαχείρισή του από τον κώδικα του περιβάλλοντος-πλαισίου.

- **Σχέση με Στοιχειώδη Λογική Άλγεβρα**. Το δέντρο στόχων, όπως περιγράφεται στο [2] και χρησιμοποιείται και στην παρούσα διπλωματική, αποτελεί ουσιαστικά μια δενδρική αναπαράσταση εκφράσεων της δίτιμης άλγεβρας του Boole ([11]). Αυτό δίνει μια πληθώρα αλγορίθμων που εφαρμόζονται και στην άλγεβρα Boole, οι οποίοι μπορούν να χρησιμοποιηθούν και από το περιβάλλον-πλαίσιο.

- **Επεκτασιμότητα.** Το δέντρο στόχων έχει τύχει μεγάλης αποδοχής από την επιστημονική – ερευνητική κοινότητα του χώρου της μοντελοποίησης απαιτήσεων λογισμικού και υπάρχει ένας ικανός αριθμός από εργασίες που αναφέρονται στην αυτόματη εξαγωγή δέντρου στόχων, είτε από κώδικα, είτε από άλλα μοντέλα. Ενδεικτικά, στην αναφορά [12], έχουμε μεθοδολογία παραγωγής του δέντρου στόχων, από παρωχημένα συστήματα, μέσω αντίστροφης μηχανικής κώδικα (reverse engineering). Συνεπώς το δέντρο στόχων, αποτελεί μοντέλο το οποίο μπορεί εύκολα να επεκταθεί αλλά και να εξαχθεί από υπάρχουσες τεχνικές που χρησιμοποιούνται στην Τεχνολογία Λογισμικού.

Το δέντρο Στόχων, προτιμήθηκε και έναντι της χρήσης διαγραμμάτων κατάστασης (που παρουσιάζονται στο [13]) για την παρακολούθηση και επαλήθευση της καλής λειτουργίας του υπό έλεγχο συστήματος λογισμικού. Ο λόγος για αυτήν την επιλογή, είναι πως σε ένα υπηρεσιοκεντρικό σύστημα λογισμικού, υπάρχουν δομικές εξαρτήσεις μεταξύ παραγωγών και καταναλωτών υπηρεσιών εκτός από λειτουργικές. Τα διαγράμματα κατάστασης μειονεκτούν έναντι των δέντρων στόχων στην μοντελοποίηση δομικών εξαρτήσεων συστημάτων λογισμικού, ενώ τα δέντρα στόχων με την σειρά τους, μειονεκτούν έναντι των διαγραμμάτων κατάστασης στην μοντελοποίηση των λειτουργικών εξαρτήσεων. Εξαιτίας αυτών, στην παρούσα διπλωματική χρησιμοποιούνται μεν δέντρα στόχων, τροποποιημένα δε, ώστε να παρέχουν και την απαραίτητη επιπλέον πληροφορία για τις λειτουργικές απαιτήσεις κάθε στόχου.

2.4 Service Component Architecture (SCA)

Η πιο πρόσφατη και ελπιδοφόρα υλοποίηση προγραμματιστικού παραδείγματος που βασίζεται στο υπηρεσιοκεντρικό πρότυπο ανάπτυξης λογισμικού, είναι η Service Component Architecture (συντομογραφικά SCA). Το πρότυπο αυτό, έρχεται να προσφέρει ένα ολοκληρωμένο και λειτουργικό περιβάλλον μοντελοποίησης υπηρεσιοκεντρικών συστημάτων λογισμικού, με έννοιες

που στηρίζονται στο υπηρεσιοκεντρικό πρότυπο και περιλαμβάνουν μέριμνα για τις πιο διαδεδομένες και σύγχρονες τεχνολογίες ανάπτυξης λογισμικού (Java, BPEL, C++, Spring framework). Το πρότυπο SCA πρωτοπαρουσιάστηκε από την ομάδα OSOA (Open Service Oriented Architecture), ως μια σειρά προτύπων (βρίσκονται στην ιστοσελίδα [14]) που ορίζουν όλες τις πλευρές της μοντελοποίησης ενός υπηρεσιοκεντρικού συστήματος λογισμικού. Το πρότυπο αυτό, χρησιμοποιείται από την παρούσα διπλωματική, ως το πρότυπο της αρχιτεκτονικής των υπηρεσιοκεντρικών συστημάτων που θα ελεγχθούν από το περιβάλλον-πλαίσιο. Με λίγα λόγια, το περιβάλλον-πλαίσιο (στην μορφή που παρουσιάζεται στην παρούσα διπλωματική) αναμένει το υπηρεσιοκεντρικό σύστημα λογισμικού που θα ελέγχει, να είναι μοντελοποιημένο σύμφωνα με το SCA πρότυπο.

Το μοντέλο SCA, στηρίζεται σε δυο πυλώνες. Το μοντέλο συναρμογής (Assembly Model) το οποίο χρησιμοποιείται για την μοντελοποίηση της δομής ενός υπηρεσιοκεντρικού συστήματος λογισμικού και το πλαίσιο πολιτικών (Policy Framework) που χρησιμοποιείται για την μοντελοποίηση περιορισμών ποιότητας της υπηρεσίας του υπηρεσιοκεντρικού συστήματος λογισμικού. Στην συνέχεια αναλύονται οι βασικότερες έννοιες και τα βασικότερα σημεία αυτών των δυο.

2.4.1 Μοντέλο Συναρμογής (Assembly Model)

Στο μοντέλο συναρμογής, το πρότυπο SCA, περιλαμβάνει τα βασικά στοιχεία που συγκροτούν την δομική κατασκευή ενός υπηρεσιοκεντρικού συστήματος λογισμικού, κατά το παράδειγμα SCA. Η προδιαγραφή του Μοντέλου Συναρμογής SCA, μπορεί να βρεθεί online [15].

Το Μοντέλο Συναρμογής, βασίζεται σε πέντε βασικά τεχνουργήματα (artifacts).

1. **Δομοστοιχείο (Component).** Το δομοστοιχείο αποτελεί το βασικό τεχνούργημα SCA το οποίο μπορεί να λειτουργεί ως καταναλωτής ή παραγωγός υπηρεσίας. Το δομοστοιχείο, αποτελεί μια γενική – αφαιρετική περιγραφή επιχειρησιακών λειτουργιών που πρέπει να εμπεριέχει το σύστημα λογισμικού που μοντελοποιείται. Αυτή η γενική – αφαιρετική περιγραφή μπορεί να υλοποιείται με πολλούς τρόπους και από διαφορετικές μεταξύ τους τεχνολογίες. Τα δομοστοιχεία σε ένα SCA σύστημα λογισμικού, μπορεί να περιέχουν υπηρεσίες που προσφέρουν, αναφορές σε υπηρεσίες που καταναλώνουν και να οργανώνονται σε Σύνθετα Δομοστοιχεία.

2. **Σύνθετα Δομοστοιχεία (Composites).** Τα σύνθετα δομοστοιχεία, αποτελούν το κύριο τεχνούργημα οργάνωσης των επιχειρησιακών λειτουργιών σε επιχειρησιακές λύσεις, δηλαδή οργανώνουν τα δομοστοιχεία που προσφέρουν επιχειρησιακή λειτουργικότητα σε αυτόνομα πακέτα επιχειρησιακής λογικής. Ένα σύνθετο δομοστοιχείο, μπορεί να παρέχει

υπηρεσίες στο περιβάλλον του, μέσω προώθησης των αιτήσεων σε αντίστοιχη υπηρεσία που προσφέρει ένα δομοστοιχείο που ανήκει σε αυτό (διαδικασία που ονομάζεται **προαγωγή υπηρεσίας – service promote**), ενώ όσα δομοστοιχεία περιέχουν αναφορές που έχουν πέρας υπηρεσίες εκτός του Σύνθετου Δομοστοιχείου, οι αναφορές αυτές μεταφέρονται προς το περιβάλλον του Σύνθετου Δομοστοιχείου με παρόμοιο τρόπο (διαδικασία **προαγωγής για αναφορές – reference promote**).

3. **Υπηρεσία (Service).** Η υπηρεσία αποτελεί το άτομο της επιχειρησιακής λογικής που καλείται να παρέχει/υλοποιήσει το μοντελοποιούμενο σε SCA υπηρεσιοκεντρικό σύστημα λογισμικού. Οι υπηρεσίες προσφέρονται από τις εκάστοτε υλοποιήσεις των δομοστοιχείων. Σε ορισμένες περιπτώσεις, είναι εφικτό μέσω της διαδικασίας της προαγωγής υπηρεσίας να προωθούνται αιτήσεις χρήσης υπηρεσίας από το περιβάλλον ενός Σύνθετου Δομοστοιχείου, στο Δομοστοιχείο που την προσφέρει.

4. **Αναφορά (Reference).** Η αναφορά, αποτελεί την δήλωση εξάρτησης ενός δομοστοιχείου από ένα άλλο δομοστοιχείο το οποίο προσφέρει μια υπηρεσία που χρειάζεται το δομοστοιχείο που έχει την αναφορά. Η αναφορά έχει αρχή το δομοστοιχείο που χρειάζεται να χρησιμοποιήσει τα αποτελέσματα μιας επιχειρησιακής λειτουργίας (να καταναλώσει μια υπηρεσία που προσφέρεται από άλλο δομοστοιχείο) και πέρας την υπηρεσία που προσφέρει την λειτουργία αυτή. Μια αναφορά μπορεί να έχει περισσότερα του ενός πέρατα και στην περίπτωση αυτή, η επιχειρησιακή λειτουργία αρκεί να προσφερθεί από ένα από τους πιθανούς παρόχους της (ποιος θα επιλεγεί είναι θέμα του περιβάλλοντος εκτέλεσης του SCA υπηρεσιοκεντρικού λογισμικού). Και στις αναφορές υπάρχει η δυνατότητα προαγωγής τους, ώστε επιχειρησιακή λειτουργία που χρειάζεται από ένα δομοστοιχείο, να μπορεί να βρεθεί εκτός του περιβάλλοντος του σύνθετου δομοστοιχείου στο οποίο περιέχεται.

5. **Σύνδεση (Wire).** Η σύνδεση, αποτελεί το συνδετικό κρίκο μεταξύ της αναφοράς και της υπηρεσίας δυο δομοστοιχείων, είτε αυτά ανήκουν στο ίδιο σύνθετο δομοστοιχείο, είτε όχι. Σε οποιαδήποτε προηγούμενη αναφορά του παρόντος κειμένου σε έννοιες σύνδεσης και ανταλλαγής πληροφοριών επιχειρησιακής λογικής μεταξύ δυο δομοστοιχείων, αυτό γίνεται μονάχα στην περίπτωση όπου υπάρχει η κατάλληλη σύνδεση μεταξύ της αντίστοιχης αναφοράς με την αντίστοιχη υπηρεσία. Η σύνδεση, περιέχει πληροφορίες για το δέσιμο (binding), την τεχνολογία δηλαδή που θα χρησιμοποιηθεί από τα δυο συμβαλλόμενα δομοστοιχεία, για την ανταλλαγή αποτελεσμάτων επιχειρησιακών λειτουργιών από το δομοστοιχείο που τα παράγει (έχει την υπηρεσία) στο δομοστοιχείο που τα χρειάζεται (έχει την αναφορά).

Αυτά τα 5 βασικά τεχνουργήματα της δομής ενός SCA συστήματος λογισμικού, αποτελούν μέρη ενός σχήματος xml, το οποίο χρησιμοποιείται για την μοντελοποίηση τέτοιου είδους συστημάτων. Η μοντελοποίηση συνήθως μορφώνεται ανά σύνθετο δομοστοιχείο, σε αρχεία xml του τύπου *name.composite* όπου name, το όνομα του σύνθετου δομοστοιχείου. Στις αναφορές για τα SCA συστήματα λογισμικού υπάρχουν οι ιστοσελίδες όπου κανείς μπορεί να ανατρέξει και να βρει περισσότερες πληροφορίες για το πρότυπο αυτό.

Τα σύνθετα δομοστοιχεία δεν είναι η ιεραρχικά πατρική θα λέγαμε δομή των SCA συστημάτων λογισμικού. Τα σύνθετα δομοστοιχεία αποτελούν οργανώσεις της επιχειρησιακής λογικής που επιχειρείται να μοντελοποιηθεί στο σύστημα. Όσον αφορά την παράταξη του SCA συστήματος λογισμικού, αυτό μπορεί να γίνει σε έναν οι περισσότερους υπολογιστές, αρκεί οι τελευταίοι να συνδέονται μεταξύ τους σε δίκτυο. Για την ανάπτυξη του SCA συστήματος λογισμικού, τα σύνθετα δομοστοιχεία που το απαρτίζουν, χωρίζονται σε **Τομείς** (**Domains**), οι οποίοι κατά κάποιο τρόπο δηλώνουν πώς θα παραταχθούν στους κόμβους του δικτύου, τα σύνθετα δομοστοιχεία. Η φράση "κατά κάποιο τρόπο" αναφέρεται στο γεγονός ότι ορισμένες υλοποιήσεις περιβαλλόντων εκτέλεσης SCA συστημάτων λογισμικού (Tuscany), ορίζουν δικές τους ιεραρχικές δομές, παραδείγματος χάριν Node, οι οποίες αναλαμβάνουν τον διαχωρισμό των σύνθετων δομοστοιχείων (ή ακόμα και δομοστοιχείων) σε υπολογιστικούς κόμβους στο δίκτυο.

2.4.2 Πλαίσιο Πολιτικών (Policy Framework)

Το πλαίσιο πολιτικών στο πρότυπο SCA, αναφέρεται στον τρόπο με τον οποίο μοντελοποιούνται προδιαγραφές ποιότητας των παρεχόμενων υπηρεσιών (Quality of Service, QoS) σε ένα σύστημα λογισμικού που μοντελοποιείται με το παράδειγμα SCA. Το κείμενο που περιγράφει την προδιαγραφή αυτού του μοντέλου μπορεί να βρεθεί στο [16]. Τα βασικά τεχνουργήματα που διέπουν το περιβάλλον-πλαίσιο πολιτικών ενός SCA συστήματος, είναι η **πρόθεση (intent)**, το **σύνολο πολιτικών (PolicySet)** και η **αντιστοίχηση σε προθέσεις (intentMap)**.

Ως πρόθεση σε ένα σύστημα λογισμικού SCA, αναφέρεται μια προδιαγραφή ποιότητας υπηρεσίας που οφείλει αυτό να προσφέρει στο περιβάλλον του. Οι προθέσεις μπορεί να απαιτούνται από μεμονωμένες υπηρεσίες όπου πρέπει να εφαρμόζονται, ή να διέπουν συνολικά την λειτουργία ενός δομοστοιχείου και σε αυτήν την περίπτωση είναι σαν η πρόθεση να απαιτείται από καθεμιά υπηρεσία που προσφέρει το δομοστοιχείο. Προθέσεις, μπορεί να απαιτούνται και από αναφορές και σε αυτήν την περίπτωση υποδηλώνουν προδιαγραφές ποιότητας της υπηρεσίας που πρέπει να πληρούν οι υπηρεσίες στο πέρας της αναφοράς. Μια πρόθεση, εκτός από το όνομά της, οφείλει να δηλώνει τις ιδιότητές της, **περιορισμόςΣε (constrains)** και **εξαρτάται (requires)**. Η πρώτη ιδιότητα, είναι μια έκφραση XPath, η οποία δηλώνει ένα συγκεκριμένο τύπο xml element στην

ιεραρχία των σχημάτων xml που χρησιμοποιούνται για την μοντελοποίηση SCA συστημάτων λογισμικού. Δηλώνει ότι η πρόθεση αυτή περιορίζει (μέσω της προδιαγραφής ποιότητας υπηρεσίας που περιέχει) την λειτουργία των αντίστοιχων τεχνουργημάτων SCA τα οποία επαληθεύουν την έκφραση XPath που δίνεται. Η δεύτερη ιδιότητα, δηλώνει μια σειρά από προθέσεις τις οποίες η πρόθεση που τις αναφέρει, απαιτεί να ικανοποιούνται, ώστε να ικανοποιείται και η ίδια.

Οι προθέσεις, μπορεί να υπάγονται σε δυο κατηγορίες, τις **κατανεμημένες προθέσεις** (**profile intents**) και τις **προσδιορισμένες προθέσεις** (**qualifiable intents**). Η διαφορά τους έγκειται στον τρόπο με τον οποίο μπορούν να ικανοποιούνται από τις εξαρτώμενες σε αυτή προθέσεις (ιδιότητα requires προηγουμένως). Στην περίπτωση των κατανεμημένων προθέσεων, η ικανοποίηση της πρόθεσης απαιτεί την πρότερη ικανοποίηση *όλων* των προθέσεων που αναφέρονται στην ιδιότητα requires. Αντίθετα, στην περίπτωση των προσδιορισμένων προθέσεων, η πρόθεση μπορεί να ικανοποιηθεί από μια εκ περισσότερων, χαμηλότερου επιπέδου, προθέσεις.

Τα σύνολα πολιτικών, αποτελούν τον τρόπο με τον οποίο το μοντελοποιημένο σύστημα λογισμικού SCA, δηλώνει τις πολιτικές που μπορούν να εφαρμοστούν στην σύνδεση (binding) ή υλοποίηση (implementation) του συστήματος, ώστε να ικανοποιούνται οι προθέσεις που απαιτούνται. Τα σύνολα πολιτικών περιγράφονται στο αρχείο *definitions.xml* το οποίο κατά την αρχικοποίηση του περιβάλλοντος εκτέλεσης του SCA συστήματος λογισμικού δηλώνει τις πολιτικές που πρέπει να εφαρμοστούν από το περιβάλλον εκτέλεσης, σε ορισμένα τεχνουργήματα SCA. Το policySet, εκτός από το όνομά του, οφείλει να ορίζει δυο ιδιότητες, την **εφαρμόζεταιΣε (appliesTo)** που αποτελεί την αντίστοιχη XPath έκφραση με αυτήν της ιδιότητας constrains της πρόθεσης και την ιδιότητα **προσφέρει (provide)** που αποτελεί λίστα με τις προθέσεις που ισχυρίζεται ότι προσφέρει το σύνολο πολιτικών. Ένα xml element τύπου policySet μπορεί επίσης να έχει οποιοδήποτε αριθμό από τα εξής στοιχεία – παιδιά:

- **intentMap**
- **policySetReference**
- **wsp:PolicyAttachment**
- **wsp:Policy**
- **wsp:PolicyReference**
- **xs:any**

Από τα παραπάνω, το περιβάλλον-πλαίσιο που παρουσιάζεται στην παρούσα διπλωματική, ασχολείται κυρίως με την πρώτη περίπτωση όπου αποτελεί τον εγγενή τρόπο μοντελοποίησης των πολιτικών που εφαρμόζονται από το περιβάλλον εκτέλεσης SCA συστημάτων λογισμικού, προς ικανοποίηση των απαιτούμενων προθέσεων. Τα υπόλοιπα, αποτελούν είτε ειδικές περιπτώσεις για ολοκλήρωση του περιβάλλοντος-πλαισίου πολιτικών SCA με το περιβάλλον-πλαίσιο WS-Policy, είτε συντομεύσεις για ευκολότερο χειρισμό των intentMaps.

Ένα xml element τύπου intentMap, οφείλει να ορίζει την ιδιότητά του **προσφέρει (provides)**, η οποία αποτελεί λίστα προθέσεων που ικανοποιεί η πολιτική που μοντελοποιείται μέσω του intentMap. Η λίστα αυτή, οφείλει να είναι υποσύνολο της λίστας προθέσεων που αναφέρει ότι προσφέρει η policySet στην οποία ανήκει το intentMap. Επίσης, το xml element τύπου intentMap μπορεί να έχει παιδιά τύπου **qualifier,** ο οποίος μέσω της ιδιότητας **name** του δηλώνει ποια από τα χαμηλότερου επιπέδου πρόθεση προσφέρεται στην περίπτωση της προσδιορισμένης πρόθεσης.

Κεφάλαιο 3: Αρχιτεκτονική του Περιβάλλοντος-Πλαισίου

3.1 Εισαγωγή

Στο κεφάλαιο αυτό, παρουσιάζουμε την δομή της αρχιτεκτονικής του προτεινόμενου συστήματος, το οποίο καλείται να λύσει το πρόβλημα του «Πριν την πτήση» ελέγχου λογισμικού.

Η σχεδίαση ενός τέτοιου συστήματος, σύμφωνα με την Τεχνολογία Λογισμικού, πρέπει αρχικά να περιλαμβάνει όλα τα δομοστοιχεία που είναι απαραίτητα ώστε να εκτελεστούν οι λειτουργίες που απαιτούνται για την επίλυση του προβλήματος. Ένα δεύτερο και πολύ σημαντικό γνώρισμα που θα πρέπει να έχει, είναι η επεκτασιμότητα, κάτι που θα εγγυηθεί την μακροβιότητά του, την ικανότητά του να προσαρμοστεί σε σύγχρονες ανάγκες, αλλά και να εφαρμοστεί σε τεχνολογίες, διαφορετικές από τις αρχικές που είχε στο μυαλό του ο σχεδιαστής του.

Το σύστημα που θα παρουσιάσουμε, εντάσσεται στην κατηγορία των Αυτόνομων Συστημάτων, συστημάτων δηλαδή που μπορούν να επιδεικνύουν κάποια νοημοσύνη κατά την διάρκεια της εκτέλεσής τους. Με τον όρο αυτό, εννοούμε ότι μπορούν να προσαρμόζονται κατά την εκτέλεσή τους στο περιβάλλον λειτουργίας τους και τις διαφορετικές ανάγκες που μπορεί να παραστούν στην διάρκεια ζωής τους. Για να γίνει αυτό εφικτό, ένα τέτοιο σύστημα, οφείλει να έχει ένα βρόχο ανατροφοδότησης (feedback loop) το οποίο θα ενημερώνει ανά πάσα στιγμή το σύστημα για τις αλλαγές κατάστασης που παρουσιάζονται. Επίσης, θα πρέπει να υπάρχει και ένα δομοστοιχείο το οποίο να εγγυάται την αλλαγή της συμπεριφοράς του συστήματος, με βάση την κατάστασή του ή, στην περίπτωση του πριν την χρήση έλεγχο συστημάτων λογισμικού, την κατάσταση του υπό έλεγχο συστήματος.

Για την λειτουργία οποιουδήποτε συστήματος λογισμικού, είναι απαραίτητο πρώτα να του δώσουμε είσοδο, τα δεδομένα πάνω στα οποία θέλουμε να λειτουργήσει. Το σύστημα λογισμικού που προτείνεται, έχει ως είσοδό του πρακτικά ένα άλλο σύστημα λογισμικού, στο οποίο θέλουμε να εφαρμόσουμε τεχνικές ελέγχου πριν την χρήση. Για να γίνει εφικτό να δοθεί ως είσοδος ένα

σύστημα λογισμικού, το μοντελοποιούμε πρώτα με βάση τα Δέντρα Στόχων, που αναφέρθηκαν προηγουμένως. Συνεπώς, εκτός από την παρουσίαση των δομικών στοιχείων του συστήματος λογισμικού που προτείνουμε (την Αρχιτεκτονική του), είναι σημαντικό να δοθεί και το μοντέλο των δεδομένων που θα χρησιμοποιεί, το μοντέλο των Δέντρων Στόχων που θα χρησιμοποιεί το σύστημά μας εσωτερικά, ως είσοδο για να επιτελεί τις λειτουργίες ελέγχου πριν την πτήση. Το Μοντέλο Δέντρου Στόχων, παρουσιάζεται στο κεφάλαιο 4, όμως έχει αρκετά μεγάλη σχέση με την αρχιτεκτονική που παρουσιάζεται στο παρόν κεφάλαιο, καθώς αποτελεί το μοντέλο δεδομένων που χρησιμοποιεί η αρχιτεκτονική.

Με βάση τα παραπάνω, στο παρόν κεφάλαιο αρχικά δίνεται μια εποπτική παρουσίαση της αρχιτεκτονικής του συστήματός μας. Ακολουθεί η δομοστοιχείο προς δομοστοιχείο ανάλυσή του και των λειτουργιών που επιτελεί καθένα από αυτά. Στο τελευταίο μέρος του παρόντος κεφαλαίου, αναφέρονται ποιες σχεδιαστικές επιλογές στην αρχιτεκτονική του συστήματος, βοήθησαν ώστε το σύστημα να πληροί τις απαραίτητες προϋποθέσεις για επεκτασιμότητα, προσαρμοστικότητα της συμπεριφοράς του και στην δυνατότητα υποστήριξης από αυτό, διαφόρων τεχνολογιών.

3.2 Αρχιτεκτονική

Κατά την σχεδίαση ενός σύγχρονου περιβάλλοντος-πλαισίου λογισμικού το πιο σημαντικό μέρος είναι η αρχιτεκτονική του. Με τον όρο αρχιτεκτονική ενός περιβάλλοντος-πλαισίου, εννοούμε τα διάφορα δομοστοιχεία που θα πρέπει να υλοποιεί είτε το ίδιο το περιβάλλον πλαίσιο είτε όσα συστήματα λογισμικού το χρησιμοποιούν ή το επεκτείνουν. Επίσης, στην αρχιτεκτονική ενός περιβάλλοντος-πλαισίου περιλαμβάνονται και οι αλληλεπιδράσεις μεταξύ των βασικών δομοστοιχείων που ορίζονται.

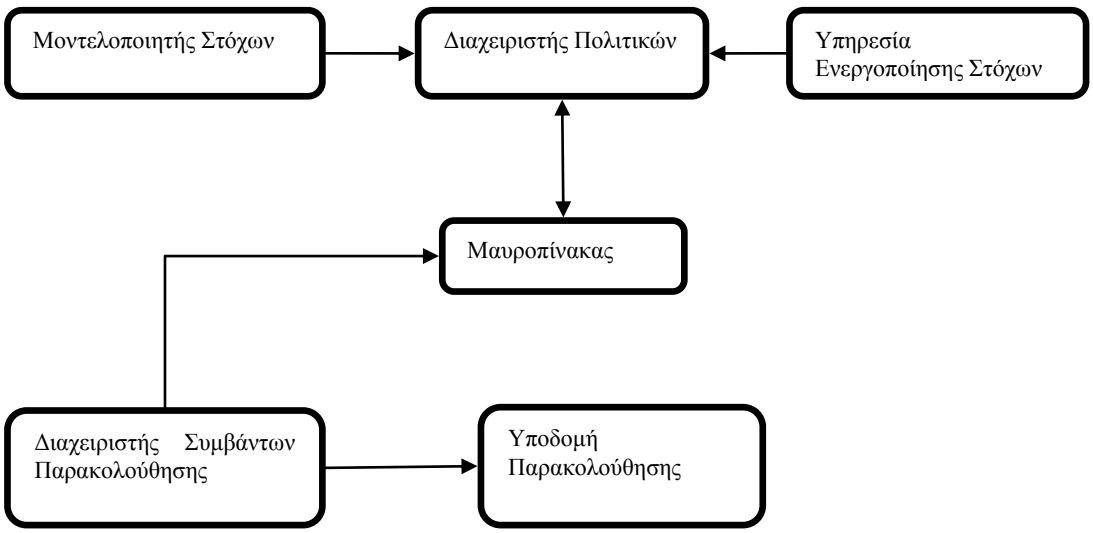
3.2.1 Γενική Επισκόπηση Αρχιτεκτονικής

Τα σύγχρονα συστήματα λογισμικού που βασίζονται σε αρχιτεκτονικές προσανατολισμένες σε υπηρεσίες χαρακτηρίζονται από την κατανεμημένη φύση τους, τον διαχωρισμό διαφόρων λειτουργιών σε διαφορετικά δομοστοιχεία, καθένα από τα οποία επιτελεί είτε συγκεκριμένες είτε μέρος των υπηρεσιών που το σύστημα προσφέρει στους χρήστες του. Το γεγονός αυτό, συνεπάγεται ότι έχουμε διαφορετικές παραμέτρους προερχόμενες από διαφορετικά δομοστοιχεία, που πρέπει να λάβουμε υπόψη μας ώστε να διενεργήσουμε πριν την χρήση έλεγχο της λειτουργίας του συστήματος. Όσο αυξάνεται η πολυπλοκότητα των παραμέτρων ανά δομοστοιχείο αλλά και το πλήθος των δομοστοιχείων που αποτελούν το προς έλεγχο σύστημα λογισμικού, καθίσταται σαφές ότι τακτικές σταθμοσκόπησης (polling) δεν είναι ενδεδειγμένες καθώς θα δημιουργούσαν μεγάλο πρόβλημα απόδοσης του συστήματος. Μια σχεδιαστική λύση σε αυτό το θέμα, που έχει ήδη δοθεί

από την Τεχνολογία Λογισμικού, είναι η χρήση αρχιτεκτονικής τύπου μαυροπίνακα, με στοιχεία έμμεσης κλήσης. Σε αυτήν την περίπτωση, τα δομοστοιχεία που παρακολουθούν το σύστημα, εγγράφουν τις αλλαγές των παραμέτρων του προς ελέγχου συστήματος, στο δομοστοιχείο που λέγεται Μαυροπίνακας. Τα υπόλοιπα δομοστοιχεία, εκδηλώνουν ενδιαφέρον για τις παραμέτρους που μπορούν να επεξεργαστούν και εκτελούνται όποτε κάποια παράμετρος που δήλωσαν, εμφανιστεί στον Μαυροπίνακα.

Η λύση αυτή, είναι προφανές ότι είναι η ενδεδειγμένη για την περίπτωση του ελέγχου υπηρεσιοκεντρικών συστημάτων λογισμικού πριν την χρήση, όπου διαφορετικά δομοστοιχεία θέλουν να επεξεργαστούν διαφορετικές παραμέτρους του υπό έλεγχο συστήματος και αποφεύγεται η τεχνική της σταθμοσκόπησης των παραμέτρων που ενδιαφέρουν κάθε δομοστοιχείο.

Το επόμενο σχήμα, μας δίνει ένα μπλοκ διάγραμμα του περιβάλλοντος-πλαισίου που θα αναπτύξουμε.



Εικόνα 3-Ι: Μπλοκ Διάγραμμα Αρχιτεκτονικής Συστήματος

Όπως φαίνεται και στο σχήμα, η αρχιτεκτονική του συστήματος που προτείνεται αποτελείται από 6 βασικά δομικά στοιχεία:

- **Μαυροπίνακας** – Είναι το βασικό δομοστοιχείο του περιβάλλοντος-πλαισίου και χρησιμοποιείται ώστε να δέχεται την εγγραφή νέων γεγονότων και να ενημερώνει καταλλήλως τα άλλα δομοστοιχεία που έχουν εκδηλώσει ενδιαφέρον για συγκεκριμένο τύπο γεγονότων.

- **Διαχειριστής Πολιτικών** – Είναι το δομοστοιχείο το οποίο είναι υπεύθυνο για την προσαρμοστικότητα του περιβάλλοντος-πλαισίου. Με βάση τα δεδομένα που υπάρχουν στον Μαυροπίνακα αλλά και τον τρέχοντα στόχο από το Δέντρο Στόχων, ο Διαχειριστής Πολιτικών καθορίζει ποιες είναι οι παράμετροι του συστήματος που πρέπει να ελεγχθούν στο εξής και ενεργοποιεί τις κατάλληλες πολιτικές ώστε να επαληθευτούν ή όχι οι τρέχοντες στόχοι, με χρήση πληροφορίας που αντλείται από τις παραμέτρους αυτές.

- **Υπηρεσία Ενεργοποίησης Στόχων –** Το δομοστοιχείο αυτό, είναι υπεύθυνο να θέτει τον επόμενο στόχο που πρέπει να επαληθευτεί, προκειμένου να επαληθευτεί η σωστή λειτουργία του υπό έλεγχο συστήματος. Με αυτόν τον τρόπο ενεργοποιείται ο Διαχειριστής Πολιτικών και να επιτελέσει τις λειτουργίες που αναφέραμε προηγουμένως.

- **Μοντελοποιητής Στόχων** – Το δομοστοιχείο εισαγωγής δεδομένων στο περιβάλλον-πλαίσιο. Περιλαμβάνει όλα τα εργαλεία ώστε να μοντελοποιηθεί σωστά το υπό έλεγχο σύστημα λογισμικού. Επιπλέον, αναλαμβάνει να επιτελέσει την επαλήθευση του παραχθέντος μοντέλου, αλλά και την σωστή φόρτωσή του, προς αρχικοποίηση του περιβάλλοντος-πλαισίου.

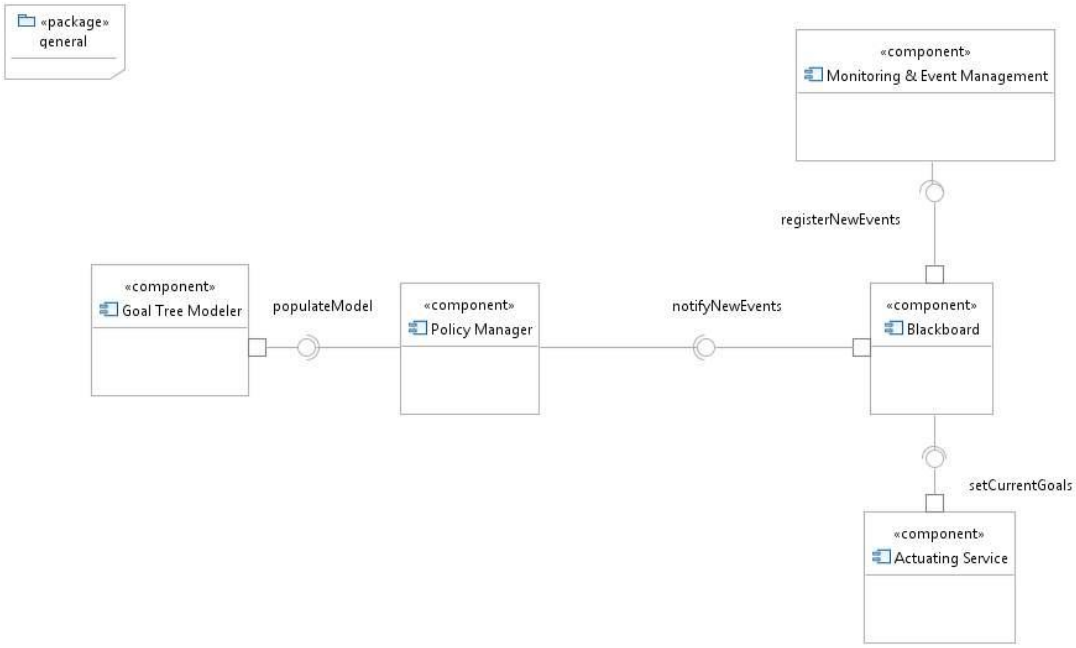
- **Συλλέκτης Συμβάντων Παρακολούθησης** – Το δομοστοιχείο που αναλαμβάνει να ερμηνεύει τα δεδομένα που λαμβάνονται από το σύστημα προς έλεγχο. Με τον όρο "ερμηνεύει", εννοούμε την εγγραφή τους στον Μαυροπίνακα, αλλά και την ενεργοποίηση όλων των κατάλληλων δομοστοιχείων που μπορούν να αντλήσουν πληροφορία για τις παραμέτρους που θεωρεί σημαντικές ο Διαχειριστής Πολιτικών.

- **Υποδομή Παρακολούθησης** – Περιλαμβάνει όλα τα δομοστοιχεία και τις τεχνικές, τα οποία είναι απαραίτητα για την λήψη των κατάλληλων γεγονότων και παραμέτρων από το υπό έλεγχο σύστημα. Τα δεδομένα που παρακολουθούνται, μπορεί να περιέχονται είτε σε αρχεία εγκατάστασης, είτε σε αρχεία ρυθμίσεων είτε να δίνονται από τον χρήστη. Επίσης, δίνεται η δυνατότητα στο σύστημα λογισμικού που θα επεκτείνει το περιβάλλον-πλαίσιο, να ορίσει τις δικές του πηγές των παραμέτρων που παρακολουθούνται, οι οποίες θα μπορούν να χρησιμοποιούνται από τους αλγορίθμους του περιβάλλοντος-πλαισίου με τον ίδιο τρόπο όπως και οι εγγενώς υποστηριζόμενες από το περιβάλλον-πλαίσιο.

3.2.2 Ψηφιδικό Διάγραμμα Αρχιτεκτονικής

Ακολουθεί το ψηφιδικό διάγραμμα της Αρχιτεκτονικής που παρουσιάστηκε προηγουμένως. Με αυτόν τον τρόπο, εκτός από τα διαφορετικά δομοστοιχεία που συνθέτουν την αρχιτεκτονική του περιβάλλοντος-πλαισίου, ορίζουμε και τις αλληλεπιδράσεις που έχουν αυτά μεταξύ τους. Η

παράθεση του διαγράμματος αυτού, αποτελεί την απαρχή της αναλυτικής παρουσίασης του περιβάλλοντος-πλαισίου που σχεδιάσθηκε και της επεξήγησης των διαφόρων λειτουργιών του.



Εικόνα 3-ΙΙ: Ψηφιδικό Διάγραμμα Αρχιτεκτονικής Περιβάλλοντος-Πλαισίου

3.3 Αναλυτική Παρουσίαση Αρχιτεκτονικής

Στο παρόν εδάφιο του τόμου αυτού, θα κάνουμε μια αναλυτική παρουσίαση της Αρχιτεκτονικής που παρουσιάσαμε ακροθιγώς στο προηγούμενο εδάφιο. Αναλύονται σχολαστικά καθένα από τα πέντε δομοστοιχεία που αναφέραμε προηγουμένως, ενώ καθώς παρουσιάζεται η εσωτερική δομή καθενός, αναλύονται και τα δομικά συστατικά του. Βασικό εργαλείο για τα παρακάτω, είναι το ψηφιδικό διάγραμμα που παρουσιάστηκε πιο πάνω, καθώς οι διάφορες πτυχές του αναλύονται στα επόμενα.

3.3.1 Μοντελοποιητής Στόχων

Το δομοστοιχείο αυτό επωμίζεται με τις λειτουργίες που χρειάζονται για την δημιουργία του Δέντρου Στόχου που μοντελοποιεί το προς έλεγχο σύστημα λογισμικού. Είναι το βασικό δομοστοιχείο που επιτρέπει την επικοινωνία του χρήστη με το περιβάλλον-πλαίσιο που παρουσιάζεται και αυτό που αναλαμβάνει την αρχικοποίησή του. Για να επιτευχθούν τα παραπάνω, το δομοστοιχείο αυτό πρέπει να αναλαμβάνει α) να δώσει την δυνατότητα στον χρήστη να μοντελοποιήσει το σύστημα που θέλει να ελεγχθεί β) να μπορεί να φορτώσει οποιεσδήποτε επιπλέον πληροφορίες που χρειάζεται για την λειτουργία του, είτε από αρχεία εγκατάστασης είτε από

διαφόρων ειδών αρχεία ρύθμισης, γ) να υποστηρίζει την δυνατότητα επαλήθευσης του μοντέλου, ώστε να καταστεί εξ' αρχής δυνατό να είναι σίγουρη και εγγυημένη η σωστή λειτουργία του περιβάλλοντος-πλαισίου όσον αφορά την συμβατότητα του δέντρου στόχων με το αναμενόμενο μοντέλο και δ) να δίνεται η δυνατότητα στον χρήστη να επεκτείνει τις υποστηριζόμενες τεχνολογίες στην φόρτωση δεδομένων από αρχεία εγκατάστασης, με ενιαίο τρόπο, με τις λιγότερες δυνατόν αλλαγές στο περιβάλλον-πλαίσιο (π.χ. με απλή υλοποίηση μιας διαπροσωπίας που προσφέρεται).

Βασική εργασία που επιτελείται από τον Μοντελοποιητή Στόχων, είναι η μοντελοποίηση των λειτουργικών απαιτήσεων του υπό ελέγχου συστήματος λογισμικού, δηλαδή η δημιουργία όλων των πτυχών του Δέντρου Στόχων ή των Δέντρων Στόχων, ανάλογα το υπό έλεγχο σύστημα. Η διαδικασία αυτή, μπορεί να γίνει είτε χειροκίνητα από τον χρήστη, ο οποίος θα υπαγορεύσει στο περιβάλλον πλαίσιο την δομή και τα περιεχόμενα του Δέντρου Στόχων ή αυτόματα, με βάση αρχεία ρυθμίσεων για υπηρεσιοκεντρικά (παραδείγματος χάριν SCA) συστήματα και αρχεία εγκατάστασης λογισμικού. Είτε στην περίπτωση που το Δέντρο Στόχων δημιουργηθεί αυτόματα, είτε στην περίπτωση που αυτό δημιουργηθεί χειροκίνητα από τον χρήστη, η διαδικασία που ακολουθείται από τον Μοντελοποιητή Στόχων είναι ακριβώς η ίδια και περιλαμβάνει συγκεκριμένα βήματα.

1. Αρχικά, πρέπει να δημιουργηθεί η δομή του Δέντρου Στόχων. Πρέπει να εντοπιστούν οι εξαρτήσεις AND-OR μεταξύ των δομοστοιχείων ενός υπηρεσιοκεντρικού συστήματος, οι οποίες θα οδηγήσουν στην δημιουργία της μορφής του Δέντρου Στόχων.

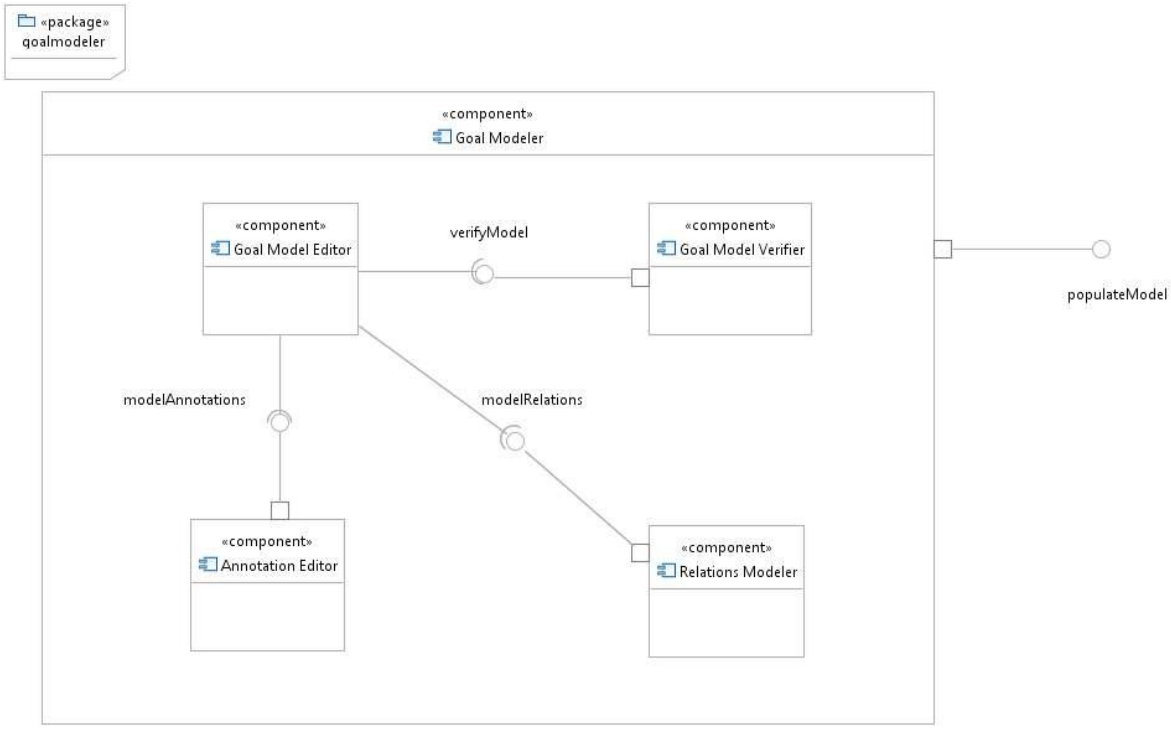
2. Έπειτα, σε κάθε κόμβο του δέντρου, πρέπει να τοποθετηθούν οι κατάλληλες επισημειώσεις. Η διαδικασία αυτή, μπορεί να γίνει και πάλι, είτε με χειροκίνητο τρόπο, είτε με αυτόματο τρόπο. Στην περίπτωση των επισημειώσεων, η πληροφορία που μπορεί να χρησιμοποιηθεί για το υπό έλεγχο σύστημα λογισμικού είναι μεγαλύτερης πολυπλοκότητας, ενώ υπόκειται και σε σημασιολογία που ενδεχομένως είναι γνωστή μόνο στον χρήστη. Οπότε προτείνεται αρχικά η εξαγωγή επισημειώσεων από αρχεία εγκαταστατών ή ρυθμίσεων και η μετέπειτα τροποποίηση/επέκτασή τους από τον χρήστη του περιβάλλοντος-πλαισίου, ως η πιο ενδεδειγμένη προσέγγιση.

3. Το επόμενο βήμα, αναφέρεται στην μοντελοποίηση των σχέσεων μεταξύ στόχων στο Δέντρο. Μια σχέση μεταξύ δυο στόχων, δεν είναι απαραίτητο ότι θα υφίσταται μεταξύ στόχων που ανήκουν στο ίδιο δέντρο. Αυτό είναι γενικά και δεκτό και επιθυμητό, καθότι ο κατανεμημένος χαρακτήρας των υπηρεσιοκεντρικών συστημάτων επιτρέπει εξαρτήσεις μεταξύ δομοστοιχείων που να βρίσκονται όχι μόνο σε ξεχωριστές διεργασίες στον ίδιο ηλεκτρονικό υπολογιστή, αλλά και σε διαφορετικούς υπολογιστές μέσα σε ένα δίκτυο. Για

να γίνει αυτό, κάθε στόχος λαμβάνει ένα χαρακτηριστικό αναγνωριστικό αριθμό, μοναδικό ανά στιγμιότυπο του περιβάλλοντος-πλαισίου που εκτελείται.

4. Έχοντας ολοκληρώσει τα παραπάνω βήματα, στην ουσία έχει ετοιμαστεί πλήρως το Δέντρο Στόχων και με βάση αυτό το περιβάλλον-πλαίσιο θα μπορούσε να λειτουργήσει χωρίς περαιτέρω ενέργειες. Για την καλύτερη διασφάλιση της σωστής λειτουργίας και για την απαλοιφή του λάθους που μπορεί να προκύψει από την παρέμβαση του χρήστη στην δημιουργία του Δέντρου Στόχων, κρίνεται σκόπιμο ένα ακόμη βήμα. Το βήμα αυτό, αποτελεί την επαλήθευση του Δέντρου Στόχων, τον έλεγχο του αν αυτό συμβαδίζει πλήρως με το μοντέλο που αναμένουν τα υπόλοιπα δομοστοιχεία του περιβάλλοντος-πλαισίου. Στο βήμα αυτό, αν επαληθευτεί το Δέντρο Στόχων, αυτό εκτίθεται στον Διαχειριστή Πολιτικών μέσω της διαπροσωπίας *populateModel*. Στην περίπτωση που υπάρξει κάποιο πρόβλημα, ζητείται, μόνο από τον χρήστη πλέον, η διόρθωση όποιων προβλημάτων έχουν προκύψει.

Τα παραπάνω βήματα, αποτελούν τις εσωτερικές διεργασίες που συμβαίνουν στον Μοντελοποιητή Στόχων, για την κατασκευή και επαλήθευση του Δέντρου Στόχων. Λόγω της πολυπλοκότητας που μπορεί να έχει κάθε βήμα, ο σχεδιασμός περιλαμβάνει 4 εσωτερικά δομοστοιχεία του Μοντελοποιητή Στόχων, καθένα από τα οποία αναλαμβάνει και μια από τις παραπάνω διεργασίες. Η εσωτερική δομή σε δομοστοιχεία του Μοντελοποιητή Στόχων δίνεται στο παρακάτω σχήμα και τα δομοστοιχεία που παρουσιάζονται, επεξηγούνται στην συνέχεια.



Εικόνα 3-ΙΙΙ: Ψηφιδικό Διάγραμμα Μοντελοποιητή Στόχων

3.3.1.1 Επεξεργαστής Στόχων

Το δομοστοιχείο που καλείται Επεξεργαστής Στόχων, είναι αυτό που αναλαμβάνει το πρώτο βήμα από τα τέσσερα που παρουσιάστηκαν στο προηγούμενο εδάφιο. Κύριο πρόβλημα που καλείται να λύσει, είναι η αναγνώριση των εξαρτήσεων μεταξύ των στόχων που τίθενται για το υπό έλεγχο σύστημα λογισμικού. Όπως αναφέρεται και στο άρθρο [2] ένα Δέντρο Στόχων ουσιαστικά δομείται αναλύοντας βασικούς στόχους που θέτουμε για το υπό έλεγχο σύστημα, σε απλούστερους που σχετίζονται μεταξύ τους με ρόλους είτε συμπληρωματικούς, είτε ανταγωνιστικούς. Στην πρώτη περίπτωση, έχουμε ο στόχος – πατέρας να συνδέεται με AND με τους στόχους – παιδιά, ενώ στην δεύτερη η σχέση μεταξύ τους είναι OR. Γίνεται φανερό συνεπώς, ότι ο Επεξεργαστής Στόχων πρέπει να αναγνωρίζει την συμπληρωματικότητα ή ανταγωνιστικότητα των αδερφών στόχων και να δημιουργεί το Δέντρο Στόχων, με βάση την απαιτούμενη πληροφορία του υπό έλεγχο συστήματος.

Παρακάτω, υιοθετούμε τον εξής συμβολισμό :

- $\mathbf{G_n}$: Ο n-οστός στόχος που θέτουμε για το σύστημα, χωρίς επισημειώσεις (καθαρός στόχος).
- $\mathbf{C_n}$: Το δομοστοιχείο του υπό έλεγχο συστήματος από το οποίο απορρέει ο στόχος G_n.

- **C** : Το σύνολο των δομοστοιχείων C_i (i=1..n) από τα οποία αποτελείται το υπό έλεγχο σύστημα (όλα τα δομοστοιχεία που αναφέρονται σε ένα αρχείο ρυθμίσεων ενός υπηρεσιοκεντρικό σύστημα).

- **AND(G_j,{G_1,G_2,…,G_m}}** : Κατάσταση όπου η επαλήθευση του στόχου G_j χρειάζεται να επαληθεύονται ταυτόχρονα **όλοι** οι στόχοι $G_{1...m}$.

- **OR(G_j,{G_1,G_2,…,G_m}}** : Κατάσταση όπου η επαλήθευση του στόχου G_j χρειάζεται να επαληθεύεται **τουλάχιστον ένας** από τους στόχους $G_{1...m}$.

- **G** : Το Δέντρο Καθαρών Στόχων (χωρίς επισημειώσεις) που προκύπτει από το σύνολο δομοστοιχείων C και εκφράζεται σαν μια αλληλουχία κατηγορημάτων **AND** και **OR**.

Με βάση τους παραπάνω συμβολισμούς που υιοθετούμε και θα χρησιμοποιούμε στο εξής, μπορούμε να ορίσουμε μαθηματικά τον Επεξεργαστή Στόχων, ως έναν ένα-προς-ένα μετασχηματισμό T_G. Με αυτόν μετασχηματίζεται το σύνολο C των δομοστοιχείων που απαρτίζουν το υπό έλεγχο σύστημα, στο Δέντρο Στόχων G, όπου ορίζονται οι AND-OR σχέσεις μεταξύ των στόχων, χωρίς προς το παρόν να εμφανίζονται οι επισημειώσεις των στόχων:

$$T_G(C) \triangleq G$$

Αξίζει στο σημείο αυτό να διευκρινιστεί, ότι σκοπίμως δεν υπάρχει κάποιος σαφής περιορισμός για την δομή, τον ορισμό ή την απεικόνιση του συνόλου C. Στην παρούσα διπλωματική εργασία, το σύνολο αυτό θεωρείται ως η ιεραρχία των δομοστοιχείων ενός υπηρεσιοκεντρικού συστήματος, όπως θα φανεί πιο κάτω. Αυτό, όμως έχει να κάνει μάλλον με την στόχευση και το πεδίο της παρούσας διπλωματικής εργασίας, παρά με τις δυνατότητες του περιβάλλοντος-πλαισίου που παρουσιάζεται. Οι μόνοι περιορισμοί που τίθενται στα όσα έχουν παρουσιασθεί ως τώρα, έγκειται στην μορφή του συνόλου G, του Δέντρου Στόχων, που χρησιμοποιείται από το περιβάλλον-πλαίσιο και στον μαθηματικό ορισμό του Επεξεργαστή Στόχων. Αυτή η επιλογή, έγινε ώστε να μπορεί το περιβάλλον-πλαίσιο να είναι επεκτάσιμο και να μπορεί να λειτουργήσει και με διαφορετικά αρχικά μοντέλα, στα οποία ενδεχομένως έχουν μοντελοποιηθεί πιθανά υπό έλεγχο συστήματα λογισμικού. Ο ορισμός του εκάστοτε μετασχηματισμού T_G επιβάλλεται στον ενδιαφερόμενο, ώστε να είναι σε θέση αυτός να μοντελοποιήσει τις σχέσεις στο σύνολο C του συστήματός του, που απεικονίζονται σε σχέσεις AND-OR του μοντέλου Δέντρου Στόχων. Έπειτα, το περιβάλλον-πλαίσιο θα μπορεί να λειτουργήσει, με τον ανανεωμένο Επεξεργαστή Στόχων, όπως θα λειτουργούσε και σε υπηρεσιοκεντρικά συστήματα, για τα οποία έχει εγγενή υποστήριξη.

Τα παραπάνω, ορίζουν την διαπροσωπία του δομοστοιχείου Επεξεργαστή Στόχων *prepareModel*. Είναι το πρώτο πράγμα που γίνεται εσωτερικά στον Μοντελοποιητή Στόχων κατά την προσπάθεια

ικανοποίησης μιας αίτησης *populateModel*. Πριν ολοκληρωθεί, όμως, το Δέντρο Στόχων πρέπει να περάσει από τα στάδια της επισύναψης επισημειώσεων και σχέσεων, καθώς και της επαλήθευσης.

3.3.1.2 Επεξεργαστής Επισημειώσεων

Το Δέντρο Στόχων G, που δημιουργείται μέσω της διαπροσωπίας *prepareModel* όπως εξηγήθηκε προηγουμένως, περιέχει μόνο την δομή του Δέντρου Στόχων. Πριν αυτό μπορέσει να χρησιμοποιηθεί από το περιβάλλον-πλαίσιο που παρουσιάζεται, πρέπει να εμπλουτιστεί με τις κατάλληλες επισημειώσεις ανά στόχο, οι οποίες θα παρέχουν την επιπλέον απαιτούμενη πληροφορία ώστε να διεκπεραιωθούν οι επαληθεύσεις των στόχων και να ληφθούν οι απαραίτητες ενέργειες κατά την αλλαγή του Δέντρου Στόχων.

Επεκτείνουμε τον συμβολισμό που ορίσαμε στην προηγούμενη παράγραφο με τα εξής στοιχεία :

- $\mathbf{G_{An}}$: Ο n-οστός στόχος εμπλουτισμένος με τις απαραίτητες επισημειώσεις που δημιουργούνται σε αυτό το βήμα
- $\mathbf{I_n}$: Η πηγή πληροφορίας που χρησιμοποιείται για να δημιουργήσει τις επισημειώσεις του στόχου G_n.
- **I** : Το σύνολο των πηγών πληροφορίας που χρησιμοποιείται ώστε να δημιουργήσει όλες τις επισημειώσεις στόχων για το σύνολο G.
- $\mathbf{A_n}$: Η επισημείωση που πρέπει να εμπλουτίσει την περιγραφή του n-οστού στόχου G_n.
- **A** : Το σύνολο των επισημειώσεων $A_{1...m}$ για κάθε στόχο $G_{1...m}$ του Δέντρου Στόχων G.
- $\mathbf{G_A}$: Το Δέντρο Στόχων, με πανομοιότυπη δομή με το G, που κάθε στόχος $G_{1...m}$ όμως έχει εμπλουτιστεί με την κατάλληλη από τις επισημειώσεις $A_{1...m}$.

Με βάση και τον παραπάνω συμβολισμό, μπορούμε να ορίσουμε το δομοστοιχείο Επεξεργαστή Επισημάνσεων, ως έναν ένα-προς-ένα μετασχηματισμό T_{GA} από το Δέντρο Στόχων G, χωρίς επισημειώσεις, στο Δέντρο Στόχων G_A, με τις απαραίτητες επισημειώσεις, δηλαδή το σύνολο A:

$$T_{G_A}(G) \triangleq G_A$$

Όπως ορίσθηκε ο μετασχηματισμός T_{GA} είναι προφανής, καθώς δεδομένων των επισημειώσεων $A_{1...m}$ για τους στόχους $G_{1...m}$ μπορούμε εύκολα να εξάγουμε το Δέντρο Στόχων G_A από το Δέντρο Στόχων G. Αξίζει επίσης να σημειωθεί ότι η δομή του Δέντρου Στόχων G_A και του G, είναι

πανομοιότυπη, καθώς στο βήμα αυτό, όλοι οι στόχοι $G_{1...m}$ έχουν ήδη αναλυθεί στους επιμέρους τους, από τον Επεξεργαστή Στόχων.

Η κυριότερη εργασία που χρειάζεται να επιτελεσθεί στον Επεξεργαστή Επισημειώσεων, είναι η δημιουργία των επισημειώσεων $A_{1...m}$ από τις αντίστοιχες πηγές $I_{1...m}$. Ορίζουμε την διαδικασία εξαγωγής των επισημειώσεων $A_{1...m}$ από τις αντίστοιχες πηγές $I_{1...m}$ ως έναν μετασχηματισμό T_I

$$T_I(I) \triangleq A$$

Ο μετασχηματισμός T_I όπως ορίσθηκε στην παραπάνω σχέση, δεν είναι ένα-προς-ένα καθώς οι πηγές των επισημειώσεων, που στην γενική περίπτωση μπορεί να είναι οποιαδήποτε αρχεία ρυθμίσεων ή εγκατάστασης, είναι σίγουρα πολυπλοκότερα των επισημειώσεων, οι οποίες εξάγονται από υποσύνολα των πηγών $I_{1...m}$.

Όπως και στην περίπτωση του μετασχηματισμού T_G, ο μετασχηματισμός T_I δεν θέτει κανέναν περιορισμό στην μορφή, τις ιδιότητες ή την απεικόνιση των δεδομένων εισόδου του, του συνόλου δηλαδή I. Ο λόγος που επιλέχθηκε αυτό είναι προφανής, καθώς δίνεται η δυνατότητα και η ελευθερία στον χρήστη του περιβάλλοντος-πλαισίου, να ορίσει τον δικό του μετασχηματισμό T_I ο οποίος θα μετασχηματίζει οποιαδήποτε συλλογή πληροφοριών είναι επιθυμητό στις σαφώς καθορισμένες επισημειώσεις για το Δέντρο Στόχων, που ορίζουμε σε επόμενο κεφάλαιο. Αυτή είναι μια σημαντική σχεδιαστική επιλογή που εξασφαλίζει την επεκτασιμότητα του περιβάλλοντος-πλαισίου, με το μειονέκτημα όμως του μη ελέγχου του μετασχηματισμού T_I για το αν δίδει ή όχι ορθά αποτελέσματα. Αυτό μπορεί να ελεγχθεί μόνο στο τέλος, όπου γίνεται πλήρης έλεγχος του παραγόμενου Δέντρου Στόχων, με το αν πληροί τις προϋποθέσεις που τίθενται από το περιβάλλον-πλαίσιο.

3.3.1.3 Επεξεργαστής Σχέσεων

Το τρίτο βήμα του αλγορίθμου που παρουσιάστηκε αρχικά, προβλέπει την μοντελοποίηση των σχέσεων που ενδεχομένως έχουν στόχοι στο ίδιο ή σε διαφορετικά Δέντρα Στόχων. Κάτι τέτοιο κρίνεται απαραίτητο, καθώς οι διάφορες εξαρτήσεις μεταξύ στόχων δεν μπορούν να μοντελοποιηθούν μόνο με σχέσεις συμπληρωματικότητας ή ανταγωνιστικότητας κατά την διασαφήνιση των υποστόχων ενός στόχου. Ένα χαρακτηριστικό παράδειγμα, αποτελεί η κατάσταση στην οποία δυο στόχοι G_i και G_j εξαρτώνται μεταξύ τους, λόγω κοινής χρήσης ενός πόρου (ενός αρχείου), παρότι η ικανοποίηση του στόχου G_i μπορεί να μην έχει συσχετισμό με τον στόχο G_j.

Για την αντιμετώπιση τέτοιων καταστάσεων, κρίνεται σκόπιμο να ορισθεί το δομοστοιχείο Επεξεργαστής Σχέσεων, το οποίο αναλαμβάνει να μοντελοποιήσει τις επιπλέον σχέσεις που

παρατηρούνται μεταξύ στόχων στο ίδιο ή σε διαφορετικά Δέντρα Στόχων. Ο συμβολισμός, επεκτείνεται ως εξής :

- **G_{AR} :** Το Δέντρο Στόχων με την ίδια δομή με το G, εμπλουτισμένο με τις επισημειώσεις ανά στόχο του G_A που τώρα έχει και τις εξαρτήσεις των στόχων του Δέντρου G_A από στόχους του G_A ή κάποιου άλλου Δέντρου Στόχων.

Με βάση την παραπάνω επέκταση, μπορούμε να ορίσουμε τον Επεξεργαστή Σχέσεων, ως έναν μετασχηματισμό T_{GR} από το Δέντρο Στόχων G_A στο Δέντρο Στόχων G_{AR}.

$$T_{G_R}(G_A) \triangleq G_{AR}$$

Ο μετασχηματισμός αυτός, συνηθέστερα θα είναι αρκετά πιο πολύπλοκος από τους προηγούμενους μετασχηματισμούς που ορίσαμε. Επίσης, ενώ στις προηγούμενες περιπτώσεις υπάρχει δυνατότητα απευθείας εξαγωγής του αποτελέσματος από πηγές πληροφοριών, στην περίπτωση των σχέσεων μεταξύ στόχων, πρέπει να χρησιμοποιηθεί σημασιολογία για τους στόχους, που είναι ενδεχομένως δύσκολο να χειρισθεί αυτόματα από ένα σύστημα λογισμικού.

Σε κάθε περίπτωση, οι σχέσεις που μπορεί να υπάρξουν μεταξύ στόχων του Δέντρου Στόχων είναι σαφώς ορισμένες στο επόμενο κεφάλαιο και μπορούν να χρησιμοποιηθούν στους αλγορίθμους και τεχνικές του περιβάλλοντος-πλαισίου για τον έλεγχο και την προσαρμοστικότητα των πολιτικών διαχείρισης και επαλήθευσης στόχων.

3.3.1.4 Επαληθευτής Μοντέλου Δέντρου Στόχων

Έχοντας δημιουργήσει το Δέντρο Στόχων G_{AR}, έχουμε μοντελοποιήσει πλήρως το υπό έλεγχο σύστημα λογισμικού, σύμφωνα με τις ανάγκες του περιβάλλοντος-πλαισίου που παρουσιάζεται. Όμως, στην παραπάνω περιγραφή των υποσυστημάτων του Μοντελοποιητή Στόχων, μπορεί εύκολα να εντοπισθούν ορισμένα σημεία τα οποία ενδεχομένως να είναι πηγές προβλήματος για την ορθότητα του Δέντρου Στόχων, συνεπώς και για την λειτουργία του περιβάλλοντος-πλαισίου που θα το χρησιμοποιήσει. Ήδη έχει αναφερθεί, ότι οι μετασχηματισμοί T_I και T_G ενδεχομένως να περιλαμβάνουν και χειροκίνητη παρέμβαση από τον χρήστη, καθώς ο μετασχηματισμός T_{GR} στηρίζεται σε σημασιολογία, μη ελεγχόμενη από ένα σύστημα λογισμικού. Όλα αυτά, καθιστούν αναμενόμενη και όχι αδύνατη, την περίπτωση που είτε το Δέντρο Στόχων G_{AR} να έχει κάποια σφάλματα και να προκαλέσει δυσλειτουργίες στο περιβάλλον-πλαίσιο. Για την αντιμετώπιση των παραπάνω, ορίζουμε το δομοστοιχείο Επαληθευτή Μοντέλου Δέντρου Στόχων.

Το δομοστοιχείο που ορίζεται εδώ, επωμίζεται τον έλεγχο του Δέντρου Στόχων G_{AR} ώστε να επαληθευτεί η συμβατότητά του με τις προϋποθέσεις που έχουν τεθεί από το περιβάλλον πλαίσιο.

Από το δομοστοιχείο αυτό, ελέγχεται όσο το δυνατόν πιο εξαντλητικά το Δέντρο Στόχων G_{AR} ώστε να αποφευχθούν προβλήματα, λόγω κακής δημιουργίας του, στην λειτουργία του περιβάλλοντος-πλαισίου.

Με το βήμα αυτό, στην περίπτωση που εξασφαλιστεί η συμβατότητα του παραχθέντος Δέντρου Στόχων G_{AR} με τα αναμενόμενα από το περιβάλλον-πλαίσιο, μπορεί να γίνει η έκθεση του G_{AR} στα υπόλοιπα δομοστοιχεία του περιβάλλοντος-πλαισίου και να αρχίσει η λειτουργία του. Στο βήμα αυτό δηλαδή, με την προϋπόθεση ότι το G_{AR} επαληθεύεται, έχουμε την ολοκλήρωση μιας κλήσης της διαπροσωπίας *populateModel*, του δομοστοιχείου Μοντελοποιητής Στόχων.

3.3.2 Διαχειριστής Πολιτικών

Ο Διαχειριστής Πολιτικών, είναι το δομοστοιχείο που είναι υπεύθυνο για την προσαρμοστικότητα του περιβάλλοντος-πλαισίου στις αλλαγές συνθηκών στο υπό έλεγχο σύστημα λογισμικού. Κάθε αλλαγή στο υπό έλεγχο σύστημα λογισμικού, προκαλεί και την αλλαγή των στόχων που πρέπει να ικανοποιούνται, ώστε να πληρούνται οι προϋποθέσεις που έχουν μοντελοποιηθεί στο Δέντρο Στόχων. Ο Διαχειριστής Πολιτικών, σε κάθε αλλαγή των στόχων που πρέπει να ελεγχθεί αν ικανοποιούνται, οφείλει να ενεργοποιήσει τις πολιτικές επαλήθευσης και ελέγχου που χρειάζονται για την ικανοποίηση των στόχων. Τρία είναι τα βασικά σημεία της λειτουργίας του Διαχειριστή Πολιτικών :

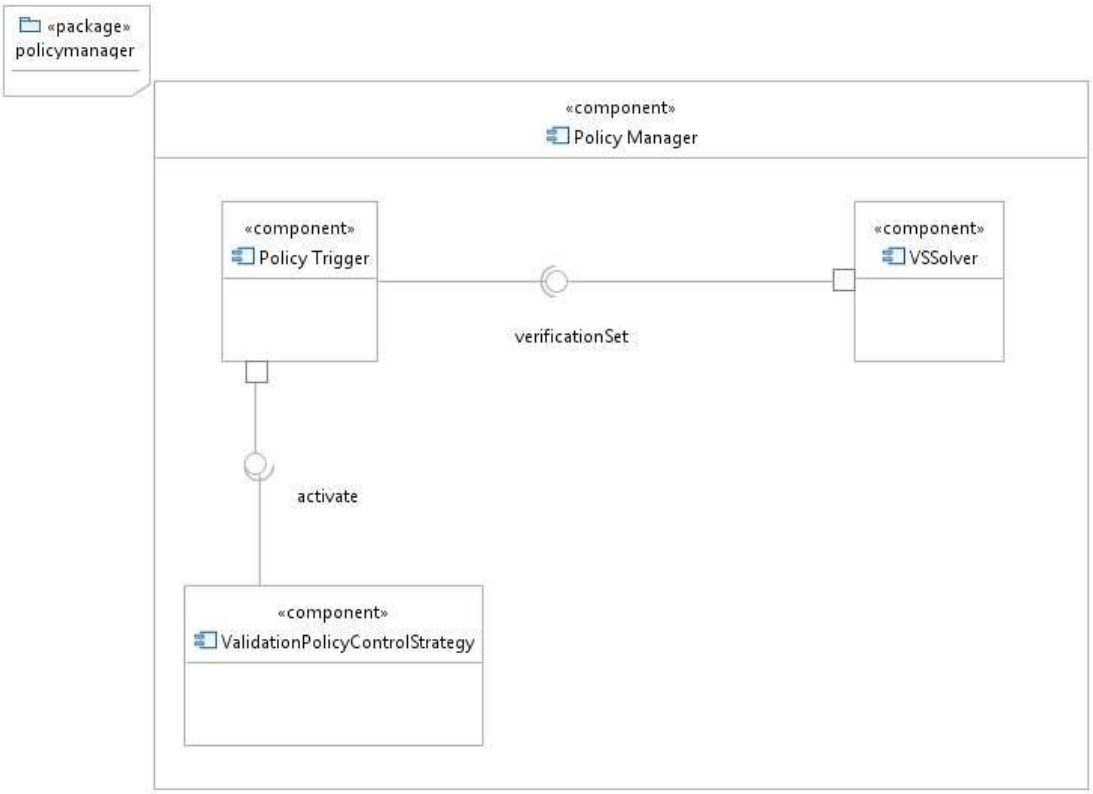
1. Εντοπισμός των πιθανών μονοπατιών στο δέντρο, όπου προκαλούν επαλήθευση του κυρίως στόχου. Αυτό προκύπτει, αν μελετηθεί το Δέντρο Στόχων ως ένα σύνολο από μεταβλητές που παίρνουν μόνο τις τιμές αληθές ή ψευδές (Boolean) οπότε η ικανοποίηση ενός στόχου στο Δέντρο Στόχων, μπορεί να προκύπτει από διαφορετικές αναθέσεις των υποστόχων του σε αληθές ή ψευδές.

2. Με βάση το σύνολο των αναθέσεων που βρέθηκε στο προηγούμενο ερώτημα, πρέπει να αποφασιστεί ποια είναι τα απαραίτητα δεδομένα για να επαληθευτούν ή να απορριφθούν οι στόχοι που έχουν τεθεί ως ενδιαφέροντες στην παρούσα κατάσταση του υπό έλεγχο συστήματος.

3. Ενεργοποίηση των πολιτικών διαχείρισης και των δομοστοιχείων, που μπορούν να συλλέξουν τα απαιτούμενα δεδομένα για την ανάλυση των τρεχόντων στόχων, που προέκυψαν στα προηγούμενα βήματα.

Τα παραπάνω σημεία, συμβαίνουν εσωτερικά στο δομοστοιχείο Διαχειριστής Πολιτικών και για την όσο το δυνατόν μεγαλύτερη επεκτασιμότητα του περιβάλλοντος-πλαισίου, κάθε σημείο

αναλαμβάνεται και από διαφορετικό εσωτερικό δομοστοιχείο του Διαχειριστή Πολιτικών. Παρακάτω ακολουθεί η δομική σύσταση του Διαχειριστή Πολιτικών, δίνεται δηλαδή, το ψηφιδικό διάγραμμά του.



Εικόνα 3-IV: Ψηφιδικό Διάγραμμα Διαχειριστή Πολιτικών

Στην συνέχεια, κάθε δομοστοιχείο που παρουσιάστηκε στην Εικόνα 3-IV θα αναλυθεί πλήρως και ως προς την λειτουργικότητα που προσφέρει στο υπόλοιπο σύστημα λογισμικού, αλλά και ως προς τις εγγενείς του λειτουργίες που οδηγούν στην εκπλήρωση του σκοπού του.

3.3.2.1 Verification Set Solver

Το πρώτο βήμα που πρέπει να γίνει άμα τη εμφανίσει ενός νέου στόχου προς επαλήθευση, είναι να βρεθεί από ποιους υποστόχους αυτός εξαρτάται και με ποιον τρόπο (ποια ανάθεση μεταβλητών δίτιμης λογικής) μπορεί η επαλήθευση των επί μέρους στόχων, να μας δώσει απάντηση για την επαλήθευση του κυρίως στόχου που τέθηκε.

Το πρόβλημα αυτό, οφείλει να δέχεται ως δεδομένο μια έκφραση σε δίτιμη λογική και η έξοδός του είναι μια λίστα από αναθέσεις στις μεταβλητές της έκφρασης, οι οποίες στην αποτίμησή τους, δίνουν αληθή τιμή για τη συνολική έκφραση. Αυτός ακριβώς είναι και ο σκοπός του δομοστοιχείου Verification Set Solver, που λαμβάνει έναν νέο τρέχοντα στόχο που πρέπει να επαληθευτεί και με

βάση το Δέντρο Στόχων (δηλαδή μια σειρά από εκφράσεις δίτιμης λογικής) προσπαθεί να βρει ποιες αναθέσεις στις μεταβλητές στόχους μπορούν να δώσουν την επαλήθευση του τρέχοντα στόχου. Με λίγα λόγια, σκοπός του δομοστοιχείου αυτού, είναι να επιστρέψει ένα σύνολο από εναλλακτικές λίστες με υποστόχους του τρέχοντα στόχου, που η επαλήθευση κάποιας εξ' αυτών, επαληθεύει τον τρέχοντα στόχο.

Το σύνολο των εναλλακτικών λιστών που ορίσθηκε προηγουμένως, εκτίθεται στο περιβάλλον του Verification Set Solver, μέσω της διαπροσωπίας *verificationSet* του. Επόμενο βήμα, αφού έχει υπολογισθεί το *verificationSet* έστω VS, είναι να ενεργοποιηθούν οι κατάλληλες πολιτικές που θα επαληθεύσουν ή θα απορρίψουν τα μέλη του συνόλου VS. Αυτό είναι κάτι που επωμίζεται το επόμενο δομοστοιχείο του Διαχειριστή Πολιτικών, ο Πυροκροτητής Πολιτικών.

3.3.2.2 Πυροκροτητής Πολιτικών – Πολιτικές Επαλήθευσης

Το δομοστοιχείο αυτό, αναλαμβάνει να πυροδοτήσει τις κατάλληλες πολιτικές που χρειάζονται ώστε να επαληθευτούν ή να απορριφθούν οι στόχοι του VS, που υπολογίσθηκε από τον Verification Set Solver.

Στην γενική περίπτωση, οι πολιτικές που μπορεί να είναι αναγκαίες για την επαλήθευση των στόχων, δεν μπορούν να ενταχθούν σε κατηγορίες με βάση την λειτουργία τους. Το μόνο κοινό που έχουν, είναι ότι οφείλουν να ορίζουν μια μεθοδολογία η οποία να είναι επεκτάσιμη και εξελίξιμη και να μπορεί να δώσει την επαλήθευση ή μη του στόχου. Κάθε μια από τις πολιτικές που εννοούνται στα προηγούμενα, πρέπει να σχετίζεται με έναν στόχο από το σύνολο VS των στόχων που συλλέγονται από την διαπροσωπία verificationSet του δομοστοιχείου Verification Set Solver. Από την στιγμή που οι πολιτικές αυτές είναι διαθέσιμες, το δομοστοιχείο, διατρέχει το σύνολο VS και ενεργοποιεί τις πολιτικές που σχετίζονται με κάθε στόχο μέλος του VS.

Όπως είναι προφανές, η πολυπλοκότητα των διαφορετικών ειδών στόχων, μπορεί να προκαλέσει και πολλά διαφορετικά είδη πολιτικών επαλήθευσης που να πρέπει να ενεργοποιηθούν. Για τον λόγο αυτό, όπως θα αναφερθεί και παρακάτω, το μοντέλο του Δέντρου Στόχων επιβάλλει σε κάθε στόχο τον ορισμό των πολιτικών επαλήθευσης που πρέπει να τον συνοδεύουν. Αυτός ο ορισμός, γίνεται με μια αναφορά κάθε στόχου σε μια Πολιτική Επαλήθευσης (ValidationPolicyControlStrategy). Αναφορές σε τέτοια αντικείμενα, έχει και ο Πυροκροτητής Πολιτικών και πιο συγκεκριμένα, αναφορές του Πυροκροτητή Πολιτικών σε Πολιτικές Επαλήθευσης, δημιουργούνται κατά την ανάγνωση του συνόλου VS. Κάθε Πολιτική Επαλήθευσης που εμφανίζεται στους στόχους του συνόλου VS, γίνεται αυτομάτως αναφορά από τον Πυροκροτητή Πολιτικών και στην συνέχεια ενεργοποιείται από αυτόν, ώστε να ξεκινήσει η διαδικασία επαλήθευσης ή απόρριψης του στόχου.

Τα παραπάνω, ορίζουν την διαπροσωπία *activate* του Πυροκροτητή Πολιτικών. Η ενεργοποίησή της προκαλείται έπειτα από χρήση της διαπροσωπίας *verificationSet* του Verification Set Solver (αλλαγή του συνόλου VS) και προκαλεί τις ενέργειες που αναφέραμε προηγουμένως.

3.3.3 Μαυροπίνακας

Το δομοστοιχείο αυτό, είναι η βασική γέφυρα μεταξύ των άλλων δομοστοιχείων της αρχιτεκτονικής του περιβάλλοντος-πλαισίου που παρουσιάζουμε. Όπως σε μια αίθουσα διδασκαλίας, ο μαυροπίνακας αποτελεί τον χώρο για να γραφτούν πράγματα που είναι ορατά σε όλους και χρησιμοποιούνται για την επικοινωνία του εκπαιδευτικού με τους μαθητές, έτσι και στην περίπτωση του Μαυροπίνακα, σαν δομοστοιχείο συστήματος λογισμικού, αποτελεί τον κοινό χώρο να μοιράζονται τα υπόλοιπα δομοστοιχεία γεγονότα και γνώση για το υπό έλεγχο σύστημα λογισμικού. Στην περίπτωση του περιβάλλοντος-πλαισίου που παρουσιάζουμε, για την μείωση των μηνυμάτων που ανταλλάσσονται μεταξύ διαφόρων δομοστοιχείων, επεκτείνουμε την έννοια του Μαυροπίνακα δίνοντάς του στοιχεία Έμμεσης Κλήσης (Implicit Invocation). Δηλαδή, αντί να μοιράζονται όλα τα γεγονότα μεταξύ όλων των δομοστοιχείων, τα δομοστοιχεία δηλώνουν ενδιαφέρον (register) για συγκεκριμένους τύπους γεγονότων (ή μοτίβα γεγονότων – event patterns) και ο Μαυροπίνακας αναλαμβάνει να ειδοποιήσει τα δομοστοιχεία αυτά, στην εμφάνιση των αντίστοιχων μοτίβων γεγονότων.

Από τα παραπάνω, γίνεται φανερό ότι ο Μαυροπίνακας πρέπει εγγενώς να υποστηρίζει τα εξής:

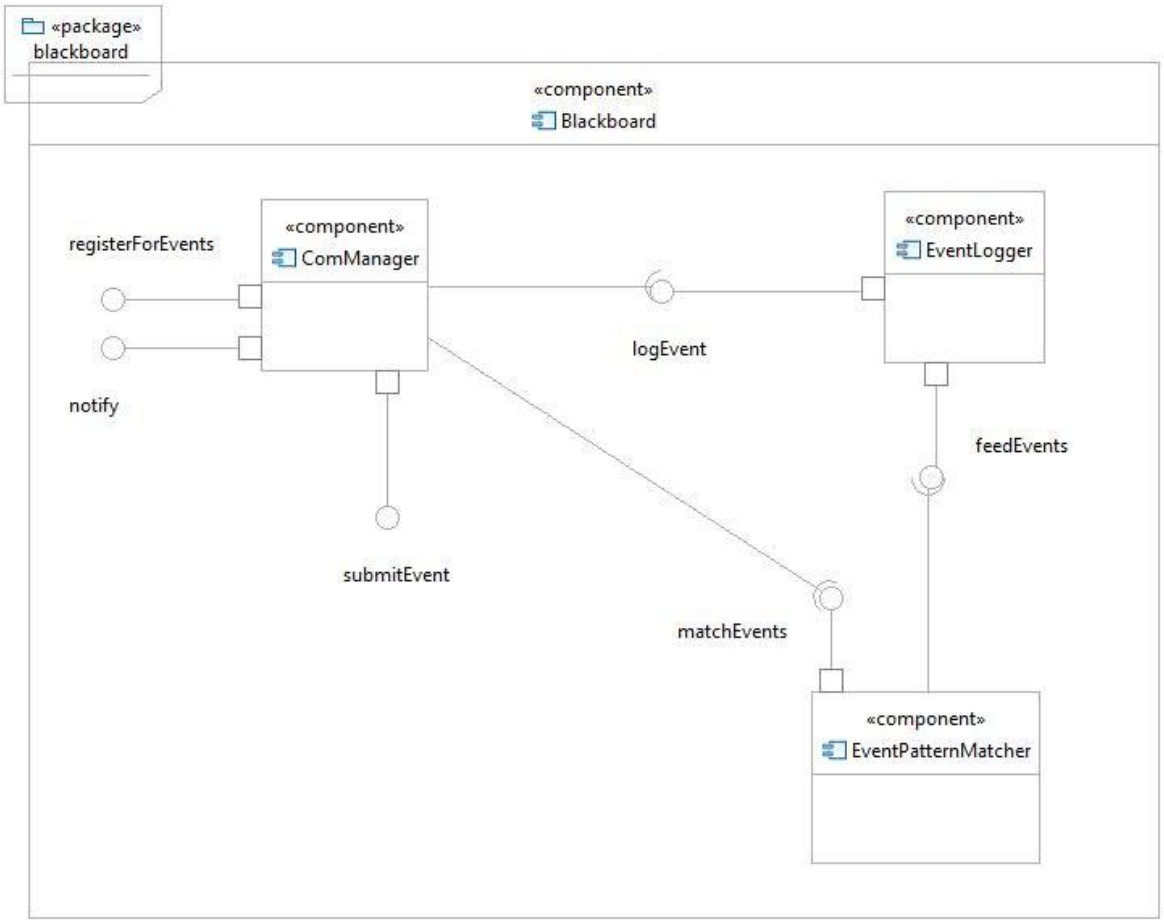
1. **Καταγραφή Γεγονότων** : Μηχανισμός ώστε να καταγράφονται τα γεγονότα που παρουσιάζονται για να είναι δυνατή η επεξεργασία τους και η ενημέρωση των δομοστοιχείων που ενδιαφέρονται. Εδώ, μπορούμε να πούμε ότι το κάθε γεγονός χειρίζεται ως μια εγγραφή σε ένα ημερολόγιο (log entry, timestamps).

2. **Ειδοποίηση** : Μηχανισμός, ώστε με την εμφάνιση μοτίβου ή τύπου γεγονότος για το οποίο έχει εκδηλωθεί ενδιαφέρον από ένα ή περισσότερα δομοστοιχεία, να ενημερώνονται τα δομοστοιχεία αυτά.

3. **Εγγραφή Δομοστοιχείων** : Μέθοδος εκδήλωσης ενδιαφέροντος για συγκεκριμένο τύπο ή μοτίβο γεγονότων από τα δομοστοιχεία του περιβάλλοντος-πλαισίου.

4. **Αναγνώριση Μοτίβου Γεγονότων** : Μηχανισμός αναγνώρισης του μοτίβου γεγονότων και του τύπου γεγονότος, ώστε να είναι σε θέση να ελεγχθεί σε σχέση με τα μοτίβα και τους τύπους που έχουν εγγραφεί τα δομοστοιχεία.

Στην παραπάνω θεώρηση, εύκολα κανείς αναγνωρίζει την συγγένεια των λειτουργιών 2 και 3. Οπότε μπορούμε να πούμε ότι οι παραπάνω λειτουργίες μπορούν να ικανοποιηθούν με 3 εσωτερικά δομοστοιχεία του Μαυροπίνακα. Συγκεκριμένα, ένα δομοστοιχείο θα αναλαμβάνει την καταγραφή των γεγονότων, ένα άλλο δομοστοιχείο θα αναλαμβάνει την ειδοποίηση και εγγραφή των δομοστοιχείων εκτός του Μαυροπίνακα και ένα τρίτο δομοστοιχείο θα αναλαμβάνει την αναγνώριση μοτίβων γεγονότων, ώστε να αναγνωρίζεται πότε πρέπει να γίνει ειδοποίηση. Τα δομοστοιχεία που αναφέραμε παραπάνω, καθώς και η εσωτερική σύσταση του Μαυροπίνακα σε αυτά, φαίνονται στην παραπάνω εικόνα, το ψηφιδικό διάγραμμα του Μαυροπίνακα.



Εικόνα 3-V: Ψηφιδικό Διάγραμμα Μαυροπίνακα

Στην συνέχεια, αναλύονται τα παραπάνω δομοστοιχεία καθώς και η διαπροσωπία που εκθέτει καθένα για τις εσωτερικές λειτουργίες του Μαυροπίνακα.

3.3.3.1 Διαχειριστής Επικοινωνιών

Το δομοστοιχείο αυτό, αναλαμβάνει την επικοινωνία του Μαυροπίνακα με τα υπόλοιπα δομοστοιχεία του συστήματος λογισμικού που προτείνεται. Για να διεκπεραιώσει αυτόν τον στόχο

ορίζει τρεις διαπροσωπίες τις οποίες μπορούν να χρησιμοποιήσουν τα εκτός του Μαυροπίνακα δομοστοιχεία.

1. **Εγγραφή Για Γεγονότα** : Η διαπροσωπία αυτή (*registerForEvents*) επιτρέπει στα εξωτερικά δομοστοιχεία, να εκδηλώσουν ενδιαφέρον για κάποιο συγκεκριμένο τύπο ή μοτίβο γεγονότων. Το δομοστοιχείο που θα χρησιμοποιήσει αυτήν την διαπροσωπία, οφείλει να ορίσει το μοτίβο ή τον τύπο γεγονότων για το οποίο ενδιαφέρεται. Για την εξυπηρέτηση της συγκεκριμένης διαπροσωπίας, πρέπει να αποθηκεύεται στον Μαυροπίνακα ένας πίνακας κατακερματισμού (hashtable), με δυο εγγραφές, το δομοστοιχείο και το μοτίβο (ή τύπο) γεγονότων.

2. **Ενημέρωση** : Η διαπροσωπία αυτή, αναφέρεται στην περίπτωση εκείνη που αναγνωριστεί κάποιο μοτίβο ή τύπος γεγονότος για τον οποίο έχει εκδηλωθεί ενδιαφέρον από κάποιο δομοστοιχείο, μέσω κλήσης της προηγούμενης διαπροσωπίας (άρα θα υπάρχει και αντίστοιχη εγγραφή στο hashtable που αναφέραμε προηγουμένως). Η διαπροσωπία *notify* λοιπόν, καλείται όταν εμφανισθεί κάποιο μοτίβο ή τύπος γεγονότος και ενημερώνονται τα κατάλληλα δομοστοιχεία.

3. **Υποβολή Γεγονότων** : Η διαπροσωπία αυτή (*submitEvent*) καλείται από συγκεκριμένα δομοστοιχεία ώστε να υποβάλλουν νέα γεγονότα στο σύστημα με σκοπό να καταγραφούν στον Καταγραφέα Γεγονότων. Πιο συγκεκριμένα, η διαπροσωπία αυτή χρησιμοποιείται από την Υπηρεσία Ενεργοποίησης Στόχων ώστε να θέσει το νέο τρέχοντα στόχο που πρέπει να επαληθευτεί και από τον Διαχειριστή Συμβάντων Παρακολούθησης, όπου ζητά την καταγραφή νέων συμβάντων για το σύστημα, στο Μαυροπίνακα.

Η διαπροσωπία της Υποβολής Γεγονότων, συμβαίνει όταν κάποιο δομοστοιχείο επιθυμεί να καταγραφεί ένα νέο συμβάν στο Μαυροπίνακα. Έπεται ότι η καταγραφή αυτή πρέπει να γίνει σε κάποια δομή που θα αρχειοθετεί τα συμβάντα που έχουν παρουσιαστεί. Αυτά τα αναλαμβάνει το επόμενο δομοστοιχείο που θα αναλύσουμε ο Καταγραφέας Γεγονότων (Event Logger).

3.3.3.2 Καταγραφέας Γεγονότων

Το δομοστοιχείο αυτό, όπως έχει γραφεί, αναλαμβάνει την καταγραφή και αρχειοθέτηση των γεγονότων που έχουν παρουσιαστεί στο υπό έλεγχο σύστημα λογισμικού. Τα γεγονότα που καλείται να καταγράψει είναι δυο τύπων, αφενός νέοι τρέχοντες στόχοι που υποβάλλονται από την Υπηρεσία Ενεργοποίησης Στόχων, αφετέρου συμβάντα του υπό παρακολούθηση συστήματος.

Για την καταγραφή των δυο τύπων γεγονότων που αναφέραμε προηγουμένως, ο Καταγραφέας Γεγονότων, χρησιμοποιεί μια δομή *Στοίβας* στην οποία αποθηκεύονται τα γεγονότα. Η δομή αυτή, επιτρέπει την άμεση χρονολογική ταξινόμηση των γεγονότων, από τα πιο πρόσφατα στα πιο παλαιά, καθώς πάντα στην κεφαλή θα είναι το πιο πρόσφατα εμφανισθέν γεγονός. Στην *Στοίβα* που αναφέρεται, αποθηκεύονται ανά θέση, δυο πράγματα, με το πρώτο να είναι το συμβάν αυτό καθεαυτό και το δεύτερο μια χρονοσφραγίδα, της στιγμής κατά την οποία καταγράφηκε το γεγονός.

Ο Καταγραφέας Γεγονότων, εκθέτει στα υπόλοιπα δομοστοιχεία του Μαυροπίνακα, τις εξής δυο διαπροσωπίες:

1. **Καταγραφή Γεγονότος (*logEvent*)** : Η διαπροσωπία αυτή, χρησιμοποιείται από τον Διαχειριστή Επικοινωνιών. Μέσω της διαπροσωπίας αυτής, προωθείται από κάποιο ενδιαφερόμενο δομοστοιχείο ένα νέο γεγονός (είτε νέος στόχος προς επαλήθευση, είτε συμβάν του υπό έλεγχο συστήματος) ώστε να καταγραφεί στην *Στοίβα* του Καταγραφέα Γεγονότος. Η προώθηση αυτή, γίνεται με την διαμεσολάβηση του Διαχειριστή Επικοινωνιών και πιο συγκεκριμένα στην Καταγραφή Γεγονότος προωθούνται αιτήσεις στην διαπροσωπία Υποβολή Γεγονότος του Διαχειριστή Επικοινωνιών. Με την κλήση αυτής της διαπροσωπίας, ο Καταγραφέας Γεγονότων υποχρεούται να σημειώσει την χρονοσφραγίδα της άφιξης της αίτησης καταγραφής του γεγονότος και να την αποθηκεύσει σε νέα θέση της Στοίβας, μαζί με το περιεχόμενο του γεγονότος που αφίχθη. Επίσης, υποχρεούται να ενημερώσει τον Ελεγκτή Μοτίβων Γεγονότων, για το νέο γεγονός, ώστε να ελεγχθεί αν πρέπει να ειδοποιηθεί κάποιο δομοστοιχείο για το νεοεμφανισθέν γεγονός.

2. **Τροφοδότηση Γεγονότων (*feedEvents*)** : Η διαπροσωπία αυτή αποτελεί τον συνδετικό κρίκο μεταξύ των δομοστοιχείων Καταγραφέας Γεγονότων και Ελεγκτής Μοτίβων Γεγονότων. Χρησιμοποιείται από το τελευταίο δομοστοιχείο, στην περίπτωση που αυτό είναι στην διαδικασία ελέγχου της παρουσίας κάποιου μοτίβου. Με την διαπροσωπία αυτή, ζητείται μια σειρά από γεγονότα που συνέβησαν σε ένα χρονικό πλαίσιο (time frame) και ο Καταγραφέας Γεγονότων, οφείλει να αναζητήσει για τα γεγονότα του συγκεκριμένου χρονικού πλαισίου στην *Στοίβα* του. Επιστρέφει στο αιτών δομοστοιχείο (Ελεγκτής Μοτίβων Γεγονότων) το αποτέλεσμα της αναζήτησης αυτής. Η διαπροσωπία αυτή, μπορεί να ενεργοποιηθεί και στην παρουσία κάποιου νέου γεγονότος, χωρίς την αίτηση του Ελεγκτή Μοτίβων Γεγονότων. Κατά την καταγραφή ενός νέου συμβάντος, το δομοστοιχείο Καταγραφέας Γεγονότων, οφείλει να ενημερώσει άμεσα τον Ελεγκτή Μοτίβων Γεγονότων, μέσω της διαπροσωπίας Τροφοδότησης Γεγονότων, για το νέο γεγονός που εμφανίσθηκε, ώστε να ενταχθεί στους αλγορίθμους αναγνώρισης μοτίβων. Με λίγα λόγια, η Τροφοδότηση

Γεγονότων, ενεργοποιείται και έπειτα από αίτηση του Ελεγκτή Μοτίβων Γεγονότων και "αυτεπάγγελτα" κατά την εμφάνιση και αίτηση καταγραφής κάποιου νέου γεγονότος.

Ο Καταγραφέας Γεγονότων επιτελεί σημαντικές λειτουργίες όχι μόνο για το Μαυροπίνακα, αλλά και για το σύνολο της αρχιτεκτονικής του περιβάλλοντος-πλαισίου. Στην *Στοίβα* του, όπως έχει ήδη αναφερθεί, αποθηκεύεται στην ουσία μια χρονική καταγραφή της λειτουργίας του υπό έλεγχο συστήματος λογισμικού, αλλά και των στόχων που έχουν τεθεί προς επαλήθευση από το περιβάλλον-πλαίσιο.

Οι στόχοι που καταγράφονται, απευθύνονται στο δομοστοιχείο Διαχειριστής Πολιτικών, το οποίο ενημερώνεται κάθε φορά για το νέο στόχο προς επαλήθευση. Τα συμβάντα για το υπό έλεγχο σύστημα λογισμικού, μπορεί να ενδιαφέρουν οποιοδήποτε δομοστοιχείο έχει εκδηλώσει ενδιαφέρον, είτε για κάποιο συγκεκριμένο τύπο τους είτε για κάποιο συγκεκριμένο μοτίβο τους. Η αναγνώριση του μοτίβου ή τύπου γεγονότος που ενδιαφέρουν τα δομοστοιχεία του περιβάλλοντος-πλαισίου, γίνεται στο επόμενο δομοστοιχείο του Μαυροπίνακα που παρουσιάζεται, τον Ελεγκτή Μοτίβων Γεγονότων.

3.3.3.3 Ελεγκτής Μοτίβων Γεγονότων

Το δομοστοιχείο αυτό (*EventPatternMatcher*), αναλαμβάνει την αναγνώριση μοτίβων και τύπων γεγονότων τα οποία ενδιαφέρουν τα δομοστοιχεία του περιβάλλοντος-πλαισίου. Η διαδικασία αυτή είναι η καρδιά της φιλοσοφίας της Έμμεσης Κλήσης, φιλοσοφία στην οποία, όπως έχει αναφερθεί, βασίζεται η αρχιτεκτονική του περιβάλλοντος-πλαισίου που παρουσιάζεται.

Η διαδικασία που αναφέρεται, ξεκινά με την εγγραφή των δομοστοιχείων σε μοτίβα ή τύπους γεγονότων χρησιμοποιώντας την διαπροσωπία Εγγραφή Για Γεγονότα, στο δομοστοιχείο Διαχειριστή Επικοινωνιών, όπως αναφέρεται και στην παράγραφο 3.3.3.1 . Δεύτερο βήμα, είναι η αναγνώριση του μοτίβου ή του τύπου γεγονότος από τον Ελεγκτή Μοτίβων Γεγονότων. Τέλος, στην αναγνώριση κάποιου μοτίβου ή τύπου γεγονότος, πρέπει να ενημερωθεί το κατάλληλο δομοστοιχείο, μέσω της διαπροσωπίας *notify* του δομοστοιχείου Διαχειριστής Επικοινωνιών, όπως αναφέρθηκε στην παράγραφο 3.3.3.1 .

Η μετάβαση από το δεύτερο βήμα, στην ενημέρωση του κατάλληλου δομοστοιχείου, γίνεται μέσω της διαπροσωπίας *matchEvent* του δομοστοιχείου Ελεγκτή Μοτίβων Γεγονότων. Με την διαπροσωπία αυτή, εκτίθενται από το δομοστοιχείο Ελεγκτής Μοτίβων Γεγονότων ο τύπος ή το μοτίβο γεγονότων που έχει παρουσιαστεί, κατά την λειτουργία του υπό έλεγχο συστήματος λογισμικού. Η διαπροσωπία αυτή, χρησιμοποιείται από το δομοστοιχείο Διαχειριστής Επικοινωνιών, ούτως ώστε να εντοπισθούν ποια δομοστοιχεία πρέπει να ενημερωθούν για τους

αναγνωρισθέντες τύπους ή μοτίβα γεγονότων. Υπενθυμίζεται, όπως αναφέρεται και στην παράγραφο 3.3.3.1, η αντιστοίχιση μεταξύ των αναγνωρισθέντων γεγονότων και του ενδιαφερόμενου δομοστοιχείου, γίνεται μέσω αναζήτησης στον πίνακα κατακερματισμού του δομοστοιχείου Διαχειριστής Επικοινωνιών.

3.3.4 Υπηρεσία Ενεργοποίησης Στόχων

Το περιβάλλον-πλαίσιο που παρουσιάζεται, εντάσσεται στην κατηγορία των Αυτόνομων Συστημάτων, δηλαδή επιδεικνύει μια νοημοσύνη, μια προσαρμοστικότητα στις αλλαγές κατάστασης και του ίδιου, αλλά κυρίως του υπό ελέγχου συστήματος λογισμικού. Για να είναι σε θέση να δείξει αυτήν την προσαρμοστικότητα, όπως έχει παρουσιασθεί, υιοθετεί μια φιλοσοφία λειτουργίας, οδηγούμενη από στόχους. Σε κάθε χρονική στιγμή, υπάρχει ένας (ή περισσότεροι) κύριος στόχος που πρέπει να επαληθευτεί και ο οποίος αποτελεί την συνθήκη για την επαλήθευση της ομαλής λειτουργίας του υπό έλεγχο συστήματος λογισμικού. Αυτός ο κύριος στόχος, αναλύεται περαιτέρω σε υπό στόχους και βρίσκεται ένα σύνολο από τρέχοντες στόχους που πρέπει να επαληθευτούν προκειμένου να επαληθευτεί ο κύριος στόχος και συνεπώς η ομαλή λειτουργία του υπό έλεγχο συστήματος λογισμικού. Αυτή η διαδικασία, αναλύθηκε στην παράγραφο 3.3.2, όπου αναλύθηκε η λειτουργία του δομοστοιχείου Διαχειριστής Πολιτικών.

Στην παρούσα παράγραφο αναλύεται η μεθοδολογία μέσω της οποίας εντοπίζεται αυτός ο κύριος στόχος, ποια δεδομένα χρησιμοποιούνται στον εντοπισμό του, αλλά και πώς ενημερώνεται ο Διαχειριστής Πολιτικών για τον εκάστοτε κύριο στόχο. Αυτές τις λειτουργίες επωμίζεται το δομοστοιχείο που λέγεται Υπηρεσία Ενεργοποίησης Στόχων, που είναι το αντικείμενο μελέτης της παρούσας παραγράφου.

Όπως όλα τα δομοστοιχεία του περιβάλλοντος-πλαισίου, το δομοστοιχείο Υπηρεσία Ενεργοποίησης Στόχων λαμβάνει τις απαραίτητες πληροφορίες για την κατάσταση του υπό έλεγχο συστήματος λογισμικού, από το Μαυροπίνακα, όπως και στον Μαυροπίνακα εκθέτει τον κύριο στόχο προς επαλήθευση κάθε φορά. Σύμφωνα με την λειτουργία του Μαυροπίνακα, που αναλύθηκε στην παράγραφο 3.3.3, η Υπηρεσία Ενεργοποίησης Στόχων, πρέπει αρχικά να εγγραφεί για ενημέρωση σε συγκεκριμένα μοτίβα γεγονότων που αφορούν στο υπό έλεγχο σύστημα λογισμικού. Όπως είναι λογικό, η Υπηρεσία Ενεργοποίησης Στόχων, ενδιαφέρεται για συμβάντα του επηρεάζουν την δομή του Δέντρου Στόχων του υπό έλεγχο συστήματος λογισμικού. Τέτοιου είδους συμβάντα είναι, για παράδειγμα, η αλλαγή ενός δομοστοιχείου με κάποιο άλλο που φιλοδοξεί να παρέχει την ίδια λειτουργικότητα, ή η αφαίρεση κάποιου δομοστοιχείου από το υπό έλεγχο σύστημα λογισμικού, λόγω βλάβης. Τέτοιου είδους συμβάντα, δημιουργούν σημαντικές δομικές και λειτουργικές αλλαγές

στο υπό έλεγχο σύστημα λογισμικού και προκαλούν την αλλαγή του τρέχοντα στόχου, από την Υπηρεσία Ενεργοποίησης Στόχων.

Η εγγραφή της Υπηρεσίας Ενεργοποίησης Στόχων για κάποιο δεδομένο μοτίβο γεγονότων στο Μαυροπίνακα, αποτελεί το πρώτο βήμα διεκπεραίωσης της λειτουργικότητας που προσφέρει στο περιβάλλον-πλαίσιο. Επόμενο βήμα, είναι η διαδικασία αναγνώρισης του νέου κύριου στόχου, με βάση το ιστορικό καταγραφής των συμβάντων του υπό έλεγχο συστήματος λογισμικού.

Στην διαδικασία εντοπισμού του νέου κύριου στόχου, έγκειται κατά κύριο λόγω η προσαρμοστικότητα του περιβάλλοντος-πλαισίου. Ο αλγόριθμος που χρησιμοποιείται, έχει ως βάση του το Μοντέλο Δέντρο Στόχων του υπό έλεγχο συστήματος λογισμικού. Αρχικά, εντοπίζεται με βάση το περιεχόμενο του συμβάντος, σε ποιο τμήμα του Δέντρου Στόχων επήλθε αλλαγή. Έπειτα, ανάλογα την περίπτωση τίθεται ως νέος στόχος συνήθως ο στόχος που άλλαξε (ή ο στόχος – πατέρας αυτού) και σε ορισμένες περιπτώσεις στόχοι οι οποίοι σχετίζονται με αυτόν. Για παράδειγμα, αν τα συμβάντα του υπό έλεγχο συστήματος, δείχνουν ότι αντικαθίσταται κάποιο δομοστοιχείο το οποίο αντιστοιχεί και σε κάποιον συγκεκριμένο στόχο του Δέντρου Στόχων, τότε νέος κύριος στόχος προς επαλήθευση, γίνεται ο στόχος που απορρέει από το δομοστοιχείο.

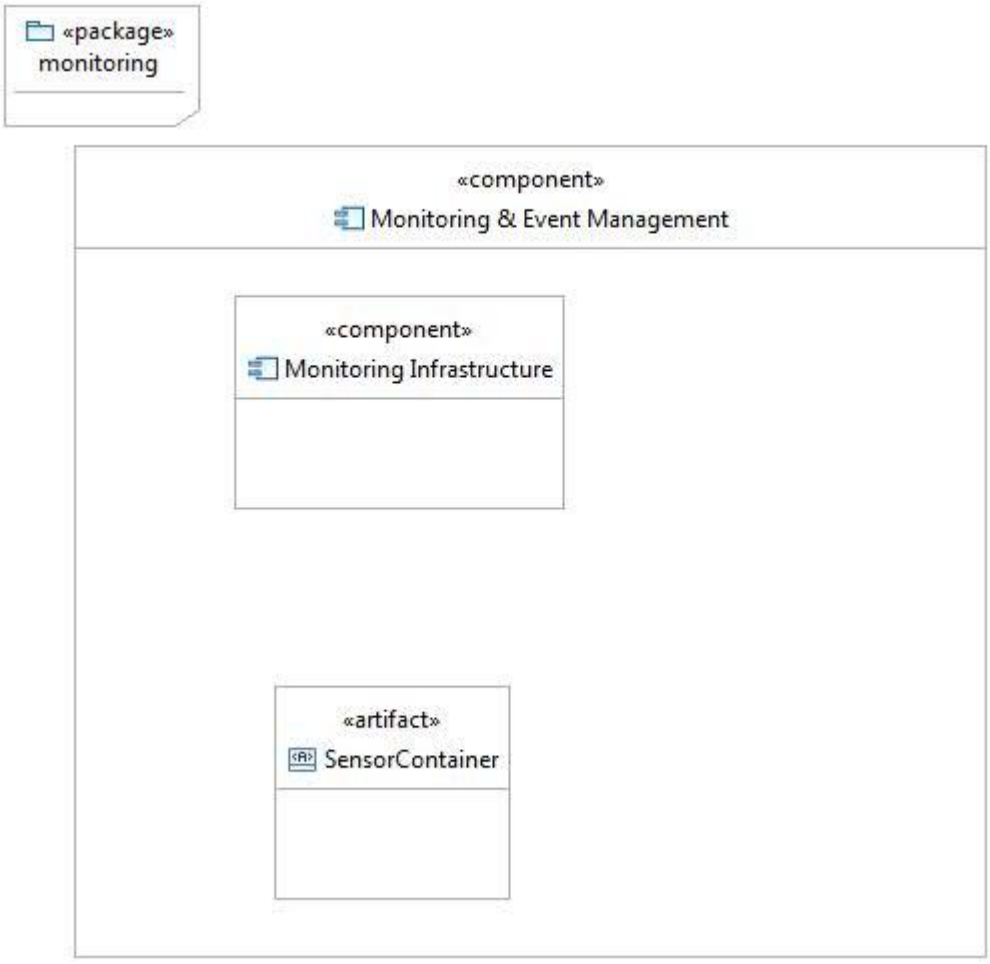
Αξίζει στο σημείο αυτό να τονιστεί, ότι ο χρήστης του περιβάλλοντος-πλαισίου, έχει την δυνατότητα να ορίσει όπως τους αλγορίθμους αναγνώρισης του τρέχοντος κύριου στόχου, κατά το δοκούν. Έτσι δίνεται η ελευθερία στο περιβάλλον-πλαίσιο, να μπορεί να λειτουργήσει και σε διαφορετικής αρχιτεκτονικής συστήματα λογισμικού, από τα υπηρεσιοκεντρικά συστήματα.

3.3.5 Συλλέκτης Συμβάντων Παρακολούθησης

Η ανάλυση που αφορούσε στα προηγούμενα δομοστοιχεία του περιβάλλοντος-πλαισίου, κατέδειξε την εσωτερική λειτουργία του. Στα περισσότερα σημεία, η λειτουργία αυτή καθορίζεται σε μεγάλο βαθμό, από πληροφορία που υπάρχει ή συλλέγεται από το υπό έλεγχο σύστημα λογισμικού. Η πληροφορία που δεν συλλέγεται, είναι κυρίως το Δέντρο Στόχων, το οποίο αναφέραμε πώς παράγεται στην παράγραφο 3.3.1 και η δομή του θα παρουσιαστεί στην συνέχεια. Τα δομοστοιχείο που παρουσιάζεται στην παράγραφο αυτή, επωμίζεται το ρόλο της συλλογής της απαραίτητης πληροφορίας από το υπό έλεγχο σύστημα.

Το δομοστοιχείο Συλλέκτης Συμβάντων Παρακολούθησης, είναι ο συνδετικός κρίκος του περιβάλλοντος-πλαισίου, με το υπό έλεγχο σύστημα λογισμικού. Η δομή του και η λειτουργία του πρέπει να συμβάλλουν στην παρακολούθηση του υπό έλεγχο συστήματος λογισμικού, στην επεξεργασία της πληροφορίας αυτής και στην καταγραφή των σημαντικών συμβάντων στο υπό

έλεγχο σύστημα λογισμικού (με τον τρόπο που περιγράφηκε στην παράγραφο 3.3.3.2). Η εσωτερική δομή του Συλλέκτη Συμβάντων Παρακολούθησης φαίνεται στο παρακάτω ψηφιδικό διάγραμμα.



Εικόνα 3-VI: Ψηφιδικό Διάγραμμα Συλλέκτη Συμβάντων Παρακολούθησης

Το σημαντικότερο δομοστοιχείο είναι η Υποδομή Παρακολούθησης. Το δομοστοιχείο αυτό, λαμβάνει εντολές από τον Διαχειριστή Πολιτικών και είναι υπεύθυνο για την ενεργοποίηση κατάλληλων Δοχείων Αισθητήρων, για την συλλογή στοιχείων.

Πιο συγκεκριμένα, τα Δοχεία Αισθητήρων (SensorContainer), αποτελούν τεχνήματα (artifacts) τα οποία περιλαμβάνουν διάφορους Αισθητήρες, άλλα τεχνήματα δηλαδή, τα οποία χρησιμοποιούνται για την παρακολούθηση του υπό έλεγχο συστήματος. Οι Αισθητήρες (*Sensors*) έχουν τον τρόπο να λάβουν τις απαραίτητες πληροφορίες για το υπό έλεγχο σύστημα. Πιο συγκεκριμένα, οι τρόποι με τους οποίους μπορεί να ληφθούν πληροφορίες από το υπό έλεγχο σύστημα είναι οι εξής:

- «Πιάσιμο» των εξαιρέσεων, αν πρόκειται για λογισμικό γραμμένο σε κάποια γλώσσα που υποστηρίζει εξαιρέσεις (exceptions).

- Έλεγχος της εξόδου του λογισμικού ή των ημερολογίων (logs) που γράφονται κατά την εκτέλεσή του, και επεξεργασία αυτών για την εξαγωγή συμβάντων.

- Μετρικές πόρων που μπορεί να χρησιμοποιεί το υπό έλεγχο σύστημα λογισμικού, όπως μέγεθος της RAM που χρησιμοποιείται, χρήση του δικτύου, χρήση του επεξεργαστή ενός Υπολογιστικού Συστήματος και άλλα.

- Ανάγνωση και επεξεργασία των αρχείων ρυθμίσεων ή αρχείων εγκαταστατών για το υπό έλεγχο σύστημα λογισμικού και εξαγωγή πληροφοριών από αυτά.

Με βάση τις παραπάνω πηγές συμβάντων για το υπό έλεγχο σύστημα, υπάρχουν και αντίστοιχοι τύποι Αισθητήρων. Οι Αισθητήρες οργανώνονται σε Δοχεία Αισθητήρων οι οποίοι ενεργοποιούνται κατά το δοκούν, από την Υποδομή Παρακολούθησης. Για παράδειγμα, αν ο κύριος στόχος είναι κάποιος στόχος που αφορά σε κάποιο Service Level Agreement, δηλαδή έναν αριθμό επίδοσης του συστήματος λογισμικού, τότε θα ενεργοποιηθεί από την Υποδομή Παρακολούθησης το Δοχείο Αισθητήρων που περιέχει τους Αισθητήρες που λαμβάνουν μετρικές από το υπό έλεγχο σύστημα. Έτσι, θα μπορεί να συλλεχθεί η πληροφορία που χρειάζεται το περιβάλλον-πλαίσιο, για την επαλήθευση ή όχι του Service Level Agreement που έχει τεθεί στον κύριο τρέχοντα στόχο.

Κεφάλαιο 4: Μοντέλο Δέντρου Στόχων

4.1 Εισαγωγή

Στο προηγούμενο κεφάλαιο, παρουσιάσθηκε η αρχιτεκτονική του περιβάλλοντος-πλαισίου που είναι ο σκοπός της παρούσας διπλωματικής. Όπως φάνηκε, ένα σημαντικό μέρος της ανάλυσης περιείχε ένα μοντέλο Δέντρων Στόχων το οποίο χρησιμοποιείται κατά κόρον από το περιβάλλον-πλαίσιο ως μια αναπαράσταση της μοντελοποίησης των λειτουργικών απαιτήσεων του υπό έλεγχο συστήματος λογισμικού.

Το μοντέλο Δέντρου Στόχων που παρουσιάζεται στο κεφάλαιο αυτό, δανείζεται την δομή και αρκετά χαρακτηριστικά από το Δέντρα Στόχων που παρουσιάστηκαν για πρώτη φορά στο [2]. Στο άρθρο αυτό, παρουσιάσθηκε μια αλλαγή νοοτροπίας στην μοντελοποίηση των απαιτήσεων των συστημάτων λογισμικού, από ένα αντικειμενοστραφές μοντέλο προς ένα προσανατολισμένο σε στόχους μοντέλου.

Η ιδέα της προσανατολισμένης σε στόχους προσέγγισης, έγκειται στην μοντελοποίηση των απαιτήσεων του συστήματος λογισμικού σε στόχους και την ανάλυσή τους σε υποστόχους. Κάθε στόχος μπορεί να εξαρτάται από τους υποστόχους του είτε συμπληρωματικά, είτε από όλους ανεξαιρέτως, οπότε έχουμε την AND-OR Decomposition. Αυτά, αναφέρονται στο [2] και στο κεφάλαιο 2.

Για τις ανάγκες του περιβάλλοντος-πλαισίου που παρουσιάζουμε, εμπλουτίζουμε το μοντέλο δέντρου στόχων με επισημειώσεις, που βοηθούν στην επισύναψη επιπλέον πληροφορίας για κάθε στόχο, πληροφορίας η οποία χρειάζεται από το περιβάλλον-πλαίσιο ώστε να ενεργοποιήσει τις κατάλληλες πολιτικές και να προβεί στις απαραίτητες ενέργειες.

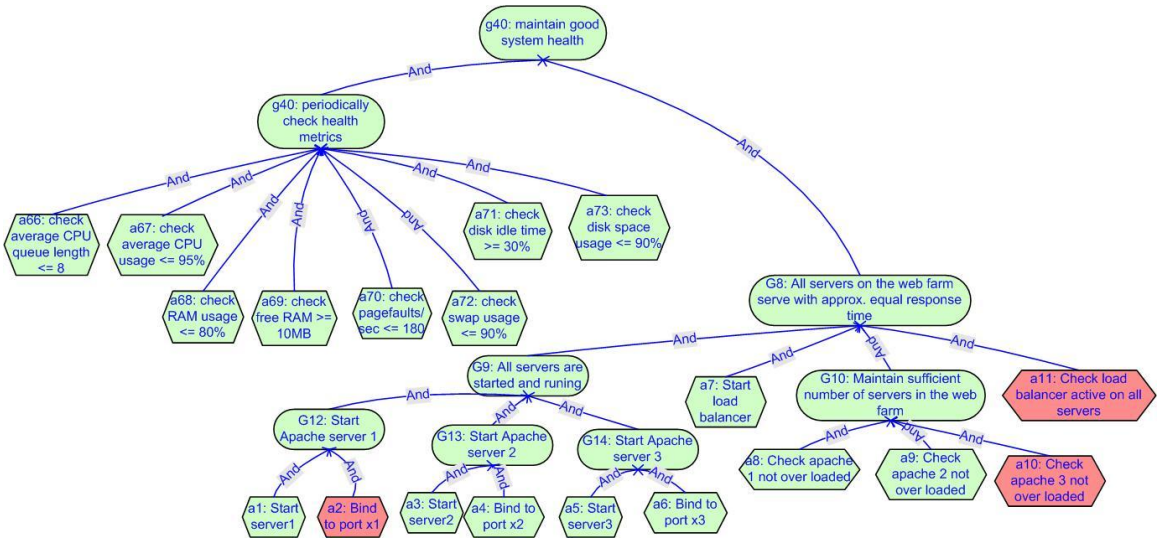
Τα παραπάνω δυο σημεία του δέντρου στόχου που προτείνουμε, παρουσιάζονται αναλυτικά στο κεφάλαιο αυτό. Ξεκινάμε με την γενική δομή που θέλουμε αυτό να έχει και καταλήγουμε στο τέλος του κεφαλαίου, παρουσιάζοντας λεπτομερώς τις επισημειώσεις που μπορεί αυτό να δέχεται.

4.2 Γενική Επισκόπηση του Μοντέλου Στόχων - Δομή

Βασική έννοια στο μοντέλο που θα παρουσιασθεί, είναι ο στόχος. Ως στόχο, στο άρθρο [2] αναφέρεται ως μια λειτουργική απαίτηση του συστήματος λογισμικού. Με τον ορισμό αυτό, ένας στόχος αναλύεται σε συγκεκριμένους υποστόχους μέσω δυο πιθανών συσχετίσεων. Της συσχέτισης ΚΑΙ (AND) η οποία δηλώνει ότι για να επαληθευτεί ο στόχος, πρέπει να επαληθευτούν **όλοι** οι υποστόχοι που τον απαρτίζουν. Επίσης της συσχέτισης Ή (OR) η οποία δηλώνει ότι για να επαληθευτεί ο στόχος, πρέπει να επαληθευτεί **τουλάχιστο ένας** από τους υποστόχους του. Τέτοιου είδους στόχοι αναφέρονται ως **hard goals**, όπως αναφέρονται στο [2].

Υπάρχουν όμως και περιπτώσεις, όπου ένας στόχος που αναφέρεται σε μια λειτουργική απαίτηση ενός συστήματος λογισμικού, να μην μπορεί να αναλυθεί σε ένα αυστηρό υποσύνολο AND ή OR υποστόχων. Αντιθέτως, υπάρχει η δυνατότητα να ελεγχθεί η επαλήθευση ενός στόχου, με ένα σύστημα ψηφοφορίας (voting) δηλαδή επαλήθευση ανάλογα με το αν επαληθεύονται περισσότεροι στόχοι που συνεισφέρουν θετικά στην επαλήθευση του στόχου ή όχι. Στόχοι που επιβάλλουν τέτοια μεθοδολογία για την επαλήθευσή τους λέγονται **soft goals** ([2]).

Η μορφολογία του δέντρου στόχων δεν αλλάζει είτε πρόκειται για hard goals είτε για soft goals. Σε κάθε περίπτωση η δομή του δέντρου παραμένει. Ένα παράδειγμα δέντρου στόχων, δίνεται στο παρακάτω σχήμα. Το σχήμα αυτό, ακολουθεί το πρότυπο του δέντρου στόχων, όπως ορίσθηκε στο [2].

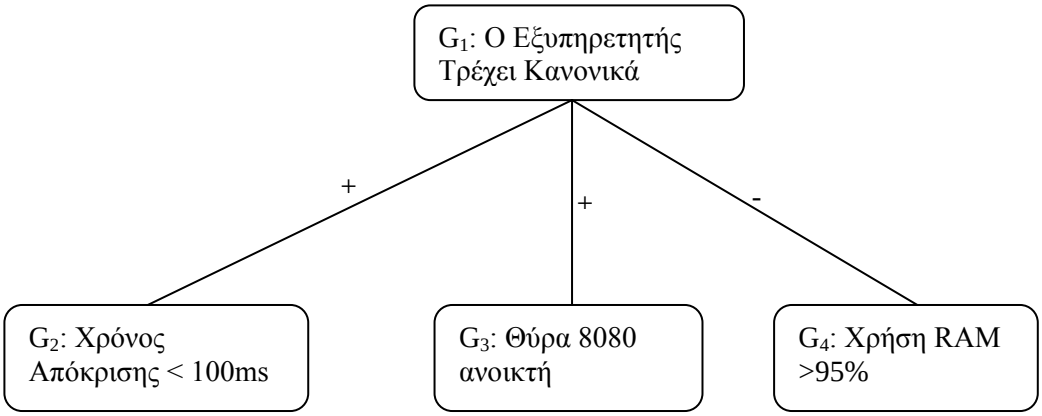


Εικόνα 4-Ι: Παράδειγμα Δέντρου Στόχων (hard goals)

Το προηγούμενο παράδειγμα δείχνει πώς γίνεται η ανάλυση ενός στόχου σε υποστόχους. Κύριος στόχος στο παραπάνω παράδειγμα είναι η διατήρηση της καλής λειτουργίας του συστήματος. Ο στόχος αυτός, αναλύεται στους στόχους του περιοδικού ελέγχου μετρικών καλής λειτουργίας και

στον στόχο όλοι οι εξυπηρετητές του συστήματος να έχουν παρόμοιους χρόνους απόκρισης. Αυτό σημαίνει ότι για να επαληθευτεί ο στόχος της διατήρησης της καλής λειτουργίας του συστήματος πρέπει να επαληθεύεται και ο ένας υποστόχος και ο δεύτερος. Αντίστοιχες αναλύσεις έχουμε στους υποστόχους του κυρίως στόχου και με τον τρόπο αυτό δομείται το δέντρο στόχων με βάση τις απαιτήσεις προς επαλήθευση κάθε στόχου, είτε του κύριου είτε ενδιάμεσων.

Το παραπάνω παράδειγμα, αναφέρεται στην περίπτωση που το υπό έλεγχο σύστημα λογισμικού, μπορεί εξ' ολοκλήρου να μοντελοποιηθεί με hard goals. Αυτό, δεν είναι πάντα εφικτό, καθώς έχει αναφερθεί ότι υπάρχει η περίπτωση ο στόχος να μην αναλύεται σε συγκεκριμένο, αυστηρό σύνολο υποστόχων, αλλά σε ένα σύνολο υποστόχων που καθένας συμμετέχει στην επαλήθευση του στόχου, δεν την καθορίζει σχεδόν εξ' ολοκλήρου. Αυτή η περίπτωση των soft goals παρουσιάζεται στην παρακάτω εικόνα, που δείχνει ένα τέτοιο δέντρο στόχων.



Εικόνα 4-ΙΙ: Παράδειγμα Δέντρου Στόχων (soft goals)

Γίνεται φανερό, ότι σε αυτήν την περίπτωση, δεν μπορούμε να αποφανθούμε για την επαλήθευση του κύριου στόχου (G1), απλά και μόνο από την επαλήθευση ή μη κάποιου από τους στόχους $G_2 \ldots G_4$. Αντιθέτως, για την επαλήθευση του στόχου G_1, βλέπουμε τη συνισταμένη της συνεισφοράς των soft goals, που είτε συνεισφέρουν θετικά (σημειώνεται με ένα '+' στην εικόνα), είτε αρνητικά (σημειώνεται με ένα '-' στην εικόνα). Συνεπώς, αν ο χρόνος απόκρισης του εξυπηρετητή είναι μικρότερος των 100 ms ή η θύρα 8080 του, φανεί ανοικτή (δέχεται συνδέσεις), τότε έχουμε θετική συνεισφορά στην επαλήθευσή του. Αν βρεθεί ότι ο εξυπηρετητής χρησιμοποιεί πέραν του 95% της

διαθέσιμης RAM στο υπολογιστικό σύστημα που τρέχει, τότε έχουμε μια αρνητική συνεισφορά στην επαλήθευση του στόχου G_1. Ή αν είναι προτιμητέο, η επαλήθευση του στόχου G4, παρέχει μια θετική συνεισφορά στην **μη επαλήθευση** του κυρίως στόχου G**1**.

Σε κάθε περίπτωση, μετρώνται οι θετικές και οι αρνητικές συνεισφορές στην επαλήθευση του κυρίως στόχου και αναλόγως λαμβάνεται η απόφαση αν ο κύριος στόχος επαληθεύεται ή όχι.

Το περιβάλλον-πλαίσιο που παρουσιάζεται, όπως φάνηκε και στο Κεφάλαιο 3, δεν θέτει περιορισμούς για τους τύπους στόχων (hard ή soft) θα περιέχει το μοντέλο Δέντρου Στόχων με το οποίο μοντελοποιείται το υπό έλεγχο σύστημα λογισμικού, καθώς δεν διαχωρίζει τους δυο αυτούς τύπους. Εντούτοις, όπως έγινε φανερό στο Κεφάλαιο 3 λόγω της δομής που αναμένεται για το δέντρο στόχων, αναμένει να υπάρχουν μόνο hard goals στο μοντέλο Δέντρου Στόχων που θα μοντελοποιηθεί στον Μοντελοποιητή Στόχων. Αναμένεται δηλαδή, στο μεγαλύτερο βαθμό, η ύπαρξη σαφούς ανάλυσης σε υποστόχους που συνδέονται με AND ή OR, ώστε να είναι δυνατόν το περιβάλλον-πλαίσιο να εφαρμόσει, ανά περίπτωση, τις κατάλληλες πολιτικές.

Η έννοια των soft goals, στο μοντέλο δέντρου στόχων που παρουσιάζουμε, κρύβεται σε κάτι το οποίο δεν υπήρχε στο αρχικό μοντέλο που εισήχθη στο [2]. Η έννοια των soft goals, υφίσταται στις σχέσεις που μπορεί να έχουν οι στόχοι μεταξύ τους και οι οποίοι μοντελοποιούνται από τον Επεξεργαστή Σχέσεων, όπως έχει αναλυθεί στην παράγραφο 3.3.1.3. Οι σχέσεις που μπορεί να υπάρχουν μεταξύ στόχων στο Δέντρο Στόχων, είναι, μεταξύ άλλων, σχέσεις Ανταγωνισμού (Conflicts) και Εξάρτησης (Depends) οι οποίες δηλώνουν αφενός η πρώτη μια αρνητική συνεισφορά στην επαλήθευση του στόχου, αφετέρου η δεύτερη μια θετική συνεισφορά στην επαλήθευση του στόχου. Αξίζει εδώ να σημειωθεί, ότι σε αυτήν την περίπτωση δεν έχουμε μεθοδολογία επαλήθευσης του στόχου, με βάση την αντίστοιχη περίπτωση όπως αναφέρεται στο [2] (μέσω ψηφοφορίας θετικών και αρνητικών συνεισφορών). Στο μοντέλο Δέντρου Στόχων που παρουσιάζουμε, οι σχέσεις μεταξύ στόχων χρησιμοποιούνται ως αρωγός στον Διαχειριστή Πολιτικών ώστε να βρει το σύνολο VS, έχοντας τον τρέχοντα στόχο (βλ. παράγραφο 3.3.2).

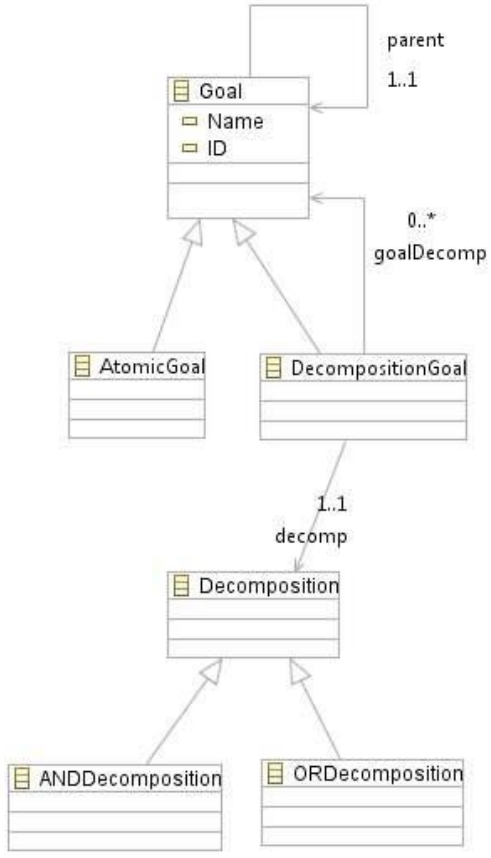
4.3 AND/OR Αποσυνθέσεις Στόχων

Όπως αναφέρθηκε, το περιβάλλον-πλαίσιο αναμένει δομή μοντέλου Δέντρου Στόχων, που ομοιάζει περισσότερο στην Εικόνα 4-I παρά στην Εικόνα 4-II. Δηλαδή, ομοιάζει περισσότερο με ένα Δέντρο Στόχων με hard goals παρά με soft goals. Αυτό είναι αναμενόμενο, καθώς είναι επιθυμητό να υπάρχει αποσύνθεση των στόχων σε συγκεκριμένο και αυστηρό σύνολο υποστόχων που η επαλήθευσή ενός εξ' αυτών (περίπτωση OR) ή όλων εξ' αυτών (περίπτωση AND) να μπορεί να

επαληθεύσει ή όχι τον κυρίως στόχο. Διασφαλίζεται, με αυτόν τον τρόπο, η αξιοπιστία του περιβάλλοντος-πλαισίου και η εμπιστοσύνη που εμπνέει στους χρήστες του.

Η σημασιολογία των αποσυνθέσεων στόχων είναι προφανής. Στους στόχους που αναλύονται με υποστόχους μέσω AND, η επαλήθευσή τους έγκειται στην επαλήθευση όλων των υποστόχων τους. Στους στόχους που αναλύονται σε υποστόχους μέσω OR, η επαλήθευσή τους έγκειται στην επαλήθευση τουλάχιστον ενός, από τους υποστόχους τους. Η σημασιολογία αυτή, οφείλει να είναι ο οδηγός οποιουδήποτε χρήστη επιθυμεί να ορίσει τους δικούς του μετασχηματισμούς για το Μοντελοποιητή Στόχων, όπως αυτοί παρουσιάσθηκαν στην παράγραφο 3.3.1.

Στην συνέχεια του κεφαλαίου, θα δοθεί τμήμα προς τμήμα το μοντέλο πεδίου (domain model) του προτεινόμενου μοντέλου δέντρου στόχων καθώς θα εξηγείται αναλυτικά σε όλα του τα χαρακτηριστικά, δομικά και λειτουργικά. Όπως γίνεται φανερό και από τον τίτλο της παραγράφου αυτής, πρώτο τμήμα του μοντέλου δέντρου στόχων, παρατίθεται αυτό που αφορά στις αποσυνθέσεις των στόχων.



Εικόνα 4-ΙΙΙ: Μοντέλο Πεδίου Μοντέλου Δέντρου Στόχων (1 από 4)

Όπως φαίνεται και στην Εικόνα 4-ΙΙΙ, οι στόχοι στο μοντέλο δέντρου στόχου που παρουσιάζεται, έχουν ως ιδιότητες ένα όνομα **Name** (ένα αλφαριθμητικό αναγνωριστικό του στόχου) και έναν μοναδικό αριθμό **ID**, ο οποίος λειτουργεί ως κλειδί με το οποίο μπορεί να αναγνωριστεί ένας στόχος από τους υπόλοιπους που μπορεί να μοιράζονται κοινό όνομα μαζί του. Αυτό γίνεται, γιατί δεν μπορεί να υπάρξει εγγύηση ότι κάθε στόχος θα έχει διαφορετικό όνομα στο ίδιο δέντρο στόχων. Επίσης, δεν μπορεί να υπάρξει εγγύηση ότι ο μοναδικός αριθμός **ID**, είναι πράγματι μοναδικός για στόχους δυο ή περισσότερων Δέντρων Στόχων, που χρησιμοποιούνται, όμως, στο ίδιο στιγμιότυπο του περιβάλλοντος-πλαισίου. Στην υλοποίηση του Δέντρου Στόχων που παρουσιάζεται, ο μοναδικός αριθμός **ID** θεωρείται καθολικώς παραγόμενος για το στιγμιότυπο του περιβάλλοντος-πλαισίου. Αυτό, οφείλει να το σεβαστεί και οποιοσδήποτε χρήστης θέλει να ορίσει με τον δικό του τρόπο τον Μοντελοποιητή Στόχων και να παράγει με δικό του αλγόριθμο αριθμούς **ID.**

Στην Εικόνα 4-ΙΙΙ, επισημαίνεται ακριβώς ο τρόπος με τον οποίο παρουσιάζεται στο μοντέλο δέντρου στόχων, η ανάλυση σε AND/OR αποσυνθέσεις, κάποιου στόχου. Όπως φαίνεται, ένας στόχος είναι μια διαπροσωπία (interface) η οποία υλοποιείται από δυο κλάσεις, την κλάση ατομικός στόχος (AtomicGoal) και την κλάση σύνθετος στόχος (DecompositionGoal). Η πρώτη κλάση δεν περιλαμβάνει επιπλέον πληροφορία, παρά μόνο αυτή που κληρονομεί από τη διαπροσωπία Goal (Όνομα και Αναγνωριστικό Αριθμό). Η δεύτερη κλάση, περιλαμβάνει ακριβώς μια αναφορά σε μια διαπροσωπία Decomposition, η οποία υλοποιείται από δυο κλάσεις, την ANDDecomposition και την ORDecomposition. Επίσης, έχει οσοδήποτε αριθμό αναφορών σε αντικείμενα τύπου της διαπροσωπίας Goal και μια αναφορά parent, επίσης σε αντικείμενα τύπου της διαπροσωπίας Goal. Με αυτά τα χαρακτηριστικά, επιτυγχάνεται η ανάλυση ενός στόχου σε υποστόχους με AND ή OR Decomposition. Προφανώς, ο τύπος της αναφοράς decomp, δηλώνει τον τύπο της σύνθεσης του στόχου σε υποστόχους (AND ή OR) και η λίστα των αναφορών σε Goal, δηλώνει τους υποστόχους του DecompositionGoal, ενώ η αναφορά parent δηλώνει τον στόχο – πατέρα.

Σημασιολογικά, ο ατομικός στόχος (AtomicGoal), δηλώνει μια λειτουργική απαίτηση του υπό έλεγχο συστήματος, η οποία μπορεί να επαληθευτεί από μόνη της, μέσω των πολιτικών επαλήθευσης που μοντελοποιούνται για αυτήν και οι οποίες θα παρουσιαστούν στην συνέχεια αυτού του κεφαλαίου. Δομικά στην κατασκευή του δέντρου στόχων, AtomicGoal είναι προφανώς τα φύλλα του δέντρου στόχων. Αντίστοιχα, σημασιολογικά ο DecompositionGoal, δηλώνει μια λειτουργική απαίτηση του υπό έλεγχο συστήματος η οποία για να επαληθευτεί χρειάζεται πέραν των πολιτικών επαλήθευσης με τις οποίες επισημαίνεται, να επαληθευτούν κατάλληλα οι υποστόχοι της, ανάλογα με τον τύπο (AND/OR) της σύνθεσης. Στην κατασκευή του δέντρου στόχων, ο DecompositionGoal βρίσκεται στους ενδιάμεσους κόμβους και στην ρίζα του δέντρου στόχων.

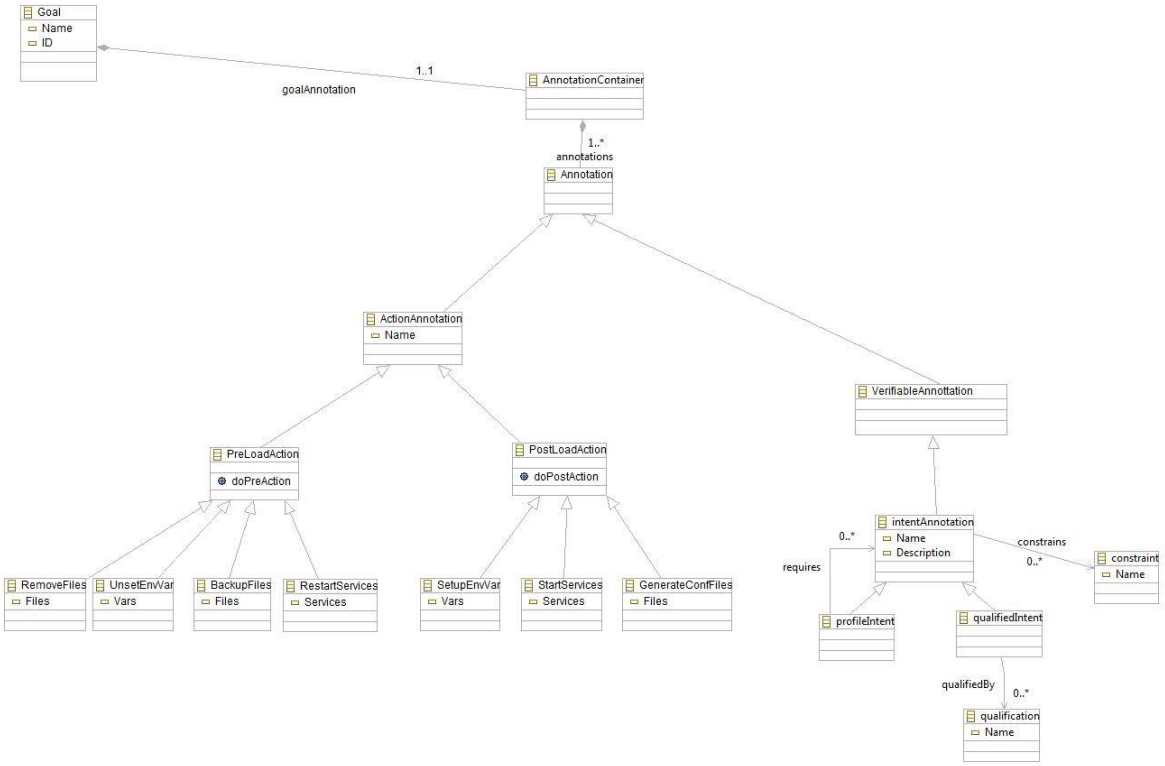
4.4 Επισημειώσεις

Η μεγαλύτερη διαφορά του μοντέλου δέντρου στόχων που παρουσιάζεται στην παρούσα διπλωματική και του αρχικού σχεδιασμού που παρουσιάζεται στο [2], είναι στις επισημειώσεις (Annotations). Με τον όρο επισημειώσεις, εννοούμε επιπρόσθετη, δομημένη πληροφορία η οποία τίθεται ανά στόχο στο δέντρο στόχων και η οποία μπορεί να βοηθήσει σε διάφορες πτυχές της λειτουργίας του περιβάλλοντος-πλαισίου. Ο επιθετικός προσδιορισμός "δομημένη", αναφέρεται στο γεγονός ότι οι γενικοί τύποι που μπορεί να έχουν οι επισημειώσεις, ορίζονται σαφώς στο Μοντέλο Δέντρου Στόχων που παρουσιάζεται. Με τον όρο "πτυχές της λειτουργίας" εννοούμε τα εξής δυο βασικά σημεία στα οποία οι επισημειώσεις βοηθούν την μοντελοποίηση:

- Αναγνώριση όλων εκείνων των πληροφοριών για κάθε στόχο, οι οποίες μπορούν να επαληθευτούν (μπορούν να λάβουν τιμή "ψευδής" ή "αληθής"), ώστε να επαληθεύσουν τον στόχο (στην περίπτωση του AtomicGoal), ή να συμβάλλουν στην επαλήθευση στόχου, μαζί με τις αντίστοιχες επισημειώσεις των υποστόχων (στην περίπτωση του DecompositionGoal).

- Εντοπισμός όλων των ενεργειών ανά στόχο, οι οποίες πρέπει να ενεργοποιηθούν πριν και μετά την επαλήθευση ή εγκαθίδρυση του στόχου. Οι ενέργειες αυτές, μπορούμε να πούμε ότι είναι παρόμοιες με αυτές που συμβαίνουν κατά την εγκατάσταση ενός νέου πακέτου λογισμικού σε ένα λειτουργικό σύστημα. Δηλαδή, αναφέρονται στο "συμμάζεμα" του συστήματος πριν και μετά την εγκατάσταση του πακέτου λογισμικού. Κάτι αντίστοιχο έχουμε και στην περίπτωση αυτή, μόνο που τώρα γίνεται αναφορά σε στόχους και στο "συμμάζεμα" του υπό έλεγχο συστήματος, κατ' αντιστοιχία με το παράδειγμα που δόθηκε.

Τα δυο παραπάνω σημεία, ορίζουν και τους δυο διαφορετικούς, γενικούς τύπους επισημειώσεων που υφίστανται στο μοντέλο δέντρων στόχων που παρουσιάζεται. Παρακάτω αναλύονται σχολαστικά αυτοί οι δυο τύποι επισημειώσεων, αφού δοθεί το δεύτερο τμήμα του μοντέλου πεδίου του μοντέλου δέντρου στόχων, που αφορά στην μοντελοποίηση των επισημειώσεων.



Εικόνα 4-IV: Μοντέλο Πεδίου Μοντέλου Δέντρου Στόχων (2 από 4)

Στην παραπάνω εικόνα, φαίνεται ο τρόπος που μοντελοποιήθηκαν οι επισημειώσεις ανά στόχο για το μοντέλο που παρουσιάζεται. Αρχικά, πρέπει να υποστηρίζεται η δυνατότητα να εμπλουτίζεται κάθε στόχος στο δέντρο στόχων, με οποιοδήποτε αριθμό επισημειώσεων. Για αυτό το σκοπό, στο μοντέλο στην Εικόνα 4-IV για τις επισημειώσεις που μπορεί να έχει κάποιος στόχο χρησιμοποιείται ένα είδος δοχείου (container). Αυτό είναι ο AnnotationContainer που βρίσκεται μόνο ένα ανά στόχο, μέσω της αναφοράς goalAnnotation, η οποία έχει πηγή αντικείμενο Goal και πέρας αντικείμενο AnnotationContainer. Το δοχείο AnnotationContainer, περιέχει οποιοδήποτε αριθμό (μεγαλύτερο ή ίσο του ένα) από αναφορές σε αντικείμενα τύπου Annotation, μέσω της σχέσης annotations. Με αυτόν τον τρόπο, εξασφαλίζεται ότι κάθε δοχείο AnnotationsContainer, μοναδικό ανά στόχο στο δέντρο στόχων, θα εμπεριέχει όλες τις επισημειώσεις που αναφέρονται στον στόχο αυτό.

Αυτό που αναφέρεται στο μοντέλο με το όνομα Annotation, είναι μια διαπροσωπία η οποία ορίζει τον τύπο των επισημειώσεων γενικά. Όπως αναφέρθηκε ήδη, οι επισημειώσεις μπορεί να χωρίζονται σε δυο κατηγορίες, σε επισημειώσεις που επαληθεύονται ή όχι (λαμβάνουν τιμές ψευδές ή αληθές) και σε επισημειώσεις που δηλώνουν ενέργειες. Αυτό εντοπίζεται στο μοντέλο δέντρου στόχων, μέσω των δυο κλάσεων που υλοποιούν την διαπροσωπία επισημείωση, την επαληθεύσιμη επισημείωση (VerifiableAnnotation) και την επισημείωση ενέργειας (ActionAnnotation). Παρακάτω, αναλύονται αυτά τα δυο είδη επισημειώσεων που μπορεί να υπάρχουν.

4.4.1 Επαληθεύσιμες Επισημειώσεις

Ο τύπος των επαληθεύσιμων επισημειώσεων, περιλαμβάνει όλες εκείνες τις πληροφορίες οι οποίες επαληθεύονται ή όχι. Στις επισημειώσεις που μπορούν να επαληθευτούν, έχουν θέση οι πολιτικές και οι προθέσεις που περιλαμβάνονται στο πρότυπο των SCA Συστημάτων Λογισμικού (για τους σκοπούς της παρούσας διπλωματικής εργασίας). Υπενθυμίζεται, ότι οι προθέσεις (intents) στο πρότυπο SCA, αναφέρονται σε περιορισμούς (constraints) που πρέπει να υλοποιεί ένα σύστημα λογισμικού SCA (μια υπηρεσία, μια αναφορά ή ένα δομοστοιχείο). Οι πολιτικές (policies) αναφέρονται στον τρόπο με τον οποίο προσφέρονται αυτοί οι περιορισμοί από το περιβάλλον εκτέλεσης SCA.

Ως επαληθεύσιμο στοιχείο στο μοντέλο δέντρου στόχων, θεωρούμε τις προθέσεις που ισχυρίζεται κατά κάποιο τρόπο ότι προσφέρει ένα δομοστοιχείο (Component) σε ένα SCA Σύστημα Λογισμικού. Προφανώς, η επαλήθευση ή μη των προθέσεων θα γίνει αντιπαραβάλλοντάς τες με τις πολιτικές που προσφέρονται στο περιβάλλον εκτέλεσης SCA, όπως θα φανεί σε επόμενο κεφάλαιο.

Με βάση το παραπάνω, μια άλλη διαπροσωπία που αποτελεί υποτύπο της διαπροσωπίας των επαληθεύσιμων επισημειώσεων, είναι η intentAnnotation. Η διαπροσωπία αυτή, ανά στόχο, αναφέρει ποιες προθέσεις πρέπει να προσφέρονται και να επαληθευτούν κατά την ανάπτυξη του συστήματος. Όπως είναι γνωστό από το πρότυπο SCA, οι προθέσεις μπορεί να είναι δυο τύπων. Πιο συγκεκριμένα, υπάρχουν οι κατανεμημένες προθέσεις (profile intents) και οι προσδιορισμένες προθέσεις (qualified intents). Εξ' ου και οι δυο κλάσεις profileIntent και qualifiedIntent οι οποίες υλοποιούν την διαπροσωπία intentAnnotation και προσφέρει καθεμιά τον αντίστοιχο τύπο προθέσεων.

Σε μια κατανεμημένη πρόθεση, έστω A, αναφέρεται μια λίστα προθέσεων, έστω των $A_{1...m}$. Για να θεωρηθεί ότι ικανοποιείται από μια συγκεκριμένη υλοποίηση η πρόθεση A, πρέπει να ικανοποιούνται όλες οι προθέσεις $A_{1...m}$, που αναφέρονται. Στο μοντέλο δέντρου στόχων που παρουσιάζεται, αυτό εμπεριέχεται στην αναφορά requires που έχει η κλάση profileAnnotation. Η αναφορά αυτή, μπορεί να περιλαμβάνει ως πέρας μια λίστα από οποιοδήποτε αριθμό profileAnnotations (όπως και μια κατανεμημένη πρόθεση μπορεί να αναφέρει οσοδήποτε αριθμό προθέσεων που πρέπει να ικανοποιηθούν). Επίσης, η αναφορά αυτή δεν δεσμεύει τον τύπο των προθέσεων που μπορεί να αναφέρονται, αν θα είναι κατανεμημένες ή προσδιορισμένες, όπως γίνεται και στο πρότυπο SCA.

Σε μια προσδιορισμένη πρόθεση, αναφέρεται ο τρόπος με τον οποίο αυτή η πρόθεση θα ικανοποιηθεί σε επίπεδο σύνδεσης (binding). Για παράδειγμα, η προσδιορισμένη πρόθεση, "confidentiality.transport", δηλώνει ότι η πρόθεση της εμπιστευτικότητας για τα μηνύματα που

ανταλλάσσονται, θα ικανοποιηθούν στο επίπεδο μεταφοράς των μηνυμάτων, όταν η υπηρεσία που περιγράφεται υλοποιηθεί. Κάθε προσδιορισμένη πρόθεση, μπορεί να αναφέρει πολλούς τρόπους με τους οποίους μπορεί να υλοποιηθεί, ή καλύτερα πολλές χαμηλότερου επιπέδου προθέσεις που μπορούν να την προσφέρουν. Για παράδειγμα, στην πρόθεση confidentiality που αναφέραμε προηγουμένως, μπορεί να βρεθούν οι εξής προσδιορισμοί "confidentiality.transport", "confidentiality.message". Οπότε, πρέπει να υποστηρίζεται αυτή η πληθώρα των προσδιορισμών που μπορεί να δεχτεί η προσδιορισμένη πρόθεση. Για αυτό το σκοπό, ορίζεται στο Μοντέλο Δέντρου Στόχων, η αναφορά qualifiedBy του qualifiedIntent. Η αναφορά αυτή έχει πέρας μια λίστα από αντικείμενα qualification, που είναι στα πιο πάνω παραδείγματα το "transport" ή το "message". Με αυτόν τον τρόπο, εξασφαλίζεται η πληθώρα των προσδιορισμών που μπορεί να απαιτούνται.

Εκτός από τις ιδιαίτερες ιδιότητες που έχει κάθε πρόθεση, ανάλογα τον τύπο της και οι οποίες αναλύθηκαν προηγουμένως, υπάρχει μια ιδιότητα η οποία είναι κοινή για όλες τις προθέσεις ανεξαιρέτως. Αυτή η ιδιότητα, είναι το δόμημα (construct) στο οποίο εφαρμόζεται ο περιορισμός της πρόθεσης. Για τον λόγο ότι αυτή η ιδιότητα είναι κοινή για κάθε είδους πρόθεση, μοντελοποιήθηκε ως αναφορά με πέρας την κλάση construct, με προφανείς λόγους ύπαρξης. Η κλάση αυτή, περιλαμβάνει μόνο ένα όνομα (Name) ως αλφαριθμητικό (string) που προσδιορίζει το δόμημα. Η αναφορά, μπορεί να περιλαμβάνει μια λίστα από οποιοδήποτε αριθμό από construct αντικείμενα, καθώς μια πρόθεση μπορεί να εφαρμόζεται σε αρκετά δομήματα.

Πριν κλείσει η ανάλυση των επαληθεύσιμων επισημειώσεων, αξίζει να αναφερθεί ότι ο όρος δόμημα, αναφέρεται σε ένα συγκεκριμένο τύπο SCA αντικειμένων (είτε στο στάδιο της σύνδεσης-binding, είτε στο στάδιο της υλοποίησης-implementation). Αυτό μπορεί να αναφέρεται, για παράδειγμα στον τύπο ενός δομοστοιχείου που ορίζεται σε ένα μοντέλο SCA (Component Type). Επίσης, υπάρχει η δυνατότητα να αναφέρεται ένας αφαιρετικός τύπος XML Element (μια έκφραση XPath), για παράδειγμα, "sca:binding" ή "sca:implementation", οπότε η πρόθεση εφαρμόζεται σε κάθε αντικείμενο SCA, στο επίπεδο της σύνδεσης ή της υλοποίησης. Αυτό αποτελεί την συνηθέστερη περίπτωση στο στάδιο μοντελοποίησης SCA συστημάτων λογισμικού.

4.4.2 Επισημειώσεις Ενέργειας

Στο μοντέλο δέντρου στόχων που παρουσιάζεται, οι επαληθεύσιμες επισημειώσεις δεν είναι ο μοναδικός τύπος επισημειώσεων που υπάρχουν. Υπάρχει και ο τύπος των επισημειώσεων ενέργειας. Οι επισημειώσεις αυτές, μοντελοποιήθηκαν περισσότερο για την ικανοποίηση απαιτήσεων από δεδομένα που λαμβάνονται από αρχεία εγκατάστασης ή ρυθμίσεων και όχι τόσο προς ικανοποίηση της συμβατότητας με το μοντέλο SCA συστημάτων λογισμικού.

Ο τύπος των επισημειώσεων ενέργειας, αναφέρεται στην οποιαδήποτε επιπλέον πληροφορία ανά στόχο, η οποία αναφέρεται στις λειτουργίες που πρέπει να εκτελεστούν από το υπό έλεγχο σύστημα, πριν ή μετά την επαλήθευση του στόχου. Η ιδέα των λειτουργιών πριν και μετά την επαλήθευση ενός στόχου, προέρχεται από τον τρόπο που λειτουργούν στα σύγχρονα συστήματα λογισμικού, τα αρχεία/πακέτα εγκατάστασης. Ένα αρχείο εγκατάστασης ενός πακέτου λογισμικού προβλέπει, αποθηκευμένες σε διάφορες μορφές (εκτελέσιμα αρχεία σεναρίων – scripts, μεταβλητές περιβάλλοντος – environmental variables), ενέργειες που πρέπει να γίνουν πριν και μετά την εγκατάσταση του πακέτου λογισμικού στο υπολογιστικό σύστημα. Επίσης, τα αρχεία ρυθμίσεων ενός συστήματος λογισμικού, οφείλουν να συνυπολογίζουν την κατάσταση του υπολογιστικού συστήματος στο οποίο πρόκειται να εγκατασταθεί ή έχει ήδη εγκατασταθεί το πακέτο λογισμικού.

Όσα αναφέρθηκαν πιο πάνω, οδηγούν την μοντελοποίηση των επισημειώσεων ενέργειας, αρχικά να διαχωρίζει χρονικά τις ενέργειες που πρέπει να συμβούν για κάθε στόχο, προφανώς, σε ενέργειες πριν την επαλήθευση και ενέργειες μετά την επαλήθευση. Όπως φαίνεται και στην Εικόνα 4-IV, οι επισημειώσεις ενέργειας, μοντελοποιούνται ως την διαπροσωπία ActionAnnotation, που αποτελεί υποτύπο της διαπροσωπίας Annotation και υλοποιείται με την σειρά της, από δυο διαπροσωπίες-υποτύπους της, PreLoadAction και PostLoadAction. Οι δυο τελευταίες διαπροσωπίες, αναφέρονται προφανώς στις ενέργειες που πρέπει να γίνουν πριν και μετά την επαλήθευση του στόχου, αντίστοιχα. Αξιοσημείωτο είναι το γεγονός, ότι αυτές οι διαπροσωπίες περιέχουν τον περιορισμό, ότι όποια κλάση της υλοποιεί πρέπει να προσφέρει και μια υλοποίηση της μεθόδου doPreAction ή doPostAction, αναλόγως. Αυτό είναι μια απαίτηση που ζητείται από τον χρήστη του περιβάλλοντος-πλαισίου που παρουσιάζεται, η οποία ταυτόχρονα του δίνει και την δυνατότητα να χρησιμοποιήσει το μοντέλο δέντρου στόχων, για τεχνολογίες που μπορεί να περιλαμβάνουν οποιουσδήποτε τύπος επισημειώσεων ενέργειας. Ακολουθεί η ανάλυση καθεμιάς από τις δυο κατηγορίες ενεργειών.

4.4.2.1 Πριν την Επαλήθευση Ενέργειες

Ακολουθεί μια λίστα με την αφαιρετική περιγραφή των ενεργειών που συνήθως αναφέρονται στην χρονική στιγμή πριν την επαλήθευση του στόχου, καθώς και το πώς μοντελοποιήθηκαν στο μοντέλο δέντρου στόχων που παρουσιάζεται.

- **Αφαίρεση Αρχείων.** Σε πολλές περιπτώσεις εγκατάστασης λογισμικού, υπάρχει η ανάγκη για την διαγραφή από το υπολογιστικό σύστημα προορισμού, όλων των αρχείων που ενδεχομένως σχετίζονται με το σύστημα λογισμικού που πρόκειται να εγκατασταθεί. Σε τέτοια αρχεία, ενδεχομένως περιλαμβάνονται αρχεία εκτελέσιμα, αρχεία ρυθμίσεων που χρησιμοποιεί το σύστημα λογισμικού ή αρχεία που διατηρούν κάποια δεδομένα βάσεων δεδομένων για το σύστημα λογισμικού. Η αφαίρεση αυτών των αρχείων από το

υπολογιστικό σύστημα προορισμού, είναι σε πολλές περιπτώσεις απαραίτητη, ώστε να αποφευχθούν προβλήματα συμβατότητας του λογισμικού που θα εγκατασταθεί με στιγμιότυπα του ιδίου, πιθανώς παλαιότερων εκδόσεων που έχουν εγκατασταθεί προηγουμένως στο υπολογιστικό σύστημα προορισμού. Η περίπτωση αυτή μοντελοποιείται στο μοντέλο δέντρου στόχων, μέσω της κλάσης *RemoveFiles* η οποία υλοποιεί την διαπροσωπία PreLoadAction. Η κλάση αυτή, περιέχει ως ιδιότητα την *Files* μια λίστα από αρχεία, τα μονοπάτια (paths) στο υπολογιστικό σύστημα προορισμού των αρχείων που πρέπει να αφαιρεθούν πριν την εγκατάσταση.

- **Διαγραφή Μεταβλητών Περιβάλλοντος.** Αντίστοιχη με την προηγούμενη περίπτωση, είναι και η διαγραφή μεταβλητών περιβάλλοντος που χρησιμοποιούνται από το σύστημα λογισμικού που θα εγκατασταθεί, στην περίπτωση που αυτές υπάρχουν και έχουν τεθεί σε συγκεκριμένες τιμές από άλλα συστήματα λογισμικού ή από στιγμιότυπα του προς εγκατάσταση λογισμικού, αλλά διαφορετικών εκδόσεων. Με αυτόν τον τρόπο, γίνεται προσπάθεια να βρεθούν κενές οι μεταβλητές περιβάλλοντος από τις ενέργειες μετά την εγκατάσταση, όπου θα τις θέσουν στις σωστές τιμές για την ασφαλή λειτουργία του στιγμιότυπου του συστήματος λογισμικού που εγκαθίσταται. Η περίπτωση αυτή μοντελοποιείται στο μοντέλο δέντρου στόχων, μέσω της κλάσης *UnsetEnvVar* η οποία υλοποιεί την διαπροσωπία PreLoadAction. Η κλάση αυτή, περιέχει ως ιδιότητα την Vars, μια λίστα από μεταβλητές περιβάλλοντος, που (στην περίπτωση που υπάρχουν) πρέπει να τεθούν σε κενές τιμές.

- **Αντιγραφή Ασφαλείας Αρχείων.** Αρκετά συνηθισμένο στις περιπτώσεις εγκατάστασης συστημάτων λογισμικού, είναι η αντιγραφή ασφαλείας των αρχείων (συνήθως ρυθμίσεων και εκτελέσιμα) που περιλαμβάνονται σε ένα στιγμιότυπο εγκατάστασης παλιότερης έκδοσης του λογισμικού που πρόκειται να εγκατασταθεί. Αυτό είναι απαραίτητο και συνηθισμένο στις περιπτώσεις όπου εγκαθίσταται λογισμικό το οποίο δεν έχει ακόμα εκδοθεί επίσημα (official release) και είναι σε στάδιο ανάπτυξης και αποσφαλμάτωσης. Η λήψη αντιγράφου ασφαλείας για την παλαιότερη έκδοση του λογισμικού που εγκαθίσταται, γίνεται για την περίπτωση που χρειασθεί υποβάθμιση έκδοσης (downgrade) στην παλαιότερη έκδοση, λόγω σοβαρού προβλήματος που προκαλείται από την νέα έκδοση που εγκαθίσταται. Η περίπτωση αυτή, μοντελοποιείται με την κλάση *BackupFiles* η οποία υλοποιεί την διαπροσωπία PreLoadAction. Η κλάση αυτή, περιέχει μια ιδιότητα την *Files* η οποία είναι μια λίστα των μονοπατιών των αρχείων που πρέπει να συμπεριληφθούν σε ένα αντίγραφο ασφαλείας.

- **Επανεκκίνηση Υπηρεσιών.** Με τον όρο "Υπηρεσία" στην προκειμένη περίπτωση (σε αντιπαραβολή με αυτό που θεωρείται υπηρεσία για το πρότυπο SCA), θεωρούμε μια διεργασία που λειτουργεί στο παρασκήνιο αενάως στο λειτουργικό σύστημα ενός υπολογιστικού συστήματος, εκτελώντας περιοδικά κάποιες ενέργειες ή περιμένοντας για κλήση της. Στην περίπτωση που το υπό εγκατάσταση λογισμικό, προσφέρει μια νέα έκδοση αυτής της υπηρεσίας, πρέπει να τερματιστεί η υπηρεσία που ήδη τρέχει στο λειτουργικό σύστημα του υπολογιστικού συστήματος προορισμού. Με αυτόν τον τρόπο, θα αποφευχθεί πρόβλημα από την εγγραφή των νέων εκτελέσιμων αρχείων της υπηρεσίας (τα οποία έχουν ήδη φορτωθεί και εκτελούνται) και δημιουργούνται οι σωστές προϋποθέσεις να ξεκινήσει μετά την εγκατάσταση η νέα έκδοση της υπηρεσίας. Η περίπτωση αυτή μοντελοποιείται με την κλάση *RestartServices* η οποία υλοποιεί την διαπροσωπία PreLoadAction. Η κλάση αυτή, περιέχει μια ιδιότητα την *Services* η οποία είναι μια λίστα με τις υπηρεσίες που πρέπει να τερματισθούν πριν η εγκατάσταση προβεί στην αναβάθμισή τους (ή στην γενική περίπτωση, στην αλλαγή έκδοσής τους).

4.4.2.2 Μετά την Επαλήθευση Ενέργειες

Ακολουθεί μια λίστα με την αφαιρετική περιγραφή των ενεργειών που συνήθως αναφέρονται στην χρονική στιγμή μετά την επαλήθευση του στόχου, καθώς και το πώς μοντελοποιήθηκαν στο Μοντέλο Δέντρου Στόχων που παρουσιάζεται.

- **Ανάθεση Μεταβλητών Περιβάλλοντος.** Αφού οι "παλιές" (ή στην γενική περίπτωση διαφορετικές από του στιγμιότυπου που πρόκειται να εγκατασταθεί) τιμές των μεταβλητών περιβάλλοντος έχουν εκκαθαριστεί, η μετά την εγκατάσταση ρύθμιση του συστήματος λογισμικού, πρέπει να αναλάβει να δημιουργήσει τις απαραίτητες μεταβλητές περιβάλλοντος που θα χρειαστούν, καθώς και να τους αναθέσει τις κατάλληλες τιμές. Η περίπτωση αυτή μοντελοποιείται με την κλάση *SetupEnvVar* η οποία υλοποιεί την διαπροσωπία PostLoadAction. Η κλάση αυτή περιέχει την ιδιότητα *Vars* η οποία είναι μια λίστα από μεταβλητές περιβάλλοντος που πρέπει να δημιουργηθούν (αν δεν υπάρχουν ήδη) και τις τιμές που πρέπει να λάβουν.

- **Εκκίνηση Υπηρεσιών.** Όπως αναφέρθηκε και στην παράγραφο 4.4.2.1 πριν την εγκατάσταση ενός συστήματος λογισμικού, πρέπει να τερματισθούν οι υπηρεσίες που ενδεχομένως να ενημερωθούν ή να αλλάξουν από αυτό. Μετά την εγκατάσταση οι υπηρεσίες αυτές καθώς και όσες δεν υπήρχαν στο υπολογιστικό σύστημα προορισμού και προσφέρονται από το σύστημα λογισμικού που εγκαταστάθηκε, πρέπει να εκκινήσουν. Η περίπτωση αυτή μοντελοποιείται με την κλάση *StartServices* η οποία υλοποιεί την

διαπροσωπία PostLoadAction. Η κλάση αυτή περιέχει την ιδιότητα *Services* η οποία είναι μια λίστα με τις υπηρεσίες που προσφέρονται από το σύστημα λογισμικού που εγκαταστάθηκε και πρέπει να εκκινήσουν.

- **Παραγωγή Αρχείων Ρυθμίσεων.** Στα περισσότερα αρχεία εγκατάστασης λογισμικού, περιέχονται μικρά εκτελέσιμα σενάρια, τα οποία επιτρέπουν την αυτόματη παραγωγή των αρχείων ρυθμίσεων που θα χρησιμοποιεί το σύστημα που εγκαταστάθηκε. Η αυτοματοποιημένη παραγωγή των αρχείων ρυθμίσεων, είναι απαραίτητη καθώς το περιεχόμενο των αρχείων ρυθμίσεων μπορεί να εξαρτάται από διάφορες παραμέτρους του υπολογιστικού συστήματος προορισμού (χαρακτηριστικά λειτουργικού συστήματος, μονοπάτι βιβλιοθηκών που είναι απαραίτητες και άλλα). Η περίπτωση αυτή μοντελοποιείται με την κλάση *GenerateConfFiles* η οποία υλοποιεί την διαπροσωπία PostLoadActions. Η κλάση αυτή, περιέχει την ιδιότητα *Files* η οποία είναι μια λίστα με τα αρχεία των εκτελέσιμων σεναρίων που θα δημιουργήσουν τα κατάλληλα αρχεία ρυθμίσεων που χρειάζεται το σύστημα λογισμικού που μόλις εγκαταστάθηκε.

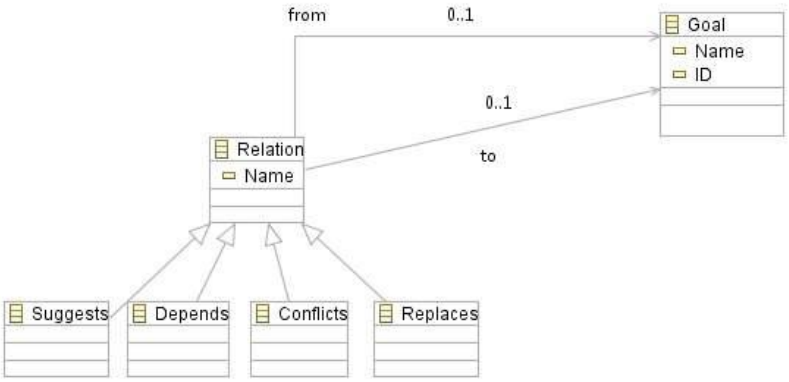
Στο σημείο αυτό αξίζει να σημειωθεί ότι στην παραπάνω ανάλυση προέκυψε μια σιωπηρή ταύτιση των όρων "επαλήθευση ενός στόχου" και "εγκατάστασης ενός πακέτου/συστήματος λογισμικού". Αυτό, όπως θα αιτιολογηθεί καλύτερα στο κεφάλαιο της υλοποίησης, γίνεται επειδή το περιβάλλον-πλαίσιο που παρουσιάζεται, στοχεύει στην εξέταση του υπό έλεγχο συστήματος λογισμικού στις περιπτώσεις της εισαγωγής, ανανέωσης και αφαίρεσης δομοστοιχείων του. Αυτές οι περιπτώσεις, ομοιάζουν σε μεγάλο βαθμό με τις περιπτώσεις όπου έχουμε εγκατάσταση, ανανέωση (αναβάθμιση ή υποβάθμιση) και απεγκατάσταση πακέτων λογισμικού από ένα λειτουργικό σύστημα. Εξ' ου και η ταύτιση των όρων. Αξίζει όμως να σημειωθεί, ότι αυτό το γεγονός δεν βάλλει κατά της γενικότητας της χρήσης του περιβάλλοντος-πλαισίου που παρουσιάζεται, καθώς τα συνηθέστερα σενάρια αλλαγής της συμπεριφοράς ενός συστήματος λογισμικού μπορούν να προσεγγιστούν/προσομοιωθούν με σενάρια αλλαγής, εισαγωγής ή αφαίρεσης δομοστοιχείων του. Χαρακτηριστικά παραδείγματα, η περίπτωση που μια συγκεκριμένη κλάση ενός συστήματος λογισμικού παρουσιάζει εσφαλμένη λειτουργία συμπίπτει με την περίπτωση της αλλαγής ενός δομοστοιχείου με ένα εσφαλμένο και η περίπτωση του αναπάντεχου τερματισμού ενός νήματος εκτέλεσης συμπίπτει με την αφαίρεση ενός δομοστοιχείου.

4.5 Σχέσεις μεταξύ Στόχων

Η αποσύνθεση στόχων στους υποστόχους του, δεν είναι η μόνη σχέση που μπορεί να παρουσιάζεται μεταξύ στόχων σε ένα σύστημα λογισμικού. Όπως αναφέρεται και στο [2], υπάρχει και η έννοια των *softgoals*, δηλαδή στόχων, οι οποίοι δεν αναλύονται απαραίτητα σε μια αυστηρή λίστα υποστόχων η

οποία πρέπει να ικανοποιείται. Οι *softgoals* όπως έχει ήδη εξηγηθεί επαληθεύονται με βάση το αν υπάρχουν περισσότερα "αποδεικτικά στοιχεία" υπέρ ή κατά τους. Επίσης, ένα μειονέκτημα του μοντέλου δέντρου στόχων που έχει παρουσιασθεί, είναι το γεγονός ότι εξαρτήσεις μέσω AND/OR αποσυνθέσεων στόχων, επιβάλλουν οι σχετιζόμενοι στόχοι να προέρχονται μόνο από ένα δέντρο στόχων και όχι από περισσότερα. Στις περιπτώσεις των κατανεμημένων και υπηρεσιοκεντρικών συστημάτων όμως, οι εξαρτήσεις μπορεί να ποικίλλουν και να αφορούν δομοστοιχεία λογισμικού τα οποία βρίσκονται σε διαφορετικά υπολογιστικά συστήματα.

Πρέπει, λοιπόν, να μοντελοποιηθούν και σχέσεις μεταξύ στόχων, που να μην περιορίζονται στα στενά σύνορα του Δέντρου Στόχων, αλλά και να περιλαμβάνουν μια μάλλον λειτουργική και όχι δομική σχέση μεταξύ στόχων. Η μοντελοποίηση αυτή, φαίνεται σχηματικά στην Εικόνα 4-V.



Εικόνα 4-V: Μοντέλο Πεδίου Μοντέλου Δέντρου Στόχων (3 από 4)

Όπως φαίνεται και στο σχήμα, κάθε στόχος στο μοντέλο δέντρου στόχων, περιέχει και μια αναφορά *relation* σε αντικείμενο τύπου *Relation.* Η *Relation* είναι η διαπροσωπία στην οποία μοντελοποιήθηκαν οι σχέσεις που παρουσιάσθηκαν προηγουμένως. Περιέχει μια ιδιότητα, *Name,* ένα αλφαριθμητικό το οποίο δηλώνει το όνομα με το οποίο απευθύνεται το περιβάλλον-πλαίσιο στην συγκεκριμένη σχέση. Στην διαπροσωπία *Relation* υπάρχουν δυο αναφορές, *from* και *to* στους δυο στόχους που σχετίζονται με αυτήν. Η πρώτη αναφορά είναι ο στόχος από τον οποίο πηγάζει η εξάρτηση και η δεύτερη ο στόχος ο οποίος δέχεται την εξάρτηση. Όπως έχει ήδη αναφερθεί, οι στόχοι οι οποίοι σχετίζονται, μπορεί να αποτελούν μέλη δυο διαφορετικών δέντρων στόχων. Συνεπώς, οι δυο αναφορές from και to, μπορεί να είναι στόχοι σε δυο διαφορετικά δέντρα στόχων, τα οποία να υφίστανται στην μνήμη δυο διαφορετικών ηλεκτρονικών υπολογιστών. Για να μπορέσει να υλοποιηθεί αυτό, η σχέση ορίζεται πως μοντελοποιείται πάντα στο δέντρο στόχων στο οποίο ανήκει ο στόχος από τον οποίο πηγάζει η σχέση (στόχος στην from αναφορά). Στην περίπτωση όπου ο στόχος στην αναφορά to, ανήκει σε διαφορετικό δέντρο στόχων, το οποίο επιπροσθέτως υπάρχει

στην μνήμη διαφορετικού ηλεκτρονικού υπολογιστή, τότε ορίζεται ένας εικονικός στόχος στην αναφορά αυτή, ο οποίος περιέχει απλά μια μέθοδο για την προώθηση της οποιαδήποτε αίτησης, στον κατάλληλο υπολογιστή που περιέχει το κατάλληλο δέντρο στόχων.

Η διαπροσωπία *Relation* είναι ο γενικός τύπος ο οποίος μοντελοποιεί τις σχέσεις μεταξύ στόχων που δεν αναλύονται σε AND/OR αποσυνθέσεις. Ακολουθώντας το αντικειμενοστραφές μοντέλο προγραμματισμού, η σημασιολογία της κάθε σχέσης που μπορεί να εμπίπτει στην κατηγορία της μη AND/OR αποσύνθεσης, μπορεί να μοντελοποιηθεί από κλάσεις που θα υλοποιούν την διαπροσωπία *Relation*. Δίνεται έτσι η δυνατότητα στον χρήστη του περιβάλλοντος-πλαισίου, να ορίσει τις δικές του σχέσεις που εμπίπτουν στην *Relation,* με τον ορισμό των κλάσεων που θα τις υλοποιούν και θα εμπεριέχουν την σημασιολογία που επιθυμεί ο ίδιος να δώσει στις σχέσεις/κλάσεις αυτές. Κατά την σχεδίαση του συστήματος όμως, ορίσαμε τέσσερις βασικές σχέσεις που υλοποιούν την διαπροσωπία *Relations* και θεωρούνται οι πιο γενικές που μπορεί να ορισθούν για το πρόβλημα που προσπαθεί το Μοντέλο Δέντρου Στόχων να επιλύσει.

Παρακάτω, αναλύονται μια προς μια οι σχέσεις αυτές καθώς και η σημασιολογία που τους έχει δοθεί από τον σχεδιαστή του συστήματος.

- **Προτείνει (Suggests).** Η σχέση αυτή, αναφέρεται στην περίπτωση όπου ο στόχος πηγή εξαρτάται από τον στόχο πέρας, για την πλήρη λειτουργία του και τη χρησιμοποίηση όλων των δυνατοτήτων που έχει. Αυτό όμως δεν σημαίνει ότι ο στόχος πέρας είναι απαραίτητος για την λειτουργία του στόχου πηγή. Ένα παράδειγμα μπορεί να δοθεί στην περίπτωση εγκατάστασης ενός συστήματος λογισμικού, με στόχο πηγή την εγκατάσταση του ίδιου του λογισμικού και στόχο πέρας την εγκατάσταση των αρχείων βοήθειάς του. Στην περίπτωση αυτή ο στόχος της εγκατάστασης του λογισμικού θα **πρότεινε** και την εγκατάσταση των αρχείων βοήθειάς του, για την καλύτερη εξυπηρέτηση του χρήστη, όμως αυτό δεν είναι απαραίτητο για την λειτουργία του λογισμικού. Ένα άλλο παράδειγμα, μπορεί να δοθεί στην περίπτωση που μια δικτυακή υπηρεσία μπορεί να υποστηρίξει και ασφαλή και μη ασφαλή σύνδεση από τους πελάτες της. Στην περίπτωση αυτή, η υπηρεσία θα **πρότεινε** την χρήση της υποστήριξης κάποιου πρωτοκόλλου ασφαλούς μεταφοράς δεδομένων (certificates,https) αλλά κάτι τέτοιο δεν είναι απαραίτητο για την λειτουργία της, αφού υπάρχει πάντα η δυνατότητα μη ασφαλούς χρήσης της. Μια τέτοια σχέση αποτελεί μια θετική συνεισφορά στην επαλήθευση του στόχου πέρας.

- **Εξαρτάται (Depends).** Η σχέση αυτή, αναφέρεται στην περίπτωση όπου ο στόχος πηγή έχει μια λειτουργική εξάρτηση από τον στόχο πέρας. Η εξάρτηση αυτή, λόγω του ότι είναι λειτουργική, δηλώνει ότι η πλήρης λειτουργία του στόχου πηγή έχει ανάγκη την λειτουργία

του στόχου πέρας. Ένα παράδειγμα χαρακτηριστικό της εξάρτησης αυτής, είναι οι βιβλιοθήκες με των οποίων την βοήθεια γράφεται ένα πρόγραμμα και το πρόγραμμα το ίδιο. Αν το πρόγραμμα θεωρηθεί ως στόχος πηγή και οι βιβλιοθήκες ως στόχος πέρας, τότε μεταξύ τους έχουμε μια σχέση **Εξαρτάται (Depends).** Η εξάρτηση αυτή, υποδηλώνει ότι χωρίς την ύπαρξη των βιβλιοθηκών, το πρόγραμμα δεν μπορεί καν να λειτουργήσει. Η ύπαρξη μιας τέτοιας σχέσης αποτελεί μια θετική συνεισφορά στην επαλήθευση του στόχου πηγή.

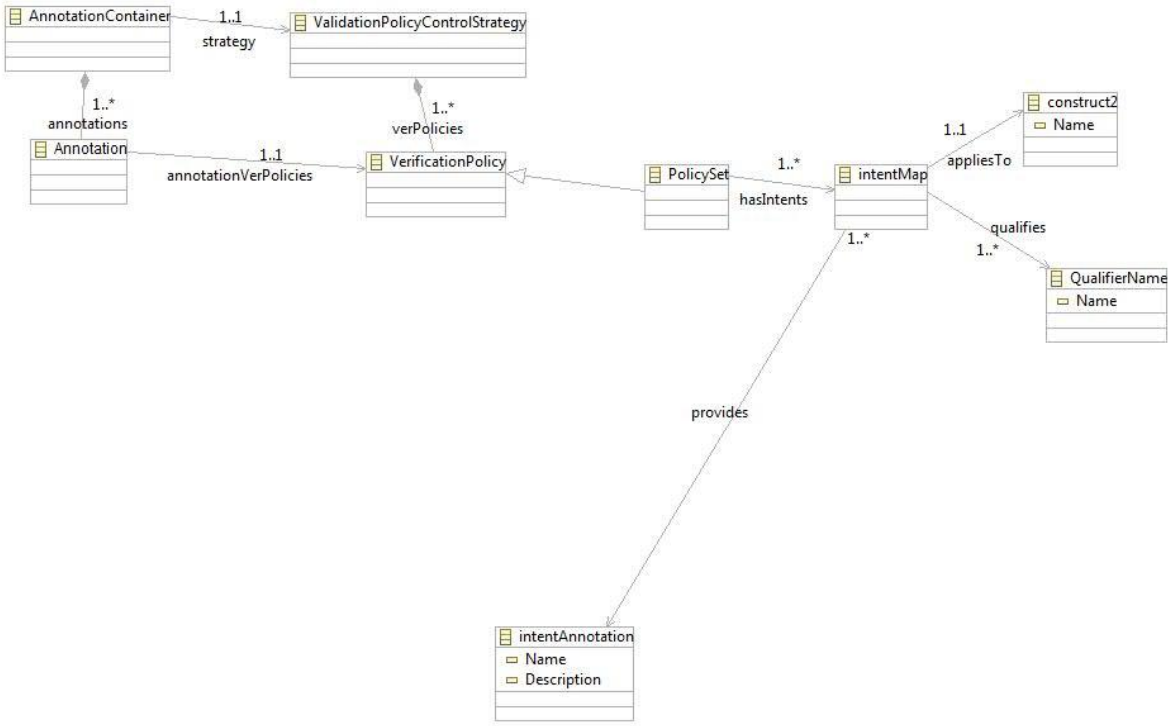
- **Αντιτίθεται (Conflicts).** Η σχέση αυτή, αναφέρεται στην περίπτωση όπου ο στόχος πηγή αντιτίθεται στην λειτουργία του στόχου πέρας. Η σημασιολογία της σχέσης αυτής, είναι ότι οι δυο στόχοι (πηγή και πέρας) προσφέρουν την ίδια λειτουργικότητα, αλλά με εντελώς διαφορετικούς τρόπους, οι οποίο δεν είναι συμβατοί μεταξύ τους. Ένα τέτοιο παράδειγμα, είναι η περίπτωση που οι δυο στόχοι είναι στην ουσία το ίδιο σύστημα λογισμικού, αλλά διαφορετικών εκδόσεων, η λειτουργικότητα των οποίων δεν είναι συμβατή από την μια έκδοση στην άλλη. Η ύπαρξη μια τέτοιας σχέσης αποτελεί αρνητική συνεισφορά στην επαλήθευση του στόχου πέρας.

- **Αντικαθιστά (Replaces).** Η σχέση αυτή αναφέρεται στην περίπτωση όπου ο στόχος πηγή αντικαθιστά την λειτουργικότητα του στόχου πέρατος. Η σημασιολογία αυτής της σχέσης, είναι η αντικατάσταση του ενός στόχου από τον άλλο, δηλαδή ότι και οι δυο προσφέρουν με παρόμοιο τρόπο την ίδια λειτουργικότητα. Αυτό που διαφέρει όμως από την σχέση **Conflicts** είναι το γεγονός ότι οι δυο λειτουργικότητες μπορεί να συνυπάρξουν στο ίδιο υπολογιστικό σύστημα ή στο ίδιο περιβάλλον εκτέλεσης (runtime). Ένα τέτοιο παράδειγμα, είναι η περίπτωση δυο δικτυακών υπηρεσιών οι οποίες προσφέρουν ασφαλή επικοινωνία μεταξύ δυο υπολογιστών στο δίκτυο. Η διαφορά τους είναι, ότι η πρώτη υπηρεσία προσφέρει την ασφαλή επικοινωνία μέσω πρωτοκόλλου *https* ενώ η δεύτερη υπηρεσία την προσφέρει μέσω σύνδεσης με *SSL*. Στην περίπτωση αυτή, μπορούμε να πούμε ότι οι στόχοι που θα αναφέρονται στις δυο αυτές υπηρεσίες συνδέονται μεταξύ τους με μια σχέση **Replaces.** Η ύπαρξη μιας τέτοιας σχέσης, αποτελεί θετική συνεισφορά στην επαλήθευση και των δυο στόχων (πηγής και πέρατος).

4.6 Πολιτικές Επαλήθευσης Επισημειώσεων

Σκοπός του περιβάλλοντος-πλαισίου είναι η προσπάθεια της επαλήθευσης των στόχων που βρίσκονται στο μοντέλο δέντρου στόχων, με την χρήση των πληροφοριών που υπάρχουν στην δομή του δέντρου αλλά και στις επισημειώσεις ανά στόχο που εμπεριέχονται στο μοντέλο. Στο ίδιο το μοντέλο δέντρου στόχων, δίνεται η δυνατότητα να οριστούν από τον χρήστη οι πολιτικές

επαλήθευσης που θα συνεπικουρήσουν το περιβάλλον-πλαίσιο στην επαλήθευση του τρέχοντα στόχου. Η μοντελοποίηση αυτή, με βάση και το πρότυπο SCA, φαίνεται στην παρακάτω εικόνα.



Εικόνα 4-VI: Μοντέλο Πεδίου Μοντέλου Δέντρου Στόχων (4 από 4)

Στο μοντέλο δέντρου στόχων, δίνεται η δυνατότητα ανά επισημείωση να δίνονται οι πολιτικές/στρατηγικές που θα βοηθήσουν το περιβάλλον-πλαίσιο στην επαλήθευσή της. Σε κάθε δοχείο επισημειώσεων δίνεται μια αναφορά σε αντικείμενο τύπου ValidationPolicyControlStrategy. Η σημασιολογία αυτού του αντικειμένου είναι ως το κοινό δοχείο όλων των πολιτικών και στρατηγικών επαλήθευσης έχει ο στόχος, κατ' αντιστοιχία με την σημασιολογία του δοχείου επισημειώσεων (AnnotationContainer), που είναι το κοινό σημείο που περιέχει όλες τις επισημειώσεις του στόχου. Το αντικείμενο αυτό, ValidationPolicyControlStrategy, είναι το ίδιο αντικείμενο που μνημονεύθηκε στην παράγραφο 3.3.2 καθώς ο Διαχειριστής Πολιτικών, ανάλογα τον τρέχοντα στόχο καλείται να ενεργοποιήσει τα δοχεία Στρατηγικών και πολιτικών επαλήθευσης επισημειώσεων, για να επαληθευτούν ή όχι οι στόχοι του συνόλου VS.

Κάθε αντικείμενο τύπου ValidationPolicyControlStrategy, περιέχει μια λίστα με αντικείμενα τύπου πολιτικής επαλήθευσης (VerificationPolicy). Η Πολιτική Επαλήθευσης είναι μια γενική διαπροσωπία που μοντελοποιεί την πολιτική που πρέπει να ενεργοποιηθεί ανά επισημείωση, ώστε η επισημείωση να επαληθευτεί ή όχι. Βλέπουμε ότι τα δοχεία (AnnotationContainer και ValidationPolicyControlStrategy) και τα αντικείμενα που περιέχονται σε αυτά (Annotation και VerificationPolicy), έχουν βίους παράλληλους, καθώς μοντελοποιούνται με παράλληλους τρόπους

όπου κάθε δοχείο έχει μια λίστα με τα αντίστοιχα "περιεχόμενα" και τα δοχεία και τα "περιεχόμενα" έχουν αναφορές μεταξύ τους, ώστε να δηλώνεται η ιεραρχία και σε ποια επισημείωση εφαρμόζεται κάθε πολιτική επαλήθευσης.

Η VerificationPolicy είναι μια διαπροσωπία, όπως ακριβώς και το "ανάλογό" της, η επισημείωση (Annotation). Με αυτόν τον τρόπο, δίνεται η δυνατότητα στον χρήστη του περιβάλλοντος-πλαισίου να ορίσει τις δικές του υλοποιήσεις της VerficationPolicy, ώστε να προσφέρει τις δικές του πολιτικές επαλήθευσης, είτε στις υπάρχουσες επισημειώσεις, είτε στις επισημειώσεις που θα έχει και ο ίδιος ορίσει (υλοποιώντας την διαπροσωπία Annotation).

Για τους σκοπούς του πριν την χρήση ελέγχου SCA συστημάτων λογισμικού, προτείνεται μια υλοποίηση της πολιτικής επαλήθευσης, η οποία αποσκοπεί στο να δώσει τρόπους να ελεγχθούν οι προθέσεις (intents) των στόχων ενός δέντρου στόχων που μοντελοποιεί ένα SCA Σύστημα. Η υλοποίηση ονομάζεται PolicySet και αναφέρεται στον βασικό τρόπο που περιγράφει το SCA πρότυπο για την επαλήθευση των προθέσεων των δομοστοιχείων ενός SCA συστήματος, ο οποίος μάλιστα φέρει και το ίδιο όνομα.

Στο πρότυπο SCA, PolicySet ορίζονται για να δείξουν τις συγκεκριμένες πολιτικές που ακολουθούνται από το SCA σύστημα λογισμικού, ώστε να ικανοποιηθούν οι προθέσεις (intents) που υπάρχουν από τις διάφορες υπηρεσίες του συστήματος. Ένα PolicySet, μπορεί να ικανοποιεί μια λίστα από προθέσεις (intents). Για να είναι εφικτό αυτό, το πρότυπο SCA, προβλέπει κάθε PolicySet, να μπορεί να έχει μια λίστα από intentMap. Καθένα αντικείμενο intentMap είναι υπεύθυνο για την προσφορά στο περιβάλλον εκτέλεσης SCA (SCA runtime), μιας πρόθεσης που έχει ζητηθεί από κάποιο δομοστοιχείο. Όλα τα παραπάνω, μοντελοποιήθηκαν στην Εικόνα 4-VI μέσω της αναφοράς hasIntents από PolicySet σε intentMap και της αναφοράς provides από intentMap σε intentAnnotation, που υποδεικνύει ποιες προθέσεις ικανοποιεί το κάθε intentMap.

Κάθε αντικείμενο intentMap, κατά το πρότυπο SCA, μπορεί εκτός από την πρόθεση που προσφέρει να έχει και άλλες δυο ιδιότητες, την *construct* και την *qualifies*. Η πρώτη είναι παρόμοια με την ιδιότητα *construct* του intentAnnotation και αναφέρεται σε ποιο δόμημα του SCA μοντέλου πρόκειται να εφαρμοστεί το συγκεκριμένο intentMap. Η ιδιότητα αυτή μοντελοποιείται μέσω της αναφοράς *appliesTo* σε αντικείμενο *construct2* (το όνομα επιλέχθηκε ώστε να μην συγχέεται με το *construct* του intentAnnotation). Η δεύτερη ιδιότητα, αναφέρεται σε intentMap που θέλουν να ικανοποιήσουν προσδιορισμένες προθέσεις (qualified intents), οπότε δηλώνουν με αυτήν την ιδιότητα τον τρόπο με τον οποίο προσφέρεται η προσδιορισμένη πρόθεση (π.χ. το "transport" στην πρόθεση "confidentiality" της παραγράφου 4.4.1). Η ιδιότητα αυτή μοντελοποιείται με την αναφορά *qualifies* από intentMap σε αντικείμενα *QualifierName*.

Κεφάλαιο 5: Πολιτικές και Αλγόριθμοι Επαλήθευσης και Ελέγχου Διαδικαστικών Μοντέλων

5.1 Εισαγωγή

Τα δυο προηγούμενα κεφάλαια, παρουσίασαν το περιβάλλον-πλαίσιο και το άρρηκτα συνδεδεμένο με αυτό, μοντέλο δέντρου στόχων, από την οπτική γωνία της δομής καθενός. Επίσης, στα κεφάλαια αυτά, οι δυο αυτές οντότητες εξετάστηκαν, ως δυο ξεχωριστά πράγματα χωρίς να αναφέρεται άμεσα πώς μπορούν να συνδυαστούν.

Έχοντας ορίσει στα προηγούμενα κεφάλαια την αρχιτεκτονική του περιβάλλοντος-πλαισίου αλλά και την δομή του μοντέλου δέντρου στόχων, στο κεφάλαιο αυτό αναλύονται, από λειτουργικής πλέον σκοπιάς, οι δυνατότητες που προσφέρουν στον πριν την χρήση έλεγχο SCA συστημάτων λογισμικού οι δυο αυτές οντότητες μαζί.

Στο κεφάλαιο αυτό, αρχικά θα οριστούν στρώματα, καθένα από τα οποία θα αναφέρεται σε συγκεκριμένο τύπο λειτουργικότητας που προσφέρει το περιβάλλον-πλαίσιο που ορίζεται. Τα στρώματα αυτά, τα οποία στο εξής θα καλούνται *Στρώματα Λειτουργικότητας (Function Layers)* θα παρουσιαστούν αναλυτικά στην αμέσως επόμενη παράγραφο. Εκεί θα δοθεί και η ένταξη κάθε δομοστοιχείου του περιβάλλοντος-πλαισίου σε ένα ή περισσότερα *Στρώματα Λειτουργικότητας*, καθώς ένα δομοστοιχείο μπορεί να μην ανήκει αποκλειστικά σε ένα *Στρώμα*.

Στην συνέχεια θα αναλυθούν τα *Στρώματα Λειτουργικότητας* ένα προς ένα. Σε κάθε στρώμα, θα δοθούν οι αλγόριθμοι οι οποίοι συμβάλλουν στην προσφορά της λειτουργικότητας του στρώματος. Οι αλγόριθμοι θα εξετασθούν αφενός από την σκοπιά της αρχιτεκτονικής του περιβάλλοντος-πλαισίου, δηλαδή ποιες ενέργειες εκτελούνται και ποια μηνύματα ανταλλάσσονται μεταξύ δομοστοιχείων του περιβάλλοντος-πλαισίου καθώς και η συγκεκριμένη χρονική ακολουθία αυτών. Επίσης, οι αλγόριθμοι θα εξετασθούν και από την σκοπιά του μοντέλου δέντρου στόχων και πώς αυτό χρησιμοποιείται από το περιβάλλον-πλαίσιο.

5.2 Στρώματα Λειτουργικότητας

Από την δομική σκοπιά, το περιβάλλον-πλαίσιο χωρίσθηκε σε πακέτα και δομοστοιχεία με βάση την ανάλυσή του που παρουσιάσθηκε στο κεφάλαιο 3. Στο κεφάλαιο αυτό, θα παρουσιασθεί από την λειτουργική του σκοπιά, οπότε θα πρέπει, αφού χωρίζεται με άλλα κριτήρια, να χωρισθεί και σε διαφορετικές οντότητες. Στο παρόν κεφάλαιο, λοιπόν, το περιβάλλον-πλαίσιο θα χωρισθεί σε *Στρώματα Λειτουργικότητας*. Ο όρος "λειτουργικότητα", αναφέρεται προφανώς ότι στο κεφάλαιο αυτό αναλύεται το δραστικό/λειτουργικό κομμάτι του περιβάλλοντος-πλαισίου. Ο όρος "στρώμα", χρησιμοποιείται καθώς δηλώνει ότι κάθε επίπεδο διαχωρισμού δεν είναι ανεξάρτητο από τα άλλα και ότι υφίσταται ένα είδος "χτισίματος" των επιπέδων το ένα πάνω στο άλλο, σαν μια πολυκατοικία όπου ο δεύτερος όροφος δεν θα στηριζόταν αν δεν υπήρχε το ισόγειο.

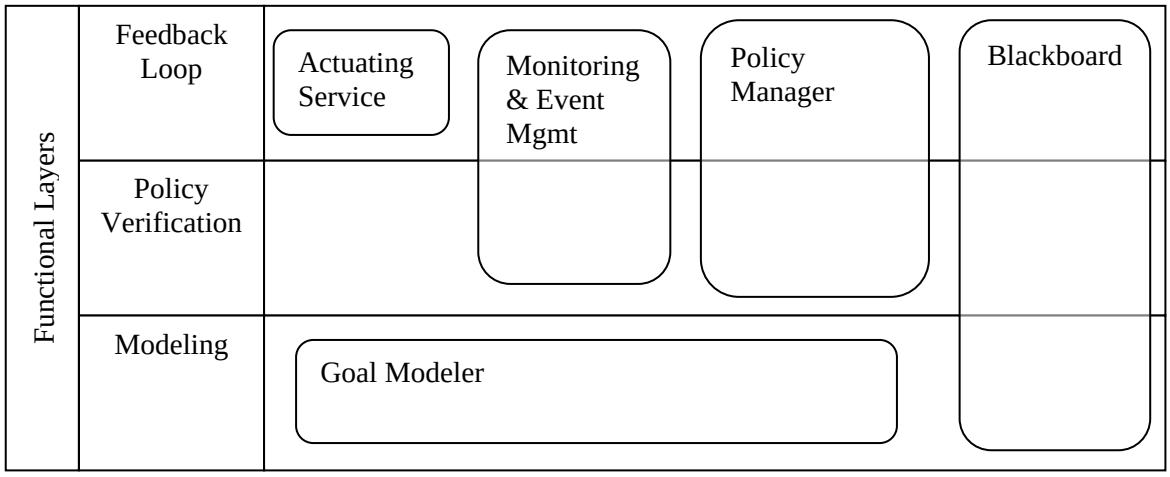
Στο περιβάλλον-πλαίσιο που παρουσιάζεται στην παρούσα διπλωματική εργασία, υφίστανται τα εξής *Στρώματα Λειτουργικότητας*.

- **Μοντελοποίηση.** Το στρώμα αυτό, περιλαμβάνει όλες τις λειτουργίες που έχουν να κάνουν με την δημιουργία του μοντέλου δέντρου στόχων για το υπό έλεγχο σύστημα λογισμικού.

- **Πολιτικές Επαλήθευσης.** Το στρώμα αυτό, περιλαμβάνει όλες εκείνες τις λειτουργίες που αφορούν στον τρόπο λειτουργίας των διάφορων πολιτικών επαλήθευσης των στόχων του Μοντέλου Δέντρου Στόχων.

- **Βρόχος Ανατροφοδότησης.** Το στρώμα αυτό, περιλαμβάνει όλες τις λειτουργίες που αφορούν στον τρόπο με τον οποίο προσαρμόζεται το περιβάλλον-πλαίσιο στις αλλαγές του υπό έλεγχο συστήματος λογισμικού, δηλαδή στο πώς κατά την εκτέλεση αλλάζουν οι κύριοι στόχοι (κάτι που οδηγεί και στην αλλαγή πολιτικών επαλήθευσης και το συνόλου των στόχων που πρέπει να επαληθευτούν, VS).

Μια οπτική αναπαράσταση των *Στρωμάτων Λειτουργικότητας* φαίνεται στο παρακάτω σχήμα.



Εικόνα 5-I: Στρώματα Λειτουργικότητας Περιβάλλοντος-Πλαισίου

Όπως φαίνεται και από το σχήμα, κατά τον διαχωρισμό, κάποια δομοστοιχεία της αρχιτεκτονικής του περιβάλλοντος-πλαισίου, δεν εντάσσονται αποκλειστικά σε κάποιο *Στρώμα Λειτουργικότητας*. Ο διαχωρισμός των δομοστοιχείων σε *Στρώματα Λειτουργικότητας*, εξηγείται παρακάτω.

- **Goal Modeler.** Το δομοστοιχείο αυτό είναι υπεύθυνο για την μοντελοποίηση του υπό έλεγχο συστήματος λογισμικού. Προφανώς η χρησιμότητα του δομοστοιχείου αυτού είναι στο στρώμα μοντελοποίησης, για αυτό και εντάσσεται εξ' ολοκλήρου στο Modeling Functional Layer.

- **Actuating Service.** Το δομοστοιχείο αυτό, επωμίζεται την λειτουργία του εντοπισμού των νέων κυρίων στόχων, ανάλογα με την κατάσταση λειτουργίας του υπό έλεγχο συστήματος λογισμικού. Επομένως, η λειτουργικότητά του περιορίζεται στο επίπεδο του βρόχου ανατροφοδότησης, για αυτό και εντάσσεται πλήρως στο Feedback Loop Functional Layer.

- **Blackboard.** Το δομοστοιχείο αυτό, προσφέρει την δυνατότητα επικοινωνίας μεταξύ των άλλων δομοστοιχείων αλλά και την παρακολούθηση της κατάστασης του περιβάλλοντος-πλαισίου (τρέχοντες στόχοι) και του υπό έλεγχο συστήματος λογισμικού (συμβάντα που έχουν παρατηρηθεί). Συνεπώς, το δομοστοιχείο αυτό εντάσσεται σε όλα τα Στρώματα Λειτουργικότητας του περιβάλλοντος-πλαισίου.

- **Policy Manager.** Το δομοστοιχείο αυτό, είναι υπεύθυνο για την προσαρμοστικότητα που δείχνει το περιβάλλον-πλαίσιο στις αλλαγές κατάστασης του υπό έλεγχο συστήματος λογισμικού. Αναγνωρίζει όλα τα δυνατά υποδέντρα του δέντρου στόχων που μπορούν να επαληθεύσουν τους τρέχοντες στόχους και φροντίζει ώστε να είναι ενεργοποιημένες οι

σωστές πολιτικές κάθε φορά, για την επαλήθευση ή μη των στόχων των υποδέντρων αυτών. Για αυτό, ο Διαχειριστής Πολιτικών εντάσσεται και στο στρώμα του βρόχου ανατροφοδότησης και στο στρώμα των πολιτικών επαλήθευσης των τρεχόντων στόχων.

- **Monitoring & Event Management.** Το δομοστοιχείο αυτό, αναλαμβάνει να λάβει και να επεξεργαστεί συμβάντα από το υπό έλεγχο σύστημα. Σε αυτά τα συμβάντα που αναλαμβάνει να συλλέξει, περιέχονται και συμβάντα τα οποία αλλάζουν τους τρέχοντες στόχους και συμβάντα τα οποία βοηθούν στην επαλήθευση τρεχόντων στόχων. Συνεπώς, το δομοστοιχείο αυτό εντάσσεται στα ίδια στρώματα λειτουργικότητας που εντάσσεται και ο Διαχειριστής Πολιτικών.

Με την ανάλυση και του διαχωρισμού των δομοστοιχείων σε Στρώματα Λειτουργικότητας, η συνέχεια του κεφαλαίου αυτού, περιλαμβάνει την ανάλυση των αλγορίθμων που χρησιμοποιούνται σε κάθε Στρώμα Λειτουργικότητας.

5.3 Στρώμα Μοντελοποίησης

Το στρώμα μοντελοποίησης, αναφέρεται στην λειτουργία που πρέπει να επιτελεί το περιβάλλον-πλαίσιο στην μοντελοποίηση του υπό έλεγχο συστήματος λογισμικού στο μοντέλο δέντρου στόχων που έχει παρουσιασθεί. Στην παράγραφο αυτή, θα παρουσιαστούν οι βασικοί αλγόριθμοι που εξάγουν από ένα υπηρεσιοκεντρικό σύστημα μοντελοποιημένο με SCA το Δέντρο Στόχων και τις Επισημάνσεις που αυτό χρειάζεται.

5.3.1 Σύμβαση Σημειογραφίας

Για την διευκόλυνση της περαιτέρω ανάλυσης και εξήγησης των αλγορίθμων που χρησιμοποιούνται κατά την μοντελοποίηση SCA συστημάτων λογισμικού, κρίνεται απαραίτητο να δοθεί μια σημειογραφία η οποία θα τηρείται στην παρακάτω ανάλυση.

Αρχικά, ορίζονται δυο βοηθητικοί στόχοι. Ο στόχος $\mathbf{G_T}$ για τον οποίο θεωρείται ότι η διαδικασία επαλήθευσής του δίνει πάντοτε αληθές και ο στόχος $\mathbf{G_F}$ για τον οποίο θεωρείται ότι η διαδικασία επαλήθευσής του δίνει πάντοτε ψευδές. Αυτοί οι δυο στόχοι θα βοηθήσουν ιδιαιτέρως στην διαδικασία δημιουργίας της δομής του δέντρου στόχων είτε στην αρχή, είτε κατά τις αλλαγές κατάστασης στο υπό έλεγχο σύστημα λογισμικού, ώστε να διατηρηθεί η σημασιολογία και η πληρότητα της δομής ενός AND/OR δέντρου (παραδείγματος χάριν, να μην υπάρχει στόχος που να έχει μόνο ένα υποστόχο).

Ο στόχος σε ένα δέντρο στόχων, εκπροσωπεί για οποιοδήποτε υπηρεσιοκεντρικό σύστημα, μια υπηρεσία. Η σημασιολογική εξήγηση για αυτήν την απόφαση είναι προφανής. Ένα υπηρεσιοκεντρικό σύστημα, προσφέρει υπηρεσίες στο περιβάλλον του, οπότε η λειτουργικότητά του ελέγχεται με το αν οι υπηρεσίες που (δηλώνει ότι) προσφέρει, όντως προσφέρονται. Άρα, η λειτουργία του υπηρεσιοκεντρικού συστήματος επαληθεύεται, αν επαληθευτεί η διαθεσιμότητα όλων των υπηρεσιών που αυτό προσφέρει. Για την καλύτερη και πιο δομημένη παρουσίαση του Δέντρου Στόχων, ακολουθούμε την παρακάτω σύμβαση.

- **Υπηρεσία (Στόχος).** Κάθε υπηρεσία που προσφέρεται στο υπηρεσιοκεντρικό σύστημα λογισμικού, αποτελεί και έναν στόχο στο δέντρο στόχων. Στην σημειογραφία κάθε υπηρεσία του συστήματος που ελέγχεται, πρέπει να αναφέρεται το δομοστοιχείο ή το σύνθετο δομοστοιχείο (Composite) το οποίο την προσφέρει καθώς και το όνομα που έχει αποδοθεί στην υπηρεσία, κατά το στάδιο της μοντελοποίησης του υπηρεσιοκεντρικού συστήματος. Ορίζεται, συνεπώς, μια υπηρεσία στο δομοστοιχείο C, με όνομα N, να γράφεται για τους σκοπούς της παρούσας διπλωματικής ως S_N^C. Κάθε στόχος στο μοντέλο στόχων, έχει ένα όνομα (Name) και έναν αναγνωριστικό αριθμό (ID). Ο αριθμός δίνεται από το δομοστοιχείο Goal Modeler της αρχιτεκτονικής του περιβάλλοντος-πλαισίου, ενώ το όνομα του στόχου μορφώνεται με τον κανόνα: (*g<ID>: <ServiceNotation>*). Ο κανόνας αυτός δηλώνει πως αρχικά τίθεται το γράμμα g, έπειτα ο αναγνωριστικός αριθμός του στόχου, έπειτα ένας χαρακτήρας άνω-κάτω τελείας και ένα κενό ακολουθούμενα από το όνομα της υπηρεσίας που αντιστοιχεί στον στόχο. Το όνομα της υπηρεσίας, μορφώνεται όπως αναφέρθηκε προηγουμένως, αλλά για λόγους χρήσης του σε προγράμματα (εκτύπωση σε κονσόλα), το όνομα της υπηρεσίας του παραδείγματος θα γράφεται ως: S_N^C. Ένα παράδειγμα, έστω ότι έχουμε την υπηρεσία *CheckCard* του δομοστοιχείου *ATM*, και ότι ο αντίστοιχος στόχος όταν είναι να μοντελοποιηθεί στο Goal Modeler δομοστοιχείο, λαμβάνει τον αναγνωριστικό αριθμό 5. Το όνομα του στόχου αυτού, θα είναι: (g5: $S_{CheckCard}^{ATM}$). Αυτή θα είναι και η αλφαριθμητική απεικόνιση του στόχου και στο δέντρο στόχων και κατά την εκτέλεση του περιβάλλοντος πλαισίου.

- **Αναφορά (Reference).** Οι αναφορές σε ένα SCA μοντέλο, δεν χρησιμοποιούνται κατά την δημιουργία του αντίστοιχου Δέντρου Στόχων. Για την πληρότητα όμως της σημειογραφίας που παρουσιάζεται, μια αναφορά ορίζεται να αναπαρίσταται με ίδιο τρόπο όπως και η υπηρεσία, με χρήση του γράμματος R (από το Reference), αντί του S. Συνεπώς, μια αναφορά με όνομα name, στο δομοστοιχείο (σύνθετο, ή μη) comp, θα σημειώνεται ως R_{name}^{comp}.

- **Δομοστοιχεία.** Τα δομοστοιχεία ενός συστήματος λογισμικού που έχει μοντελοποιηθεί ως SCA, θα σημειώνονται για τις ανάγκες της παρούσας διπλωματικής, με το όνομα που τους έχει αποδοθεί στο SCA μοντέλο. Επειδή κάθε δομοστοιχείο, ανήκει σε ένα σύνθετο δομοστοιχείο, η σημειογραφία για τα δομοστοιχεία, περιλαμβάνει μπροστά από το όνομά τους το όνομα του σύνθετου δομοστοιχείου που ανήκουν, ακολουθούμενα από μια τελεία και τέλος το όνομα του δομοστοιχείου. Η πληροφορία αυτή, χρησιμοποιείται στο μοντέλο δέντρου στόχων, ως ο εκθέτης στην σημειογραφία των υπηρεσιών – στόχων.

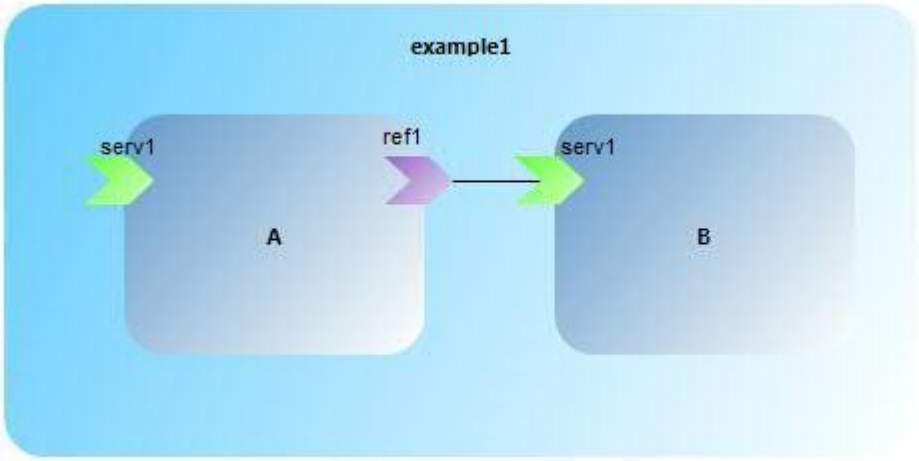
- **Σύνδεση (Wire).** Στο μοντέλο SCA, οι συνδέσεις χρησιμοποιούνται για να δηλώσουν ποια αναφορά χρησιμοποιεί ποια υπηρεσία. Εκφράζουν μια σχέση από μια αναφορά προς μια υπηρεσία. Μια σύνδεση, μεταξύ της αναφοράς $R_{refname}^{refcomp}$ και της υπηρεσίας $S_{servname}^{servcomp}$ θα σημειώνεται ως ${}_{refname}^{refcomp} W_{servname}^{servcomp}$. Η πληροφορία αυτή, δεν χρησιμοποιείται στο Μοντέλο Δέντρου Στόχων και η σημειογραφία της ορίζεται εδώ, για πληρότητα.

Με τους παραπάνω ορισμούς, καθορίζεται πλήρως η σημειογραφία που είναι αναγκαία για την αναφορά σε δομήματα του SCA μοντέλου που θα χρησιμοποιηθούν εκτενώς στην συνέχεια, για την επεξήγηση των αλγορίθμων χειρισμού SCA μοντέλων, αλλά και του μοντέλου δέντρου στόχων.

5.3.2 Εξαγωγή Δέντρου Στόχων από SCA Μοντέλο

Η παράγραφος αυτή, θα δώσει τον ακριβή αλγόριθμο με τον οποίο γίνεται η εξαγωγή του δέντρου στόχων, από ένα δεδομένο μοντέλο περιγραφής ενός υπηρεσιοκεντρικού συστήματος λογισμικού, σε SCA. Για την μέγιστη δυνατή πληρότητα και ανάλυση του αλγορίθμου, τα βασικά του βήματα, θα παρουσιασθούν μέσω ορισμένων απλούστατων περιπτώσεων SCA μοντέλου και του αντίστοιχου Δέντρου Στόχων που τις μοντελοποιεί. Αυτές οι αντιστοιχήσεις, παραδίδουν τους κανόνες με τους οποίους δουλεύει ο αλγόριθμος εξαγωγής δέντρου στόχων από το μοντέλο SCA.

5.3.2.1 Μια Σύνδεση

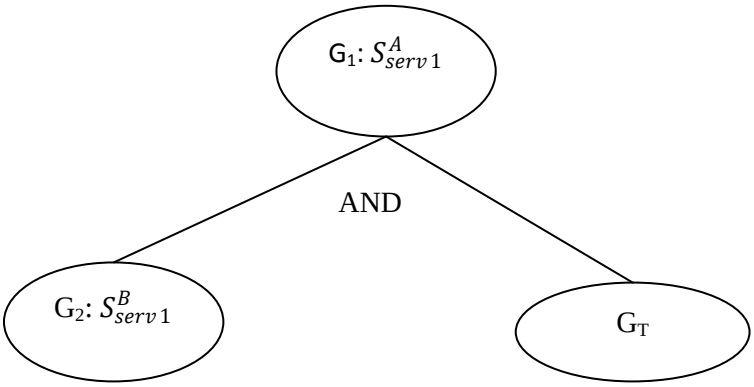


Εικόνα 5-II: Παράδειγμα SCA - Μια σύνδεση

Το παράδειγμα του SCA συστήματος που παρουσιάζεται στην Εικόνα 5-II είναι το απλούστερο που μπορεί να υπάρξει και αναφέρεται στην περίπτωση δυο δομοστοιχείων (Components) A και B. Το δομοστοιχείο A προσφέρει στο περιβάλλον του την υπηρεσία S^{A}_{serv1} και χρησιμοποιεί από το περιβάλλον του την αναφορά R^{A}_{ref1}. Το δομοστοιχείο B, προσφέρει στο περιβάλλον του την υπηρεσία S^{B}_{serv1}, ενώ δεν χρειάζεται την χρήση κάποιας υπηρεσίας ώστε να προσφέρει την λειτουργικότητά του. Η υπηρεσία S^{B}_{serv1}, ικανοποιεί την αναφορά R^{A}_{ref1}, μέσω της αντίστοιχης σύνδεσης.

Εξετάζοντας την σημασιολογία του παραδείγματος, βλέπουμε ότι το Δομοστοιχείο A εξαρτάται από την υπηρεσία που προσφέρει το Δομοστοιχείο B, ώστε να προσφέρει την δική του υπηρεσία στο περιβάλλον του. Αντιθέτως, το Δομοστοιχείο B, μπορεί και προσφέρει την υπηρεσία S^{B}_{serv1}, χωρίς να χρειάζεται να χρησιμοποιήσει υπηρεσία από κάποιο άλλο δομοστοιχείο. Ανάγοντας αυτές τις παρατηρήσεις, στο μοντέλο δέντρου στόχων που παρουσιάσθηκε στο Κεφάλαιο 4, εύκολα γίνεται κατανοητό ότι ο στόχος που θα αντιστοιχεί στην υπηρεσία S^{A}_{serv1} είναι ένας στόχος που μπορεί να αναλυθεί σε AND/OR αποσυνθέσεις (DecompositionGoal), ενώ ο στόχος που θα αντιστοιχεί στην υπηρεσία S^{B}_{serv1}, είναι ατομικός (AtomicGoal), καθώς το δομοστοιχείο που προσφέρει την υπηρεσία, δεν χρησιμοποιεί κάποια αναφορά που να δηλώνει εξάρτηση από υπηρεσίες τρίτων Δομοστοιχείων. Συνεπώς, στο δέντρο στόχων που θα δημιουργηθεί, με αυθαίρετη επιλογή των αναγνωριστικών αριθμών **ID** που δίνονται σε κάθε στόχο, έχουμε τους δυο στόχους (g1: S^{A}_{serv1}) και (g2: S^{B}_{serv1}). Ο στόχος g2, είναι ατομικός στόχος όπως έχουμε πει, ενώ ο στόχος g1, έχει εξάρτηση από τον στόχο g2. Ορίζεται η περίπτωση μιας εξάρτησης, όπως στο παράδειγμά που παρουσιάζεται,

να μοντελοποιείται στο Δέντρο Στόχων, ως μια AND αποσύνθεση, όπως στο δέντρο στόχων της παρακάτω εικόνας.



Εικόνα 5-III: Δέντρο Στόχων για 1ο Παράδειγμα SCA

Πρακτικά, το Δέντρο Στόχων στην Εικόνα 5-III, δηλώνει την εξάρτηση της υπηρεσίας που προσφέρει το Δομοστοιχείο A, από την υπηρεσία που προσφέρει το Δομοστοιχείο B. Η εισαγωγή του εικονικού στόχου $\mathbf{G_T}$ γίνεται για να διατηρηθεί η σημασιολογία της AND αποσύνθεσης του στόχου g1, αλλά και αν διατηρηθεί η γενικότητα που θα μετατρέψει το Δέντρο Στόχων αυτό, σε έναν από τους βασικούς κανόνες οι οποίοι θα οδηγούν στα δέντρα στόχων για μεγαλύτερα και πληρέστερα συστήματα SCA.

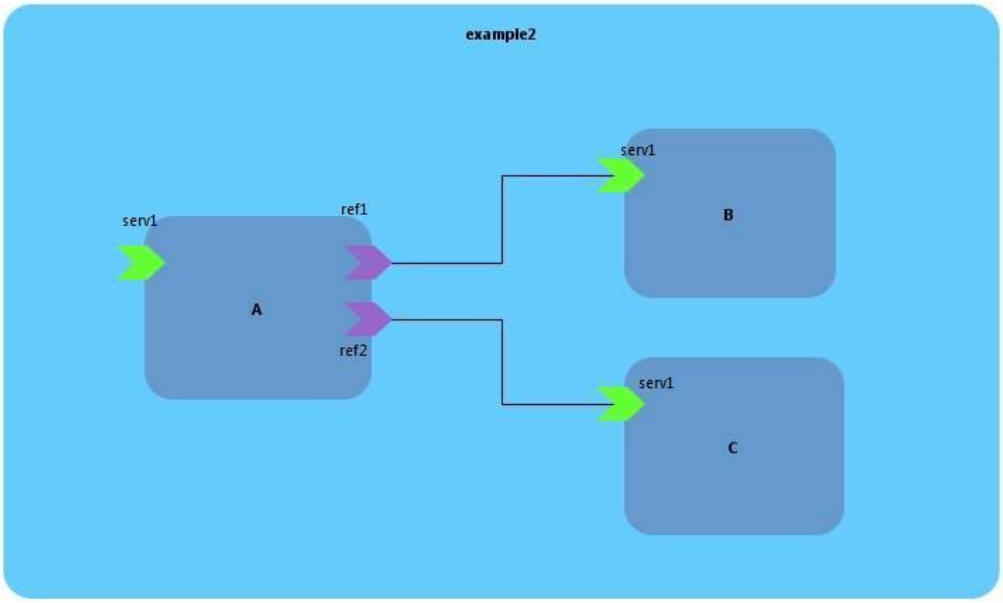
Από το παράδειγμα αυτό, προέκυψαν τα εξής βασικά σημεία:

- Κάθε υπηρεσία που ανήκει σε Δομοστοιχείο (σύνθετο ή μη) το οποίο δεν περιέχει αναφορές σε άλλες υπηρεσίες, αντιστοιχεί σε έναν ατομικό στόχο (AtomicGoal) στο Δέντρο Στόχων.

- Κάθε υπηρεσία που ανήκει σε Δομοστοιχείο (σύνθετο ή μη) το οποίο περιέχει αναφορές σε άλλες υπηρεσίες, αντιστοιχεί σε έναν στόχο που αναλύεται σε AND/OR αποσυνθέσεις στόχων (DecompositionGoal). Η μεθοδολογία με την οποία γίνεται αυτή η ανάλυση, θα παρουσιαστεί στις επόμενες παραγράφους.

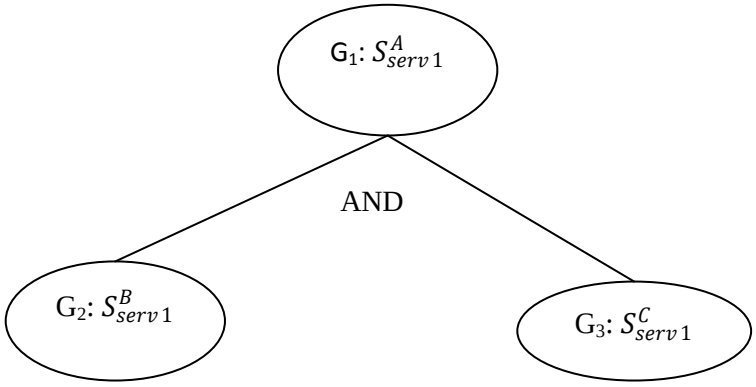
- Οι εικονικοί στόχοι $\mathbf{G_T}$ και $\mathbf{G_F}$ ορίσθηκαν ώστε να χρησιμοποιούνται για την διατήρηση της γενικότητας και της σημασιολογίας του μοντέλου δέντρου στόχων. Αυτό οφείλει να είναι οδηγός και για όσους χρήστες θέλουν να επεκτείνουν το περιβάλλον-πλαίσιο που παρουσιάζεται, ώστε να μοντελοποιήσουν άλλα συστήματα (πέραν των SCA) στο μοντέλο δέντρων στόχων.

5.3.2.2 2-σε-2 Σύνδεση



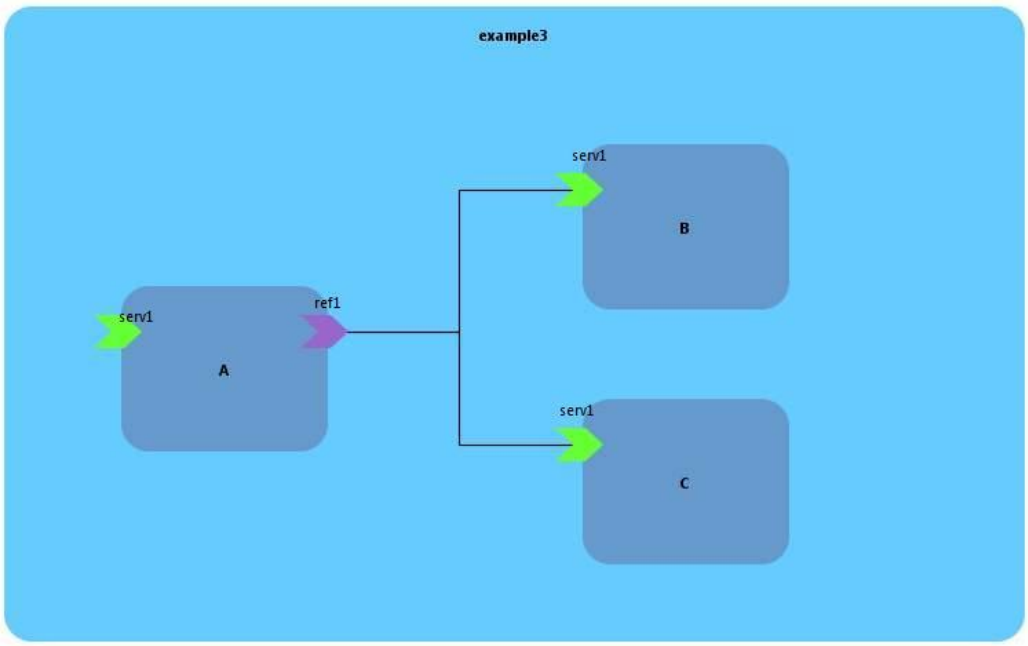
Εικόνα 5-IV: Παράδειγμα SCA, 2-σε-2 σύνδεση

Στο παράδειγμα αυτό, το Δομοστοιχείο Α, από το προηγούμενο παράδειγμα, εκτός από την αναφορά $R^A_{ref\,1}$ έχει και την αναφορά $R^A_{ref\,2}$ η οποία δηλώνει εξάρτηση από την υπηρεσία $S^C_{serv\,1}$, που την προσφέρει το καινούριο Δομοστοιχείο C. Σημασιολογικά, το SCA διάγραμμα στην Εικόνα 5-IV δηλώνει ότι για να μπορέσει το Δομοστοιχείο Α να προσφέρει την υπηρεσία $S^A_{serv\,1}$, χρειάζεται (στην συνηθέστερη και πιο γενική περίπτωση) να λάβει τα αποτελέσματα των υπηρεσιών $S^B_{serv\,1}$ και $S^C_{serv\,1}$. Όσον αφορά στο μοντέλο δέντρου στόχων, είναι προφανής η ομοιότητα με το προηγούμενο παράδειγμα, με την διαφορά ότι δεν χρειάζεται να ορισθεί εικονικός στόχος για να συμπληρώσει την γενικότητα. Το αντίστοιχο Δέντρου Στόχων είναι:



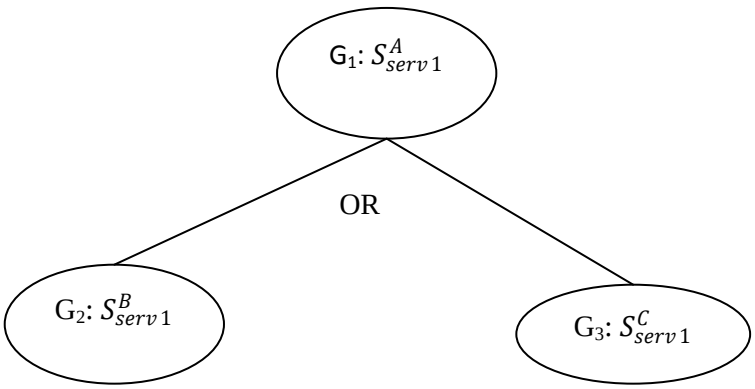
Εικόνα 5-V: Δέντρο Στόχων για 2ο παράδειγμα SCA

5.3.2.3 1-σε-2 Σύνδεση



Εικόνα 5-VI: Παράδειγμα SCA, 1-σε-2 Σύνδεση

Στο παράδειγμα αυτό, η αναφορά $R^{A}_{ref\,1}$, έχει πολλαπλότητα 2 (ιδιότητα multiplicity του Reference Element στο αντίστοιχη xml περιγραφή) και συνδέεται με δυο παρόμοιες υπηρεσίες, $S^{B}_{serv\,1}$ και $S^{C}_{serv\,1}$, οι οποίες έχουν την δυνατότητα να ικανοποιήσουν την αναφορά. Μια τέτοια δομή στην μοντελοποίηση ενός υπηρεσιοκεντρικού συστήματος λογισμικού, σημαίνει ότι η αναφορά μπορεί να ικανοποιηθεί με κλήσεις είτε στην μια είτε στην άλλη υπηρεσία. Η αναγωγή στο σημασιολογικό αντίστοιχο του μοντέλου δέντρου στόχων, δίνει ότι ο στόχος που μοντελοποιεί την υπηρεσία $S^{A}_{serv\,1}$ θα είναι τύπου DecompositionGoal και θα αναλύεται με OR αποσύνθεση στους στόχους που μοντελοποιούν τις υπηρεσίες $S^{B}_{sev\,1}$ και $S^{C}_{serv\,1}$. Το αντίστοιχο Δέντρο Στόχων είναι:



Εικόνα 5-VII: Δέντρο Στόχων για 3ο Παράδειγμα SCA

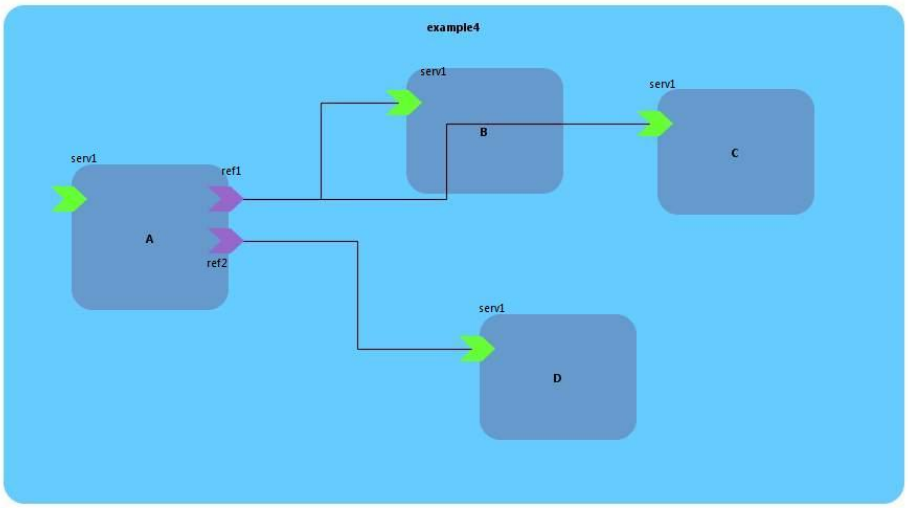
Από τα δυο προηγούμενα παραδείγματα, προκύπτουν οι δυο βασικότεροι κανόνες κατασκευής του δέντρου στόχων, από την περιγραφή ενός SCA συστήματος λογισμικού. Οι κανόνες αυτοί, αναφέρονται στον εντοπισμό του είδους της ανάλυσης του στόχου σε AND ή OR αποσύνθεση. Πιο συγκεκριμένα:

- **ORDecomposition.** Ο στόχος που αντιστοιχεί σε μια υπηρεσία, είναι είδους OR αποσύνθεσης, όταν το δομοστοιχείο που περιέχει την υπηρεσία έχει μια αναφορά με πολλαπλότητα μεγαλύτερη του μηδενός (συνδέεται με περισσότερες του ενός υπηρεσίες).
- **ANDDecomposition.** Ο στόχος που αντιστοιχεί σε μια υπηρεσία, είναι είδους AND αποσύνθεσης, όταν το δομοστοιχείο που περιέχει την υπηρεσία έχει πολλές αναφορές με πολλαπλότητα 1 (καθεμιά συνδέεται σε μια υπηρεσία).

Αυτοί είναι οι βασικοί κανόνες για τον εντοπισμό του είδους της αποσύνθεσης ενός DecompositionGoal. Το μοντέλο για την περιγραφή υπηρεσιοκεντρικών συστημάτων σε SCA, δεν περιορίζεται στις δυο παραπάνω περιπτώσεις. Το επόμενο παράδειγμα, αναφέρεται σε μεικτές συνδέσεις.

5.3.2.4 Μεικτές Συνδέσεις

Οι παράγραφοι 5.3.2.2 και 5.3.2.3 εξέφρασαν τους κανόνες που λειτουργούν στις απλές περιπτώσεις όπου το περιέχον δομοστοιχείο της υπηρεσίας έχει είτε αναφορές πολλαπλότητας ένα, είτε μια αναφορά μεγαλύτερης πολλαπλότητας. Τώρα εξετάζουμε μια μεικτή περίπτωση, όπου το δομοστοιχείο περιέχει συνδυασμούς των ανωτέρω περιπτώσεων.

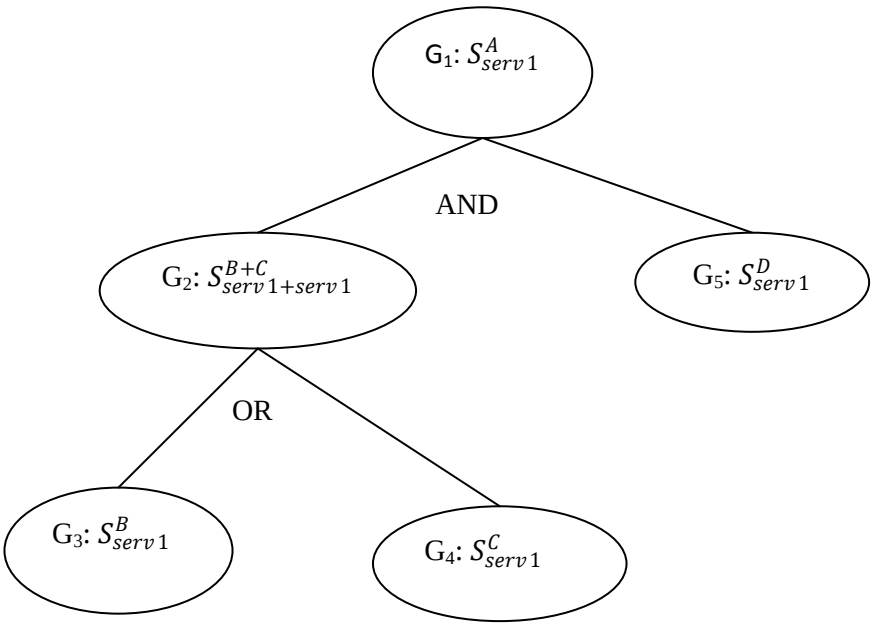


Εικόνα 5-VIII: Παράδειγμα SCA, μεικτές συνδέσεις

Σημασιολογικά, το παράδειγμα στην Εικόνα 5-VIII δηλώνει πως στην γενική περίπτωση, η προσφορά της υπηρεσίας $S^A_{serv\,1}$, εξαρτάται από τις δυο αναφορές που έχει το δομοστοιχείο Α, τις $R^A_{ref\,1}$ και $R^A_{ref\,2}$. Από τις δυο αυτές αναφορές, η πρώτη βλέπουμε να συνδέεται με δυο υπηρεσίες, τις $S^B_{serv\,1}$ και $S^C_{serv\,1}$ ενώ η δεύτερη αναφορά συνδέεται με την υπηρεσία $S^D_{serv\,1}$. Το συμπέρασμα που βγαίνει, είναι πως για την προσφορά της υπηρεσίας $S^A_{serv\,1}$ είναι αναγκαία η σωστή προσφορά μιας υπηρεσίας εκ των $S^B_{serv\,1}$, $S^C_{serv\,1}$ *και* της υπηρεσίας $S^A_{serv\,1}$. Η εξάρτηση της υπηρεσίας $S^A_{serv\,1}$ από τις άλλες τρεις, είναι μεικτού τύπου καθώς περιλαμβάνει και *OR* και *AND* μέρος. Στο Κεφάλαιο 4, όμως, στον ορισμό του Μοντέλου Δέντρου Στόχων δηλώθηκε σαφέστατα ότι ένας στόχος που αναλύεται σε υποστόχους, μπορεί να το κάνει αναλυόμενος είτε αποκλειστικά σε OR είτε αποκλειστικά σε AND αποσύνθεση.

Για την μοντελοποίηση της μεικτής εξάρτησης του παραδείγματός αυτού, ορίζουμε έναν εικονικό στόχο, ο οποίος μπορεί να μοντελοποιεί την περίπτωση εξαρτήσεων μιας υπηρεσίας από μια αναφορά πολλαπλότητας μεγαλύτερης του ένα. Ο στόχος αυτός, θα μοντελοποιεί περιπτώσεις, όπου το δομοστοιχείο που περιέχει την αναφορά του στόχου πατέρα του, έχει μια αναφορά που μπορεί να ικανοποιηθεί από περισσότερες της μιας υπηρεσίες. Επίσης, στην περίπτωση όπου η αναφορά πολλαπλότητας μεγαλύτερης του ένα, είναι μοναδική στο δομοστοιχείο, τότε δεν χρειάζεται η εισαγωγή του επιπλέον εικονικού στόχου. Ένα τέτοιο παράδειγμα αναφέρεται στο παράδειγμα της παραγράφου 5.3.2.3, όπου δεν χρειάσθηκε η εισαγωγή του εικονικού στόχου, λόγω της απλότητας του παραδείγματος.

Η ονοματολογία του εικονικού στόχου που ορίζεται στην παράγραφο αυτή, ορίζεται να είναι η εξής: $S^{C1+C2}_{name\,1+name\,2}$, όπου *name1* είναι το όνομα της πρώτης υπηρεσίας, *name2* είναι το όνομα της δεύτερης υπηρεσίας και *C1, C2* τα δομοστοιχεία στα οποία ανήκουν, αντίστοιχα. Να σημειωθεί ότι το σύμβολο '+' στην συγκεκριμένη περίπτωση συμβολίζει τον λογικό τελεστή OR. Το δέντρο στόχων για αυτό το παράδειγμα είναι:



Εικόνα 5-IX: Δέντρο Στόχων για 4ο Παράδειγμα SCA

Τα παραδείγματα των τριών προηγούμενων παραγράφων, δίνουν τους βασικότερους κανόνες με τους οποίους μορφώνεται το δέντρο στόχων από ένα την περιγραφή της μοντελοποίησης ενός συστήματος λογισμικού SCA. Ο γενικός κανόνας δημιουργίας του δέντρου στόχων που εισάγεται στην παρούσα παράγραφο, είναι ο εξής:

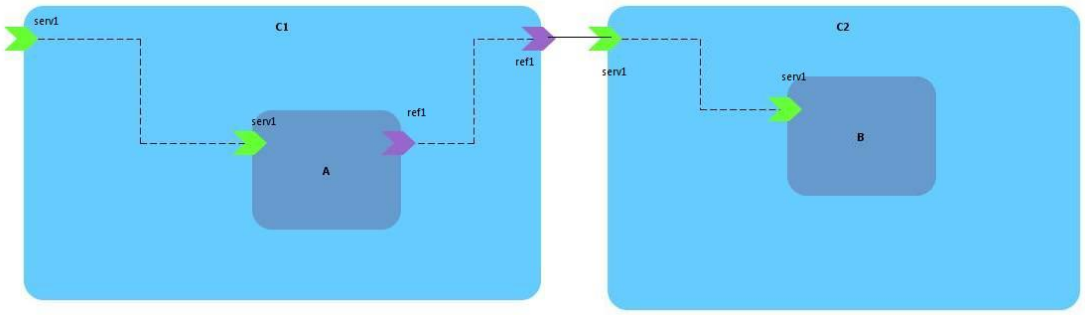
1. Ο στόχος – υπηρεσία που ανήκει σε δομοστοιχείο που περιέχει αναφορές σε άλλους στόχους-υπηρεσίες, μοντελοποιείται ως DecompositionGoal και μάλιστα τύπου KAI (ANDDecomposition). Στις αποσυνθέσεις του, οι στόχοι υπηρεσίες – πέρατα των αναφορών πολλαπλότητας 1, εισάγονται αυτόματα αφού ο αλγόριθμος εξαγωγής του δέντρου στόχων τρέξει και για αυτούς, ενώ σε αυτές αναφορές πολλαπλότητας μεγαλύτερης του 1, εισάγεται πρώτα ένας εικονικός στόχος τύπου ORDecomposition και ως αποσυνθέσεις αυτού, τα δέντρα στόχων κάθε υπηρεσίας στόχου – πέρατος.

2. Ο στόχος – υπηρεσία που ανήκει σε δομοστοιχείο που δεν περιέχει αναφορές σε άλλους στόχους – υπηρεσίες, μοντελοποιείται ως AtomicGoal, αφού δεν έχει εξαρτήσεις από υπηρεσίες που προσφέρονται από άλλα δομοστοιχεία.

Οι δυο παραπάνω κανόνες, αναφέρονται στα ενδότερα ενός Σύνθετου Δομοστοιχείου (Composite) και όπως φάνηκε στα προηγούμενα παραδείγματα, η ένταξη των δομοστοιχείων σε Σύνθετα Δομοστοιχεία, δεν είχε αντίκτυπο στον αλγόριθμο που παρουσιάσθηκαν. Στην παράγραφο που ακολουθεί, γενικεύεται ο παραπάνω αλγόριθμος, για τις περιπτώσεις όπου έχουμε περισσότερα του ενός Σύνθετα Δομοστοιχεία στο SCA σύστημα λογισμικού, καθώς αναφορές και υπηρεσίες που

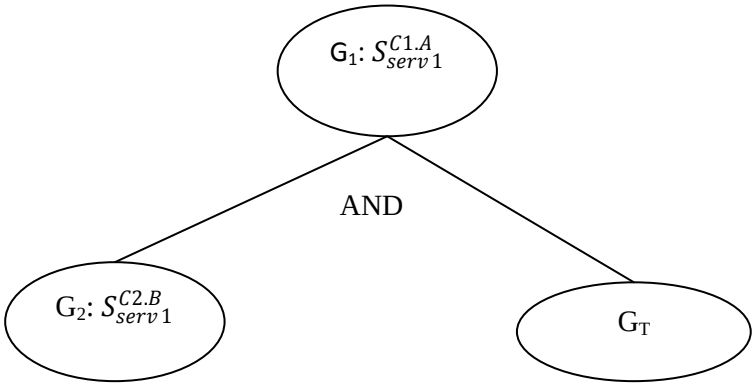
προάγονται (promote) από υπηρεσίες – αναφορές του δομοστοιχείου σε αναφορές – υπηρεσίες του Σύνθετου Δομοστοιχείου.

5.3.2.5 Μία σύνδεση, μεταξύ Σύνθετων Δομοστοιχείων



Εικόνα 5-X: Μία σύνδεση μεταξύ Σύνθετων Δομοστοιχείων

Το παράδειγμα στην Εικόνα 5-X είναι το ίδιο με το παράδειγμα της παραγράφου 5.3.2.1, μόνο που η μια σύνδεση που παρουσιάζεται, εμφανίζεται μεταξύ των προηγμένων αναφορών και υπηρεσιών δυο Σύνθετων Δομοστοιχείων και όχι μεταξύ δομοστοιχείων που ανήκουν στο ίδιο Σύνθετο Δομοστοιχείο. Από την άποψη της λειτουργικότητας που προσφέρεται και απαιτείται από τα σύνθετα ή μη δομοστοιχεία των SCA συστημάτων λογισμικού, η περίπτωση της παραγράφου αυτής, με το παράδειγμα της παραγράφου 5.3.2.1 είναι πανομοιότυπα. Η σημασιολογία του παραδείγματος της παραγράφου αυτής, δηλώνει ότι η υπηρεσία $S_{serv1}^{C1.A}$, η οποία προάγεται στην υπηρεσία S_{serv1}^{C1} εξαρτάται, μέσω της αναφοράς $R_{ref1}^{C1.A}$, η οποία προάγεται στην αναφορά R_{ref1}^{C1}, από την υπηρεσία $S_{serv1}^{C2.B}$ η οποία προάγεται στην υπηρεσία S_{serv1}^{C2}. Με λίγα λόγια, οι εξαρτήσεις παρακάμπτουν τις προαγωγές των αναφορών και υπηρεσιών από αυτές του Σύνθετου Δομοστοιχείου σε αυτές του Δομοστοιχείου και είναι πανομοιότυπες με τις εξαρτήσεις που υπήρχαν στο παράδειγμα της παραγράφου 5.3.2.1. Το μοντέλο Δέντρου Στόχων, είναι το εξής:



Εικόνα 5-XI: Δέντρο Στόχων για 5ο παράδειγμα SCA

Παρατηρούμε πως η Εικόνα 5-XI καθώς και Εικόνα 5-III παρουσιάζουν δομικά ένα ίδιο δέντρο στόχων, με την μόνη διαφορά ότι στην πρώτη εικόνα, ο στόχος πατέρας και ο στόχος παιδί αναφέρονται σε υπηρεσίες που εκτός από διαφορετικά δομοστοιχεία, ανήκουν και σε διαφορετικά σύνθετα δομοστοιχεία.

Η δομική ομοιότητα των δυο δέντρων στόχων, εξηγείται από το γεγονός πώς η δομή του δέντρου στόχων, έχει ως σκοπό να εκφράσει σημασιολογικά και δομικά τις εξαρτήσεις μεταξύ των υπηρεσιών που εμφανίζονται στο δεδομένο SCA μοντέλο του υπό έλεγχο συστήματος λογισμικού. Όπως εξηγήθηκε προηγουμένως, σημασιολογικά οι εξαρτήσεις του παραδείγματος της παραγράφου 5.3.2.1 και της παρούσας παραγράφου, είναι απολύτως ίδιες.

Από το παράδειγμα της παραγράφου αυτής, βλέπουμε πως το δέντρο στόχων για παραδείγματα που περιλαμβάνουν περισσότερα του ενός Σύνθετα Δομοστοιχεία, εξάγεται μέσω των κανόνων που παρουσιάσθηκαν στις προηγούμενες παραγράφους, αν οι προηγμένες υπηρεσίες/αναφορές ανά Σύνθετο Δομοστοιχείο βραχυκυκλωθούν με τις αντίστοιχες υπηρεσίες/αναφορές στο Δομοστοιχείο. Με τον όρο βραχυκυκλωθούν, εννοείται ότι μορφώνεται ένας στόχος, για την περίπτωση των προηγμένων υπηρεσιών και ότι στην περίπτωση των προηγμένων αναφορών, μορφώνεται ένας στόχος που είναι κοινός για την υπηρεσία πέρας της προηγμένης αναφοράς. Η υπηρεσία πέρας, θα έχει και αυτή μια μορφή προαγωγής από Δομοστοιχείο σε Σύνθετο Δομοστοιχείο, στο εσωτερικό του Σύνθετου Δομοστοιχείου που αποτελεί πέρας της προηγμένης αναφοράς. Αντίστοιχα παραδείγματα με 1-σε-2 και 2-σε-2 συνδέσεις μεταξύ προηγμένων αναφορών και υπηρεσιών διαφορετικών Σύνθετων Δομοστοιχείων, εμφανίζουν ομοιότητες με τις αντίστοιχες περιπτώσεις Δομοστοιχείων που ανήκουν σε ένα Σύνθετο Δομοστοιχείο και για αυτό δεν μελετώνται εκτενέστερα.

5.4 Στρώμα Πολιτικών Επαλήθευσης

Στο στρώμα πολιτικών επαλήθευσης, ανήκουν οι λειτουργίες του περιβάλλοντος-πλαισίου, οι οποίες αναφέρονται στην επαλήθευση των επισημειώσεων που έχουν τεθεί ανά στόχο, αλλά και την επαλήθευση του συνόλου VS, του συνόλου των διαφορετικών συνόλων επαλήθευσης που δίνουν την επαλήθευση των τρεχόντων στόχων. Αναλύουμε τις δυο αυτές ξεχωριστές περιπτώσεις στις επόμενες παραγράφους.

5.4.1 Επαλήθευση Επαληθεύσιμων Επισημειώσεων

Επαληθεύσιμες επισημειώσεις είναι εκείνες που ανάλογα με την κατάσταση του στόχου στον οποίο είναι συνημμένες, μπορούν να έχουν τιμή αληθής ή ψευδής. Στο κεφάλαιο 4, έγινε σαφές ότι για SCA συστήματα λογισμικού, τέτοιες επισημειώσεις είναι οι *προθέσεις* που αναφέρονται σε κάθε υπηρεσία ότι είναι απαραίτητες για την διατήρηση κάποιων περιορισμών, κυρίως τύπου ποιότητας υπηρεσίας (QoS). Ο ορισμός των Επαληθεύσιμων Επισημάνσεων και των Πολιτικών Επαλήθευσης που αναφέρονται σε αυτές, βρίσκονται στα διαγράμματα τομέων στην Εικόνα 4-IV και στην Εικόνα 4-VI, αντίστοιχα. Ο παρακάτω πίνακας, συνοψίζει τις ενέργειες που πρέπει να γίνουν, για την επαλήθευση επισήμανσης πρόθεσης (intentAnnotation).

intentAnnotation	intentMap	Method
Requires	Provides	Intersection
Constraint	appliesTo	Xpath matching
Qualification.name	Qualifier.Name	String matching

Η πρώτη στήλη του παραπάνω πίνακα, δηλώνει την ιδιότητα μιας επισήμανσης τύπου intentAnnotation που μπορεί να επαληθευτεί. Η δεύτερη στήλη, δηλώνει την αντίστοιχη ιδιότητα μιας πολιτικής επαλήθευσης (VerificationPolicy), τύπου intentMap, με βάση την οποία ελέγχεται η ιδιότητα του intentAnnotation. Η τρίτη στήλη δηλώνει με ποια μέθοδο θα γίνει ο έλεγχος ισότητας/ομοιότητας της τιμής των ιδιοτήτων που αναφέρονται στην πρώτη και δεύτερη στήλη. Αναλυτικότερα, για την επαλήθευση των προθέσεων που απαιτεί μια υπηρεσία έναντι των PolicySet που προσφέρει το περιβάλλον εκτέλεσης SCA, ακολουθούνται τα εξής βήματα:

1. Αρχικά, επαληθεύεται αν είναι δυνατόν οι προθέσεις που απαιτούνται, έναντι των προθέσεων που προσφέρονται από κάθε intentMap. Σε κάθε στόχο, υπολογίζεται και ελέγχεται η τομή των ονομάτων των προθέσεων που απαιτούνται από κάθε intentAnnotation και αυτών στα intentMap, των αντίστοιχων PolicySet. Αυτός ο έλεγχος χρησιμοποιείται για

τον έλεγχο των κατανεμημένων προθέσεων (profile intents). Μια κατανεμημένη πρόθεση, υπενθυμίζεται ότι επαληθεύεται όταν προσφέρονται από το περιβάλλον εκτέλεσης όλες οι προθέσεις από τις οποίες εξαρτάται.

2. Έπειτα, ελέγχεται το SCA δόμημα το οποίο περιορίζει μια πρόθεση έναντι του SCA δομήματος στο οποίο ένα intentMap εφαρμόζεται (η ιδιότητα appliesTo κληρονομείται στο intentMap από τον κόμβο PolicySet που είναι πατέρας του). Αναφέρθηκε στην παράγραφο 4.4.1, ότι ως δόμημα SCA, συνήθως απαντάται μια έκφραση XPath που δηλώνει τον αφαιρετικό τύπο XML Element τον οποίο περιορίζει η πρόθεση ή στον οποίο εφαρμόζεται η PolicySet. Συνεπώς, κατάλληλος τρόπος ελέγχου αυτής της ιδιότητας, είναι ο έλεγχος αν οι δυο XPath εκφράσεις, αναφέρονται στον ίδιο αφαιρετικό τύπο XML Element.

3. Στην περίπτωση που έχουμε μια προσδιορισμένη πρόθεση, πρέπει να εξεταστεί αν προσφέρεται στις PolicySet, κάποιος χαμηλότερου επιπέδου προσδιοριστής ο οποίος επαληθεύει την πρόθεση. Αυτό, όπως αναφέρεται στον πίνακα που είδαμε προηγουμένως, ελέγχεται με τον έλεγχο των ονομάτων των προσδιοριστών που επιδέχεται η πρόθεση, αν κάποιο από αυτά περιέχεται στο όνομα του προσδιοριστή του intentMap που προσφέρεται από το PolicySet.

Στο σημείο αυτό, σκόπιμη κρίνεται η παράθεση ενός παραδείγματος επαλήθευσης προθέσεων σε ένα SCA σύστημα λογισμικού, από το περιβάλλον-πλαίσιο που παρουσιάζεται στην παρούσα διπλωματική. Για το παράδειγμα αυτό, θεωρούμε ότι μια υπηρεσία – στόχος σε ένα SCA σύστημα λογισμικού, ορίζει ως QoS περιορισμό της, την παρακάτω πρόθεση.

```
<intent       name="messageProtection"
              constrains="sca:binding"
              requires="confidentiality integrity">
              <description>
                     Protect messages from unauthorized reading or
                     modification.
              </description>
</intent>

<intent name="confidentiality.transport"/>
<intent name="confidentiality.message"/>
```

Η αρχική πρόθεση αυτή δηλώνει πρόθεση για την διατήρηση της ποιότητας μηνυμάτων και προϋποθέτει την ύπαρξη δυο άλλων προθέσεων, της πρόθεσης “confidentiality” (εμπιστευτικότητας) και integrity (ακεραιότητας). Το δεύτερο και τρίτο xml element intent, δηλώνουν πως η πρόθεση confidentiality είναι προσδιορισμένη και ότι δυο δυνατοί προσδιοριστές της είναι το “transport” και το “message”. Η πρόθεση messageProtection, μπορεί να ικανοποιηθεί από το παρακάτω PolicySet.

```
<policySet    name="MessagingPolicies"
              provides="confidentiality integrity"
              appliesTo="sca:binding"
              xmlns="http://www.osoa.org/xmlns/sca/1.0">
       <intentMap provides="confidentiality">
              <qualifier name="transport">
              </qualifier>
       </intentMap>
</policySet>
```

Για την επαλήθευση της πρόθεσης messageProtection από το σύνολο πολιτικών MessagingPolicies, ακολουθείται ο παρακάτω αλγόριθμος.

1. Αρχικά, ελέγχονται τα παιδιά requires της messageProtection και provides της MessagingPolicies. Βλέπουμε, πως όποιες προθέσεις απαιτούνται από την πρόθεση, προσφέρονται από το σύνολο πολιτικών.

2. Έπειτα, ελέγχονται οι εκφράσεις XPath που βρίσκονται στα πεδία constrains της πρόθεσης και appliesTo του συνόλου πολιτικών. Ως XPath εκφράσεις είναι ίδιες, άρα έχουμε και αυτή την ιδιότητα να επαληθεύεται.

3. Τέλος, ελέγχεται η προσδιορισμένη πρόθεση που χρειάζεται για την επαλήθευση της messageProtection, confidentiality. Παρατηρείται πως προσφέρεται από το σύνολο πολιτικών ένας από τους πιθανούς προσδιοριστές και συγκεκριμένα ο “transport”.

5.4.2 Επαλήθευση Επισημειώσεων Ενέργειας

Στην περίπτωση των επισημειώσεων ενέργειας, τα πράγματα είναι πιο απλά από την περίπτωση των επαληθεύσιμων επισημειώσεων. Όπως αναφέρθηκε στην παράγραφο 4.4.2, οι επισημειώσεις ενέργειας, χωρίζονται σε δυο κατηγορίες τις επισημειώσεις πριν και σε επισημειώσεις μετά την επαλήθευση του στόχου.

Η επισημείωση ενέργειας, όπως ορίζεται στο διάγραμμα τομέα στην Εικόνα 4-IV, επιβάλλει στις κλάσεις που την υλοποιούν να προσφέρουν μια μέθοδο με το όνομα “doAction”, η οποία ανά περίπτωση είναι η “doPreAction” ή “doPostAction”. Η μέθοδος αυτή, οφείλει να επιστρέφει μια boolean τιμή true ή false, ανάλογα με την κατάσταση επιστροφής της, δηλαδή κατά πόσον η ενέργεια που έπρεπε να διεκπεραιωθεί κατά την κλήση της, τελείωσε επιτυχώς ή όχι. Αυτή ακριβώς η μεταβλητή επιστροφής ελέγχεται από το περιβάλλον-πλαίσιο, ώστε να επαληθευτεί ή όχι κάθε επισημείωση ενέργειας που επισημαίνεται ανά στόχο.

5.4.3 Διαδικασία Επαλήθευσης σε Σύνολα Στόχων

Οι προηγούμενες παράγραφοι, ανέφεραν τις μεθοδολογίες οι οποίες ακολουθούνται για την επαλήθευση κάθε είδους Επισήμανσης ανά στόχο σε ένα δέντρο στόχων. Το περιβάλλον-πλαίσιο, όμως, καλείται να κάνει κάτι πιο πολύπλοκο από αυτό, καλείται να επαληθεύσει κάθε φορά πιθανά σύνολα στόχων, τα οποία προκύπτουν ανά περίπτωση από τον τρέχοντα στόχο. Συνεπώς, οφείλει να υπάρχει σαφής ορισμός της διαδικασίας που πρέπει να ακολουθηθεί, αφενός στην περίπτωση επαλήθευσης όλων των επισημειώσεων ενός συγκεκριμένου στόχου και αφετέρου στην περίπτωση επαλήθευσης όλων των στόχων σε ένα τρέχοντα σύνολο στόχων.

Αρχικά, ορίζεται η διαδικασία που ακολουθείται από το περιβάλλον-πλαίσιο για την επαλήθευση ενός στόχου, δηλαδή την διαδικασία επαλήθευσης όλων των επισημειώσεων ενός στόχου. Η διαδικασία που ακολουθείται σε αυτήν την περίπτωση, είναι η εξής:

- Αρχικά, εκτελούνται και επαληθεύονται όλες οι επισημειώσεις τύπου επισημειώσεις ενέργειας πριν την επαλήθευση.
- Έπειτα, επαληθεύονται όλες οι επαληθεύσιμες επισημειώσεις, διαδικασία που θα μπορούσε να χαρακτηριστεί ως η κύρια επαλήθευση του στόχου.
- Τέλος, εκτελούνται και επαληθεύονται όλες οι επισημειώσεις τύπου επισημειώσεις ενέργειας μετά την επαλήθευση.

Η σειρά με την οποία συμβαίνουν οι επαληθεύσεις στην διαδικασία αυτή, είναι προφανής και αναδεικνύει την σημασιολογία κάθε επισημείωσης που προσθέσαμε στο μοντέλο δέντρου στόχων. Στην σειρά αυτή, εξηγούνται και οι ονομασίες των δυο υποτύπων των επισημειώσεων ενέργειας, καθώς η επαλήθευση των επαληθεύσιμων επισημειώσεων χαρακτηρίζεται ως η κύρια (και πιο σημαντική) επαλήθευση στις επισημειώσεις του στόχου, ενώ εκατέρωθεν αυτής βρίσκονται ενέργειες που πρέπει να συμβούν για τις οποίες η μοναδική επαλήθευση που χρειάζεται, είναι η επαλήθευση του ότι συνέβησαν.

Στο σημείο αυτό, πρέπει να ορισθεί η μεθοδολογία που πρέπει να ακολουθηθεί στην περίπτωση όπου πρέπει να επαληθευτεί ένα σύνολο από στόχους. Ο ορισμός αυτός, θα γίνει με την βοήθεια ενός παραδείγματος, το οποίο είναι αυτό της παραγράφου 5.3.2.4 το οποίο έχει ως δέντρο στόχων αυτό που εμφανίζεται στην Εικόνα 5-IX. Η μελέτη του δέντρου στόχων αυτού, υπό την υπόθεση ότι ως τρέχοντας στόχος είναι ο G_1, δίνει ως σύνολα στόχων που πρέπει να επαληθευτούν τα: VS_1={G_3, G_5, G_1} και VS_2={G_4, G_5, G_1}. Για το VS_1, η πορεία επαλήθευσής του (με την προϋπόθεση ότι κάθε βήμα επαληθεύεται από στοιχεία που εκμαιεύονται από το υπό έλεγχο σύστημα λογισμικού), είναι η εξής:

- Εκτέλεση και επαλήθευση των πριν την επαλήθευση επισημειώσεων ενέργειας του στόχου G_3.
- Επαλήθευση των επαληθεύσιμων επισημειώσεων του στόχου G_3.
- Εκτέλεση και επαλήθευση των μετά την επαλήθευση επισημειώσεων ενέργειας του στόχου G_3.
- Η ίδια διαδικασία για τον στόχο G_5 και έπειτα για τον στόχο G_1.

Η παρατήρηση είναι, πως η μεθοδολογία επαλήθευσης ενός συνόλου στόχων, περιλαμβάνει επαλήθευση καθενός στόχου, ξεκινώντας από τους στόχους που ανήκουν πιο κοντά στα φύλλα του δέντρου και προχωρώντας προς στόχους που ανήκουν πιο κοντά στην ρίζα του δέντρου. Δηλαδή, η επαλήθευση ενός συνόλου στόχων, γίνεται από επάνω προς τα κάτω.

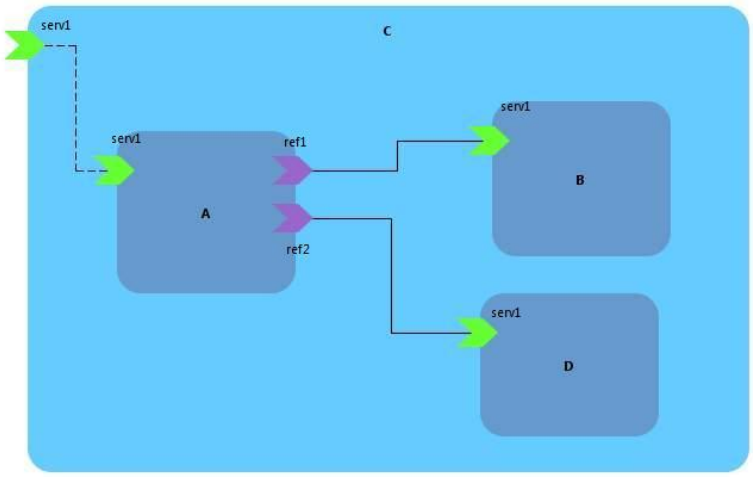
5.5 Στρώμα Βρόχου Ανατροφοδότησης

Το στρώμα βρόχου ανατροφοδότησης, περιλαμβάνει τις λειτουργίες που πρέπει να επιτελεστούν από το περιβάλλον-πλαίσιο κατά την εύρεση του τρέχοντος στόχου, με βάση την κατάσταση του υπό έλεγχο συστήματος λογισμικού. Στην παράγραφο αυτή, θα παρουσιαστούν τρεις αλγόριθμοι, που αφορούν στην εισαγωγή, την διαγραφή και την ενημέρωση δομοστοιχείου στο SCA σύστημα λογισμικού. Για την παρουσίαση των αλγορίθμων αυτών, θα χρησιμοποιηθεί ως βάση το παράδειγμα της παραγράφου 5.3.2.4 το οποίο έχει ως δέντρο στόχων, αυτό που παρουσιάζεται στην Εικόνα 5-IX.

5.5.1 Εισαγωγή Δομοστοιχείου

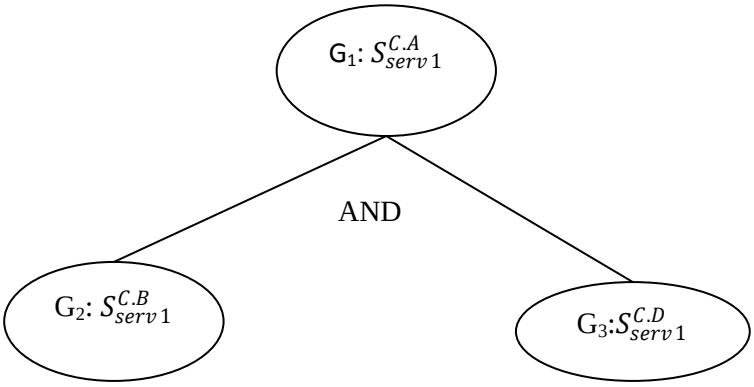
Μια βασική περίπτωση που χρειάζεται προσαρμοστικότητα στον έλεγχο των υπηρεσιοκεντρικών συστημάτων λογισμικού, είναι η εισαγωγή ενός δομοστοιχείου στην υπάρχουσα δομή του υπηρεσιοκεντρικού συστήματος λογισμικού. Για να υπάρχει συνέπεια στις αλλαγές που γίνονται στο σύστημα λογισμικού, προφανώς το δομοστοιχείο που εισάγεται ως καινούριο, θα προσφέρει εναλλακτικές υλοποιήσεις υπηρεσιών που ήδη προσφέρονται από άλλα δομοστοιχεία στο σύστημα λογισμικού. Αν συνέβαινε το αντίθετο, τότε είτε το σύστημα λογισμικού δεν θα λειτουργούσε κανονικά πριν την εισαγωγή του επιπλέον δομοστοιχείου (καθώς το δομοστοιχείο προσφέρει μια υπηρεσία που χρειάζεται από το σύστημα), είτε το νέο δομοστοιχείο θα προσέφερε μια υπηρεσία στο σύστημα, η οποία δεν είναι αναγκαία. Στο πρότυπο SCA, αυτό σημαίνει ότι μπορεί να εισαχθεί στο υπό έλεγχο σύστημα λογισμικού, ένα δομοστοιχείο του οποίου η υπηρεσία θα συνδεθεί ως πέρας αναφοράς πολλαπλότητας μεγαλύτερης του 1 (άρα μπορεί να δεχτεί μια επιπλέον υπηρεσία).

Για την συνέχεια του παραδείγματος της παραγράφου αυτής, υποθέτουμε ότι το σύστημα λογισμικού, πριν την εισαγωγή νέου δομοστοιχείου παρουσίαζε την εξής δομή:



Εικόνα 5-XII: Δομή SCA συστήματος, πριν την εισαγωγή

Το SCA σύστημα λογισμικού που παρουσιάζεται στην παραπάνω εικόνα, έχει ως δέντρο στόχων το εξής:



Εικόνα 5-XIII: Δέντρο στόχων, πριν την εισαγωγή

Έστω ότι σε μια τυχαία χρονική στιγμή της λειτουργίας του συστήματος λογισμικού που περιγράφεται από τις πιο πάνω εικόνες, εισάγεται το δομοστοιχείο C, το οποίο προσφέρει την υπηρεσία $S^{C.C}_{serv\,1}$, η οποία αποτελεί εναλλακτική υλοποίηση της υπηρεσίας $S^{C.B}_{serv\,1}$, που προσφέρει το δομοστοιχείο B. Έστω επίσης, ότι στο σύστημα εισάγεται επιπλέον και η σύνδεση ${}^{C.A}_{ref\,1}W^{C.C}_{serv\,1}$, οπότε η αναφορά $R^{C.A}_{ref\,1}$ συνδέεται με την υπηρεσία που μόλις εισήχθη. Το νέο SCA σύστημα είναι

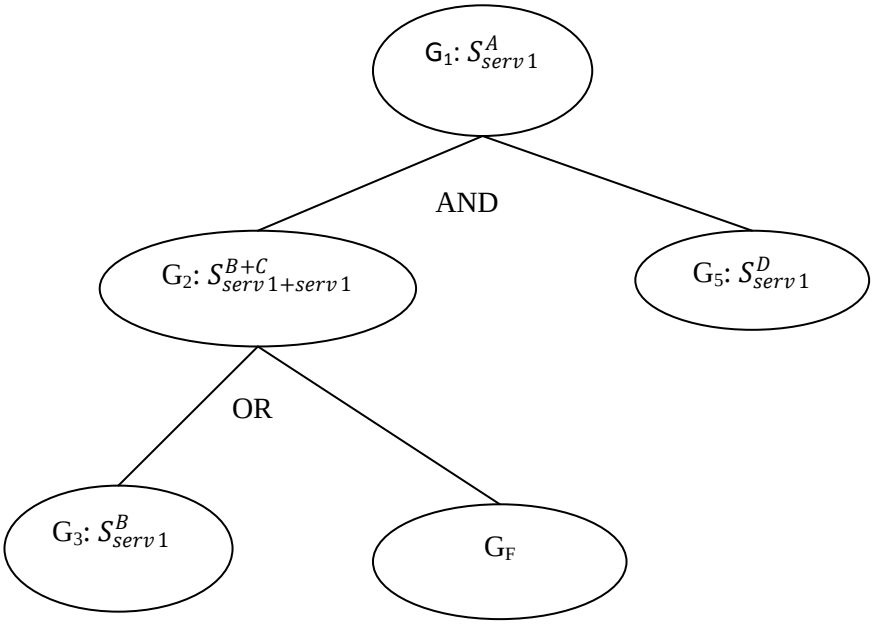
πανομοιότυπο με το σύστημα που παρουσιάσθηκε στην παράγραφο 5.3.2.4, το οποίο έχει ως δέντρο στόχων, αυτό στην Εικόνα 5-IX. Συνεπώς, η αλλαγή που παρατηρείται στο δέντρο στόχων, είναι η μετατροπή του αρχικού στόχου G_2 (στην πιο πάνω εικόνα), στον ORDecomposition στόχο G_2 στην Εικόνα 5-IX. Προφανώς, νέος κύριος στόχος που τίθεται είναι ο G_1, ο πατέρας στόχος αυτού που άλλαξε από Atomic Goal σε DecompositionGoal, καθώς αυτός επηρεάζεται από την αλλαγή αυτή (ο πατέρας στόχος είναι αυτός που εξαρτάται από τους στόχους παιδιά του). Ο νέος κύριος στόχος που τέθηκε, δίνει ως σύνολα στόχων που πρέπει να ελεγχθούν από τις πολιτικές επαλήθευσης τα VS_1={G_3, G_5, G_1} και VS_2={G_4, G_5, G_1}.

Συνοψίζοντας το παράδειγμα της εισαγωγής δομοστοιχείου, μπορούμε να πούμε ότι ως νέος τρέχοντας στόχος κάθε φορά τίθεται ο στόχος – υπηρεσία ο οποίος είναι πατέρας του υποδέντρου που αλλάζει. Αν είναι πέραν του ενός τα υποδέντρα που αλλάζουν, τίθενται ως τρέχοντες στόχοι όλοι οι πατέρες – στόχοι των υποδέντρων που αλλάζουν και από καθένα τρέχοντα στόχο, βρίσκονται τα αντίστοιχα σύνολα στόχων που πρέπει να επαληθευτούν από το περιβάλλον-πλαίσιο. Πρακτικά, στην εισαγωγή ενός νέου δομοστοιχείου, ως τρέχοντες στόχοι τίθενται οι στόχοι – υπηρεσίες που ανήκουν σε δομοστοιχεία που περιέχουν αναφορές που συνδέονται σε υπηρεσίες που προσφέρει το νέο δομοστοιχείο.

5.5.2 Διαγραφή Δομοστοιχείου

Η περίπτωση της διαγραφής ενός δομοστοιχείου από το υπό-έλεγχο SCA σύστημα λογισμικού, αναφέρεται στην περίπτωση εκείνη όπου ένα δομοστοιχείο, παρουσιάζει εσφαλμένη λειτουργία και/ή τίθεται εκτός λειτουργίας. Σε κάτι τέτοιο, μπορεί να συντελέσει μια πληθώρα από παράγοντες, όπως δικτυακές ανωμαλίες, απουσία ή πρόβλημα σε υπολογιστικούς πόρους που χρειαζόταν το δομοστοιχείο για να λειτουργήσει κ.ά. Κατά την διαγραφή ενός δομοστοιχείου, στο αντίστοιχο δέντρο στόχων, θα παρατηρείται αλλαγή στους στόχους – υπηρεσίες που ανήκουν σε δομοστοιχεία τα οποία είχανε αναφορές στις υπηρεσίες που προσέφερε το δομοστοιχείο που διεγράφη.

Σαν υπόθεση, τίθεται αρχικά το SCA σύστημα λογισμικού του παραδείγματος της παραγράφου 5.3.2.4, που έχει ως δέντρο στόχων αυτό που παρουσιάζεται στην Εικόνα 5-IX. Έστω, ότι κατά την ομαλή λειτουργία του υπό έλεγχο συστήματος λογισμικού, αφαιρείται το δομοστοιχείο C, το οποίο προσφέρει την υπηρεσία $S_{serv1}^{C.C}$. Το σύστημα τότε, εκφυλλίζεται στο σύστημα λογισμικού που παρουσιάσθηκε στην Εικόνα 5-XII. Με την αλλαγή αυτή, το δέντρο στόχων μετασχηματίζεται στο παρακάτω:



Εικόνα 5-XIV: Δέντρο στόχων για SCA σύστημα μετά την διαγραφή

Παρατηρούμε ότι το δέντρο στόχων της παραπάνω εικόνας, είναι διαφορετικό από αυτό που παρουσιάζεται στην Εικόνα 5-XIII. Όμως, μια προσεκτικότερη ματιά, αναδεικνύει την σημασιολογική ισότητα που υπάρχει μεταξύ των δυο αυτών δέντρων στόχων και το γεγονός ότι εξάγουν αμφότερα το ίδιο σύνολο στόχων που πρέπει να επαληθευτεί από τον Διαχειριστή Πολιτικών. Η διαφορά τους, είναι πώς το δέντρο στόχων που παρουσιάζεται πιο πάνω, είναι αυτό που μπορεί να εξαχθεί αλγοριθμικά από το περιβάλλον-πλαίσιο μετά από μια διαγραφή ενός δομοστοιχείου, βάζοντας τον εικονικό στόχο G_F στην θέση κάθε στόχου – υπηρεσίας που διαγράφηκε, ως συνέπεια της διαγραφής του δομοστοιχείου. Ως κύριοι στόχοι, τίθενται οι στόχοι γονείς των G_F που εισάγονται λόγω της διαγραφής κάποιου δομοστοιχείου.

Συνοψίζοντας για την περίπτωση της διαγραφής ενός δομοστοιχείου, έχουμε αρχικά την αντικατάσταση των υπηρεσιών – στόχων που προσέφερε το δομοστοιχείο με τον εικονικό στόχο G_F. Έπειτα, ως τρέχοντες στόχοι, τίθενται οι γονείς στόχοι των στόχων που άλλαξαν, ενώ τα παιδιά των στόχων που άλλαξαν διαγράφονται από το δέντρο στόχων. Πρακτικά, αυτό σημαίνει πως η αλλαγή του στόχου σε G_F προκαλεί και την διαγραφή του υποδέντρου – παιδιού του στόχου που άλλαξε κατά την διαγραφή του δομοστοιχείου.

5.5.3 Ενημέρωση Δομοστοιχείου

Η παράγραφος αυτή, αναφέρεται στην περίπτωση όπου στο υπό έλεγχο σύστημα λογισμικού, ένα δομοστοιχείο ενημερώνεται/ανανεώνεται. Αυτό μπορεί να υποδηλώνει μια σειρά από γεγονότα που

μπορεί να σχετίζονται, όπως για παράδειγμα ενημέρωση ενός δομοστοιχείου με νεότερη έκδοση του ιδίου, ή επανεκκίνηση ενός δομοστοιχείου που σταμάτησε να λειτουργεί σε κάποια χρονική στιγμή, εξαιτίας κάποιου σφάλματος.

Αυτό που γίνεται άμεσα αντιληπτό, είναι πως η περίπτωση της ενημέρωσης ενός δομοστοιχείου, περιλαμβάνει σειριακά, την διαγραφή και έπειτα την εισαγωγή του υπόλογου δομοστοιχείου. Συνεπώς, μια σωστή και πλήρης λύση για το πρόβλημα αυτό, είναι κατά την ενημέρωση ενός δομοστοιχείου, να εκτελούνται διαδοχικά στο περιβάλλον-πλαίσιο, οι αλγόριθμοι διαγραφής και έπειτα εισαγωγής του δομοστοιχείου αυτού. Ως τρέχοντες στόχοι τίθενται η ένωση των συνόλων στόχων που προκύπτουν από την εκτέλεση των αλγορίθμων που παρουσιάσθηκαν στις δυο προηγούμενες παραγράφους. Η λύση αυτή είναι πλήρης, καθώς αυτό το σύνολο είναι το μέγιστο σύνολο (maximal set) που μπορεί να παρουσιασθεί κατά την ενημέρωση ενός δομοστοιχείου σε ένα SCA σύστημα λογισμικού.

Κεφάλαιο 6: Υλοποίηση και Μελέτη Περίπτωσης

6.1 Εισαγωγή

Στα 3 προηγούμενα κεφάλαια, παρουσιάσθηκαν διαδοχικά, η αρχιτεκτονική του περιβάλλοντος-πλαισίου, το μοντέλο δέντρων στόχων που αυτό χρησιμοποιεί για την μοντελοποίηση του υπό έλεγχο συστήματος λογισμικού, καθώς και οι αλγόριθμοι που διέπουν την λειτουργία του και την χρήση του Μοντέλο Δέντρου Στόχων από αυτό.

Στο παρόν κεφάλαιο, μελετάται η πρωτότυπη υλοποίηση του περιβάλλοντος-πλαισίου που δημιουργήθηκε στα πλαίσια εκπόνησης της παρούσας διπλωματικής εργασίας, καθώς και τα συμπεράσματα λειτουργίας του σε μελέτη περίπτωσης που χρησιμοποιήθηκε για την αξιολόγησή του.

Αρχικά, αναφέρονται εξαρτήσεις της υλοποίησης του περιβάλλοντος-πλαισίου, από βιβλιοθήκες οι οποίες χρησιμοποιήθηκαν κατά την διάρκεια της γραφής του κώδικά του. Στην συνέχεια, παρουσιάζεται η μελέτη περίπτωσης, ένα συγκεκριμένο παράδειγμα συστήματος λογισμικού που έχει μοντελοποιηθεί με το προγραμματιστικό μοντέλο SCA. Δείχνεται η δομή του, μαζί με το Δέντρο Στόχων που εξάγεται από το περιβάλλον-πλαίσιο. Τέλος, παρατίθενται τα αποτελέσματα (έξοδος) της λειτουργίας του σε δυο αλλαγές του υπό έλεγχο συστήματος λογισμικού. Η πρώτη περίπτωση, αναφέρεται σε ανανέωση δομοστοιχείου, με νεότερο που είναι πλήρως συμβατό με το προηγούμενο. Αντίθετα, η δεύτερη περίπτωση αναφέρεται σε ανανέωση δομοστοιχείου, με άλλο που παρουσιάζει ασυμβατότητα με το προηγούμενο στην σχεδίασή του.

6.2 Βιβλιοθήκες που βοήθησαν στην Υλοποίηση

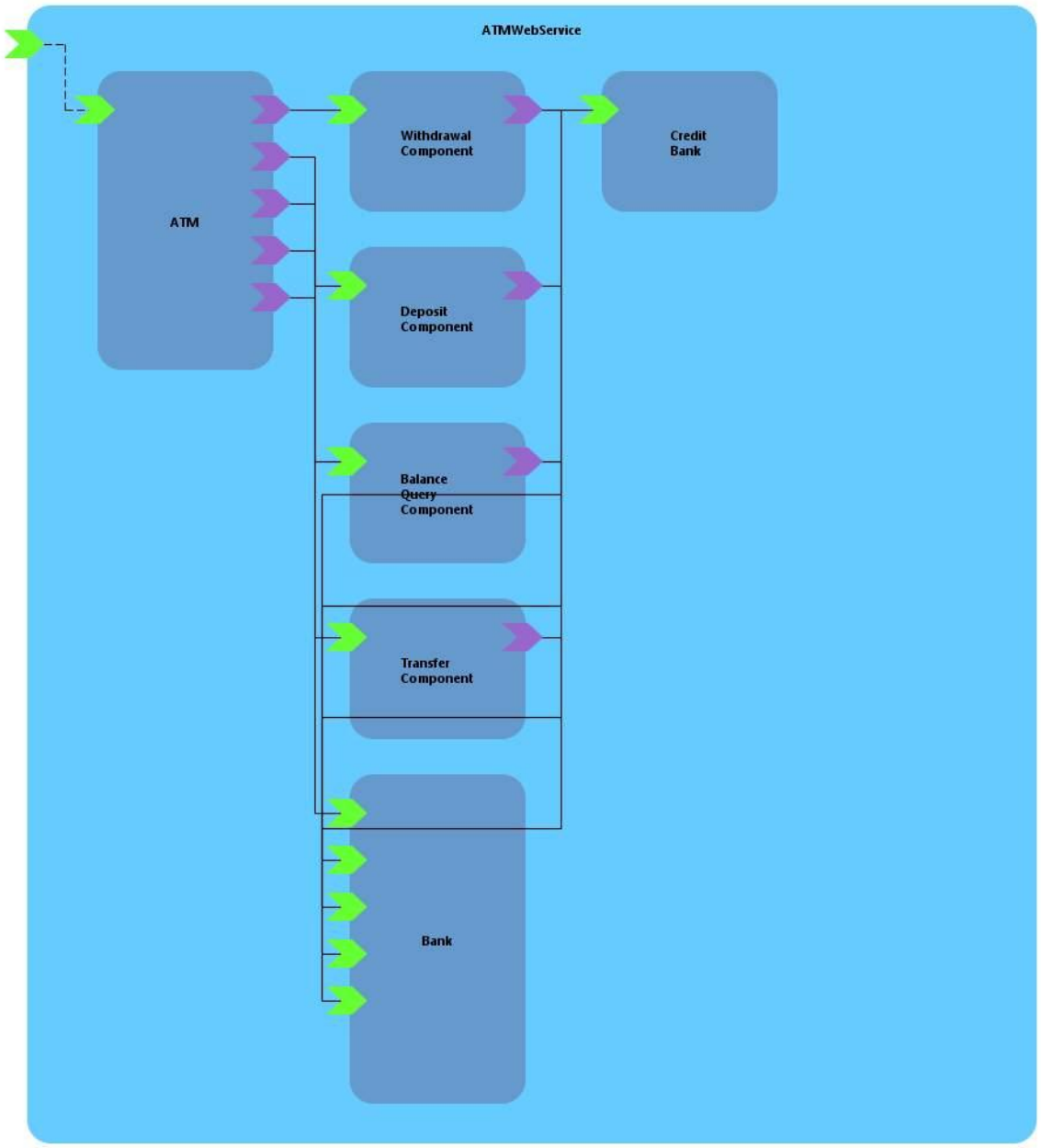
Η πρότυπη υλοποίηση του περιβάλλοντος-πλαισίου χρησιμοποιεί τις υλοποιήσεις δυο πολύ διαδεδομένων περιβαλλόντων-πλαισίων, που τείνουν να καθιερωθούν στην υλοποίηση συστημάτων λογισμικού. Η πρώτη βιβλιοθήκη λογισμικού που χρησιμοποιείται, καλείται να αποτελέσει αρωγό

του περιβάλλοντος-πλαισίου στον τομέα της καταγραφής γεγονότων (logging). Με την έννοια καταγραφή γεγονότων, στην συγκεκριμένη περίπτωση εννοούμε την καταγραφή των ενεργειών που πραγματοποιούνται στα ενδότερα του περιβάλλοντος-πλαισίου, για λόγους αποσφαλμάτωσης και παρακολούθησής του. Η βιβλιοθήκη λογισμικού που χρησιμοποιήθηκε για αυτούς τους λόγους καλείται *log4j* και αποτελεί προϊόν του Apache Software Foundation ([17]).

Η δεύτερη βιβλιοθήκη που χρησιμοποιείται από την υλοποίηση του περιβάλλοντος-πλαισίου, χρησιμοποιείται για την αντιστοίχηση των περιγραφών των SCA XML σχημάτων σε κλάσεις Java, ώστε να μπορεί να χρησιμοποιηθεί το μοντέλο ενός SCA συστήματος (που περιγράφεται σε xml αρχεία), προγραμματιστικά από το περιβάλλον-πλαίσιο. Η τεχνολογία αντιστοίχησης ενός σχήματος XML (XSD, XML Schema Description [18]), αναφέρεται στο πρότυπο Java Architecture for XML Binding (JAXB, Sun Microsystems [19]). Μέρη της τεχνολογίας αυτής έχουν υλοποιηθεί απευθείας στα περιβάλλοντα εκτέλεσης Java εφαρμογών (Java Virtual Machine, JVM), όμως μια πλήρης και ευρέως χρησιμοποιούμενη υλοποίηση του προτύπου JAXB, έχει γίνει από το Apache Software Foundation στο έργο JAXME 2 ([20]). Η τελευταία υλοποίηση, χρησιμοποιείται από το περιβάλλον-πλαίσιο ως αρωγό στην προγραμματιστική χρήση μοντέλων SCA, από xml αρχεία.

6.3 Μελέτη Περίπτωσης

Για την ανάγκη ελέγχου του περιβάλλοντος-πλαισίου για τον πριν την χρήση έλεγχο, μοντελοποιήθηκε ένα σύστημα λογισμικού με βάση το προγραμματιστικό μοντέλο SCA. Το μοντέλο του υπό έλεγχο συστήματος φαίνεται στην παρακάτω εικόνα.



Εικόνα 6-Ι: Μελέτη Περίπτωσης Δικτυακής Υπηρεσίας ATM σε SCA

Το σύστημα της μελέτης περίπτωσης, αποτελεί μια δικτυακή υπηρεσία που προσπαθεί να προσομοιώσει την λειτουργία ενός ATM, ενός τραπεζικού οργανισμού. Κεντρικό σημείο του είναι το Σύνθετο Δομοστοιχείο *ATMWebServiceSCA*, το οποίο προσφέρει την δικτυακή υπηρεσία στην οποία συνδέεται ο χρήστης ώστε να διεκπεραιώσει μια από τις λειτουργίες, που θα μπορούσε να διεκπεραιώσει και σε ένα κλασσικό σύστημα ATM. Η μοντελοποίηση και υλοποίηση αυτής της μελέτης περίπτωσης στηρίχτηκαν στο κλασσικό παράδειγμα αντικειμενοστραφούς σχεδίασης του Bjork [21], *An ATM Example*. Το παράδειγμα του Bjork, μετατράπηκε σε υπηρεσιοκεντρικό, για τις ανάγκες της παρούσας διπλωματικής. Στο παράδειγμα αυτό, ένα σύστημα λογισμικού που

προσομοιώνει την λειτουργία ενός ATM οφείλει να προσφέρει λειτουργικότητα για *Ανάληψη* (*Withdrawal*), *Κατάθεση* (*Deposit*), *Ενημέρωση Υπολοίπου* (*Balance Query*), *Μεταφορά Χρημάτων* (Transfer) και *Αναγνώριση Κάρτας* (*Card Authentication*). Η λειτουργικότητα αυτή, διατηρήθηκε και στο υπηρεσιοκεντρικό ATM, με τις όποιες απαραίτητες αλλαγές στην σχεδίαση, ώστε η λειτουργικότητα να προσφέρεται υπηρεσιοκεντρικά και όχι αντικειμενοστραφώς. Ως υπηρεσιοκεντρικό σύστημα, το *ATMWebServiceSCA* έχει την εξής δομή:

- **ATM:** Το δομοστοιχείο αυτό, είναι αυτό που αναλαμβάνει την επικοινωνία με τον χρήστη και την προώθηση των αιτήσεών του στο υπόλοιπο σύστημα. Προσφέρει μια υπηρεσία, την *ATMService*, η οποία είναι προηγμένη υπηρεσία, στην αντίστοιχη υπηρεσία του Σύνθετου Δομοστοιχείου *ATMWebServiceSCA*. Η υπηρεσία *ATMService*, μοντελοποιεί το σύνολο της λειτουργικότητας που θέλουμε να έχει ένα σύστημα λογισμικού που προσομοιώνει ένα ATM. Συνεπώς, το δομοστοιχείο ATM, έχει αναφορές σε 5 άλλα δομοστοιχεία του υπό έλεγχο συστήματος, καθένα από τα οποία προσφέρει μια συγκεκριμένη υπηρεσία από αυτές που παρουσιάσθηκαν προηγουμένως.

- **BalanceQueryComponent.** Το δομοστοιχείο αυτό, αναλαμβάνει την λειτουργικότητα της *Ερώτησης Υπολοίπου* ενός ATM. Προσφέρει την υπηρεσία *BalanceQuery*, στο δομοστοιχείο ATM, χρησιμοποιώντας την υπηρεσία *BalanceQueryCheck* από το δομοστοιχείο Bank.

- **TransferComponent.** Το δομοστοιχείο αυτό, αναλαμβάνει να επιτελέσει την λειτουργία *Μεταφοράς Χρημάτων* ενός ATM. Προσφέρει την υπηρεσία *Transfer*, στο δομοστοιχείο ATM, ενώ χρησιμοποιεί την υπηρεσία *TransferCheck* του δομοστοιχείου Bank.

- **DepositComponent.** Το δομοστοιχείο αυτό, παρέχει την λειτουργία *Κατάθεσης Χρημάτων*. Η λειτουργία αυτή, υλοποιείται από την υπηρεσία *Deposit*, η οποία προσφέρεται στο δομοστοιχείο ATM. Το δομοστοιχείο Deposit, περιέχει μια αναφορά στην υπηρεσία *DepositCheck* του δομοστοιχείου Bank.

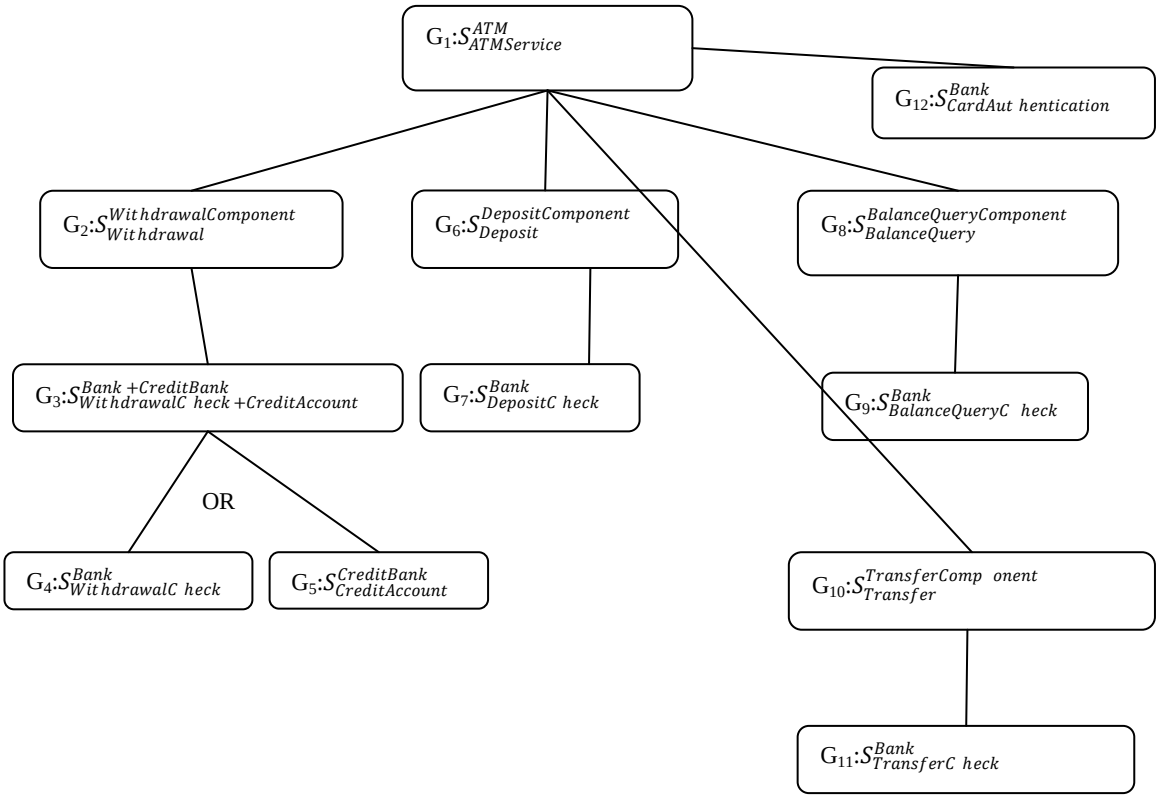
- **WithdrawalComponent**. Το δομοστοιχείο αυτό αναλαμβάνει τις αιτήσεις *Ανάληψης*, για το ATM. Η υπηρεσία που υλοποιεί την *Ανάληψη*, είναι η *Withdrawal*, η οποία προσφέρεται στο δομοστοιχείο ATM. Η περίπτωση του δομοστοιχείου αυτού, είναι ιδιαίτερη, καθώς εξαρτάται από τις υπηρεσίες δυο άλλων δομοστοιχείων, όχι μόνο ενός. Η υπηρεσία *Withdrawal* του δομοστοιχείου αυτού μπορεί να διεκπεραιωθεί είτε λαμβάνοντας τα αποτελέσματα εκτέλεσης της υπηρεσίας *WithdrawalCheck* του δομοστοιχείου Bank, είτε αυτά της υπηρεσίας *CreditAccount* του δομοστοιχείου CreditBank.

- **Bank**. Το Bank είναι το κεντρικό δομοστοιχείο, το οποίο παρέχει τις υπηρεσίες που συνεπικουρούν τα δομοστοιχεία που αναφέραμε προηγουμένως στην λειτουργία του έργου τους. Προσφέρει τις 5 λειτουργίες που πρέπει να προσφέρει ένα ATM σύστημα, ελέγχοντας από τα δεδομένα της τράπεζας αν η κάθε υπηρεσία μπορεί να εκτελεστεί.

- **CreditBank**. Το δομοστοιχείο αυτό, αναφέρεται στην λειτουργία ενός πιστωτικού οργανισμού, ο οποίος δανείζει τον ενδιαφερόμενο με τα απαραίτητα χρήματα, όταν το υπόλοιπο του λογαριασμού του δεν επαρκεί για *Ανάληψη*. Το έργο του δομοστοιχείου αυτού, είναι να προσφέρει αντίστοιχη υπηρεσία με την WithdrawalCheck, στην περίπτωση που η WithdrawalCheck αποτύχει.

Στο σημείο αυτό, αξίζει να σημειωθεί ότι το δομοστοιχείο CreditBank, μαζί με την λειτουργικότητα που προσφέρει, δεν ορίζονται στο παράδειγμα του Bjork [21], αλλά εισάγεται για τις ανάγκες της παρούσας διπλωματικής, ώστε να υπάρξει ένα πλουσιότερο παράδειγμα δέντρου στόχων.

Το Δέντρο Στόχων που προκύπτει από το παραπάνω SCA σύστημα λογισμικού, παρατίθεται στην συνέχεια.



Εικόνα 6-ΙΙ: Δέντρο Στόχων για το Σύστημα Λογισμικού της Μελέτης Περίπτωσης

Όπου δεν αναφέρεται διαφορετικά οι στόχοι στην Εικόνα 6-ΙΙ οι οποίοι αναλύονται σε περισσότερους υποστόχους, είναι τύπου ΚΑΙ αποσύνθεσης (ANDDecomposition). Στην Εικόνα 6-ΙΙ, αξίζει να σημειωθεί ότι οι εικονικοί στόχοι G_T ή G_F δεν αναφέρονται, καθώς αποτελούν πλεονασματική πληροφορία για τις απαιτήσεις του υπό έλεγχο συστήματος λογισμικού και στην υλοποίηση, όπως θα αναφερθεί και πιο κάτω, αγνοούνται. Στην Εικόνα 6-ΙΙ παρατηρείται επίσης, η μετατροπή ενός SCA μοντέλου στο δέντρο στόχων. Η υπηρεσία *ATMService*, που αποτελεί την βασική υπηρεσία που προάγεται η αντίστοιχη υπηρεσία που δίνει το Σύνθετο Δομοστοιχείο, αποτελεί την ρίζα του Δέντρου. Λόγω των αναφορών που έχει το δομοστοιχείο ATM (που περιέχει την υπηρεσία *ATMService*), βλέπουμε η *ATMService* να μορφώνεται ως ένας σύνθετος στόχος τύπου ΚΑΙ αποσύνθεσης (ANDDecomposition). Ως υποστόχοι αυτού, τίθενται όλες οι υπηρεσίες που βρίσκονται στα πέρατα των αναφορών του δομοστοιχείου ATM, ήτοι οι υπηρεσίες που προσφέρουν τα δομοστοιχεία WithdrawalComponent, TransferComponent, DepositComponent και BalanceQueryComponent και η υπηρεσία *CardAuthentication* του δομοστοιχείου Bank.

Από την Εικόνα 6-ΙΙ αξίζει επίσης να σημειωθεί και ο εικονικός σύνθετος τύπου Ή αποσύνθεσης στόχος G_3. Αναφέρθηκε προηγουμένως, ότι το δομοστοιχείο που αναλαμβάνει την λειτουργία της *Ανάληψης*, περιέχει μεν μια αναφορά, η οποία όμως έχει δυο πέρατα σε δυο διαφορετικά δομοστοιχεία. Η περίπτωση αυτή, όπως αναφέρθηκε στο κεφάλαιο 5, μοντελοποιείται μέσω σύνθετου στόχου τύπου Ή αποσύνθεσης. Ο στόχος αυτός, είναι ο στόχος G_3 και οι υποστόχοι του G_4 και G_5 αντιστοιχούν στις δυο εναλλακτικές περιπτώσεις ικανοποίησης της αναφοράς του δομοστοιχείου WithdrawalComponent.

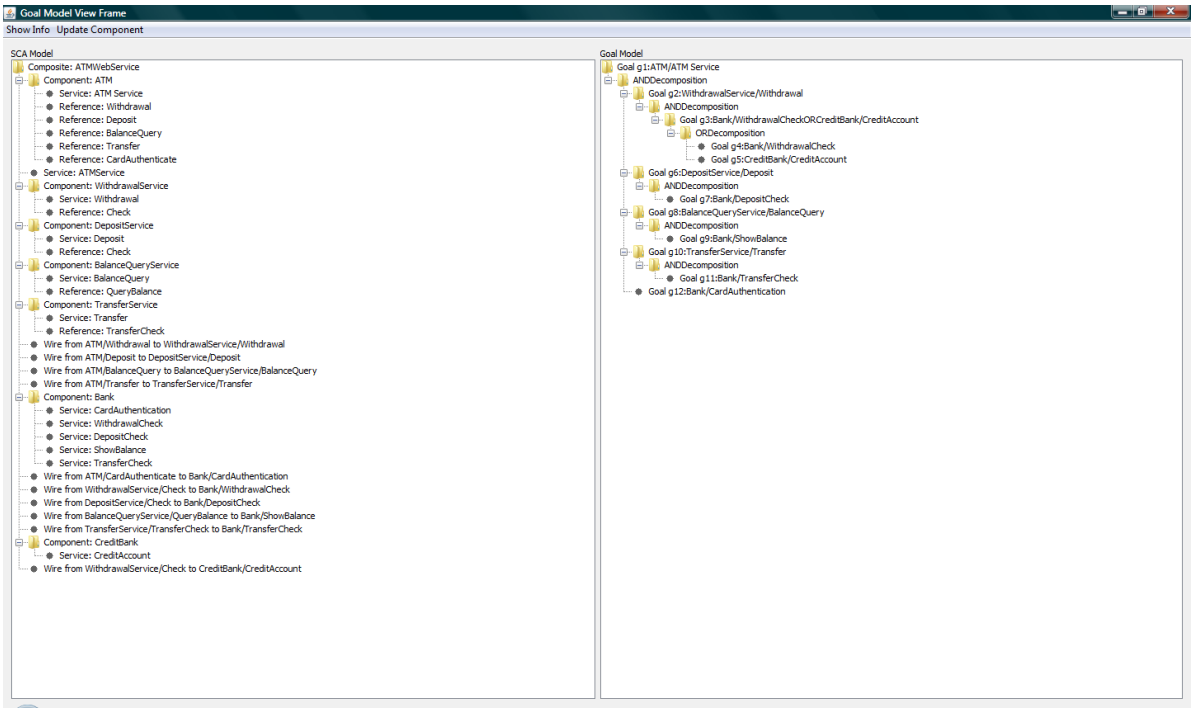
Στο παραπάνω σύστημα λογισμικού, θα εφαρμόζουμε την πρωτότυπη υλοποίηση του περιβάλλοντος-πλαισίου που παρουσιάζεται στην παρούσα διπλωματική.

6.4 Παραδείγματα και Στιγμιότυπα Εκτέλεσης

Εφαρμόζοντας το περιβάλλον-πλαίσιο στο SCA σύστημα λογισμικού που παρουσιάσθηκε στην παράγραφο 6.4, θα παρουσιάσουμε δυο σενάρια εκτέλεσης. Και τα δυο αφορούν στην ενημέρωση του δομοστοιχείου CreditBank, το πρώτο όμως με δομοστοιχείο που είναι συμβατό με το υπόλοιπο σύστημα, ενώ το δεύτερο με δομοστοιχείο το οποίο είναι ασύμβατο με την κατάσταση του περιβάλλοντος εκτέλεσης SCA. Αρχικά όμως, κρίνεται σκόπιμο να παρουσιαστούν τα πρώτα στάδια της λειτουργίας του περιβάλλοντος-πλαισίου, τα στάδια που αφορούν στην μοντελοποίηση.

6.4.1 Μοντελοποίηση – Εισαγωγή SCA Μοντέλου

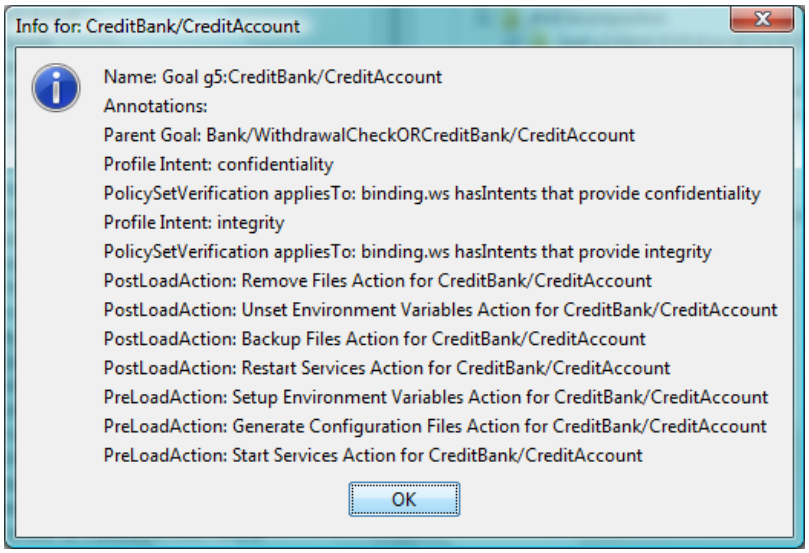
Στο παρακάτω στιγμιότυπο φαίνεται το κεντρικό παράθυρο της λειτουργίας του περιβάλλοντος πλαισίου, αφού έχουν ζητηθεί από τον χρήστη και έχουν φορτωθεί τα απαραίτητα αρχεία *ATMWebServiceSCA.composite* και *definitions.xml*, από τα οποία εξάγεται το μοντέλο δέντρου στόχων και οι επισημειώσεις του, αντίστοιχα.



Εικόνα 6-III: Κυρίως Παράθυρο - Σε αντιδιαστολή SCA και Goal Tree Μοντέλα

Στο κυρίως παράθυρο της υλοποίησης, φαίνονται σε αντιδιαστολή τα μοντέλα SCA (αριστερά) και Goal Tree (δεξιά). Οι παρατηρήσεις που αναφέρθηκαν για την εξαγωγή του μοντέλου δέντρου στόχων στην προηγούμενη παράγραφο, φαίνονται εναργώς και στην Εικόνα 6-III. Ο αναγνώστης καλείται να αντιπαραβάλλει την Εικόνα 6-III με την Εικόνα 6-II και την Εικόνα 6-I, ώστε να επαληθεύσει την λειτουργία του αλγόριθμου εξαγωγής του δέντρου στόχων από το SCA μοντέλο.

Το κυρίως παράθυρο, εμπεριέχει δυο μενού, *Show Info* και *Update Component* τα οποία χρησιμοποιούνται για να ληφθεί η πληροφορία των επισημειώσεων κάθε στόχου και να προσομοιωθεί η διαδικασία ενημέρωσης ενός δομοστοιχείου του υπό έλεγχο συστήματος, αντίστοιχα. Ένα παράδειγμα χρήσης του μενού *Show Info* για την εύρεση των πληροφοριών των επισημειώσεων του στόχου G_5, θα προκαλούσε την εμφάνιση του εξής διαλόγου πληροφορίας:



Εικόνα 6-IV: Παράθυρο Πληροφοριών Στόχου G_5

Στην Εικόνα 6-IV, παρατηρούνται όλες οι εξαγόμενες πληροφορίες για την υπηρεσία που αντιστοιχεί στον στόχο G_5. Αναφέρεται το όνομά του, ο στόχος – πατέρας του (ο εικονικός στόχος Ή αποσύνθεση) καθώς και οι πληροφορίες των επαληθεύσιμων επισημειώσεων και των επισημειώσεων ενέργειας. Όσον αφορά τις δεύτερες, όπως αναφέρθηκε στο κεφάλαιο 5, προς το παρόν αποτελούν εικονικές ενέργειες. Οι επαληθεύσιμες επισημειώσεις μοντελοποιούνται με βάση το μοντέλο SCA και (όπως και οι πολιτικές επαλήθευσης) το αρχείο definitions.xml το οποίο αναφέρει τις προθέσεις και τις πολιτικές που προσφέρει η ρύθμιση του περιβάλλοντος εκτέλεσης SCA.

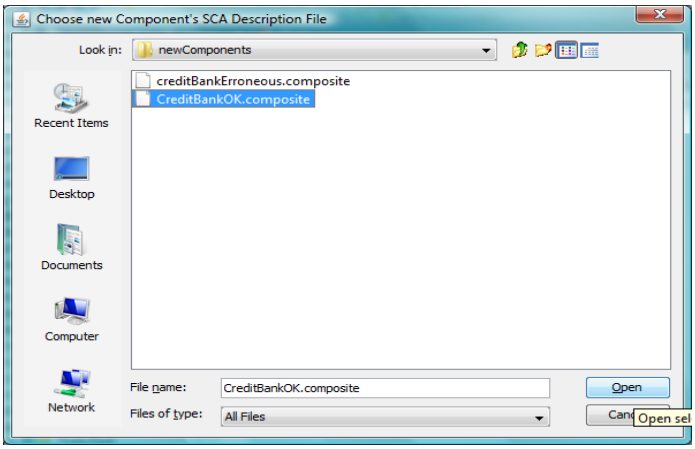
Στην Εικόνα 6-IV φαίνονται οι προθέσεις που απαιτεί ο στόχος G_5 καθώς και οι πολιτικές επαλήθευσης που έχουν επισυναφθεί, προς επαλήθευση των προθέσεων αυτών, όποτε χρειαστεί. Για την επιβεβαίωση της ορθότητας της παραπάνω εικόνας, παρατίθεται το χρησιμοποιούμενο xml αρχείο definitions το οποίο οδήγησε στις πιο πάνω επισημειώσεις.

```
<definitions xmlns="http://www.osoa.org/xmlns/sca/1.0"
                    targetNamespace="http://www.osoa.org/xmlns/sca/1.0"
                    xmlns:sca="http://www.osoa.org/xmlns/sca/1.0"
                    xmlns:tuscany="http://tuscany.apache.org/xmlns/sca/1.0">

    <policySet name="SecureMessagingPolicies"
                    provides="confidentiality integrity"
                    appliesTo="binding.ws"
                    xmlns="http://www.osoa.org/xmlns/sca/1.0"
                    xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
            <intentMap provides="confidentiality" default="transport">
                    <qualifier name="transport">
                    </qualifier>
                    <qualifier name="message">
                    </qualifier>
            </intentMap>
            <intentMap provides="integrity" default="transport">
                    <qualifier name="transport">
```

```
                    </qualifier>
                    <qualifier name="message">
                    </qualifier>
             </intentMap>
    </policySet>

       <!-- Policy Intents Used by the SCA Runtime -->
       <intent name="authentication"
                    constrains="sca:binding">
             <description>
                    Specifying this intent on references requires necessary
authentication information
                    to be sent along with outgoing messages. Specifying this
intent on service requires
                    incoming messages to be authenticated
             </description>
       </intent>

       <intent name="confidentiality"
                    constrains="sca:binding">
             <description>
                    Specifying this intent requires message exchanged to be
encrypted
             </description>
       </intent>

       <intent name="integrity"
                    constrains="sca:binding">
             <description>
                    Specifying this intent requires message exchanged to be signed
             </description>
       </intent>
</definitions>
```

6.4.2 Σενάριο Σωστής Ανανέωσης Δομοστοιχείου

Χρησιμοποιώντας το μενού *Update Component* υποδεικνύουμε στο περιβάλλον-πλαίσιο να επιχειρήσει ενημέρωση του δομοστοιχείου CreditBank. Η πρώτη αντίδραση του περιβάλλοντος-πλαισίου, είναι να ζητήσει το αρχείο περιγραφής του μοντέλου SCA του νέου δομοστοιχείου.



Εικόνα 6-V: Παράθυρο Επιλογής Μοντέλου SCA Δομοστοιχείου - Αντικαταστάτη

Έπειτα από την φόρτωση του νέου αυτού δομοστοιχείου που φιλοδοξούμε να αντικαταστήσει το παλιό CreditBank δομοστοιχείο, το περιβάλλον-πλαίσιο θέτει τους νέους στόχους ώστε να επαληθεύσει την συμβατότητα του νέου δομοστοιχείου με το περιβάλλον-πλαίσιο και το υπάρχον υπό έλεγχο σύστημα. Η καταγραφή της διαδικασίας αυτής φαίνεται παρακάτω:

```
17:13:02 [AWT-EventQueue-0] DEBUG SCABlackboard - Update Component event for
CreditBank occured
17:13:02 [AWT-EventQueue-0] DEBUG ActuatingServiceImpl - Setting new Current Goal
17:13:02 [AWT-EventQueue-0] INFO  SCABlackboard - New current goal event occured.
New Current Goals: Bank/WithdrawalCheckORCreditBank/CreditAccount
17:13:02 [AWT-EventQueue-0] INFO  PolicyManagerImpl - Setting new Verification Set
17:13:02 [AWT-EventQueue-0] INFO  PolicyManagerImpl - ##### Presentation of
Verification SubSets for each Current Goal #####
17:13:02 [AWT-EventQueue-0] INFO  PolicyManagerImpl - Verification SubSet for
current Goal: Bank/WithdrawalCheckORCreditBank/CreditAccount
OR{
Bank/WithdrawalCheck
CreditBank/CreditAccount
}
17:13:02 [AWT-EventQueue-0] INFO  PolicyManagerImpl - Done Verfication SubSet for
current Goal: Bank/WithdrawalCheckORCreditBank/CreditAccount
17:13:02 [AWT-EventQueue-0] INFO  PolicyManagerImpl - ##### Presentation of
Verification Policies for each Goal independently #####
17:13:02 [AWT-EventQueue-0] INFO  PolicyManagerImpl - Verifying Annotations of
CreditBank/CreditAccount
17:13:02 [AWT-EventQueue-0] INFO  PolicyManagerImpl - ### Checking Pre Action
Annotations for CreditBank/CreditAccount ###
Remove Files Pre Action performed
Unset Environment Variables Pre Action performed
Backup Files Pre Action Performed
Restart Services Pre Action performed
17:13:02 [AWT-EventQueue-0] INFO  PolicyManagerImpl - ### Checking Verifiable
Annotations for CreditBank/CreditAccount ###
17:13:02 [AWT-EventQueue-0] INFO  PolicyManagerImpl - Verification of Intents was
successful
17:13:02 [AWT-EventQueue-0] INFO  PolicyManagerImpl - ### Checking Post Action
Annotations for CreditBank/CreditAccount ###
Setup Environmental Variables Post Action performed
Generate Conf Files Post Action performed
Start Services Post Action performed
17:13:02 [AWT-EventQueue-0] INFO  PolicyManagerImpl - Done Verifying Annotations
of CreditBank/CreditAccount
17:13:02 [AWT-EventQueue-0] INFO  PolicyManagerImpl - Verifying Annotations of
```

```
Bank/WithdrawalCheck
17:13:02 [AWT-EventQueue-0] INFO  PolicyManagerImpl - ### Checking Pre Action
Annotations for Bank/WithdrawalCheck ###
Remove Files Pre Action performed
Unset Environment Variables Pre Action performed
Backup Files Pre Action Performed
Restart Services Pre Action performed
17:13:02 [AWT-EventQueue-0] INFO  PolicyManagerImpl - ### Checking Verifiable
Annotations for Bank/WithdrawalCheck ###
17:13:02 [AWT-EventQueue-0] INFO  PolicyManagerImpl - Verification of Intents was
successful
17:13:02 [AWT-EventQueue-0] INFO  PolicyManagerImpl - ### Checking Post Action
Annotations for Bank/WithdrawalCheck ###
Setup Environmental Variables Post Action performed
Generate Conf Files Post Action performed
Start Services Post Action performed
17:13:02 [AWT-EventQueue-0] INFO  PolicyManagerImpl - Done Verifying Annotations
of Bank/WithdrawalCheck
```

Παρατηρούμε ότι η λειτουργία του περιβάλλοντος-πλαισίου είναι όπως έχει παρουσιασθεί στα προηγούμενα κεφάλαια. Αρχικά ο μαυροπίνακας (SCABlackboard) ενημερώνεται (το ρόλο της υποδομής παρακολούθησης στην υλοποίηση αυτή, παίζουν οι Ακροατές – Listeners των Ενεργειών των κουμπιών των μενού) για την ύπαρξη γεγονότος ενημέρωσης του δομοστοιχείου CreditBank. Έπειτα ενημερώνεται η υπηρεσία ενεργοποίησης στόχων η οποία θέτει τους νέους τρέχοντες στόχους, στο παράδειγμά μας είναι μόνο ένας, ο εικονικός στόχος τύπου Ή αποσύνθεση G_3. Αυτός είναι ο νέος τρέχοντας στόχος, καθότι είναι ο πατέρας – στόχος του στόχου που αντιστοιχεί στην υπηρεσία *CreditAccount* που προσέφερε το δομοστοιχείο που ενημερώνεται. Έπειτα, ο διαχειριστής πολιτικών, υπολογίζει τους πιθανούς τρόπους με τους οποίους μπορεί αυτός ο νέος στόχος να επαληθευτεί, λαμβάνοντας υπόψη το υποδέντρο του τρέχοντα στόχου. Όπως παρατηρούμε, δυο επιλογές επαλήθευσης του τρέχοντα στόχου υπάρχουν, αφενός επαλήθευση του στόχου G_4 ή επαλήθευση του στόχου G_5. Αυτό είναι αναμενόμενο, αφού ο τρέχοντας στόχος είναι σύνθετος τύπου Ή αποσύνθεση με υποστόχους τους G_4 και G_5.

Στο σημείο αυτό, το περιβάλλον-πλαίσιο έχει υποδείξει ποιες πολιτικές επαλήθευσης στόχων θα ενεργοποιηθούν. Συγκεκριμένα, θα ενεργοποιηθούν οι πολιτικές επαλήθευσης για τους στόχους G_4 και G_5. Οι πολιτικές επαλήθευσης αυτές, θα προσπαθούν να επαληθεύσουν τον στόχο επ' άπειρον εκτός αν υπάρξει κάποιο δεδομένο που επαληθεύει ή όχι τον στόχο. Στην συνέχεια, το περιβάλλον-πλαίσιο υποδεικνύει την σειρά και την μέθοδο επαλήθευσης των επισημειώσεων καθενός από τους στόχους που παρουσιάσθηκαν στις εναλλακτικές περιπτώσεις επαλήθευσης του τρέχοντα στόχου. Η πληροφορία αυτή, φαίνεται στην συνέχεια της εξόδου του περιβάλλοντος-πλαισίου όπου παρατίθενται με την σειρά που γίνεται η επαλήθευση, οι επισημειώσεις που θα γίνει προσπάθεια να επαληθευτούν. Όπως έχει ήδη αναφερθεί η διαδικασία αυτή περιλαμβάνει πρώτα την εκτέλεση –

επαλήθευση των πριν την επαλήθευση ενεργειών, έπειτα την επαλήθευση των επαληθεύσιμων επισημειώσεων και τέλος την εκτέλεση – επαλήθευση των Μετά την Επαλήθευση Ενεργειών. Αυτή η διαδικασία ακολουθείται για καθένα από τους στόχους G_4 και G_5. Κλείνοντας το παράδειγμα αυτό, παραθέτουμε το αρχείο *CreditBankOK.composite*, τον σωστό αντικαταστάτη του CreditBank δομοστοιχείου.

```
<?xml version="1.0" encoding="UTF-8"?>
<sca:composite xmlns:sca="http://www.osoa.org/xmlns/sca/1.0" name="CreditBankOK"
targetNamespace="http://eclipse.org/ATMWebServiceSCA/src/newComponents/CreditBankO
K">
  <sca:component name="CreditBank">
    <sca:service name="CreditAccount" requires="confidentiality integrity"/>
  </sca:component>
</sca:composite>
```

Αξίζει να σημειωθεί ότι για την διατήρηση των αρχείων *CreditBankOK.composite* και *CreditBankErroneous.composite* (χρησιμοποιείται στο επόμενο παράδειγμα), έχει επιλεγεί να ενθυλακώνεται η περιγραφή του δομοστοιχείου σε ένα εικονικό Σύνθετο Δομοστοιχείο.

6.4.3 Σενάριο Προβληματικής Ανανέωσης Δομοστοιχείου

Η ίδια διαδικασία ακολουθείται όπως πριν, μόνο που στην περίπτωση αυτή, φορτώνεται ως αντικαταστάτης του δομοστοιχείου CreditBank, ένα δομοστοιχείο που δεν είναι συμβατό με το περιβάλλον εκτέλεσης SCA. Πιο συγκεκριμένα, το νέο αυτό δομοστοιχείο απαιτεί την ικανοποίηση της πρόθεσης *authentication2* η οποία δεν προσφέρεται από την παρούσα διαμόρφωση του περιβάλλοντος εκτέλεσης SCA. Το προβληματικό δομοστοιχείο μοντελοποιείται στο παρακάτω xml αρχείο, *CreditBankErroneous.composite*.

```
<?xml version="1.0" encoding="UTF-8"?>
<sca:composite xmlns:sca="http://www.osoa.org/xmlns/sca/1.0"
name="creditBankErroneous"
targetNamespace="http://eclipse.org/ATMWebServiceSCA/src/newComponents/creditBankE
rroneous">
  <sca:component name="CreditBank">
    <sca:service name="CreditAccount" requires="confidentiality integrity
authentication2"/>
  </sca:component>
</sca:composite>
```

Παρατηρούμε την απαίτηση για την πρόθεση authentication2, η οποία δεν υπάρχει στην διαμόρφωση του περιβάλλοντος εκτέλεσης SCA (βλ. αρχείο definitions.xml πιο πάνω). Είναι προφανές ότι δεν είναι σωστό να χρησιμοποιηθεί αυτό το δομοστοιχείο κατά την ενημέρωση, καθότι επιζητά διαφορετικές προδιαγραφές ποιότητας της υπηρεσίας – QoS από ότι μπορεί να

προσφέρει το περιβάλλον εκτέλεσης SCA. Η έξοδος του περιβάλλοντος-πλαισίου στην συγκεκριμένη περίπτωση, παρουσιάζεται πιο κάτω.

```
17:38:40 [AWT-EventQueue-0] DEBUG SCABlackboard - Update Component event for
CreditBank occured
17:38:40 [AWT-EventQueue-0] DEBUG ActuatingServiceImpl - Setting new Current Goal
17:38:40 [AWT-EventQueue-0] INFO  SCABlackboard - New current goal event occured.
New Current Goals: Bank/WithdrawalCheckORCreditBank/CreditAccount
17:38:40 [AWT-EventQueue-0] INFO  PolicyManagerImpl - Setting new Verification Set
17:38:40 [AWT-EventQueue-0] INFO  PolicyManagerImpl - ##### Presentation of
Verification SubSets for each Current Goal #####
17:38:40 [AWT-EventQueue-0] INFO  PolicyManagerImpl - Verification SubSet for
current Goal: Bank/WithdrawalCheckORCreditBank/CreditAccount
OR{
Bank/WithdrawalCheck
CreditBank/CreditAccount
}
17:38:40 [AWT-EventQueue-0] INFO  PolicyManagerImpl - Done Verfication SubSet for
current Goal: Bank/WithdrawalCheckORCreditBank/CreditAccount
17:38:40 [AWT-EventQueue-0] INFO  PolicyManagerImpl - ##### Presentation of
Verification Policies for each Goal independently #####
17:38:40 [AWT-EventQueue-0] INFO  PolicyManagerImpl - Verifying Annotations of
CreditBank/CreditAccount
17:38:40 [AWT-EventQueue-0] INFO  PolicyManagerImpl - ### Checking Pre Action
Annotations for CreditBank/CreditAccount ###
Remove Files Pre Action performed
Unset Environment Variables Pre Action performed
Backup Files Pre Action Performed
Restart Services Pre Action performed
17:38:40 [AWT-EventQueue-0] INFO  PolicyManagerImpl - ### Checking Verifiable
Annotations for CreditBank/CreditAccount ###
17:38:40 [AWT-EventQueue-0] INFO  PolicyManagerImpl - Missing authentication2
intent for SCA Runtime
17:38:40 [AWT-EventQueue-0] INFO  PolicyManagerImpl - Done Verifying Annotations
of CreditBank/CreditAccount
17:38:40 [AWT-EventQueue-0] INFO  PolicyManagerImpl - Verifying Annotations of
Bank/WithdrawalCheck
17:38:40 [AWT-EventQueue-0] INFO  PolicyManagerImpl - ### Checking Pre Action
Annotations for Bank/WithdrawalCheck ###
Remove Files Pre Action performed
Unset Environment Variables Pre Action performed
Backup Files Pre Action Performed
Restart Services Pre Action performed
17:38:40 [AWT-EventQueue-0] INFO  PolicyManagerImpl - ### Checking Verifiable
Annotations for Bank/WithdrawalCheck ###
17:38:40 [AWT-EventQueue-0] INFO  PolicyManagerImpl - Verification of Intents was
successful
17:38:40 [AWT-EventQueue-0] INFO  PolicyManagerImpl - ### Checking Post Action
Annotations for Bank/WithdrawalCheck ###
Setup Environmental Variables Post Action performed
Generate Conf Files Post Action performed
Start Services Post Action performed
17:38:40 [AWT-EventQueue-0] INFO  PolicyManagerImpl - Done Verifying Annotations
of Bank/WithdrawalCheck
```

Παρατηρούμε ότι στο σενάριο αυτό, η έξοδος είναι πανομοιότυπη με ότι παρουσιάσθηκε στην παράγραφο 6.4.2, με την διαφορά ότι στην περίπτωση των επαληθεύσιμων επισημειώσεων για τον

στόχο G_5, λόγω απουσίας της πρόθεσης authentication2 που επιζητά το νέο δομοστοιχείο, αναφέρεται ως ανεπιτυχής.

Κεφάλαιο 7: Επίλογος

7.1 Συμπεράσματα

Η παρούσα διπλωματική, φιλοδοξούσε να προτείνει ένα περιβάλλον-πλαίσιο για τον πριν την χρήση έλεγχο υπηρεσιοκεντρικών συστημάτων λογισμικού και πιο συγκεκριμένα συστημάτων λογισμικού που είχαν μοντελοποιηθεί με το πρότυπο SCA. Η εκπόνηση της διπλωματικής αυτής, κινήθηκε σε τρεις βασικούς άξονες.

Πρώτος άξονας, ήταν ο σχεδιασμός της αρχιτεκτονικής ενός περιβάλλοντος-πλαισίου η οποία χρησιμοποιώντας τεχνικές αυτόνομων συστημάτων, να μπορεί να προσφέρει επαλήθευση της δομικής και λειτουργικής διαμόρφωσης υπηρεσιοκεντρικών συστημάτων, μέσω πολιτικών. Η αρχιτεκτονική που παρουσιάζεται περιλαμβάνει όλα τα απαραίτητα δομικά στοιχεία τα οποία χρειάζονται σε ένα αυτόνομο σύστημα λογισμικού, ώστε να προσφέρει προσαρμοστικότητα στις αλλαγές του υπό έλεγχο συστήματος λογισμικού. Η χρήση των αρχιτεκτονικών τεχνοτροπιών του Μαυροπίνακα και της Έμμεσης Κλήσης αποσυνδέει τα δομοστοιχεία μεταξύ τους και προσφέρει ελευθερία στην υλοποίηση καθενός, με τα όποια πλεονεκτήματα στην επεκτασιμότητα αυτό συνεπάγεται. Ο χρήστης του περιβάλλοντος-πλαισίου, σεβόμενος τον ορισμό των διαπροσωπιών μεταξύ των δομοστοιχείων, όπως έχουν παρουσιασθεί, μπορεί να επεκτείνει το περιβάλλον-πλαίσιο, ώστε να ανταποκρίνεται στις ανάγκες ενός διαφορετικού προβλήματος.

Δεύτερος άξονας, υπήρξε ο σχεδιασμός ενός μοντέλου δεδομένων που θα χρησιμοποιείται από το περιβάλλον-πλαίσιο αυτό (ή όποιο άλλο), με το οποίο να μπορεί να γίνει μοντελοποίηση των απαιτήσεων ενός συστήματος λογισμικού, υπό την οπτική γωνία των πολιτικών που θα τις επαληθεύσουν. Στον άξονα αυτό, εντάσσεται το Μοντέλο Δέντρου Στόχων που παρουσιάσθηκε στο κεφάλαιο 4. Η μοντελοποίηση μέσω μιας δενδρικής δομής, προσφέρει την δυνατότητα μοντελοποίησης εξαρτήσεων μεταξύ των κόμβων του δένδρου, ενώ αποτελεί συμπληρωματική τάση στον τομέα της Μηχανικής Απαιτήσεων, της αντικειμενοστραφούς μοντελοποίησης. Το Μοντέλο δέντρου στόχων που παρουσιάσθηκε, έχει εμπλουτιστεί με επισημειώσεις που αποτελούν έναν

ειδικό τρόπο δόμησης και επισύναψης της επιπλέον πληροφορίας που χρειάζεται το περιβάλλον-πλαίσιο για την επαλήθευση των στόχων. Η χρήση των επισημειώσεων όμως μπορεί να είναι γενικότερη, καθώς μπορεί να περιέχουν διαφόρων ειδών πληροφορίες για συστήματα λογισμικού που μπορεί να φανούν χρήσιμες στους τομείς της Τεχνολογίας Λογισμικού και ιδιαίτερα της Συντήρησης Λογισμικού.

Τρίτος άξονας υπήρξε η προδιαγραφή μεθόδων και αλγορίθμων που μπορούν να χρησιμοποιηθούν για την μοντελοποίηση υπηρεσιοκεντρικών συστημάτων λογισμικού με δέντρα στόχων, αλλά και για την επαλήθευση των επισημειώσεων. Στον τομέα αυτό, εντάσσεται ο χωρισμός του περιβάλλοντος-πλαισίου σε Στρώματα Λειτουργικότητας, με την δυνατότητα να επεκταθεί κάποιο στρώμα και να προσφέρει νέες δυνατότητες στο περιβάλλον-πλαίσιο. Επίσης, μεγάλης σημασίας είναι και η δυνατότητα εξαγωγής του δέντρου στόχων και των επισημειώσεών του, αυτόματα από αρχεία ρυθμίσεων. Στην παρούσα διπλωματική, παρουσιάσθηκαν απλοί αλγόριθμοι για την εξαγωγή του δέντρου στόχων από το μοντέλο SCA ενός υπηρεσιοκεντρικού συστήματος λογισμικού. Η πλήρης αυτοματοποιημένη διαδικασία εξαγωγής του δέντρου στόχων, ή έστω του βασικού μέρους του, αποτελεί θέμα μείζονος σημασίας για την βιωσιμότητα και την χρήση των δέντρων στόχων ως εργαλεία της Μηχανικής Απαιτήσεων.

Από την εκπόνηση της παρούσας διπλωματικής, καταδεικνύεται η σημασία της χρήσης μεθόδων αυτόνομων συστημάτων, στην συντήρηση υπηρεσιοκεντρικών και όχι μόνο συστημάτων λογισμικού. Λόγω της πολυπλοκότητας που αποκτούν τα σύγχρονα υπηρεσιοκεντρικά συστήματα λογισμικού και των απαιτήσεων που υπάρχουν για συνεχή και αδιάκοπη λειτουργία τους, είναι σημαντικό να υπάρχουν τεχνικές πρόβλεψης λαθών, έστω στις πιο βασικές – στοιχειώδεις λειτουργίες που μπορεί να επιτελεστούν κατά το στάδιο της συντήρησης λογισμικού. Μείζονος σημασίας, είναι επίσης ο τρόπος μοντελοποίησης σύγχρονων συστημάτων λογισμικού, να μπορεί να γίνει όσο το δυνατόν πιο αυτοματοποιημένα ώστε να μπορεί να λαμβάνει μεγάλο όγκο πληροφορίας από μεικτές πηγές.

Σημαντικό συμπέρασμα που απορρέει από την παρούσα διπλωματική, είναι πως ακόμα και στην περίπτωση αρχιτεκτονικών οι οποίες επιτρέπουν την αποσύμπλεξη της υλοποίησης των διαφορετικών δομοστοιχείων ενός συστήματος λογισμικού (όπως είναι το υπηρεσιοκεντρικό πρότυπο) υπάρχουν σχέσεις και εξαρτήσεις μεταξύ των δομοστοιχείων, λόγω χρήσης σε κάποια, των αποτελεσμάτων της λειτουργίας άλλων. Αυτό το συμπέρασμα, οδηγεί ενδεχομένως, στην χρήση προσανατολισμένων σε στόχους τεχνικών μοντελοποίησης όπως είναι το δέντρο στόχων που παρουσιάσθηκε στην παρούσα διπλωματική, για την μοντελοποίηση των απαιτήσεων συστημάτων λογισμικού με χαμηλή σύζευξη.

7.2 Επεκτάσεις

Κάθε άξονας στον οποίο στηρίχθηκε αυτή η διπλωματική, χρήζει επεκτάσεων αρκετά σημαντικών για την εξέλιξη του τομέα της Συντήρησης Λογισμικού.

Η πιο σημαντική επέκταση, αφορά στην πλήρως αυτοματοποιημένη εξαγωγή του δέντρου στόχων και των επισημειώσεών του από αρχεία ρυθμίσεων και εγκαταστατών. Μέλλουσα δουλειά, θα μπορούσε να ορίσει ένα μοντέλο δεδομένων το οποίο να απεικονίζει τις πληροφορίες που μπορούν να εξαχθούν από αρχεία ρυθμίσεων ή εγκαταστατών και έπειτα να ορίσει πώς από στιγμιότυπα του μοντέλου αυτού, θα μπορούσε κανείς να εξάγει αυτόματα πληροφορία όπως οι επισημειώσεις ενέργειας που παρουσιάζονται στην παρούσα διπλωματική. Κάτι τέτοιο θα είχε τεράστια θετική συνεισφορά στην μηχανική απαιτήσεων και στον τομέα συντήρησης λογισμικού.

Μια ακόμα επέκταση αφορά στις βελτιστοποιήσεις που μπορούν να γίνουν στους αλγόριθμους. Ιδιαίτερα οι αλγόριθμοι επαλήθευσης πολιτικών αποτελούν ένα είδος στενωπού απόδοσης για το όλο σύστημα λογισμικού, καθότι αποτελούν όλη την λογική της προσαρμοστικότητας του περιβάλλοντος-πλαισίου, σε αλλαγές της κατάστασης του υπό έλεγχο συστήματος λογισμικού. Είναι αρκετά σημαντικό συνεπώς, οι αλγόριθμοι αυτοί να γίνουν όσο το δυνατόν αποδοτικότεροι, ώστε να μειωθεί ο χρόνος αντίδρασης του περιβάλλοντος-πλαισίου στις αλλαγές κατάστασης του υπό έλεγχο συστήματος. Στα πλαίσια μιας τέτοιας επέκτασης, θα μπορούσε να ελεγχθεί και ο ταυτοχρονισμός των αλγορίθμων, πώς μπορούν δηλαδή να χωριστούν σε υποπροβλήματα τα οποία να λυθούν από ξεχωριστές επεξεργαστικές μονάδες. Κάτι τέτοιο, επίσης θα συμβάλλει στην ταχύτητα απόκρισης του περιβάλλοντος-πλαισίου.

Ίσως η πιο άμεση και μελλοντική επέκταση που μπορεί να γίνει στην παρούσα διπλωματική είναι η επέκταση του περιβάλλοντος-πλαισίου, καθώς και του μοντέλου δέντρου στόχων, ώστε να περιλαμβάνουν το πλήρες πρότυπο SCA. Μια τέτοια επέκταση, θα καθιστούσε το παραχθέν έργο της παρούσας διπλωματικής ένα πολύ σημαντικό και χρήσιμο εργαλείο για την συντήρηση και έλεγχο υπηρεσιοκεντρικών συστημάτων λογισμικού, που ακολουθούν το προγραμματιστικό παράδειγμα SCA. Τέτοια επέκταση, θα έπρεπε να προσφέρει μέριμνα για τα εξής δομήματα SCA: componentType, interface, policySetReference, wsp:Policy, wsp:PolicyAttachment. Επίσης, επεκτάσεις που σκοπό έχουν την υλοποίηση υποστήριξης μεγαλύτερου μέρους του προτύπου SCA, θα πρέπει να λάβουν υπόψη τους τα στάδια δέσμευσης πλατφόρμας (binding) και δέσμευσης υλοποίησης (implementation).

Τέλος, μελλοντική δουλειά μπορεί να επικεντρωθεί στην ενσωμάτωση μιας πιο ολοκληρωμένης υλοποίησης του περιβάλλοντος-πλαισίου, σε περιβάλλοντα εκτέλεσης SCA που υπάρχουν ήδη, όπως είναι το Apache Tuscany και το FraSCAti.

Βιβλιογραφία

[1] S. R. Schach, *Object-Oriented and Classical Software Engineering*. McGraw-Hill Pub. Co., 2001.

[2] 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.

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

[4] E. Foundation. (2009, Oct.) Eclipse IDE. [Online]. http://www.eclipse.org/

[5] Wikipedia. (2009, Oct.) Meta Object Facility - Wikipedia. [Online]. http://en.wikipedia.org/wiki/Meta-Object_Facility

[6] O. M. Group. (2009, Oct.) OMG - MOF. [Online]. http://www.omg.org/mof

[7] Wikipedia. (2009, Oct.) XML Metadata Interchange - Wikipedia. [Online]. http://en.wikipedia.org/wiki/XML_Metadata_Interchange

[8] O. M. Group. (2009, Oct.) OMG - Specifications. [Online]. http://www.omg.org/technology/documents/modeling_spec_catalog.htm

[9] Wikipedia. (2009, Oct.) Unified Modeling Language - Wikipedia. [Online]. http://en.wikipedia.org/wiki/Unified_Modeling_Language

[10] Wikipedia. (2009, Oct.) eXtended Markup Language - Wikipedia. [Online].

http://en.wikipedia.org/wiki/XML

[11] Wikipedia. (2009, Oct.) Boolean algebra - Wikipedia. [Online]. http://en.wikipedia.org/wiki/Boolean_algebra_%28logic%29

[12] Y. Yu, et al., "Reverse Engineering Goal Models from Legacy Code," *Requirements Engineering, IEEE International Conference on*, pp. 363-372, 2005.

[13] J. Mylopoulos and Y. Wang, "Monitoring and Diagnosing Software Using Statecharts".

[14] OSOA. (2007, Mar.) Service Component Architecture Home. [Online]. http://www.osoa.org/display/Main/Service+Component+Architecture+Home

[15] OSOA. (2007, Mar.) Service Component Architecture Specifications. [Online]. http://www.osoa.org/download/attachments/35/SCA_Policy_Framework_V100.pdf?version=1

[16] OSOA. (2007, Mar.) Service Component Architecture Specifications. [Online]. http://www.osoa.org/download/attachments/35/SCA_AssemblyModel_V100.pdf?version=1

[17] A. S. Foundation. (July,) Apache log4j Homepage. [Online]. http://logging.apache.org/log4j/1.2/index.html

[18] W3C. (May,) W3C - XML Schema Description. [Online]. http://www.w3.org/XML/Schema

[19] S. Microsystems. (May,) Java Architecture for XML Binding. [Online]. http://java.sun.com/developer/technicalArticles/WebServices/jaxb/

[20] A. S. Foundation. (July,) Apache JAXME2. [Online]. http://ws.apache.org/jaxme/

[21] R. C. Bjork. An Example of Object-Oriented Design: An ATM Simulation. [Online]. http://www.math-cs.gordon.edu/courses/cs211/ATMExample/

[22] R. Calinescu, "Implementation of a Generic Autonomic Framework," *Autonomic and Autonomous Systems, International Conference on*, vol. 0, pp. 124-129, 2008.

[23] S. A. Cook, "The complexity of theorem-proving procedures," in *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, Shaker Heights, Ohio, United States, 1971, pp. 151-158.

[24] L. Ardissono, et al., "Enhancing Web Services with Diagnostic Capabilities," *ECOWS '05: Proceedings of the Third European Conference on Web Services*, p. 182, 2005.

[25] Y. Wang, S. A. McIlraith, Y. Yu, and J. Mylopoulos, "An automated approach to monitoring and diagnosing requirements," *ASE '07: Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, pp. 293-302, 2007.

[26] A. Razavi and K. Kontogiannis, "Pattern and Policy Driven Log Analysis for Software Monitoring," *COMPSAC '08: Proceedings of the 2008 32nd Annual IEEE International Computer Software and Applications Conference*, pp. 108-111, 2008.

[27] A. R. Haydarlou, M. A. Oey, B. J. Overeinder, and F. M. T. Brazier, "Use Case Driven Approach to Self-Monitoring in Autonomic Systems," *ICAS '07: Proceedings of the Third International Conference on Autonomic and Autonomous Systems*, p. 50, 2007.

[28] L. Liu, S. Thanheiser, and H. Schmeck, "A Reference Architecture for Self-organizing Service-Oriented Computing," *Lectures in Computer Science*, no. 4934, pp. 205-219, 2008.

[29] D. Gracanin, S. A. Bohner, and M. Hinchey, "Towards a Model-Driven Architecture for Autonomic Systems," *ECBS '04: Proceedings of the 11th IEEE International Conference and Workshop on Engineering of Computer-Based Systems*, p. 500, 2004.