ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Περιβάλλον-Πλαίσιο Ανάλυσης Συμμόρφωσης Πολιτικών Λειτουργίας Υπηρεσιοκεντρικών Συστημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Κωνσταντίνου Αγγελόπουλου

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

Αθήνα, Ιούλιος 2011

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

Περιβάλλον-Πλαίσιο Ανάλυσης Συμμόρφωσης Πολιτικών Λειτουργίας Υπηρεσιοκεντρικών Συστημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Κωνσταντίνου Αγγελόπουλου

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

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 27η Ιουλίου 2011.

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

Αθήνα, Ιούλιος 2011

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ευχαριστίες

Θέλω αρχικά να ευχαριστήσω θερμά τον επιβλέποντα της διπλωματικής μου εργασίας, Αναπληρωτή Καθηγητή κύριο Κώστα Κοντογιάννη, για τη βοήθεια, τη στήριξη και την υπομονή του κατά την εκπόνησή της. Υπήρξε ο άνθρωπος, ο οποίος με ενέπνευσε να ασχοληθώ με τα ερευνητικά θέματα της Τεχνολογίας Λογισμικού, ενώ με τις γνώσεις και την πολυετή του πείρα σε αυτόν τον κλάδο με καθοδήγησε με τρόπο τέτοιο, ώστε να αποκομίσω τα μέγιστα.

Θα ήθελα επίσης να ευχαριστήσω τους συμφοιτητές μου, που κατά την πενταετή μου φοίτηση στάθηκαν δίπλα μου ως συνεργάτες, συνάδελφοι και φίλοι. Τους ευχαριστώ για τις γνώσεις που αντάλλαξαν μαζί μου και για όλες τις όμορφες και δύσκολες στιγμές που μοιραστήκαμε όλα αυτά τα χρόνια. Η συμβολή τους στην διαμόρφωση του ακαδημαϊκού μου υποβάθρου αλλά και προσωπικού χαρακτήρα υπήρξε αμέριστη.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου που όλα αυτά τα χρόνια κατέβαλλε κάθε δυνατή προσπάθεια, ώστε να πραγματοποιήσω τα όνειρα και τις φιλοδοξίες μου.

Περίληψη

Στην παρούσα διπλωματική εργασία αναπτύσσεται ένα περιβάλλον - πλαίσιο (framework) το οποίο έχει σαν σκοπό α) τη μοντελοποίηση επιχειρησιακών πολιτικών συμμόρφωσης (bussiness compliance policies) Υπηρεσιοκεντρικών συστημάτων και β) τον έλεγχο συμβατότητας του συστήματος με αυτές τις πολιτικές συμμόρφωσης. Συγκεκριμένα, αναλυτές συστημάτων θα μπορούν να μοντελοποιήσουν πολιτικές συμμόρφωσης χρησιμοποιώντας αντικειμενοστραφή μεταμοντέλα (MOF), ενώ ο έλεγχος συμβατότητας γίνεται με αλγορίθμους που αναλύουν τα αρχεία καταγραφής γεγονότων κατά την λειτουργία του συστήματος και τα συνδυάζουν με αυτές τις πολιτικές συμμόρφωσης, ώστε να εξαχθεί η πιθανότητα ότι το παρόν σύστημα τις επαληθεύει .

Σε πιο τεχνικό επίπεδο, το Υπηρεσιοκεντρικό Σύστημα αποτελείται από ένα σύνολο υπηρεσιών ιστού (web – services) που ενορχηστρώνονται χρησιμοποιώντας τη γλώσσα BPEL, παράγοντας έτσι ολοκληρωμένες επιχειρησιακές λογικές. Η ανάπτυξη των web-services γίνεται στο περιβάλλον Netbeans. Οι υπηρεσίες παρατάσσονται στον Glassfish Server.

Κατά το σενάριο λειτουργίας του συστήματος που προτείνεται, θεωρούμε την ύπαρξη ενός περιβάλλοντος παρακολούθησης, το οποίο παράγει αρχεία καταγραφής γεγονότων (log files). Τα αρχεία καταγραφής γεγονότων αναλύονται και φιλτράρονται ανάλογα με το "προφίλ παρακολούθησης" που επιθυμεί ο χρήστης. Τα δεδομένα που εξάγονται από τη διαδικασία αυτή μετατρέπονται σε μορφή λογικής πρώτης τάξης και δημιουργούν μια λογική βάση δεδομένων με κατηγορήματα, που σχετίζονται με τις ζητούμενες πολιτικές συμμόρφωσης.

Τέλος, χρησιμοποιώντας τεχνικές λογικού συμπερασμού, που βασίζονται στις θεωρίες Markov Logic Networks και Markov Logic Learning, μπορούμε να απαντήσουμε σε ερωτήματα που αφορούν στο ποσοστό που ικανοποιείται η ζητούμενη πολιτική συμμόρφωσης. Τα αποτελέσματα που παράγονται μέσω του εργαλείου Alchemy, το οποίο υλοποιεί τη διαδικασία του λογικού συμπερασμού, βασίζονται α) σε ένα γράφο αλληλεπίδρασης κατηγορημάτων, τα οποία αναφέρονται στη σχετική βιβλιογραφία ως Markov Logic Networks και β) στην αυξητική μάθηση του συστήματος για τον προσδιορισμό των ποσοστών και των πιθανοτήτων ικανοποιησιμότητας των κατηγορημάτων. Η χρήση ενός τέτοιου συμπερασματικού μηχανισμού, σε σχέση με απλές τεχνικές λογικής πρώτης τάξεως, μας διευκολύνει στο γεγονός ότι είναι δυνατόν να παράγουμε αποτελέσματα ακόμα και όταν δεν είναι πάντα διαθέσιμη σε όλο της το σύνολο η πληροφορία που χρειαζόμαστε για να εξακριβώσουμε την ορθή λειτουργία του συστήματος σε σχέση με τις πολιτικές συμμόρφωσης που έχουν τεθεί από τον χρήστη.

Λέξεις Κλειδιά: <<υπηρεσιοκεντρικό σύστημα, πολιτικές συμμόρφωσης, Μαρκοβιανό Λογικό Δίκτυο, περιβάλλον εξόρυξης γεγονότων, εξακρίβωση ορθής λειτουργίας >>

Abstract

The goal of this diploma thesis is to propose a framework that allows for a) the modeling of business compliance policies of Service Oriented systems and b) controlling the conformance of these systems to the predefined compliance policies. More specifically, the proposed framework provides system analysts, the ability to model compliance policies through object oriented metamodels (MOF), while the conformance to the compliance policies is evaluated by algorithms that analyze the system's log files. A reasoning procedure is then used to estimate the likelihood of the system's conformance to the compliance policies.

On a more technical level, a Service Oriented system is composed of Web – Services, which are orchestrated by using the web service orchestration language BPEL. The result of the Web – Service orchestration leads to the production of integrated business solutions. For the purposes of this diploma thesis, we have used the IDE Netbeans to develop our Web – Services and the Glassfish Server to deploy them.

We suppose that during the execution of the system, which we are trying to evaluate, a monitoring mechanism produces log files. These log files are analyzed and filtered according to the monitoring profile that is set by the user. Consequently, the log files are interpreted into first order ground predicates, composing thus a knowledge base related to the compliance policies

Finally, we are using reasoning techniques based on the theories of Markov Logic Networks and Markov Logic Learning, in order to evaluate the system's conformance to the predefined compliance policies. The result of the evaluation procedure is produced by the reasoning tool Alchemy and is based on a) an interaction graph of predicates, known as Markov Logic Network and b) the weights of the rules, which are calculated through learning. The Alchemy's output is an estimation of the percentage value pertaining to the degree of satisfaction of a given compliance policy that is being evaluated. This way, it is possible to make inferences about the operation of the system, even if the total amount of information required is not fully available.

Keywords: <<SOA, Compliance Policies, Markov Logic Networks, Monitoring, Verification >>

Πίνακας Περιεχομένων

Κεφάλαιο 1: Εισαγωγή 1

1.1 Λογισμικό Προσανατολισμένο στις Υπηρεσίες 1

1.2 Υπηρεσιοκεντρικά Συστήματα και Συμμόρφωση 2

1.3 Περιγραφή του Προβλήματος 3

1.4 Κίνητρο 4

1.5 Συνεισφορά της Διπλωματικής Εργασίας 4

1.6 Οργάνωση του Κειμένου 5

Κεφάλαιο 2: Σχετικές Εργασίες 8

2.1 Εισαγωγή 8

2.2 Μοντελοποίηση και Διαχείριση Επιχειρησιακών Διαδικασιών 8

2.2.1 Business Process Modeling Notation - BPMN 10

2.2.2 Business Process Execution Language – BPEL 17

2.3 MOF/XMI 22

2.4 Περιβάλλοντα Monitoring 24

2.5 Περιβάλλοντα Ελέγχου Συμμόρφωσης 26

2.6 Μοντέλα Στόχων 27

2.7 Περιβάλλοντα Συμπερασμού 30

2.7.1 Λογική Πρώτης Τάξης 30

2.7.2 Μαρκοβιανά Λογικά Δίκτυα 33

Κεφάλαιο 3: Αρχιτεκτονική του Συστήματος 38

3.1 Εισαγωγή 38

3.2 Γενική Επισκόπηση Αρχιτεκτονικής 38

3.3 Περιγραφή Δομικών Στοιχείων 40

3.3.1 Data Manager Component 41

3.3.2 Data Modeler Component 43

3.3.3 Business Logic Modeler Component 44

3.3.4 Compliance Modeler Component 46

3.3.5 Information Integrator Component 47

3.3.6 Strategy Manager Component *48*

3.3.7 UI Component *50*

3.4 Λειτουργικότητα του Συστήματος 51

Κεφάλαιο 4: Εννοιολογικό Μοντέλο **56**

4.1 Εισαγωγή 56

4.2 Εννοιολογικό Μοντέλο Συμμόρφωσης 56

4.2.1 Κλάση BusinessProcess *56*

4.2.2 Κλάση BusinessProcessTask *58*

4.2.3 Κλάση Business Process Transition *58*

4.2.4 Κλάση Operation *58*

4.2.5 Κλάση Service *59*

4.2.6 Κλάση VerifiableEntity *59*

4.2.7 Κλάση Data *59*

4.2.8 Κλάση GroundedLogicExpression *60*

4.2.9 Κλάση DataPool *60*

4.2.10 Κλάση Property *60*

4.2.11 Κλάση CompliancePolicy *61*

4.2.12 Κλάση GoalTree *61*

4.2.13 Κλάση GoalNode *62*

4.2.14 Κλάση AtomicGoal *62*

4.2.15 Κλάση DecompositionGoal *62*

4.2.16 Κλάση AnnotationContainer *62*

4.2.17 Κλάση Annotation *63*

4.2.18 Κλάση StrategyController *63*

4.2.19 Κλάση AnnotationController *63*

Κεφάλαιο 5: Αλγόριθμοι Εξακρίβωσης Συμμόρφωσης **66**

5.1 Εισαγωγή 66

5.2 Αλγόριθμος Εξακρίβωσης Συμμόρφωσης Πολιτικών 68

5.3 Αλγόριθμος Εύρεσης Μονοπατιού Εκτέλεσης 72

Κεφάλαιο 6: Περιπτώσεις Χρήσης **75**

6.1 Εισαγωγή 75

6.2 Περίπτωση Χρήσης 1.....75
6.3 Περίπτωση Χρήσης 2.....83
Κεφάλαιο 7: Συμπεράσματα 96
Κεφάλαιο 8: Βιβλιογραφία.....Σφάλμα! Δεν έχει οριστεί σελιδοδείκτης.

Πίνακας Σχημάτων

Εικόνα 2.2.1: Κύκλος ζωής διαχείρισης επιχειρησιακών διαδικασιών [4] 9
Εικόνα 2.2.2: Παραδείγματα Γεγονότων.....11
Εικόνα 2.2.3: Παραδείγματα Δραστηριοτήτων.....11
Εικόνα 2.2.4: Παραδείγματα Πυλών.....11
Εικόνα 2.2.5: Σύμβολο Ροής 12
Εικόνα 2.2.6: Σύμβολο Ροής Μηνυμάτων.....12
Εικόνα 2.2.7: Σύμβολα Σύνδεσης.....12
Εικόνα 2.2.8: Σύμβολο Ενότητας.....13
Εικόνα 2.2.9: Παράδειγμα χρήσης Ενότητας.....13
Εικόνα 2.2.10: Σύμβολο Υπο-Ενοτήτων.....13
Εικόνα 2.2.11: Παράδειγμα χρήσης Υπο-Ενοτήτων.....13
Εικόνα 2.2.12: Σύμβολο Αντικειμένου Δεδομένων 14
Εικόνα 2.2.13: Σύμβολο Επισημείωσης.....14
Εικόνα 2.2.14: Σύμβολο Ομαδοποίησης.....14
Εικόνα 2.2.15: Παράδειγμα Ιδιωτικής Επιχειρησιακής Διαδικασίας.....15
Εικόνα 2.2.16: Παράδειγμα Δημόσιας Επιχειρησιακής Διαδικασίας 15
Εικόνα 2.2.17: Παράδειγμα Συνεργασίας 16
Εικόνα 2.2.18: Παράδειγμα Χορογραφίας.....17
Εικόνα 2.2.19: Μοντέλο Ενορχήστρωσης.....18
Εικόνα 2.2.20: Μοντέλο Χορογραφίας 19
Εικόνα 2.2.21: Flowchart για την διαδικασία λήψης δανείου.....21
Εικόνα 2.2.22: Γραφική απεικόνιση της διαδικασία δανειοληψία από τον BPEL Designer του Netbeans.....22
Εικόνα 2.2.23: Πηγαίος κώδικας της διεργασίας BPEL 22
Εικόνα 2.3.1: Αρχιτεκτονική Μετα-δεδομένων Τεσσάρων Επιπέδων.....23
Εικόνα 2.6.1: Παράδειγμα απλού μοντέλου στόχων.....28
Εικόνα 2.6.2: Παράδειγμα μοντέλου στόχων με soft goals.....28
Εικόνα 2.6.3: Παράδειγμα σύνδετου μοντέλου στόχων [20] 29
Εικόνα 3.3.1: Ψηφιδικό διάγραμμα του Data Modeler 44
Εικόνα 3.3.2: Ψηφιδικό διάγραμμα του Business Logic Modeler 45
Εικόνα 3.3.3: Ψηφιδικό διάγραμμα του Compliance Modeler.....47
Εικόνα 3.3.4: Το Ψηφιδικό διάγραμμα του Information Integrator 48
Εικόνα 3.3.5: Ψηφιδικό διάγραμμα του Strategy Manager.....49

Εικόνα 3.4.1: Ακολουθιακό Διάγραμμα του Περιβάλλοντος - Πλαισίου 51
Εικόνα 4.2.1: Διάγραμμα Κλάσεων Εννοιολογικού Μοντέλου Συμμόρφωσης.....57
Εικόνα 5.2.1: Διάγραμμα Δραστηριοτήτων του περιβάλλοντος – πλαισίου.....67
Εικόνα 6.2.2: Μοντέλο Στόχων Περίπτωσης Χρήσης 1 78
Εικόνα 6.2.1: Επιχειρησιακή Διαδικασία Κράτησης Εισιτηρίου.....77
Εικόνα 6.3.1: Επιχειρησιακή Διαδικασία Έγκρισης Δανείου 85

Κεφάλαιο 1: Εισαγωγή

1.1 Λογισμικό Προσανατολισμένο στις Υπηρεσίες

Την τελευταία δεκαετία, υπήρξε μια ραγδαία εξέλιξη στις τεχνολογίες του Διαδικτύου οι οποίες επηρέασαν σημαντικά τις σύγχρονες τεχνοτροπίες ανάπτυξης λογισμικού. Η ραγδαία αυτή εξέλιξη, σε συνδυασμό με την τάση της βιομηχανίας λογισμικού για παραγωγή κατανεμημένου λογισμικού οδήγησαν στην ανάπτυξη μιας νέας αρχιτεκτονικής λογισμικού, της Υπηρεσιοκεντρικής Αρχιτεκτονικής (Service Oriented Architecture ή SOA) [1,2].

Η αρχιτεκτονική αυτή έχει ως βασική δομική μονάδα την Υπηρεσία (Service). Ένα Υπηρεσιοκεντρικό σύστημα απαρτίζεται από ένα σύνολο Υπηρεσιών, χαλαρά συνδεμένων μεταξύ τους, η κάθε μία από τις οποίες ενσωματώνει ένα τμήμα της λειτουργικότητας του συστήματος, όπως η αγορά ενός βιβλίου μέσω Διαδικτύου ή η κράτηση ενός αεροπορικού εισιτηρίου. Οι Υπηρεσίες αυτές ανταλλάσουν δεδομένα μεταξύ τους μέσω δικτυακών τεχνολογιών και υλοποιούν έτσι μια ενιαία και ολοκληρωμένη εφαρμογή λογισμικού. Επίσης, κάθε Υπηρεσία χαρακτηρίζεται από ένα Συμβόλαιο Υπηρεσίας (Service Contract) [3] το οποίο διέπει τον τρόπο κλήση της από προγράμματα καταναλωτές (consumer programs) για τη συγκεκριμένη Υπηρεσία. Όλη αυτή η χαλαρή δομή της Υπηρεσιοκεντρικής Αρχιτεκτονικής έχει δώσει μεγάλη ευελιξία στην ανάπτυξη ευρείας κλίμακας συστημάτων λογισμικού, παράλληλα όμως γεννούνται ζητήματα σχετικά με την διασύνδεση των Υπηρεσιών και την οργάνωσή τους.

Όπως αναφέρθηκε παραπάνω τα Υπηρεσιοκεντρικά συστήματα αναπτύσσονται για να υποστηρίξουν ευρείας κλίμακας εφαρμογές, οι οποίες με της σειρά τους συμβάλουν στην αυτοματοποίηση διαφόρων επιχειρησιακών διαδικασιών (business processes) σε πολλές εταιρίες. Η ολοκλήρωση μιας επιχειρησιακής διαδικασίας η οποία υλοποιείται από ένα Υπηρεσιοκεντρικό σύστημα είναι το αποτέλεσμα ενός συνόλου κλήσεων διαφορετικών Υπηρεσιών του συστήματος αυτού. Είναι πολύ σημαντικό επομένως, η διασύνδεση των Υπηρεσιών και αλληλουχία των κλήσεών τους να γίνει με τέτοιο τρόπο ώστε το υλοποιημένο σύστημα να είναι απόλυτα σύμφωνο με τη σχεδίαση των εταιρικών διαδικασιών που υλοποιεί. Το πρόβλημα αυτό είναι γνωστό ως το πρόβλημα της ενορχήστρωσης (orchestration). Η επίλυση αυτού το προβλήματος αποτελεί σημαντικό αντικείμενο έρευνας, με ιδιαίτερα αξιόλογα αποτελέσματα τα τελευταία χρόνια. Ένα από τα αποτελέσματα αυτά είναι και η γλώσσα ενορχήστρωσης Business Process Execution Language (BPEL) [4,5], η οποία περιγράφεται με περισσότερη λεπτομέρεια στο κεφάλαιο 2.

Τα πλεονεκτήματα της Υπηρεσιοκεντρικής Αρχιτεκτονικής αφορούν την ευελιξία των παραγόμενων συστημάτων και την αποδοτική ανταπόκρισή τους στις απαιτήσεις της αγοράς από την πλευρά του κόστους. Αυτό οφείλεται στο γεγονός του ότι τα Υπηρεσιοκεντρικά συστήματα προωθούν την ιδέα της επαναχρησιμοποίησης τόσο σε μακροσκοπικό (Υπηρεσίες), όσο και μικροσκοπικό (ψηφίδες λογισμικού – software components) επίπεδο. Επίσης, έχουν το πλεονέκτημα της εύκολης διασύνδεσης με υπάρχοντα συστήματα (legacy systems) τα οποία έχουν αναπτυχθεί με παλαιότερες τεχνολογίες, παρέχοντας έτσι ένα υψηλό επίπεδο επεκτασιμότητας.

1.2 Υπηρεσιοκεντρικά Συστήματα και Συμμόρφωση

Τα σημερινά επιχειρησιακά περιβάλλοντα υπαγορεύουν στους οργανισμούς να είναι όσο το δυνατόν πιο ευέλικτοι, ώστε να μπορούν να προσαρμόζουν τις επιχειρησιακές διαδικασίες τους στις ραγδαίες εξελίξεις των όρων της αγοράς. Οι εξελίξεις αυτές μπορεί να αφορούν αλλαγές σε κάποιο νομικό πλαίσιο, στις απαιτήσεις των καταναλωτών ή κάποια τεχνολογική καινοτομία που μπορεί να μεταβάλει τη λειτουργία της αγοράς. Τα Υπηρεσιοκεντρικά συστήματα ενσωματώνουν αυτή την ευελιξία στο αρχιτεκτονικό μοντέλο, το οποίο προτείνουν και αυτός είναι ένας λόγος που γίνονται ολοένα και πιο αποδεκτά από τον επιχειρηματικό κόσμο.

Οι επιχειρήσεις προκειμένου να ανταποκριθούν στις αλλαγές που προκύπτουν στα πλαίσια της αγοράς ή των νομικών περιορισμών, καταστρώνουν πολιτικές, τις οποίες πρέπει να ακολουθούν οι επιχειρησιακές διαδικασίες. Ο όρος "συμμόρφωση" αναφέρεται στην ικανοποίηση αυτών των πολιτικών από τις επιχειρησιακές διαδικασίες κατά την εκτέλεσή τους. Ο έλεγχος της συμμόρφωσης αποτελεί θέμα ζωτικής σημασίας, καθώς μία παραβίαση κάποιας επιχειρησιακής πολιτικής μπορεί να επιφέρει μεγάλο κόστος για τη λειτουργία της επιχείρησης. Για τον λόγο αυτό, οι επιχειρήσεις δίνουν μεγάλο βάρος στον έλεγχο αυτό.

Η διαχείριση της συμμόρφωσης γίνεται σε δύο επίπεδα. Το πρώτο επίπεδο είναι αυτό της σχεδίασης της επιχειρησιακής διαδικασίας. Στο στάδιο αυτό, οι επιχειρησιακές πολιτικές μοντελοποιούνται με τέτοιο τρόπο ώστε να καθοδηγούν τη λειτουργία της επιχειρησιακής διαδικασίας. Το δεύτερο επίπεδο είναι αυτό της εκτέλεσης της επιχειρησιακής διαδικασίας. Όταν μια επιχειρησιακή διαδικασία εκτελείται είναι απαραίτητο να παρακολουθείται το επίπεδο συμμόρφωσής της ως προς τις πολιτικές με τις οποίες πρέπει να συμβαδίζει.

Η συνεχής υιοθέτηση των Υπηρεσιοκεντρικών συστημάτων για την αυτοματοποίηση των επιχειρησιακών διαδικασιών οδήγησε στην δημιουργία μεθόδων μοντελοποίησης και ελέγχου συμμόρφωσης. Τα Υπηρεσιοκεντρικά συστήματα προσφέρουν τη δυνατότητα αξιοποίησης

νέων τεχνολογιών για τον έλεγχο της συμμόρφωσης με αξιόπιστο τρόπο. Επίσης, η δομή τους επιτρέπει την εύκολη αναπροσαρμογή του συστήματος, σε περίπτωση που παρατηρηθεί παραβίαση κάποιας επιχειρησιακής πολιτικής.

1.3 Περιγραφή του Προβλήματος

Το πρόβλημα το οποίο προσπαθούμε να επιλύσουμε με την παρούσα διπλωματική εργασία αφορά τη σχεδίαση και την υλοποίηση ενός περιβάλλοντος – πλαισίου (framework) το οποίο θα έχει την ικανότητα να αποφαίνεται για τα επίπεδα συμμόρφωσης ενός Υπηρεσιοκεντρικού συστήματος με διάφορες επιχειρησιακές πολιτικές. Όπως κάθε λογισμικό το οποίο υποστηρίζει ένα σύνολο λειτουργιών μιας επιχείρησης, έτσι και τα Υπηρεσιοκεντρικά συστήματα πρέπει να είναι συμβατά με τις επιχειρησιακές πολιτικές των επιχειρησιακών διαδικασιών που υλοποιούν. Η ιδιαιτερότητα που έχουν τα Υπηρεσιοκεντρικά συστήματα σε σχέση με τα υπόλοιπα συστήματα λογισμικού είναι η αυτονομία που παρουσιάζεται στις δομικές μονάδες τους, τις Υπηρεσίες, καθώς και την επιχειρησιακή λογική που ενσωματώνει κάθε μία από αυτές.

Η αυτονομία των Υπηρεσιοκεντρικών συστημάτων δημιουργεί πρόβλημα σε ότι αφορά την παρακολούθηση της λειτουργίας (monitoring) του συστήματος λογισμικού. Όταν το σύστημα λογισμικού είναι υλοποιημένο βάσει μιας συνεκτικής αρχιτεκτονικής, τότε η παρακολούθηση της λειτουργίας του μπορεί να γίνει αρκετά εύκολα με τη χρήση κατάλληλων εργαλείων πάνω σε κάθε δομική του μονάδα. Σε ένα Υπηρεσιοκεντρικό σύστημα όμως αυτό δεν είναι πάντα εφικτό. Ο λόγος που συμβαίνει αυτό είναι επειδή μπορεί κάποιες Υπηρεσίες του συστήματος να ανήκουν σε διαφορετικούς παρόχους. Οπότε για λόγους ανταγωνισμού, κάποιος πάροχος, δεν θέλει να δώσει πλήρη ή και καθόλου δικαιώματα παρακολούθησης για το τι συμβαίνει στην Υπηρεσία του κατά την εκτέλεσή της. Το γεγονός αυτό δημιουργεί δυσκολία στην απόκτηση μια σαφούς εικόνας για το αν το Υπηρεσιοκεντρικό σύστημα είναι απόλυτα συμβατό με τις επιχειρησιακές πολιτικές των επιχειρησιακών διαδικασιών που υλοποιεί. Είναι προφανής λοιπόν η ανάγκη ανάπτυξης τεχνικών που θα μας επιτρέπουν την εξαγωγή συμπερασμάτων για τη συμβατότητα του συστήματος με προσεγγιστικό τρόπο.

1.4 Κίνητρο

Στη σημερινή εποχή οι επιχειρήσεις επενδύουν υψηλά ποσοστά των χρηματικών και ανθρωπίνων πόρων τους για την αγορά και τη λειτουργία συστημάτων λογισμικού, προκειμένου να αυξήσουν την ανταγωνιστικότητά τους. Υπάρχει επομένως μια προφανής εξάρτηση της επιτυχίας των επιχειρήσεων και της ορθής λειτουργίας των συστημάτων λογισμικού που τις υποστηρίζουν. Η ορθή λειτουργία τέτοιων συστημάτων εξαρτάται άμεσα από τη συμμόρφωσή τους με τις επιχειρησιακές πολιτικές που έχουν οριστεί από τους σχεδιαστές των επιχειρησιακών διαδικασιών. Όπως αναφέρθηκε και προηγουμένως, μια ασυμβατότητα με κάποια πολιτική μπορεί να επιφέρει μεγάλες οικονομικές ή και νομικές συνέπειες.

Οι επιχειρήσεις προκειμένου να αποφύγουν τις ασυμβατότητες των επιχειρησιακών τους διαδικασιών με τις διάφορες πολιτικές, προσλαμβάνουν εξειδικευμένο προσωπικό, το οποίο έχει ως αποκλειστικό ρόλο τον έλεγχο της συμμόρφωσης. Η οικονομική επιβάρυνση που επιφέρει η πρόσληψη τέτοιου προσωπικού, είναι αρκετά μεγάλη και μάλιστα χωρίς να εξασφαλίζεται η απόλυτη επιτυχία στην επίτευξη αυτού του ρόλου, καθώς πάντα υπάρχει το ενδεχόμενο του ανθρωπίνου λάθους. Είναι προφανής λοιπόν η ανάγκη επομένως για αυτοματοποίηση της διαδικασίας ελέγχου της συμμόρφωσης των επιχειρησιακών διαδικασιών με τις επιχειρησιακές πολιτικές πετυχαίνοντας έτσι τη μείωση του κόστους και την αύξηση της αξιοπιστίας του αποτελέσματος. Το γεγονός αυτό αποτελεί σημαντικό κίνητρο για την ανάπτυξη μεθόδων που θα επιτρέπουν την αυτοματοποίηση του ελέγχου της συμμόρφωσης των επιχειρησιακών διαδικασιών με τις διάφορες πολιτικές.

1.5 Συνεισφορά της Διπλωματικής Εργασίας

Η παρούσα διπλωματική εργασία παρουσιάζει τον σχεδιασμό και την λειτουργία ενός περιβάλλοντος – πλαισίου, το οποίο εξετάζει την ορθή λειτουργία ενός Υπηρεσιοκεντρικού συστήματος, δίνοντας στον χρήστη τη δυνατότητα να αποφανθεί γι’ αυτή με ένα βαθμό βεβαιότητας εκφρασμένο σε μορφή πιθανότητας. Οι καινοτομίες του συγκεκριμένου περιβάλλοντος – πλαισίου σε σχέση με άλλα που έχουν σχεδιαστεί για τον ίδιο σκοπό, συνοψίζονται στα εξής:

- Τον ορισμό μιας πολυστρωματικής (multi - layered) αρχιτεκτονικής, η οποία παρέχει τη δυνατότητα μοντελοποίησης των επιχειρησιακών πολιτικών και των πολιτικών συμμόρφωσης με αντικειμενοστραφή τρόπο (MOF) [6,7] και της χρήσης αλγορίθμων που

αξιοποιούν τα αρχεία καταγραφής γεγονότων (logs) που μπορούν εξακριβώσουν το βαθμό συμβατότητας του συστήματος με τις πολιτικές αυτές.

- Τη χρήση των Μοντέλων Στόχων για την μοντελοποίηση των επιχειρησιακών πολιτικών. Τα Μοντέλα Στόχων είναι γράφοι, που ακολουθούν δενδρική δομή και κάθε κόμβος τους αντιπροσωπεύει ένα στόχο. Οι μοντελοποιούμενες πολιτικές αποτελούν το πρωταρχικό στόχο, και αναλύονται σε υποστόχους (subgoals) που συνδέονται μεταξύ τους με διαζεύξεις (OR) ή συζεύξεις (AND). Οι κόμβοι του Μοντέλου Στόχων επισημειώνονται με κατηγορήματα Λογικής Πρώτης Τάξης (First Order Logic), με κανόνες σε Object Constraint Language κ.ά. προκειμένου να εφαρμοστούν τεχνικές συμπερασμού για τον έλεγχο της συμμόρφωσης. Στα πλαίσια αυτής της διπλωματικής θα ασχοληθούμε με την περίπτωσης της επισημείωσης με κατηγορήματα Λογικής Πρώτης Τάξης.
- Την αξιοποίησης της θεωρίας των Μαρκοβιανών Λογικών Δικτύων (Markov Logic Networks) η οποία παρέχει τη δυνατότητα πιθανοτικού συμπερασμού (probabilistic reasoning) μέσω στατιστικής συσχετιστικής μάθησης (statistical relational learning). Το περιβάλλον – πλαίσιο εξάγει το Μαρκοβιανό Λογικό Δίκτυο αυτόματα από το επισημειωμένο Δέντρο Στόχων και το εργαλείο λογικού συμπερασμού Alchemy εξάγει σε μορφή πιθανότητας το βαθμό συμβατότητας του κάθε στόχου. Η θεωρία των Μαρκοβιανών Λογικών Δικτύων δίνει τη δυνατότητα διαχείρισης της αβεβαιότητας, που μπορεί να προκύψει από ελλιπή δεδομένα κατά την καταγραφή εκτέλεσης του συστήματος, κάτι που ίσως αποτελεί τη σημαντικότερη καινοτομία του συγκεκριμένου περιβάλλοντος – πλαισίου.

1.6 Οργάνωση του Κειμένου

Η δομή της παρούσας διπλωματικής παρουσιάζεται κάτωθι:

Κεφάλαιο 1: Το κεφάλαιο αυτό αποτελεί την εισαγωγή στις βασικές έννοιες που αναφέρονται στην παρούσα διπλωματική εργασία, περιγράφει το πρόβλημα που πραγματεύεται καθώς και τις λύσεις που προτείνονται για την επίλυσή του.

Κεφάλαιο 2: Στο κεφάλαιο αυτό παρουσιάζεται όλο το θεωρητικό υπόβαθρο που είναι απαραίτητο για την κατανόηση του θέματος της παρούσας εργασίας. Επίσης, γίνεται αναφορά και στο σύνολο των τεχνολογιών που χρησιμοποιήθηκαν κατά την εκπόνησή της.

Κεφάλαιο 3: Στο κεφάλαιο αυτό παρουσιάζεται λεπτομερώς την αρχιτεκτονική του περιβάλλοντος – πλαισίου που αναπτύχθηκε στα πλαίσια αυτής της διπλωματικής, καθώς και τη λειτουργία κάθε συστατικού του.

Κεφάλαιο 4: Στο κεφάλαιο αυτό περιγράφεται το δομικό μοντέλο του περιβάλλοντος – πλαισίου, αναλύοντάς τη λειτουργικότητά του σε επίπεδο κλάσεων.

Κεφάλαιο 5: Στο κεφάλαιο αυτό περιγράφεται ο αλγόριθμος που αναπτύχθηκε για την εξακρίβωση ορθής λειτουργίας σε Υπηρεσιοκεντρικά συστήματα.

Κεφάλαιο 6: Το κεφάλαιο αυτό περιλαμβάνει ένα σύνολο περιπτώσεων χρήσης, παρουσιάζοντας πιθανές εφαρμογές του περιβάλλοντος – πλαισίου που αναπτύχθηκε.

Κεφάλαιο 7: Στο κεφάλαιο αυτό παρουσιάζονται τα αποτελέσματα της παρούσας εργασίας, καθώς και πιθανές επεκτάσεις για μελλοντική μελέτη και έρευνα.

Κεφάλαιο 2: Σχετικές Εργασίες

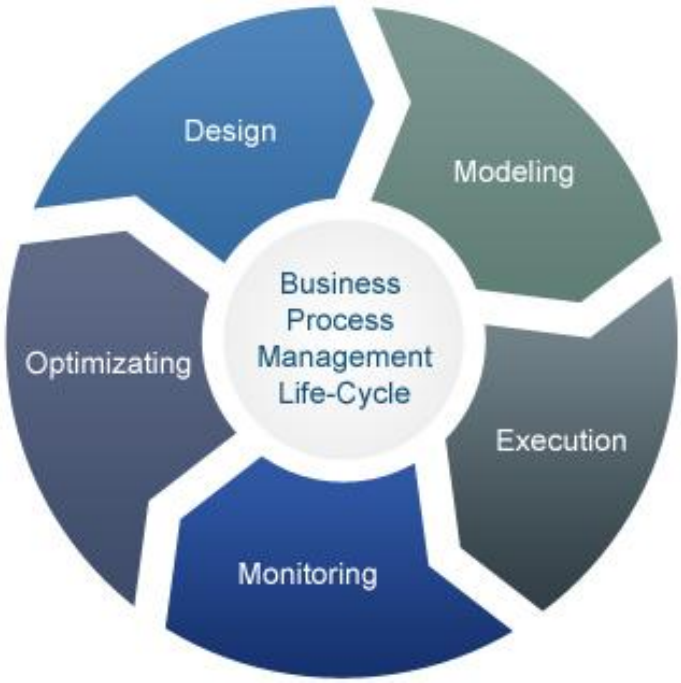
2.1 Εισαγωγή

Στο παρόν κεφάλαιο παραθέτουμε σύνολο των τεχνολογιών που χρησιμοποιήθηκαν για την ανάπτυξη του περιβάλλοντος – πλαισίου που αναπτύχθηκε κατά την εκπόνηση της παρούσας διπλωματικής εργασίας, καθώς και το θεωρητικό υπόβαθρο που απαιτείται για την κατανόηση της λειτουργίας του. Το γνωστικό αντικείμενο των θεμάτων που πραγματεύεται αυτό το κεφάλαιο είναι αρκετά εκτενές και ξεπερνά τα πλαίσια της διπλωματικής εργασίας, γι αυτό το παρακάτω κείμενο περιορίζεται μόνο σε βασικές έννοιες. Για περισσότερες λεπτομέρειες ο αναγνώστης μπορεί να ανατρέξει στις βιβλιογραφικές αναφορές, στο κεφάλαιο 8.

2.2 Μοντελοποίηση και Διαχείριση Επιχειρησιακών Διαδικασιών

Καθημερινά οι επιχειρήσεις προσπαθούν να εφαρμόσουν ένα μεθόδους και τεχνικές προκειμένου να κάνουν τις επιχειρησιακές διαδικασίες τους πιο αποδοτικές και να επιτύχουν τους στόχους τους. Η συστηματική αυτή διαδικασία ονομάζεται διαχείριση επιχειρησιακών διαδικασιών (business process management) και αποτελεί θεμέλιο λίθο για τη λειτουργία κάθε επιχείρησης. Η διαχείριση επιχειρησιακών διαδικασιών αποτελείται από πέντε στάδια. Το πρώτο στάδιο είναι αυτό της Σχεδίασης (Design), όπου καθορίζονται τα βασικά χαρακτηριστικά μια επιχειρησιακής διαδικασίας, όπως το ποιοι θα συμμετέχουν σε αυτή, πόσο χρόνο θα διαρκέσει αλλά και τι λειτουργίες ακριβώς θα επιτελέσει. Το δεύτερο στάδιο στο οποίο και θα επικεντρωθούμε είναι αυτό της Μοντελοποίησης (Modeling). Στο στάδιο αυτό δημιουργείται ένα αρχέτυπο της επιχειρησιακής διαδικασίας, το οποίο αναπαριστά με πλήρη λεπτομέρεια κάθε τμήμα της. Στη συνέχεια, το στάδιο της Εκτέλεσης (Execution) είναι αυτό στο οποίο υλοποιείται το μοντέλο και η επιχειρησιακή διαδικασία διεκπεραιώνεται. Το επόμενο στάδιο είναι η Παρακολούθηση (Monitoring) της επιχειρησιακής διαδικασίας, όπου καταγράφεται κάθε ενέργεια που προκύπτει κατά την εκτέλεσή της δημιουργώντας ένα αρχείο εκτέλεσης. Το πέμπτο και τελευταίο στάδιο είναι της Βελτιστοποίησης (Optimization), όπου γίνεται η αξιολόγηση της εκτέλεσης της επιχειρησιακής διαδικασίας συγκρίνοντας το αρχείο εκτέλεσης με όσα είχαν προδιαγραφεί στα στάδια της Σχεδίασης και της Μοντελοποίησης. Κατά την αξιολόγηση αυτή εντοπίζονται τυχόν σφάλματα στην

επιχειρησιακή διαδικασία για τα οποία προτείνονται διορθώσεις, οι οποίες εφαρμόζονται εκ νέου στο στάδιο της Σχεδίασης, συνεχίζοντας έτσι τον κύκλο ζωής της επιχειρησιακής διαδικασίας.



Εικόνα 2.2.1: Κύκλος ζωής διαχείρισης επιχειρησιακών διαδικασιών [8]

Από τα πέντε στάδια που παρουσιάστηκαν παραπάνω αυτό που θα μας απασχολήσει περισσότερο σε αυτή την ενότητα είναι αυτό της Μοντελοποίησης. Όπως αναφέρθηκε προηγουμένως κατά την μοντελοποίηση, αναπαριστώνται όλες οι λεπτομέρειες μιας επιχειρησιακής διαδικασίας. Η αναπαράσταση είναι απαραίτητη, ώστε να καταστεί σαφές τι πρέπει να υλοποιηθεί και πως πρέπει να εκτελεστεί. Με λίγα λόγια η Μοντελοποίηση είναι ο συνδετικός κρίκος ανάμεσα σε αυτό που έχουν προδιαγράψει οι επιχειρησιακοί αναλυτές (business analysts) και οι διευθυντές στο στάδιο της Σχεδίασης με αυτό που πρέπει να υλοποιήσουν εργαζόμενοι που ασχολούνται με τα τεχνικά θέματα μιας επιχειρησιακής διαδικασίας όπως αρχιτέκτονες συστημάτων (system architects) και μηχανικούς λογισμικού (software engineers). Η μετάβαση από το επιχειρησιακό στο τεχνικό μέρος είναι κρίσιμη καθώς μια λανθασμένη αντίληψη της Σχεδίασης μπορεί να οδηγήσει σε ένα εντελώς διαφορετικό αποτέλεσμα από το αναμενόμενο και ενδεχομένως με καταστροφικές συνέπειες για την επιχείρηση. Επομένως, γίνεται εμφανής η ανάγκη τυποποιημένων μεθόδων μοντελοποίησης επιχειρησιακών διαδικασιών που να μπορούν να εξομαλύνουν τη μετάβαση αυτή.

Από τις αρχές του 20ού αιώνα, άρχισαν να αναπτύσσονται διάφορες μέθοδοι μοντελοποίησης επιχειρησιακών διαδικασιών όπως τα διαγράμματα ροής (flow charts), τα διαγράμματα ροής ελέγχου (control flow charts), τα διαγράμματα Gantt, PERT, ροής δεδομένων (data flow diagrams) κ.ά. Το διάγραμμα Gantt ήταν το πρώτο που εμφανίστηκε το 1899 περίπου και ακολούθησαν τα διαγράμματα ροής το 1920, το διάγραμμα PERT το 1987 και τα διαγράμματα ροής δεδομένων το 1970. Σήμερα, για τη μοντελοποίηση επιχειρησιακών διαδικασιών χρησιμοποιούνται πιο σύγχρονες τυποποιημένες μέθοδοι όπως η Unified Modeling Language (UML) [9] και η Business Process Modeling Notation (BPMN) [10,11], την οποία και θα παρουσιάσουμε αναλυτικότερα παρακάτω.

Καθώς όμως η ανάγκη για αυτοματοποίηση της διαχείρισης επιχειρησιακών διαδικασιών μεγάλωνε, έγινε φανερό ότι η γραφική αναπαράσταση δεν ήταν αρκετή. Την ανάγκη αυτή έρχονται να ικανοποιήσουν εκτελέσιμες γλώσσες μοντελοποίησης επιχειρησιακών διαδικασιών, όπως η Business Process Execution Language (BPEL) [4,5]. Οι γλώσσες αυτές αναπτύχθηκαν με βάση τις Διαδικτυακές Υπηρεσίες (Web Services) και συνέβαλαν στην εξέλιξη των Υπηρεσιοκεντρικών συστημάτων. Από μία άποψη έδωσαν στην Υπηρεσιοκεντρική Αρχιτεκτονική ένα χαρακτήρα διαχείρισης επιχειρησιακών διαδικασιών. Το πλεονέκτημα όμως που παρέχουν γλώσσες όμως η BPMN και η BPEL είναι πως παρέχουν έναν κοινό τρόπο επικοινωνίας ανάμεσα στους συμμετέχοντες στη διαχείριση επιχειρησιακών διαδικασιών.

2.2.1 Business Process Modeling Notation - BPMN

Η BPMN είναι ένα πρότυπο που χρησιμοποιείται στο στάδιο της Μοντελοποίησης κατά τη διαχείριση επιχειρησιακών διαδικασιών. Το πρότυπο αυτό παρέχει τη δυνατότητα του ορισμού επιχειρησιακών διαδικασιών μέσω γραφικής αναπαράστασης. Το βασικό του στοιχείο είναι το Business Process Diagram (BPD), το οποίο συνδυάζει στοιχεία από τα διαγράμματα ροής αλλά και τα Activity Diagrams της UML. Το πλεονέκτημα του BPD είναι ότι μπορεί να χρησιμοποιηθεί και να γίνει κατανοητό από χρήστες που το γνωστικό τους αντικείμενο είναι είτε οι επιχειρήσεις, είτε η τεχνολογία. Παρά την απλότητά του αυτή είναι ικανό να αναπαραστήσει αρκετά πολύπλοκα στοιχεία των επιχειρησιακών διαδικασιών. Ένα επίσης σημαντικό στοιχείο της BPMN είναι η δυνατότητα απεικόνισής της σε άλλες γλώσσες, οι οποίες είναι εκτελέσιμες (π.χ. BPEL).

Τα βασικά στοιχεία της BPMN που ορίζονται από την προδιαγραφή της [10] κατατάσσονται στις ακόλουθες κατηγορίες:

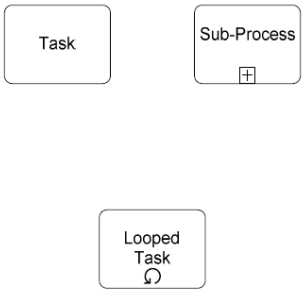
- **Αντικείμενα Ροής (Flow Objects)**, τα οποία αποτελούν τα βασικά δομικά στοιχεία μιας επιχειρησιακής διαδικασίας. Υπάρχουν τρία είδη αντικειμένων ροής:
    - **Γεγονός (Event)**

      Ένα γεγονός υποδεικνύει κάτι που συμβαίνει σε μια συγκεκριμένη χρονική στιγμή μιας διαδικασίας και συνήθως έχει ένα συγκεκριμένο αποτέλεσμα (π.χ. τερματισμός διαδικασίας) ή εκκινεί (trigger) κάποια ενέργεια (π.χ. αποστολή ενός μηνύματος).

Εικόνα 2.2.2: Παραδείγματα Γεγονότων

    - **Δραστηριότητα (Activity)**

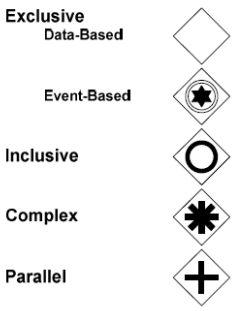
      Μια δραστηριότητα απεικονίζει συγκεκριμένες ενέργειες μιας επιχειρησιακής διαδικασίας (π.χ. καταβολή χρημάτων μισθοδοσίας).



Εικόνα 2.2.3: Παραδείγματα Δραστηριοτήτων

    - **Πύλη (Gateway)**

      Η πύλη αναπαριστά διαφόρους τύπος διακλαδώσεων που μπορεί να συμβούν στη ροή μιας επιχειρησιακής διαδικασίας.



Εικόνα 2.2.4: Παραδείγματα Πυλών

- **Αντικείμενα Σύνδεσης (Connecting Objects)**

  Τα αντικείμενα σύνδεσης χρησιμοποιούνται για να συσχετίσουνε τα αντικείμενα ροής είτε μεταξύ τους είτε με άλλα αντικείμενα. Υπάρχουν τρία είδη αντικειμένων σύνδεσης:

  - **Ροή (Sequence Flow)**

    Η ροή αναπαριστά τη ροή με την οποία συμβαίνουν οι δραστηριότητες σε μια επιχειρησιακή διαδικασία.

Εικόνα 2.2.5: Σύμβολο Ροής

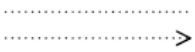
  - **Ροή Μηνυμάτων (Message Flow)**

    Η ροή μηνυμάτων αναπαριστά την ανταλλαγή μηνυμάτων μεταξύ διαφόρων οντοτήτων υποδεικνύοντας παράλληλα ποιος είναι ο αποστολέας και ποιος ο παραλήπτης.

Εικόνα 2.2.6: Σύμβολο Ροής Μηνυμάτων

  - **Σύνδεση (Association)**

    Μια σύνδεση αναπαριστά συσχετίσεις μεταξύ οντοτήτων που συμμετέχουν σε μια επιχειρησιακή διαδικασία.



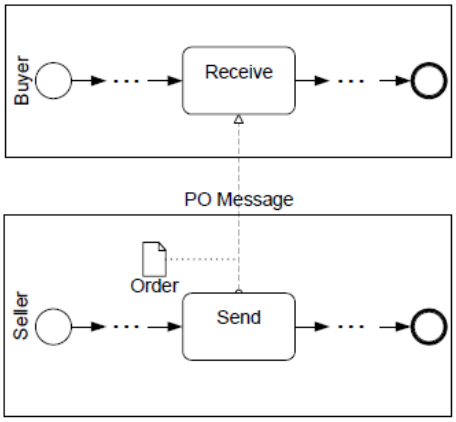
Εικόνα 2.2.7: Σύμβολα Σύνδεσης

- **Δεξαμενές (Swimlanes)**, οι οποίες ομαδοποιούν τα πρωτεύοντα στοιχεία σε δύο κατηγορίες:

  - **Ενότητες (Pools)**

    Η ενότητα αναπαριστά τους συμμετέχοντες σε μια επιχειρησιακή διαδικασία. Ένας συμμετέχοντας μπορεί να αναπαρασταθεί είτε ως ρόλος (π.χ.) είτε ως συγκεκριμένο πρόσωπο ή οργανισμός (π.χ. Steve Jobs, Apple).

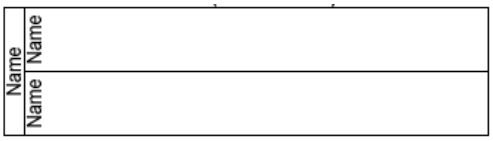
Εικόνα 2.2.8: Σύμβολο Ενότητας



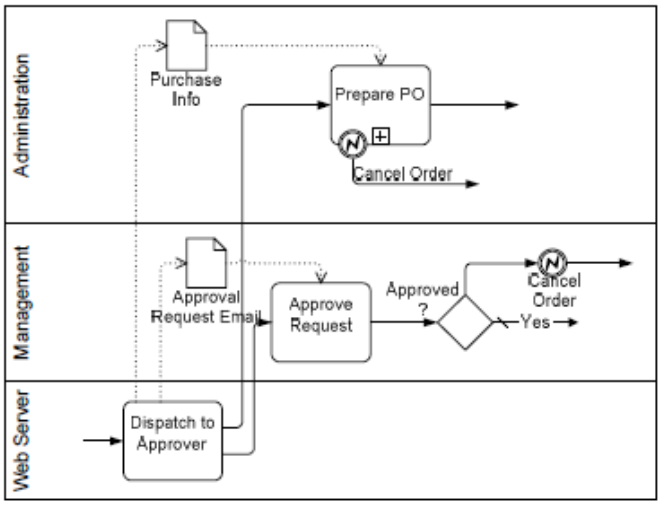
Εικόνα 2.2.9: Παράδειγμα χρήσης Ενότητας

- **Υπο-Ενότητες (Lanes)**

  Οι υπο-ενότητες αντιπροσωπεύουν τμήματα μιας ενότητας υπό μορφή ρόλων.



Εικόνα 2.2.10: Σύμβολο Υπο-Ενοτήτων



Εικόνα 2.2.11: Παράδειγμα χρήσης Υπο-Ενοτήτων

- **Συμπληρωματικά Αντικείμενα** (**Artifacts**), τα οποία επιτρέπουν στο χρήστη να προσθέσει επιπλέον πληροφορία στα διάφορα σημεία της διαδικασίας που σχεδιάζει. Ένα συμπληρωματικό αντικείμενο μπορεί να είναι κάτι από τα παρακάτω:
    - **Αντικείμενο Δεδομένων** (**Data Object**)

      Τα αντικείμενα δεδομένων αναπαριστούν τα δεδομένα που απαιτούνται ή παράγονται από μια δραστηριότητα.

      Εικόνα 2.2.12: Σύμβολο Αντικειμένου Δεδομένων

    - **Επισημείωση** (**Annotation**)

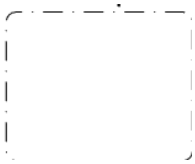
      Η επισημείωση χρησιμοποιείται για να δώσει περισσότερες λεπτομέρειες για τη λειτουργία κάποιου στοιχείου ενός BPD.

      Εικόνα 2.2.13: Σύμβολο Επισημείωσης

    - **Ομαδοποίηση** (**Group**)

      Η ομαδοποίηση χρησιμοποιείται για να οργανώνει σε ομάδες στοιχεία που έχουν κοινά χαρακτηριστικά, χωρίς όμως να επηρεάζει την ροή της διαδικασίας.

      Εικόνα 2.2.14: Σύμβολο Ομαδοποίησης

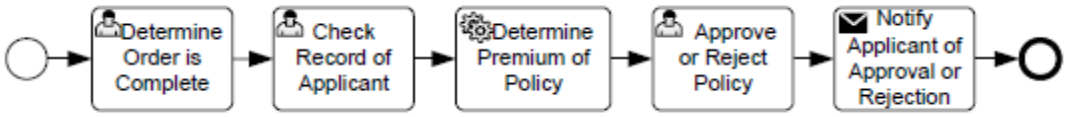
Η BPMN έχει σχεδιαστεί ώστε να μπορεί να αναπαραστήσει ένα ευρύ φάσμα πληροφορίας που σχετίζεται με τις επιχειρησιακές διαδικασίες. Η πληροφορία αυτή μπορεί να αφορά τις λεπτομέρειες εκτέλεσης της επιχειρησιακής διαδικασίας ή τον τρόπο επικοινωνίας μεταξύ διαφόρων επιχειρησιακών διαδικασιών. Προκειμένου η πληροφορία αυτή να γίνει αντιληπτή από τον αναγνώστη ενός διαγράμματος BPMN, το μοντέλο της γλώσσα περιλαμβάνει κάποια υπομοντέλα τα οποία κατηγοριοποιούν τις επιχειρησιακές διαδικασίες που σχεδιάζονται. Οι κατηγορίες αυτές είναι οι εξής:

1. **Διαδικασίες Ενορχήστρωσης** (**Orchestration**)
    - *Ιδιωτικές μη εκτελέσιμες* (*Private non - executable*)

- *Ιδιωτικές εκτελέσιμες* (*Private executable*)
- *Δημόσιες* (*Public*)

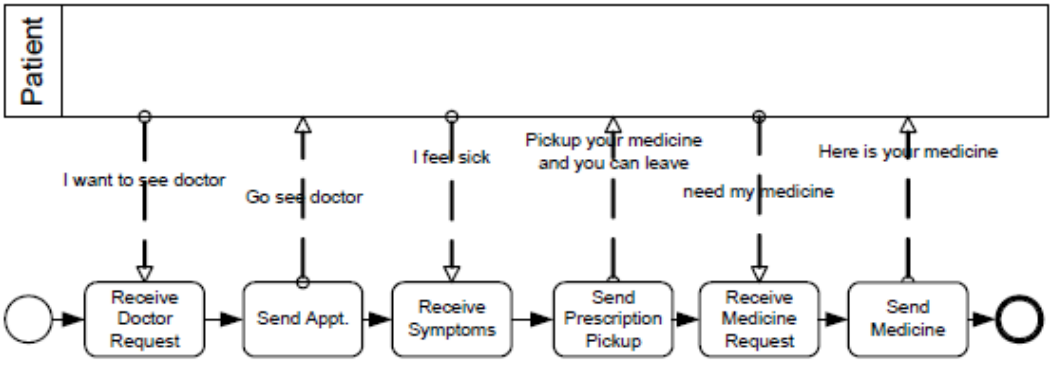
2. **Χορογραφίες** (**Choreographies**)
3. **Συνεργασίες** (**Collaborations**)

Οι Ιδιωτικές επιχειρησιακές διαδικασίες χρησιμοποιούνται για να περιγράψουν τις λειτουργίες που εκτελούνται στο εσωτερικό ενός οργανισμού. Μια επιχειρησιακή διαδικασία που περιγράφει με πλήρη λεπτομέρεια τα βήματα εκτέλεσής είναι μια εκτελέσιμη διαδικασία. Αντίθετα, αν μια διαδικασία περιγράφεται σε επίπεδο συμπεριφοράς, για λόγους μοντελοποίησης και τεκμηρίωσης, είναι μη εκτελέσιμη.



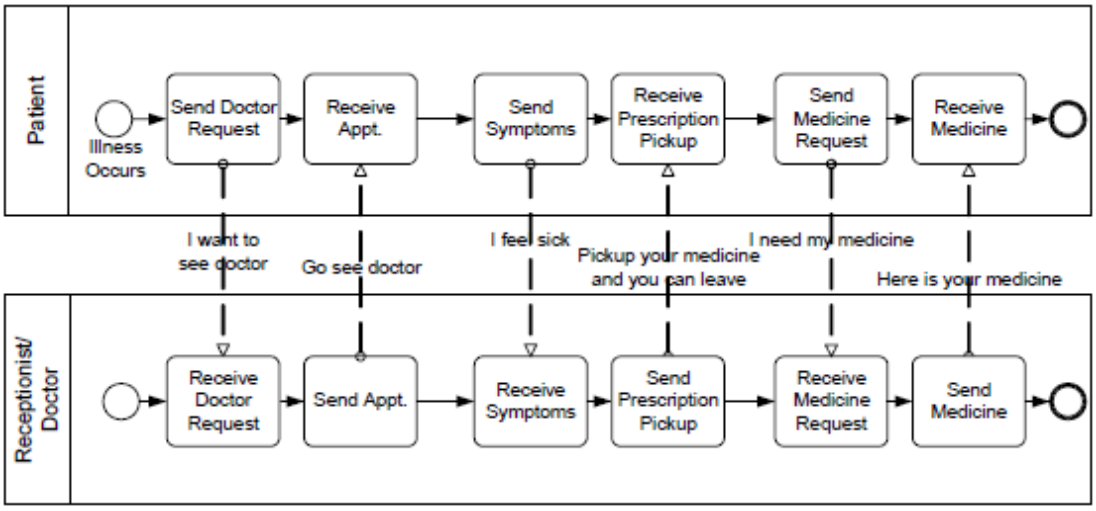
Εικόνα 2.2.15: Παράδειγμα Ιδιωτικής Επιχειρησιακής Διαδικασίας

Οι Δημόσιες επιχειρησιακές διαδικασίες αναπαριστούν μόνο τις εξωτερικές αλληλεπιδράσεις μιας διαδικασίες με εξωτερικούς εταίρους. Οι εσωτερικές Δραστηριότητες και μηνύματα αποκρύπτονται πλήρως. Στις Δημόσιες επιχειρησιακές διαδικασίες αναπαριστώνται τα Μηνύματα που ανταλλάσσονται ανάμεσα σε στην επιχειρησιακή διαδικασία ενός οργανισμού και πρόσωπα που δεν ανήκουν στον οργανισμό αυτό. Επίσης, απεικονίζονται και όλες οι δραστηριότητες τη επιχειρησιακής διαδικασίας του οργανισμού, οι οποίες συμμετέχουν στην αποστολή και την λήψη των μηνυμάτων.



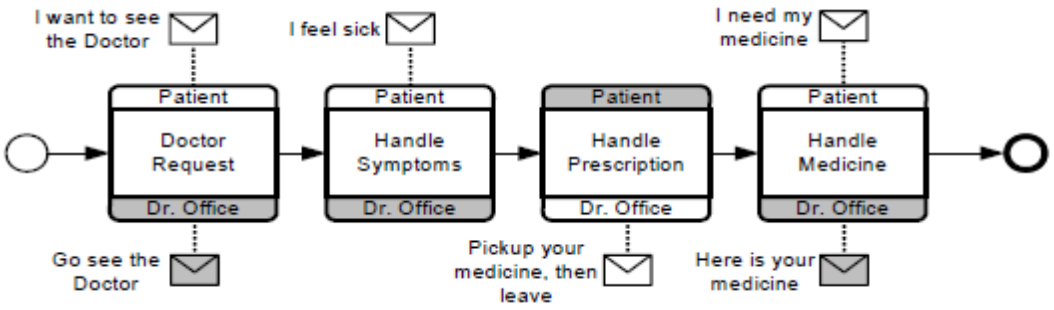
Εικόνα 2.2.16: Παράδειγμα Δημόσιας Επιχειρησιακής Διαδικασίας

Οι Συνεργασίες αναπαριστούν αλληλεπιδράσεις ανάμεσα σε δύο ή περισσότερες επιχειρησιακές οντότητες. Οι επιχειρησιακές οντότητες αυτές απεικονίζονται από δύο ή περισσότερες Ενότητες. Σε μια συνεργασία αναπαριστώνται τα Μηνύματα που ανταλλάσσονται μεταξύ των εταίρων, καθώς και οι Δραστηριότητες που συμμετέχουν στην αποστολή και την λήψη τους. Οι Συνεργατικές επιχειρησιακές διαδικασίες μπορούν να θεωρηθούν ως αλληλεπίδραση μεταξύ δύο ή περισσότερων Δημόσιων διαδικασιών.



Εικόνα 2.2.17: Παράδειγμα Συνεργασίας

Η Χορογραφία ορίζει μια αλληλουχία διεργασιών μεταξύ διαφόρων εταίρων που συμμετέχουν σε μια επιχειρησιακή διαδικασία. Αξίζει να σημειώσουμε, ότι ενώ σε μια επιχειρησιακή διαδικασία των παραπάνω κατηγοριών οι Δραστηριότητες υπάρχουν εντός των Ενοτήτων, στη Χορογραφία υπάρχουν ανάμεσα στους συμμετέχοντες εταίρους. Μπορούμε να θεωρήσουμε τη Χορογραφία σαν μια Ιδιωτική επιχειρησιακή διαδικασία, με τη διαφορά πως οι Δραστηριότητες αναπαριστούν ένα σύνολο ανταλλαγής μηνυμάτων. Επίσης στη Χορογραφία δεν υπάρχει κάποιος εταίρος που συντονίζει την επιχειρησιακή διαδικασία. Στην επόμενη ενότητα θα γίνει πιο σαφές, τι σημαίνει η παρουσία ή η απουσία κεντρικού ελέγχου, αναλύοντας τη διαφορά ανάμεσα στην Ενορχήστρωση και την Χορογραφία.



Εικόνα 2.2.18: Παράδειγμα Χορογραφίας

2.2.2 Business Process Execution Language – BPEL

Η BPEL είναι μια γλώσσα που χρησιμοποιείται για τον ορισμό και την εκτέλεση επιχειρησιακών διαδικασιών χρησιμοποιώντας Διαδικτυακές υπηρεσίες (Web services). Η γλώσσα αυτή δίνει τη δυνατότητα υλοποίησης της Υπηρεσιοκεντρικής αρχιτεκτονικής μέσω της σύνθεσης, της ενορχήστρωσης και του συντονισμού Διαδικτυακών υπηρεσιών. Με την BPEL μπορούν να υλοποιηθούν επιχειρησιακές διαδικασίες συνθέτοντας υπάρχουσες Διαδικτυακές υπηρεσίες, παρέχοντας έτσι ένα ισχυρό εργαλείο στους σχεδιαστές επιχειρησιακών διαδικασιών.

Η γλώσσα BPEL είναι βασισμένη στην XML [12] και αξιοποιεί ένα σύνολο τεχνολογιών που υποστηρίζουν τις Διαδικτυακές υπηρεσίες, όπως SOAP [13], WSDL [14], UDDI [15], WS - Reliable Messaging, WS - Addressing, WS - Coordination, και WS - Transaction.

Πριν την ανάπτυξη της BPEL υπήρξαν και άλλες προσπάθειες για τη δημιουργία γλωσσών που να μπορούν να περιγράφουν μια ροή εργασιών (workflow), όπως η Web Services Flow Language (WSFL) [16]και η XLANG [17]. Η WSFL σχεδιάστηκε από την IBM και βασίστηκε στη θεωρία των κατευθυνόμενων γράφων. Η XLANG από την άλλη ήταν μια γλώσσα που χρησιμοποιούσε μπλοκ δομές και σχεδιάστηκε από την Microsoft. Η BPEL είναι απόρροια αυτών των δύο γλωσσών, συνδυάζοντας τις προσεγγίσεις τους και παρέχοντας ένα ευρύ φάσμα στοιχείων, που μπορούν να περιγράψουν επιχειρησιακές διαδικασίες με μεγάλη λεπτομέρεια.

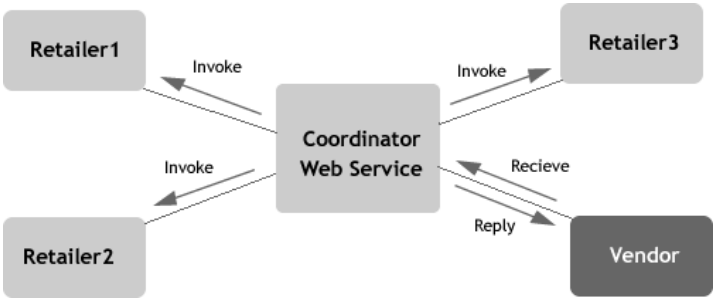
Κάθε επιχείρηση έχει τον δικό της τρόπο σχεδίασης και περιγραφής επιχειρησιακών διαδικασιών, με αποτέλεσμα να προκύπτουν ασάφειες και ασυμβατότητες όταν αυτές οι διαδικασίες πρέπει να επεκταθούν και να επικοινωνήσουν με διαδικασίες άλλων

επιχειρήσεων. Η BPEL επιλύει αυτό το πρόβλημα προσφέροντας ένα κοινό πρότυπο ανάπτυξης επιχειρησιακών διαδικασιών, αλλά και έναν τρόπο ενοποίησης συστημάτων τα οποία ήταν απομονωμένα. Η περιγραφή των επιχειρησιακών διαδικασιών σε BPEL δεν επηρεάζει τα ήδη υπάρχοντα συστήματα, ωστόσο ενθαρρύνει τη βελτίωσή τους. Η λειτουργικότητα της BPEL είναι άμεσα συνδεδεμένη με τις Διαδικτυακές υπηρεσίες και όσο η τεχνολογία τους εξελίσσεται, τόσο αυξάνεται και η σπουδαιότητα της BPEL.

Όπως αναφέραμε προηγουμένως η BPEL παρέχει τη δυνατότητα συνδυασμού διαφόρων Διαδικτυακών υπηρεσιών για την ανάπτυξη εφαρμογών που ακολουθούν την Υπηρεσιοκεντρική αρχιτεκτονική. Ο συνδυασμός αυτός μπορεί να γίνει με δύο τρόπους:

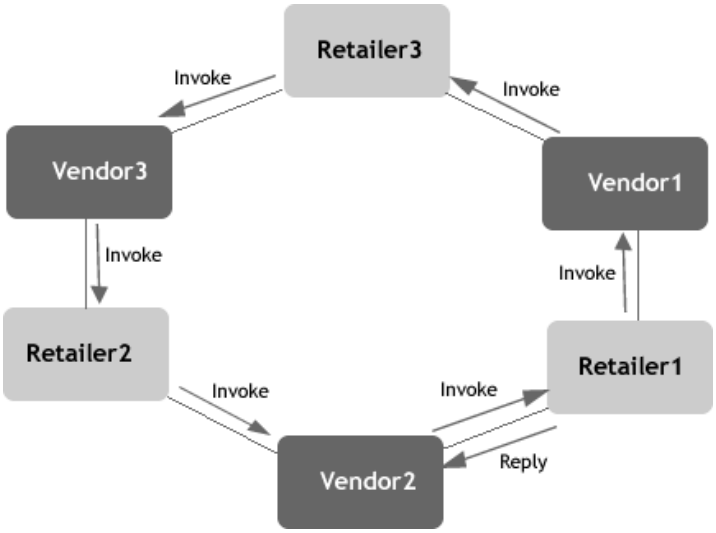
- Ενορχήστρωση
- Χορογραφία (choreography)

Στο μοντέλο της ενορχήστρωσης υπάρχει μια κεντρική Διαδικτυακή υπηρεσία η οποία παίζει το ρόλο του συντονιστή (coordinator) και ελέγχει τις υπόλοιπες υπηρεσίες που λαμβάνουν μέρος στην επιχειρησιακή διαδικασία. Οι επιμέρους διαδικασίες δεν γνωρίζουν ότι συμμετέχουν σε μια επιχειρησιακή διαδικασία, ούτε ότι αποτελούν μέρος μιας σύνθεσης υπηρεσιών. Μόνο η διαδικασία συντονιστής γνωρίζει στο σύνολό της την επιχειρησιακή διαδικασία και την εκτελεί βάσει ενός προσχεδιασμένου συνόλου κλήσεων (invocations) προς τις επιμέρους υπηρεσίες.



Εικόνα 2.2.19: Μοντέλο Ενορχήστρωσης

Από την άλλη πλευρά, στο μοντέλο της χορογραφίας δεν υπάρχει κάποια υπηρεσία συντονιστής. Όλες οι διαδικασίες γνωρίζουν την επιχειρησιακή διαδικασία στην οποία συμμετέχουν, ποιες λειτουργίες πρέπει να εκτελέσουν και με ποιες υπηρεσίες πρέπει να αλληλεπιδράσουν.



Εικόνα 2.2.20: Μοντέλο Χορογραφίας

Η BPEL παρέχει δύο τρόπους περιγραφής επιχειρησιακών διαδικασιών υποστηρίζοντας έτσι και το μοντέλο της ενορχήστρωσης και το μοντέλο της χορογραφίας. Οι δύο αυτοί τύποι είναι εξής:

- Εκτελέσιμες διαδικασίες (Executable processes), οι οποίες περιγράφουν με λεπτομέρεια τις λειτουργίες μια επιχειρησιακής διαδικασίας, εκτελούνται πάνω σε μηχανές ενορχήστρωσης (orchestration engines) και ακολουθούν το μοντέλο της ενορχήστρωσης.
- Αφηρημένα επιχειρησιακά πρωτόκολλα, τα οποία περιγράφουν την ανταλλαγή μηνυμάτων μεταξύ συμμετεχόντων σε μια επιχειρησιακή διαδικασία, χωρίς όμως να δίνουν πληροφορίες για τον τρόπο εκτέλεσής τους, ακολουθώντας το μοντέλο της χορογραφίας.

Μια διαδικασία BPEL ορίζει την ακριβή ακολουθία με την οποία θα πρέπει να κληθούν όλες οι συμμετέχουσες Διαδικτυακές υπηρεσίες που συμμετέχουν σε μια Υπηρεσιοκεντρική εφαρμογή. Επίσης, με την BPEL μπορούν να περιγραφούν συνθήκες εκτέλεσης διαφορετικών μονοπατιών σε μια επιχειρησιακή διαδικασία. Για παράδειγμα, η κλήση ή μη μιας Διαδικτυακής υπηρεσίας μπορεί να εξαρτάται από την τιμή μιας προηγούμενες κλήσης. Επιπλέον, ο χρήστης μπορεί να ορίσει βρόχους επαναλήψεων (loops), να δηλώσει μεταβλητές, να αντιγράψει και να αναθέσει τιμές, καθώς και να ορίσει χειριστές σφαλμάτων (fault handlers). Με τη χρήση όλων αυτών των δομών μπορούν να περιγραφούν σύνθετες επιχειρησιακές διαδικασίες με έναν σαφή και αλγοριθμικό τρόπο.

Σε ένα τυπικό σενάριο, μια διαδικασία BPEL λαμβάνει ένα αίτημα (request), το οποίο για να το ικανοποιήσει καλεί μια Διαδικτυακή υπηρεσία και τότε απαντά στον καλούντα της διαδικασίας με το αποτέλεσμα που προέκυψε. Επειδή η BPEL είναι σχεδιασμένη για να

επικοινωνεί με Διαδικτυακές υπηρεσίες, η λειτουργία της είναι στενά συνδεδεμένη με την περιγραφή των Διαδικτυακών υπηρεσιών σε WSDL. Η σχέση αυτή χαρακτηρίζει τα στοιχεία της γλώσσας που θα δούμε παρακάτω.

Κάθε επιχειρησιακή διαδικασία σχεδιασμένη με τη γλώσσα BPEL αποτελείται από μία σειρά βημάτων, όπου το καθένα από αυτά ονομάζεται δραστηριότητα (activity). Μια δραστηριότητα μπορεί να είναι :

- Κλήση προς μια Διαδικτυακή υπηρεσία, με τη χρήση του <invoke>
- Αναμονή λήψης μηνύματος για τον πελάτη που κάλεσε μία υπηρεσία, με τη χρήση του <receive>
- Απόκριση όταν πρόκειται για σύγχρονες λειτουργίες, με τη χρήση του <reply>
- Διαχείριση μεταβλητών, με τη χρήση του <assign>
- Εντοπισμός σφαλμάτων και εξαιρέσεων, με τη χρήση του <throw>
- Αναμονή για κάποιο χρονικό διάστημα, με τη χρήση του <wait>
- Τερματισμός ολόκληρης της διαδικασίας, με τη χρήση του <terminate>

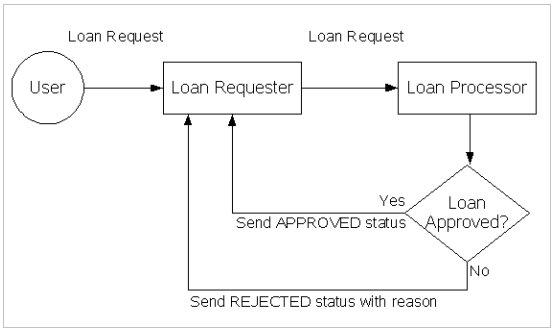
Χρησιμοποιώντας τις παραπάνω κύριες δραστηριότητες μπορούμε να περιγράψουμε με σαφή και αλγοριθμικό τρόπο όλα τα βήματα μιας επιχειρησιακής διαδικασίας. Συνδυάζοντας τις κύριες δραστηριότητες η BPEL υποστηρίζει και άλλη μια κατηγορία της δραστηριότητες δομών (structure activities), οι οποίες μπορούν να περιγράψουν πιο σύνθετες λειτουργίες. Μερικές από τις πιο βασικές δραστηριότητες δομών είναι οι εξής:

- Sequence (<sequence>), για την αναπαράσταση εκτέλεσης ενός συνόλου δραστηριοτήτων με συγκεκριμένη σειρά.
- Flow (<flow>), για την αναπαράσταση ενός συνόλου δραστηριοτήτων που θα εκτελεστούν παράλληλα.
- Case-switch δομή (<switch>), για την αναπαράσταση διακλαδώσεων
- While (<while>), για τον ορισμό βρόχων

Με την αξιοποίηση όλων των παραπάνω χαρακτηριστικών της BPEL ο χρήστης της γλώσσας μπορεί να δημιουργήσει νέες Διαδικτυακές υπηρεσίες, ενορχηστρώνοντας υπηρεσίες που λειτουργούσαν ανεξάρτητα μεταξύ τους. Πρέπει λοιπόν ο κώδικας που γράφει ο χρήστης εγκατασταθεί σε έναν εξυπηρετητή (server), οποίος θα έχει τη δυνατότητα να εκτελεί διαδικασίες BPEL, θα έχει δηλαδή ένα BPEL engine. Τα περιβάλλοντα που χρησιμοποιήθηκαν για την εκπόνηση αυτής της διπλωματικής είναι το openESB που αποτελεί μια επέκταση του Netbeans, ο Intalio Desinger και οι εξυπηρετητής Glassfish και Intalio Server. Στο σημείο αυτό πρέπει να αναφέρουμε ότι η BPEL ως γλώσσα δε διαθέτει κάποια γραφική σημειολογία, ωστόσο πολλά εργαλεία ανάπτυξης διαθέτουν γραφικές απεικονίσεις

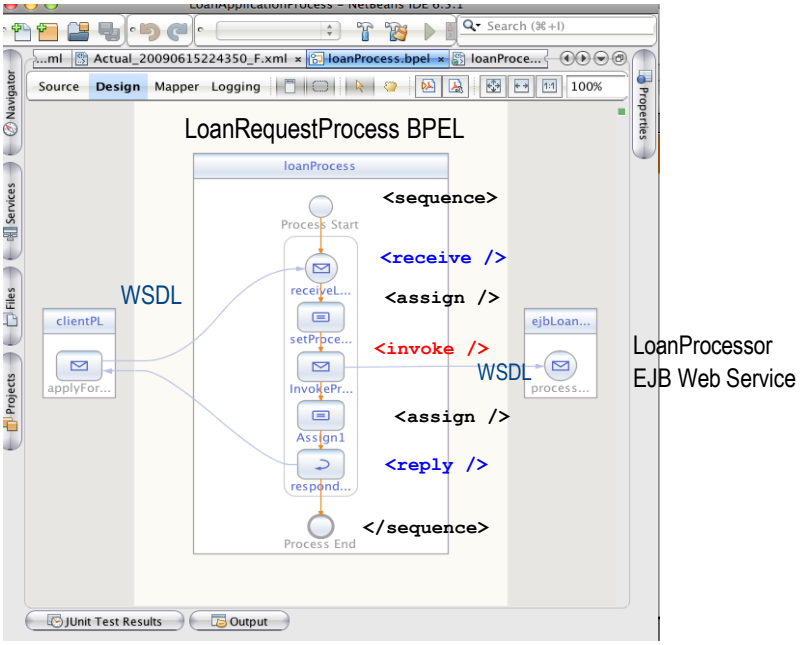
των λειτουργιών της γλώσσας, προς διευκόλυνση του χρήστη. Παρακάτω θα παρουσιάσουμε μια περίπτωση χρήσης για να γίνει πιο κατανοητή η λειτουργία της γλώσσας BPEL.

Έστω λοιπόν ότι έχουμε το σενάριο στο οποίο ένας πελάτης θέλει να λάβει ένα δάνειο από μία τράπεζα. Οι συμβαλλόμενοι σε μια τέτοια περίπτωση είναι ο πελάτης που έχει τον ρόλο του δανειολήπτη και η τράπεζα που έχει τον ρόλο του δανειστή. Η διαδικασία ξεκινά με μια αίτηση του πελάτη προς την τράπεζα με τα στοιχεία του και το ποσό που επιθυμεί ως δάνειο. Στη συνέχεια, η υπεύθυνη υπηρεσία για την εξέταση δανείων της τράπεζας θα εξετάσει τα στοιχεία του πελάτη και θα αποφασίσει αν ο πελάτης μπορεί να λάβει το δάνειο. Όταν τελειώσει τον απαραίτητο έλεγχο η υπηρεσία εξέτασης δανείων θα απαντήσει στο πελάτη είτε θετικά ανακοινώνοντάς του ότι μπορεί να λάβει το δάνειο, είτε αρνητικά αναφέροντας τους λόγους για τους οποίους η αίτησή τους απορρίφθηκε.



Εικόνα 2.2.21: Flowchart για την διαδικασία λήψης δανείου

Για τον σκοπό αυτό έχουν δημιουργηθεί δύο Διαδικτυακές υπηρεσίες που υλοποιούν τις λειτουργίες των δύο συμβαλλόμενων σε αυτή την επιχειρησιακή διαδικασία. Η πρώτη διαδικτυακή υπηρεσία είναι η client η οποία αντιπροσωπεύει τον πελάτη και έχει τον ρόλο της υπηρεσίας συντονιστή. Η υπηρεσία αυτή λαμβάνει τα στοιχεία του πελάτη και το επιθυμητό ποσό δανείου και τα προωθεί μέσω της διεργασίας BPEL στην υπηρεσία LoanProcessor. Η υπηρεσία αυτή, η οποία είναι υλοποιημένη με τεχνολογία Enterprise JavaBeans, διαβάζει το μήνυμα που έστειλε η υπηρεσία client, επεξεργάζεται τα στοιχεία του και στη συνέχεια αποκρίνεται για το αν ο πελάτης δικαιούται το δάνειο ή όχι. Την απάντηση αυτή τη προωθεί η υπηρεσία LoanProcessor μέσω της διεργασίας BPEL στην υπηρεσία client.



Εικόνα 2.2.22: Γραφική απεικόνιση της διαδικασία δανειοληψία από τον BPEL Designer του Netbeans

```
<sequence>
 <receive partnerLink="clientPL" portType="loanProcessPortType"
          operation="applyForLoanOperation" variable="IN" />
  <assign name="setProcessorInput">
    <copy><from variable="clientIn"/><to variable="ejbIn"/> </copy>
  <assign/>
  <invoke name="InvokeProcessLoan" partnerLink="ejbLoanProcessorPL"
          portType="loanProcessPortType"
          operation="processLoanApplicationEJB"
          inputVariable="inventoryRequest"
          outputVariable="inventoryResponse" />
  <assign name="setProcessorOutput">
    <copy><from variable="ejbOut"/><to variable="clientOut"/> </copy>
  <assign/>
 <reply partnerLink="clientPL" portType="loanProcessPortType"
        operation="applyForLoanWSDLOperation"
        variable="ApplyForLoanOperationOut"/>
</sequence>
```

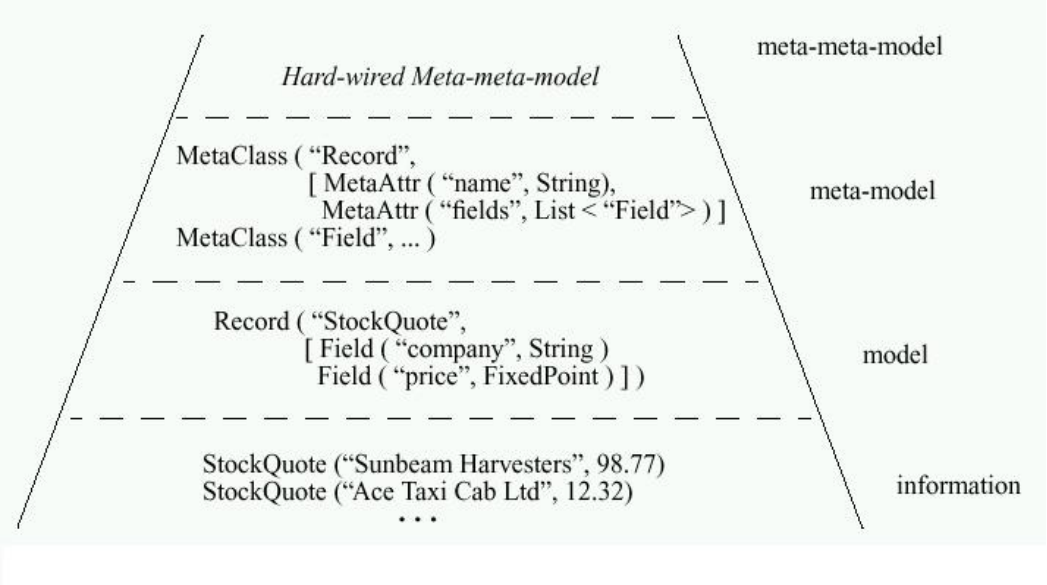
Εικόνα 2.2.23: Πηγαίος κώδικας της διεργασίας BPEL

2.3 MOF/XMI

Το **Meta Object Facility** (**MOF**) [6] αποτελεί ένα πρότυπο του OMG και ένα από τα θεμέλια της μοντελοδηγούμενης μηχανικής (model-driven engineering). Το πρότυπο αυτό ορίζει μια αρχιτεκτονική τεσσάρων επιπέδων με σκοπό τη διαχείριση μετα-δεδομένων, η οποία έχει προδιαγραφεί ώστε να εξασφαλίζει απεριόριστη επεκτασιμότητα. Τα επίπεδα αυτά είναι τα εξής:

1. Το επίπεδο M0 ή επίπεδο πληροφορίας (information layer). Στο επίπεδο αυτό αναπαρίσταται κάθε δεδομένο ή αντικείμενο που επιθυμούμε να μοντελοποιήσουμε.

2. Το επίπεδο M1ή επίπεδο μοντέλου (model layer). Στο επίπεδο αυτό αναπαριστώνται τα μετα-δεδομένα που προέρχονται από το επίπεδο πληροφορίας, παρέχοντας έτσι ένα μοντέλο που περιγραφής τους.
3. Το επίπεδο M2 ή επίπεδο μετα-μοντέλου (meta-model layer). Στο επίπεδο αυτό περιγράφονται τα μοντέλα του επιπέδου M1, προσδιορίζοντας τη δομή και τη σημασιολογία των μετα-δεδομένων τους.
4. Το επίπεδο M3 ή επίπεδο μετα-μετα-μοντέλου (meta-meta-model layer). Το επίπεδο αυτό είναι ουσιαστικά μια αφηρημένη γλώσσα περιγραφής μετα-μοντέλων και αποτελεί την κορυφή στην ιεραρχία των επιπέδων της αρχιτεκτονικής του MOF.



Εικόνα 2.3.1: Αρχιτεκτονική Μετα-δεδομένων Τεσσάρων Επιπέδων

Μια αξιοσημείωτη χρήση του επιπέδου M2 είναι ο ορισμός ενός μετα-μοντέλου που περιγράφει την ίδια τη UML. Επίσης, το πρότυπο MOF παρέχει ένα χρήσιμο εργαλείο στους προγραμματιστές, επιτρέποντας υψηλού επιπέδου σχεδίαση. Πιο συγκεκριμένα το MOF παρέχει φορητότητα και επεκτασιμότητα των μοντέλων που περιγράφει από εφαρμογή σε εφαρμογή, μέσω δικτύου, καθώς και τη δυνατότητα αποθήκευσής τους χρησιμοποιώντας πρότυπα όπως το XMI.

Το **XML Metadata Interchange** (**XMI**) είναι ένα πρότυπο του OMG, το οποίο σχεδιάστηκε για την ανταλλαγή μετα-δεδομένων μέσω της γλώσσας XML. Μπορεί να χρησιμοποιηθεί για την περιγραφή κάθε τύπου μετα-δεδομένων που ακολουθούν το πρότυπο MOF. Ο στόχος του OMG με το πρότυπο του XMI ήταν να ορίσει δύο κατηγορίες μοντελοποίησης δεδομένων, τα αφηρημένα μοντέλα (abstract models) και τα συγκεκριμένα μοντέλα (concrete models). Τα αφηρημένα μοντέλα θα περιγράφουν την σημασιολογία της πληροφορίας, ενώ τα

συγκεκριμένα μοντέλα θα αποτελούν γραφικές απεικονίσεις. Παραδείγματα αφηρημένων μοντέλων αποτελούν όλες οι γλώσσες μοντελοποίησης που βασίζονται στο πρότυπο του MOF, όπως η UML και η SysML [18]. Το XMI χρησιμοποιείται για τη ανταλλαγή μεταδεδομένων ανάμεσα σε ετερογενή περιβάλλοντα ανάπτυξης που βασίζονται στη UML. Μία ακόμα βασική χρήση τους είναι η μεταφορά μοντέλων από εργαλεία μοντελοποίησης σε εργαλεία αυτόματης παραγωγής πηγαίου κώδικα.

Στα πλαίσια αυτής της διπλωματικής το εργαλείο που χρησιμοποιήθηκε για την ανάπτυξη των μοντέλων σε γλώσσα UML είναι το Magic Draw. Μέσω του εργαλείου αυτού εξάγαμε σε μορφή XMI τα μοντέλα που σχεδιαστήκαν και στη συνέχεια εισήχθησαν στο Eclipse Modeling Framework παράγοντας αυτόματα τον πηγαίο κώδικα Java, υλοποιώντας το περιβάλλον - πλαίσιο το οποίο πραγματευόμαστε.

2.4 Περιβάλλοντα Monitoring

Το monitoring, ως μια γενική έννοια αναφέρεται στην διαδικασία της παρακολούθησης της συμπεριφοράς ενός συστήματος κατά τη διάρκεια της λειτουργίας του. Η διαδικασία αυτή είναι απαραίτητη για την αξιολόγηση ενός συστήματος με διάφορα κριτήρια. Μερικά από τα κριτήρια αυτά μπορεί να είναι η ορθότητα λειτουργίας, η απόδοση και η ποιότητα. Κατά την υλοποίηση του συστήματος τα κριτήρια αυτά ελέγχονται διεξοδικά, πραγματοποιώντας ένα όσο το δυνατόν μεγαλύτερο εύρος δοκιμών. Παρά τον έλεγχο του συστήματος πριν την εφαρμογή του στην παραγωγική διαδικασία, η διαδικασία της παρακολούθησης παραμένει απαραίτητη, καθώς σπάνια το σύνολο των προκαταρκτικών δοκιμών μπορούν να καλύψουν όλες τις περιπτώσεις πραγματικής λειτουργίας.

Για κάθε τύπο συστήματος, αλλά πιο συγκεκριμένα για τα συστήματα λογισμικού, η μεθοδολογίες και τα εργαλεία που χρησιμοποιούνται για monitoring διαφοροποιούνται ανάλογα με το αν το σύστημα είναι κατανεμημένο ή όχι, την αρχιτεκτονική του, τις γλώσσες προγραμματισμού που χρησιμοποιήθηκαν για την υλοποίησή του κ.ά. . Στην ενότητα αυτή θα παρουσιάσουμε κάποια περιβάλλοντα monitoring για παρακολούθηση λειτουργίας Υπηρεσιοκεντρικών συστημάτων. Η δυσκολία που προκύπτει στα συγκεκριμένα συστήματα έχει να κάνει με τη χαλαρή διασύνδεση των συστατικών τους, δηλαδή τις υπηρεσίες. Επομένως, η αποτελεσματική παρακολούθηση μεμονωμένα των υπηρεσιών που απαρτίζουν ένα Υπηρεσιοκεντρικό σύστημα δεν είναι αρκετή, αλλά χρειάζεται και ο συσχετισμός των επιμέρους πληροφοριών που συλλέγονται για την εξαγωγή συμπερασμάτων για τη λειτουργία του συστήματος. Επίσης, ένα ακόμα σημαντικό ζήτημα που τίθεται, είναι το εάν η διαδικασία του monitoring θα εκτελείται στην πλευρά του παρόχου της υπηρεσίας, του καταναλωτή ή

κάποιου τρίτου φορέα. Κατά την σχεδίαση μίας υπηρεσίας, ο πάροχος και ο καταναλωτής συνάπτουν μια συμφωνία, η οποία αναφέρεται ως συμφωνία σε επίπεδο υπηρεσίας (service level agreement - SLA), η οποία καθορίζει τον ρόλο του κάθε μετέχοντα στην λειτουργία της υπηρεσίας, αλλά και τις λειτουργικές και μη λειτουργικές απαιτήσεις που πρέπει να ικανοποιεί.

Στο [19], οι συγγραφείς προτείνουν ένα περιβάλλον-πλαίσιο για την παρακολούθηση Υπηρεσιοκεντρικών συστημάτων βασισμένο στην τεχνολογία πρακτόρων (agent technology). Το προτεινόμενο περιβάλλον-πλαίσιο έχει ως στόχο την παρακολούθηση μη λειτουργικών απαιτήσεων που βασίζεται σε ένα μοντέλο ποιότητας που περιγράφεται στο [20], αλλά και την αυτοματοποίηση του ελέγχου των απαιτήσεων, όπως αυτές ορίζονται στη συμφωνία σε επίπεδο υπηρεσίας. Η παρακολούθηση περιλαμβάνει τρεις ομάδες πρακτόρων, στην πλευρά του παρόχου, στην πλευρά του καταναλωτή και έναν τρίτο παράγοντα, οποίος καλείται να εξασφαλίσει την αξιοπιστία ανάμεσα στους δύο προηγούμενους. Οι ομάδες του παρόχου και του καταναλωτή παρακολουθούν τα μηνύματα SOAP που ανταλλάσσονται και κρατούν ένα ιστορικό καταγραφής. Σε περίπτωση που σε κάποια από τις δύο πλευρές παρουσιαστεί παραβίαση κάποιας απαίτησης που περιγράφεται στη συμφωνία σε επίπεδο υπηρεσίας, αποστέλλεται μια ειδοποίηση στην ομάδα πρακτόρων του τρίτου παράγοντα. Η ομάδα αυτή ζητά τα αρχεία καταγραφής και από τις δυο πλευρές και γνωρίζοντας τους κανόνες που αναφέρονται στην συμφωνία σε επίπεδο υπηρεσίας, ελέγχει την ύπαρξη παραβίασης συμπεραίνοντας παράλληλα και τον υπαίτιό της.

Ο Robinson W. N. στο [21], προτείνει ένα περιβάλλον-πλαίσιο για την παρακολούθηση διαδικασιών BPEL. Η μεθοδολογία που ακολουθεί το συγκεκριμένο περιβάλλον-πλαίσιο επιτρέπει την παρακολούθηση λειτουργικών και μη λειτουργικών απαιτήσεων. Επίσης, η διαδικασία της παρακολούθησης διεξάγεται από τη μεριά του πελάτη με τη χρήση του προτύπου WS-Policy , με βάση το οποίο μοντελοποιούνται οι απαιτήσεις, οι οποίες πρέπει να ικανοποιούνται κατά την εκτέλεση των Διαδικτυακών υπηρεσιών που συμμετέχουν στη διαδικασίας BPEL. Πιο συγκεκριμένα, όλοι περιορισμοί που προκύπτουν από τις απαιτήσεις του συστήματος, γράφονται στη γλώσσα WS-CoL (Web Service Constraint Language) [22].

Οι συγγραφείς στο [23], παρουσιάζουν ένα περιβάλλον-πλαίσιο για monitoring Υπηρεσιοκεντρικών συστημάτων, δίνοντας έμφαση στην ανάγκη για δυναμική προσαρμογή των πολιτικών παρακολούθησης του συστήματος. Αναλυτικότερα, οι υπηρεσίες, οι οποίες είναι οργανωμένες σε τοπολογίες, περιγράφονται από σημασιολογίες (semantics) που προκύπτουν από τις ιδιότητές τους. Οι σημασιολογίες αυτές παίζουν σημαντικό ρόλο ως μηχανισμός αναφοράς σε συγκεκριμένες υπηρεσίες στα σενάρια παρακολούθησης που ορίζονται από τον χρήστη. Τα σενάρια αυτά ορίζουν τις υπηρεσίες που πρέπει να τεθούν υπό παρακολούθηση, αλλά και τις μετρικές (metrics) στις οποίες θα εξεταστούν. Το περιβάλλον-

πλαίσιο που παρουσιάζεται, αρχιτεκτονικά ορίζεται από τέσσερα στρώματα. Το πρώτο στρώμα είναι υπεύθυνο για την εύρεση κάθε φορά των υπό παρακολούθηση υπηρεσιών. Το δεύτερο στρώμα συλλέγει τα δεδομένα αυτά, ενώ το τρίτο χρησιμοποιώντας μεθόδους επεξεργασίας σύνθετων γεγονότων (complex event processing) υπολογίζει τα αποτελέσματα των απαιτούμενων μετρικών. Τέλος το τέταρτο στρώμα απλά παρουσιάζει τα αποτελέσματα στον τελικό χρήστη.

2.5 Περιβάλλοντα Ελέγχου Συμμόρφωσης

Στην προηγούμενη ενότητα παρουσιάσαμε ένα σύνολο μεθόδων που χρησιμοποιούνται ή έχουν προταθεί για την παρακολούθηση των Υπηρεσιοκεντρικών συστημάτων. Η διαδικασία της παρακολούθησης έχει ως αποτέλεσμα τη συγκέντρωση σημαντικής πληροφορίας για την εκτέλεση του συστήματος. Η πληροφορία αυτή αξιοποιείται από κάποιους τελικούς χρήστες οι οποίοι ελέγχουν την ορθή λειτουργία του συστήματος. Μια τέτοια ομάδα χρηστών είναι αυτή που πραγματοποιεί τον έλεγχο για τη συμμόρφωση του συστήματος με τι πολιτικές που αναπτύξαμε στο κεφάλαιο 1. Η απουσία ενός ικανοποιητικού αριθμού εργαλείων, τα οποία υποστηρίζουν την διαδικασία του ελέγχου συμμόρφωσης ενός συστήματος με διάφορες πολιτικές, έχει τραβήξει τα τελευταία χρόνια το ενδιαφέρων των ερευνητών από τον ακαδημαϊκό και επιχειρησιακό κόσμο. Στην ενότητα αυτή θα παρουσιάσουμε διάφορες μεθοδολογίες που έχουν προταθεί για τον έλεγχο της συμμόρφωσης σε διαφόρων ειδών πολιτικές.

Μια πρώτη προσέγγιση για τον έλεγχο της συμμόρφωσης, γίνεται με την προσθήκη σημειολογίας στη σχεδίαση των επιχειρησιακών διαδικασιών. Ένα τέτοιο παράδειγμα αποτελεί το [24], όπου οι Wolter και Schaad προτείνουν μια επέκταση του μοντέλου της BPMN προκειμένου να υποστηρίξει περιορισμούς εξουσιοδότησης (authorization constraints) και υποδείγματα δέσμευσης πόρων (resource allocation patterns). Πιο αναλυτικά ορίζονται κάποια συμπληρωματικά αντικείμενα τα οποία επιτρέπουν την μοντελοποίηση των καθηκόντων του κάθε συμμετέχοντα σε μια εργασία (task) μιας επιχειρησιακής διαδικασίας αλλά των εξουσιών που έχει κατά την εκτέλεσή της. Στο [25], οι συγγραφείς ορίζουν την Formal Contract Language (FCL), η οποία είναι μια γλώσσα που μοντελοποίησης κανόνων και τη χρησιμοποιούν ως μέθοδο επισημείωσης περιορισμών σε επιχειρησιακές διαδικασίες.

Στη συνέχεια, στο [26] οι συγγραφείς προτείνουν την εισαγωγή κάποιων υποδειγμάτων ελέγχου (control patterns), εισάγοντας έτσι στη διαδικασία της μοντελοποίησης των επιχειρησιακών διαδικασιών με ένα σύνολο ελέγχων.

Ένα επίσης ενδιαφέρον περιβάλλον-πλαίσιο που έχει προταθεί και υλοποιηθεί για τη διαχείριση του ελέγχου συμμόρφωσης είναι το ευρωπαϊκό έργο COMPAS το οποίο παρουσιάζεται στα [27,28] με αρκετή λεπτομέρεια. Ο διαδικασία του ελέγχου περιλαμβάνει τρία στάδια. Το πρώτο στάδιο είναι αυτό της σχεδίασης στο οποίο οι προς έλεγχο πολιτικές μοντελοποιούνται με την χρήση Domain Specific Languages (DSLs). Το δεύτερο στάδιο περιλαμβάνει την εκτέλεση και τη παρακολούθηση των διαδικασιών. Τέλος, το τρίτο στάδιο αφορά την ανάλυση των γεγονότων που έχουν καταγραφεί κατά την παρακολούθηση της εκτέλεση, προκειμένου να εξαχθούν αποτελέσματα για τη συμμόρφωση του συστήματος, τα οποία και παρουσιάζονται στον τελικό χρήστη. Το συγκεκριμένο περιβάλλον πλαίσιο έχει την ικανότητα να εντοπίζει παραβιάσεις πολιτικών κατά τον χρόνο εκτέλεσης, όποτε αυτό είναι δυνατό.

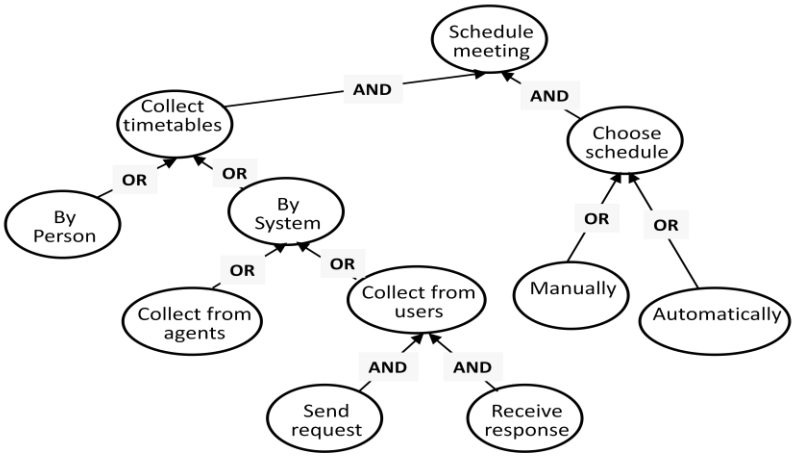
Μέχρι τώρα έχουμε αναφέρει μόνο περιβάλλοντα τα οποία πραγματοποιούν έλεγχο συμμόρφωσης είτε κατά τον χρόνο εκτέλεσης, είτε μετά το πέρας της επιχειρησιακής διαδικασίας. Στο [29], οι συγγραφείς παρουσιάζουν μια μεθοδολογία για στατικό έλεγχο της συμμόρφωσης, πριν την εκτέλεση της διαδικασίας. Πιο συγκεκριμένα, οι επιχειρησιακές διαδικασίες σχεδιάζονται αρχικά με τη γλώσσα BPEL και οι περιορισμοί, που αντιπροσωπεύουν τις πολιτικές στις οποίες πρέπει να συμμορφώνεται το σύστημα, μοντελοποιούνται στη γλώσσα Business Property Specification Language (BPSL). Στη συνέχεια οι BPEL διαδικασίες μετασχηματίζονται σε Π-Λογισμό (Pi-Calculus) και έπειτα σε κάποιο Finite State Machine (FSM). Παράλληλα οι περιορισμοί από BPSL μετασχηματίζονται σε Linear Temporal Logic (LTL) και με τεχνικές ελέγχου μοντέλων (model checking) τα μετασχηματισμένα μοντέλα συγκρίνονται μεταξύ τους, ώστε να επιβεβαιωθεί αν οι επιχειρησιακές διαδικασίες συμμορφώνονται με τους περιορισμούς που έχουν προδιαγραφεί.

2.6 Μοντέλα Στόχων

Η σημερινή αυξανόμενη πολυπλοκότητα των συστημάτων οδήγησαν στην ανάπτυξη ενός νέου κλάδου της Μηχανικής Λογισμικού, την Μηχανική Απαιτήσεων. Ο κλάδος αυτός ασχολείται με την εξαγωγή και την ανάλυση απαιτήσεων συστημάτων. Ένα βασικό εργαλείο της Μηχανικής Απαιτήσεων για την μοντελοποίηση λειτουργικών και μη-λειτουργικών απαιτήσεων είναι τα μοντέλα στόχων (goal models).

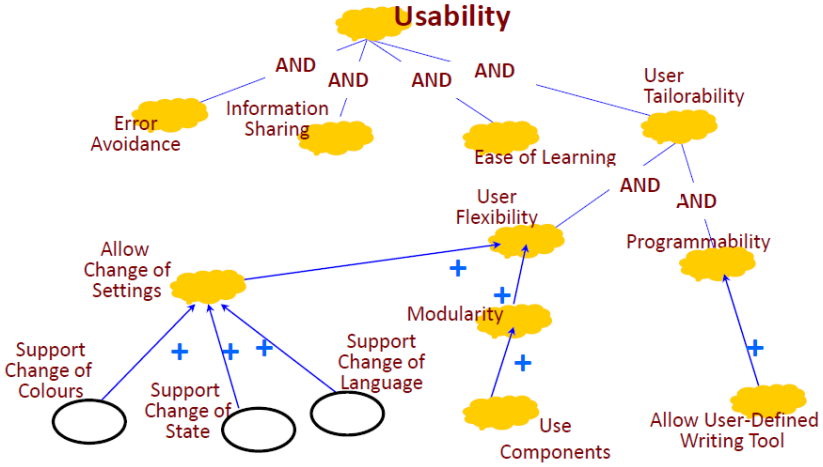
Τα μοντέλα στόχων είναι γράφοι, των οποίων κάθε κόμβος αποτελεί ένα στόχο (goal). Αναλύοντας ένα μοντέλο στόχων από πάνω προς τα κάτω, όπως στην Εικόνα βλέπουμε ότι κάθε στόχος συντίθεται από υποστόχους, μέσω λογικών συζεύξεων ή διαζεύξεων. Όταν ένας

στόχος G συντίθεται μέσω σύζευξης (AND) από τους υποστόχους G_1,… G_n, τότε προκειμένου να ικανοποιηθεί ο G, θα πρέπει να ικανοποιηθούν όλοι οι υποστόχοι. Αντίστοιχα, όταν ένας στόχος G συντίθεται μέσω διάζευξης (OR) από τους υποστόχους G_1,… G_n, τότε προκειμένου να ικανοποιηθεί ο G, θα πρέπει να ικανοποιηθεί τουλάχιστον ένας υποστόχος.



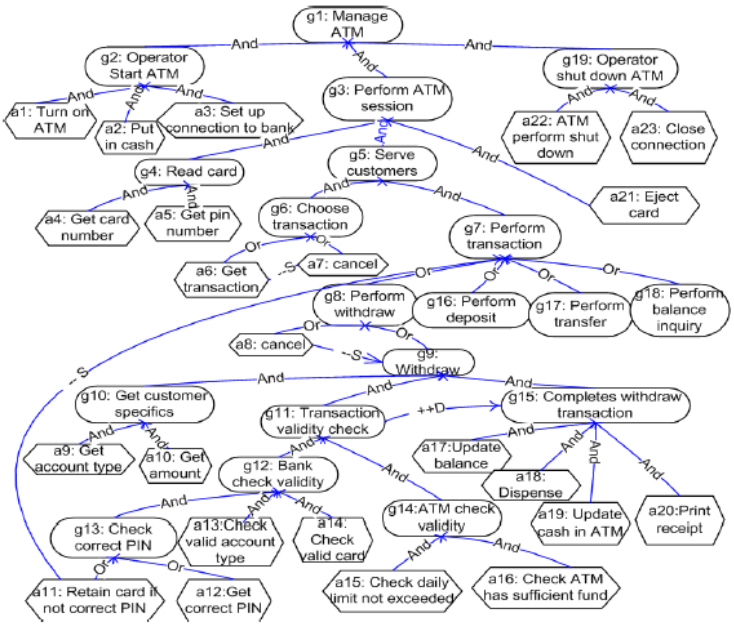
Εικόνα 2.6.1: Παράδειγμα απλού μοντέλου στόχων

Οι παραπάνω στόχοι, με βάση το [30], χαρακτηρίζονται ως hard goals και αναπαριστούν μόνο λειτουργικές απαιτήσεις. Η ικανοποίηση το κόμβου πατέρα εξαρτάται αποκλειστικά από τον λογική ζεύξη του με τα παιδιά του και την ικανοποίησή τους. Τα μοντέλα στόχων όμως, παρέχουν έναν ακόμα τύπο στόχων, τους soft goals. Η ικανοποίηση των στόχων αυτών δεν εξαρτάται από τη θετική ή αρνητική συνεισφορά που προσδίδουν τα παιδιά τους. Η θετική συνεισφορά αναπαριστάται με το σύμβολο "+" και η αρνητική με το σύμβολο "–".



Εικόνα 2.6.2: Παράδειγμα μοντέλου στόχων με soft goals

Για την κατηγορία των hard goals, υπάρχει μια επιπλέον δυνατότητα αναπαράστασης της επιρροής που μπορεί να έχει ένας στόχος σε έναν άλλο με τη χρήση συνδέσμων συνεισφοράς: $\xrightarrow{--S}$, $\xrightarrow{++S}$, $\xrightarrow{--D}$ και $\xrightarrow{++D}$. Δοσμένων δύο στόχων G_1 και G_2, η σύνδεση $G_1 \xrightarrow{++S} G_2$ (αντίστοιχα $G_1 \xrightarrow{--S} G_2$) σημαίνει ότι αν ο G_1 ικανοποιείται, τότε και ο G_2 ικανοποιείται (αντίστοιχα αποτυγχάνει). Οι σύνδεσμοι $\xrightarrow{--D}$ και $\xrightarrow{++D}$ έχουν δυαδική σχέση με τους $\xrightarrow{--S}$ και $\xrightarrow{++S}$ εναλλάσσοντας την ικανοποιησιμότητα με την μη ικανοποιησιμότητα.



Εικόνα 2.6.3: Παράδειγμα ολοκληρωμένου μοντέλου στόχων [31]

Η ευρεία αποδοχή των μοντέλων στόχων από τη βιομηχανία, οφείλεται στα παρακάτω χαρακτηριστικά τους:

- **Απλότητα.** Τα μοντέλα στόχων βασίστηκαν πάνω σε τεχνικές αναπαράστασης απαιτήσεων που ήταν γνωστές στη βιομηχανία, παρέχοντας έτσι ένα οικείο εργαλείο σε συνδυασμό με ένα ισχυρό θεωρητικό υπόβαθρο.
- **Πληρότητα**. Στην προδιαγραφή των απαιτήσεων ενός συστήματος, δύο από τα βασικά προβλήματα ήταν η μοντελοποίηση των μη λειτουργικών απαιτήσεων και αναπαράσταση των μη αυστηρών σχέσεων μεταξύ των απαιτήσεων. Τα μοντέλα

στόχων επιλύουν τα προβλήματα, παρέχοντας την κατάλληλη σημειολογία, που παρουσιάστηκε νωρίτερα.

- **Σύνδεση με Άλγεβρα Boole**. Η σχέση των μοντέλων στόχων με την Άλγεβρα Boole, παρέχει τη δυνατότητα χρήσης μεθόδων της Τεχνητής Νοημοσύνης για την εξαγωγή συμπερασμάτων για τις απαιτήσεις ενός συστήματος. Ένα τέτοιο παράδειγμα παρουσιάζεται στο [31].

2.7 Περιβάλλοντα Συμπερασμού

Η εξέλιξη της τεχνητής νοημοσύνης [32,33] τα τελευταία χρόνια έχει συμβάλει σημαντικά στη ενίσχυση πολλών άλλων επιστημονικών κλάδων. Το κίνητρο της τεχνητής νοημοσύνης είναι να δημιουργήσει μαθηματικά τεκμηριωμένες μεθοδολογίες και εργαλεία, με σκοπό την επίλυση προβλημάτων λογικής, των οποίων η δυσκολία πολλές φορές ξεπερνάει τις δυνατότητες του ανθρώπινου μυαλού. Η έρευνα για τέτοιες μεθοδολογίες έχει φέρει ιδιαίτερα αξιόλογα αποτελέσματα, τα οποία βρίσκουν ένα ευρύ φάσμα εφαρμογών. Στη συνέχεια αυτής της ενότητας θα αναφερθούμε σε ορισμένα από τα αποτελέσματα αυτά, όπως η Λογική Πρώτης Τάξης και τα Μαρκοβιανά Λογικά Δίκτυα.

2.7.1 Λογική Πρώτης Τάξης

Η Λογική Πρώτης Τάξης (First Order Logic) είναι ένας φορμαλιστικός τρόπος αναπαράστασης γνώσης και εκτέλεσης συμπερασμών, ο οποίος προσπαθεί να προσεγγίσει την ανθρώπινη λογική σκέψη. Τα θεμέλια της Λογικής Πρώτης Τάξης είναι η Μαθηματική Λογική και κατ' επέκταση η Προτασιακή Λογική. Η λειτουργία της σε γενικές γραμμές αφορά στον ορισμό ενός κόσμου, στον οποίο ισχύουν κάποιοι κανόνες. Με τη χρήση τεχνικών συμπερασμού (reasoning) μπορούν να απαντηθούν ερωτήματα που σχετίζονται με τον κόσμο αυτό.

Η Λογική Πρώτης Τάξης επεκτείνει την προτασιακή λογική εισάγοντας τους όρους (terms), τα κατηγορήματα (predicates) και τους ποσοδείκτες. Για την επίτευξη της επέκτασης αυτής υιοθετείται το παρακάτω αλφάβητο :

- Σταθερές: a,b,c κ.λ.π.
- Μεταβλητές: x,y,z κλπ

- Σύμβολα συναρτήσεων: f,g κ.λ.π. Οι συναρτήσεις δέχονται ένα πλήθος ορισμάτων το οποίο καθορίζει και την τάξη της συνάρτησης.
- Σύμβολα κατηγορημάτων: P,Q κ.λ.π. Τα κατηγορήματα χρησιμοποιούνται για να δηλώνουν σχέσεις που ισχύουν σε έναν κόσμο. Το πλήθος των ορισμάτων του κατηγορήματος ορίζει την τάξη του κατηγορήματος.
- Συνδέσμους: ¬ (άρνηση), Λ (σύζευξη), V (διάζευξη), → (συνεπαγωγή) και ↔(ισοδυναμία).
- Δύο ποσοδείκτες: ∃(υπαρξιακός ποσοδείκτης) και ∀(καθολικός ποσοδείκτης).
- Τρία σημεία στίξης: ``(", ``)" και ``," .

Ας δούμε τώρα πώς ορίζονται οι όροι (terms):

- Μια σταθερά είναι όρος.
- Μια μεταβλητή είναι όρος.
- Αν *f* είναι μια *n*-αδική συνάρτηση και $t_{1,...,}t_n$ είναι *n* στο πλήθος όροι, τότε η $f(t_{1,...,}t_n)$ είναι όρος.

Οι όροι κατασκευάζονται με τους τρεις παραπάνω κανόνες και μόνον αυτούς.

Θα ονομάζουμε καλά σχηματισμένους τύπους ή απλά τύπους, τις ``οντότητες" που κατασκευάζονται από τους επόμενους κανόνες:

- Αν P είναι ένα κ-δικό κατηγόρημα και $t_{1,...,}t_κ$ είναι κ-στο πλήθος όροι, τότε $P(t_{1,...,}t_κ)$ είναι ένας *ατομικός τύπος* (*atomic formula*) ή απλά *άτομο* (*atom*) ή *θετικό λεκτικό στοιχείο* (*positive literal*).
- Αν *φ* είναι τύπος, τότε και ο ¬ *φ* είναι τύπος.
- Αν *φ,ψ* είναι τύποι, τότε και οι (*φ*Λ*ψ*),(*φ*V*ψ*),(*φ*→*ψ*),(*φ*↔*ψ*) είναι τύποι.
- Αν φ είναι τύπος και *x* μεταβλητή, τότε και οι (∃*x*)*φ*,(∀*x*)*φ* είναι τύποι.

Οι τύποι κατασκευάζονται με τους τέσσερις παραπάνω κανόνες και μόνον αυτούς.

Βασικός όρος (ground term) (ή τύπος ή άτομο), είναι ένας όρος (ή τύπος ή άτομο) που δεν περιέχει καμία μεταβλητή. Για παράδειγμα, ο όρος *f*(*a, b*) και ο τύπος *Q*(*a, g*(*b, c*)) είναι βασικοί, ενώ ο όρος *f*(*a, x*) αλλά και ο τύπος (∀*y*)Q(a,y)ΛR(x) δεν είναι.

Η Λογική Πρώτης Τάξης συνήθως χρησιμοποιείται με δύο τρόπους. Ο πρώτος τρόπος είναι η παραγωγή νέας γνώσης από αυτή που ήδη έχει αναπαρασταθεί και ο δεύτερος είναι η αποτίμηση λογικών υποθέσεων με τη χρήση μεθόδων συμπερασμού. Στα πλαίσια αυτής της διπλωματικής θα μελετήσουμε τη δεύτερη χρήση. Πριν όμως συνεχίσουμε είναι σημαντικό να εισάγουμε δύο βασικές έννοιες: την αντικατάσταση (substitution) και την ενοποίηση (unification). Η έννοια της αντικατάστασης αφορά την αντικατάσταση των μεταβλητών που εμφανίζονται σε έναν τύπο από κάποιους όρους. Η αντικατάσταση παριστάνεται με

$\{X_i/t_i\}$όπου X_i η μεταβλητή που θα αντικατασταθεί και t_i ο όρος που θα την αντικαταστήσει. Για παράδειγμα η αντικατάσταση {X/nick}στον τύπο:

Parent(X,john)

θα δώσει τον τύπο:

Parent(nick,john)

Στη συνέχεια, ως ενοποίηση ορίζουμε τη διαδικασία κατά την οποία δύο εκφράσεις γίνονται συντακτικά όμοιες με τη χρήση αντικαταστάσεων. Για παράδειγμα οι ακόλουθες προτάσεις:

Parent(X,john)

Parent(nick,Y)

ενοποιούνται με τις αντικαταστάσεις {X/nick,Y/john}.

Οι έννοιες της αντικατάστασης και της ενοποίησης αποτελούν βασικά εργαλεία για την εκτέλεση αλγορίθμων συμπερασμού που υλοποιούνται από γλώσσες λογικού προγραμματισμού όπως η Prolog [34]. Παρακάτω δίνεται ένα παράδειγμα χρήσης της γλώσσα αυτής, με σκοπό να γίνει σαφής ο τρόπος χρήσης της Λογικής Πρώτης Τάξης:

```
parent(kim,holly).
parent(margaret,kim).
parent(margaret,kent).
parent(esther,margaret).
parent(herbert,margaret).
parent(herbert,jean).
greatgrandparent(GGP,GGC) :- parent(GGP,GP),parent(GP,P), parent(P,GGC).
```

Στο παραπάνω παράδειγμα βλέπουμε ότι αρχικά ορίζεται μια γνωσιακή βάση (knowledge base) η οποία αποτελείται από βασικούς τύπους ή αλλιώς γεγονότα, όπως parent(kim,holy) κ.λ.π. Οι βασικοί τύποι ορίζουν σχέσεις μεταξύ αντικειμένων και πιο συγκεκριμένα τη σχέση γονέα ανάμεσα στα πρόσωπα (τα αντικείμενα) του κόσμου που ορίζεται. Έπειτα δηλώνεται ένα κανόνας που υπαγορεύει τη συνθήκη για να είναι κάποιο πρόσωπο προπάππους ενός άλλου προσώπου. Στον κανόνα, όπως βλέπουμε χρησιμοποιούνται μεταβλητές. Μετά τις παραπάνω δηλώσεις σε κάποιο σύστημα Prolog μπορούμε να εισάγουμε κάποιο ερώτημα, όπως «Έχει η esther κάποιο δισέγγονο και αν ναι ποιό;» με την εξής μορφή:

?- greatgrandparent(esther, GreatGrandchild).

Το σύστημα τη Prolog τότε με τη χρήση της γνωσιακής βάσης και την εκτέλεση αλγορίθμων συμπερασμού που περιγράφονται με λεπτομέρεια στα [32-34], θα προσπαθήσει να βρει κάποια ενοποίηση για την μεταβλητή GreatGrandchild. Η απάντηση του συστήματος τελικά θα είναι :

GreatGrandchild = holly ;

No

Το οποίο σημαίνει πως η esther έχει ένα δισέγγονο, την holly και κανένα άλλο. Στο σημείο αυτό πρέπει να προσθέσουμε ότι η Prolog κάνει μια θεώρηση κλειστού κόσμου, το οποίο σημαίνει ότι αν κάτι δεν υπάρχει στην γνωσιακή βάση της, είναι αυτομάτως ψευδές. Την αδυναμία αυτή θα την αντιμετωπίσουμε στη συνέχεια με την εισαγωγή εννοιών από τη θεωρία πιθανοτήτων και τα Μαρκοβιανά Λογικά Δίκτυα.

2.7.2 Μαρκοβιανά Λογικά Δίκτυα

Η θεωρία των Μαρκοβιανών Λογικών Δικτύων συνδιάζει τα πλεονεκτήματα της Λογικής Πρώτης Τάξης με αυτά των πιθανοτικών γραφικών μοντέλων. Όπως περιγράφεται στα [35,36], το Μαρκοβιανό Δίκτυο (Markov Network) είναι ένα μοντέλο της από κοινού κατανομής ενός συνόλου μεταβλητών $X = (X_1, X_2, \ldots, X_n)$. Το μοντέλο αυτό αποτελείται από έναν μη κατευθυνόμενο γράφο G και μια σειρά από συναρτήσεις δυναμικού (potential functions) φ_k. Κάθε κόμβος του γράφου αντιστοιχεί σε μια μεταβλητή του συνόλου X και υπάρχει μια συνάρτηση φ_k για κάθε κλίκα του γράφου. Οι συναρτήσεις δυναμικού έχουν θετικό πραγματικό πεδίο τιμών και κάθε μία από αυτές αναπαριστά την κατάσταση της αντίστοιχης κλίκας. Λαμβάνοντας υπόψη τα παραπάνω ο τύπος που δίνει την από κοινού κατανομή είναι ο εξής:

$$P(x) = \frac{1}{Z} \prod_k \phi_k(x_k) \qquad (2.7.1)$$

όπου x_k είναι η κατάσταση των μεταβλητών της k – κλίκας και Z η συνάρτηση διαμέρισης:

$$Z = \sum_x \prod_k \phi_k(x_k) \qquad (2.7.2)$$

Τα Μαρκοβιανά Δίκτυα αναπαριστώνται συχνά ως λογαριθμογραμμικά μοντέλα (log – linear models). Σε αυτή την περίπτωση οι συναρτήσεις δυναμικού αντικαθιστώνται από εκθετικά αθροίσματα βαρών των χαρακτηριστικών (features) της κατάστασης. Ένα χαρακτηριστικό είναι συνάρτηση της κατάστασης με πραγματικό πεδίο τιμών. Επομένως, η από κοινού κατανομή παίρνει την παρακάτω μορφή:

$$P(x) = \frac{1}{Z} \exp\left(\sum_i w_i f_i(x) \right) \qquad (2.7.3)$$

Στον παραπάνω τύπο τα βάρη αναπαριστώνται με w_i και τα χαρακτηριστικά με $f_i(x)$, για τα οποία θα θεωρήσουμε πως $f_i(x) \epsilon \{0,1\}$.

Όπως είδαμε στην προηγούμενη υποενότητα μια γνωσιακή βάση στη Λογική Πρώτης Τάξης αναπαριστά ένα σύνολο αυστηρών κανόνων (hard constraints), οι οποίοι ορίζουν έναν πιθανό κόσμο. Όταν ένας έστω και ένας κανόνας παραβιαστεί, τότε η πιθανότητα ύπαρξης αυτού του κόσμου καθίσταται μηδενική. Η θεωρία των Μαρκοβιανών Λογικών Δικτύων με τη χρήση των μεθόδων αναπαράστασης γνώσης της Λογικής Πρώτης τάξης και τις πιθανοτικές μεθόδους που εισάγουν τα Μαρκοβιανά Δίκτυα, επιχειρεί να ξεπεράσει την αδυναμία αυτή. Η Μαρκοβιανή λογική δίνει τη δυνατότητα χαλάρωσης των κανόνων, επιτρέποντας έτσι σε έναν κόσμο, όπου παραβιάζεται κάποιος κανόνας, να έχει πιθανότητα ύπαρξης. Όσο περισσότεροι κανόνες παραβιάζονται, τόσο λιγότερο πιθανή γίνεται και η ύπαρξη του κόσμους αυτού. Στο σημείο αυτό, πρέπει να σημειωθεί ότι η συμβολή κάθε κανόνα στο Μαρκοβιανό Λογικό Δίκτυο δεν έχει την ίδια βαρύτητα. Αυτό σημαίνει ότι η ικανοποίηση ενός πολύ σημαντικού κανόνα αυξάνει περισσότερο την πιθανότητα ύπαρξης του κόσμου, από ότι ένα λιγότερο σημαντικός κανόνας. Για τον λόγο αυτό κάθε κανόνας ενός Μαρκοβιανού Λογικού Δικτύου, οποίος αναπαριστάται με κάποιο τύπο Λογικής Πρώτης Τάξης, συνοδεύεται από κάποιο βάρος w. Το βάρος αυτό απεικονίζει τη σημασία που έχει ο συγκεκριμένος τύπος για το Μαρκοβιανό Λογικό Δίκτυο. Λαμβάνοντας υπόψη όλα τα παραπάνω μπορούμε να δώσουμε τον επίσημο ορισμό του Μαρκοβιανού Λογικού Δικτύου:

> Ένα Μαρκοβιανό Λογικό Δίκτυο είναι ένα σύνολο L από ζεύγη (F_i,w_i), όπου F_i είναι ένας τύπος σε Λογική Πρώτης Τάξης και w_i ένας πραγματικός αριθμός. Μαζί με ένα πεπερασμένο σύνολο σταθερών $C = \{c_1,c_2,\ldots,c_{|C|}\}$, ορίζουν ένα Μαρκοβιανό Λογικό Δίκτυο $M_{L,C}$ ως ακολούθως:
>
> 1. Ένα ψ περιέχει έναν δυαδικό κόμβο για κάθε πιθανή θεμελίωση (grounding) κάθε κατηγορήματος που εμφανίζεται στο L. Η τιμή του κόμβου είναι 1 εάν το βασικό άτομο είναι αληθές και 0 στην αντίθετη περίπτωση.
> 2. Ένα $M_{L,C}$ περιέχει ένα χαρακτηριστικό για κάθε πιθανή θεμελίωση κάθε τύπου F_i στο L. Η τιμή του χαρακτηριστικού αυτού είναι 1εάν η θεμελίωση του τύπου είναι αληθής, και 0 διαφορετικά. Το βάρος του χαρακτηριστικού είναι το w_i που σχετίζεται με το F_i στο L.

Ένα Μαρκοβιανό Λογικό Δίκτυο μπορεί να παραγάγει διαφορετικά δίκτυα με διαφορετικά σύνολα σταθερών. Τα δίκτυα αυτά ονομάζονται βασικά Μαρκοβιανά Δίκτυα (ground Markov Networks) και ενώ μπορεί να διαφέρουν σε έκταση, παρουσιάζουν ομοιότητες όσων αφορά τη δομή τους και τις παραμέτρους που καθορίζονται από το Μαρκοβιανό Λογικό Δίκτυο. Επομένως, η κατανομή πιθανότητας όλων των x που ορίζονται από το βασικό Μαρκοβιανό Δίκτυο $M_{L,C}$, έχει την ακόλουθη μορφή:

$$P(x) = \frac{1}{Z}\exp\left(\sum_i w_i n_i(x)\right) = \frac{1}{Z}\left(\prod_i \phi_i(x_i)^{n_i(x)}\right) \qquad (2.7.4)$$

όπου $n_i(x)$ είναι ο αριθμός των αληθών θεμελιώσεων του F_i στο x, το x_i είναι η κατάσταση των ατόμων που εμφανίζονται στο F_i και $\varphi_i(x_i)=e^{w_i}$. Ένα ενδιαφέρον παράδειγμα που παρουσιάζει πως τα Μαρκοβιανά Λογικά Δίκτυα γενικεύουν τη Λογική Πρώτης Τάξης είναι το εξής:

Έστω ένα Μαρκοβιανό Λογικό Δίκτυο που περιέχει τον τύπο $\forall x R(x) \rightarrow S(x)$ με βάρος w και C={A}. Η υπόθεση αυτή οδηγεί σε τέσσερις πιθανούς κόσμους {¬R(A),¬S(A)},{¬R(A),S(A)}, {R(A),¬S(A)} και {R(A),S(A)}. Από την εξίσωση 2.7.4 συμπεραίνουμε ότι P({R(A),¬S(A)}) = $1/(3e^w+1)$ και η πιθανότητα για κάνε έναν από τους υπολοίπους τρείς κόσμους είναι $e^w/(3e^w+1)$. (Ο παρονομαστής δίνεται από τη συνάρτηση διαμέρισης της εξίσωσης 2.7.3).

Η θεωρία των Μαρκοβιανών Λογικών Δικτύων περιλαμβάνει δύο αλγοριθμικές διαδικασίες που μπορούν να εκτελεστούν πάνω σε αυτά. Οι διαδικασίες αυτές είναι οι εξής:

- Μάθηση (Learning). Η διαδικασία αυτή χρησιμοποιεί μία ή περισσότερες γνωσιακές βάσεις με βασικά άτομα και κάνοντας μια θεώρηση κλειστού κόσμο εξάγει τιμές για τα βάρη των τύπων που συνθέτουν το Μαρκοβιανό Λογικό Δίκτυο. Οι δύο βασικές μέθοδοι μάθησης είναι η discriminative learning και η generative learning, οι οποίες περιγράφονται με λεπτομέρεια στο [35].
- Συμπερασμός (Inference). Η διαδικασία αυτή υλοποιείται από αρκετούς διαφορετικούς αλγορίθμους [35], οι οποίοι επιλέγονται ανάλογα με την εφαρμογή του εκάστωτε Μαρκοβιανού Λογικού Δικτύου. Οι αλγόριθμοι αυτοί με τη χρήση κάποιας γνωσιακής βάσης και του Μαρκοβιανού Λογικού Δικτύου υπολογίζει τις πιθανότητες ύπαρξης των διαφορετικών κόσμων που προκύπτουν. Ο βασικός αλγόριθμος συμπερασμού είναι ο MaxWalkSat, του οποίου ο ψευδοκώδικας παρατίθεται παρακάτω:

```
for i ← 1 to max-tries do
   solution = random truth assignment
   for j ← 1 to max-flips do
      if ∑ weights(sat. clauses) > threshold then
         return solution
      c ← random unsatisfied clause
      with probability p
         flip a random variable in c
      else
         flip variable in c that maximizes
            ∑ weights(sat. clauses)
return failure, best solution found
```

Όταν κατασκευάζεται ένα Μαρκοβιανό Λογικό Δίκτυο, τα βάρη των τύπων μπορούν να προκύψουν είτε από τη διαδικασία της μάθησης, είτε να οριστούν από τον ίδιο τον χρήστη. Σε κάθε περίπτωση είναι απαραίτητη η κατανόηση της έννοιας του βάρους. Ας υποθέσουμε τώρα ότι έχουμε έναν τύπο *F* με βάρος *w*. Αυτό σημαίνει ότι σε έναν κόσμο που ο F είναι αληθής είναι e^w φορές πιο πιθανό να συμβεί από ότι ένας κόσμος στον οποίο είναι ψευδής. Έστω U_t και U_f ο αριθμός των πιθανών κόσμων στους οποίους το F είναι αληθές και ψευδές, αντίστοιχα. Εάν ο F είναι ανεξάρτητος από τους υπόλοιπους βασικούς τύπους, τότε η πιθανότητά τους δίνεται από τον παρακάτω τύπο:

$$P(F) = \frac{1}{1+\frac{U_f}{U_t}} e^{-w}$$

και λύνοντας ως προς w:

$$w = \log\frac{P(F)}{P(\neg F)} - \log\frac{U_t}{U_f}$$

Ο παραπάνω τύπος μπορεί να χρησιμοποιηθεί μόνο σε περίπτωση που ο F δεν μοιράζεται άτομα με άλλους τύπους, κάτι το οποίο είναι δύσκολο πρακτικά να συμβεί.

Στα πλαίσιο της παρούσας διπλωματικής εργασίας το εργαλείο που θα χρησιμοποιήσουμε για την εκτέλεση των αλγορίθμων που σχετίζονται με τα Μαρκοβιανά Λογικά Δίκτυα είναι το Alchemy, λεπτομέρειες για τη λειτουργία του οποίου μπορούν να βρεθούν στο [35].

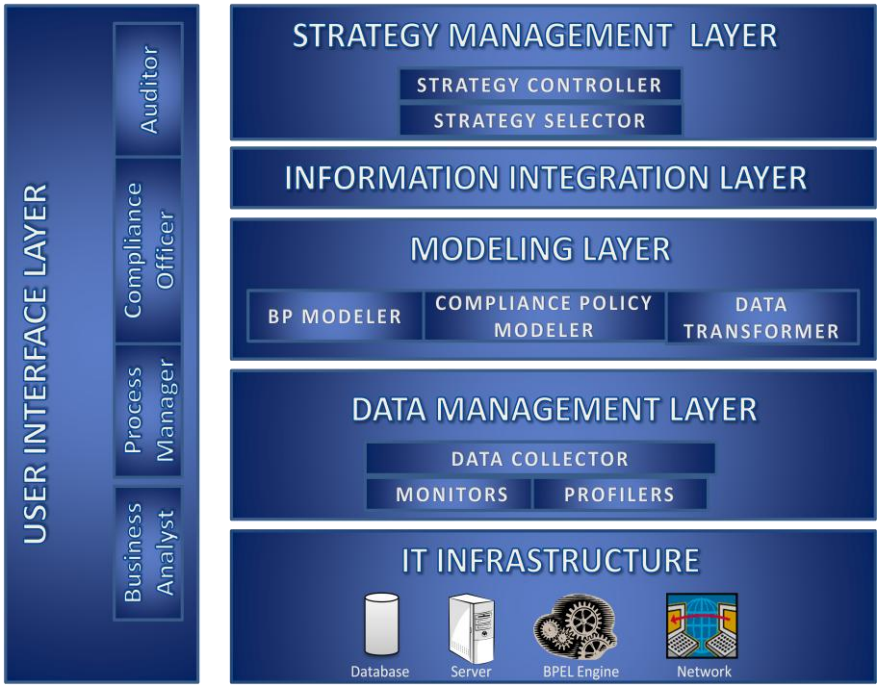
Κεφάλαιο 3: Αρχιτεκτονική του Συστήματος

3.1 Εισαγωγή

Στο κεφάλαιο αυτό θα περιγράψουμε και θα αναλύσουμε την αρχιτεκτονική του συστήματος που πραγματεύεται η παρούσα διπλωματική εργασίας. Αναλυτικότερα, θα αναφερθούμε σε όλα τα δομικά στοιχεία του συστήματος, τις λειτουργικότητές τους αλλά και τον τρόπο διασύνδεσής τους μεταξύ τους. Θα αναπτύξουμε τις σχεδιαστηκές αποφάσεις που πάρθηκαν και τις αρχιτεκτονικές τεχνοτροπίες που χρησιμοποιήθηκαν. Για τον σκοπό αυτό, θα χρησιμοποιήσουμε μπλοκ και ψηφιδικά διαγράμματα της γλώσσας UML.

3.2 Γενική Επισκόπηση Αρχιτεκτονικής

Όπως έχει ήδη αναφερθεί, ο κύκλος ζωής της διαχείρησης επιχειρησιακών διαδικασιών περιλαμβάνει ένα σύνολο προσώπων, με διαφορετικό γνωστικό αντικείμενο και αρμοδιότητες. Την διάκριση αυτή επιλέξαμε να την εκφράσουμε σχεδιαστικά, με τη χρήση μιας Αρχιτεκτονικής Στοιβάδας (Layered Architecture) με Διαφανή Στρωμάτωση (Transparent Layering), που περιλαμβάνει στοιχεία Αρχιτεκτονικής Αυλού/Φίλτρου (Pipe and Filter).



Εικόνα 3.2.1: Αρχιτεκτονική του Συστήματος

IT INFRASTRUCTURE: Η στοιβάδα αυτή αντιπροσωπεύει το σύνολο της τεχνικής υποδομής που εκτελεί την υλοποίηση των επιχειρησιακών διαδικασιών. Η υποδομή αυτή μπορεί να περιλαμβάνει βάσεις δεδομένων, εξυπηρετητές, BPEL engines, αλλά και το δίκτυο επικοινωνίας όλων αυτών.

DATA MANAGEMENT LAYER: Η στοιβάδα αυτή περιλαμβάνει, τη συλλογή, την επεξεργασία και την αποθήκευση των δεδομένων που παράγονται από τις μονάδες τις προηγούμενης στοιβάδας. Τα δεδομένα αυτά καταγράφονται από εργαλεία όπως monitors και profilers που παρακολουθούν την εκτέλεση του συστήματος. Στη συνέχεια, τα δεδομένα αυτά συλλέγονται, επεξεργάζονται ώστε να αποκτήσουν ομοιόμορφη δομή και ύστερα αποθηκεύονται ώστε να είναι προσβάσιμα από τις παραπάνω στοιβάδες.

MODELING LAYER: Στη στοιβάδα αυτή εκτελούνται τρεις αυτόνομες διαδικασίες μοντελοποίησης. Η μοντελοποίηση των επιχειρησιακών διαδικασιών, ορίζοντας τις εργασίες, τις λειτουργίες και τις υπηρεσίες που συνδέονται με αυτές. Η μοντελοποίηση των πολιτικών στις οποίες θα πρέπει να συμμορφώνονται οι επιχειρησιακές διαδικασίες και η μοντελοποίηση των δεδομένων ώστε να είναι συμβατά με τη μεθοδολογία ελέγχου της συμμόρφωσης από παραπάνω στοιβάδες.

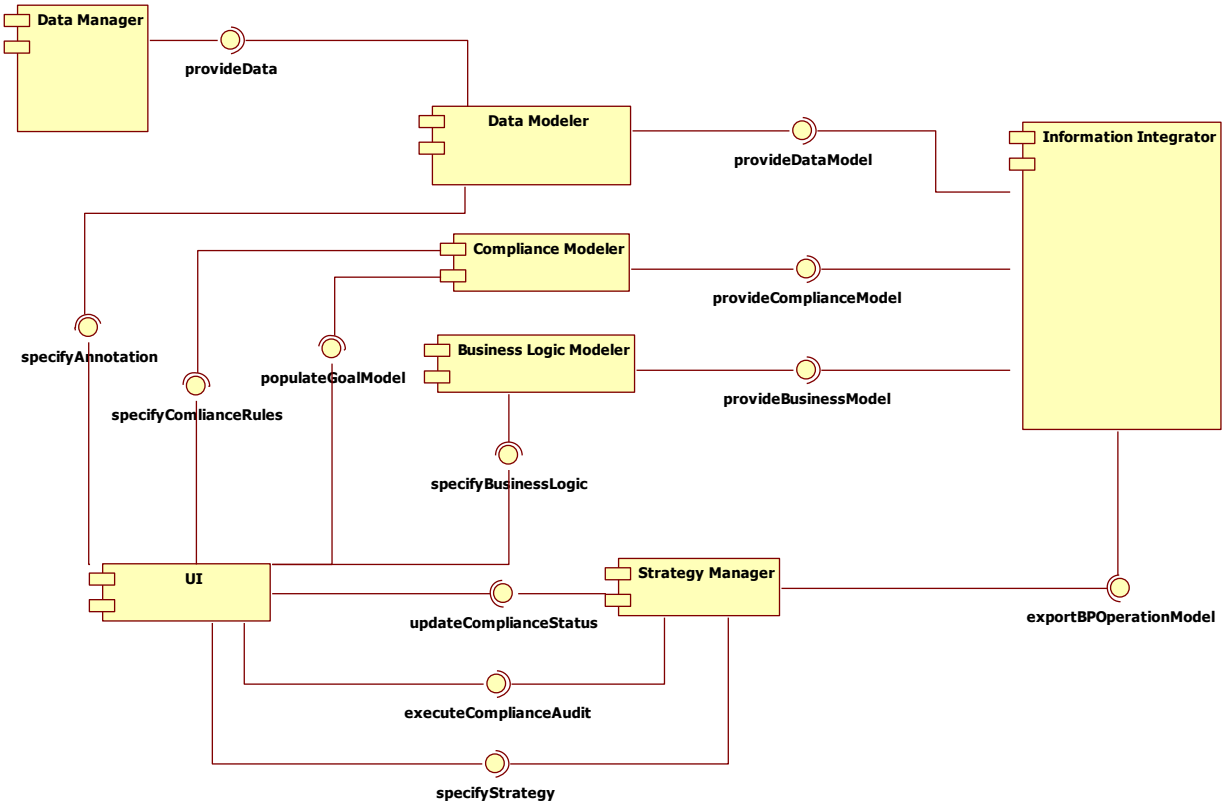
INFORMATION INTEGRATION LAYER: Στη στοιβάδα αυτή ενοποιούνται τα μοντέλα που έχουν παραχθεί από την προηγούμενη στοιβάδα. Πιο συγκεκριμένα, γίνεται μια αντιστοίχιση μεταξύ των μοντέλων των επιχειρησιακών διαδικασιών και των μοντέλων πολιτικών συμμόρφωσης. Στη συνέχεια επιλέγεται το κατάλληλο μοντέλο δεδομένων, το οποίο συσχετίζεται με τα προηγούμενα, παράγοντας ένα ενοποιημένο λειτουργικό μοντέλο.

STRATEGY MANAGEMENT LAYER: Η στοιβάδα αυτή δίνει τη δυνατότητα επιλογής στρατηγικής ελέγχου συμμόρφωσης, η οποία σχετίζεται άμεσα και από τη μέθοδο μοντελοποίησης των δεδομένων στο Data Management Layer. Επίσης σε αυτή τη στοιβάδα βρίσκεται και μηχανισμός εκτέλεσης του ελέγχου συμμόρφωσης.

USER INTERFACE LAYER: Η στοιβάδα αυτή παρέχει στους διάφορους χρήστες του συστήματος διαπροσωπείες, με τις οποίες μπορούν να επικοινωνήσουν με άλλες στοιβάδες, ανάλογα με την κατηγορία που ανήκουν.

3.3 Περιγραφή Δομικών Στοιχείων

Στην ενότητα αυτή παρουσιάζεται το ψηφιδικό διάγραμμα του προτεινόμενου περιβάλλοντος - πλαισίου. Σκοπός του διαγράμματος αυτού είναι να περιγράψει με λεπτομέρεια τα δομικά στοιχεία του συστήματος και τις λειτουργίες που επιτελούν. Παρακάτω παρουσιάζουμε μια γενική όψη του διαγράμματος αυτού, το οποίο παρουσιάζεις όλες τις βασικές ψηφίδες και τους τρόπους αλληλεπίδρασης μεταξύ τους. Στη συνέχεια περιγράφεται κάθε ψηφίδα ξεχωριστά αναλύοντας το εσωτερικό της.



Εικόνα 3.3.1: Ψηφιδικό Διάγραμμα της Αρχιτεκτονικής του Συστήματος

3.3.1 Data Manager Component

Το δομικό αυτό στοιχείο έχει αναλαμβάνει την εξόρυξη δεδομένων από την τεχνική υποδομή πάνω στην οποία εκτελείται μια επιχειρησιακή διαδικασία. Επίσης, αναλαμβάνει την ενοποίηση των εξαγόμενων δεδομένων σε μια κοινή δομή και την αποθήκευσή τους. Τις λειτουργίες αυτές τις εκτελούν τα εξής εσωτερικά δομικά στοιχεία: Monitor, Profiler, Data Collector, Log Conciliator, Storage Processor και Log DB.

Το δομικό στοιχείο **Monitor** παρακολουθεί την εκτέλεση του συστήματος. Πιο συγκεκριμένα, καταγράφει την εκτέλεση διεργασιών του λογισμικού που έχει παραταχθεί της εξυπηρετητές του συστήματος, την κίνηση του δικτύου και μηνύματα μεταξύ εφαρμογών που συνθέτουν το σύστημα. Η μορφή αυτών των δεδομένων εξαρτάται άμεσα από την προέλευσή της. Μέσω αυτού του δομικού στοιχείου παρέχεται η διαπροσωπεία **provideRawData**, με την οποία γίνονται διαθέσιμα τα καταγεγραμμένα δεδομένα σε ακατέργαστη μορφή.

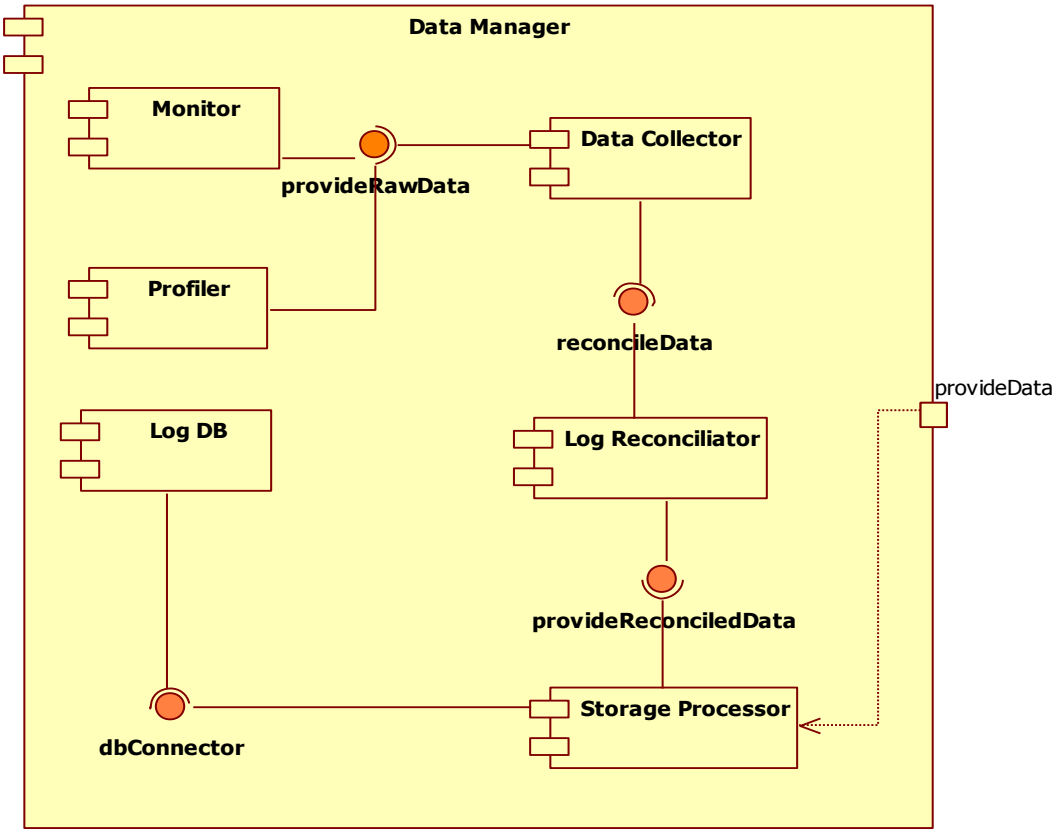
Το δομικό στοιχείο **Profiler** καταγράφει δεδομένα σε πιο μικροσκοπικό επίπεδο. Παρεμβαίνει στην εκτέλεση του κώδικα και παρακολουθεί της κλήσεις των μεθόδων/συναρτήσεων, τη δημιουργία αντικειμένων, των αριθμό των threads, τη μνήμη που καταναλώθηκε αλλά και την επεξεργαστική ισχύ που αξιοποιήθηκε. Ο Profiler παρέχει της τη διαπροσωπεία **provideRawData**, κάνοντας διαθέσιμα τα δεδομένα που έχουν καταγραφεί σε ακατέργαστη μορφή.

Το δομικό στοιχείο **Data Collector** αναλαμβάνει τη συλλογή των ακατέργαστων δεδομένων που παρέχονται από τα δομικά στοιχεί Monitor και Profiler. Η συλλογή αυτή γίνεται μέσω της διαπροσωπείας provideRawData που παρέχουν τα δομικά στοιχεία αυτά. Στη συνέχεια, το σύνολο ακατέργαστων δεδομένων από τις πηγές, γίνονται διαθέσιμα μέσω της διαπροσωπείας **provideCollectedData**.

Το δομικό στοιχείο **Log Reconciliator** επεξεργάζεται τα δεδομένα που έχουν καταγραφή και τα μετατρέπει σε μία κοινή τυποποιημένη μορφή που καθορίζεται από τον χρήστη. Ύστερα, τα δεδομένα αυτά παρέχονται μέσω της διαπροσωπείας **providereconciledData**.

Το δομικό στοιχείο **Storage Processor** είναι υπεύθυνο για την διαχείριση της βάσης δεδομένων στην οποία αποθηκεύονται τα επεξεργασμένα δεδομένα. Παρέχει τη διαπροσωπείες **dbConnector** με την οποία γίνεται η αποθήκευση και εξαγωγή δεδομένων από τη βάση και την εξωτερική διαπροσωπεία **provideData**, μέσω της οποίας διαθέτει τα δεδομένα της βάσης σε άλλα δομικά στοιχεία.

Το δομικό στοιχείο **Log DB** αναπαριστά τη βάση δεδομένων στην οποία αποθηκεύονται τα επεξεργασμένα δεδομένα που έχουν εξαχθεί.



Εικόνα 3.3.2: Ψηφιδικό διάγραμμα του Data Manager

Η λειτουργία του δομικού στοιχείου Data Manager μπορεί να αναλυθεί στα εξής βήματα:

1. Παρακολούθηση και Καταγραφή
2. Συλλογή Δεδομένων
3. Επεξεργασία Δεδομένων
4. Αποθήκευση Δεδομένων

Το δομικό αυτό στοιχείο αντιπροσωπεύει το λειτουργικό κομμάτι του Data Management Layer, το οποίο περιγράφηκε στην προηγούμενη ενότητα. Η λειτουργία του είναι πλήρως αυτόνομη και ακολουθιακή.

3.3.2 Data Modeler Component

Το δομικό στοιχείο αυτό έχει ως στόχο την μοντελοποίηση των δεδομένων που παρέχονται από τη διαπροσωπεία provideData του Data Manager. Η μοντελοποίηση αυτή γίνεται με σκοπό τα δεδομένα να μεταφραστούν σε κατάλληλη μορφή, ώστε να μπορούν να αξιοποιηθούν για τον έλεγχο της συμμόρφωσης του συστήματος. Για τον σκοπό αυτό έχουν οριστεί τα εξής επιμέρους δομικά στοιχεία: Data Extractor, Data Interpreter, Interpreted Data Reposotory, Annotation Editor, Annotation Repository και Model Creator.

Το δομικό στοιχείο **Data Extractor** είναι ο συνδετικός κρίκος μεταξύ του Data Manager και του Data Modeler. Λαμβάνει τα δεδομένα που έχουν καταγραφεί μέσω της διαπροσωπείας provideData και τα προωθεί για επεξεργασία και μετάφραση μέσω της διαπροσωπείας **supplyData**.

Το δομικό στοιχείο **Annotation Editor** συνδέει το User Interface με τον Data Modeler μέσω της διαπροσωπείας **specifyAnnotation**. Η διαπροσωπεία αυτή επιτρέπει στον χρήστη να ορίσει τον τύπο των επισημειώσεων (π.χ. κατηγορήματα λογικής πρώτης τάξης) που θα χρησιμοποιηθούν από τον μηχανισμό ελέγχου της συμμόρφωσης του συστήματος. Επίσης, ορίζονται και οι ίδιες οι επισημειώσεις, οι οποίες αποθηκεύονται μέσω της διαπροσωπείας **insertAnnotation**.

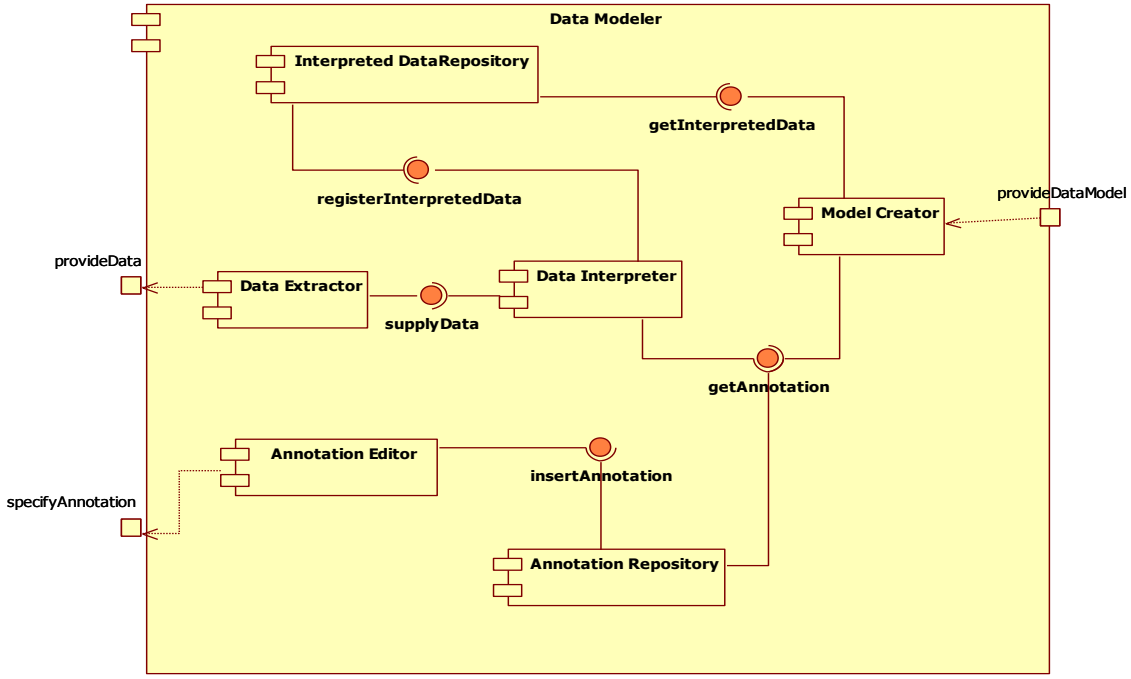
Το δομικό στοιχείο **Annotation Repository**, αντιπροσωπεύει την αποθηκευτική δομή στην οποία καταχωρούνται οι επισημειώσεις που ορίζονται από τον χρήστη.

Το δομικά στοιχείο **Data Interpreter** λαμβάνει τα καταγεγραμμένα δεδομένα και με βάση κάποια προκαθορισμένη λογική που έχει οριστεί, μεταφράζει τα δεδομένα αυτά αξιοποιώντας τις επισημειώσεις που έχουν οριστεί. Η πρόσβαση στο Annotation Repository για τη χρήση των επισημειώσεων γίνεται μέσω της διαπροσωπείας **getAnnotation**. Ένα παράδειγμα λειτουργίας του συγκεκριμένου δομικού στοιχείου μπορεί να είναι η μετατροπή των δεδομένων καταγραφής σε ground predicates, σε περίπτωση που ο τύπος επισημείωσης που έχει οριστεί είναι τα Μαρκοβιανά Λογικά Δίκτυα. Στη συνέχεια, τα μετεφρασμένα δεδομένα αποθηκεύονται μέσω της διαπροσωπείας **registerInterpretedData**.

Το δομικό στοιχείο Interpreted **Data Repository** αντιπροσωπεύει την αποθηκευτική δομή στην οποία καταχωρούνται τα δεδομένα που έχουν μεταφραστεί από τον Data Interpreter.

Το δομικό στοιχείο **Model Creator** αξιοποιώντας τα μεταφρασμένα δεδομένα και τις επισημειώσεις που έχουν καταχωρηθεί, παράγει το λειτουργικό μοντέλο δεδομένων που απαιτείται για τον έλεγχο της συμμόρφωσης των επιχειρησιακών διαδικασιών. Το μοντέλο αυτό μπορεί να είναι μια γνωσιακή βάση (knowledge base) και ένα σύνολο κατηγορημάτων. Τα δύο αυτά στοιχεία είναι απαραίτητα για την μοντελοποίηση των πολιτικών συμμόρφωσης, αλλά και την εκτέλεση μηχανισμών συμπερασμού για τον έλεγχό τους. Τέλος το δομικό

στοιχείο Model Creator είναι αυτό που παρέχει τη διαπροσωπεία **provideDataModel**, μέσω της οποίας γίνεται διαθέσιμο το μοντέλο δεδομένων.



Εικόνα 3.3.1: Ψηφιδικό διάγραμμα του Data Modeler

Η λειτουργία του δομικού στοιχείου Data Modeler αντιπροσωπεύει ένα μέρος του Modeling Layer. Το στοιχείο αυτό αξιοποιείται από χρήστες οι οποίες έχουν γνώση και για τις πολιτικές συμμόρφωσης, επιστημονικό αλλά και τεχνικό υπόβαθρο (π.χ. Μαρκοβιανά Λογικά Δίκτυα, OCL κ.ά.), ώστε να ορίζουν τις κατάλληλες επισημειώσεις που απαιτούνται για την παραγωγή του Μοντέλου Δεδομένων.

3.3.3 Business Logic Modeler Component

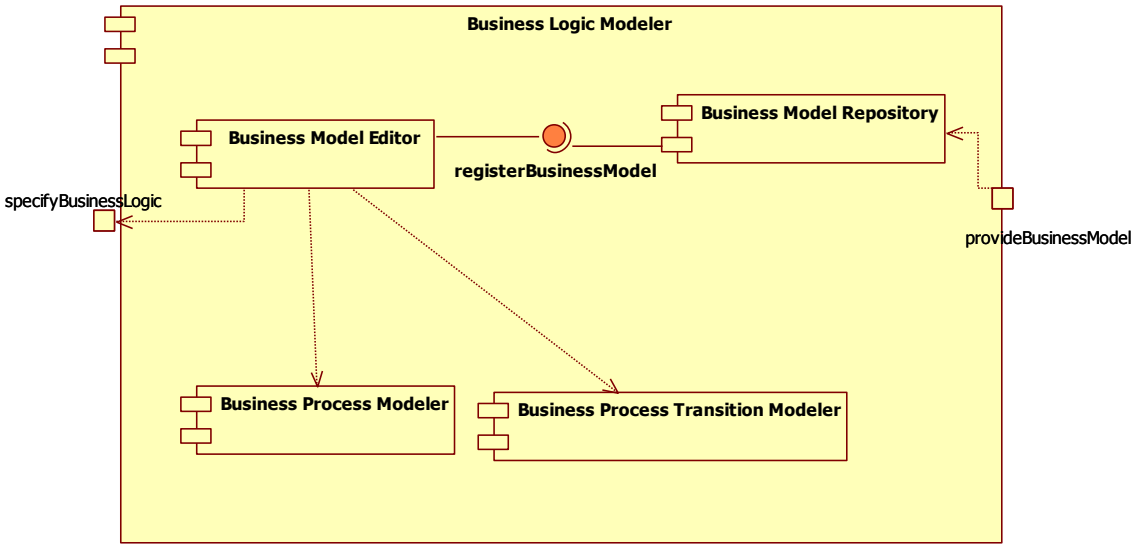
Το δομικό στοιχείο αυτό έχει ως ρόλο την μοντελοποίηση όλης της επιχειρησιακής λογικής που υλοποιείται. Το παραγόμενο λειτουργικό μοντέλο αντιπροσωπεύει όλα τα τμήματα της επιχειρησιακής διαδικασίας, όπως αυτά έχουν σχεδιαστεί με κάποια επιχειρησιακή γλώσσα (π.χ. BPMN ή BPEL). Για τη δημιουργία αυτού του μοντέλου ορίζονται τα εξής εσωτερικά δομικά στοιχεία: Business Model Editor, Business Process Modeler, Business Process Transition Editor και Business Model Repository.

Το δομικό στοιχείο **Business Model Editor** αποτελεί μια διασύνδεση με το χρήστη, ο οποίος μέσω της διαπροσωπείας **specifyBusinessLogic** αναπαριστά όλα το σύνολο των τμημάτων, τα οποία συνθέτουν την επιχειρησιακής διαδικασίας που θα ελεγχθεί ως προς τη συμμόρφωση. Το σύνολο αυτό μπορεί να απαρτίζεται από business process tasks, operations, services κ.ά. Όταν η σχεδίαση του επιχειρησιακού μοντέλου ολοκληρωθεί τότε αυτό αποθηκεύεται μέσω της διαπροσωπείας **registerBusinessModel**.

Το δομικό στοιχείο **Business Process Modeler** χρησιμοποιείται από το δομικό στοιχείο Business Model Editor, προκειμένου να ορίσει τα τμήματα της επιχειρησιακής διεργασίας που αναφέρθηκαν προηγουμένως.

Το δομικό στοιχείο **Business Process Transition Modeler** χρησιμοποιείται από το δομικό στοιχείο Business Model Editor, προκειμένου να ορίσει της μεταβάσεις ανάμεσα στα τμήματα της επιχειρησιακής διαδικασίας. Οι μεταβάσεις αυτές μπορεί να απεικονίζουν διαχωρισμούς, ενώσεις ή παράλληλες εκτελέσεις εργασιών.

Το δομικό στοιχείο **Business Process Repository**, αποτελεί την αποθηκευτική δομή του επιχειρησιακού μοντέλου, το οποίο γίνεται διαθέσιμο προς τα άλλα δομικά στοιχεία μέσω της διαπροσωπείας **provideBusinessModel**.



Εικόνα 3.3.2: Ψηφιδικό διάγραμμα του Business Logic Modeler

Η λειτουργία του δομικού στοιχείου Business Logic Modeler αποτελεί μέρος του Modeling Layer. Η χρήση του δομικού στοιχείου αυτού γίνεται από τους Σχεδιαστές κα Αναλυτές επιχειρησιακών διαδικασιών, οι οποίοι είναι διαθέτουν τις λεπτομέρειες για τη σύνθεση της επιχειρησιακής διαδικασίας που θα ελεγχθεί ως προς τη συμμόρφωση.

3.3.4 Compliance Modeler Component

Το δομικό στοιχείο αυτό έχει ως ρόλο την μοντελοποίηση των πολιτικών και των κανόνων που θα χρησιμοποιηθούν για τον έλεγχο της συμμόρφωσης. Η μοντελοποίηση γίνεται με τη χρήση των Μοντέλων Στόχων, τα οποία περιγράφηκαν αναλυτικά στην Ενότητα 2.6 . Τα επιμέρους δομικά στοιχεία του Compliance Modeler είναι τα εξής: Goal Model Editor, Decomposition Modeler, Goal Model Repository Compliance Rule Editor, Compliance Rule Repository.

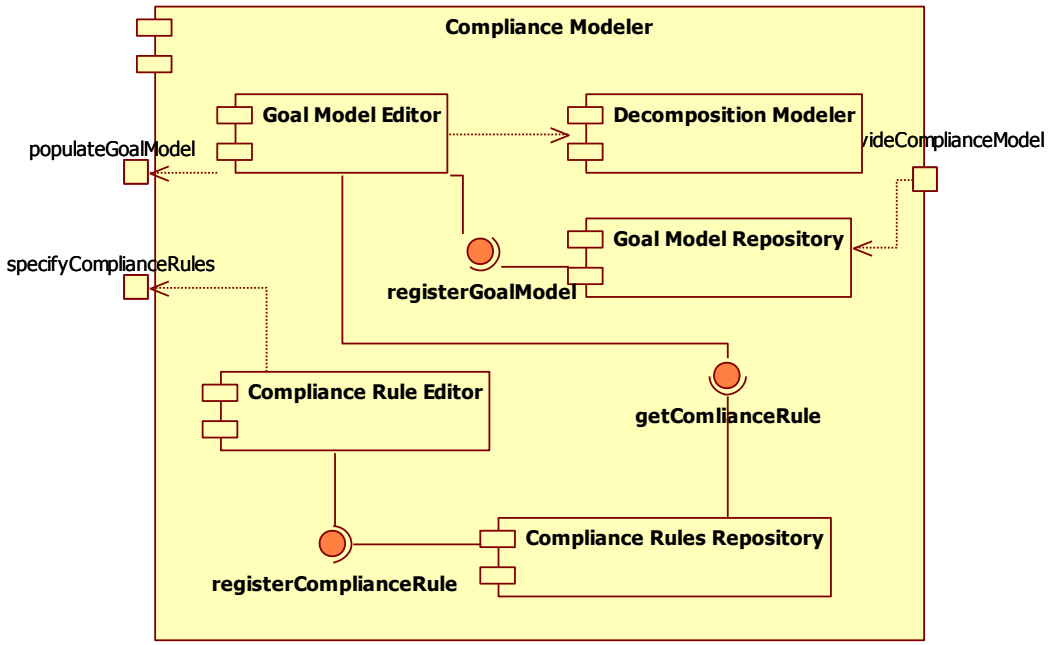
Το δομικό στοιχείο **Compliance Rule Editor** δίνει τη δυνατότητα στο χρήστη, μέσω της διαπροσωπείας **specifyComplianceRules**, να ορίσει κανόνες στους οποίους θα πρέπει να συμμορφώνεται η εκτέλεση της επιχειρησιακής διαδικασίας, που υλοποιεί το σύστημα. Επίσης, μέσω της διαπροσωπείας **registerComplianceRule** δίνεται η δυνατότητα αποθήκευσης κάθε κανόνα που ορίζεται.

Το δομικό στοιχείο **Compliance Rule Repository** αποτελεί την αποθηκευτική δομή των κανόνων που ορίζονται από τον Compliance Rule Editor.

Το δομικό στοιχείο **Goal Model Editor** δίνει στο χρήστη τη δυνατότητα ορισμού Μοντέλων Στόχων, των οποίων οι κόμβοι σχετίζονται με κάποιον κανόνα. Με αυτόν τον τρόπο ορίζονται οι πολιτικές στις οποίες θα πρέπει να συμμορφώνεται το σύστημα. Επίσης, μέσω της διαπροσωπείας **registerGoalModel** γίνεται η αποθήκευση του Μοντέλου Στόχων.

Το δομικό στοιχείο **Decomposition Modeler** χρησιμοποιείται για τον ορισμό των διαζεύξεων ή συζεύξεων που συνδέουν τους κόμβους του Μοντέλου Στόχων.

Το δομικό στοιχείο **Goal Model Repository** αναπαριστά την αποθηκευτική δομή των Μοντέλων Στόχων που δημιουργούνται στον Goal Model Editor.



Εικόνα 3.3.3: Ψηφιδικό διάγραμμα του Compliance Modeler

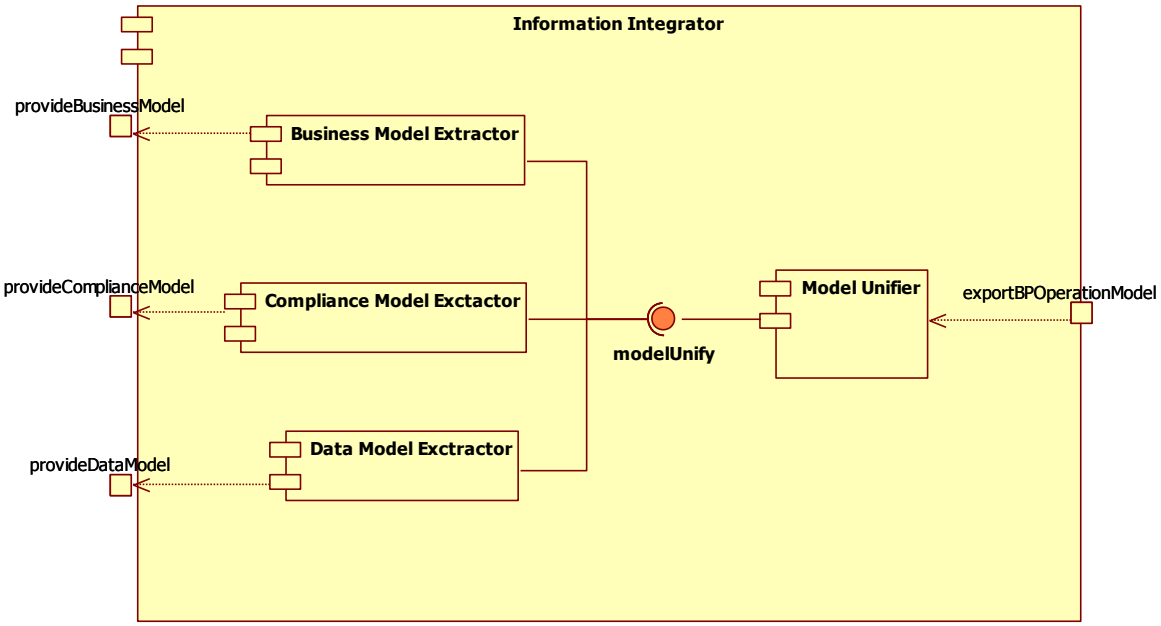
Η λειτουργία του Compliance Modeler ολοκληρώνει τις ενέργειες που συντελούνται στο Modeling Layer με την παραγωγή του μοντέλου των προς εξέταση πολιτικών. Το δομικό αυτό στοιχείο χρησιμοποιείται από εξειδικευμένους χρήστης, οι οποίοι γνωρίζουν όλες της πολιτικές και τους κανόνες, στους οποίους θα πρέπει να συμμορφώνεται η επιχειρησιακή διαδικασία που έχει υλοποιηθεί.

3.3.5 Information Integrator Component

Η λειτουργία του δομικού αυτού στοιχείου είναι σχετικά απλή, αλλά θεμελιώδης για τη λειτουργία του περιβάλλοντος πλαισίου. Πιο συγκεκριμένα, ενοποιεί τα μοντέλα που έχουν παραχθεί μέχρις στιγμής από τα υπόλοιπα δομικά στοιχεία, δημιουργώντας ένα σύνθετο λειτουργικό μοντέλο. Την διαπεραίωση αυτής της λειτουργίας αναλαμβάνουν τα εξής εσωτερικά δομικά στοιχεία: Business Model Extractor, Compliance Model Extractor, Data Model Extractor και Model Unifier.

Τα δομικά στοιχεία **Business Model Extractor**, **Compliance Model Extractor** και **Data Model Extractor** παραλαμβάνουν το Επιχειρησιακό Μοντέλο, το Μοντέλο Συμμόρφωσης και Μοντέλο Δεδομένων από τα ανάλογα εξωτερικά δομικά στοιχεία.

Το δομικό στοιχείο **Model Unifier** παρέχει τη διαπροσωπεία **modelUnify**, η οποία αποτελεί τον πυρήνα της λειτουργίας του Information Integrator. Η διαπροσωπεία αυτή περιέχει μεθόδους, οι οποίες συσχετίζουν τα τμήματα της επιχειρησιακής διαδικασίας που περιγράφονται από το Επιχειρησιακό Μοντέλο με τα Μοντέλα Στόχων του Μοντέλου Συμμόρφωσης. Στη συνέχεια σε κάθε κόμβο των Μοντέλων Στόχων επισυνάπτεται κάποια επισημείωση από το Μοντέλο Δεδομένων, η οποία επιλέγεται με κριτήρια που έχουν καθοριστεί κατά την υλοποίηση του περιβάλλοντος πλαισίου. Με την ολοκλήρωση όλων αυτών των διαδικασιών παράγεται ένα συνολικό λειτουργικό μοντέλο, το οποίο είναι διαθέσιμο μέσω της διαπροσωπείας **exportBPOperationModel**. Το λειτουργικό αυτό μοντέλο μπορεί να χρησιμοποιηθεί πλέον για τον έλεγχο της συμμόρφωσης της επιχειρησιακής διαδικασίας.



Εικόνα 3.3.4: Το Ψηφιδικό διάγραμμα του Information Integrator

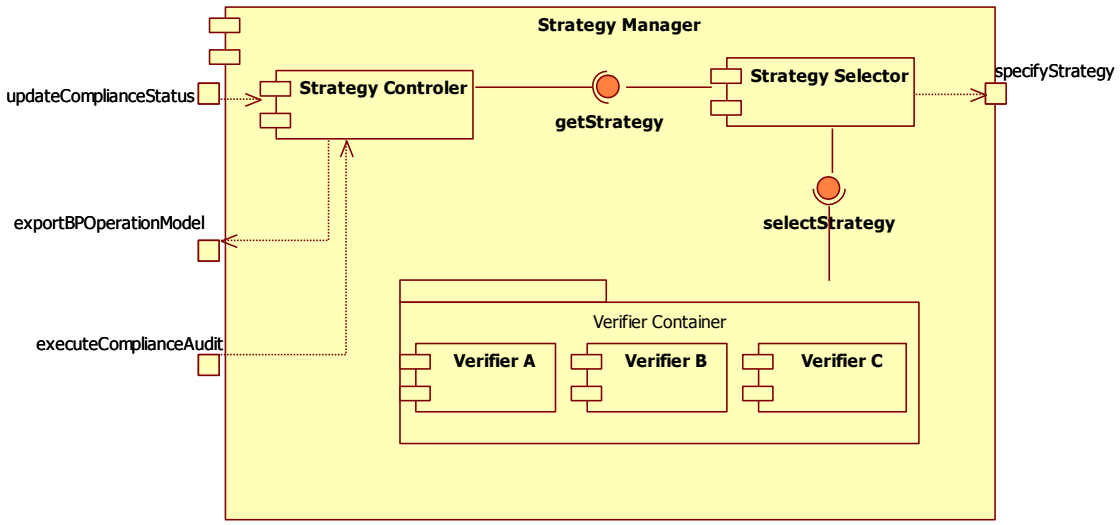
3.3.6 Strategy Manager Component

Το δομικό στοιχείο αυτό είναι υπεύθυνο για την εκτέλεση του ελέγχου συμμόρφωσης. Η λειτουργία του περιλαμβάνει την επιλογή της στρατηγικής ελέγχου της συμμόρφωσης και τον χειρισμό της εκτέλεσής του. Τα εσωτερικά δομικά στοιχεία που έχουν οριστεί για τον σκοπό αυτό είναι τα εξής: Strategy Controller, Strategy Selector και Verifier Container.

Το δομικό στοιχείο **Verifier Container** περιέχει ένα σύνολο από Verifiers. Ένας Verifier συμβολίζει μια στρατηγική, με την οποία μπορεί να ελεγχθεί η συμμόρφωση (π.χ. Μαρκοβιανά Λογικά Δίκτυα).

Το δομικό στοιχείο **Strategy Selector** έχει ως ρόλο την επιλογή κάποιας στρατηγικής, από αυτές που είναι διαθέσιμες μέσω του Verifier Container. Η επιλογή γίνεται μέσω της διαπροσωπείας **selectStrategy**.

Το δομικό στοιχείο **Strategy Controller** λαμβάνει τον Verifier που έχει επιλεχθεί μέσω της διαπροσωπείας **getStrategy**, καθώς και το λειτουργικό μοντέλο που παράγει ο Infromation Integrator. Στη συνέχεια, όταν του ζητηθεί μέσω της διαπροσωπείας **executeComplianceAudit**, πραγματοποιεί τον έλεγχο συμμόρφωσης σε όλες τις πολιτικές και τους κανόνες που έχουν οριστεί. Τα αποτελέσματα επιστρέφονται στον χρήστη μέσω της διαπροσωπείας **updateComplianceStatus**.



Εικόνα 3.3.5: Ψηφιδικό διάγραμμα του Strategy Manager

Η λειτουργία του Strategy Manager αποτελεί το τελικό στάδιο της εκτέλεσης του περιβάλλοντος – πλαισίου. Η χρήση αυτού του δομικού στοιχείου γίνεται από χρήστες, οι οποίοι έχουν πλήρη εποπτεία της επιχειρησιακής διαδικασίας που υλοποιείται, αλλά και των των πολιτικών στις οποίες πρέπει να συμμορφώνεται.

3.3.7 UI Component

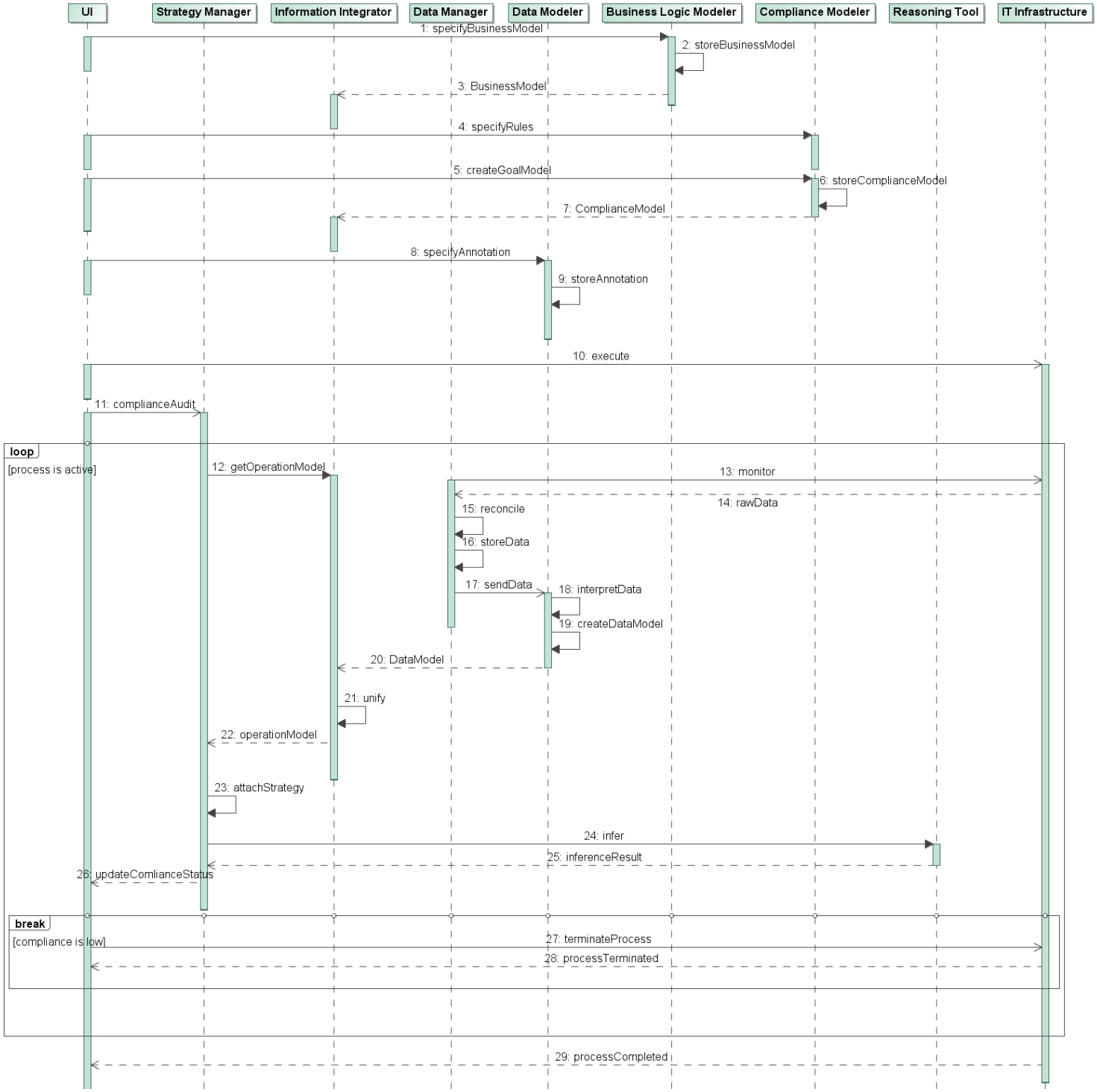
Το δομικό αυτό στοιχείο αποτελεί το σημείο διασύνδεσης του χρήστη με το περιβάλλον – πλαίσιο. Επιτρέπει στον χρήστη να παρεμβαίνει και να μεταβάλλει όλες τις δυναμικές παραμέτρους (π.χ. στρατηγικές ελέγχου συμμόρφωσης, τύπος επισημειώσεων κλπ). Επίσης,

του δίνει τη δυνατότητα να περιγράψει μοντέλα επιχειρησιακών διαδικασιών και πολιτικών συμμόρφωσης. Τέλος, δίνει τη δυνατότητα ελέγχου των πολιτικών συμμόρφωσης και την παρακολούθηση των αποτελεσμάτων που προκύπτουν.

Όλα τα παραπάνω επιτυγχάνονται μέσω των διαπροσωπειών που συνδέουν του δομικό στοιχείο UI με τα υπόλοιπα δομικά στοιχεία του περιβάλλοντος - πλαισίου. Οι διαπροσωπείες αυτές είναι οι εξής: specifyAnnotation, specifyBusinessLogic, specifyComplianceRules, populateGoalModel, specifyStrategy και executeComplianceAudit. Οι λειτουργίες των διαπροσωπειών αυτών έχουν ήδη περιγραφεί στια παραπάνω υποενότητες.

3.4 Λειτουργικότητα του Συστήματος

Στην ενότητα αυτή θα παρουσιάσουμε το ακολουθιακό διάγραμμα της λειτουργίας του περιβάλλοντος πλαισίου. Η παρουσίαση του ακολουθιακού διαγράμματος είναι απαραίτητη για την κατανόηση της αλληλεπίδρασης των δομικών στοιχείων που παρουσιάστηκαν στην ενότητα 3.3, αλλά και της σειράς εκτέλεσης των λειτουργιών που διαθέτουν.



Εικόνα 3.4.1: Ακολουθιακό Διάγραμμα του Περιβάλλοντος - Πλαισίου

1. **specifyBusinessModel**: Ο χρήστης μέσω του UI περιγράφει όλα τα τμήματα που συνθέτουν μια επιχειρησιακή διεργασία. Η περιγραφή αυτή γίνεται με άξονα την υλοποιημένη επιχειρησιακή διεργασία σε κάποια εκτελέσιμη γλώσσα, όπως η BPEL. Για την διαδικασία αυτή χρησιμοποιείται το δομικό στοιχείο Business Model Editor του Business Logic Modeler.
2. **storeBusinessModel**: Η επιχειρησιακή διαδικασία αποθηκεύεται στο Business Model Repository του Business Logic Modeler.
3. **BusinessModel**: Το ολοκληρωμένο μοντέλο της επιχειρησιακής διαδικασίας προωθείται στον Information Integrator, όπου και παραμένει προσωρινά στον Business Model Extractor.
4. **specifyRules**: Ο χρήστης μέσω του UI περιγράφει το σύνολο των κανόνων που πρέπει ακολουθούνται από τις επιχειρησιακές διαδικασίες που έχουν υλοποιηθεί. Η περιγραφή αυτή γίνεται με τη χρήση του δομικού στοιχείου Compliance Rule Editor του Compliance Modeler.
5. **createGoalModel**: Ο χρήστης μέσω του UI δημιουργεί ένα Μοντέλο Στόχων, κάνοντας χρήση του δομικού στοιχείου Compliance Modeler. Στο βήμα αυτό ορίζεται η δομή του δενδρικού γράφου, δηλαδή το μέγεθος και τύποι ζεύξης των κόμβων (AND ή OR), καθώς και οι κανόνες που αντιπροσωπεύει κάθε κόμβος.
6. **storeComplianceModel**: Το Μοντέλο Στόχων, ολοκληρωμένο και επισημειωμένο με κανόνες σε κάθε κόμβο αποθηκεύεται στο Goal Model Repository.
7. **ComplianceModel**: Το σύνολο των Μοντέλων Στόχων που αντιπροσωπεύει τις επιχειρησιακές πολιτικές με τις οποίες πρέπει να συμμορφώνεται η επιχειρησιακή διαδικασία που εκτελείται προωθείται στον Information Integrator, όπου αποθηκεύεται προσωρινά στο δομικό στοιχείο Compliance Model Extractor.
8. **specifyAnnotation**: Ο χρήστης μέσω του UI ορίζει ένα σύνολο επισημειώσεων (π.χ. κατηγορήματα λογικής πρώτης τάξης) κάνοντας χρήση του δομικού στοιχείου Annotation Editor του Data Modeler.
9. **storeAnnotation**: Οι επισημειώσεις που ορίζονται από τον χρήστη αποθηκεύονται στο Annotation Repository.
10. **execute:** Ο χρήστης και χειριστής του συστήματος που εκτελεί την επιχειρησιακή διαδικασία στέλνει εντολή εκκίνησης της εκτέλεσής της, μέσω του UI.
11. **complianceAudit**: Ο χρήστης που είναι υπεύθυνος για τον έλεγχο της συμμόρφωσης της επιχειρησιακής διαδικασίας δίνει εντολή εκκίνησης του ελέγχου στο δομικό στοιχείο Strategy Manager, μέσω του UI.

    loop:Τα βήματα που περικλείονται στο συγκεκριμένο τμήμα επαναλαμβάνονται, όσο η εκτέλεση της επιχειρησιακής διαδικασίας παραμένει ενεργή.

12. **getOperationModel**: Στο βήμα αυτό το δομικό στοιχείο Strategy Manager ζητά από το δομικό στοιχείο Information Integrator, το ολοκληρωμένο λειτουργικό μοντέλο της επιχειρησιακής διαδικασίας για να μπορέσει να προχωρήσει στον έλεγχο της συμμόρφωσης.
13. **monitor**: Η εκτέλεση του συστήματος παρακολουθείται και τα δεδομένα που προκείπτουν καταγράφονται.
14. **rawData**: Τα δεδομένα που προκύπτουν από την παρακολούθηση προωθούνται στον Data Manager.
15. **reconcile**: Τα δεδομένα καταγραφής που λαμβάνονται, υπόκεινται σε επεξεργασία, ώστε να έρθουν σε μια ενιαία και ομοιόμορφη δομή.
16. **storeData**:Τα επεξεργασμένα δεδομένα αποθηκεύονται στο δομικό στοιχείο Log DB.
17. **sendData**: Τα επεξεργασμένα δεδομένα προωθούνται στο δομικό στοιχείο Data Modeler.
18. **interpretData**: Τα καταγεγραμμένα δεδομένα επεξεργάζονται, περνώντας μια διαδικασία μετάφρασης, η οποία θα τα μετατρέψει σε μορφή κατάλληλη για χρήση από τις τεχνικές συμπερασμού που έχουν οριστεί (π.χ. βασικούς τύπους).
19. **createDataModel**: Χρησιμοποιώντας τα μεταφρασμένα δεδομένα και τις επισημειώσεις που έχουν οριστεί, ο Data Modeler δημιουργεί το πλήρες μοντέλο δεδομένων που απαιτείται για τον έλεγχο της συμμόρφωσης.
20. **DataModel**: Το μοντέλο δεδομένων προωθείται στο δομικό στοιχείο Information Integrator, όπου αποθηκεύεται προσωρινά στο δομικό στοιχείο Data Model Extractor.
21. **unify**: Το επιχειρησιακό μοντέλο συσχετίζεται με το μοντέλο συμμόρφωσης, αναθέτοντας σε κάθε συνθετικό στοιχείο της επιχειρησιακής διαδικασίας, ένα ή περισσότερα Μοντέλα Στόχων. Στη συνέχεια, οι κόμβοι των Μοντέλων Στόχων επισημειώνονται με κατηγορήματα που προκύπτουν από το μοντέλο δεδομένων.
22. **operationModel**: Το τελικό λειτουργικό μοντέλο προωθείται στο δομικό στοιχείο Strategy Manager, ώστε να χρησιμοποιηθεί για τον έλεγχο της συμμόρφωσης.
23. **attachStrategy**: Επιλέγεται η στρατηγική με την οποία θα γίνει ο έλεγχος της συμμόρφωσης.
24. **infer**: Εκτελείται η διαδικασία του συμπερασμού για τον έλεγχο της συμμόρφωσης μέσω κάποιου εργαλείου συμπερασμού (π.χ. Alchemy σε περίπτωση που η στρατηγική είναι τα Μαρκοβιανά Λογικά Δίκτυα).
25. **inferenceResult**: Το αποτέλεσμα του συμπερασμού επιστρέφεται στο δομικό στοιχείο Strategy Manager.
26. **updateComplianceStatus**: Το δομικό στοιχείο Strategy Manager ανανεώνει τα δεδομένα που προβάλλονται στον χρήστη για το επίπεδο της συμμόρφωσης, μέσω του UI.

break: Σε περίπτωση που ο χρήστης διαπιστώσει χαμηλά επίπεδα συμμόρφωσης εκτελούνται τα βήματα 27-28 και η ακολουθία των βημάτων τερματίζει εκεί.

27. **terminateProcess**: Ο χρήστης τερματίζει την επιχειρησιακή διαδικασία σε περίπτωση που διαπιστωθεί χαμηλό επίπεδο συμμόρφωσης.
28. **processTerminated**: Η διαδικασία τερματίστηκε ομαλά.
29. **processCompleted**: Η διαδικασία ολοκληρώθηκε ομαλά.

Κεφάλαιο 4: Εννοιολογικό Μοντέλο

4.1 Εισαγωγή

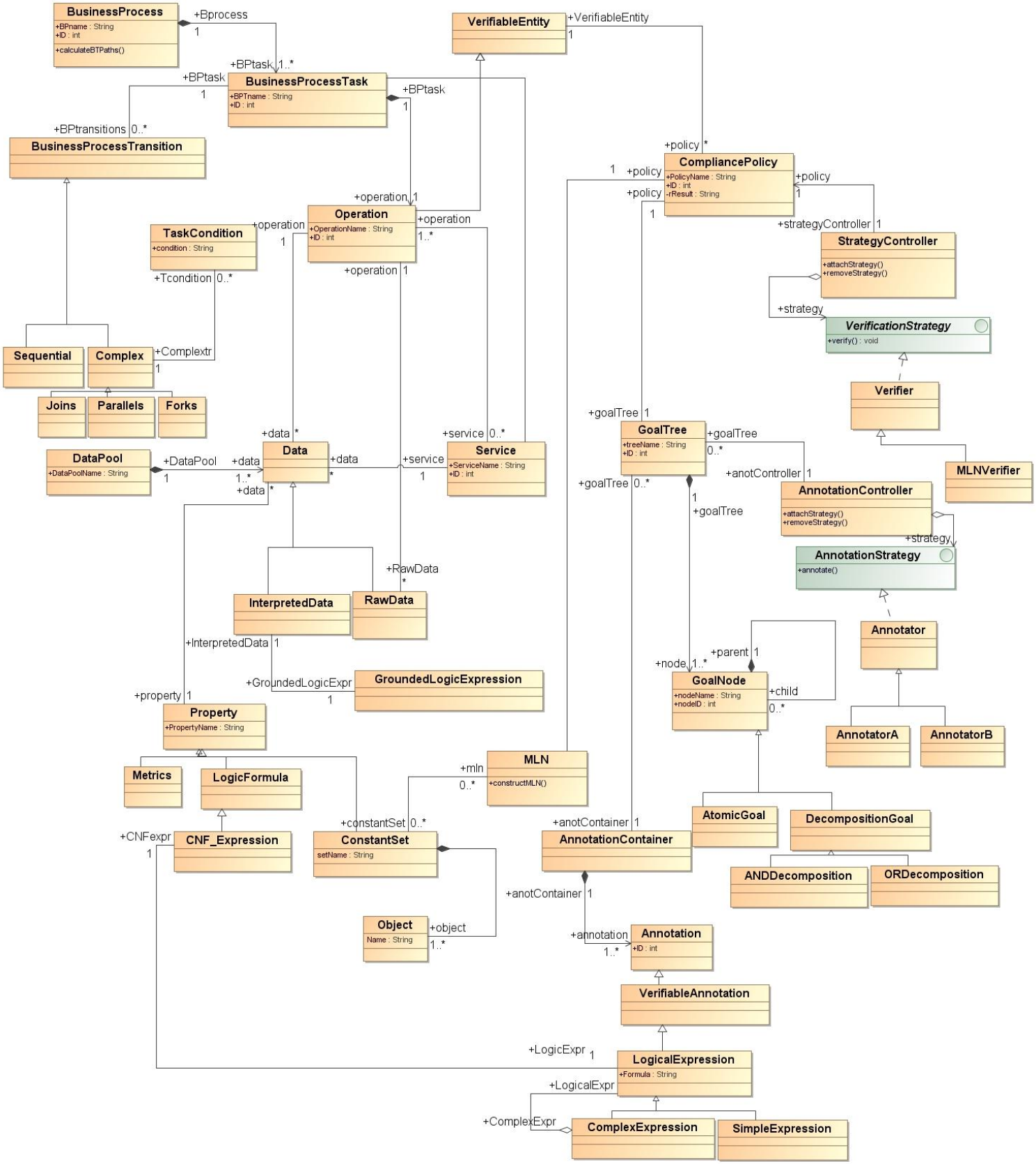
Στο κεφάλαιο αυτό θα περιγράψουμε το εννοιολογικό μοντέλο (domain model) του προτεινόμενου περιβάλλοντος – πλαισίου. Το μοντέλο αυτό περιγράφει τις οντότητες που συνθέτουν το συγκεκριμένο περιβάλλον – πλαίσιο και τις σχέσεις μεταξύ τους. Το μοντέλο που ακολουθεί, ακολουθεί την προδιαγραφή του MOF και αποτέλεσε τη βάση για την παραγωγή του κώδικα του περιβάλλοντος – πλαισίου μέσω του EMF.

4.2 Εννοιολογικό Μοντέλο Συμμόρφωσης

Το Εννοιολογικό Μοντέλο Συμμόρφωσης, το οποίο παρουσιάζεται στην Εικόνα , δίνει τη δυνατότητα στους συμμετέχοντες της διαχείρισης των επιχειρησιακών διαδικασιών που εκτελούνται σε ένα Υπηρεσιοκεντρικό σύστημα να περιγράψουν τα συνθετικά στοιχεία των διαδικασιών αυτών και να δημιουργήσουν διαχειρίσιμες οντότητες που τα αντιπροσωπεύουν. Επίσης, ανάλογες οντότητες μπορούν να δημιουργήσουν και οι υπεύθυνοι για τον έλεγχο της συμμόρφωσης των εκτελούμενων διαδικασιών, οι οποίες σχετίζονται με διάφορες πολιτικές. Στη συνέχεια ακολουθεί η περιγραφή των βασικών κλάσεων του μοντέλου ώστε να γίνει κατανοητή η λειτουργία τους.

4.2.1 Κλάση BusinessProcess

Η κλάση BusinessProcess αντιπροσωπεύει μια οντότητα επιχειρησιακής διαδικασίας. Η κλάση αυτή διαθέτει δύο ιδιότητες BPname και ID, οι οποίες αναπαριστούν το όνομα της επιχειρησιακής διαδικασίας που εκτελείται και τον διακριτικό της αριθμό. Τα στοιχεία αυτά ορίζονται με τέτοιο τρόπο ώστε να είναι σε απόλυτη συμφωνία, με αυτά της εκτελέσιμης διαδικασίας. Η οντότητα BusinessProcess αποτελείται από ένα ή περισσότερα BusinessProcessTask και γι αυτό συνδέεται με την αντίστοιχη κλάση μέσω μιας σχέσης σύνθεσης (composition) υπό την ιδιότητα +BPtask, με πολλαπλότητα 1..* .



Εικόνα 4.2.1: Διάγραμμα Κλάσεων Εννοιολογικού Μοντέλου Συμμόρφωσης

4.2.2 Κλάση BusinessProcessTask

Η κλάση BusinessProcessTask αντιπροσωπεύει μια οντότητα διεργασίας μιας επιχειρησιακής διαδικασίας. Η κλάση αυτή διαθέτει δύο ιδιότητες BPTname και ID, οι οποίες αναπαριστούν το όνομα της διεργασίας και το διακριτικό αριθμό της. Επίσης, συνδέεται μέσω μιας απλής συσχέτισης σύνθεσης με την κλάση Operation, υπό την ιδιότητα +operation, με πολλαπλότητα 1 και με την κλάση BusinessProcessTransition με απλή συσχέτιση, υπό την ιδιότητα +BPtransiotions με πολλαπλότητα 0..* .

4.2.3 Κλάση Business Process Transition

Η κλάση BusinessProcessTransition αντιπροσωπεύει τον τύπο τον μεταβάσεων από τη μια διεργασία στην επόμενη. Γι' αυτό το λόγο, αποτελεί γενίκευση για της κλάσης Sequential και Complex. Η κλάση Sequential αναπαριστά τη σειριακή μετάβαση και η κλάση Complex τη σύνθετη μετάβαση, η οποία όπως φαίνεται από τη δική της γενίκευση, μπορεί να είναι Join, Fork ή Parallel. Κάθε οντότητας σύνθετης μετάβασης συνδέεται μέσω απλής συσχέτισης με την κλάση TaskCondition, η οποία αναπαριστά τη συνθήκη της σύνθετης μετάβασης, εάν υπάρχει.

4.2.4 Κλάση Operation

Η κλάση Operation αναπαριστά τις λειτουργίες που εκτελούνται από τις διεργασίες της επιχειρησιακής διαδικασίας. Η κλάση αυτή διαθέτει δύο ιδιότητες OperationName και ID, οι οποίες αναπαριστούν το όνομα της λειτουργίας και το διακριτικό αριθμό της. Μια λειτουργία, μπορεί να είναι η αποστολή ενός μηνύματος, μια χρονική αναμονή κ.λ.π. Κάθε οντότητα Operation συνδέεται με την κλάση Service με μια απλή συσχέτιση, υπό την ιδιότητα +service με πολλαπλότητα 0..*, η οποία υποδηλώνει την υπηρεσία ή τις υπηρεσίες, στις οποίες απευθύνεται η λειτουργία. Επίσης, η κλάση Operation συνδέεται με την κλάση RawData μέσω μιας συσχέτισης, υπό την ιδιότητα +RawData πολλαπλότητας 1..*, η οποία δηλώνει τα παραγόμενα δεδομένα που προκύπτουν από την εκτέλεση της λειτουργίας.

4.2.5 Κλάση Service

Η κλάση Service αντιπροσωπεύει την οντότητα κάθε Υπηρεσίας του Υπηρεσιοκεντρικού συστήματος, στο οποίο εκτελούνται οι επιχειρησιακές διαδικασίες. Η κλάση αυτή διαθέτει δύο ιδιότητες ServiceName και ID, οι οποίες αναπαριστούν το όνομα της λειτουργίας και το διακριτικό αριθμό της. Κάθε Υπηρεσία συνδέεται μέσω απλής συσχέτισης με τουλάχιστο μία λειτουργία, υπό την ιδιότητα +operation πολλάπλότητας 1..* . Επίσης συνδέεται με απλή συσχέτιση, υπό την ιδιότητα +data πολλάπλότητας *, με την κλάση Data που θα αναλύσουμε παρακάτω.

4.2.6 Κλάση VerifiableEntity

Η κλάση VerifiableEntity αποτελεί μια γενίκευση των κλάσεων BusinessProcess, BusinessProcessTask, Operation και Service. Κάθε μία οντότητα των κλάσεων αυτών συνδέεται με πολιτικές, στις οποίες πρέπει να συμμορφώνεται η εκτελούμενη επιχειρησιακή διαδιακασία. Επομένως, η κλάση VerifiableEntity συνδέεται με την κλάση CompliancePolicy μέσω μιας απλής συσχέτισης, υπό την ιδιότητα +policy πολλαπλότητας *.

4.2.7 Κλάση Data

Η κλάση αυτή αντιπροσωπεύει το σύνολο των δεδομένων που παράγονται κατά τη λειτουργία του συστήματος. Τα δεδομένα αυτά μέσω μιας σχέσης γενίκευσης κατηγοριοποιούνται σε δύο κλάσεις, την RawData και την InterpretedData. Η πρώτη αντιπροσωπεύει τα ακατέργαστα δεδομένα που προκύπτουν από κάποια λειτουργία της επιχειρησιακής διαδικασίας και γι αυτό συνδέονται με την κλάση Operation με μια απλή συσχέτιση, υπό την ιδιότητα +operation πολλαπλότητας 1. Η κλάση InterpretedData αντιπροσωπεύει τα δεδομένα που προκύπτουν από την διαδικασία επεξεργασίας των ακατέργαστων δεδομένων και έχουν την μορφή βασικών κατηγορημάτων. Επομένως, η κλάση InterpretedData συνδέεται με τη κλάση GroundedLogivExpression με μία απλή συσχέτιση, υπό την ιδιότητα +GroundedLogicExpr πολλαπλότητας 1. Επίσης, ένα σύνολο οντοτήτων της κλάσης Data συνθέτουν μια οντότητα DataPool. Για το λόγο αυτό υπάρχει μια σχέση σύνθεσης ανάμεσα στις κλάσεις Data και DataPool. Τέλος, κάθε δεδομένο που προκύπτει από την εκτέλεση του συστήματος, σχετίζεται με μια κάποια ιδιότητα. Η σχέση αυτή απεικονίζεται με μια απλή συσχέτιση της κλάσης Data με την κλάση Property, υπό την ιδιότητα +property με πολλαπλότητα 1.

4.2.8 Κλάση GroundedLogicExpression

Η κλάση GroundedLogicExpression αναπαριστά τα βασικά κατηγορήματα, στα οποία μεταφράζονται τα δεδομένα που προκύπτουν από την εκτέλεση του συστήματος. Τα βασικά αυτά κατηγορήματα συνθέτουν την γνωσιακή βάση δεδομένων, η οποία αποτελεί την βάση για την εκτέλεση των αλγορίθμων συμπερασμού και την εξαγωγή αποτελεσμάτων για τη συμμόρφωση της επιχειρησιακής διαδικασίας με κάποια πολιτική. Η κλάση αυτή συνδέεται με μια απλή συσχέτιση μην την κλάση InterpretedData, υπό την ιδιότητα +interpetedData με πολλαπλότητα 1. Η συσχέτιση αυτή δηλώνει πως κάθε μεταφρασμένο δεδομένο συνδέεται με ένα βασικό κατηγόρημα.

4.2.9 Κλάση DataPool

Η κλάση DataPool αναπαριστά μία οντότητα όπου περιλαμβάνει κάθε δεδομένο που είτε προκύπτει απευθείας από την εκτέλεση του συστήματος, είτε ύστερα από επεξεργασία. Η κλάση αυτή έχει την ιδιότητα DataPoolName, ώστε κάθε οντότητα αυτής της κλάσης να διαφοροποιείται από τις υπόλοιπες. Επίσης, όπως αναφέραμε παραπάνω, κάθε οντότητα DataPool συντίθεται από ένα σύνολο οντοτήτων Data και γι αυτό υπάρχει μια σχέση σύνθεσης μεταξύ τους.

4.2.10 Κλάση Property

Η κλάση Property αποτελεί μια γενίκευση των ιδιοτήτων που σχετίζονται με τα δεδομένα της εκτέλεσης του συστήματος. Μια οντότητα Property μπορεί να προκύψει είτε από την κλάση Metrics, η οποία αντιπροσωπεύει μετρικές του συστήματος, είτε από την κλάση LogicFormula, η οποία αναπαριστά τύπους λογικής πρώτης τάξης, είτε από την κλάση ConstantSet, η οποία εκφράζει σύνολα αντικειμένων που συμμετέχουν σε κάποιο κόσμο κανόνων, όπως περιγράφηκε στην ενότητα 2.7 . Επίσης, κλάση Property συνδέεται με τα δεδομένα που προκύπτουν από την εκτέλεση του συστήματος, οπότε και με την κλάση Data, υπό την ιδιότητα +data με πολλαπλότητα *.

4.2.11 Κλάση CompliancePolicy

Η κλάση CompliancePolicy αντιπροσωπεύει κάθε πολιτική που προδιαγράφεται για μια επιχειρησιακή διεργασία και τα συνθετικά της μέρη. Η κλάση αυτή διαθέτει τρεις ιδιότητες PlicyName, ID και Result. Οι δύο πρώτες αναπαριστούν το όνομα της πολιτικής και τον διακριτικό της αριθμό αντίστοιχα. Η μεταβλητή Result δέχεται το αποτέλεσμα που προκύπτει ύστερα από τον έλεγχο της συγκεκριμένες πολιτικής. Το αποτέλεσμα αυτό είναι είτε "Compliant", είτε "Policy Violation". Επίσης η κλάση CompliancePolicy συνδέεται με απλή συσχέτιση με την κλάση VerfiableEntity, υπό την ιδιότητα +VerifiableEntity με πολλαπλότητα 1. Η συσχέτιση αυτή δείχνει πως κάθε πολιτική που προδιαγράφεται συνδέεται με κάποιο συνθετικό μέρος της επιχειρησιακής διαδικασίας. Κάθε πολιτική συνδέεται με ένα μοντέλο στόχων, γι αυτό κλάση CompliancePolicy συνδέεται με την κλάση GoalTree, υπό την ιδιότητα +goalTree με πολλαπλότητα 1. Όταν σε σαν στρατηγική ελέγχου της συμμόρφωσης επιλέγονται τα Μαρκοβιανά Λογικά Δίκτυα, τότε κάθε πολιτική σχετίζεται με ένα Μαρκοβιανό Λογικό Δίκτυο που περιγράφει τους κανόνες της με τον τρόπο που περιγράφηκε στην ενότητα 2.7. Έτσι, η κλάση CompliancePolicy συνδέεται με την κλάση MLN με μια απλή συσχέτιση, υπό την ιδιότητα +mln με πολλαπλότητα 1. Τέλος, η κλάση CompliancePolicy συνδέεται με απλή συσχέτιση με την κλάση StrategyController, υπό την ιδιότητα +strategyController με πολλαπλότητα 1. Η σχέση με την κλάση αυτή, η οποία περιγράφεται αναλυτικά παρακάτω, υποδηλώνει τη σχέση κάθε πολιτικής με μια οντότητα που είναι υπεύθυνη για την διεξαγωγή του ελέγχου συμμόρφωσης.

4.2.12 Κλάση GoalTree

Η κλάση GoalTree αντιπροσωπεύει μια οντότητα ενός Μοντέλου Στόχων, το οποίο χρησιμοποιείται για να αναπαραστήσει τις πολιτικές, στις οποίες θα πρέπει να συμμορφώνονται οι επιχειρησιακές διαδικασίες. Η κλάση αυτή διαθέτει δύο ιδιότητες treeName και ID, οι οποίες αναπαριστούν το όνομα του Μοντέλου Στόχου και τον διακριτικό της αριθμό αντίστοιχα. Όπως περιγράψαμε στην ενότητα 2.6, ένα Μοντέλο Στόχος είναι ένας γράφος οργανωμένος σε δενδρική δομή. Επίσης, στο κεφάλαιο 3 έχουμε αναφέρει ότι κάθε κόμβος του γράφου αυτού λαμβάνει κατάλληλες επισημειώσεις με σκοπό την εκτέλεση κάποιας στρατηγικής ελέγχου συμμόρφωσης που να σχετίζεται με τις επισημειώσεις αυτές.

Για τους παραπάνω λόγους, η κλάση GoalTree συνδέεται με σύνθεση με την κλάση GoalNode και με μια απλή συσχέτιση με την κλάση AnnotationConainer, υπό την ιδιότητα +annotContainer με πολλαπλότητα 1.

4.2.13 Κλάση GoalNode

Η κλάση GoalNode αντιπροσωπεύει τους κόμβους του δενδρικού γράφου του Μοντέλου Στόχου. Όπως σε κάθε δέντρο, έτσι και εδώ κάθε κόμβος, εκτός από την ρίζα, έχει ένα κόμβο πατέρα και όλοι οι κόμβοι, εκτός από τα φύλλα, έχουν τουλάχιστον ένα παιδί. Οι σχέσεις αυτές περιγράφονται από την αυτοαναφερόμενη σύνθεση, υπό τις ιδιότητες +parent με πολλαπλότητα 1 και +child με πολλαπλότητα 0..* . Επίσης, επειδή το Μοντέλο Στόχων παρέχει λογικές σχέσεις ανάμεσα στα παιδιά του ίδιου πατέρα, η κλάση GoalTree εξειδικεύεται στις κλάσεις AtomicGoal και DecompositionGoal.

4.2.14 Κλάση AtomicGoal

Η κλάση αυτή αποτελεί μια εξειδίκευση της κλάσης GoalNode, προκειμένου να αναπαραστήσει του κόμβους φύλλα του Μοντέλου Στόχων.

4.2.15 Κλάση DecompositionGoal

Η κλάση αυτή αποτελεί μια εξειδίκευση της κλάσης GoalNode, προκειμένου να αναπαραστήσει τους κόμβους πατέρες, δηλώνοντας τον τύπο της λογικής ζεύξης των παιδιών τους. Για την επίτευξη του στόχου αυτού η κλάση DecompositionGoal αποτελεί γενίκευση των κλάσεων ANDDecomposition και ORDecomposition. Η κλάση ANDDecomposition αναπαριστά έναν κόμβο πατέρα που συνδέει λογικά τα παιδιά του με σύζευξη (AND), ενώ η κλάση ORDecomposition έναν κόμβο πατέρα που συνδέει λογικά τα παιδιά του διάζευξη (OR).

4.2.16 Κλάση AnnotationContainer

Η κλάση AnnotationContainer αντιπροσωπεύει μια οντότητα, η οποία ουσιαστικά είναι η αποθηκευτική δομή των επισημειώσεων που ορίζονται από τον χρήστη του περιβάλλοντος πλαισίου. Επομένως, συνδέεται με μια σχέση σύνθεσης με την κλάση Annotation.

4.2.17 Κλάση Annotation

Τα στιγμιότυπα της κλάση Annotation συνθέτουν την οντότητα AnnotationContainer που περιγράψαμε προηγουμένως και αντιπροσωπεύουν όλες τις επισημειώσεις που έχουν οριστεί από τον χρήστη και συνδέονται στους κόμβους του Μοντέλου Στόχων. Μια επισημείωση μπορεί να έχει διάφορους τύπους, ωστόσο στα πλαίσια της παρούσας διπλωματικής θα εξετάσουμε μόνο επισημειώσεις που μπορούν να χρησιμοποιηθούν για την εξακρίβωση της συμμόρφωσης. Επομένως, η κλάση Annotation αποτελεί γενίκευση της κλάσης VerifiableAnnotation, η οποία αντιπροσωπεύει επισημειώσεις που χρησιμοποιούνται αποκλειστικά για την εξακρίβωση της συμμόρφωσης. Στη συνέχεια, παρατηρούμε ότι και η κλάση VerifiableAnnotation αποτελεί με τη σειρά της γενίκευση της κλάσης LogicalExpression, η οποία είναι ένας ακόμα πιο συγκεκριμένος τύπος επισημειώσεων. Τα στιγμιότυπα της κλάσης LogicalExpression είναι λογικές εκφράσεις, όπως τύποι λογικής πρώτης τάξης.

4.2.18 Κλάση StrategyController

Η κλάση StrategyController παίζει ιδιαίτερα σημαντικό ρόλο για τον έλεγχο της συμμόρφωσης. Η κλάση αυτή διαθέτει μεθόδους, οι οποίες προσθέτουν ή αφαιρούν κάποια στρατηγική για την εξακρίβωση της συμμόρφωσης. Κάθε στρατηγική που επιλέγεται μπορεί να εκτελέσει τη μέθοδο verify της διαπροσωπείας VerificationStrategy. Η μέθοδος αυτή εξάγει κάποιο αποτέλεσμα για τη συμμόρφωση κάθε πολιτικής, το οποίο και αξιολογεί. Η στρατηγική που ακολουθείται κάθε φορά έχει να κάνει με τον τρόπο με τον οποίο έχει υλοποιηθεί η μέθοδος verify από κάποια "Verifier" (βλέπε υποενότητα 3.3.6) κλάση, όπως η MLNVerifier. Οι κλάσεις StrategyController και MLNVerifier σε συνδυασμό με τη διαπροσωπεία VerificationStrategy, υλοποιούν ένα Strategy Design Pattern, το οποίο επιτρέπει την επιλογή ανάμεσα σε διάφορους αλγορίθμους για τον έλεγχο της συμμόρφωσης.

4.2.19 Κλάση AnnotationController

Η κλάση AnnotationController αναλαμβάνει τον ρόλο του ταιριάσματος κάθε κόμβου του Μοντέλου Στόχων με κάποια επισημείωση. Μπορεί με τις μεθόδους attachStrategy() και removeStrategy() να ενσωματώνει ή να αφαιρεί στρατηγικές επισημείωσης ενός Μοντέλου Στόχων. Η μέθοδος επισημείωσης είναι η μέθοδος annotate() που παρέχεται από τη διαπροσωπεία VerificationStrategy. Ο τρόπος λειτουργίας της μεθόδου αυτής έχει να κάνει

με το πώς θα υλοποιηθεί από κάποια κλάση "Annotator" όπως AnnotatorA ή AnnotatorB. Επομένως έχουμε ένα Strategy Design Pattern, το οποίο επιτρέπει την επιλογή ανάμεσα σε διάφορους αλγορίθμους για την επισημείωση του Μοντέλου Στόχων.

Κεφάλαιο 5: Αλγόριθμοι Εξακρίβωσης Συμμόρφωσης

5.1 Εισαγωγή

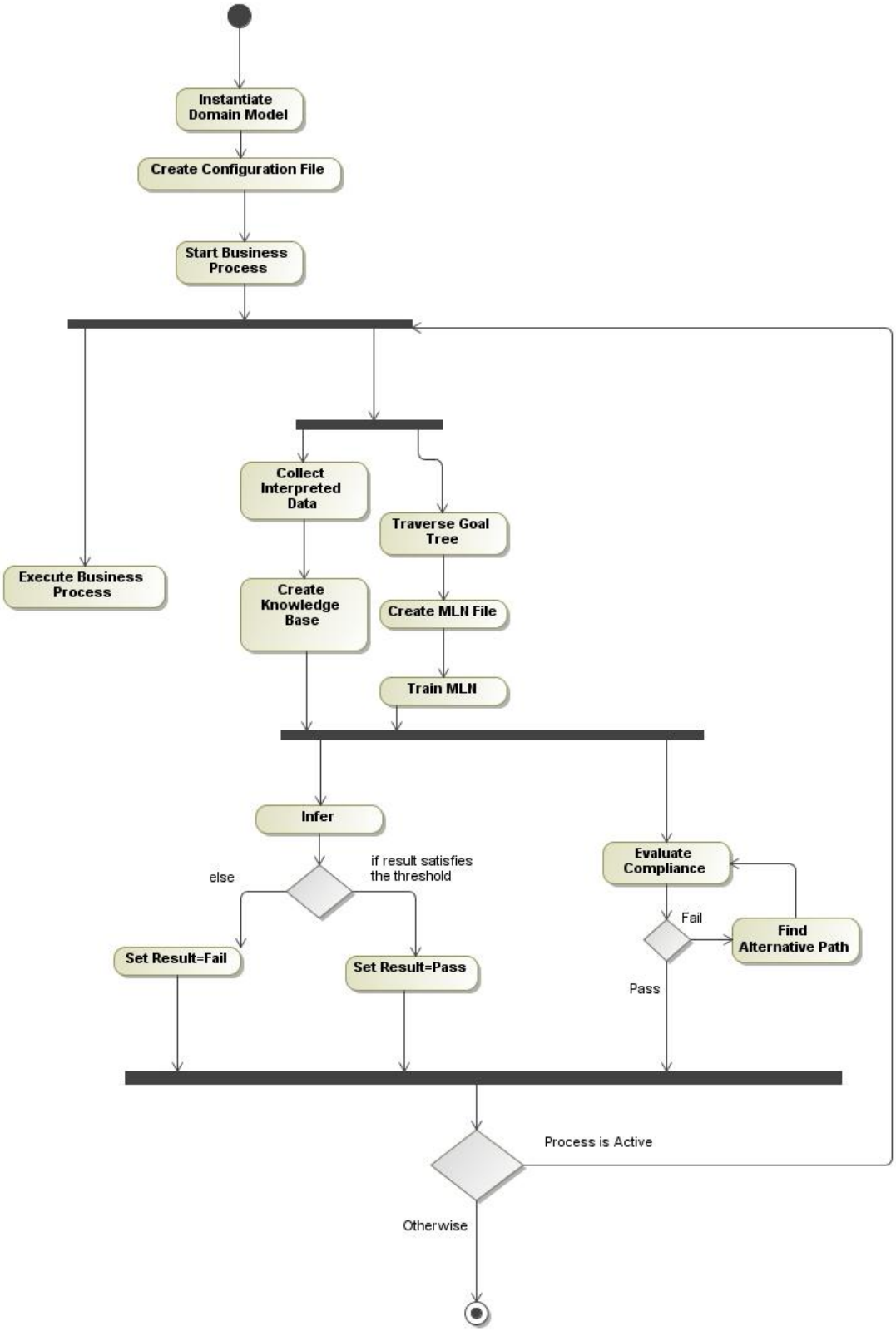
Στο κεφάλαιο αυτό θα περιγράψουμε τους βασικούς αλγορίθμους που εκτελούνται από το περιβάλλον πλαίσιο, το οποίο προτείνεται στην παρούσα διπλωματική εργασία. Οι αλγόριθμοι αυτοί αναπαριστούν την πρακτική λειτουργία του Εννοιολογικού Μοντέλου Συμμόρφωσης του προηγούμενου κεφαλαίου. Ο βασικός στόχος αυτού του κεφαλαίου είναι να γίνει κατανοητή η μεθοδολογία που προτείνεται για τον έλεγχο της συμμόρφωσης σε ένα Υπηρεσιοκεντρικό σύστημα.

5.2 Γενική Περιγραφή της Διαδικασίας

Πριν ξεκινήσει η εκτέλεση των αλγορίθμων που περιγράφονται στις δύο επόμενες ενότητες, ο χρήστης εισέρχεται στο περιβάλλον του Eclipse Modeling Frameworks και δημιουργεί αντικείμενα των κλάσεων που περιγράψαμε στο Εννοιολογικό Μοντέλο. Στη συνέχεια, για κάθε πολιτική συμμόρφωσης που έχει δημιουργήσει από το Εννοιολογικό Μοντέλο πρέπει να καταγράψει σε κάποιο configuration αρχείο τα όρια των τιμών που θα πρέπει να ικανοποιεί ο έλεγχος της συγκεκριμένης πολιτικής, ώστε να θεωρείται επιτυχής.

Όταν το Εννοιολογικό Μοντέλο έχει αρχικοποιηθεί και τα όρια επιτυχίας των πολιτικών έχουν οριστεί, ο χρήστης μπορεί να εκκινήσει την εκτέλεση της επιχειρησιακής διαδικασίας και παράλληλα την εκτέλεση της λειτουργίας ελέγχου του περιβάλλοντος – πλαισίου. Η διαδικασία ελέγχου υλοποιεί δύο αλγορίθμους τους οποίους θα περιγράψουμε στη συνέχεια με περισσότερη λεπτομέρεια. Ο πρώτος αλγόριθμος είναι ο Αλγόριθμος Εξακρίβωσης Συμμόρφωσης Πολιτικών. Ο ρόλος του συγκεκριμένου αλγορίθμου επικεντρώνεται στην δημιουργία των Μαρκοβιανών Λογικών Δικτύων για κάθε πολιτική συμμόρφωσης, την εκτέλεση του συμπερασμού για κάθε ένα από αυτά και της τιμής της πιθανότητας που προκύπτει σε κάθε πολιτική. Ο δεύτερος αλγόριθμος είναι ο Αλγόριθμος Εύρεσης Μονοπατιού Εκτέλεσης, οποίος συγκρίνει τις τιμές που αποδίδονται σε κάθε πολιτική συμμόρφωσης και τις συγκρίνει με τις επιτρεπτές τιμές που έχουν καταγραφεί στο configuration αρχείο. Σε περίπτωση που διαπιστωθεί κάποια παραβίαση των επιτρεπτών ορίων, ο αλγόριθμος επιχειρεί να βρει ένα εναλλακτικό μονοπάτι εκτέλεσης της διαδικασίας,

εφόσον αυτό είναι δυνατό. Παρακάτω παρουσιάζεται το UML διάγραμμα δραστηριοτήτων, το οποίο απεικονίζει τα βήματα των αλγορίθμων που αναλύονται παρακάτω.



Εικόνα 5.2.1: Διάγραμμα Δραστηριοτήτων του περιβάλλοντος – πλαισίου

5.3 Αλγόριθμος Εξακρίβωσης Συμμόρφωσης Πολιτικών

Ο Αλγόριθμος Εξακρίβωσης Συμμόρφωσης Πολιτικών αποτελεί τον άξονα της λειτουργίας του προτεινόμενου περιβάλλοντος πλαισίου. Ο αλγόριθμος αυτός δέχεται ως είσοδο κάποια οντότητα της επιχειρησιακής διαδικασίας, τα δεδομένα που συνδέονται με την οντότητα αυτή και ένα configuration αρχείο. Η οντότητα, η οποία δίνεται ως είσοδος είναι συνδεδεμένη με ορισμένες πολιτικές. Κάθε πολιτική με τη σειρά της είναι συνδεδεμένη με ένα επισημειωμένο Μοντέλο Στόχων. Στο σημείο αυτό, θα πρέπει να σημειώσουμε ότι ο αλγόριθμος αυτός λαμβάνει ως προϋπόθεση, ότι η στρατηγική ελέγχου συμμόρφωσης που ακολουθείται είναι αυτή των Μαρκοβιανών Λογικών Δικτύων. Επομένως, οι επισημειώσεις θεωρούμε πως είναι τύποι λογικής πρώτης τάξης. Στη συνέχεια, το δέντρο του Μοντέλου Στόχων διασχίζεται με τέτοιο τρόπο ώστε να καταγραφούν όλοι κανόνες του κόσμου του Μαρκοβιανού Λογικού Δικτύου. Αποτέλεσμα αυτής της διαδικασίας είναι η παραγωγή ενός ".mln" αρχείου το οποίο περιέχει τους κανόνες που αναφέραμε προηγουμένως. Στη συνέχεια, το αρχείο ".mln" εκχωρηθεί στο εργαλείο συμπερασμού Alchemy [**Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**] ώστε να εκπαιδευτεί το Μαρκοβιανό Λογικό Δίκτυο και οι κανόνες να αποκτήσουν κατάλληλα βάρη. Έπειτα, πάλι με τη χρήση του Alchemy θα εκτελεστεί μια διαδικασία συμπερασμού στο εκπαιδευμένο πλέον Μαρκοβιανό Λογικό Δίκτυο, η οποία θα εξάγει την πιθανότητα ικανοποίησης της εξεταζόμενης πολιτικής. Τέλος, η πιθανότητα αυτή εξετάζεται αν ικανοποιεί κάποια όρια (thresholds) τα οποία ορίζονται στο configuration αρχείο που δίνεται στην είσοδο. Παρακάτω παρουσιάζουμε σε μορφή ψευδοκώδικα τον αλγόριθμο που περιγράψαμε:

```
Input:  VerifiableEntity: Η οντότητα της επιχειρησιακής διαδικασίας την οποία θα εξετάσουμε
        ως προς τη συμμόρφωση.
        DataPool: Τα δεδομένα που έχουν προκύψει από την παρακολούθηση της εκτέλεσης του
        συστήματος.
        configurationFile: Το μονοπάτι του αρχείου όπου περιέχει τα όρια επιτυχίας των
        πολτικών.
Output: void
procedure verification(VerifiableEntity,DataPool,configurationFile):
        var goalTree,result

        for each policy of VerifiableEntity do
            goalTree = goalTree of VerifiableEntity
            mln =constructMLN(goalTree,DataPool)
            train(mln)
            result=infer(mln)
            if (verify(result,policy,configurationFile)==”Pass”) then
                set policy’s Result = ”Compliant”
             else
              set policy’s Result = ”Policy Violation”
```

```
Input:  goalTree: Το Μοντέλο Στόχων που συνδέεται με κάποια οντότητα.
        DataPool: Τα δεδομένα που έχουν προκύψει από την παρακολούθηση της εκτέλεσης του
συστήματος.
Output: mln: Το αρχείο που περιέχει το Μαρκοβιανό Λογικό Δίκτυο

procedure constructMLN(goalTree,DataPool)
        var mln,Q,source,logicform
        add all InterpretedData of VerifiableEntity into knowldge db
        add all ConstantSet of VerifiableEntity into mln

        for each node of the tree do
            add node’s formula into mln
        create queue Q
        enqueue source into Q
        while Q is not empty
```

```
                dequeue a node
                if node is ANDDecomposition then
                    add the conjuction of childrens' formulas implys father's formula into mln
                else if node is ORDecomposition then
                    add the dijuction of childrens' formulas implys father's formula into mln
                for all children of node do
                    enqueue child into Q
return mln
```

```
Input:  result: Το αποτέλεσμα του συμπερασμού για κάποια πολιτική.
        policy: Η πολιτική πάνω στην οποία γίνεται ο συμπερασμός.
        configurationFile: Το μονοπάτι του αρχείου όπου περιέχει τα όρια επιτυχίας των πολτικών.
Output: Pass ή Fail ανάλογα με το αν η πολιτική ικανοποιεί τα όρια του configurationFile.

procedure verify(result,policy,configurationFile)
        parse configurationFile
        get maxthreshold for policy
        get minthreshold for policy
        if result < maxthreshold and result>minthreshold then
                return "Pass"
        else
                return "Fail"
```

5.4 Αλγόριθμος Εύρεσης Μονοπατιού Εκτέλεσης

Ο Αλγόριθμος Εύρεσης Μονοπατιού Εκτέλεσης είναι αυτός που χρησιμοποιεί τα αποτελέσματα του Αλγορίθμου Εξακρίβωσης Συμμόρφωσης Πολιτικών προκειμένου να βρεθεί, εάν υπάρχει κάποιο μονοπάτι εκτέλεσης της επιχειρησιακής διαδικασίας, ώστε να μην παραβιάζεται καμία πολιτική. Η λογική του αλγορίθμου είναι απλή, εάν όσο εκτελείται

κάποια διεργασία της επιχειρησιακής διαδικασίας παρατηρηθεί κάποια παραβίαση πολιτικής, τότε αναζητείται εναλλακτική διεργασία, ώστε να συνεχιστεί η εκτέλεση της διαδικασίας. Σε περίπτωση που δε βρεθεί εναλλακτικό μονοπάτι, ακυρώνονται όλες οι ενέργειες που έχουν πραγματοποιηθεί (rollback) και επιστρέφεται κενό μονοπάτι. Ακολουθεί ο ψευδοκώδικας του αλγορίθμου που περιγράψαμε:

```
Input:  businessProcess: η εξεταζόμενη επιχειρησιακή διαδικασία.
        sourceTask: Η αρχική διεργασία που εκτελείται στην επιχειρησιακή διαδικασία.
Output: path: Το μονοπάτι εκτέλεσης της διαδικασίας.
procedure findExecutionPath(businessProcess,sourceTask)
        var path
        path=∅
        while(businessProcess is active)
            if  ∃policy ∈ businessProcess and policy fails then
                rollback businessProcess
                return ∄ path
            else
                for each active task ∈ businessProcess do
                    if ( (∃policy ∈ task) or(∃policy ∈ operation and operation ∈ task) or
                       (∃policy ∈ service and task interacts with service)) and (policy fails) then
                        rollback task
                        if ∃ alternative task then
                            execute alternative task
                        else
                            return ∄ path
                    else
                            add task into path
return path
```

Κεφάλαιο 6: Περιπτώσεις Χρήσης

6.1 Εισαγωγή

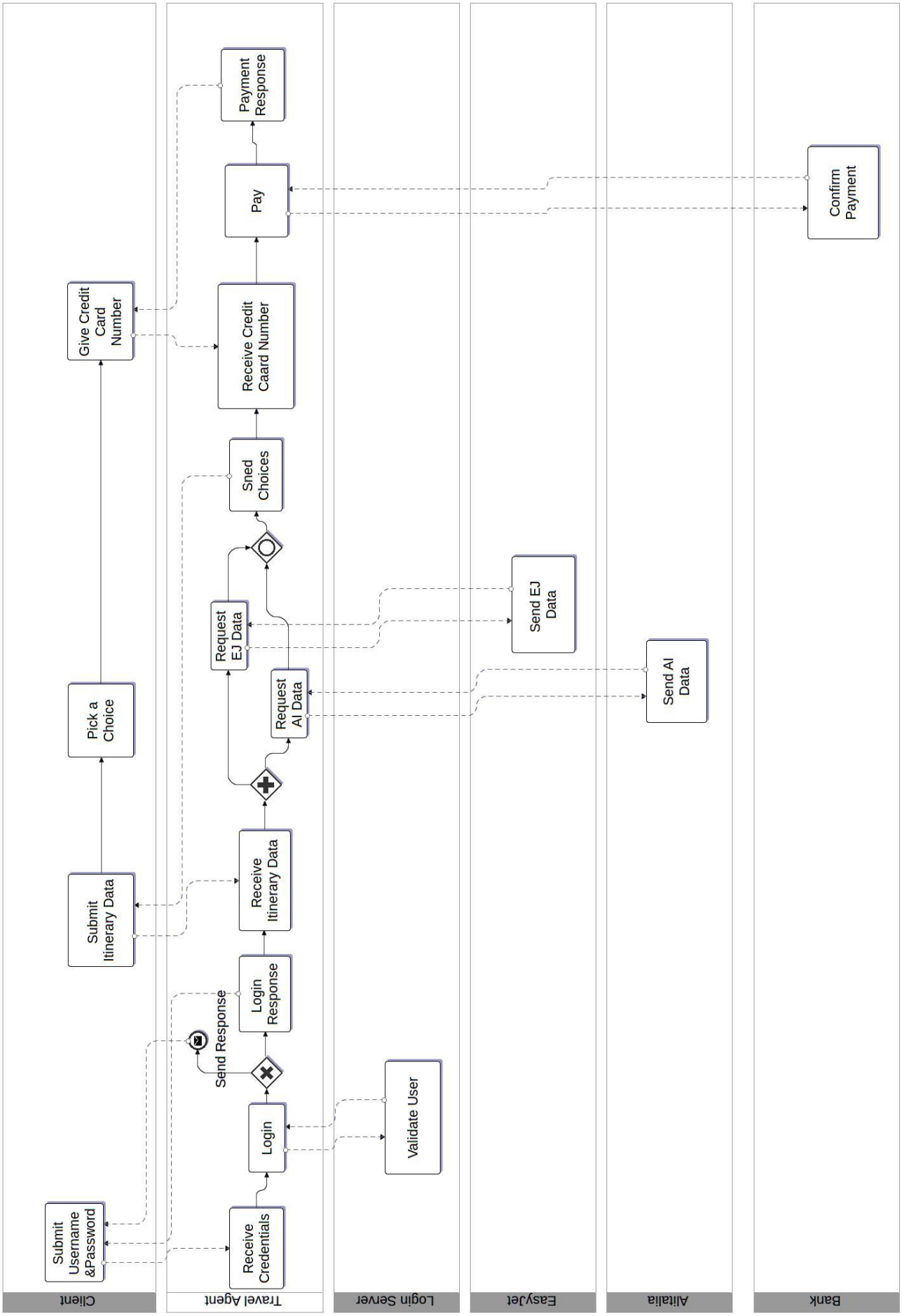
Στο κεφάλαιο αυτό θα παρουσιάσουμε την εκτέλεση της μεθοδολογίας ελέγχου συμμόρφωσης που προτείνουμε στην παρούσα διπλωματική εργασία. Για το λόγο αυτό έχουν σχεδιαστεί δύο επιχειρησιακές διαδικασίες υλοποιημένες με βάση την Υπηρεσιοκεντρική αρχιτεκτονική. Για τις διαδικασίες αυτές έχουμε ορίσει κάποιες πολιτικές, οι οποίες ελέγχονται με τη χρήση του εργαλείου συμπερασμού Alchemy. Οι επιχειρησιακές διαδικασίες έχουν σχεδιαστεί και υλοποιηθεί στο εργαλείο Intalio Designer, το οποίο προσφέρει τη δυνατότητα σχεδίασης επιχειρησιακών διαδικασιών σε BPMN και τη δυνατότητα μετασχηματισμού τους σε BPEL.

6.2 Περίπτωση Χρήσης 1

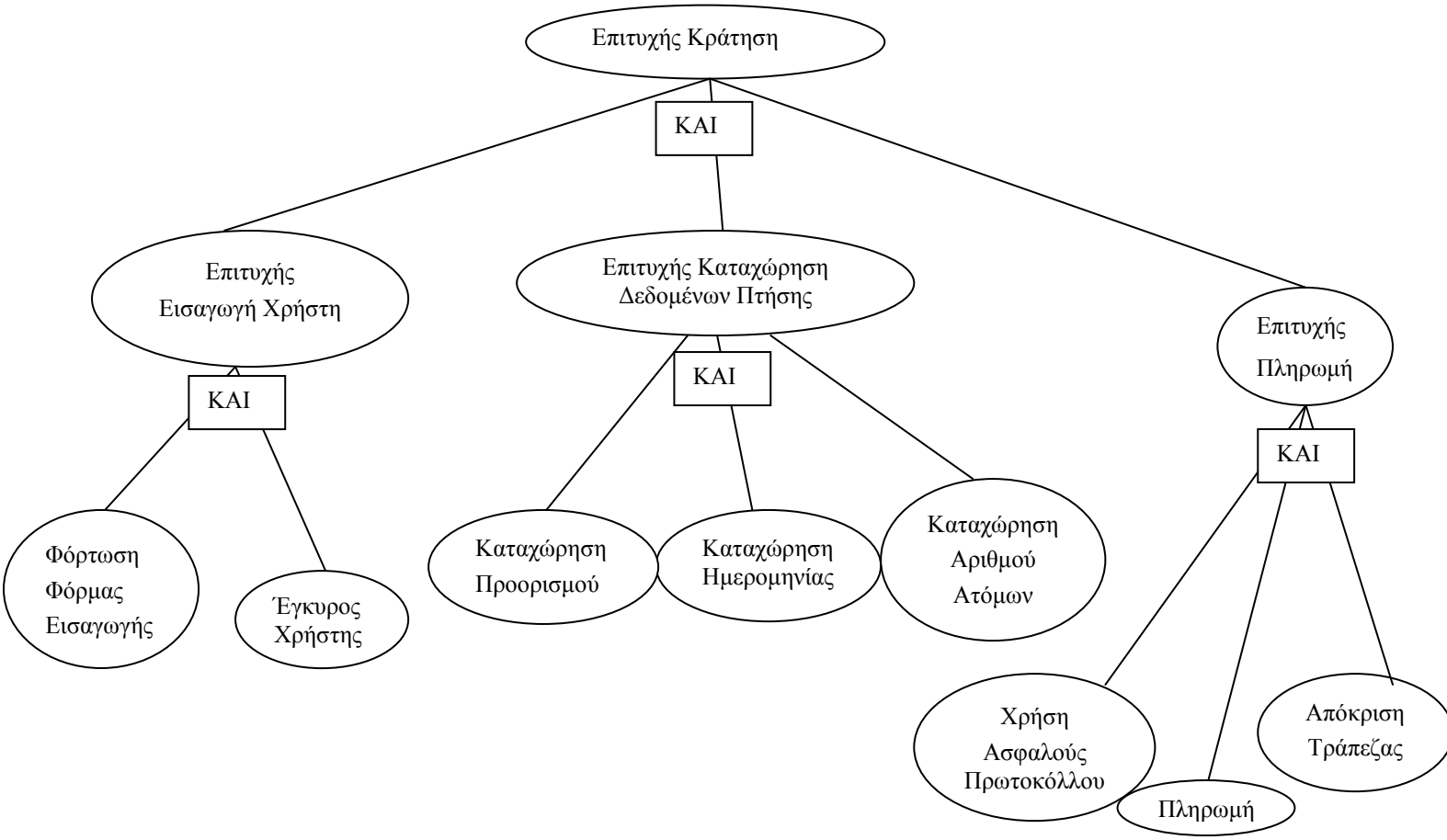
Η πρώτη επιχειρησιακή διαδικασία αφορά την κράτηση ενός αεροπορικού εισιτηρίου. Στην επιχειρησιακή διαδικασία αυτή υπάρχουν έξι συμμετέχοντες, ο χρήστης (Client), ο ταξιδιωτικός πράκτορας (Travel Agent), ο οποίος παίζει το ρόλο του ενορχηστρωτή, υπηρεσία του διακομιστή πιστοποίησης χρηστών (Login Server), οι υπηρεσίες δύο αεροπορικών εταιριών (EasyJet και AlItalia), η κάθε μία από τις οποίες παρέχει πληροφορίες για τα δρομολόγια της αντίστοιχης εταιρίας και η υπηρεσία της τράπεζας (Bank), οποία πραγματοποιεί την χρέωση του χρήστη κατά την αγορά του εισιτηρίου. Αρχικά ο χρήστης καταχωρεί τα στοιχεία του προκειμένου να εισέλθει στο σύστημα κρατήσεων(Submit Username & Passsword). Τα στοιχεία αυτά τα λαμβάνει ο ταξιδιωτικός πράκτορας (Receive Credentials) και τα στέλνει στον διακομιστή πιστοποίησης χρηστών (Login), ο οποίος με τη σειρά του θα προσπαθήσει να ταυτοποιήσει τον χρήστη (Validate User) και θα απαντήσει στον ταξιδιωτικό πράκτορα το αποτέλεσμα της ενέργειάς του. Έπειτα ο ταξιδιωτικός πράκτορας, σε περίπτωση που ο χρήστης δεν είναι έγκυρος, του στέλνει ένα μήνυμα αποτυχίας (Send Failure Response) και τερματίζει την επιχειρησιακή διαδικασία. Εάν ο χρήστης είναι έγκυρος, τότε εισάγει τα στοιχεία του ταξιδιού του, όπως προορισμός, ημερομηνία και αριθμός ατόμων (Submit Travel Data), τα οποία και παραλαμβάνει ο ταξιδιωτικός πράκτορας (Receive Travel Data) και στη συνέχεια τα προωθεί στις υπηρεσίες των δύο αεροπορικών εταιριών (Request EJ Data και Request AI Data), οι οποίες με τη σειρά τους για τα δεδομένα στοιχεία αποστέλλουν όλα τα πιθανά δρομολόγια (Send EJ Data και

Send AI Data). Στη συνέχεια, το ταξιδιωτικός πράκτορας στέλνει όλες τις δυνατές επιλογές (Send Choices) στον χρήστη, ο οποίος επιλέγει κάποια από αυτές (Pick a Choice). Στο τελευταίο βήμα αυτής της διαδικασίας, αφού ο χρήστης επιλέξει το δρομολόγιό του, προχωράει στην πληρωμή του εισιτηρίου εισάγοντας τα στοιχεία της κάρτας του (Give Credit Card Number), τα οποία και λαμβάνει ο ταξιδιωτικός πράκτορας (Receive Credit Card Number). Έπειτα, ο ταξιδιωτικός πράκτορας προχωρά στην πληρωμή του ζητά από την υπηρεσία της τράπεζας την χρέωση του πελάτη (Pay) και η τράπεζα προσπαθεί να επαληθεύσει και να χρεώσει τον χρήστη για την τιμή του εισιτηρίου (Confirm Payment), απαντώντας στον ταξιδιωτικό πράκτορα με ανάλογο μήνυμα, το οποίο στη συνέχει προωθείται στον χρήστη (Payment Response). Το διάγραμμα της επιχειρησιακής διαδικασίας, όπως αυτή υλοποιήθηκε στο περιβάλλον ανάπτυξης Intalio Designer παρουσιάζεται στην Εικόνα 6.2.1 .

Για την παραπάνω επιχειρησιακή διαδικασία θα εξετάσουμε μια πολιτική συμμόρφωσης, η οποία συνδέεται με την επιχειρησιακή διαδικασία στο σύνολό της. Η πολιτική αφορά τις προδιαγραφές που πρέπει να πλήρη κατά την εκτέλεσή της η διαδικασία προκειμένου η κράτηση να θεωρείται επιτυχής. Το Μοντέλο Στόχων της πολιτικής παρουσιάζεται στη Εικόνα 6.2.2 . Ξεκινώντας από την ρίζα κάθε στόχος, ικανοποιείται αν ικανοποιούνται όλοι οι υποστόχοι του. Βασικός σκοπός της πολιτικής αυτής είναι να εξεταστεί, εάν η συνεργασία των υπηρεσιών που απαρτίζουν το σύστημα γίνεται με σωστό τρόπο και δεν παρακάμπτεται κάποιο στάδιο της επιχειρησιακής διαδικασίας. Ο χαρακτήρας δηλαδή της πολιτικής είναι καθαρά τεχνικός.



Εικόνα 6.2.1: Επιχειρησιακή Διαδικασία Κράτησης Εισιτηρίου



Εικόνα 6.2.2: Μοντέλο Στόχων Περίπτωσης Χρήσης 1

Όπως έχει ήδη αναφερθεί κάθε στόχος του ενός μοντέλου στόχων επισημειώνεται με κάποιο κατηγόρημα. Η επισημείωση αυτή γίνεται με την εκτέλεση της μεθόδου annotate() της κλάσης Annotator που παρουσιάστηκε στο Κεφάλαιο 4 κατά την λειτουργία του περιβάλλοντος πλαισίου. Παρακάτω παρουσιάζεται η ένα προς ένα αντιστοίχιση των στόχων με τα κατηγορήματα που θα χρησιμοποιηθούν για τον έλεγχο της πολιτικής:

Επιτυχής Κράτηση → `Reservation(x)`
Επιτυχής Εισαγωγή Χρήστη → `Login(x)`
Επιτυχής Καταχώρηση Δεδομένων Πτήσης → `SelectDestination(x)`
Επιτυχής Πληρωμή → `Confirm(x)`
Φόρτωση Φόρμας Εισαγωγής → `LoadForm(x)`
Έγκυρος Χρήστης → `ValidUser(x)`

Καταχώρηση Προορισμού → `SubmitDestination(x)`

Καταχώρηση Ημερομηνίας → `SubmitDate(x)`

Καταχώρηση Αριθμού Ατόμων → `SubmitPersons(x)`

Χρήση Ασφαλούς Πρωτοκόλλου → `SecureProtocol(x)`

Πληρωμή → `Pay(x)`

Απόκριση Τράπεζας → `BankReply(x)`

Το όρισμα x των παραπάνω κατηγορημάτων αναπαριστά το αναγνωριστικό της επιχειρησιακής διαδικασίας.

Μετά την επισημείωση του μοντέλου στόχων με τα κατηγορήματα που παρουσιάστηκαν παραπάνω το περιβάλλον – πλαίσιο εκτελώντας τη μέθοδο constructMLN() της κλάσης MLN που περιγράφηκε στο Κεφάλαιο 4 παράγει το εξής Μαρκοβιανό Λογικό Δίκτυο:

```
Reservation(bp)
Login(bp)
SelectDestination(bp)
Confirm(bp)

LoadForm(bp)
ValidUser(bp)

SubmitDestination(bp)
SubmitDate(bp)
SubmitPersons(bp)

Pay(bp)
SecureProtocol(bp)
BankReply(bp)


Login(x)^SelectDestination(x)^Confirm(x) => Reservation(x).
LoadForm(x) ^ ValidUser(x) => Login(x).
SubmitDestination(x) ^ SubmitDate(x) ^ SubmitPersons(x) => SelectDestination(x).
Pay(x) ^ SecureProtocol(x) ^ BankReply(x) => Confirm(x).
```

Στη συνέχεια προκειμένου να αποδώσουμε βάρη στο παραπάνω Μαρκοβιανό Λογικό Δίκτυο χρησιμοποιούμε την παρακάτω γνωσιακή βάση εκπαίδευσης, η οποία προκύπτει από προηγούμενες εκτελέσεις του συστήματος.

```
LoadForm(BP1)
ValidUser(BP1)
Login(BP1)

SubmitDestination(BP1)
SubmitDate(BP1)
SubmitPersons(BP1)
SelectDestination(BP1)

Pay(BP1)
!Confirm(BP1)
!Reservation(BP1)


LoadForm(BP2)
!Login(BP2)
```

Η διαδικασία της εκπαίδευσης γίνεται μέσω του εργαλείου συμπερασμού Alchemy.

Το εκπαιδευμένο Μαρκοβιανό Λογικό Δίκτυο που προκύπτει από το Alchemy ύστερα από την εκτέλεση generative learning είναι το εξής:

```
//predicate declarations
LoadForm(bp)
BankReply(bp)
ValidUser(bp)
Reservation(bp)
SubmitDestination(bp)
Pay(bp)
Login(bp)
SubmitPersons(bp)
SubmitDate(bp)
SecureProtocol(bp)
Confirm(bp)
SelectDestination(bp)
// Login(x) ^ SelectDestination(x) ^ Confirm(x) => Reservation(x).
Reservation(a1) v !Login(a1) v !SelectDestination(a1) v !Confirm(a1).
// LoadForm(x) ^ ValidUser(x) => Login(x).
Login(a1) v !LoadForm(a1) v !ValidUser(a1).
// SubmitDestination(x) ^ SubmitDate(x) ^ SubmitPersons(x) => SelectDestination(x).
SelectDestination(a1) v !SubmitDestination(a1) v !SubmitDate(a1) v
!SubmitPersons(a1).
// Pay(x) ^ SecureProtocol(x) ^ BankReply(x) => Confirm(x).
Confirm(a1) v !Pay(a1) v !SecureProtocol(a1) v !BankReply(a1).

// -7.23102  Reservation(a1)
-7.23102  Reservation(a1)
// -6.3492  Login(a1)
-6.3492  Login(a1)
// -6.24508  SelectDestination(a1)
-6.24508  SelectDestination(a1)
// -6.61505  Confirm(a1)
-6.61505  Confirm(a1)
// 7.23102  LoadForm(a1)
7.23102  LoadForm(a1)

// 6.3492  ValidUser(a1)
6.3492  ValidUser(a1)

// 0       SubmitDestination(a1)
0       SubmitDestination(a1)
// 0       SubmitDate(a1)
0       SubmitDate(a1)
// 0       SubmitPersons(a1)
0       SubmitPersons(a1)
// 0       Pay(a1)
0       Pay(a1)
// -7.23102  SecureProtocol(a1)
-7.23102  SecureProtocol(a1)
// -7.23102  BankReply(a1)
-7.23102  BankReply(a1)
```

Το μόνο που μένει τώρα είναι εκτέλεση της επιχειρησιακής διαδικασίας και η συλλογή των δεδομένων που θα αποτελέσουν τη βάση για τον έλεγχο της πολιτικής που έχουμε προδιαγράψει. Η συλλογή των δεδομένων αυτών πραγματοποιείται από κάποιο περιβάλλον παρακολούθησης. Τα δεδομένα αυτά απαρτίζουν μια γνωσιακή βάση δεδομένων, η οποία εισάγεται μαζί με το εκπαιδευμένο Μαρκοβιανό Λογικό Δίκτυο στο εργαλείο συμπερασμού Alchemy. Στη συνέχεια θα παρουσιάσουμε δύο πιθανές γνωσιακές βάσεις που έχουν προκύψει από διαφορετικές εκτελέσεις τις επιχειρησιακής διαδικασίας και το αποτέλεσμα της εκτέλεσης του συμπερασμού.

Η πρώτη γνωσιακή βάση είναι η εξής:

```
LoadForm(BP001)
ValidUser(BP001)

SubmitDestination(BP001)
SubmitDate(BP001)
SubmitPersons(BP001)

Pay(BP001)
SecureProtocol(BP001)
BankReply(BP001)
```

Το αποτέλεσμα της διαδικασίας του συμπερασμού είναι το εξής:

```
Reservation(BP001) 0.99995
```

Παρατηρούμε λοιπόν ότι στη συγκεκριμένη περίπτωση, η επιχειρησιακή διαδικασία εκτελέστηκε με πλήρη επιτυχία, όσον αφορά την εξεταζόμενη πολιτική, καθώς η πιθανότητα ικανοποίησής της είναι 0.99995.

Η δεύτερη γνωσιακή βάση είναι η εξής:

```
LoadForm(BP001)
SubmitDestination(BP001)
SubmitDate(BP001)
SubmitPersons(BP001)
Pay(BP001)
SecureProtocol(BP001)
BankReply(BP001)
```

Το αποτέλεσμα της διαδικασίας του συμπερασμού είναι το εξής:

```
Reservation(BP001) 0.000549945
```

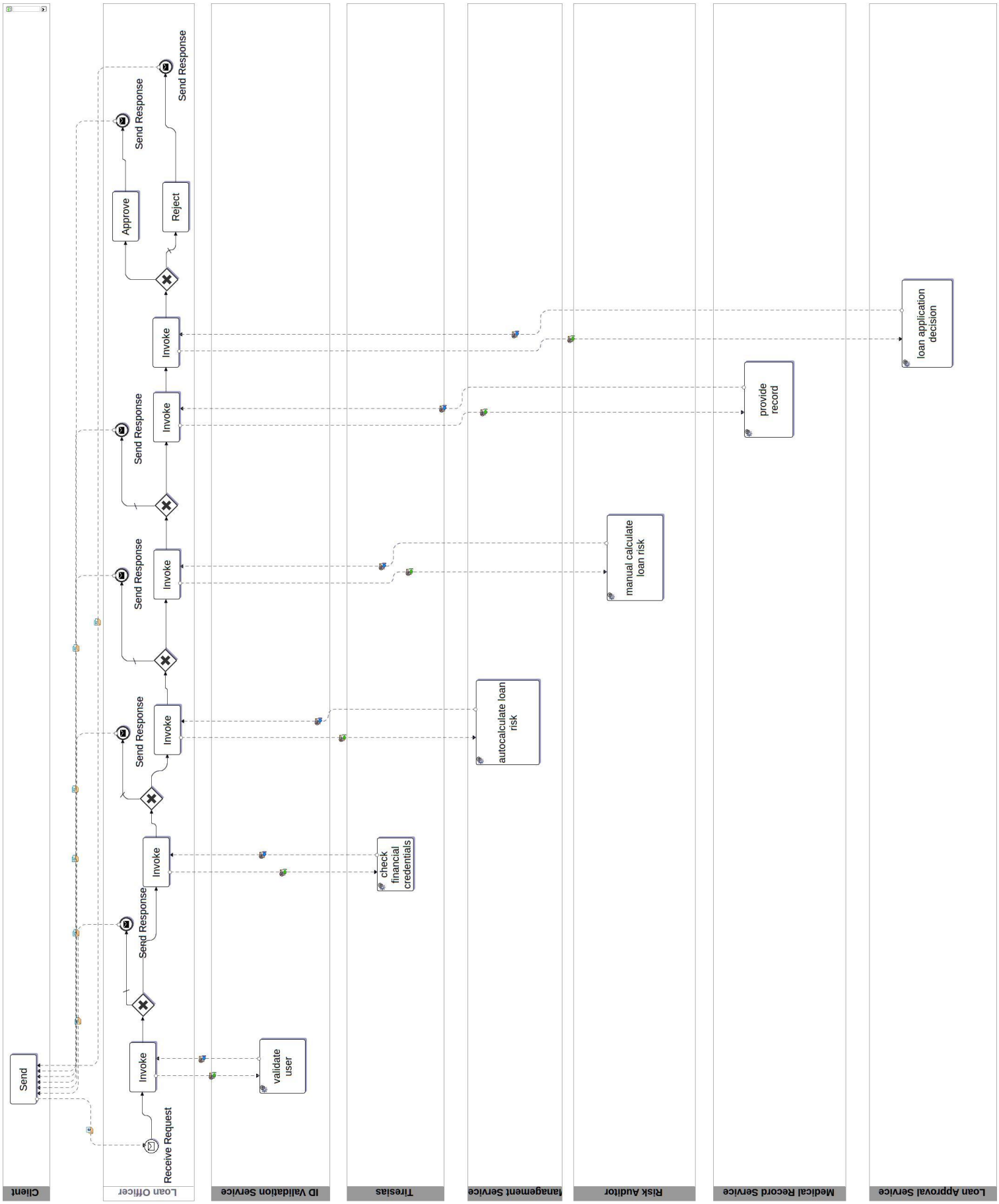
Σε αυτή την περίπτωση η πιθανότητα ικανοποίησης της πολιτικής μας είναι πολύ χαμηλή. Το αποτέλεσμα αυτό είναι λογικό, καθώς αν παρατηρήσει κανείς την γνωσιακή βάση που δίνεται ως είσοδος, δεν υπάρχει πληροφορία για έγκυρη εισαγωγή του χρήστη. Το γεγονός αυτό έχει θεωρηθεί πολύ σημαντικό κατά την εκπαίδευση του Μαρκοβιανού Λογικού Δικτύου και κατά συνέπεια η μη γνώση ικανοποίησής του επηρεάζει αρνητικά το αποτέλεσμα.

6.3 Περίπτωση Χρήσης 2

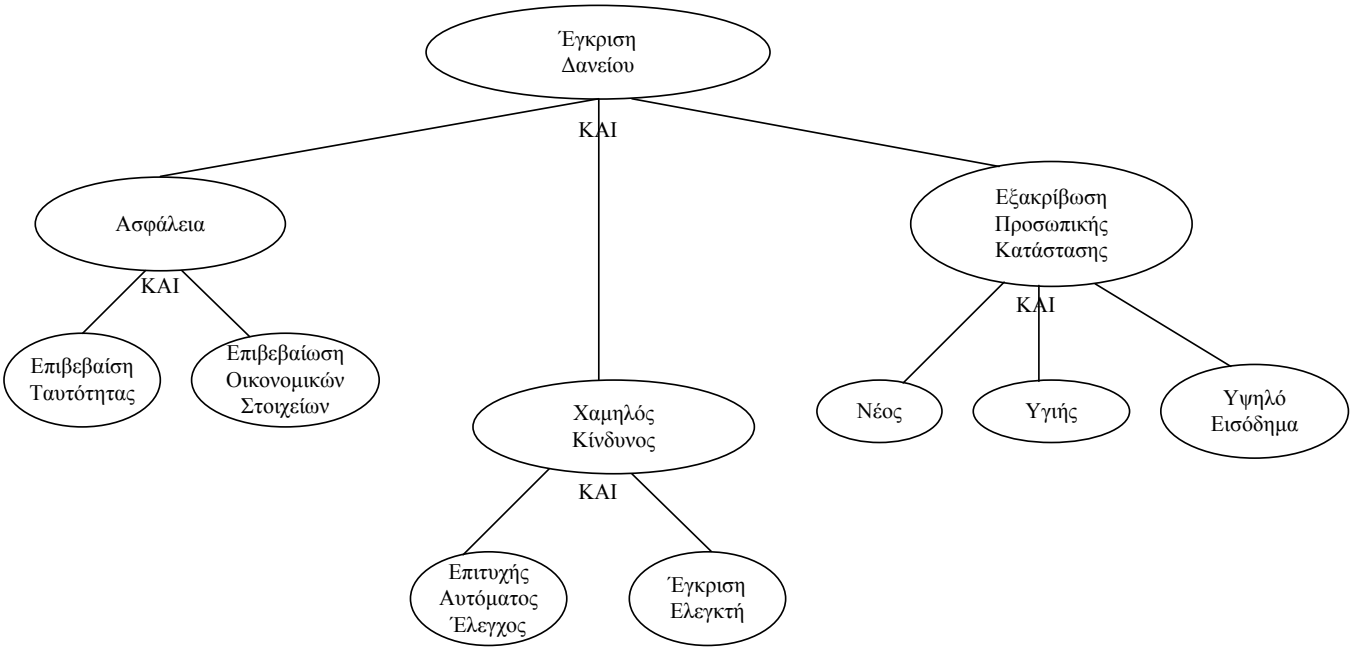
Η δεύτερη περίπτωση χρήσης που θα παρουσιάσουμε αφορά μια επιχειρησιακή διαδικασία, η οποία αφορά την έγκριση ενός δανείου. Όπως και στο προηγούμενο παράδειγμα, έτσι και αυτή η επιχειρησιακή διαδικασία είναι σχεδιασμένη και υλοποιημένη στο εργαλείο Intalio Designer, βασισμένη στο πρότυπο της Υπηρεσιοκεντρικής Αρχιτεκτονικής. Το διάγραμμα της επιχειρησιακής διαδικασίας φαίνεται στην Εικόνα 6.3.1 . Στην επιχειρησιακή διαδικασία αυτή λαμβάνουν μέρος οκτώ συμμετέχοντες. Ο πρώτος συμμετέχων είναι ο πελάτης (Client), ο οποίος καταθέτει στον αρμόδιο υπάλληλο της τράπεζας (Loan Officer) την αίτηση για το δάνειό του (Send). Ο αρμόδιος υπάλληλος έχει τον ρόλο του ενορχηστρωτή της επιχειρησιακής διαδικασίας. Πρώτα λαμβάνει την αίτηση του πελάτη (Receive Request) και στη συνέχεια καλεί (Invoke) την υπηρεσία της Αρχής Ταυτοποίησης Στοιχείων (ID Validation Service) με σκοπό τον έλεγχο της εγκυρότητας των προσωπικών στοιχείων που έχει υποβάλει ο πελάτης. Η υπηρεσία της Αρχής Ταυτοποίησης Στοιχείων εξετάζει τα στοιχεία του πελάτη (validate user) και επιστρέφει στον αρμόδιο τραπεζικό υπάλληλο το αποτέλεσμα του ελέγχου. Στην περίπτωση που τα στοιχεία δεν είναι έγκυρα ο υπάλληλος στέλνει ένα μήνυμα στον πελάτη για το αποτέλεσμα και η διαδικασία τερματίζει. Εάν τα στοιχεία είναι έγκυρα τότε ο τραπεζικός υπάλληλος καλεί (Invoke) την υπηρεσία Εξακρίβωσης Οικονομικών Στοιχείων (Tiresias), η οποία με τη σειρά της ελέγχει τα οικονομικά στοιχεία που έχει υποβάλει ο πελάτης και επιστρέφει το αποτέλεσμα του ελέγχου στον υπάλληλο. Εάν υπάρξει κάποιο αρνητικό αποτέλεσμα κατά τον έλεγχο των οικονομικών τότε ο υπάλληλος αποστέλλει ένα μήνυμα στον πελάτη και τερματίζει την επιχειρησιακή διαδικασία. Στην περίπτωση που ο έλεγχος των οικονομικών στοιχείων είναι επιτυχής ο τραπεζικός υπάλληλος καλεί (Invoke) την υπηρεσία Διαχείρισης Κινδύνου Δανειοδότησης (Risk Management Service), η οποία υπολογίζει τη βαθμίδα κινδύνου στην οποία ανήκει ο πελάτης (autocalculate loan risk). Όπως και πριν εάν προκύψει κάποιο σφάλμα κατά τη διαδικασία αποστέλλεται ένα μήνυμα στον πελάτη από τον υπάλληλο και τερματίζεται η διαδικασία. Σε αντίθετη περίπτωση η επιχειρησιακή διαδικασία συνεχίζεται με τον υπάλληλο

να στέλνει (Invoke) τα δεδομένα της αίτηση στον Ελεγκτή Κινδύνου Δανειοδότησης (Risk Auditor), ο οποίος θα κατατάξει και αυτός σε κάποια βαθμίδα κινδύνου (manual calculate loan risk) τον πελάτη που έκανε την αίτηση δανείου. Εάν προκύψει κάποιο σφάλμα στέλνεται ανάλογο μήνυμα στον πελάτη από τον υπάλληλο, αλλιώς η επιχειρησιακή διαδικασία συνεχίζεται και ο υπάλληλος καλεί (Invoke) την υπηρεσία Ιατρικού Ιστορικού (Medical Record Service) με σκοπό να ενημερωθεί για την κατάσταση της υγείας του πελάτη. Όταν ο υπάλληλος λάβει τα αποτελέσματα από την υπηρεσία Ιατρικού Ιστορικού, προωθεί (Invoke) όλες τις πληροφορίες που έχει συγκεντρώσει στην υπηρεσία του Τμήματος Έγκρισης Δανείων (Loan Approval Service). Στην περίπτωση που η απάντηση είναι θετική, ο υπάλληλος ετοιμάζει μια αναφορά έγκρισης (Approve) και την αποστέλλει στον πελάτη, αλλιώς ετοιμάζει μια αναφορά απόρριψης (Reject), η οποία πάλι αποστέλλεται στον πελάτη.

Η πολιτική που θα παρουσιάσουμε στο παράδειγμα αυτό αφορά την δραστηριότητα της δημιουργίας αναφοράς έγκρισης δανείου. Στόχος της συγκεκριμένης πολιτικής είναι ο έλεγχος των προϋποθέσεων που πρέπει να πληρούνται προκειμένου να είναι επιτυχής η έγκριση ενός δανείου. Τα τρία βασικά σημεία στα οποία επικεντρώνεται η πολιτική είναι η εγκυρότητα των στοιχείων του πελάτη, η ανάλυση κινδύνου δανειοδότησης και κατάσταση της υγείας του πελάτη. Η ιδιαιτερότητα της συγκεκριμένης πολιτικής έχει να κάνει με την πιθανή έλλειψη στοιχείων, όπως το ιατρικό ιστορικό του αιτούντος. Η έλλειψη αυτή προκαλεί μία αβεβαιότητα ως προς την ικανοποίηση της πολιτικής, η οποία θα αντιμετωπιστεί με τη χρήση ενός εκπαιδευμένου Μαρκοβιανού Λογικού Δικτύου, αξιοποιώντας προηγούμενη εμπειρία κατά την διαδικασία της εκπαίδευσής του. Το μοντέλο στόχων της πολιτικής αυτής παρουσιάζεται στην Εικόνα 6.3.2 .



Εικόνα 6.3.1: Επιχειρησιακή Διαδικασία Έγκρισης Δανείου



Εικόνα 6.3.2: Μοντέλο Στόχων Περίπτωσης Χρήσης 2

Όπως και στο προηγούμενο παράδειγμα, έτσι και εδώ κάθε κόμβος του μοντέλου στόχων επισημειώνεται με κάποιο κατηγόρημα ή κάποιο τύπο λογικής πρώτης τάξης. Η αντιστοίχιση έχει ως εξής:

Έγκριση Δανείου → `LoanApproval(x)`

Ασφάλεια → `Security(x)`

Χαμηλός Κίνδυνος → `LowRisk(x)`

Εξακρίβωση Προσωπικής Κατάστασης → `Validation(x)`

Επιβεβαίωση Ταυτότητας → `IDValidation(x)`

Επιβεβαίωση Οικονομικών Στοιχείων → `FinancialCredentialsValidation(x)`

Επιτυχής Αυτόματος Έλεγχος → `AutoCalculatedRisk(x,z) ^lessThan(z,4) => AutoRisk(x)`

Έγκριση Ελεγκτή → `ManualCalculatedRisk(x,y)^lessThan(y,4) => ManualRisk(x)`

Νέος → `Age(x,w) ^ lessThan(w,47) => Young(x)`

Υγιής → `!LifeThreateningDisease(x)^!InheritedDisease(x)^ !RecentlyHospitalised(x) => Healthy(x)`

Υψηλό Εισόδημα → `Income(x,k)^greaterThan(k,2)=> HighIncome(x)`

Το όρισμα x των παραπάνω κατηγορημάτων αναπαριστά το αναγνωριστικό της επιχειρησιακής διαδικασίας. Το όρισμα w αναπαριστά την ηλικία του αιτούντος. Τα ορίσματα z και y αναπαριστούν την κατηγορία κινδύνου δανειοδότησης, ενώ το όρισμα k αναπαριστά την κατηγορία εισοδήματος του αιτούντος.

Μετά την επισημείωση του μοντέλου στόχων με τα κατηγορήματα που παρουσιάστηκαν παραπάνω παράγεται το εξής Μαρκοβιανό Λογικό Δίκτυο:

```
risk = {0,...,5}
age = {18,...,100}
incomecategory = {1,...,5}

LoanApproval(bp)
Security(bp)
LowRisk(bp)
Validation(bp)

IDValidation(bp)
FinancialCredentialsValidation(bp)

ManualRisk(bp)
AutoRisk(bp)
AutoCalculatedRisk(bp,risk)
ManualCalculatedRisk(bp,risk)

Young(bp)
Healthy(bp)
HighIncome(bp)

Age(bp,age)
LifeThreateningDisease(bp)
InheritedDisease(bp)
RecentlyHospitalised(bp)
Income(bp,incomecategory)

Security(x) ^ LowRisk(x) ^ Validation(x) => LoanApproval(x).

IDValidation(x) ^ FinancialCredentialsValidation(x) => Security(x).

ManualRisk(x) ^ AutoRisk(x) => LowRisk(x).

ManualCalculatedRisk(x,y)^lessThan(y,4) => ManualRisk(x).

AutoCalculatedRisk(x,z)^lessThan(z,4) => AutoRisk(x).

Young(x) ^ Healthy(x) ^ HighIncome(x) => Validation(x).

Age(x,w) ^ lessThan(w,47) => Young(x).

!Cancer(x) ^ !InheritedDisease(x) ^ !RecentlyHospitalised(x) => Healthy(x).

Income(x,k) ^ greaterThan(k,2) => HighIncome(x).
```

Στη συνέχεια προκειμένου να αποδώσουμε βάρη στο παραπάνω Μαρκοβιανό Λογικό Δίκτυο χρησιμοποιούμε την παρακάτω γνωσιακή βάση εκπαίδευσης, η οποία έχει προκύψει από προηγούμενες εκτελέσεις του συστήματος.

```
!LoanApproval(BP1)
!Security(BP1)
!IDValidation(BP1)
!FinancialCredentialsValidation(BP1)
LowRisk(BP1)
ManualRisk(BP4)
!AutoRisk(BP4)
Validation(BP1)
Young(BP1)
Age(BP1,35)
Healthy(BP1)
!LifeThreateningDisease(BP1)
HighIncome(BP1)
Income(BP1,5)
!LoanApproval(BP2)
!Security(BP2)
IDValidation(BP2)
!FinancialCredentialsValidation(BP2)
LowRisk(BP2)
ManualRisk(BP2)
AutoRisk(BP2)
Validation(BP2)
Young(BP2)
Age(BP2,35)
Healthy(BP2)
!LifeThreateningDisease(BP2)
!RecentlyHospitalised(BP2)
HighIncome(BP2)
Income(BP2,4)
!LoanApproval(BP3)
!Security(BP3)
!IDValidation(BP3)
FinancialCredentialsValidation(BP3)
LowRisk(BP3)
ManualRisk(BP3)
AutoRisk(BP3)
Validation(BP3)
Young(BP3)
Age(BP3,35)
Healthy(BP3)
!LifeThreateningDisease(BP3)
!RecentlyHospitalised(BP3)
HighIncome(BP3)
!LoanApproval(BP4)
Security(BP4)
IDValidation(BP4)
FinancialCredentialsValidation(BP4)
!LowRisk(BP4)
!ManualRisk(BP4)
AutoRisk(BP4)
Validation(BP4)
Young(BP4)
Age(BP4,35)
Healthy(BP4)
!LifeThreateningDisease(BP4)
HighIncome(BP4)
Income(BP4,4)
!LoanApproval(BP5)
Security(BP5)
IDValidation(BP5)
FinancialCredentialsValidation(BP5)
LowRisk(BP5)
ManualRisk(BP5)
AutoRisk(BP5)
!Validation(BP5)
Young(BP5)
Age(BP5,35)
!Healthy(BP5)
Cancer(BP5)
HighIncome(BP5)
Income(BP5,4)
!LoanApproval(BP6)
Security(BP6)
IDValidation(BP6)
FinancialCredentialsValidation(BP6)
!LowRisk(BP6)
!ManualRisk(BP6)
!AutoRisk(BP6)
Validation(BP6)
Young(BP6)
Age(BP6,35)
Healthy(BP6)
!LifeThreateningDisease(BP6)
HighIncome(BP6)
Income(BP6,4)
!LoanApproval(BP7)
Security(BP7)
IDValidation(BP7)
FinancialCredentialsValidation(BP7)
!LowRisk(BP7)
!ManualRisk(BP7)
AutoRisk(BP7)
Validation(BP7)
Young(BP7)
Age(BP7,35)
Healthy(BP7)
!LifeThreateningDisease(BP7)
HighIncome(BP7)
Income(BP7,4)
```

Το εκπαιδευμένο Μαρκοβιανό Λογικό Δίκτυο που προκύπτει από το Alchemy ύστερα από την εκτέλεση generative learning είναι το εξής:

```
//predicate declarations
Income(bp,incomecategory)
ManualCalculatedRisk(bp,risk)
RecentlyHospitalised(bp)
LifeThreateningDisease(bp)
FinancialCredentialsValidation(bp)
LoanApproval(bp)
IDValidation(bp)
ManualRisk(bp)
HighIncome(bp)
Security(bp)
Age(bp,age)
InheritedDisease(bp)
AutoRisk(bp)
Validation(bp)
Young(bp)
AutoCalculatedRisk(bp,risk)
LowRisk(bp)
Healthy(bp)
// Security(x) ^ LowRisk(x) ^ Validation(x) => LoanApproval(x).
LoanApproval(a1) v !Security(a1) v !LowRisk(a1) v !Validation(a1).
// IDValidation(x) ^ FinancialCredentialsValidation(x) => Security(x).
Security(a1) v !IDValidation(a1) v !FinancialCredentialsValidation(a1).
// ManualRisk(x) ^ AutoRisk(x) => LowRisk(x).
LowRisk(a1) v !ManualRisk(a1) v !AutoRisk(a1).
// ManualCalculatedRisk(x,y) ^ lessThan(y,4) => ManualRisk(x).
ManualRisk(a1) v !ManualCalculatedRisk(a1,a2) v !(a2 < 4).
// AutoCalculatedRisk(x,z) ^ lessThan(z,4) => AutoRisk(x).
AutoRisk(a1) v !AutoCalculatedRisk(a1,a2) v !(a2 < 4).
// Young(x) ^ Healthy(x) ^ HighIncome(x) => Validation(x).
Validation(a1) v !Young(a1) v !Healthy(a1) v !HighIncome(a1).
// Age(x,w) ^ lessThan(w,47) => Young(x).
Young(a1) v !Age(a1,a2) v !(a2 < 47).
// !LifeThreateningDisease(x) ^ !InheritedDisease(x) ^ !RecentlyHospitalised(x)
=> Healthy(x).
Healthy(a1) v LifeThreateningDisease(a1) v InheritedDisease(a1) v
RecentlyHospitalised(a1).
// Income(x,k) ^ greaterThan(k,2) => HighIncome(x).
HighIncome(a1) v !Income(a1,a2) v !(a2 > 2).
// -7.53268  LoanApproval(a1)
-7.53268  LoanApproval(a1)
// 0.082079  Security(a1)
0.082079  Security(a1)
// 5.56881  LowRisk(a1)
5.56881  LowRisk(a1)
// 0.595748  Validation(a1)
0.595748  Validation(a1)
// 1.60708  IDValidation(a1)
1.60708  IDValidation(a1)
```

```
// 1.60708  FinancialCredentialsValidation(a1)
1.60708  FinancialCredentialsValidation(a1)
// 0.400122  ManualRisk(a1)
0.400122  ManualRisk(a1)
// 0.919202  AutoRisk(a1)
0.919202  AutoRisk(a1)
// -7.31949  AutoCalculatedRisk(a1,a2)
-7.31949  AutoCalculatedRisk(a1,a2)
// -7.06317  ManualCalculatedRisk(a1,a2)
-7.06317  ManualCalculatedRisk(a1,a2)
// 1.26407  Young(a1)
1.26407  Young(a1)
// -0.571018  Healthy(a1)
-0.571018  Healthy(a1)
// 1.26407  HighIncome(a1)
1.26407  HighIncome(a1)
// -4.33438  Age(a1,a2)
-4.33438  Age(a1,a2)
// -6.96541  LifeThreateningDisease(a1)
-6.96541  LifeThreateningDisease(a1)
// -7.53268  InheritedDisease(a1)
-7.53268  InheritedDisease(a1)
// -7.53268  RecentlyHospitalised(a1)
-7.53268  RecentlyHospitalised(a1)
// -1.38435  Income(a1,a2)
-1.38435  Income(a1,a2)
```

Για το παραπάνω παράδειγμα θα παρουσιάσουμε τέσσερις γνωσιακές βάσεις δεδομένων που έχουν προκύψει από την εκτέλεση της επιχειρησιακής διαδικασίας. Κάθε γνωσιακή βάση δεδομένων, όπως και το εκπαιδευμένο Μαρκοβιανό Λογικό Δίκτυο θα εισαχθούν στο εργαλείο συμπερασμού Alchemy προκειμένου να ελέγξουμε την πιθανότητα ικανοποίησης αυτής της πολιτικής που προδιαγράψαμε.

Η πρώτη γνωσιακή βάση δεδομένων που προέκυψε είναι η εξής:

```
Age(BP1,32)
!LifeThreateningDisease(BP1)
!InheritedDisease(BP1)
!RecentlyHospitalised(BP1)
Income(BP1,5)


AutoCalculatedRisk(BP1,5)
ManualCalculatedRisk(BP1,1)

IDValidation(BP1)
FinancialCredentialsValidation(BP1)
```

Το αποτέλεσμα του συμπερασμού για την παραπάνω γνωσιακή βάση είναι το εξής:

```
LoanApproval(BP1) 0.225027
```

Από την παραπάνω γνωσιακή βάση μπορούμε να διακρίνουμε ότι η υπηρεσία αυτόματης διαχείρισης κινδύνου δανειοδότησης κρίνει πως η δανειοδότηση στη συγκεκριμένη περίπτωση είναι υψηλού κινδύνου, ωστόσο κατά την εκπαίδευση του Μαρκοβιανού Λογικού Δικτύου δόθηκε μεγαλύτερη έμφαση στο αποτέλεσμα το ελεγκτή το οποίο είναι θετικό. Για το λόγο αυτό η πιθανότητα ικανοποίησης της πολιτικής είναι μεν χαμηλή αλλά όχι τόσο χαμηλή ώστε να απορριφθεί άμεσα

Η δεύτερη γνωσιακή βάση δεδομένων που προέκυψε είναι η εξής:

```
Age(BP1,32)
!LifeThreateningDisease(BP1)
!InheritedDisease(BP1)
!RecentlyHospitalised(BP1)
Income(BP1,1)


AutoCalculatedRisk(BP1,5)
ManualCalculatedRisk(BP1,1)

IDValidation(BP1)
FinancialCredentialsValidation(BP1)
```

Το αποτέλεσμα του συμπερασμού για την παραπάνω γνωσιακή βάση είναι το εξής:

```
LoanApproval(BP1) 0.00184982
```

Η πιθανότητα ικανοποίησης της πολιτικής σε αυτή την περίπτωση είναι πολύ χαμηλή. Το γεγονός αυτό οφείλεται στο ότι ο αιτών ανήκει σε χαμηλή κατηγορία εισοδήματος, ένας παράγοντας που έχει θεωρηθεί ιδιαίτερα σημαντικός κατά την εκπαίδευση του Μαρκοβιανού Λογικού Δικτύου.

Η τρίτη γνωσιακή βάση δεδομένων που προέκυψε είναι η εξής:

```
Age(BP1,32)
!LifeThreateningDisease(BP1)

Income(BP1,5)

AutoCalculatedRisk(BP1,3)
ManualCalculatedRisk(BP1,1)

IDValidation(BP1)
FinancialCredentialsValidation(BP1)
```

Το αποτέλεσμα του συμπερασμού για την παραπάνω γνωσιακή βάση είναι το εξής:

```
LoanApproval(BP1) 0.79807
```

Σε αυτή την περίπτωση η πιθανότητα ικανοποίησης είναι αρκετά υψηλή. Παρατηρούμε ότι απουσιάζουν δύο βασικά στοιχεία που αφορούν το ιατρικό ιστορικό του αιτούντος, ωστόσο το Μαρκοβιανό Λογικό Δίκτυο έχει εκπαιδευτεί με τέτοιο τρόπο, ώστε να αντιμετωπίζει τέτοιες περιπτώσεις αβεβαιότητας.

Η τέταρτη γνωσιακή βάση δεδομένων που προέκυψε είναι η εξής:

```
Age(BP1,32)
!LifeThreateningDisease(BP1)
!InheritedDisease(BP1)
!RecentlyHospitalised(BP1)

Income(BP1,5)

AutoCalculatedRisk(BP1,3)
ManualCalculatedRisk(BP1,1)

IDValidation(BP1)
FinancialCredentialsValidation(BP1)
```

Το αποτέλεσμα του συμπερασμού για την παραπάνω γνωσιακή βάση είναι το εξής:

```
LoanApproval(BP1) 0.99995
```

Σε αυτή την περίπτωση η πιθανότητα ικανοποίησης της πολιτικής είναι ιδιαίτερα υψηλή, καθώς όλες η προϋποθέσεις έχουν ικανοποιηθεί.

Κεφάλαιο 7: Συμπεράσματα

Στα παραπάνω κεφάλαια παρουσιάστηκε η αρχιτεκτονική και η λειτουργία ενός περιβάλλοντος – πλαισίου που έχει ως στόχο τον έλεγχο της ορθής λειτουργίας Υπηρεσιοκεντρικών συστημάτων, όσον αφορά τις πολιτικές που πρέπει να ικανοποιούν κατά την εκτέλεσή τους. Το περιβάλλον – πλαίσιο αυτό επιχειρεί να δώσει τη δυνατότητα αυτοματοποίησης του ελέγχου συμμόρφωσης, παρακάμπτοντας χρονοβόρες και δαπανηρές διαδικασίες, οι οποίες μέχρι τώρα συνήθως πραγματοποιούνται από εξειδικευμένο προσωπικό. Οι μεθοδολογία ελέγχου που παρουσιάστηκε μπορεί να δώσει ώθηση στην αυξανόμενη αποδοχή της Υπηρεσιοκεντρικής αρχιτεκτονικής για την υλοποίηση επιχειρησιακών διαδικασιών, καθώς υπάρχει πλέον η δυνατότητα ελέγχου της συμμόρφωσης ως προς τις διάφορες πολιτικές που διέπουν την εκτέλεση των διαδικασιών αυτών.

Ένας ακόμη παράγοντας που καθιστά σημαντική τη συνεισφορά του προτεινόμενου περιβάλλοντος – πλαισίου είναι η επιτυχημένη αντιμετώπιση της αβεβαιότητας που μπορεί να προκύψει κατά την εκτέλεση μιας επιχειρησιακής διαδικασίας. Η αβεβαιότητα αυτή μπορεί να είναι προϊόν εσφαλμένης καταγραφής δεδομένων ή να εμπεριέχεται στη φύση των δεδομένων, όπως στην δεύτερη περίπτωση χρήσης που παρουσιάσαμε. Η χρήση των Μαρκοβιανών Λογικών Δικτύων ήταν καταλυτική για την αντιμετώπιση του προβλήματος αυτού, δίνοντας ένα επιπλέον πλεονέκτημα στο προτεινόμενο περιβάλλον – πλαίσιο, σε σχέση με ό,τι έχει προταθεί μέχρι στιγμής από τον τομέα της έρευνας στην Τεχνολογία Λογισμικού.

Αξιολογώντας την απόδοση και την πολυπλοκότητα του περιβάλλοντος – πλαισίου που προτείνουμε, θα επικεντρωθούμε σε δύο βασικά σημεία. Το πρώτο σημείο αφορά την απόδοση, η οποία εξαρτάται άμεσα από τον αλγόριθμο που θα χρησιμοποιηθεί για την διαδικασία του συμπερασμού και τη λεπτομερή ανάλυση που επιθυμεί ο χρήστης ως προς το αποτέλεσμα. Το δεύτερο σημείο αφορά την πολυπλοκότητα η οποία σχετίζεται με την μοντελοποίηση των πολιτικών και τη σύνδεσή τους με κατηγορήματα και τύπους λογικής πρώτης τάξης. Ο χρήστης κατά τη διαδικασία αυτή θα πρέπει να διατηρεί στο ελάχιστο το πλήθος των ορισμάτων των κατηγορημάτων, καθώς μπορεί να δημιουργηθούν μη επιθυμητές εξαρτήσεις, οι οποίες δύσκολα ελέγχονται.

7.1 Μελλοντικές Εργασίες

Η παρούσα διπλωματική εργασία είχε ως στόχο την ανάπτυξη ενός περιβάλλοντος πλαισίου μέσω του οποίου θα γινόταν δυνατή η δοκιμή ορισμένων θεωριών, όπως αυτή των Μαρκοβιανών Λογικών Δικτύων, για τον έλεγχο της συμμόρφωσης επιχειρησιακών διαδικασιών. Η επικέντρωση σε αυτό τον σκοπό δίνει τη δυνατότητα επέκτασης του συγκεκριμένου περιβάλλοντος – πλαισίου προσδίδοντάς του βελτιώσεις και νέα χαρακτηριστικά.

Μία από τις βελτιώσεις που επιδέχεται το προτεινόμενο περιβάλλον – πλαίσιο είναι η προσθήκη μιας, φιλικής προς το χρήστη, διαπροσωπείας, κάτι το οποίο θα αύξανε δραστικά τη λειτουργικότητά του. Στα πλαίσια της παρούσας διπλωματικής εργασίας ως διαπροσωπεία χρησιμοποιήθηκε το περιβάλλον που διαθέτει το Eclipse Modeling Framework.

Ένα ακόμη σημαντικό χαρακτηριστικό που μπορεί να βελτιώσει την αποδοτικότητα του προτεινόμενου περιβάλλοντος – πλαισίου είναι η προσθήκη μηχανισμών ανάλυσης αιτίας σφάλματος (root cause analysis), ώστε μετά τον έλεγχο της συμμόρφωσης να καθίσταται δυνατή η εξεύρεση του συστατικού λογισμικού που προκάλεσε μια πιθανή παραβίαση πολιτικής. Τέτοιοι μηχανισμοί μπορούν να υλοποιηθούν και με τη χρήση των Μαρκοβιανών Λογικών Δικτύων ή άλλων μεθόδων που παρέχει ο κλάδος της Τεχνητής Νοημοσύνης.

Τέλος, θα μπορούσε να εξεταστεί η αξιοποίηση του παρόντος περιβάλλοντος – πλαισίου για τον έλεγχο συμμόρφωσης σε συστήματα τύπου cloud. Τέτοια συστήματα παρουσιάζουν μεγάλο πλήθος από ομοιότητες με τα Υπηρεσιοκεντρικά συστήματα, με τη διαφορά ότι τα συστήματα cloud διατηρούν ακόμα πιο χαλαρούς δεσμούς ανάμεσα στις δομικές τους μονάδες.

Κεφάλαιο 8: Βιβλιογραφία

[1] T. Erl, *SOA Principles of Service Design (Prentice Hall Service-Oriented Computing Series from Thomas Erl)*, Prentice Hall, 2007.

[2] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design (Prentice Hall Service-Oriented Computing Series from Thomas Erl)*, Prentice Hall, 2005.

[3] T. Erl, A. Karmarkar, P. Walmsley, H. Haas, L.U.Y. Ph.D., K. Liu, D.U. Orchard, A. Tost, and J. Pasley, *Web Service Contract Design for SOA (Prentice Hall Service-Oriented Computing Series from Thomas Erl)*, Prentice Hall, 2008.

[4] Y. Vasiliev, *SOA and WS-BPEL*, PACKT PUBLISHING, 2007.

[5] C. Barreto, A. Systems, V. Bullard, T. Erl, J. Evdemon, D. Jordan, K. Kand, S. Moser, R. Stout, R. Ten-hove, I. Trickovic, D.V.D. Rijn, and A. Yiu, "Web Services Business Process Execution Language Version 2 . 0," *Language*, 2007.

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

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

[8] "Business Process Management System, BPM Consultants, BPM Software Solutions."

[9] O.M.G., "OMG Unified Modeling Language TM (OMG UML), Infrastructure," 2010.

[10] O.M.G., "Business Process Model and Notation (BPMN)," *Business*, 2011.

[11] O.M.G., "BPMN 2.0 by Example," *Group*, vol. 0, 2010.

[12] *XML 1.0 Specification*, W3.org, .

[13] "SOAP Version 1.2 Part 0: Primer (Second Edition)."

[14] "Introduction to WSDL," *Weather*, 2001.

[15] Sun-Microsystems, "UDDI Technical White Paper," *International Business*, 2002.

[16] "Web Services Flow Language (WSFL 1.0)," *Language*, 2001.

[17] "Business Process Execution Language - Wikipedia, the free encyclopedia."

[18] O.M.G., "OMG Systems Modeling Language (OMG SysML ™)," *Source*, 2010.

[19] Z. Balfagih and M.F.B. Hassan, *Agent based monitoring framework for SOA applications quality*, IEEE, 2010.

[20] Z. Balfagih and M.F. Hassan, “Quality Model for Web Services from Multi-stakeholders’ Perspective,” *2009 International Conference on Information Management and Engineering*, 2009, pp. 287-291.

[21] L. Baresi and P. Plebani, “WS-Policy for Service Monitoring,” *Manager*, 2006, pp. 72-83.

[22] A. Anderson, “XACML-Based Web Services Policy Constraint Language (WS-PolicyConstraints),” *October*, 2005, pp. 1-36.

[23] D. Żmuda, M. Psiuk, and K. Zieliński, “Dynamic monitoring framework for the SOA execution environment,” *Procedia Computer Science*, vol. 1, May. 2010, pp. 125-133.

[24] S. Sadiq, G. Governatori, and K. Namiri, “Modeling control objectives for business process compliance,” Sep. 2007, pp. 149-164.

[25] K. Namiri and N. Stojanovic, “Pattern-based design and validation of business process compliance,” *Proceedings of the 2007 OTM Confederated international conference on On the move to meaningful internet systems: CoopIS, DOA, ODBASE, GADA, and IS-Volume Part I*, Springer-Verlag, 2007, p. 59–76.

[26] H. Tran, T. Holmes, E. Oberortner, E. Mulo, A.B. Cavalcante, J. Serafinski, M. Tluczek, A. Birukou, F. Daniel, P. Silveira, U. Zdun, and S. Dustdar, *An End-to-End Framework for Business Compliance in Process-Driven SOAs*, IEEE, 2010.

[27] C. Rodr’\iguez, P. Silveira, F. Daniel, and F. Casati, “Analyzing compliance of service-based business processes for root-cause analysis and prediction,” *Current Trends in Web Engineering*, 2010, p. 277–288.

[28] F. Daniel, F. Casati, V. D’Andrea, E. Mulo, U. Zdun, S. Dustdar, S. Strauch, D. Schumm, F. Leymann, S. Sebahi, F.D. Marchi, and M.-S. Hacid, “Business Compliance Governance in Service-Oriented Architectures,” *2009 International Conference on Advanced Information Networking and Applications*, 2009, pp. 113-120.

[29] Y. Liu, S. Muller, and K. Xu, “A static compliance-checking framework for business process models,” *IBM Systems Journal*, vol. 46, Apr. 2007, pp. 335-361.

[30] J. Mylopoulos, L. Chung, and E. Yu, “From object-oriented to goal-oriented requirements analysis,” *Communications of the ACM*, vol. 42, Jan. 1999, pp. 31-37.

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

[32] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach (2nd Edition)*, Prentice Hall, 2002.

[33] D. Vrakas, *Artificial Intelligence for Advanced Problem Solving Techniques*, Information Science Reference, 2008.

[34] I. Bratko, *Prolog Programming for Artificial Intelligence (International Computer Science Series)*, Addison-Wesley Educational Publishers Inc, 2011.

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

[36] M. Richardson and P. Domingos, "Markov Logic Networks," 2006.