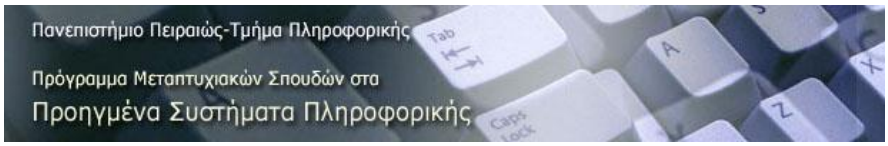
Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών
«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάλυση της Διαχείρισης Σύνθετων Συμβάντων (Complex Event Processing) στη λήψη αποφάσεων και εφαρμογή της σε χαρακτηριστικές μελέτες περίπτωσης**
Ονοματεπώνυμο Φοιτητή	**Τσικερδής Δημήτριος του Σπυρίδωνα**
Αριθμός Μητρώου	**ΜΠΣΠ/07030**
Κατεύθυνση	**Συστήματα Υποστήριξης Αποφάσεων**
Επιβλέπων	**Δημήτρης Αποστόλου, Επίκουρος**



Ημερομηνία Παράδοσης **Απρίλιος 2011**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα

Περίληψη

Σήμερα οι επιχειρήσεις καλούνται να αναπτύξουν μεθοδολογίες και τεχνολογίες που θα τους επιτρέπουν να διαχειρίζονται όλο το πλήθος των δεδομένων που διακινούνται στις επιχειρησιακές διαδικασίες και που προέρχονται από πολλές πηγές, προκειμένου να λάβουν άμεσα τις απαραίτητες αποφάσεις. Η Επεξεργασία Συμβάντων, η οποία αποτελεί ένα νέο τομέα της Πληροφορικής, επιδιώκει να επιλύσει το πρόβλημα διαχείρισης πληροφοριών από ετερογενείς πηγές και να συνεισφέρει στη λήψη αποφάσεων.

Η παρούσα διπλωματική εργασία επιδιώκει να αναλύσει την τεχνολογία αυτή και το κατά πόσο αποτελεί μια αποτελεσματική απάντηση στην ανάγκη αυτή των επιχειρήσεων. Επίσης, γίνεται διερεύνηση του πώς μπορεί να συνδυαστεί με την Τεχνολογία Έμπειρων Συστημάτων και πώς μπορεί τελικά να εφαρμοστεί σε χαρακτηριστικά προβλήματα λήψης απόφασης.

Η δομή της εργασίας μπορεί να χωριστεί σε δύο μέρη. Στο πρώτο μέρος αναλύεται η Επεξεργασία Συμβάντων και ο τρόπος υλοποίησής της μέσω της Αρχιτεκτονικής Καθοδηγούμενης από Συμβάντα (Event Driven Architecture – EDA). Στη συνέχεια παρουσιάζονται βασικά χαρακτηριστικά των Έμπειρων Συστημάτων για να καταλήξουμε σε μια Υβριδική Αρχιτεκτονική όπου συνδυάζονται η EDA αρχιτεκτονική με αυτή των Έμπειρων Συστημάτων.

Στο δεύτερο μέρος παρουσιάζεται αναλυτικά μια τεχνολογία υλοποίησης συστημάτων με δυνατότητες επεξεργασίας συμβάντων, μέσω της πλατφόρμας Drools της Jboss. Παρουσιάζονται επίσης δύο χαρακτηριστικές μελέτες περίπτωσης που αφορούν στην ανίχνευση απάτης και στη διαχείριση μετοχών.

Abstract

Nowadays, companies need to develop methodologies and technologies that will enable them to manage their vast amount of data moving across operational procedures and that come from many sources, in order to make immediate decisions. Event Processing, which is a new area of Information Technology, seeks to solve the problem of managing information from heterogeneous sources and contribute to decision making.

This thesis aims to analyze this technology and whether it is an effective response to these specific needs of a company. We also investigate how this technology can be combined with Expert Systems and whether it can eventually be applied to typical problems of decision making.

The structure of the thesis can be divided into two main sections. The first section presents the Event Processing and how it can be implemented through Event Driven Architecture (EDA). We then present key features of Expert Systems in order to result in a hybrid architecture that combines the EDA architecture with that of Expert Systems.

In the second section we analyze how an event processing technology can be implemented through Drools, which is an open source platform from Jboss. We also present two typical case studies relating to fraud detection and managing stocks.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

1 Εισαγωγή

Στο σύγχρονο παγκοσμιοποιημένο επιχειρησιακό περιβάλλον οι εταιρείες καλούνται να επεξεργαστούν έναν τεράστιο όγκο πληροφοριών καθώς και να μπορούν να λαμβάνουν κρίσιμες αποφάσεις σε πολύ μικρό χρονικό διάστημα. Η ικανότητα των επιχειρήσεων να ανταποκριθούν στους όλο και ταχύτερους ρυθμούς λήψης αποφάσεων είναι ένας από τους βασικότερους παράγοντες που θα τους εξασφαλίσει όχι μόνο επιβίωση αλλά πολύ περισσότερο ανάπτυξη σε ένα όλο και πιο ανταγωνιστικό περιβάλλον.

Η αλματώδης ανάπτυξη της τεχνολογίας προσέφερε ένα πλήθος εφαρμογών για την αποτελεσματικότερη διαχείριση πελατών (CRM), πόρων (ERP), διαδικασιών (BPM), αποθηκών (logistics) κ.α. Παρά το γεγονός ότι η εισαγωγή όλων αυτών των εφαρμογών αύξησε με εκθετικούς ρυθμούς την παραγωγικότητα της εργασίας, ταυτόχρονα δημιούργησε ένα πλήθος διαδικασιών και έναν τεράστιο όγκο πληροφοριών. Αυτό είχε ως συνέπεια να επιφέρει σε πολλές περιπτώσεις το αντίθετο αποτέλεσμα, μια «υπερφόρτωση» πληροφοριών η οποία κατέστησε δυσχερή τη λήψη σωστών αποφάσεων.

Η Επεξεργασία Συμβάντων (Event Processing), έρχεται να επιλύσει το πρόβλημα διαχείρισης πληροφοριών από ετερογενείς πηγές και να επιτρέψει τη λήψη αποφάσεων σε σχεδόν πραγματικό χρόνο. Η Επεξεργασία Συμβάντων σαν σκοπό έχει να συγκεντρώσει τις πληροφορίες από βάσεις δεδομένων, εφαρμογές ακόμα και RFID αισθητήρες και να προσπαθήσει να συνδυάσει όλες αυτές τις πληροφορίες προκειμένου να ανιχνεύσει πρότυπα, τάσεις, ευκαιρίες και κινδύνους. Οι πληροφορίες αυτές παρέχουν τη δυνατότητα στις επιχειρήσεις να λαμβάνουν αποφάσεις εκμεταλλευόμενες τις ευκαιρίες και να αντιδρούν αποτελεσματικότερα στους κινδύνους. Στις περισσότερες περιπτώσεις οι αποφάσεις αυτές πρέπει να λαμβάνονται σε πραγματικό χρόνο ή σε σχεδόν πραγματικό χρόνο χωρίς να απαιτείται διαμεσολάβηση του ανθρώπου. Οπότε στην Επεξεργασία Συμβάντων παρατηρούμε στοιχεία και από την Τεχνολογία των Έμπειρων Συστημάτων.

Η διερεύνηση των δυνατοτήτων που παρέχει ο συνδυασμός νέων τεχνολογιών όπως της Επεξεργασίας Συμβάντων με τεχνολογίες όπως των Έμπειρων Συστημάτων στην λήψη αποφάσεων καθώς επίσης και η εφαρμογή τους σε επιχειρησιακό περιβάλλον αποτελεί αντικείμενο μελέτης του πεδίου των Συστημάτων Υποστήριξης Αποφάσεων (Σ.Υ.Α.)

1.1 Σκοπός

Σκοπός της παρούσας εργασίας είναι η εισαγωγή στην Επεξεργασία Συμβάντων και πιο συγκεκριμένα στην Επεξεργασία Σύνθετων Συμβάντων (Complex Event Processing – CEP), ο συνδυασμός της με την Τεχνολογία Έμπειρων Συστημάτων καθώς επίσης και η εφαρμογή της σε χαρακτηριστικά προβλήματα λήψης απόφασης.

1.2 Περιγραφή της εργασίας

Όπως διακρίνουμε από την παραπάνω περιγραφή ο σκοπός της εργασίας αυτής είναι διττός. Από τη μια γίνεται ανάλυση της επεξεργασίας συμβάντων και από την άλλη παρουσίαση των εφαρμογών της σε πραγματικά προβλήματα λήψης απόφασης. Η οργάνωση της εργασίας ακολουθεί την ίδια λογική και ουσιαστικά μπορεί να χωριστεί σε δύο μέρη.

Στο πρώτο μέρος, το οποίο είναι θεωρητικό, αναλύεται η Επεξεργασία Συμβάντων και ο τρόπος υλοποίησής της μέσω της Αρχιτεκτονικής Καθοδηγούμενης από Συμβάντα (Event Driven Architecture – EDA). Στη συνέχεια παρουσιάζονται βασικά χαρακτηριστικά των Έμπειρων Συστημάτων για να καταλήξουμε σε μια Υβριδική Αρχιτεκτονική όπου συνδυάζονται η EDA αρχιτεκτονική με αυτή των Έμπειρων Συστημάτων.

Στο δεύτερο μέρος παρουσιάζεται αναλυτικά μια τεχνολογία υλοποίησης συστημάτων με δυνατότητες επεξεργασίας συμβάντων, μέσω της πλατφόρμας Drools της Jboss. Αναλύεται η αρχιτεκτονική της, η λειτουργικότητά της, οι τρόποι συγγραφής κανόνων και οι αλγόριθμοι για την επεξεργασία συμβάντων.

Τέλος, ακολουθεί περιγραφή δύο μελετών περίπτωσης, για ανίχνευση απάτης και διαχείριση μετοχών, οι οποίες πληρούν τα ιδιαίτερα χαρακτηριστικά μιας event-driven προσέγγισης.

1.3 Οργάνωση της εργασίας

Στο 1ο κεφάλαιο γίνεται μια σύντομη περιγραφή της παρούσας εργασίας. Αναφέρονται τα προβλήματα τα οποία καλείται να επιλύσει καθώς και το επιστημονικό πεδίο στο οποίο εφαρμόζεται. Αναφέρεται επίσης ο σκοπός της και παρουσιάζεται η διάρθρωση της.

Στο 2ο κεφάλαιο γίνεται ανάλυση της επεξεργασίας συμβάντων. Το κεφάλαιο χωρίζεται σε δύο μέρη. Στο πρώτο μέρος αποσαφηνίζονται βασικές έννοιες όπως το τι θεωρείται συμβάν, πρότυπο συμβάντος κ.α. Στη συνέχεια αναλύεται η EDA αρχιτεκτονική και το πως υλοποιείται στις διάφορες μορφές της επεξεργασίας συμβάντων και κυρίως στην επεξεργασία σύνθετων συμβάντων. Στο τέλος του κεφαλαίου παρουσιάζονται χαρακτηριστικές εφαρμογές της αρχιτεκτονικής αυτής σε Συστήματα Υποστήριξης Αποφάσεων και σε επιχειρησιακές διαδικασίες.

Στο 3ο κεφάλαιο παρουσιάζονται βασικές έννοιες των Έμπειρων Συστημάτων. Επιπλέον, αναλύεται η αρχιτεκτονική τους και ο τρόπος ανάπτυξης ενός έμπειρου συστήματος. Αναφέρονται βασικά συστατικά στοιχεία και αλγόριθμοι που εφαρμόζονται σε ένα Έμπειρο Σύστημα.

Το 4ο κεφάλαιο αποτελεί μια πρόταση υβριδικής αρχιτεκτονικής όπου συνδυάζεται η επεξεργασία συμβάντων με τα έμπειρα συστήματα στη λήψη αποφάσεων. Στο κεφάλαιο αυτό επιλέγονται στοιχεία από τα δύο προηγούμενα κεφάλαια, όπου η Αρχιτεκτονική των έμπειρων συστημάτων εμπλουτίζεται με δυνατότητες επεξεργασίας συμβάντων σε πραγματικό χρόνο.

Στο 5ο κεφάλαιο αναλύεται η λειτουργία του Drools, μια πλατφόρμα ανοικτού κώδικα η οποία παρέχει λειτουργίες επεξεργασίας συμβάντων. Παρουσιάζονται τα χαρακτηριστικά της μηχανής κανόνων (rule engine), οι τρόποι συγγραφής των κανόνων καθώς επίσης και η αρχιτεκτονική του συστήματος.

Στο 6ο κεφάλαιο έχουμε τη περιγραφή δύο βασικών μελετών περίπτωσης. Παρουσιάζονται οι επιχειρησιακοί κανόνες για την υλοποίηση ενός συστήματος ανίχνευσης απάτης (fraud detection) και ενός συστήματος επεξεργασίας μετοχών (stock broker) βάσει του υποσυστήματος του Drools για event processing, το Fusion. Αναλύονται βασικές εντολές που επεκτείνουν τη λειτουργικότητα μιας μηχανής κανόνων και για την επεξεργασία συμβάντων (events).

Στο 7ο κεφάλαιο γίνεται μια επισκόπηση της διπλωματικής εργασίας με τα συμπεράσματα καθώς επίσης και τα προβλήματα που παρουσιάστηκαν κατά τη διαδικασία εκπόνησής της. Προτείνονται επίσης και μελλοντικές επεκτάσεις τόσο στην λειτουργικότητα της μηχανής κανόνων του Drools όσο και στη γενικότερη τεχνολογική προσέγγιση της επεξεργασίας συμβάντων.

Στο τέλος της εργασίας ακολουθεί Παράρτημα το οποίο αποτελείται από τρία μέρη. Το ένα αφορά στα Έμπειρα Συστήματα όπου γίνεται μια αναλυτική παρουσίαση των σημαντικότερων εξ αυτών για λόγους πληρότητας. Στη συνέχεια αναφέρονται τα βασικότερα εμπορικά συστήματα μηχανής κανόνων και συστημάτων επεξεργασίας συμβάντων.

2 Επεξεργασία Συμβάντων

Σκοπός αυτού του κεφαλαίου είναι η ανάλυση της τεχνολογίας συμβάντων και η περιγραφή ενός συστήματος το οποίο να μπορεί να επεξεργάζεται συμβάντα. Τα συμβάντα που πρέπει να επεξεργαστεί το σύστημα αυτό ανήκουν σε μια κατηγορία που στη διεθνή βιβλιογραφία ορίζονται ως σύνθετα συμβάντα (complex events) [3w12]. Η επεξεργασία τους εμπεριέχει τον συγκερασμό δύο τεχνολογιών, της αρχιτεκτονικής καθοδηγούμενης από συμβάντα (EDA - Event-Driven Architecture) και της επεξεργασίας σύνθετων συμβάντων (Complex Event Processing).

Η αρχιτεκτονική καθοδηγούμενη από συμβάντα (EDA) αποτελεί μια αρχιτεκτονική συστημάτων επικεντρωμένη σε ασύγχρονες push-based επικοινωνίες. Το πλεονέκτημα της σε σχέση με άλλες αρχιτεκτονικές είναι η άμεση αντίδραση των συστημάτων αυτών σε ερεθίσματα του περιβάλλοντος. Για το λόγο αυτό χρησιμοποιείται όλο και περισσότερο σε κρίσιμες επιχειρησιακές διαδικασίες οι οποίες απαιτούν τον λιγότερο δυνατό χρόνο διεκπεραίωσης. Οι διαδικασίες που είναι σχεδιασμένες με βάση την EDA είναι πιο εύκολο να παραμετροποιηθούν και να τροποποιηθούν σε σχέση με τις παραδοσιακές αρχιτεκτονικές.

Η επεξεργασία σύνθετων συμβάντων (CEP) αποτελεί εξελιγμένη μορφή της EDA. Σκοπός της είναι η εξαγωγή γνώσης από πολλαπλά γεγονότα. Η μετατροπή της πληροφορίας σε γνώση επιτυγχάνεται μέσω διαδικασίας αντιστοίχησης προτύπων (pattern matching). Τα συμβάντα, όταν αναγνωρισθούν ως πρότυπο κάποιου συγκεκριμένου γεγονότος ενεργοποιούν κάποιο μηχανισμό ή διαδικασία η οποία είναι υπεύθυνη για την προώθηση αυτού του γεγονότος στους κατάλληλους αποδέκτες. Αποτέλεσμα αυτής της διαδικασίας είναι ταχύτερες και αποτελεσματικότερες αποφάσεις και οι μικρότεροι χρόνοι απόκρισης.

Για την περαιτέρω ανάλυση και εμβάθυνση στις τεχνολογίες αυτές το κεφάλαιο υποδιαιρείται σε δύο μεγάλες ενότητες που αναφέρονται στην EDA και στην CEP αντίστοιχα. Πριν την ανάλυση αυτή είναι απαραίτητο να διευκρινίσουμε κάποιες βασικές έννοιες.

2.1 Βασικές έννοιες

Η πρωταρχική έννοια που πρέπει να αναλυθεί είναι αυτή του συμβάντος. Στο λεξικό, ως συμβάν ορίζεται κάτι το οποίο γίνεται ή θεωρείται ότι λαμβάνει χώρα. Στην επεξεργασία συμβάντων (event processing) η έννοια του συμβάντος έχει διττή σημασία. Η πρώτη αφορά στα περιστατικά του πραγματικού κόσμου, ενώ η δεύτερη στις αναπαραστάσεις αυτών των περιστατικών. Δηλαδή ένα περιστατικό είναι κάτι που συμβαίνει και ταυτόχρονα αυτό το περιστατικό αναπαρίσταται σε κάποια μορφή την οποία μπορούμε να επεξεργαστούμε. Αυτή η αναπαράσταση χαρακτηρίζεται σαν συμβάν, το οποίο γίνεται αντιληπτό όταν η κατάσταση του υπό εξέταση συστήματος μεταβάλλεται από μία κατάσταση η οποία έχει χαρακτηριστεί ως σταθερή.

Συνήθως αναφέρονται δύο τύποι συμβάντων. Τα «απλά συμβάντα» (simple events) και τα «σύνθετα συμβάντα» (complex events). Ένα απλό συμβάν είναι ένα μοναδικό συμβάν το οποίο δεν αναπαριστά άλλα συμβάντα και ούτε αποτελείται ή είναι αποτέλεσμα σύνθεσης άλλων. Ένα σύνθετο συμβάν αποτελεί είτε αφαίρεση είτε άθροισμα άλλων συμβάντων, σύνθετων ή απλών.

Παραδείγματα συμβάντων θα μπορούσαν να είναι:

- Μια εντολή παραγγελίας (καταγραφή όλης της δραστηριότητας για την παραγγελία)
- Ένα απλό ηλεκτρονικό μήνυμα επιβεβαίωσης
- Ένα μήνυμα το οποίο να καταγράφει τις ενδείξεις ενός RFID αισθητήρα
- Ένα μήνυμα που να αναφέρει τη δραστηριότητα μιας μετοχής

Υπάρχει στη βιβλιογραφία μια σειρά ονομάτων που αναφέρονται στην επεξεργασία συμβάντων. Μερικά από αυτά είναι η επεξεργασία σύνθετων συμβάντων (complex event processing), η επεξεργασία ροής συμβάντων (event stream processing) [3w18], η διαχείριση ροής δεδομένων (data stream management)[3w19] κ.α.

Όσον αφόρα την επεξεργασία συμβάντων υπάρχει μια γενικώς αποδεκτή κατηγοριοποίηση σε τρεις τύπους: απλή, ροής και σύνθετης. Στην αρχιτεκτονική καθοδηγούμενη από συμβάντα (Event – driven Architecture) που θα αναλυθεί σε επόμενες παραγράφους αναλύονται και οι τρεις τύποι επεξεργασίας.

2.1.1 Νέφη Συμβάντων και Ροές Συμβάντων

Τα συμβάντα, σύμφωνα με τη θεωρία επεξεργασίας συμβάντων, συνήθως υπάρχουν είτε σε «νέφη» (event clouds), είτε σε «ροές» (event streams). Ένα νέφος συμβάντων είναι ένα μερικώς διατεταγμένο σύνολο το οποίο μπορεί να είναι φραγμένο ή άπειρο. Η ροή συμβάντων είναι ένα γραμμικό πλήρως διατεταγμένο σύνολο. Στις περισσότερες περιπτώσεις η διάταξη βασίζεται στη χρονική διάσταση αλλά μπορεί να χρησιμοποιηθεί οποιοδήποτε χαρακτηριστικό προκειμένου τα συμβάντα να διαταχθούν σύμφωνα με αυτό. Ένα νέφος συμβάντων μπορεί να αποτελείται από διάφορες ροές συμβάντων συνδυασμένο και με άλλες πηγές δεδομένων. Θα μπορούσε πληρέστερα να οριστεί ως το σύνολο όλων των πηγών δεδομένων (datasource) μιας λειτουργίας, οι οποίες πηγές μπορεί να είναι εσωτερικές ή εξωτερικές της λειτουργία αυτής, και θα μπορούσε να περιλαμβάνει ροές δεδομένων, τροφοδοσίες δεδομένων (data feeds), αρχεία δεδομένων (όπως πχ xml αρχεία), βάσεις δεδομένων κ.α. Τα δεδομένα που παρέχονται μπορεί να είναι δεδομένα κρίσιμα και αναγκαία για την εκτέλεση της συγκεκριμένης λειτουργίας, αλλά μπορεί τα δεδομένα αυτά να περιέχουν και θόρυβο ή πληροφορίες που δεν έχουν σχέση με αυτή τη λειτουργία. Όλα τα παραπάνω δεδομένα μπορούν να θεωρηθούν συμβάντα, ιδιαίτερα αυτά τα οποία χαρακτηρίζονται και από κάποια χρονική καταγραφή (timestamp) της στιγμής που εμφανίστηκαν.

2.1.2 Η δομή ενός συμβάντος

Κάθε συμβάν αποτελείται από δύο μέρη, την κεφαλή (header) και το σώμα (body) [25]. Η κεφαλή περιέχει πληροφορίες όπως το όνομα του συμβάντος, το χρονικό προσδιορισμό του και τον τύπο του. Το σώμα του συμβάντος είναι το κομμάτι αυτό το οποίο περιγράφει το ίδιο το συμβάν. Δεν θα πρέπει να συγχέουμε το σώμα του συμβάντος με το πρότυπο ή τη λογική η οποία μπορεί να εφαρμοστεί ως αντίδραση σε αυτό καθαυτό το συμβάν.

Συμβάντα και Γεγονότα

Σε αυτό το σημείο πρέπει να αναφερθεί η έννοια των γεγονότων (facts) τα οποία διαφέρουν από τα συμβάντα (events) [5]. Σε ένα διάγραμμα καταστάσεων τα συμβάντα είναι οι μεταβάσεις και τα γεγονότα οι καταστάσεις. Τα γεγονότα έχουν θεωρητικά άπειρη διάρκεια ζωής. Διαγράφονται μόνο όταν ρητά αφαιρεθούν. Τα συμβάντα (events) από την άλλη έχουν συγκεκριμένο χρόνο ζωής τον οποίο καθορίζουν τα υποσυστήματα τα οποία τα παραλαμβάνουν προκειμένου να τα επεξεργαστούν και στη συνέχεια αυτομάτως αφαιρούνται.

2.1.3 Στάδια επεξεργασίας ενός συμβάντος (Event flow layers)

Σε μια αρχιτεκτονική καθοδηγούμενη από συμβάντα η επεξεργασία ενός συμβάντος στη γενική της μορφή αποτελείται από τέσσερα στάδια. Ξεκινά με την ανίχνευση ενός συμβάντος, την τεχνική αναπαράστασή του και καταλήγει σε μια σειρά ενεργειών ως αντίδραση προς το συμβάν αυτό. Ακολουθεί η αναλυτική παρουσίαση των τεσσάρων αυτών σταδίων [3w20].

1. Γεννήτρια Συμβάντων

Το πρώτο στάδιο επεξεργασίας είναι η γεννήτρια συμβάντων (event generator) η οποία ανιχνεύει ένα συμβάν και το αναπαριστά. Για παράδειγμα μια γεννήτρια συμβάντων μπορεί να είναι ένας αισθητήρας. Η μετατροπή των διαφορετικών δεδομένων από ένα σύνολο αισθητήρων σε μια μορφή στην οποία να μπορεί να γίνει επεξεργασία από ένα υπολογιστικό σύστημα αποτελεί το αντικείμενο σχεδίασης και υλοποίησης αυτού του πρώτου σταδίου.

2. Κανάλι Συμβάντων

Το κανάλι συμβάντων (event channel) αποτελεί το μηχανισμό μεταφοράς της πληροφορίας από τη γεννήτρια συμβάντων στον επεξεργαστή συμβάντων. Ο μηχανισμός αυτός θα μπορούσε να είναι μια TCP/IP σύνδεση ή ένα αρχείο XML. Πολλά κανάλια συμβάντων μπορούν να λειτουργούν ταυτόχρονα. Επειδή η επεξεργασία των συμβάντων σε μια EDA αρχιτεκτονική είναι ασύγχρονη και εκτελείται σε πραγματικό χρόνο, το κανάλι συμβάντων λειτουργεί ασύγχρονα. Συνήθως τα συμβάντα καταλήγουν σε μια ουρά αναμονής μέχρι να προχωρήσουν στο στάδιο της επεξεργασίας.

3. Μηχανή Επεξεργασίας Συμβάντων

Η μηχανή. επεξεργασίας συμβάντων (event processing engine) αναλαμβάνει να καθορίσει τις απαιτούμενες ενέργειες που πρέπει να ενεργοποιηθούν και να εκτελεστούν ως αποτέλεσμα αντίδρασης στην ανίχνευση του συμβάντος.

4. Εκκίνηση Ενεργειών

Στο τελικό στάδιο, η επεξεργασία των συμβάντων καταλήγει στην εκκίνηση μιας ή περισσοτέρων ενεργειών από τους παραλήπτες του συμβάντος. Οι παραλήπτες μπορεί να είναι υπολογιστικά συστήματα, βάσεις δεδομένων, αποθήκες δεδομένων, ευφυείς πράκτορες κ.α.. Το συμβάν μπορεί να προωθηθεί στους εκκινητές των διαδικασιών συνήθως μέσω του καναλιού επικοινωνίας το οποίο αναλαμβάνει και την μετατροπή του συμβάντος από το πρότυπο στο οποίο επεξεργάστηκε στο πρότυπο επεξεργασίας του συστήματος-παραλήπτη.

2.2 Αρχιτεκτονική Καθοδηγούμενη από Συμβάντα

Η αρχιτεκτονική καθοδηγούμενη από συμβάντα αποτελεί ένα πρότυπο αρχιτεκτονικής λογισμικού, η οποία ειδικεύεται στην παραγωγή, ανίχνευση και αντίδραση στα συμβάντα. Αποτελείται από τμήματα, ορισμένα από τα οποία ενεργοποιούνται και επικοινωνούν μεταξύ τους μέσω των συμβάντων. Ο παραπάνω ορισμός περιλαμβάνει και τα συστήματα αυτά τα οποία είναι και εν μέρει καθοδηγούμενα από συμβάντα. Μια «καθαρή» τέτοια αρχιτεκτονική θα απαιτούσε όλα τα μέρη της αρχιτεκτονικής να είναι καθοδηγούμενα από συμβάντα και η επικοινωνία μεταξύ τους να πραγματοποιείται από αυτά. Δεν υπάρχουν όμως στον πραγματικό κόσμο τέτοια συστήματα όπου όλη η επικοινωνία να γίνεται μέσω event-driven σχέσεων.

Αυτό το αρχιτεκτονικό πρότυπο μπορεί να εφαρμοστεί στη σχεδίαση και υλοποίηση εφαρμογών και συστημάτων τα οποία μεταδίδουν συμβάντα μεταξύ τμημάτων και υπηρεσιών λογισμικού που είναι χαλαρά συνδεδεμένα (loosely coupled) μεταξύ τους. Η αρχιτεκτονική καθοδηγούμενη από συμβάντα είναι καλά κατανεμημένη (well distributed) και εξαιρετικά χαλαρά συνδεδεμένη (extreme loose coupled). Η κατανομή της οφείλεται στο ότι ένα συμβάν μπορεί να είναι οτιδήποτε και να υπάρχει οπουδήποτε. Είναι επίσης και εξαιρετικά χαλαρά συνδεδεμένη επειδή το κάθε συμβάν δε γνωρίζει τις επιπτώσεις της λειτουργίας του. Για παράδειγμα, σε ένα σύστημα συναγερμού, το οποίο καταγράφει δεδομένα όποτε ανοίγει η μπροστινή πόρτα, η ίδια η πόρτα δεν γνωρίζει ότι εκείνη τη στιγμή το σύστημα εκτελεί αυτή τη λειτουργία, απλά αντιλαμβάνεται την αλλαγή της κατάστασής της από κλειστή σε ανοιχτή. Οι εφαρμογές και τα συστήματα αυτά υλοποιούνται με τέτοιο τρόπο ώστε να επιτυγχάνεται καλύτερη απόκριση, λόγω του γεγονότος ότι, από τη σχεδίασή τους, τα συστήματα αυτά είναι καταλληλότερα στο να λειτουργούν σε απρόβλεπτα και ασύγχρονα περιβάλλοντα.

Παραδείγματα τέτοιων συστημάτων τα οποία δέχονται χιλιάδες συμβάντα το λεπτό, αντιδρούν σε αυτά και επικοινωνούν μέσω αυτών είναι:

- Πληροφοριακά Συστήματα τραπεζών που ενεργούν σε νέφη συμβάντων
- Τραπεζικά συστήματα ανίχνευσης απάτης (πχ σε συναλλαγές με πιστωτικές κάρτες)
- Κατασκευή ολοκληρωμένων συστημάτων
- Συστήματα αισθητήρων
- Συστήματα αυτόματων συναλλαγών

2.2.1 Βασικά Χαρακτηριστικά της EDA

Για να μπορεί μια αρχιτεκτονική να χαρακτηριστεί ως καθοδηγούμενη από συμβάντα (EDA) θα πρέπει να πληροί τις εξής προϋποθέσεις:

1. Τα συμβάντα προωθούνται (pushed) στους κατάλληλους παραλήπτες

Τα συμβάντα αποστέλλονται από την γεννήτρια συμβάντων στο κατάλληλο υποσύστημα που θα αναλάβει την επεξεργασία του μέσω κάποιου ασύγχρονου σήματος ή μηνύματος. Η επικοινωνία δε γίνεται μέσω polling σε καθορισμένες χρονικές στιγμές αλλά από τη στιγμή που ανιχνεύεται ένα συμβάν μεταδίδεται απευθείας στους κατάλληλους επεξεργαστές συμβάντων.

2. Ασύγχρονη Λειτουργία

Η επεξεργασία των συμβάντων γίνεται κατά την άφιξή τους. Σε κάθε στάδιο της διαδικασίας εκτελείται άμεσα η επεξεργασία του συμβάντος. Αυτό συνεπάγεται ότι δεν υπάρχουν ούτε χρονικά καθορισμένες στιγμές που πραγματοποιείται η επεξεργασία ούτε μηχανισμοί μαζικής εκτέλεσης συγκεκριμένων ενεργειών. Ικανοποιείται με αυτή την έννοια η ασύγχρονη λειτουργία όλων των υποσυστημάτων που συνθέτουν την EDA αρχιτεκτονική.

3. Τα συμβάντα δεν καθορίζουν τις ενέργειες

Το ίδιο το συμβάν δεν πρέπει να προκαθορίζει τις ενέργειες που πρέπει να εκτελεστούν. Το υποσύστημα που παραλαμβάνει το συμβάν καθορίζει τις ενέργειες που πρέπει να εκτελεστούν. Στην αντίθετη περίπτωση θα μειωνόταν η χαλαρή σύνδεση μεταξύ της γεννήτριας συμβάντων και των ενεργοποιητών*. Η γεννήτρια συμβάντων ανιχνεύει και ενημερώνει για την ύπαρξη ενός συμβάντος. Η λογική, η συνθήκη ή συνθήκες, που πρέπει να ικανοποιηθούν είναι ενσωματωμένη στον επεξεργαστή συμβάντων. Αυτή η δυνατότητα διαχωρισμού λογικής από το ίδιο το συμβάν καθιστά την αρχιτεκτονική αυτή ευέλικτη αφού επιτρέπει την εισαγωγή ή αλλαγή ενεργοποιητών χωρίς της αλλαγή της γεννήτριας συμβάντων.

2.2.2 Κανόνες «Συμβάν-Συνθήκη-Ενέργεια»

Σε ένα έμπειρο σύστημα και συγκεκριμένα σε ένα σύστημα παραγωγής η βάση γνώσης αποτελείται από κανόνες συνάφειας, δηλαδή από κανόνες της μορφής:

If
<Συνθήκες>
Then
<Ενέργειες>

Στην EDA αρχιτεκτονική οι αντίστοιχοι κανόνες είναι της μορφής:

Event
<Συνθήκες>
Then
<Ενέργειες>

* Ως ενεργοποιητές, ορίζονται οι παραλήπτες των συμβάντων οι οποίοι εκκινούν τις κατάλληλες ενέργειες.

και στη διεθνή βιβλιογραφία αναφέρονται ως ECA Rules (Event-Condition-Action).

Οι κανόνες αυτής της μορφής αποτελούνται από τρία τμήματα:

- Το τμήμα του **Συμβάντος** που ορίζει το σήμα που πυροδοτεί την ενεργοποίηση του κανόνα
- Το τμήμα της **Συνθήκης**, το οποίο είναι ένας λογικός έλεγχος ο οποίος αν ικανοποιηθεί ή καθοριστεί ως αληθής τότε προκαλεί την εκτέλεση της Ενέργειας.
- Το τμήμα της **Ενέργειας** που μπορεί να είναι μια μεμονωμένη ή δέσμη ενεργειών, όπως η αποστολή ενός ηλεκτρονικού μηνύματος, η δημιουργία κάποιας αναφοράς κ.α.

Είναι σημαντικό να αναφέρουμε ότι σε μια EDA αρχιτεκτονική είναι αδιάφορο το ποιες είναι οι ενέργειες που πρέπει να εκτελεστούν. Το «λογικό» μέρος του συστήματος μπορεί να χρησιμοποιηθεί σε πολλές εφαρμογές μεταβάλλοντας μόνο τη δέσμη ενεργειών που πρέπει κάθε φορά να εκτελεστούν. Η δυνατότητα επαναχρησιμοποίησης του συστήματος για ένα πλήθος διαφορετικών εφαρμογών αποτελεί και βασικό χαρακτηριστικό των έμπειρων συστημάτων όπου η αρχιτεκτονική τους υποστηρίζει το διαχωρισμό Βάσης Γνώσης και κελύφους.

Η δομή αυτή χρησιμοποιήθηκε αρχικά στις ενεργές βάσεις δεδομένων (active databases). Σήμερα τα πιο εξελιγμένα συστήματα κανόνων (rule engines) που χρησιμοποιούνται σε EDA αρχιτεκτονικές χρησιμοποιούν πολλές παραλλαγές στη δομή των κανόνων ECA. Για παράδειγμα σε μια μηχανή κανόνων η συνθήκη θα μπορούσε να είναι κάποιος έλεγχος στη μνήμη και σαν ενέργειες θα μπορούσαν να είναι κάποιες λειτουργίες ενημέρωσης κάποιας εγγραφής σε μια βάση δεδομένων. Σε ένα σύστημα βάσης δεδομένων η συνθήκη θα μπορούσε να είναι ένα ερώτημα στη βάση και το αποτέλεσμα (εάν υπήρχαν αποτελέσματα στο ερώτημα) θα εκκινούσε την εκτέλεση μιας σειράς λειτουργιών στη βάση. Οι ενέργειες θα μπορούσαν να είναι εσωτερικές, δηλαδή να επηρεάζουν το ίδιο το σύστημα και όχι κάποιον εξωτερικά μηχανισμό, όπως την ενεργοποίηση ενός συναγερμού, και αφορούν κυρίως το σύστημα της βάσης δεδομένων ή να είναι εξωτερικές όπως στο σύστημα κανόνων όπου το σήμα από έναν αισθητήρα που καταγράφεται στη μνήμη μπορεί να εκκινήσει μια διαδικασία αποστολής email ή κάποιον άλλο ECA κανόνα.

Στους κανόνες αυτής της μορφής τόσο το τμήμα του συμβάντος όσο και αυτό της ενέργειας μπορεί να αποτελείται από πολύπλοκα και σύνθετα πρότυπα (patterns) τα οποία θα πρέπει να ικανοποιηθούν για την «πυροδότηση» της ενέργειας. Τα πρότυπα ανιχνεύονται μεταξύ δεδομένων τα οποία μπορεί να προέρχονται από πολλές πηγές και να είναι σε μορφή ροών, νεφών δεδομένων και το τμήμα τη συνθήκης πρέπει να αντιστοιχίσει τα δεδομένα αυτά με τα πρότυπα τα οποία εμπεριέχονται στη Βάση Γνώσης.

Εννοιολογικά, τα συμβάντα και οι συνθήκες αποτελούνται από διαφορετικά δεδομένα με διαφορετικές σημασιολογίες. Για αυτό είναι αναγκαίο να διαχωριστούν η έννοια του συμβάντος (event) με την έννοια του γεγονότος (fact) κατά τον ορισμό των ECA κανόνων. Έστω μια εφαρμογή η οποία συμπεριφέρεται ως μηχανή καταστάσεων, τα συμβάντα αναπαριστούν τις μεταβάσεις. Όταν ανιχνευθεί ένα συμβάν τότε εκτελείται η κατάλληλη μετάβαση. Τότε το συμβάν μπορεί να αποβληθεί, δηλαδή να αφαιρεθεί από τη μνήμη. Τα γεγονότα αναπαριστούν καταστάσεις. Οι καταστάσεις παραμένουν ενεργές στη μνήμη μέχρι ρητά να αφαιρεθούν. Σε κάθε περίπτωση εφόσον τα συμβάντα είναι παροδικά δηλαδή έχουν συγκεκριμένο χρόνο ζωής, θα πρέπει να αφαιρούνται όταν παύει η χρησιμότητά τους. Ακολουθεί πίνακας που συνοψίζει τις σημαντικότερες διαφορές στη σημασιολογία και στη συμπεριφορά μεταξύ συμβάντων και γεγονότων. Τα συμβάντα είναι παροδικά και αναλώνονται μετά την παράδοση τους στους κατάλληλους ενεργοποιητές. Τα γεγονότα από την άλλη είναι μόνιμα και θα πρέπει ρητά να αφαιρεθούν. Επίσης τα συμβάντα δεν μεταβάλλονται ενώ τα γεγονότα μπορούν να αλλάξουν.

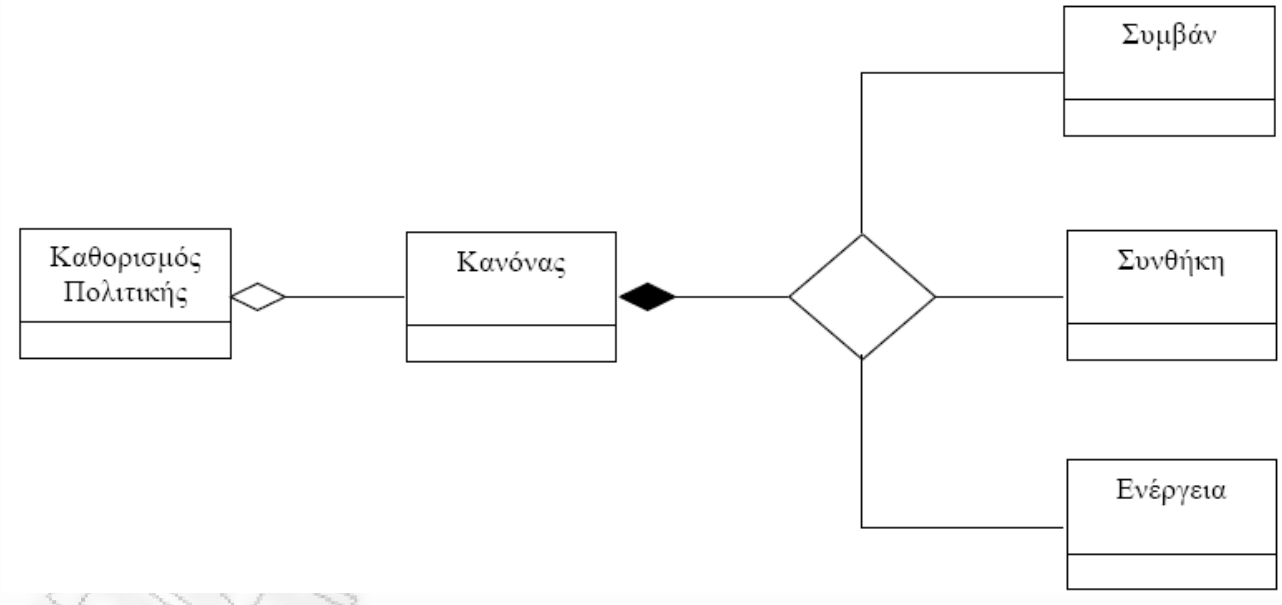
Μια κατάλληλη αντιστοίχηση μεταξύ συμβάντων και γεγονότων είναι με την προφορική και γραπτή ομιλία αντίστοιχα. Η ομιλία σε μορφή κειμένου μπορεί να μεταβληθεί όταν διορθωθεί το κείμενο, είναι διαθέσιμη σε όλους όσους έχουν πρόσβαση στο κείμενο και το κείμενο δεν μπορεί να λήξει ή πάψει να υπάρχει χωρίς τη μεσολάβηση κάποιου. Η προφορική ομιλία δεν μπορεί να

διορθωθεί και είναι διαθέσιμη μόνο σε αυτούς που εκείνη τη χρονική στιγμή την ακούνε. Μετά τη λήξη της ομιλίας παύει να υπάρχει.

Συμβάντα (Events)	**Γεγονότα (Facts)**
Παροδικά	Μόνιμα
Συνήθως δεν μεταβάλλονται	Μεταβάλλονται
Αναλογία:	
Μετάβαση	Κατάσταση
Προφορικός λόγος	Γραπτός λόγος
Συμπεριφορά:	
Λαμβάνονται μέσω προώθησης πληροφοριών (push)	Ανακτώνται μέσω αιτήματος (pull, πχ μέσω polling)

Πίνακας 1: Διαφορές μεταξύ Συμβάντων και Γεγονότων

Οι κανόνες μπορούν να εξελιχθούν σε πολιτικές, δηλαδή σε σύνολα επιχειρησιακών κανόνων, οι οποίες να καθορίζουν τους κανόνες που πρέπει να εφαρμοστούν σε ένα τομέα προβλημάτων που καλείται να αντιμετωπίσει η διαχείριση συμβάντων. Ακολουθεί η UML αναπαράσταση των ECA κανόνων [11] συμπεριλαμβανομένου και της μετεξέλιξής τους σε πολιτική:



Σχήμα 1: Αναπαράσταση σε UML των ECA κανόνων

Πλεονεκτήματα συστημάτων βασισμένων σε ECA κανόνες

Η υλοποίηση ενός συστήματος με ECA κανόνες περιλαμβάνει πλεονεκτήματα, από καθαρά τεχνική άποψη, σε σχέση με συστήματα των οποίων η υλοποίηση πραγματοποιείται με κώδικα. Καταρχήν το σύνολο της λειτουργικότητας ενός συστήματος περιέχεται εξολοκλήρου στη βάση γνώσης και όχι διεσπαρμένη σε διάφορα υποσυστήματα. Εξασφαλίζεται έτσι η εύκολη επεκτασιμότητα και συντηρησιμότητα ενός συστήματος. Δεύτερον, οι ECA κανόνες ακολουθούν υψηλού επιπέδου δηλωτική σύνταξη η οποία επιτρέπει τεχνικές βελτιστοποίησης που δεν

μπορούν να εφαρμοστούν σε κώδικα. Τέλος οι ECA κανόνες, αποτελούν μηχανισμούς οι οποίοι μπορούν να μοντελοποιήσουν γενικευμένα προβλήματα σε αντίθεση με την υλοποίηση μέσω κώδικα ο οποίος εξειδικεύεται στην επίλυση ενός αυστηρά καθορισμένου προβλήματος.

2.2.3 Συστατικά στοιχεία μιας EDA Αρχιτεκτονικής

Η αρχιτεκτονική καθοδηγούμενη από συμβάντα αποτελείται από συστήματα όπως:

- Αισθητήρες
- Επεξεργαστές συμβάντων οι οποίοι συγκεντρώνουν δεδομένα από πολλαπλούς αισθητήρες και βάσεις δεδομένων
- Συστήματα που ανταποκρίνονται και εκκινούν τις κατάλληλες ενέργειες από τη στιγμή που ανιχνεύεται ένα συμβάν
- Επικοινωνιακές συνδέσεις οι οποίες μεταδίδουν πληροφορίες μεταξύ συστημάτων
- Υπηρεσίες διαχείρισης οι οποίες βοηθούν στον καθορισμό και στην επεξεργασία λειτουργιών εφαρμογών καθοδηγούμενων από συμβάντα.

Στη συνέχεια γίνεται περαιτέρω ανάλυση των παραπάνω συστημάτων.

Αισθητήρες

Οι αισθητήρες αποτελούν συστήματα καταγραφής δεδομένων από το εξωτερικό περιβάλλον και από το εσωτερικό της επιχείρησης. Τα δεδομένα αυτά αναλύονται προκειμένου να ανιχνευθούν πρότυπα τα οποία μπορούν να ορισθούν ως συμβάντα. Υπάρχουν 2 τρόποι απόκτησης δεδομένων, είτε μέσω διαδικασίας προώθησης (push) των δεδομένων σε αυτά τα συστήματα, είτε τα συστήματα να «ανασύρουν» (pull) δεδομένα μέσω polling. Για παράδειγμα οι διακυμάνσεις μιας μετοχής μπορεί να τροφοδοτούν το σύστημα, χωρίς να απαιτείται από αυτό να αιτείται στοιχεία κάθε φορά που αλλάζει η τιμή μιας μετοχής. Η άλλη περίπτωση είναι σε συγκεκριμένα χρονικά διαστήματα να αιτούνται τα συστήματα την αποστολή δεδομένων. Στις EDA αρχιτεκτονικές προτείνεται η πρώτη μέθοδος διότι η αρχιτεκτονική υποστηρίζει ασύγχρονη επικοινωνία επιτυγχάνοντας στιγμιαίους χρόνους απόκρισης.

Επεξεργαστές Συμβάντων

Σκοπός αυτού του υποσυστήματος είναι η συγκέντρωση πληροφοριών από πολλούς αισθητήρες και η ολοκλήρωση αυτών των πληροφοριών με δεδομένα και άλλων συστημάτων όπως βάσεις δεδομένων, αποθήκες δεδομένων. Στη συνέχεια από τον όγκο όλων αυτών των πληροφοριών ανιχνεύονται πρότυπα τα οποία «πυροδοτούν» τις κατάλληλες ενέργειες. Υπάρχουν και περιπτώσεις στις οποίες οι επεξεργαστές συμβάντων μπορεί να αιτηθούν επιπλέον πληροφοριών από τους αισθητήρες δημιουργώντας έτσι ένα κλειστό κύκλωμα ανατροφοδότησης μεταξύ των δύο υποσυστημάτων.

Ενεργοποιητές

Η επεξεργασία των πληροφοριών καταλήγει στην εκκίνηση μιας ενέργειας. Κλείνει έτσι και οι κύκλος Event-Condition-Action των οποίο και θα αναλύσουμε σε επόμενη παράγραφο. Μια τυπική ενέργεια είναι η εκκίνηση μιας επιχειρησιακής διαδικασίας. Άλλα παραδείγματα ενεργειών θα μπορούσαν να είναι η αποστολή ενός email, η ενεργοποίηση κάποιου συναγερμού και η ενημέρωση μιας εγγραφής μιας βάσης δεδομένων. Υπάρχει η πιθανότητα η εκτέλεση μιας διαδικασίας να αποτελέσει την αιτία για εκτέλεση μιας σειράς και άλλων διαδικασιών.

Επικοινωνιακές Συνδέσεις

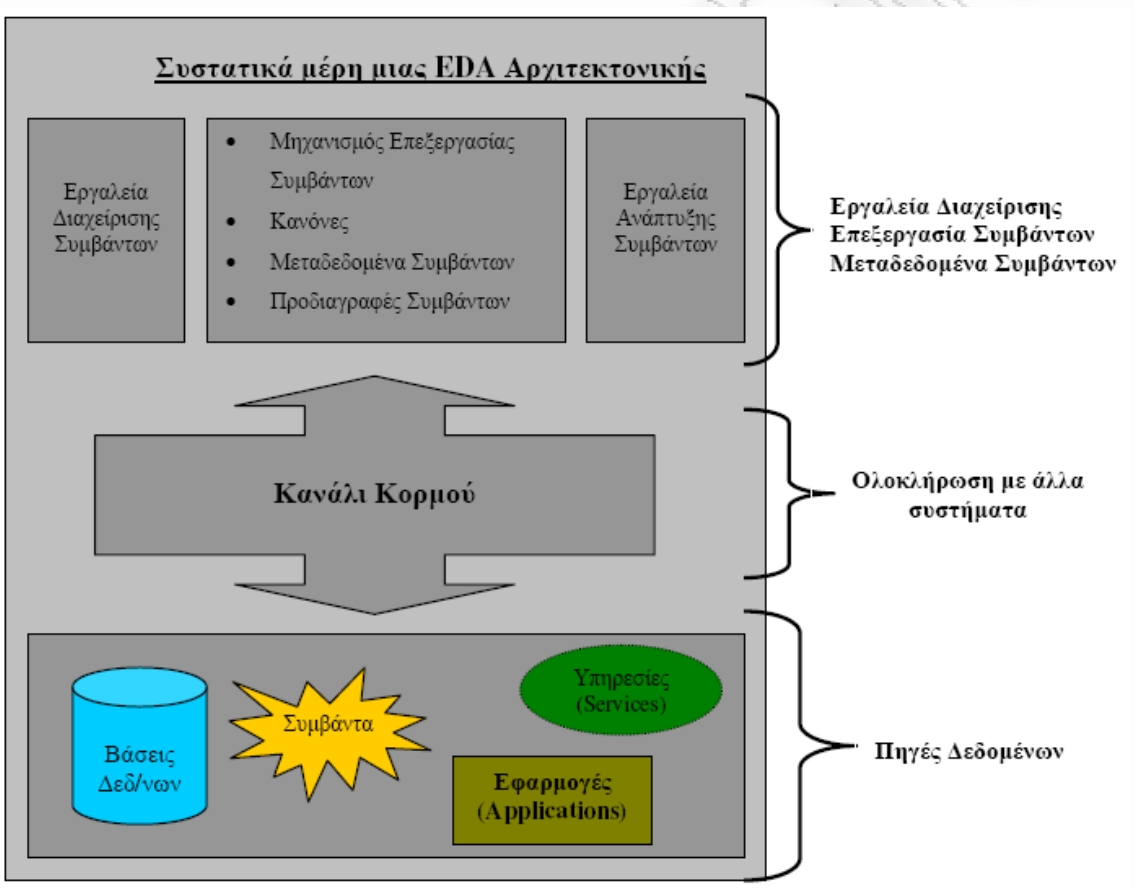
Ο τρόπος με των οποίο επικοινωνούν και διασυνδέονται το υποσυστήματα τα οποία προαναφέρθηκαν. Συνήθως οι επικοινωνίες στην EDA αρχιτεκτονική αναπαρίστανται ως ένα κεντρικό δίαυλο μέσω του οποίου διοχετεύεται το σύνολο της πληροφορίας που διακινείται ή ως το επίπεδο επικοινωνίας σε μια αναπαράσταση της αρχιτεκτονικής σε διάφορα επίπεδα. Επειδή ο όγκος της πληροφορίας από τους αισθητήρες, βάσεις δεδομένων είναι μεγάλος και περιέχει θόρυβο η τοπολογία των υποσυστημάτων και η χωρητικότητα του δίαυλου είναι κρίσιμα χαρακτηριστικά για τη συνολική απόδοση του συστήματος

Υπηρεσίες Διαχείρισης

Δεν αποτελεί βασικό στοιχείο της αρχιτεκτονικής αλλά πολύ περισσότερο ένας υποστηρικτικός μηχανισμός. Στις λειτουργίες που πρέπει να επιτελεί περιλαμβάνονται η παρακολούθηση της ορθής λειτουργίας των άλλων υποσυστημάτων, σε περιπτώσεις ανασχεδιασμού του συστήματος την εκτέλεση προσομοιώσεων για έλεγχο του βαθμού προσαρμογής του συστήματος στο νέο περιβάλλον.

2.2.4 Σχηματική Αναπαράσταση της EDA Αρχιτεκτονικής

Από τη μέχρι τώρα ανάλυση έχουμε αναφερθεί στα διάφορα τμήματα από τα οποία αποτελείται μια αρχιτεκτονική καθοδηγούμενη από τα γεγονότα. Στο σχήμα που ακολουθεί προβάλλονται τα τμήματα αυτά.



Σχήμα 2: Συστατικά μέρη EDA Αρχιτεκτονικής

Μπορούμε να διαχωρίσουμε τα συστατικά μέρη σε πέντε βασικές κατηγορίες [25]:

1. Μεταδεδομένα Συμβάντων

Μια καλή αρχιτεκτονική χαρακτηρίζεται από μια αυστηρή αρχιτεκτονική όσον αφορά στα Μεταδεδομένα. Τα μεταδεδομένα αυτά περιλαμβάνουν τις προδιαγραφές των γεγονότων και τους κανόνες επεξεργασίας των γεγονότων. Οι προδιαγραφές αυτές πρέπει να είναι διαθέσιμες στις γεννήτριες γεγονότων, στους event format transformers, στους μηχανισμούς επεξεργασίας γεγονότων και στους subscribers. Αυτή τη στιγμή δυστυχώς δεν έχει αναπτυχθεί κάποιο

πρότυπο για τον ορισμό ενός γεγονότος καθώς επίσης και για τη σημειολογία της διαδικασίας επεξεργασίας, αλλά είναι κάτι που θα συμβεί στο προσεχές μέλλον.

2. Επεξεργασία Συμβάντων

Ο πυρήνας της διαδικασίας επεξεργασίας γεγονότων είναι ο μηχανισμός κανόνων και τα συμβάντα που έχουν καταχωρηθεί. Τα συμβάντα που είναι αποθηκευμένα είναι συνήθως συνεχή και διατηρούνται για έλεγχο και ανάλυση τάσεων.

3. Εργαλεία Διαχείρισης

Τα εργαλεία ανάπτυξης για την επεξεργασία των γεγονότων είναι απαραίτητα για τον καθορισμό των προδιαγραφών των συμβάντων, των κανόνων επεξεργασίας και τον έλεγχο των συστημάτων που παραλαμβάνουν τα συμβάντα. Τα εργαλεία διαχείρισης γεγονότων παρέχουν δυνατότητες διαχείρισης και ελέγχου της υποδομής επεξεργασίας γεγονότων, την παρακολούθηση των event flows, και την θέαση στην παραγωγή συμβάντων και στα στατιστικά επεξεργασίας.

4. Ολοκλήρωση με άλλα συστήματα

Η διασύνδεση των υποσυστημάτων της αρχιτεκτονικής υλοποιείται μέσω ενός βασικού καναλιού (backbone) ή καναλιού κορμού εάν χρησιμοποιήσουμε ορολογία δικτύων. Το κανάλι αυτό πρέπει να υποστηρίζει την προεργασία των συμβάντων (φίλτρα, δρομολογήσεις), την κλήση υπηρεσιών, την κλήση επιχειρησιακών διαδικασιών και την πρόσβαση σε βάσεις δεδομένων και άλλες πηγές πληροφοριών της επιχείρησης.

5. Πηγές Δεδομένων

Αποτελούν τους πόρους της επιχείρησης και περιλαμβάνουν τις εφαρμογές, τις επιχειρησιακές διαδικασίες, τις αποθήκες δεδομένων και τους μηχανισμούς παραγωγής και επεξεργασίας των λειτουργιών καθοδηγούμενων από συμβάντα. Η τοπολογία όλων αυτών των υποσυστημάτων διαφέρουν από επιχείρηση σε επιχείρηση ανάλογα με τη ροή των συμβάντων, το βασικό κανάλι (κορμό), τον όγκο των δεδομένων, τις υπηρεσίες που υποστηρίζει.

2.3 EDA Αρχιτεκτονική και Επεξεργασία Συμβάντων

Η επεξεργασία συμβάντων διακρίνεται σε τρεις μορφές [25]:

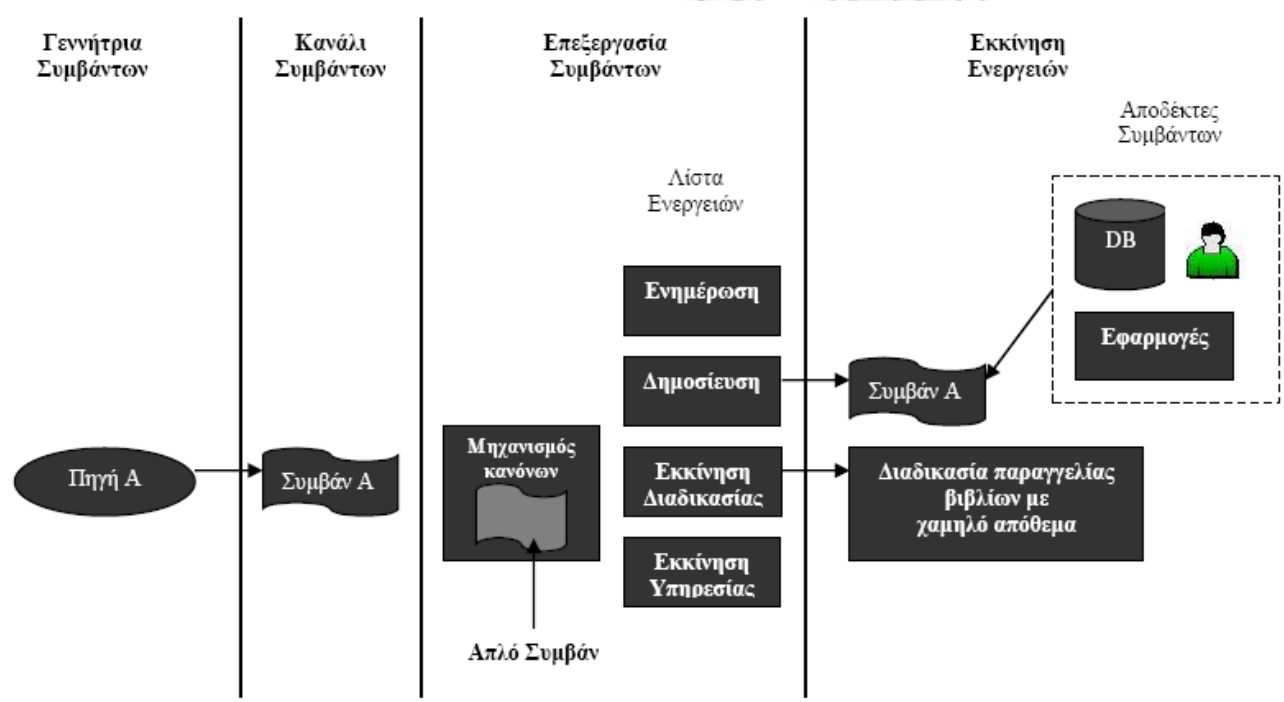
- **Επεξεργασία Απλών Συμβάντων (Simple Event Processing)**
- Συμβάντα τα οποία είναι μεμονωμένα και δεν αποτελούν τμήμα ή σύνθεση από άλλα συμβάντα
- **Επεξεργασία Ροής Συμβάντων (Stream Event Processing)**
- Συμβάντα τα οποία είναι πλήρως χρονικά διατεταγμενα
- **Επεξεργασία Σύνθετων Συμβάντων (Complex Event Processing)**
- Συμβάντα τα οποία είναι σύνθεση απλών συμβάντων ή/και ροών συμβάντων τα οποία είναι μερικώς διατεταγμένα.

Η EDA αρχιτεκτονική διαμορφώνεται ανάλογα με τον τύπο επεξεργασίας. Η ανάλυση μέσω περιπτώσεων χρήσης της κάθε μορφής επεξεργασίας θα οδηγήσει στην καλύτερη κατανόηση της EDA αρχιτεκτονικής και των δυνατοτήτων της. Σε κάθε μελέτη περίπτωσης εξασφαλίζονται όλα τα χαρακτηριστικά που καθορίζουν μια EDA αρχιτεκτονική και τονίζονται οι δυνατότητες της αρχιτεκτονικής αυτής. Η έμφυτη ευελιξία σε κάθε επίπεδο - τα συμβάντα που δημιουργούνται, το πώς αυτά επεξεργάζονται και ποιος τα παραλαμβάνει - είναι η πραγματική δύναμη που εξασφαλίζει η διαχείριση μέσω συμβάντων. Η ροή των εικόνων είναι από αριστερά προς τα δεξιά, ξεκινώντας από τις γεννήτριες των συμβάντων.

2.3.1 EDA στην επεξεργασία Απλών Συμβάντων

Στο Σχήμα 3 παρουσιάζεται η διαδικασία επεξεργασίας ενός απλού συμβάντος που αφορά στη βελτιστοποίηση του online καταλόγου παραγγελιών ενός βιβλιοπωλείου. Όταν ένας πελάτης κάνει μια παραγγελία, το βιβλιοπωλείο αμέσως δεσμεύει τα βιβλία από τον κατάλογο μέσω μιας

υπηρεσίας δέσμευσης βιβλίων καταλόγου. Η υπηρεσία δέσμευσης βιβλίων καταλόγου αλλάζει την κατάσταση των βιβλίων που επιλέχθηκαν από «διαθέσιμο» σε «δεσμευμένο» και στη συνέχεια ελέγχει τον υπόλοιπο διαθέσιμο υπόλοιπο αυτών των βιβλίων σε σχέση με τα βέλτιστα κατώτατα όρια που έχουν καθοριστεί. Αν το απόθεμα πέσει κάτω από τα κατώτατα όρια η υπηρεσία δέσμευσης βιβλίων του καταλόγου παράγει ένα συμβάν ενημέρωσης για απόθεμα μικρότερο του κατώτατου ορίου. Το συγκεκριμένο συμβάν εμφανίζεται στο Σχήμα 3 ως «Συμβάν Α»,. Το συμβάν χαμηλού αποθέματος μεταφέρεται μέσω του καναλιού συμβάντων και παραλαμβάνεται από την μηχανή επεξεργασίας συμβάντων. Στη συγκεκριμένη περίπτωση που το συμβάν είναι απλό, μπορεί να προκαλέσει την εκκίνηση μια ενέργειας εκ νέου παραγγελιάς βιβλίων που έχουν χαμηλό απόθεμα. Η ενέργεια αυτή μπορεί να εκτελεστεί αυτόματα, δηλαδή να εκκινηθεί μια διαδικασία παραγγελίας ή να απαιτείται να μεσολαβήσει πριν την παραγγελία ο ανθρώπινος παράγοντας, οπότε μπορεί απλά να αποσταλεί ένα ενημερωτικό email πχ στον υπεύθυνο παραγγελιών και αυτός να εκτελέσει τη διαδικασία παραγγελίας.



Σχήμα 3: Επεξεργασία Απλών Συμβάντων

2.3.2 EDA στην επεξεργασία Ροής Συμβάντων

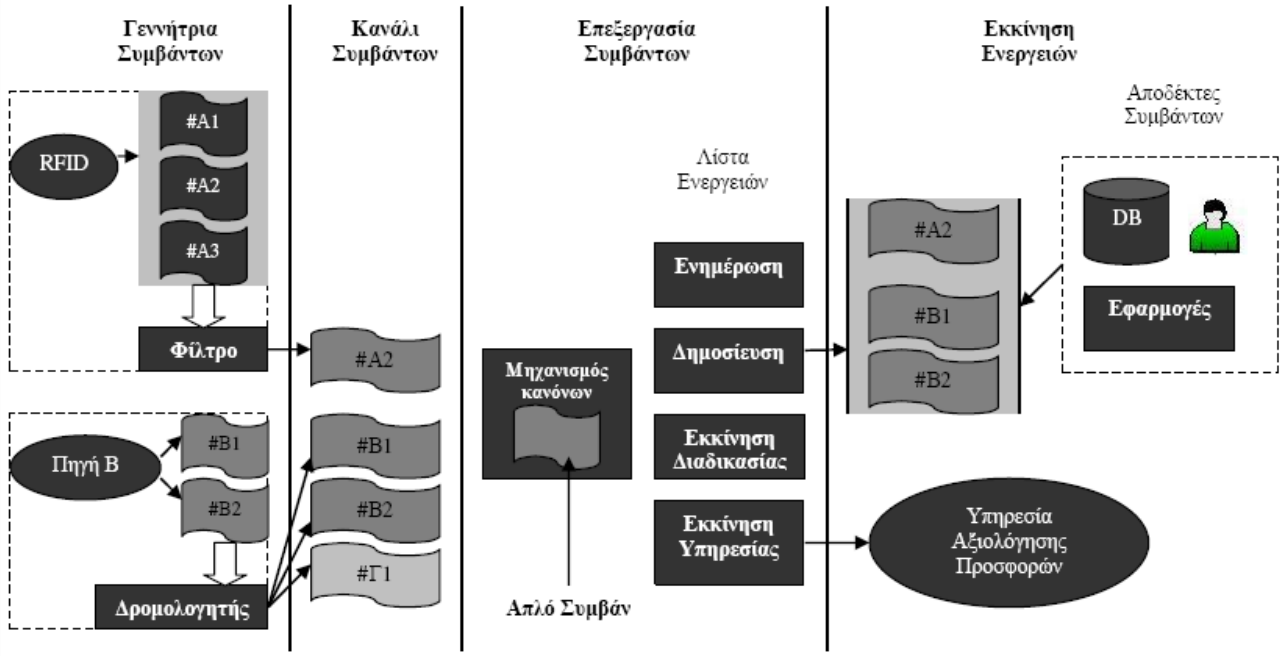
Στην επεξεργασία ροών συμβάντων υπάρχει μια συνεχής τροφοδοσία πληροφοριών. Η επεξεργασία μπορεί να αφορά μια ή περισσότερες ταυτόχρονες ροές. Στην επόμενη εικόνα παρουσιάζονται τρεις ροές επεξεργασίας συμβάντων ενός καταστήματος πώλησης ηλεκτρονικών.

Στην πρώτη ροή, Α, ένας RFID αισθητήρας εκπέμπει σήμα κάθε φορά που ένα προϊόν εξέρχεται από την αποθήκη. Ο πωλητής του καταστήματος θέλει να ενημερώνεται μόνο όταν προϊόντα με υψηλή τιμή εξέρχονται από αυτή την αποθήκη. Προκειμένου να ικανοποιείται αυτή η απαίτηση του πωλητή, μεσολαβεί ένα τοπικό φίλτρο συμβάντων το οποίο ελέγχει εάν η τιμή του προϊόντος ξεπερνά την ανώτατη τιμή που έχει καθοριστεί. Εάν η τιμή αυτή είναι πχ τα 1.000 € τότε αποκλείει όλα τα προϊόντα που έχουν χαμηλότερη τιμή. Έστω ότι το συμβάν «# Α2» έχει τιμή 2.000 € τότε το φίλτρο επιτρέπει να προωθηθεί στο κανάλι επικοινωνίας προκείμενου να συνεχιστεί η διαδικασία επεξεργασίας του. Στο τέλος της επεξεργασίας το συμβάν

«δημοσιοποιείται» και το παραλαμβάνει η βάση δεδομένων προκειμένου να ενημερωθεί για την νέα κατάσταση του αποθέματος.

Η τρίτη και η τέταρτη ροή προέρχονται από την εφαρμογή καταχώρησης παραγγελιών. Η παραγγελία κάθε πελάτη παράγει και ένα συμβάν καταχώρησης «Απλής Παραγγελίας» το οποίο μέσω του καναλιού επικοινωνίας το παραλαμβάνει ο επεξεργαστής συμβάντων. Η ενεργεία που εκκινείται προσθέτει μια εγγραφή νέας παραγγελίας σε μια αποθήκη δεδομένων. Το συμβάν αυτό αποτυπώνεται στο σχήμα σαν συμβάν «# Β1» και «# Β2» (για πχ 2 παραγγελίες). Στην περίπτωση που το σύνολο του ποσού της παραγγελίας ξεπερνά ένα ποσό τότε παράγεται ένα διαφορετικό συμβάν «Σημαντικής Παραγγελίας». Το συμβάν αυτό μετά την επεξεργασία του καταλήγει στην ενεργοποιήσει ενός επιχειρησιακού κανόνα προσφοράς επιπλέον έκπτωσης ως ανταπόδοση στην υψηλή παραγγελία. Στο σχήμα το συμβάν αυτό αποτυπώνεται ως «# Γ1». Η επιχειρησιακή ενέργεια που «πυροδοτείται» από την ύπαρξη αυτού του συμβάντος είναι η εκκίνηση της υπηρεσίας Αξιολόγησης Προσφορών (Offer Estimation Service). Η όλη διαδικασία μπορεί να αποτελέσει ταυτόχρονα και ένα παράδειγμα Event – Driven SOA.

Ο τοπικός δρομολογητής εξασφαλίζει την ανεξαρτησία μεταξύ πηγής και επεξεργαστή συμβάντος. Ο διαχωρισμός αυτός είναι σημαντικός γιατί σε μια EDA αρχιτεκτονική μπορούμε να αλλάξουμε την πηγή των συμβάντων διατηρώντας όμως τη λογική επεξεργασία τους. Η πηγή του συμβάντος ή γεννήτρια συμβάντων ανιχνεύει την ύπαρξη κάποιου γεγονότος χωρίς να απαιτείται η περαιτέρω επεξεργασία του. Ο χειρισμός αυτού του συμβάντος αυτού είναι στη δικαιοδοσία του μηχανισμού επεξεργασίας. Άρα απαιτείται η ύπαρξη του δρομολογητή που θα κατευθύνει το συμβάν στον κατάλληλο αποδέκτη. Είναι λοιπόν στη δικαιοδοσία του μηχανισμού επεξεργασίας των συμβάντων το που θα δρομολογηθούν αυτά και ο δρομολογητής είναι τμήμα της αρχιτεκτονικής επεξεργασίας ανεξάρτητος ως προς τη λειτουργία της πηγής.



Σχήμα 4: Επεξεργασία Ροής Συμβάντων

2.3.3 EDA στην επεξεργασία Σύνθετων Συμβάντων

Στο Σχήμα 5 παρουσιάζεται ένα σύνθετο σύστημα δύο διαδικασιών επεξεργασίας σύνθετων συμβάντων για τον ίδιο κατάστημα πώλησης ηλεκτρονικών. Στην πρώτη διαδικασία, πάνω αριστερά στο Σχήμα, μια ηλεκτρονική πύλη για παραγγελίες business-to-business παράγει

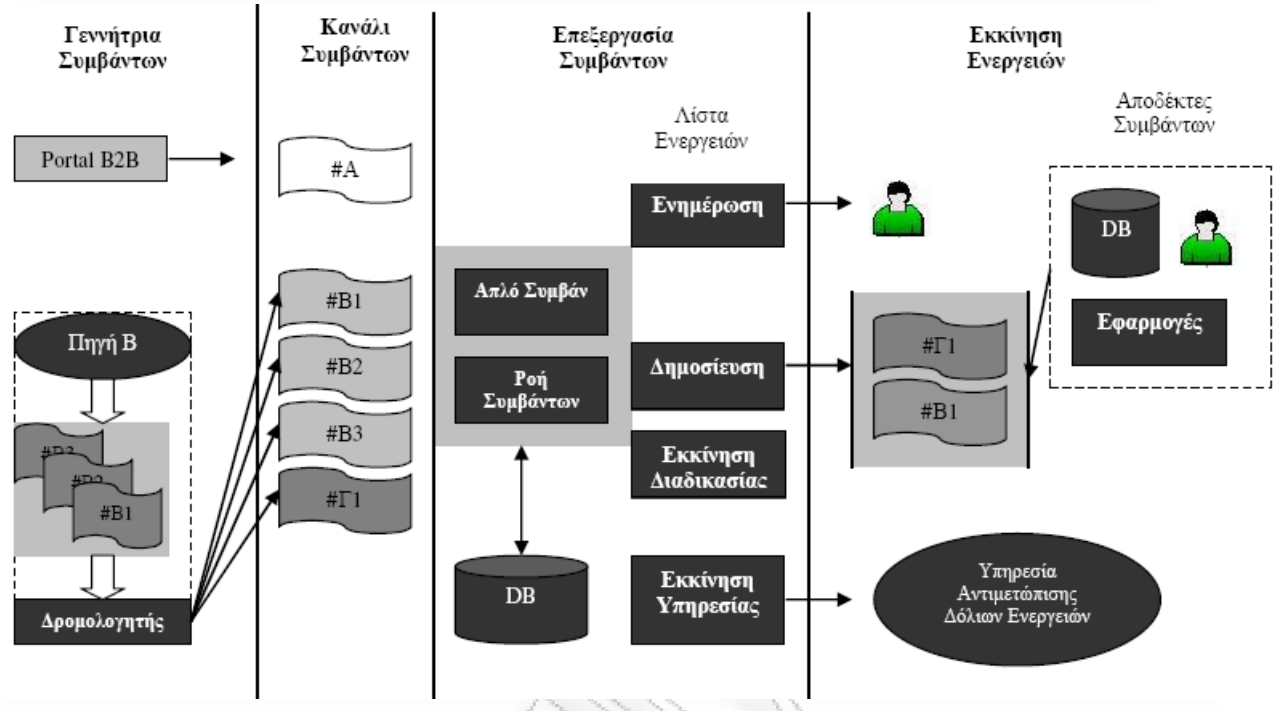
συμβάντα κάθε 15 λεπτά. Τα συμβάντα αυτά ενημερώνουν τα ηλεκτρονικά συστήματα ελέγχου της υποδομής υπολογιστικών συστημάτων του καταστήματος ότι η ηλεκτρονική πύλη λειτουργεί κανονικά. Σε περίπτωση που μετά τα 15 λεπτά δεν ληφθεί το αντίστοιχο σήμα-συμβάν υποδηλώνεται πιθανή βλάβη. Η λειτουργία αυτή είναι κρίσιμη διότι θα αποθαρρύνει πιθανούς πελάτες και θα μειώσει την ανταγωνιστικότητα της επιχείρησης.

Η μηχανή επεξεργασίας σύνθετων συμβάντων ανιχνεύει το χρονικό σημείο του τελευταίου συμβάντος που έχει λάβει μέσω του Συστήματος Ειδοποίησης. Αν περάσουν 15 λεπτά, τότε ξεκινούν ενέργειες επεξεργασίας συμβάντων που σχετίζονται με τη μη άφιξη. Σε αυτή την περίπτωση, κοινοποιείται αμέσως ο υπεύθυνος παραγγελιών και δημιουργείται ένα συμβάν Σφάλματος στην Πύλη των Παραγγελιών B2B. Αυτό το καινούργιο συμβάν τοποθετείται στο κανάλι συμβάντων. Όταν το συμβάν αυτό παραληφθεί από τη μηχανή συμβάντων, τότε ενημερώνονται τα υποσυστήματα για την ύπαρξη τεχνικού προβλήματος και το σύστημα διαχείρισης σφαλμάτων της εταιρίας αναλαμβάνει την επίλυση του πιθανού προβλήματος.

Οι δεύτερη ροή παρουσιάζονται δυο παραλλαγές ανίχνευσης απάτης (Fraud Detection). Και οι δυο διαδικασίες δημιουργούνται από την εφαρμογή που διαχειρίζεται τα σημεία πώλησης. Όπως και στο παράδειγμα της προηγούμενης παραγράφου κάθε παραγγελία πελάτη δημιουργεί ένα συμβάν. Εάν η πώληση αυτή είναι μικρού ποσού το παράγεται το συμβάν «Απλή Παραγγελία», εάν το ποσό της παραγγελίας υπερβαίνει ένα καθορισμένο ποσό τότε παράγεται το συμβάν «Σημαντική Παραγγελία». Αυτά τα συμβάντα αξιολογούνται από έναν τοπικό δρομολογητή συμβάντων, όπως και στο προηγούμενο παράδειγμα, ο οποίος διαχωρίζει τα συμβάντα σε Απλά και Σημαντικά και τα τοποθετεί ανάλογα στο κανάλι συμβάντων.

Στη πρώτη διαδικασία ανίχνευσης απάτης, η μηχανή σύνθετων συμβάντων ελέγχει την ύπαρξη πολλαπλών συναλλαγών (συνηθισμένα συμβάντα πώλησης καταστήματος) από τον ίδιο πελάτη (πιστωτική κάρτα), σε σύντομο χρονικό διάστημα (10 λεπτά), σε διαφορετικές τοποθεσίες οι οποίες απέχουν μεταξύ τους με μεγάλη απόσταση (20 μίλια). Αν πληρούνται όλες αυτές οι προϋποθέσεις, τότε εκκινείτε η υπηρεσία αντιμετώπισης δόλιων ενεργειών.

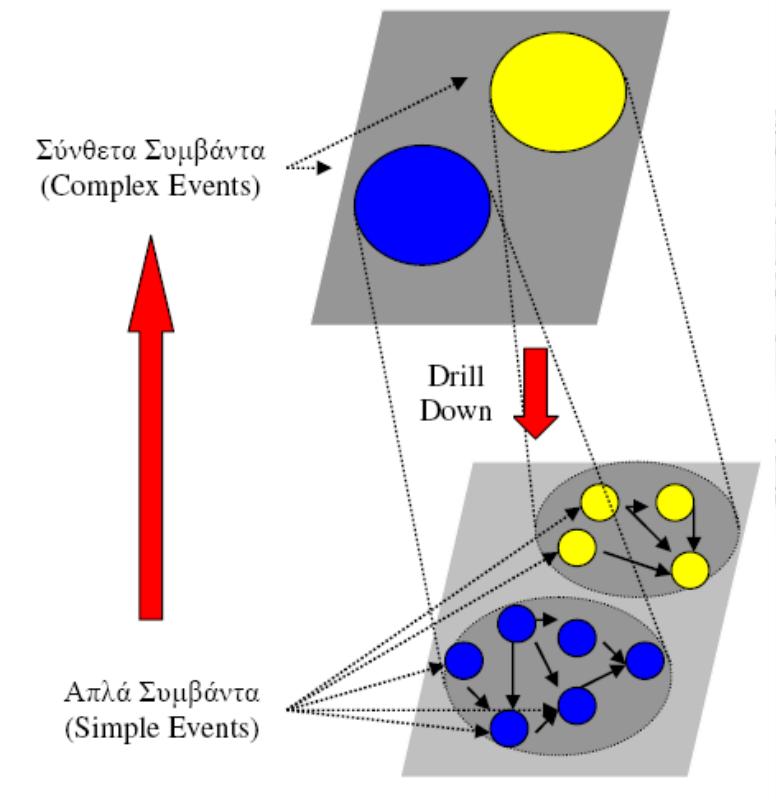
Στη δεύτερη διαδικασία ανίχνευσης απάτης, με τη λήψη ενός συμβάντος «Σημαντικής Πώλησης» (# Γ1), η μηχανή σύνθετων συμβάντων εκτελεί μια αναζήτηση στις προηγούμενες αγορές του πελάτη (στη βάση δεδομένων «DB» προκειμένου να διαπιστωθεί αν αυτή η αγορά πρέπει να θεωρηθεί ως ύποπτη. Αν το παρόν ποσό αγοράς αποκλίνει πάνω από το 50% σε σχέση με τη μεγαλύτερη ιστορικά αγορά, τότε το συμβάν δημοσιοποιείται ως ύποπτο. Η υπηρεσία αυτή μπορεί να ενημερώνει τον κάτοχο της κάρτας για την τελευταία αγορά του που υπερέβη το συνηθισμένο ποσό. Όπως μπορεί να διαπιστωθεί από τα παραπάνω παραδείγματα, οι δυνατότητες είναι σχεδόν απεριόριστες. Οι υπηρεσίες και οι επιχειρησιακές διαδικασίες μπορούν να είναι γεννήτορες συμβάντων, ενέργειες καθοδηγούμενες από σύμβάντα ή και τα δύο.



Σχήμα 5: Επεξεργασία Σύνθετων Συμβάντων

2.4 Επεξεργασία Σύνθετων Συμβάντων

Η Επεξεργασία Σύνθετων Συμβάντων (CEP – Complex Event Processing), αποτελεί την πιο εξελιγμένη μορφή επεξεργασίας συμβάντων που σαν σκοπό έχει την εξαγωγή γνώσης από πολλαπλά δεδομένα το οποίο προέρχονται από διαφορετικές πηγές. Συγκεκριμένα η CEP μπορεί να εφαρμοστεί στην ανάλυση και εξόρυξη δεδομένων από οποιοδήποτε κατανεμημένο σύστημα παραγωγής και ανταλλαγής μηνυμάτων το οποίο μπορεί να κυμαίνεται από χαμηλού επιπέδου διαχείριση δικτύων μέχρι υψηλού επιπέδου εφαρμογές επιχειρησιακής ευφυΐας. Τα σύνθετα συμβάντα αποτελούν άθροισμα συσχετιζόμενων απλών συμβάντων. Μέσω μιας διαδικασίας drill down μπορούν να προσδιοριστούν τα συστατικά μέρη ενός σύνθετου συμβάντος όπως φαίνεται και στο σχήμα που ακολουθεί [14]:



Σχήμα 6: Διαδικασία Drill-Down από Σύνθετα σε Απλά Συμβάντα

Η όλη ιδέα της CEP βασίζεται στα πρότυπα συμβάντων (event patterns) και στην αντιστοίχηση προτύπων (pattern matching). Η CEP βασίζεται σε τεχνολογίες αναγνώρισης σύνθετων προτύπων από ένα νέφος δεδομένων, αναγνωρίζει συσχετισμούς και ιεραρχίες συμβάντων, εξαρτήσεις και σχέσεις μεταξύ συμβάντων όπως αιτιότητα, συμμετοχή, χρονισμός (σε ροές συμβάντων). Σήμερα αποτελεί μια ιδέα που συνεχώς εξελίσσεται και διευρύνεται το πεδίο εφαρμογών της και η οποία οφείλεται στον Dr David Luckham του Πανεπιστημίου του Stanford [13].

Η CEP εφαρμόζεται στην διαχείριση επιχειρησιακών διαδικασιών (BPM), στη διαχείριση ρίσκου, στην ανάπτυξη πολιτικών ασφαλείας δικτύων και στην ανίχνευση κινδύνων στο τραπεζικό και ασφαλιστικό τομέα. Μια πολύ σημαντική εφαρμογή της CEP είναι στην παρακολούθηση της επιχειρησιακής δραστηριότητας (BAM – Business Activity Monitoring) των υπολογιστικών πόρων μιας επιχείρησης. Η επεξεργασία των συμβάντων που παράγονται από το σύνολο των υποσυστημάτων μιας επιχείρησης μπορεί να ανιχνεύσει έγκαιρα οποιαδήποτε δυσλειτουργία η οποία μπορεί να θέσει σε κίνδυνο πληροφορίες ή κρίσιμες λειτουργίες της επιχείρησης.

Η CEP υλοποιείται βάσει της EDA αρχιτεκτονικής η οποία ανιχνεύει, επεξεργάζεται και αντιδρά στα συμβάντα. Ο συνδυασμός CEP με EDA προσφέρει ταχύτερες και αποτελεσματικότερες επιχειρησιακές αποφάσεις και καλύτερους χρόνοι απόκρισης. Η κεντρική ιδέα των Επεξεργασίας Σύνθετων Συμβάντων βασίζεται στην ανίχνευση και αντιστοίχηση προτύπων. Ακολουθεί ανάλυση μερικών βασικών εννοιών.

2.4.1 Τι είναι ένα πρότυπο (pattern)

Για την πλειοψηφία των ανθρώπων η λέξη πρότυπο (pattern) εμφανίστηκε σχεδόν αποκλειστικά λόγω της εργασίας του Christopher Alexander, ενός καθηγητή Αρχιτεκτονικής στον

Πανεπιστήμιο του Berkeley. Το βιβλίο του που εισήγαγε τη γλώσσα για επεξεργασία προτύπων το 1977, αποτελεί τη βάση όλων των βιβλίων λογισμικού που ασχολούνται με την αναγνώριση προτύπων. Σύμφωνα με τον ορισμό που έδωσε ο C. Alexander ένα πρότυπο είναι ένας μορφολογικός κανόνας ο οποίος εξηγεί τον τρόπο σχεδίασης ενός αντικειμένου προκειμένου να επιλυθεί ένα πρόβλημα σε ένα καθορισμένο πλαίσιο. Στη πληροφορική αυτή η ιδέα χρησιμοποιήθηκε για την καθιέρωση βέλτιστων πρακτικών (best practices) στη δημιουργία λογισμικού για την επίλυση απλών και επαναλαμβανόμενων προβλημάτων. Η πρώτη δημοσίευση που έτυχε καθολικής αναγνώρισης στο χώρο της πληροφορικής ήταν το βιβλίο "Design Patterns: Elements of Reusable Object-Oriented Software" [15]. Το βιβλίο αυτό αύξησε κατακόρυφα το επιστημονικό ενδιαφέρον στη επεξεργασία προτύπων στην επιστήμη των υπολογιστών.

Πρότυπα Συμβάντων (Event Patterns)

Τα πρότυπα στο πεδίο των υπολογιστών και πολύ περισσότερο στο πεδίο της επεξεργασίας συμβάντων αφορούν σε ένα σύνολο συμβάντων που είναι απλά και σύνθετα και τα οποία περιέχονται σε ένα υπερσύνολο το οποίο είναι ένα νέφος συμβάντων. Το πρότυπο συμβάντων είναι ο αυστηρός και πλήρης καθορισμός της δομής τους που αποτελείται από την εσωτερική δομή τους, τα χαρακτηριστικά τους, και τις συνθήκες βάσει των οποίων γίνονται αντιληπτά από ένα υπολογιστικό σύστημα, όπως πχ από έναν αισθητήρα.

Εφαρμογές οι οποίες μπορούν να διαχειριστούν συμβάντα σε πραγματικό χρόνο, μπορούν να ανιχνεύσουν και πρότυπα συμβάντων σε ένα νέφος συμβάντων και να κάνουν αντιστοίχηση προτύπων σε πραγματικό χρόνο. Για να επιτευχθεί αυτό είναι αναγκαίο τα πρότυπα να περιγράφουν όχι μόνο τα συμβάντα αλλά και τις συσχετίσεις τους με άλλα συμβάντα. Τα πρότυπα συμβάντων αναπαρίστανται με ECA (Event-Condition-Action) κανόνες οι οποίοι καθορίζουν τις ενέργειες που πρέπει να εκτελεστούν όταν επιτευχθεί αντιστοίχηση κάποιου προτύπου με κάποιο συμβάν.

Χαρακτηριστικά ενός Προτύπου

Ένα συμβάν είναι ένα αντικείμενο με την έννοια της εγγραφής (σε μια βάση δεδομένων ή σε ένα αρχείο) μιας δραστηριότητας σε ένα σύστημα. Το συμβάν υποδηλώνει τη δραστηριότητα. Ένα συμβάν μπορεί να είναι αυτόνομο ή/και μπορεί να σχετίζεται και με άλλα συμβάντα. Για τον πλήρη καθορισμό ενός συμβάντος θα πρέπει να οριστούν:

- Η μορφή (form)
- Η έννοια (significance)
- Η συσχέτιση (relativity)

Η μορφή ενός συμβάντος μπορεί να είναι ένα αντικείμενο (object) με συγκεκριμένα χαρακτηριστικά ή στοιχεία δεδομένων. Η αναπαράσταση ενός συμβάντος μπορεί να είναι απλή όπως μια συμβολοσειρά δεδομένων ή τις περισσότερες φορές μια πλειάδα από δεδομένα. Παράδειγμα δεδομένων θα μπορούσε να είναι η χρονική καταγραφή, ένας ενιαίος προσδιοριστής (identifier) ή κάποιο χαρακτηριστικό βάσει του οποίου συσχετίζονται τα συμβάντα.

Ένα συμβάν αναπαριστά μια δραστηριότητα. Αυτή η δραστηριότητα καλείται «έννοια του συμβάντος». Σε ένα πρότυπο περιέχονται τα δεδομένα τα οποία προσδιορίζουν την έννοια της δραστηριότητας που αναπαρίσταται.

Η συσχέτιση αφορά στη σχέση μεταξύ διάφορων δραστηριοτήτων οι οποίες έχουν κοινό χαρακτηριστικό τη χρονική στιγμή που εμφανίζονται, τη χρονική διάρκεια ή κάποιο άλλο χαρακτηριστικό. Η σχέση των συμβάντων μεταξύ τους καθορίζει και τη συσχέτιση τους. Μπορεί για παράδειγμα η δραστηριότητα να αφορά κάποιο συγκεκριμένο τύπο του συμβάντος, πχ όλα τα συμβάντα που ανιχνεύονται από RFID αισθητήρες. Εάν η συσχέτιση γίνεται βάσει του χρονικού προσδιορισμού τότε καθορίζεται από το “Timestamp”.

2.4.2 Πρότυπα Σύνθετων Συμβάντων

Τα Πρότυπα Σύνθετων Συμβάντων δημιουργούνται από ένα σύνολο απλών συμβάντων τα οποία παράγονται από διαφορετικές διαδικασίες και υπολογιστικά συστήματα μιας επιχείρησης. Τα συμβάντα αυτά δημιουργούνται σε διαφορετικές χρονικές στιγμές, από διαφορετικά υποσυστήματα τα οποία σχετίζονται με άλλα συμβάντα που μπορεί να μην σχετίζονται άμεσα ή εννοιολογικά με την αυτή καθαυτή αντιστοίχηση στο πρότυπο του συμβάντος. Τα συμβάντα τα οποία προκαλούν την δημιουργία ενός σύνθετου συμβάντος είναι καθόλη τη διάρκεια ζωής του σύνθετου συμβάντος συσχετιζόμενο με αυτό. Και αυτό γιατί παρέχουν όλες τις πληροφορίες που είναι απαραίτητες για την επεξεργασία και εκτέλεση των λειτουργιών που εκκινούνται από την ανίχνευση του συμβάντος. Οι πληροφορίες αυτές μεταφέρονται με τη μορφή παραμέτρων.

Τα πρότυπα CEP αποτελούν ιδανικές λύσεις για την υλοποίηση λύσεων σε προβλήματα τα οποία έχουν τα ίδια χαρακτηριστικά και εμφανίζονται συχνά. Τα πλεονεκτήματα των προτύπων αυτών είναι ότι είναι προκαθορισμένα, επαναχρησιμοποιήσιμα και η δυνατότητά τους να παραμετροποιούνται δυναμικά επιτρέπει την χρήση μιας λύσης ως βάση για την υλοποίηση άλλων CEP λύσεων.

Τύποι Προτύπων Σύνθετων Συμβάντων

Έχουν προταθεί από τον Barros [16], οι απαιτήσεις για την επεξεργασία σύνθετων συμβάντων σε επιχειρησιακές διαδικασίες και η καθιέρωση ενός πλαισίου αναφοράς για την αξιολόγηση συστημάτων και προτύπων. Παρά το γεγονός ότι η συγκεκριμένη εργασία αφορά μόνο σε ένα τμήμα εφαρμογής της CEP, στην επεξεργασία επιχειρησιακών διαδικασιών, η αποτύπωση και κατηγοριοποίηση των προτύπων βάσει των συγκεκριμένων αναγκών που καλούνται να ικανοποιήσουν αποτελεί μια καλή προσέγγιση την αντιμετώπιση γενικότερων προβλημάτων:

1. Συν-εμφανιζόμενα Πρότυπα (Co-occurrence Patterns)

Αυτό ο τύπος προτύπου περιγράφει περιπτώσεις χρήσης που πρέπει να ληφθούν υπόψη μια πλειάδα γεγονότων προκειμένου να αποφασισθεί εάν υπάρχει ή όχι ταυτοποίηση με το πρότυπο. Στη περίπτωση αυτή θα πρέπει να ανιχνευθούν δύο ή περισσότερα συμβάντα προκειμένου να γίνει αντιστοίχηση. Μια διαφορετική περίπτωση θα ήταν η αντιστοίχηση να επιτυγχάνεται όταν απουσιάζει κάποιο συγκεκριμένο συμβάν.

2. Χρονικά Συσχετιζόμενα Πρότυπα (Time Relation Patterns)

Τα πρότυπα αυτά αφορούν περιορισμούς σχετικούς με το χρόνο. Παράδειγμα τέτοιων προτύπων είναι όταν δύο συμβάντα πραγματοποιούνται εντός ή εκτός ενός καθορισμένου χρονικού πλαισίου. Επίσης όταν συμβαίνουν μετά ή πριν από ένα καθορισμένο χρονικό σημείο ή όταν πραγματοποιούνται σε μια συγκεκριμένη χρονική σειρά.

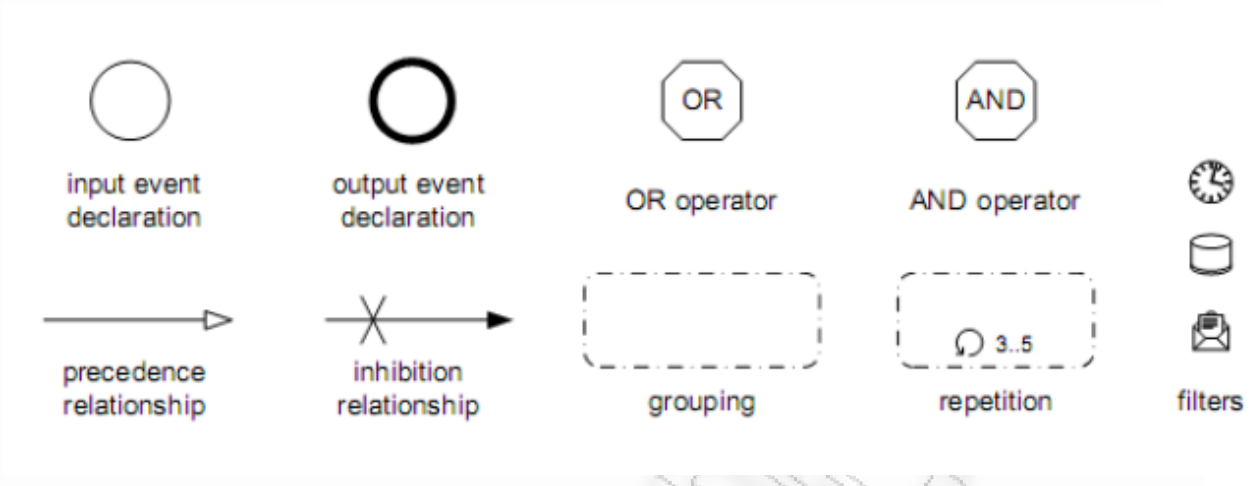
3. Πρότυπα που εξαρτώνται από τα δεδομένα (Data Dependency Patterns)

Τα συμβάντα σε αυτή την περίπτωση εμπεριέχουν δεδομένα στα οποία διακρίνουμε σχέσεις αλληλεξάρτησης. Δηλαδή δύο συμβάντα μπορεί να αντιστοιχίζονται όταν τα δεδομένα τους σχετίζονται με ένα συγκεκριμένο τρόπο. Μπορεί επίσης τα δεδομένα ενός συμβάντος να αντιστοιχίζονται με τα δεδομένα ενός στιγμιότυπου μιας επιχειρησιακής διαδικασίας.

Γλώσσες Περιγραφής Συμβάντων

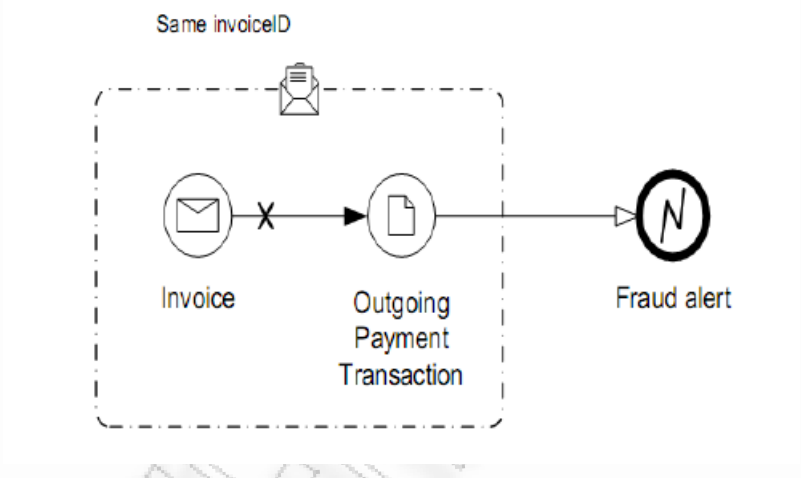
BEMN (The Business Event Modelling Notation)

Ο Decker [17] εισήγαγε την BEMN (The Business Event Modelling Notation) ως επίσημη γλώσσα περιγραφής συμβάντων που αφορούν επιχειρησιακές δραστηριότητες και πρότυπα συμβάντων. Μέσω της BEMN τα πρότυπα καθορίζονται μέσω κανόνων. Ο κάθε κανόνας αποτελείται από την περιγραφή του προτύπου του συμβάντος, τα χαρακτηριστικά ενός συμβάντος προκειμένου να αντιστοιχηθεί με το πρότυπο και το σύνολο των ενεργειών που πρέπει να εκκινηθούν ως αποτέλεσμα ενεργοποίησης του κανόνα. Δηλαδή τα συμβάντα μπορούν να αναπαρασταθούν μέσω ECA κανόνων οι οποίοι γραφικά θα ακολουθούν τη σημειογραφία της BEMN. Ο πίνακας που ακολουθεί [17] συνοψίζει αυτή τη σημειογραφία:



Σχήμα 7: Περιγραφή Συμβάντων με BEMN

Το επόμενο σχήμα περιγράφει μια περίπτωση χρήσης για έναν απλό συναγερμό ανίχνευσης απάτης εκφρασμένο σε BEMN. Όποτε πραγματοποιείται μια διαταγή πληρωμής χωρίς να έχει παραληφθεί το αντίστοιχο τιμολόγιο ενεργοποιείται ο συναγερμός.



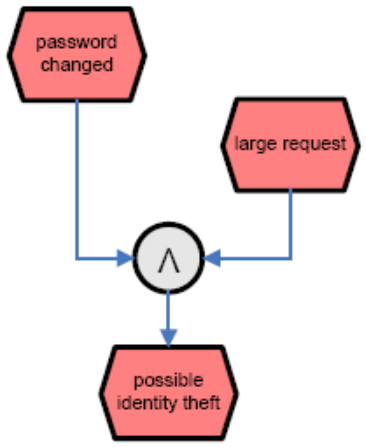
Σχήμα 8: Παράδειγμα χρήσης της BEMN

Μοντελοποίηση μέσω Event-Driven Process Chains (EPC)

Μια ακόμη μεθοδολογία μοντελοποίησης σύνθετων συμβάντων σε επιχειρησιακές διαδικασίες είναι μέσω διαγραμμάτων συμβάντων (event diagrams) και αλυσίδες διαδικασιών καθοδηγούμενες από συμβάντα (event driven process chains – EPC) [19]. Η μοντελοποίηση επιχειρησιακών διαδικασιών ακολουθούν μια προσέγγιση από πάνω προς τα κάτω, δηλαδή από τις πιο σύνθετες στις πιο απλές. Αυτή η προσέγγιση ακολουθείται και στη μοντελοποίηση συμβάντων όπου τα σύνθετα συμβάντα αναλύονται στα απλά συμβάντα από τα οποία αποτελούνται. Τα διαγράμματα συμβάντων περιγράφουν τις ιεραρχίες και αλληλεξαρτήσεις μεταξύ σύνθετων και απλών συμβάντων. Τα συμβάντα συνδέονται μέσω λογικών τελεστών (AND, OR, XOR). Στο παράδειγμα που ακολουθεί για να γίνει περισσότερο κατανοητή η διαδικασία μοντελοποίησης, αναλύεται η ανίχνευση κλοπής της του κωδικού μιας πιστωτικής κάρτας. Το συμβάν «πιθανή κλοπή κωδικού» ενεργοποιείται στην περίπτωση που έχουμε αλλαγή κωδικού και ταυτόχρονη πίστωση ενός μεγάλου ποσού μέσα σε 24 ώρες. Στο αριστερό σχήμα παρουσιάζονται τα 2 συμβάντα «αλλαγή κωδικού» και «μεγάλη πίστωση» τα οποία συνδέονται με έναν AND τελεστή, γιατί πρέπει να επαληθευτούν και τα 2 ταυτόχρονα για να

εκκινηθεί η διαδικασία «πιθανής κλοπής κωδικού». Το πρόβλημα που παρουσιάζεται είναι ότι δεν μπορεί ευκρινώς να παρουσιαστεί η χρονική αλληλουχία των γεγονότων.

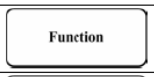
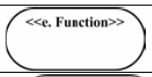
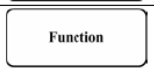
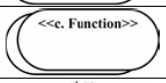
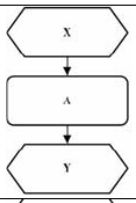
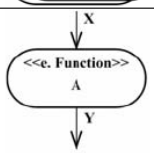
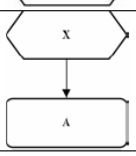
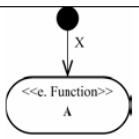
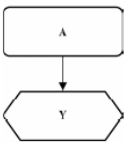
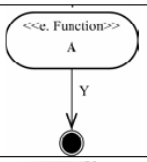
Το πρόβλημα αυτό επιλύεται με τους πίνακες απόφασης (decision tables) ως επέκταση των EPC. Παρατηρούμε ότι καταγράφονται όλες οι πιθανές εκδοχές των συμβάντων που υπολογίζονται με βάση των τελεστή AND και τα αποτελέσματα αυτού του υπολογισμού που μπορεί να εκκινήσουν μια ενέργεια όπου υπολογίζονται βάσει του τελεστή XOR. Με αυτή τη μέθοδο τα απλά συμβάντα λειτουργούν ως συνθήκες και τα σύνθετα ως ενέργειες. Η διαδικασία εκκινείται όταν αρχικά ικανοποιείται η συνθήκη «Large Request» να είναι πάντα αληθής γιατί αυτή πάντα προηγείται χρονικά [19].



	R1	R2	R3	R4
Large request	Y	Y	Y	Y
Password changed	Y	Y	N	N
$T_{password\ changed}$-$T_{large\ request}$ < 1 day	Y	N	Y	N
Possible identity theft	x			
No identity theft		x	x	x

Σχήμα 9: Περιγραφή Συμβάντων με Event-Driven Process Chains

Η μοντελοποίηση με EPC αποτελεί μια διαδεδομένη μεθοδολογία μοντελοποίησης διαδικασιών και σε αυτή την περίπτωση μοντελοποίησης και συμβάντων. Τελευταία έχει προταθεί η μία – προς – μια αντιστοίχηση μεταξύ των EPC και της UML2 διάγραμμα δραστηριοτήτων. Η αντιστοίχηση αυτή παρουσιάζεται στον πίνακα που ακολουθεί [18]:

EPC Element	EPC Notation	UML 2 Base Class	UML Profile
Elementary Function		Action	
Complex Function		Action	
Event		Control Flow	
Start Event		Initial Node	
End Event		Final Node	

Πίνακας 2: Αντιστοίχηση μεταξύ EPC και UML2

2.4.3 Αντιστοίχηση Προτύπου (Pattern Matching)

Εφόσον έχουν οριστεί τα συμβάντα και τα πρότυπα συμβάντων ακολουθεί η ανάλυση της βασικής λειτουργίας στην επεξεργασία συμβάντων που είναι η αντιστοίχηση προτύπου. Στην επιστήμη των υπολογιστών με τον όρο αντιστοίχηση προτύπου αναφερόμαστε στον έλεγχο μιας σειράς συμβόλων τα οποία μπορεί να αποτελούν τα συστατικά μέρη ενός προτύπου. Η αντιστοίχηση προτύπου διαφέρει από την αναγνώριση προτύπου (pattern recognition) γιατί η αντιστοίχηση σε ένα πρότυπο δεν απαιτείται να είναι πανομοιότυπη [3w13]. Τα πρότυπα στις περισσότερες περιπτώσεις είναι είτε μια ακολουθία συμβολοσειρών ή μια δενδρική δομή.

Αντιστοίχηση Προτύπων Συμβάντων (Event Pattern Matching)

Στην CEP η διαδικασία αντιστοίχησης προτύπου αφορά την αναγνώριση μέσα σε ένα νέφος συμβάντων αυτά τα πρότυπα τα οποία είναι σημαντικά για την επιχειρησιακή λειτουργία της επιχείρησης. Τα περισσότερα συστήματα επεξεργασίας σύνθετων συμβάντων για την αντιστοίχηση προτύπου υποστηρίζουν:

- Την λογική και χρονική συσχέτιση των συμβάντων
- Τον κύκλο ζωής ενός συμβάντος (από τη στιγμή της ανίχνευσης μέχρι τη στιγμή που θα «αναλωθεί») ελέγχεται μέσω χρονικών τελεστών
- Την προώθηση των συμβάντων που ανάγονται σε πρότυπα στους κατάλληλους αποδέκτες

Μερικοί από τους πιο γνωστούς αλγόριθμους αντιστοίχησης προτύπου είναι οι εξής:

- Linear
- Rete
- Treat
- Leaps

2.5 Διαφορές μεταξύ CEP και ESP

Στη βιβλιογραφία εμφανίζεται συχνά η επεξεργασία σύνθετων συμβάντων ως προέκταση της επεξεργασίας ροών συμβάντων (ESP – Event Stream Processing). Για την αποσαφήνιση των διαφορών μεταξύ των δύο αυτών μεθόδων επεξεργασίας συμβάντων αναφέρονται στη συνέχεια τα κύρια χαρακτηριστικά τους και οι διαφορές τους:

Επεξεργασία Ροών Συμβάντων (Event Stream Processing, ESP)

- Η εξαγωγή απλών δεδομένων από μια ροή
- Τα συμβάντα είναι πλήρως χρονικά ταξινομημένα
- Εφαρμογή στην διαχείριση μετοχών αλγοριθμικά

Επεξεργασία Σύνθετων Συμβάντων (Complex Event Processing, CEP)

- Εξαγωγή προτύπων σύνθετων γεγονότων από ένα νέφος (cloud)
- Μερική χρονική ταξινόμηση των συμβάντων
- Η ταξινόμηση μπορεί να γίνει και με βάσει και άλλα στοιχεία, εκτός από το χρόνο
- Η επεξεργασία απαιτεί περισσότερο χρόνο και μνήμη
- Μπορεί να εφαρμοστεί στην παρακολούθηση επιχειρησιακών διαδικασιών
- Η CEP αποτελεί υπερσύνολο του ESP

Σήμερα η CEP και η ESP χρησιμοποιούν αντίστοιχα η μια μεθοδολογίες και τεχνολογίες της άλλης.

2.6 Εφαρμογές της CEP

Για την υλοποίηση της ιδέας την Επεξεργασίας Σύνθετων Συμβάντων απαιτείται μια πλατφόρμα επεξεργασίας σύνθετων συμβάντων. Οι πλατφόρμες αυτές επεξεργάζονται ένα μεγάλο όγκο πληροφορίας που εμπεριέχεται σε ένα νέφος συμβάντων προκειμένου να ανιχνεύσουν σύνθετα συμβάντα και να τα επεξεργάζονται σε πραγματικό χρόνο. Ο όρος «νέφος συμβάντων» περιλαμβάνει όλα τα συμβάντα που παράγονται από την εσωτερική λειτουργία της επιχείρησης καθώς επίσης και συμβάντα που προέρχονται από την εξωτερική επικοινωνία της επιχείρησης με άλλες επιχειρήσεις. Το σύνολο της εξωτερική και εσωτερικής επικοινωνίας πραγματοποιείται μέσω συμβάντων και με αυτή την έννοια οι επιχειρήσεις λειτουργούν σε ένα παγκόσμιο νέφος συμβάντων.

Τα συστατικά μέρη μιας τέτοιας πλατφόρμας επεξεργασίας σύνθετων συμβάντων περιγράφεται στο σχήμα που ακολουθεί:

Σχήμα 10: Συστατικά μέρη μιας CEP πλατφόρμας

Η διαχείριση σύνθετων συμβάντων αφορά την αποτύπωση των συσχετίσεων μεταξύ των συμβάντων προκειμένου, όπως έχει ήδη αναλυθεί, να αντιστοιχηθούν στα καθορισμένα πρότυπα. Συγκεκριμένα οι *Προδιαγραφές των Συμβάντων* καθορίζουν επακριβώς το σύνολο των πιθανών συμβάντων με τα χαρακτηριστικά και τις εξαρτήσεις τους. Κάθε συμβάν περιέχει γενικά μεταδεδομένα, δεδομένα δηλαδή που χαρακτηρίζουν τα δεδομένα, όπως το ID του συμβάντος, το timestamp, ο τύπος του συμβάντος κ.α. Οι *Κανόνες Επεξεργασίας των Συμβάντων* αφορούν τους ECA κανόνες, όπως έχουν αναλυθεί επαρκώς σε προηγούμενες παραγράφους. Η διαδικασία επεξεργασίας αφορά τα συμβάντα, και πιο λεπτομερώς τα *Δεδομένα των Συμβάντων*, τα οποία είναι καταχωρημένα στη μνήμη του συστήματος. Ο Μηχανισμός Επεξεργασίας, είναι μια μηχανή κανόνων η οποία εκτελεί τους ECA κανόνες ανάλογα με τα δεδομένα των συμβάντων.

Επίπεδα Λειτουργικότητας μιας CEP εφαρμογής

Οι εφαρμογές που υποστηρίζουν CEP διακρίνονται ανάλογα με το επίπεδο λειτουργικότητας που υποστηρίζουν. Το Αμερικανικό Υπουργείο Εθνικής Αμύνης έχει προτείνει μια ταξινόμηση σε 4 επίπεδα λειτουργικότητας. Η ταξινόμηση επιμελήθηκε στο JDL – Joint Directors Laboratories για το JDL Data Fusion model [22]. Τα επίπεδα λειτουργικότητας είναι τα εξής:

- 1ο Επίπεδο:

  Προ-επεξεργασία συμβάντων. Ομαδοποίηση των συμβάντων σε ομάδες όπου η κάθε ομάδα αναπαριστά ένα υψηλότερου επιπέδου σύνθετο συμβάν
- 2ο Επίπεδο:

  Συνδυασμός των χαρακτηριστικών και των σχέσεων μεταξύ των ομάδων συμβάντων του 1ου επιπέδου για την εξαγωγή σύνθετων προτύπων.
- 3ο Επίπεδο:

  Ανάλυση των αποτελεσμάτων από τις διαδικασίες επεξεργασίας των προηγούμενων επιπέδων για την ανίχνευση ευκαιριών και κινδύνων
- 4ο Επίπεδο:

  Συνεχής επαναξιολόγηση της όλης διαδικασίας επεξεργασίας των συμβάντων για τη βελτιστοποίηση αυτής καθαυτής της διαδικασίας για την επίτευξη των βέλτιστων αποτελεσμάτων. Το 4ο επίπεδο λειτουργεί διαδραστικά με όλα τα προηγούμενα επίπεδα.

Στις εμπορικές εφαρμογές επεξεργασίας συμβάντων τα επίπεδα 3 και 4 χρησιμοποιούνται στον αυτόματο έλεγχο σε συστήματα διαχείρισης επιχειρησιακών διαδικασιών. Τα σύγχρονα συστήματα διαχείρισης συμβάντων προσφέρουν λειτουργικότητα που να αντιστοιχεί στα 2 πρώτα επίπεδα. Βασίζονται σε μηχανισμούς κανόνων για την αντιστοίχηση προτύπων. Τα συστήματα αυτά δεν φτάνουν στο σημείο να αναγνωρίσουν προβλήματα και κινδύνους. Τα συστήματα που πρέπει να επιλύσουν προβλήματα τα οποία μπορούν να προσεγγιστούν με διακριτούς κανόνες τότε δεν απαιτούνται πολύπλοκες λύσεις που να υποστηρίζουν και τα 4 επίπεδα, αλλά μπορούν να χρησιμοποιηθούν απλοί μηχανισμοί κανόνων χωρίς την ολοκλήρωσή τους και με άλλα υποσυστήματα.

Επιγραμματικά μπορούν να αναφερθούν οι τεχνολογίες επεξεργασίας συμβάντων:

- Τεχνολογίες Υλοποίησης/ Ολοκλήρωσης με άλλα συστήματα
  - Service Oriented Architecture (SOA)
  - Event Driven Architecture (EDA)
- Εφαρμογές
  - Επεξεργασία Επιχειρησιακών διαδικασιών (BPM)
    - Αυτοματισμός των ενεργειών που υποδεικνύονται από το αποτέλεσμα την επεξεργασίας συμβάντων
  - Παρακολούθηση Επιχειρησιακών Δραστηριοτήτων (BAM)
    - Έλεγχος της αποδοτικότητας των υπολογιστικών συστημάτων

- Συμπληρώνει την CEP με τα BPM

- Επεξεργασία Σύνθετων Συμβάντων (CEP)
  - Εφαρμογή επιχειρησιακών κανόνων σε νέφη συμβάντων

2.6.1 CEP και Συστήματα Υποστήριξης Αποφάσεων

Το κύριο ζητούμενο στα συστήματα βασισμένα στην αρχιτεκτονική EDA είναι ένα ακριβές μοντέλο συμβάντων, το οποίο να αντικατοπτρίζει τα διάφορα στάδια της επεξεργασίας τους και να αποτελεί τη βάση για τη σύνολο της αρχιτεκτονικής του συστήματος. Σε γενικές γραμμές, όλα τα συμβάντα μπορούν να είναι διαρθρωθούν σε μια πολυεπίπεδη ιεραρχία: Για τα συστήματα υποστήριξης αποφάσεων που βασίζονται σε δεδομένα από αισθητήρες η ιεραρχία των συμβάντων μπορεί να ακολουθεί την διάταξη που απεικονίζεται στο σχήμα που ακολουθεί [10]:

Σχήμα 11: Ιεραρχία Συμβάντων σε ένα σύστημα υποστήριξης αποφάσεων

Η εξέλιξη των συμβάντων είναι ανάλογη με την διαδικασία λήψης μιας απόφασης από ένα Σύστημα Υποστήριξης Αποφάσεων (ΣΥΑ). Τα ΣΥΑ αποτελούν την κορυφή μιας ανάλογης πυραμίδας όπου από επίπεδο σε επίπεδο τα υπολογιστικά συστήματα επεξεργάζονται όλο και μεγαλύτερο όγκο πληροφοριών καταλήγοντας στα ΣΥΑ όπου υπάρχει η δυνατότητα επιλογής της βέλτιστης απόφασης για την επίλυση ενός συγκεκριμένου προβλήματος. Η πυραμίδα των συστημάτων που καταλήγουν στα ΣΥΑ και τα δεδομένα που επεξεργάζονται σε κάθε επίπεδο φαίνεται στο σχήμα που ακολουθεί [3w21]:

Σχήμα 12: Ιεραρχία Επιχειρησιακών Συστημάτων

Είναι εμφανής η ομοιότητα μεταξύ των δύο σχημάτων όπου η ιεραρχία των συμβάντων και η ιεραρχία των συστημάτων ακολουθούν τη ίδια λογική.

Όσον αφορά στο πρώτο σχήμα που αποτυπώνει την ιεραρχία των συμβάντων σε ένα ΣΥΑ, αυτά αναλύονται ως εξής [10]:

Ακατέργαστα Συμβάντα:

Είναι τα συμβάντα όπως αρχικά ανιχνεύονται από τους αισθητήρες και η πληροφορία που μεταφέρουν περιέχει μεγάλο ποσοστό θορύβου.

Ταξινομημένα Δεδομένα Συμβάντων:

Είναι τα συμβάντα που ανιχνεύονται από τους αισθητήρες υπόκεινται σε επεξεργασία προκειμένου να «καθαριστούν» από το θόρυβο και να ταξινομηθούν σε κάποιο συγκεκριμένο τομέα. Τα δεδομένα αυτά παρά την επεξεργασία παραμένουν, σε αυτό το στάδιο, ασυσχέτιστα μεταξύ τους.

Συμβάντα Ταυτοποίησης Προβλήματος:

Τα ταξινομημένα, κατά πεδίο εφαρμογών, δεδομένα συμβάντων σχετίζονται με ένα πρόβλημα το οποίο πρέπει να επιλύσει ο ειδικός που είναι επιφορτισμένος με αυτή την αρμοδιότητα.

Συμβάντα Αιτιών:

Τα συμβάντα επιμερίζονται σε μια σειρά στοιχειωδών συμβάντων τα οποία αποτελούν τα πιθανά αίτια του προβλήματος.

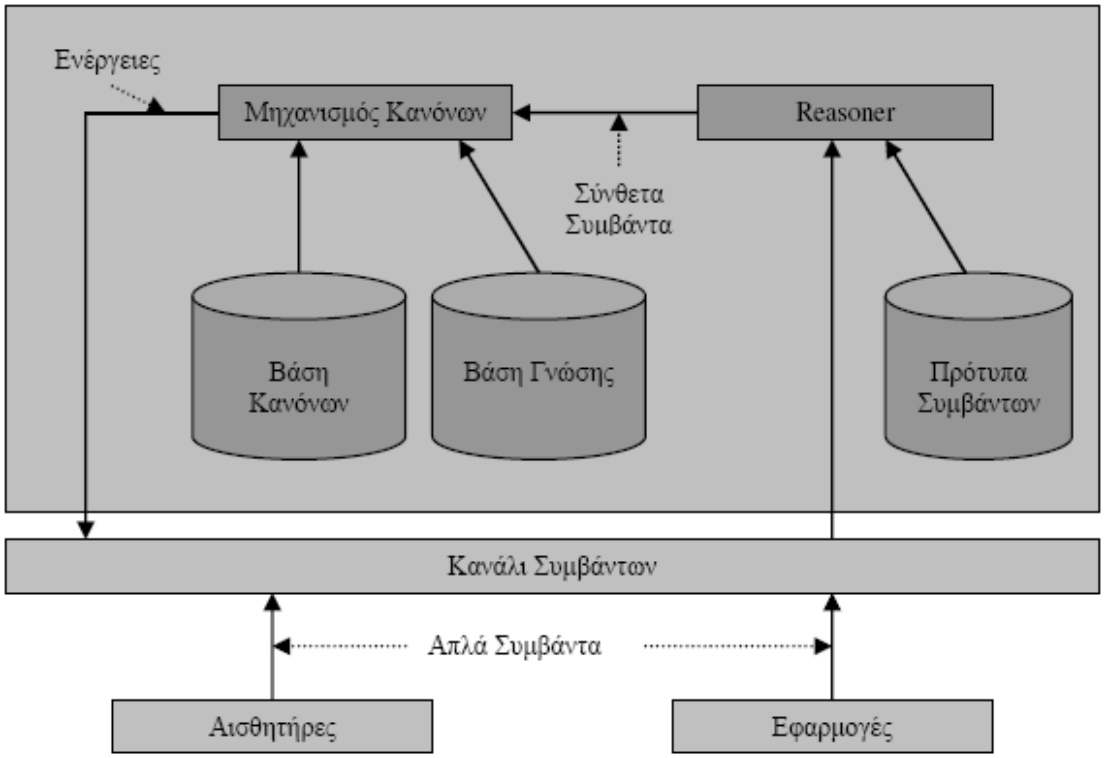
Συμβάντα Ενεργειών:

Τα συμβάντα που παράγονται, ως λύση στο πρόβλημα που ανιχνεύθηκε, αναγνωρίστηκε και αναλύθηκε. Ουσιαστικά οι ενέργειες αυτές αποτελούν τις τελικές αποφάσεις που στα πλαίσια ενός ΣΥΑ πρέπει να εφαρμοστούν για την επίλυση του προβλήματος.

2.6.2 CEP και Επιχειρησιακές Διαδικασίες

Οι Chakravarty και Singh [9] περιγράφουν την αρχιτεκτονική και τα εργαλεία σκιαγραφώντας μια μεθοδολογία βάσει της οποίας κάθε συμμετέχων σε μια επιχειρησιακή διαδικασία να μπορεί να καθορίζει, να ανιχνεύει και να ανταποκρίνεται στα συμβάντα. Προτείνουν μια αρχιτεκτονική καθοδηγούμενη από συμβάντα (EDA), η οποία έχει ως στόχο την ενσωμάτωση των συμβάντων σε μια Δικτυακή διαδικασία με επαναχρησιμοποιήσιμο τρόπο - δηλαδή, χωρίς τη χρήση αυστηρά καθορισμένων μέσω κώδικα συμβάντων. Μια αρχιτεκτονική βασισμένη σε EDA παρέχει έναν τρόπο οργάνωσης συστημάτων, τα οποία ανιχνεύουν, αναλύουν και ανταποκρίνονται στα συμβάντα. Στις επιχειρησιακές διαδικασίες, η ανίχνευση προϋποθέτει τη λήψη συμβάντων από πολλαπλές πηγές (αισθητήρες, εφαρμογές λογισμικού, και άλλα), η ανάλυση περιλαμβάνει τον καθορισμό των ενεργειών ως αντίδραση στα συμβάντα και η δέσμη ενεργειών που ενεργοποιούνται μεταβάλουν την κατάσταση του συστήματος.

Επιπλέον αναφέρεται ότι η ιδέα του προτύπου συμβάντος είναι ένας τρόπος έκφρασης ενός σύνθετου συμβάντος. Ενώ ένα απλό συμβάν προσδιορίζεται από το όνομά του και από τις τιμές των παραμέτρων του, ένα πολύπλοκο συμβάν εκφράζεται ως πρότυπο που αναπαριστά ένα σύνολο απλών συμβάντων. Τα σύνθετα συμβάντα μπορούν να εκφραστούν με κανόνες της μορφής: WHEN «event» IF «condition» THEN «action». Η προτεινόμενη αρχιτεκτονική ενός τέτοιου συστήματος απεικονίζεται στο ακόλουθο σχήμα [9]:



Σχήμα 13: Αρχιτεκτονική ενός CEP συστήματος για BPM

Όπως παρουσιάζεται στην εικόνα, ο μηχανισμός εξαγωγής συμπερασμάτων διαχωρίζεται από τους επιχειρησιακούς κανόνες. Απλά συμβάντα φθάνουν στον reasoner, ο οποίος διατηρεί (μερικώς ανιχνευόμενα) σύνθετα συμβάντα στο σημείο αποθήκευσης προτύπων. Τα πρότυπα παραμένουν στο σημείο αποθήκευσης έως ότου αυτά εξελιχθούν σε «Αληθή» (έχουν ανιχνευθεί) και «Ψευδή» (αδύνατα). Μόλις ένα πρότυπο μετατραπεί σε «Αληθές», ο μηχανισμός εξαγωγής συμπερασμάτων ενημερώνει τον μηχανισμό παραγωγής κανόνων για την εμφάνιση του αντίστοιχου σύνθετου συμβάντος, και η μηχανή εφαρμόζει τους επιχειρησιακούς κανόνες που ταιριάζουν στην περίσταση, εκτελώντας τις κατάλληλες ενέργειες. Κάθε ενέργεια

αντιμετωπίζεται ως ένα απλό συμβάν και μπορεί να εμφανιστεί σε πρότυπα συμβάντων. Υπάρχουν τρεις πηγές πληροφόρησης για κάθε τμήμα που συμμετέχει στην διαδικασία αντίστοιχα: μια βάση γνώσης, μια βάση κανόνων και ένα σημείο αποθήκευσης συμβάντων.

3 Έμπειρα Συστήματα

Τα Έμπειρα Συστήματα (Expert Systems) αποτελούν έναν από τους κλάδους της Τεχνητής Νοημοσύνης (Artificial Intelligence). Το πεδίο της Τεχνητής Νοημοσύνης (ΤΝ) έχει ιστορία περίπου 50 ετών, αν και εξαρτάται από το πότε τοποθετείται χρονικά η έναρξη της. Είναι γεγονός ότι η έρευνα σε ΤΝ άρχισε πολύ πριν επινοηθεί ο όρος «Τεχνητή Νοημοσύνη» από τον John McCarthy [28], ενώ το πρώτο επίσημο διεθνές συνέδριο σε ΤΝ διεξάχθηκε το 1969 στη Βόρειο Αμερική. Αν και δεν υπάρχει ένας γενικά αποδεκτός ορισμός για την ΤΝ, οι Luger και Stubblefield [29] έχουν προτείνει τον ακόλουθο:

> Τεχνητή Νοημοσύνη είναι ένας κλάδος της Πληροφορικής, ο οποίος ασχολείται με την αυτοματοποίηση ευφυούς συμπεριφοράς.

Το αδύνατο σημείο του ορισμού είναι ότι εξαρτάται από το τι είναι «ευφυΐα» ή «ευφυής συμπεριφορά», κάτι για το οποίο δεν υπάρχει σύγκλιση απόψεων.

Ένας εναλλακτικός ορισμός, ο οποίος έχει προταθεί από διάφορους ερευνητές, είναι ευρέως αποδεκτός και είναι αρκετά συγκεκριμένος, είναι ο ακόλουθος :

> Τεχνητή Νοημοσύνη είναι η ανάπτυξη υπολογιστικών συστημάτων για την επίλυση δύσκολων προβλημάτων, τα οποία δεν μπορούν να επιλυθούν με την εξαντλητική εξέταση όλων των πιθανών λύσεων μια και αυτές μπορεί να είναι πάρα πολλές.

Η έμφαση εδώ δίνεται σε δύσκολα προβλήματα, τα οποία παραδοσιακές υπολογιστικές μέθοδοι, δηλαδή καθαρά αλγοριθμικές μέθοδοι, είναι ανίκανες να επιλύσουν, τουλάχιστο μέσα σε λογικά χρονικά πλαίσια. Η σύνδεση του ορισμού με την «ευφυΐα» απορρέει από τη γενική αποδοχή ότι ο άνθρωπος που μπορεί να επιλύσει τέτοια δύσκολα προβλήματα, αποδοτικά και αποτελεσματικά, χαρακτηρίζεται από ευφυΐα.

Μερικές χαρακτηριστικές περιοχές επιστημονικού ενδιαφέροντος της ΤΝ είναι :

- Επίλυση Προβλημάτων
- Επεξεργασία Φυσικής Γλώσσας
- Τεχνητή Όραση
- Αυτόνομα Robot
- Έμπειρα Συστήματα και Συστήματα Γνώσης
- Ευφυείς πράκτορες (Agents)
- Ευφυείς υπηρεσίες διαδικτύου και σημασιολογικά δίκτυα (semantic web)

Ο τομέας της ΤΝ που ασχολείται με την ανάπτυξη έμπειρων συστημάτων ονομάζεται τεχνολογία της γνώσης (knowledge engineering).

Η ΤΝ αρχικά στόχευε στη δημιουργία συστημάτων γενικής επίλυσης προβλημάτων (general problem solvers). Μέσα από αυτή την προσπάθεια, και συγκεκριμένα την αποτυχία της, αποδείχτηκε ότι η αποδοτική και αποτελεσματική επίλυση ρεαλιστικών προβλημάτων είναι άμεσα συνδεμένη με τη χρήση συγκεκριμένης γνώσης. Αυτό οδήγησε στη δημιουργία των συστημάτων βάσης γνώσης (knowledge based systems). Έμπειρα συστήματα είναι ουσιαστικά συστήματα βάσης γνώσης, τα οποία μπορούν να οργανωθούν ως συστήματα παραγωγής, πλαισίων κτλ. Στα Έμπειρα Συστήματα υπάρχει σαφής και γενικά αποδεκτός ορισμός ως προς τη φύση και τη λειτουργία τους. Ο ορισμός των Έμπειρων Συστημάτων κατά την Βρετανική Εταιρία Υπολογιστών (The British Computer Society's Specialist Group on Expert Systems) είναι [3w31]:

An expert system is regarded as the embodiment within a computer of a knowledge-based component from an expert skill in such a form that the system can offer intelligent advice or take an intelligent decision about a processing function. A desirable additional characteristic, which many would consider fundamental, is the capability of the system, on demand, to justify its own line of reasoning in a manner directly intelligible to the enquirer.

Ένα έμπειρο σύστημα θεωρείται η ενσωμάτωση μέσα σε έναν υπολογιστή μιας βασισμένης-στη-γνώση συνιστώσας από την ικανότητα ενός ειδικού, με μια τέτοια μορφή ώστε το σύστημα να μπορεί να προσφέρει ευφυείς συμβουλές ή να πάρει μια ευφυή απόφαση για κάποια λειτουργία επεξεργασίας. Ένα πρόσθετο επιθυμητό χαρακτηριστικό, που πολλοί θα θεωρούσαν θεμελιώδες, είναι η ικανότητα του συστήματος, μετά από απαίτηση, να δικαιολογεί τη συλλογιστική του πορεία κατά τρόπο άμεσα κατανοητό στον ερωτώντα.

Εμβαθύνοντας τον παραπάνω ορισμό μπορούμε να εξετάσουμε λεπτομερέστερα τα στοιχεία αυτά που χαρακτηρίζουν ένα έμπειρο σύστημα. Καταρχήν είναι υπολογιστικά συστήματα τα οποία, σε αντίθεση με τα συμβατικά, επιδεικνύουν νοήμονα συμπεριφορά σε συγκεκριμένους τομείς και διαδικασίες, ανάλογη με αυτή ενός ανθρώπου εμπειρογνώμονα με ειδικότητα στον ίδιο τομέα. Απαραίτητη προϋπόθεση είναι λοιπόν η εμπειρική γνώση. Προκειμένου να εκμεταλλευτούμε την ανθρώπινη εμπειρία πρέπει η γνώση αυτή να κωδικοποιηθεί. Η γνώση αυτή προέρχεται από κάποιον άνθρωπο-ειδικό σε έναν εξειδικευμένο τομέα. Για αυτό το λόγο και χρησιμοποιούμε προγραμματιστικές τεχνικές τεχνητής νοημοσύνης και ειδικότερα αυτές της επίλυσης προβλημάτων. Ο σκοπός των έμπειρων συστημάτων είναι διπτός. Μπορούν να χρησιμοποιηθούν από κάποιον άνθρωπο μη-ειδικό, προκειμένου να παρέχει λύσεις σε συγκεκριμένα προβλήματα, ενώ μπορούν να λειτουργήσουν και συμβουλευτικά, από έναν άνθρωπο-ειδικό, ο οποίος καλείται να πάρει κάποια απόφαση.

Τυπικές κατηγορίες εφαρμογών:

Σύμφωνα με τους παραπάνω σκοπούς που πρέπει να εξυπηρετούν τα έμπειρα συστήματα, μερικά από τα πιο χαρακτηριστικά πεδία εφαρμογών τους είναι:

- Στην ερμηνεία δεδομένων (π.χ. ηχητικών ή ηλεκτρομαγνητικών σημάτων)
- Στη διάγνωση δυσλειτουργιών (π.χ. βλαβών σε μηχανήματα ή ασθενειών σε ανθρώπους)
- Στη δομική ανάλυση σύνθετων αντικειμένων (π.χ. χημικών ενώσεων)
- Στη διαμόρφωση σύνθετων αντικειμένων (π.χ. πολύπλοκων υπολογιστικών συστημάτων)
- Στην πρόβλεψη πιθανών μελλοντικών επιπτώσεων με βάση δεδομένες καταστάσεις.
- Στην κατανόηση, αξιολόγηση και διόρθωση απάντησης μαθητών σε εκπαιδευτικά προβλήματα.
- Στην παρακολούθηση καταστάσεων (monitoring).
- Στην επιδιόρθωση λαθών (repair-remedy).
- Στην ανάπτυξη και εκτέλεση σχεδίων (πλάνων) για τη διαχείριση βλαβών.
- Στην περιγραφή αντικειμένων και καταστάσεων βάσει δεδομένων από παρατηρήσεις.
- Στην ικανοποίηση απαιτήσεων και περιορισμών για τη συναρμολόγηση εξαρτημάτων.
- Στον έλεγχο της συμπεριφοράς ενός συστήματος, που περιλαμβάνει πολλά από τα παραπάνω.

Ταξινόμηση έμπειρων συστημάτων

Η ταξινόμηση των έμπειρων συστημάτων διακρίνεται σε τρία επίπεδα τεχνολογίας:

- **Εξειδικευμένα έμπειρα συστήματα** - χρησιμοποιούνται για την παροχή συμβουλών μίας συγκεκριμένης θεματικής περιοχής.
- **Κελύφη** - ένα κέλυφος είναι ένα πακέτο το οποίο επιτρέπει την σύντομη και εύκολη ανάπτυξη έμπειρων συστημάτων. Ένα κέλυφος παρέχει τα περισσότερα υποσυστήματα ενός έμπειρου συστήματος, εκτός από την βάση γνώσης.

- **Εργαλεία έμπειρων συστημάτων** - αποτελούν το χαμηλότερο επίπεδο. Πρόκειται για εργαλεία που διευκολύνουν την ανάπτυξη είτε εξειδικευμένων συστημάτων είτε κελύφων έμπειρων συστημάτων.

Σε επόμενες παραγράφους ακολουθεί λεπτομερέστερη περιγραφή των τεχνολογιών που προαναφέρθηκαν.

Χαρακτηριστικά Έμπειρων Συστημάτων

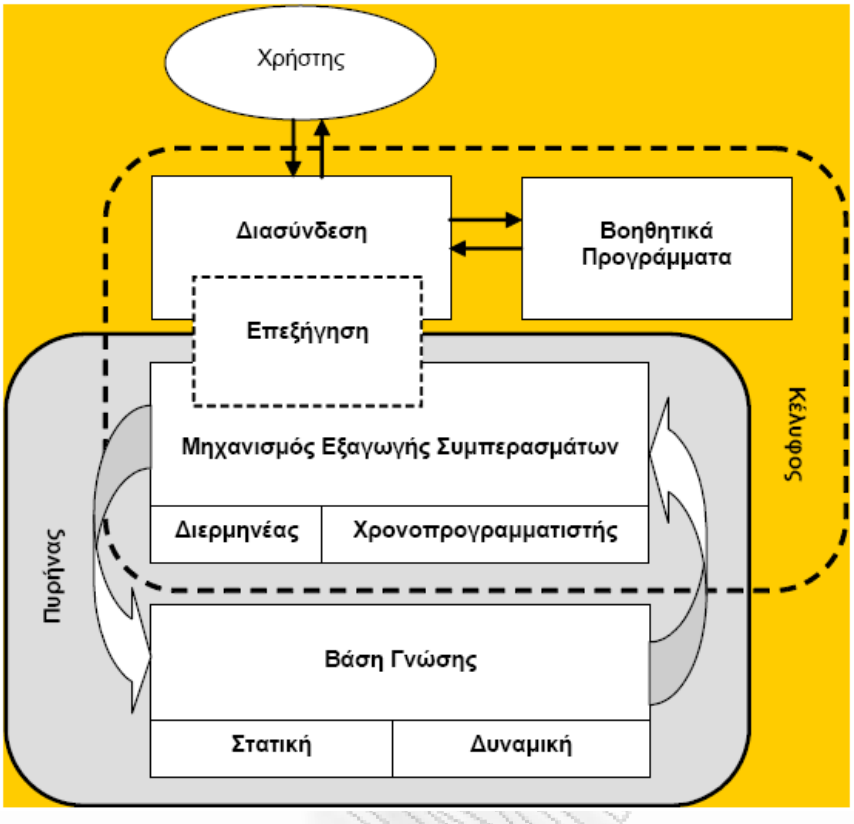
Επιγραμματικά τα χαρακτηριστικά των Έμπειρων Συστημάτων είναι:

- Προσομοιώνουν ανθρώπινο συλλογισμό και γνώση.
- Επιλύουν προβλήματα κάνοντας χρήση ευρετικών ή προσεγγιστικών μεθόδων.
- Ασχολούνται με προβλήματα ρεαλιστικής πολυπλοκότητας, η αποδοτική και αποτελεσματική επίλυση των οποίων εκ μέρους του ανθρώπου συνεπάγεται εμπειρογνωμοσύνη.
- Επιδεικνύουν υψηλά επίπεδα απόδοσης και σε ταχύτητα και σε ορθότητα λύσεων.
- Συνδιαλέγονται με το χρήστη.
- Επεξηγούν και τεκμηριώνουν τις εισηγήσεις τους.
- Αυτοαναπτύσσονται.

Συμπερασματικά, τα Έμπειρα Συστήματα αποτελούν ικανούς επιλυτές προβλημάτων (competent problem solvers) για τους συγκεκριμένους, εξειδικευμένους τομείς.

3.1 Αρχιτεκτονική Έμπειρων Συστημάτων

Ένα έμπειρο σύστημα διακρίνεται κυρίως από τον πυρήνα και το κέλυφος όπως φαίνεται και στο σχήμα που ακολουθεί [12]. Ο πυρήνας αποτελείται από τη Βάση Γνώσης (Knowledge Base) και τον Μηχανισμό Εξαγωγής Συμπερασμάτων (Inference Engine). Εάν διαχωρίσουμε τη Βάση Γνώσης από το σύστημα τότε το υπόλοιπο μέρος του πυρήνα μαζί με το σύστημα διεπαφής και τα βοηθητικά προγράμματα συγκροτούν το κέλυφος. Ο διαχωρισμός της Βάσης Γνώσης είναι ένα χαρακτηριστικό γνώρισμα της Αρχιτεκτονικής των Έμπειρων Συστημάτων που προσφέρει διαφάνεια και ευελιξία, ενώ το ίδιο σύστημα με διαφορετική Βάση Γνώσης μπορεί να επεκτείνει το πεδίο εφαρμογών του. Άλλωστε αυτή η δυνατότητα των Έμπειρων Συστημάτων χρησιμοποιείται ως μεθοδολογία ανάπτυξής τους και αναλύεται στην παράγραφο για τα Κελύφη. Τα επιμέρους τμήματα της αρχιτεκτονικής αναλύονται στις επόμενες παραγράφους.



Σχήμα 14: Αρχιτεκτονική ενός Έμπειρου Συστήματος

3.1.1 Βάση Γνώσης (Knowledge Base)

Η Βάση Γνώσης είναι μια Βάση Δεδομένων που σαν στόχο δεν έχει την διαχείριση δεδομένων αλλά τη διαχείριση γνώσης. Υποστηρίζει με αυτή την έννοια τη συλλογή, αναπαράσταση, οργάνωση και ανάκτηση της γνώσης. Για την αναπαράσταση της γνώσης μπορεί να χρησιμοποιηθούν πλαίσια, σημασιολογικά δίκτυα, κανόνες κ.α. Σε σύστημα παραγωγής η γνώση αποθηκεύεται στη βάση με κανόνες της μορφής IF <Συνθήκη> THEN <Ενέργεια>.

Οι Βάσεις Γνώσεις διακρίνονται σε δυο κατηγορίες:

- Σε Βάσεις οι οποίες αποθηκεύουν τη γνώση σε μορφή που να μπορεί να αναγνωσθεί από μηχανές: Προτείνεται σε περιπτώσεις που θα πρέπει να εφαρμοστεί στα δεδομένα επαγωγική συλλογιστική. Η δομή των δεδομένων μπορεί να περιγραφεί με οντολογίες και οι σχέσεις μεταξύ τους με λογικούς τελεστές. Τέτοια τύπου Βάσεις χρησιμοποιούνται και σε Σημασιολογικά Δίκτυα.
- Σε Βάσεις που αποθηκεύουν τη γνώση σε μορφή που μπορεί να αναγνωσθεί από τον άνθρωπο: Χρησιμοποιούνται σε περιπτώσεις που απαιτείται η διαμοίραση της γνώσης μεταξύ συναδέλφων σε μια επιχείρηση. Τα δεδομένα μπορεί να περιλαμβάνουν εγχειρίδια χρήσης, άρθρα, πληροφορίες εντοπισμού και αντιμετώπισης λαθών. Συνήθως για την ανάκτηση αυτών των δεδομένων χρησιμοποιούνται μηχανές αναζήτησης ή γίνονται μέσω ενός συστήματος ταξινόμησης.

Όπως παρατηρούμε και από το σχήμα, η Βάση Γνώσης διακρίνεται επίσης σε:

Στατική: Περιέχει διαδικασίες, κανόνες και πλαίσια που περιγράφουν το πρόβλημα και τις γνωσιολογικές διαδικασίες επίλυσής τους (αρχικά δεδομένα) και δεν μεταβάλλεται κατά τη διάρκεια εκτέλεσης του προγράμματος.

Δυναμική: Περιέχει ενδιάμεσα συμπεράσματα που παράγονται κατά την εκτέλεση του προγράμματος, καθώς και στην τελική λύση του προβλήματος. Η μνήμη αυτή ονομάζεται και μνήμη εργασίας (working memory) ή λειτουργική μνήμη.

3.1.2 Μηχανισμός Εξαγωγής Συμπερασμάτων (Inference Engine)

Ο μηχανισμός εξαγωγής συμπερασμάτων αποτελεί τον «εγκέφαλο» του έμπειρου συστήματος. Η λειτουργία του είναι η διαχείριση της Βάσης Γνώσης προκειμένου να καταλήξει σε συμπεράσματα εμπλουτίζοντας με αυτό τον τρόπο το σύστημα με νέα γνώση. Η παραγωγή νέας γνώσης επιτυγχάνεται μέσω αντιστοίχισης της γνώσης που περιέχεται στη Βάση Γνώσης σε κάποια μορφή αναπαράστασης και των γεγονότων που τροφοδοτούν το σύστημα. Η διαδικασία αυτή λέγεται αντιστοίχιση προτύπου (pattern matching). Μερικοί από τους πιο γνωστούς αλγόριθμους αντιστοίχησης προτύπου είναι οι εξής:

- Linear
- Rete
- Treat
- Leaps

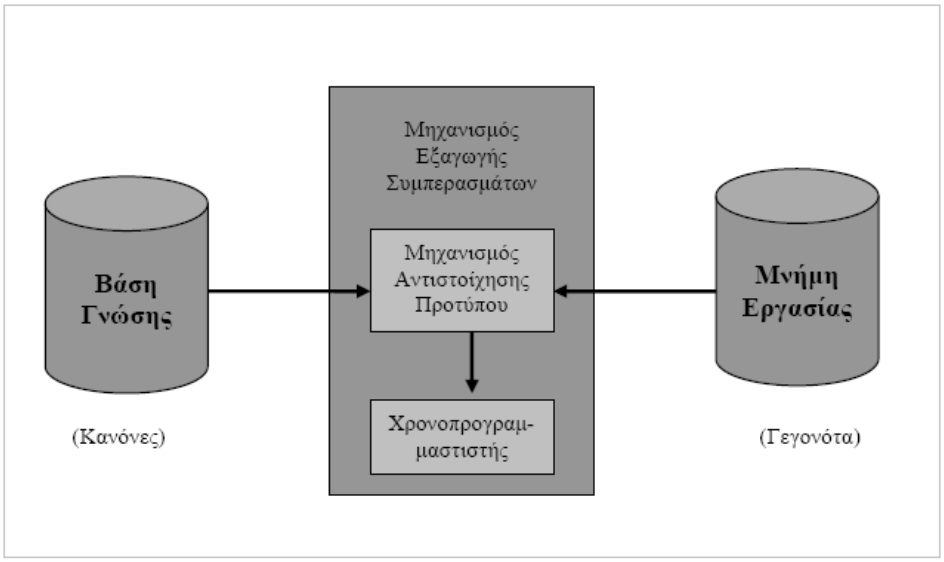
Ο μηχανισμός εξαγωγής συμπερασμάτων αποτελείται από δύο τμήματα:

- Τον Διερμηνευτή: Αποτελεί ουσιαστικά τον μηχανισμό αντιστοίχισης γνώσης
- Τον Χρονοπρογραμματιστή: Καθορίζει την σειρά με την οποία θα ενεργοποιηθούν οι κανόνες.

Σε ένα σύστημα παραγωγής οι κανόνες είναι αποθηκευμένες στη Βάση Γνώσης και τα συμβάντα στη μνήμη εργασίας. Σε συστήματα τα οποία υποστηρίζουν πολλούς κανόνες και ο όγκος των γεγονότων είναι πολύ μεγάλος είναι συχνό το φαινόμενο ταυτόχρονης ενεργοποίησης περισσότερων του ενός κανόνων. Στην περίπτωση αυτή τίθεται ζήτημα συγκρούσεων ως προς το ποιος κανόνας θα ενεργοποιηθεί πρώτος. Ο χρονοπρογραμματιστής αναλαμβάνει να καθορίσει τη σειρά με την οποία θα ενεργοποιηθούν οι κανόνες εφαρμόζοντας μια στρατηγική επίλυσης συγκρούσεων (conflict resolution strategy).

Υπάρχουν διάφοροι τρόποι με τους οποίους μπορεί να υλοποιηθεί ο χρονοπρογραμματιστής. Συνήθως υλοποιείται με μια ουρά (queue), που είναι γνωστή με το όρο ατζέντα (agenda). Η agenda είναι η λίστα των κανόνων που η μηχανή εξαγωγής συμπερασμάτων κατέταξε με βάση την προτεραιότητά τους και των οποίων τα πρότυπα (συνθήκες) ικανοποιούνται. Η σειρά ενεργοποίησης ή πυροδότησης (firing) των κανόνων καθορίζεται κάθε φορά μέσα στο τρέχον σύνολο συγκρούσεων, κάνοντας έτσι χρήση τοπικών (local) και όχι καθολικών (global) κριτήριων, χωρίς να είναι δυνατή η πρόβλεψη κατοπινών καταστάσεων (look-ahead). Εκτός από τις συνηθισμένες στρατηγικές επίλυσης συγκρούσεων (random, ordering, recency, specificity, refractoriness), μπορεί να χρησιμοποιούνται και μετα-κανόνες, οι οποίοι αποφασίζουν ποιοι κανόνες θα επιλεγούν βάσει της τρέχουσας κατάστασης της μνήμης εργασίας.

Η αλληλεπίδραση της Βάσης Γνώσης και της Μνήμης Εργασίας στον Μηχανισμό Εξαγωγής Συμπερασμάτων σε ένα σύστημα παραγωγής παρουσιάζεται στο σχήμα που ακολουθεί [2]:

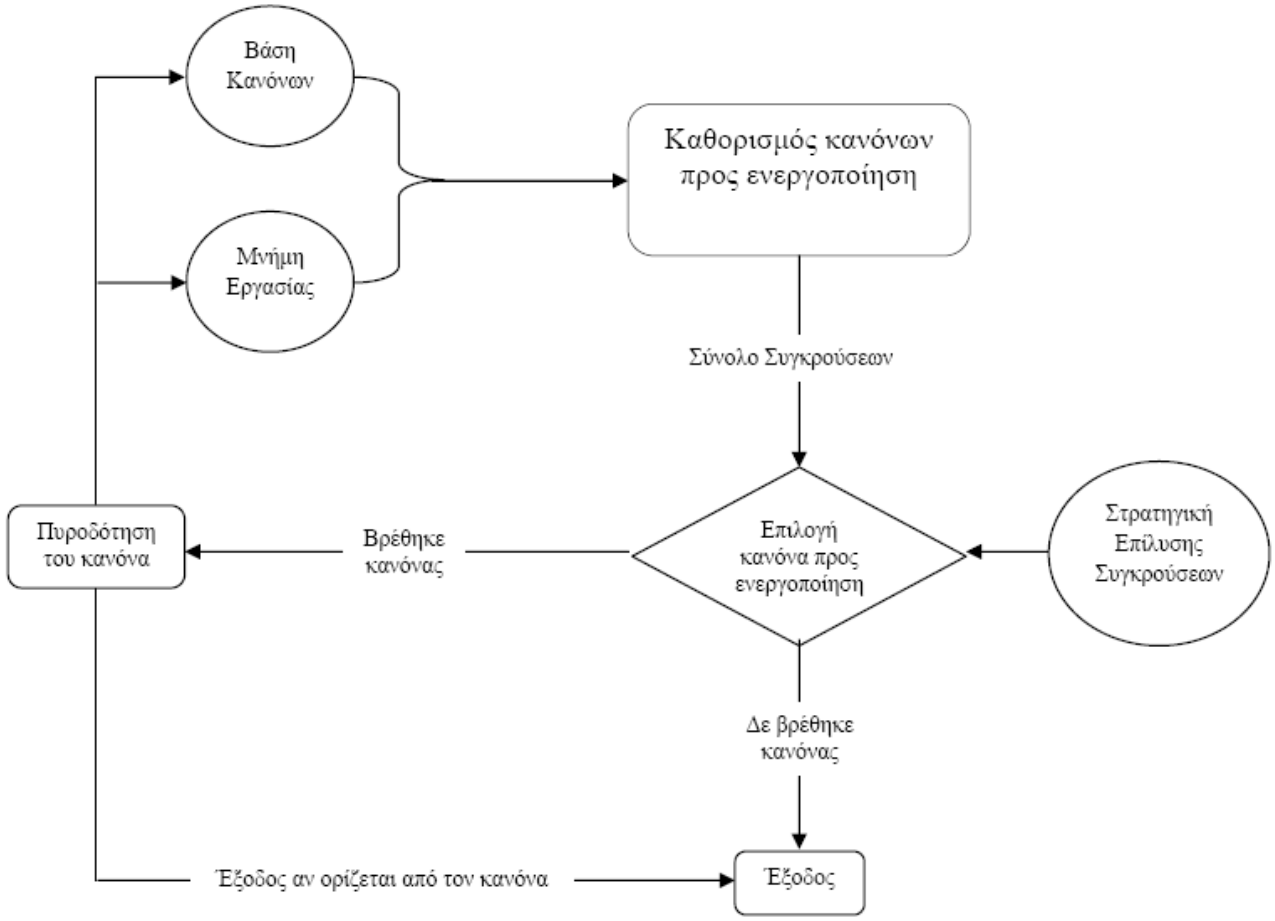


Σχήμα 15: Μηχανισμός Εξαγωγής Συμπερασμάτων

Υπάρχουν δύο βασικές στρατηγικές εκτέλεσης των κανόνων προκειμένου το σύστημα να καταλήξει σε κάποιο συμπέρασμα.

Ορθή Αλυσίδωση (Forward Chaining)

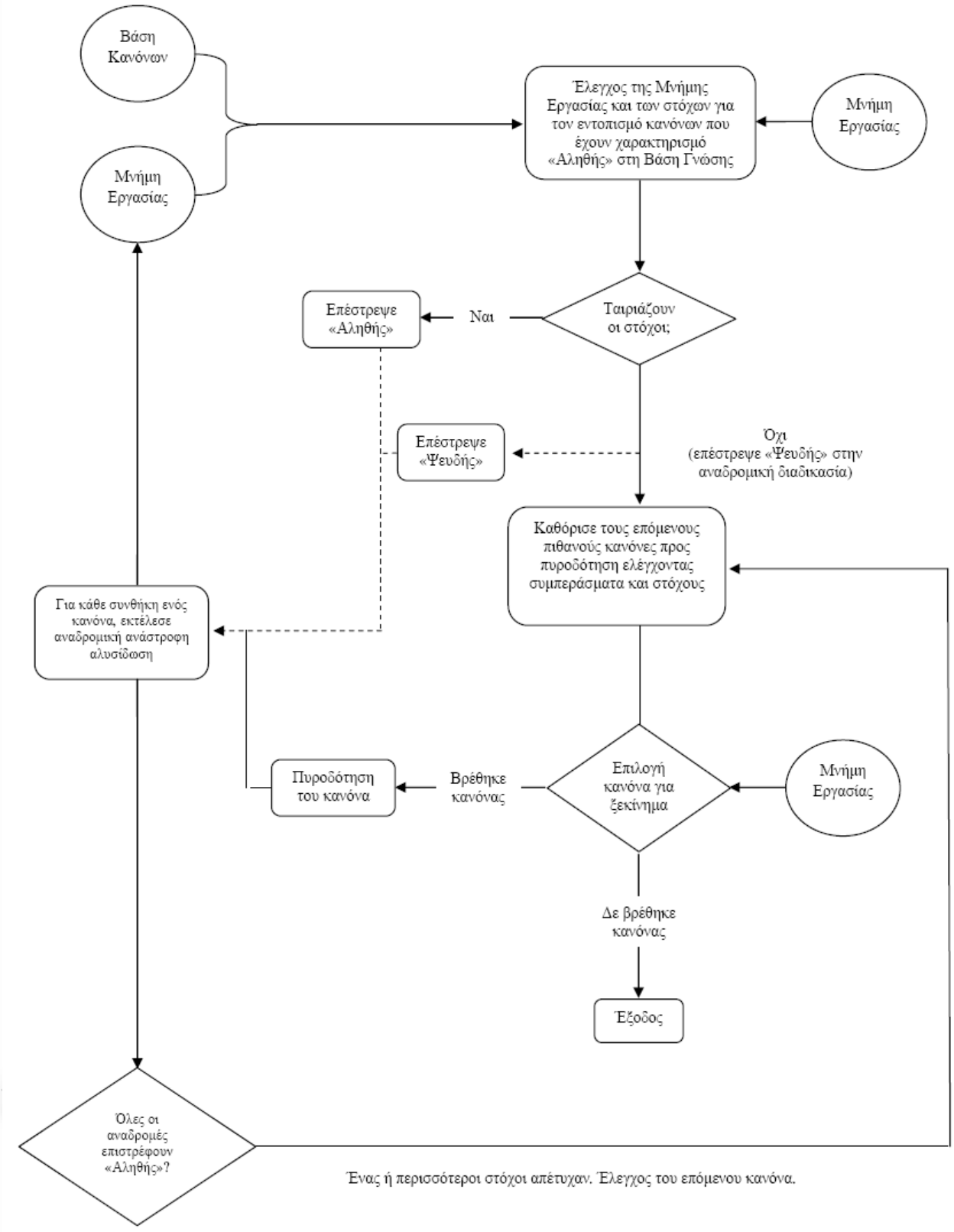
Η στρατηγική αυτή ονομάζεται και οδηγούμενη-από-τα-δεδομένα (data-driven) [3w10]. Τα δεδομένα εισέρχονται στη μνήμη εργασίας και στη συνέχεια ενεργοποιούνται κάποιοι κανόνες οι οποίοι καταλήγουν σε κάποιο συμπέρασμα. Τα δεδομένα είναι αυτά που εκκινούν τη διαδικασία και τελικά καταλήγουν σε ένα συμπέρασμα. Ο τρόπος αυτός ενδείκνυται στις περιπτώσεις που υπάρχουν λίγες υποθέσεις και πολλά συμπεράσματα. Ακολουθεί η γραφική αναπαράσταση της διαδικασίας αυτής [5]:



Σχήμα 16: Ορθή Αλυσίδωση

Ανάστροφη Αλυσίδωση (Backward Chaining)

Η Ανάστροφη Αλυσίδωση χαρακτηρίζεται και ως οδηγούμενη-από-το-αποτέλεσμα (goal-driven) [3w9]. Αρχικά επιλέγεται ένα συμπέρασμα και ο μηχανισμός προσπαθεί να ικανοποιήσει τις υποθέσεις. Εάν δεν καταλήξει στην ικανοποίηση του συμπεράσματος τότε ο μηχανισμός προσπαθεί να ικανοποιήσει κάποιους δευτερεύοντες στόχους που αποτελούν υποδιαίρεση του αρχικού στόχου. Η διαδικασία τερματίζεται όταν ικανοποιηθεί το αρχικό συμπέρασμα ή όταν τελειώσουν οι δευτερεύοντες στόχοι. . Ο τρόπος αυτός είναι προτιμότερος στην περίπτωση πολλών υποθέσεων και λιγότερων συμπερασμάτων. Η PROLOG αποτελεί χαρακτηριστικό παράδειγμα μηχανισμού ανάστροφης αλυσίδωσης. Ακολουθεί η γραφική αναπαράσταση της διαδικασίας αυτή [5]:



Σχήμα 17: Ανάστροφη Αλυσίδωση

Υπάρχουν συστήματα τα οποία υποστηρίζουν και τις δύο στρατηγικές εκτέλεσης κανόνων και στην περίπτωση αυτή τα συστήματα αυτά ονομάζονται υβριδικά ή αμφίδρομης αλυσίδωσης (sideways chaining).

3.1.3 Μηχανισμός Επεξήγησης

Ο μηχανισμός επεξήγησης αποτελεί τμήμα της αρχιτεκτονικής των πιο εξελιγμένων Έμπειρων Συστημάτων. Ένα έμπειρο σύστημα, ως σύστημα εξομοίωσης του ανθρώπινου συλλογισμού, πρέπει να διαθέτει μια μονάδα ελέγχου και επαλήθευσης των συμπερασμάτων τα οποία παράγει. Ο έλεγχος αφορά στην ορθότητα των τελικών συμπερασμάτων αλλά και της ίδιας της διαδικασίας βάσει της οποία παρήχθησαν τα τελικά συμπεράσματα.

Η διαδικασία ελέγχου για την ορθότητα των αποτελεσμάτων καλείται έλεγχος αξιοπιστίας (validation check) και πραγματοποιείται μέσω προσομοίωσης συγκεκριμένων καταστάσεων όπου είναι γνωστά τα τελικά συμπεράσματα. Στην περίπτωση που το σύστημα καταλήξει σε διαφορετικά αποτελέσματα τότε επιβάλλεται ο επανασχεδιασμός του. Με αυτή την έννοια ο έλεγχος αξιοπιστίας μπορεί να λειτουργήσει και σαν εκσφαλματωτής (debugger), όπου ο μηχανικός γνώσης έχει τη δυνατότητα να ελέγξει τη σωστή εφαρμογή της γνώσης. Εκτός από τα τελικά αποτελέσματα εξετάζεται και η συνολική διαδικασία συλλογισμού μέχρι τα τελικά αποτελέσματα συμπεριλαμβάνοντας και τα ενδιάμεσα αποτελέσματα. Η διαδικασία αυτή καλείται έλεγχος επαλήθευσης (verification check), όπου διαπιστώνεται κατά πόσο μεταφέρθηκε σωστά η συλλογιστική του ειδικού του τομέα (domain expert) από τον μηχανικό γνώσης.

Ο μηχανισμός επεξήγησης αποτελεί ένα ενδιάμεσο τμήμα μεταξύ του τμήματος της διασύνδεσης και του μηχανισμού εξαγωγής συμπερασμάτων. Η επικοινωνία με το τμήμα διασύνδεσης είναι απαραίτητη για την προβολή των αποτελεσμάτων, ενώ η αλληλεπίδραση με τον μηχανισμό εξαγωγής συμπερασμάτων επειδή η συλλογιστική που έχει υλοποιηθεί στο σύστημα είναι προβολή του τρόπου εκτέλεσης των κανόνων. Η διαφάνεια στην αρχιτεκτονική των έμπειρων συστημάτων οφείλεται στο μηχανισμό επεξήγησης που επιτρέπει την επίβλεψη όλης της αλυσίδας γεγονότων που οδηγούν στην παραγωγή συμπερασμάτων.

3.1.4 Διασύνδεση

Ο όρος διασύνδεση αναφέρεται τόσο στο περιβάλλον διεπαφής με τον χρήστη όσο και στην επικοινωνία του συστήματος με βοηθητικά προγράμματα. Σχετικά με το πρώτο, αφορά στη δυνατότητα του έμπειρου συστήματος να συνδιαλέγεται με τον χρήστη. Η τροφοδότηση του συστήματος με δεδομένα αλλά και τα συμπεράσματα του έμπειρου συστήματος εμφανίζονται μέσω του συστήματος διασύνδεσης. Το ίδιο γραφικό περιβάλλον μπορεί να χρησιμοποιήσει και ο μηχανικός γνώσης προκείμενου να εμπλουτίσει ή να μεταβάλλει τη γνώση του συστήματος. Η διασύνδεση αφορά παράλληλα και την επικοινωνία του συστήματος με βοηθητικά προγράμματα όπως στατιστικά πακέτα, βάσεις δεδομένων, συστήματα γραφική απεικόνισης. Η ολοκλήρωση ενός έμπειρου συστήματος με άλλα υπολογιστικά συστήματα συναντάται κυρίως σε εμπορικές εφαρμογές.

Πρέπει να σημειωθεί ότι στον όρο διασύνδεση περιλαμβάνεται και η διασύνδεση του έμπειρου συστήματος με αισθητήρες (sensors) ή μηχανισμούς δράσης (effectors) οι οποίοι τροφοδοτούν αυτόματα το σύστημα με δεδομένα. Θα ακολουθήσει περαιτέρω ανάλυση στη διασύνδεση με αισθητήρες στην ανάλυση που θα ακολουθήσει σε επόμενες παραγράφους για την αρχιτεκτονική συστημάτων καθοδηγούμενων από γεγονότα (event-driven architecture).

3.2 Διαδικασία Ανάπτυξης Έμπειρων Συστημάτων

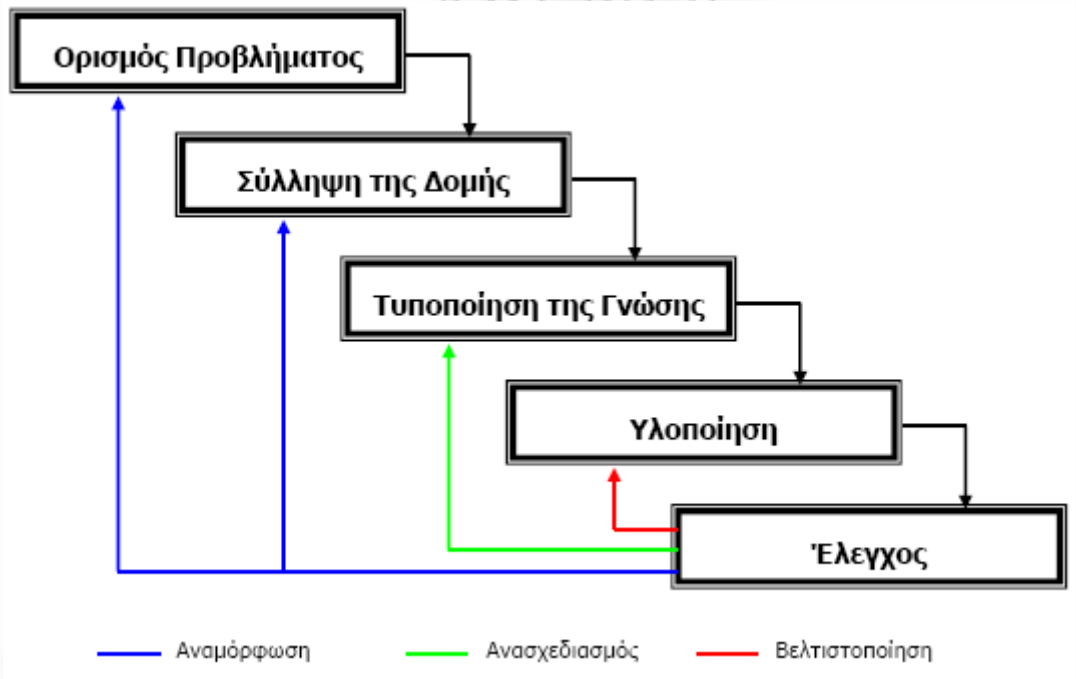
Στη διαδικασία ανάπτυξης ενός έμπειρου συστήματος εμπλέκονται κυρίως δύο ομάδες ατόμων. Αυτοί που κατέχουν τη γνώση και έχουν εμπειρία σε ένα συγκεκριμένο τομέα οι οποίοι αποτελούν τους Ειδικούς Τομέα (Human Domain Experts) και αυτοί που βασισμένοι στην πρώτη ομάδα θα υλοποιήσουν το σύστημα και είναι οι Μηχανικοί Γνώσης (Knowledge Engineers). Μία τρίτη ομάδα είναι οι τελικοί χρήστες (Users). Ο ειδικός στον κάθε τομέα είναι υπεύθυνος στο να παρέχει στον μηχανικό γνώσης την εξειδικευμένη γνώση του, τις

μεθοδολογίες επίλυσης των προβλημάτων (πχ. δένδρα απόφασης, στατιστικά μοντέλα) που χρησιμοποιεί και έχει αναπτύξει στο συγκεκριμένο πεδίο εφαρμογής. Ο ρόλος του μηχανικού γνώσης είναι να χρησιμοποιήσει τη γνώση που του παρείχε ο ειδικός προκειμένου να επιλέξει το λογισμικό του έμπειρου συστήματος και τον τρόπο αναπαράστασης της γνώσης στη Βάση Γνώσης του.

Στο πρώτο στάδιο της διαδικασίας ο μηχανικός γνώσης αποκτά μια πρώτη επαφή με το επιστημονικό πεδίο που πρέπει να αναλύσει, κυρίως μέσω της βιβλιογραφίας, και στη συνέχεια ακολουθεί τη διαδικασία πρόσληψης του μηχανισμού επίλυσης του προβλήματος από τον Ειδικό (expert's problem solving knowledge). Υπάρχουν διάφορες μεθοδολογίες βάσει των οποίων ο μηχανικός γνώσης προσπαθεί να εκμαιεύσει τη γνώση από τον ειδικό. Ο πιο διαδεδομένος και αποδοτικός τρόπος εκμαίευσης γνώσης είναι η διαδικασία των συνεντεύξεων, με τη μορφή των μη-δομημένων, ημι-δομημένων ή δομημένων συνεντεύξεων. Μια άλλη κλασσική μέθοδος είναι η χρήση πλεγμάτων ρεπερτορίων (repertory grids). Με τη μέθοδο αυτή καταγράφονται όλες οι έννοιες που πρέπει να μοντελοποιηθούν και κάθε έννοια αποκτά ένα βάρος, έναν αριθμό, και επαφίεται στον ειδικό γνώσης να ορίσει τα βάρη αυτά. Οι τιμές αυτές μαζί με τα βάρη τους δημιουργούν ένα πλέγμα. Συνήθως χρησιμοποιείται η μέθοδος αυτή σε δεδομένα που χαρακτηρίζονται από μεγάλο βαθμό υποκειμενικότητας. Εκτός από τις κλασσικές μεθόδους μπορούν να χρησιμοποιηθούν και αυτόματες μέθοδοι όπου το σύστημα χρησιμοποιεί τεχνικές μηχανικής μάθησης ή μπορεί να χρησιμοποιηθεί και ειδικό λογισμικό (πχ το OPAL).

Συνήθως σε δεύτερο στάδιο υλοποιείται ένα πρωτότυπο προκειμένου να γίνει μια αρχική γρήγορη αξιολόγηση του συστήματος και στη συνέχεια διαδοχικοί έλεγχοι και βελτιώσεις θα ολοκληρώσουν το σύστημα και θα εμπλουτίσουν τη βάση γνώσης.

Η τυποποίηση αυτής της διαδικασίας περιγράφεται στο σχήμα που ακολουθεί:



Σχήμα 18: Διαδικασία Ανάπτυξης ενός Έμπειρου Συστήματος

Όπως παρατηρούμε η διαδικασία αποτελείται από 5 διακριτά και αλληλοεξαρτώμενα στάδια:

Στάδιο 1^{ο}: Ο ορισμός του προβλήματος

Πραγματοποιείται ο προσδιορισμός των δεδομένων, των στόχων και των διαδικασιών επίλυσης.

Στάδιο 2^{ο}: Σύλληψη της Δομής

Προσδιορισμός των εννοιών και των σχέσεων μεταξύ των εννοιών αυτών. Σύνθεση των κανόνων παραγωγής που περιγράφουν τις προσδιορισθείσες σχέσεις.

Στάδιο 3^{ο}: Τυποποίηση της Γνώσης

Τα περιγραφικά δεδομένα και οι κανόνες παραγωγής που προσδιορίσθηκαν στο προηγούμενο στάδιο αναπαρίσταται στις δομές γνώσης ενός έμπειρου συστήματος.

Στάδιο 4^{ο}: Υλοποίηση

Προγραμματισμός του συστήματος είτε μέσω κάποιας γλώσσας προγραμματισμού είτε μέσω κάποιου εξειδικευμένου λογισμικού (κελύφη). Σε παράγραφο που ακολουθεί θα γίνει περαιτέρω ανάλυση των εργαλείων ανάπτυξης έμπειρων συστημάτων.

Στάδιο 5^{ο}: Έλεγχος

Είναι το τελικό στάδιο όπου αξιολογείται το σύστημα και ανιχνεύονται περαιτέρω μετατροπές και βελτιώσεις. Επιπλέον, σκοπός του ελέγχου είναι να εξασφαλιστεί η ανθεκτικότητα και η δυνατότητα απόκρισης του συστήματος σε μη-προσδοκώμενα δεδομένα.

4 Αρχιτεκτονική Συστήματος Παραγωγής για Επεξεργασία Σύνθετων Συμβάντων

Στην πορεία ανάπτυξης των Έμπειρων Συστημάτων υπήρχαν πάντα δύο βασικές προκλήσεις. Η πρώτη είναι να μπορούν τα Συστήματα αυτά να επεξεργάζονται πληροφορίες σε πραγματικό χρόνο. Για αυτό το σκοπό έχουν προταθεί πολλά συστήματα τα οποία συγκαταλέγονται στη γενικότερη προσπάθεια δημιουργίας real time έμπειρων συστημάτων. Η δεύτερη πρόκληση είναι η «τροφοδοσία» των έμπειρων συστημάτων με πληροφορίες από ένα μεγάλο σύνολο ετερογενών πηγών προκειμένου και να επιλύονται μεγαλύτερου εύρους προβλήματα αλλά και επιπροσθέτως το σύστημα να μπορεί να λαμβάνει υπόψη του όσο το δυνατόν περισσότερη πληροφορία.

Και τα δύο αυτά καίρια ζητήματα αντιμετωπίζονται στην Επεξεργασία Σύνθετων Συμβάντων και στην EDA Αρχιτεκτονική που αναλύσαμε διεξοδικά στο προηγούμενο κεφάλαιο. Καταρχήν, όσον αφορά στην επεξεργασία σε πραγματικό χρόνο, η EDA αρχιτεκτονική διασφαλίζει την επεξεργασία των συμβάντων τη στιγμή που λαμβάνουν χώρα, ενώ η Επεξεργασία Σύνθετων Συμβάντων (CEP) μπορεί να αναγνωρίζει πρότυπα από πληροφορίες οι οποίες προέρχονται από πολλές πηγές. Ο συγκερασμός λοιπόν την τεχνολογίας για την επεξεργασία συμβάντων μέσω EDA αρχιτεκτονικής με την αρχιτεκτονική των έμπειρων συστημάτων μπορεί να επιλύσει ταυτόχρονα δύο από τις βασικότερες προκλήσεις που αντιμετωπίζουν διαχρονικά τα Έμπειρα Συστήματα.

4.1 Συστήματα Παραγωγής

Το πρώτο βήμα για την υλοποίηση της προτεινόμενης αρχιτεκτονικής είναι η επιλογή του κατάλληλου τύπου έμπειρου συστήματος. Όπως έχουμε αναφέρει και στο δεύτερο κεφάλαιο, ένας τύπος έμπειρων συστημάτων είναι τα Συστήματα Παραγωγής. Στα Συστήματα Παραγωγής η αναπαράσταση γνώσης βασίζεται στους κανόνες παραγωγής. Δηλαδή σε κανόνες της μορφής IF…THEN. Γνωρίζουμε επίσης από το τρίτο κεφάλαιο ότι για την ανάλυση της επεξεργασίας συμβάντων χρησιμοποιούνται κανόνες της μορφής ECA (Event – Condition - Action). Είναι προφανές ότι το έμπειρο σύστημα θα είναι ένα σύστημα παραγωγής εφόσον υπάρχει σαφής αντιστοίχηση μεταξύ της μορφής των κανόνων παραγωγής και των κανόνων που απαιτούνται στην επεξεργασία συμβάντων.

Τα Συστήματα Παραγωγής τα οποία ανήκουν ευρύτερα στο πεδίο της Τεχνητής Νοημοσύνης βασίζονται στην πεποίθηση ότι η ανθρώπινη νοημοσύνη εν μέρει φαίνεται ότι οφείλεται στην δυνατότητα του νου να μεταχειρίζεται σύμβολα - όχι αριθμούς. Τα συστήματα παραγωγής (production system) δημιουργήθηκαν για να περιγράψουν τον τρόπο με τον οποίο ο άνθρωπος επεξεργάζεται την συμβολική πληροφορία. Είναι ουσιαστικά Έμπειρα Συστήματα τα οποία χρησιμοποιούν τους κανόνες ως ένα τρόπο αναπαράστασης της γνώσης. Η Βάση Γνώσης δηλαδή περιέχει πληροφορίες οι οποίες είναι κωδικοποιημένες με κανόνες παραγωγής.

Αυτού του είδους τα συστήματα εφαρμόζονται συνήθως σε περιπτώσεις που το πρόβλημα που καλείται να αντιμετωπίσει είναι αυστηρά καθορισμένο και κατανοήσιμο και η γνώση μπορεί να αναπαρασταθεί σε δεδομένα και κανόνες.

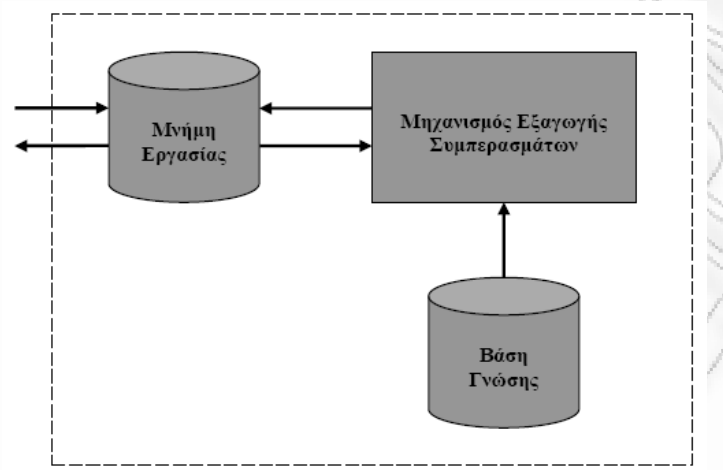
Η Αρχιτεκτονική ενός τέτοιου συστήματος είναι παρόμοια με αυτή ενός έμπειρου συστήματος και αποτελείται από τρία βασικά μέρη:

- Βάση Γνώσης
- Διερμηνευτή Κανόνων
- Μνήμη Εργασίας

Ένα σύστημα παραγωγής θεωρείται Turing Complete το οποίο επικεντρώνεται στην αναπαράσταση γνώσης ώστε να μπορεί να εκφράσει προτασιακή λογική και λογική πρώτης τάξης με συνοπτικό, σαφή και δηλωτικό τρόπο. Ο «εγκέφαλος» ενός τέτοιου συστήματος είναι η μηχανή εξαγωγής συμπερασμάτων η οποία πρέπει να διαχειριστεί έναν μεγάλο όγκο κανόνων και γεγονότων. Οι κανόνες και τα γεγονότα αντιστοιχίζονται μέσω μιας διαδικασίας αναγνώρισης

προτύπου προκειμένου να καταλήξουν σε συμπεράσματα τα οποία πυροδοτούν ενέργειες. Η μηχανή εξαγωγής συμπερασμάτων είναι statefull και ικανή να «δικαιολογήσει» τα συμπεράσματα της, δυνατότητα η οποία καλείται Truth Maintenance.

Η διαδικασία επεξεργασίας των γεγονότων που είναι στη μνήμη εργασίας αποτυπώνεται στο σχήμα που ακολουθεί:



Σχήμα 19: Αρχιτεκτονική ενός Συστήματος Παραγωγής

Όταν ικανοποιούνται οι υποθέσεις τότε ενεργοποιείται μια ενέργεια ή εξάγεται ένα συμπέρασμα. Οι κανόνες εκφράζουν διάφορα είδη γνώσης, όπως λογικές σχέσεις μεταξύ οντοτήτων για εύρεση λύσεων που απαιτούν επαγωγική διαδικασία ή αιτιακές σχέσεις μεταξύ τους για τον προσδιορισμό των αιτίων για κάποια συμβάντα κ.α.

Τα βασικά χαρακτηριστικά των συστημάτων παραγωγής συνοψίζονται στα εξής:

- Είναι ένα "έξυπνο" υπολογιστικό πρόγραμμα το οποίο χρησιμοποιεί γνώση και διαδικασίες εξαγωγής συμπερασμάτων.
- Βασίζεται σε κανόνες για την αναπαράσταση της γνώσης.
- Διαχωρίζει τη γνώση από το υπόλοιπο σύστημα το οποίο την επεξεργάζεται.
- Μπορεί να χειριστεί αποτελεσματικά την αβεβαιότητα των πληροφοριών.

Τα χαρακτηριστικά αυτά πληρούν τις προϋποθέσεις ενός έμπειρου συστήματος.

4.2 Σύστημα Παραγωγής με ECA κανόνες

Στην προτεινόμενη αρχιτεκτονική η πρώτη βασική διαφοροποίηση σε σχέση με ένα σύστημα παραγωγής είναι το γεγονός ότι θα πρέπει να επεξεργάζεται συμβάντα (events) και όχι δεδομένα (facts). Η βασική διαφορά μεταξύ event και fact είναι ότι ένα event έχει χρόνο ζωής ενώ ένα fact παραμένει αμετάβλητο. Με αυτή την έννοια το σύστημα θα πρέπει να επεξεργάζεται κάποια δομή δεδομένων που θα ακολουθεί ένα πρότυπο (pattern) και στο οποίο θα λαμβάνεται υπόψη η παράμετρος του χρόνου. Το σύστημα αυτό θα πρέπει επίσης να επεξεργάζεται δεδομένα σε πραγματικά χρόνο (push αντί για pull, πχ μέσω polling).

Αυτά τα συστήματα βασίζονται, όπως έχουμε προαναφέρει σε ECA κανόνες. Εάν ικανοποιούνται κάποιοι κανόνες τότε ενεργοποιούνται κάποιες ενέργειες. Οι ECA κανόνες καθορίζουν και την αρχιτεκτονική του συστήματος εφόσον η επικοινωνία μεταξύ των διαφόρων τμημάτων της αρχιτεκτονικής πρέπει να υλοποιηθεί με τέτοιο τρόπο που να εξασφαλίζει την «προώθηση» των συμβάντων από τη στιγμή που εμφανίζονται μέχρι τη στιγμή θα θεωρούνται πλέον παρωχημένα. Άρα η διασύνδεση των τμημάτων θα πρέπει να γίνεται μέσω των συμβάντων.

Τα συμβάντα δεν εμφανίζονται όμως σε συγκεκριμένες χρονικές στιγμές οπότε η επεξεργασία τους θα πρέπει να γίνεται ασύγχρονα. Μάλιστα είναι κρίσιμο η επεξεργασία να των συμβάντων να αρχίζει άμεσα και αυτό να διασφαλίζεται από όλα τα τμήματα της αρχιτεκτονικής σε όλα τα στάδια επεξεργασίας. Συνοψίζοντας, για να μπορεί η αρχιτεκτονική να χαρακτηρίζεται και ως EDA θα πρέπει να διασφαλίζει ότι:

1. Τα συμβάντα προωθούνται (pushed) στους κατάλληλους παραλήπτες
2. Τα συμβάντα δεν καθορίζουν τις ενέργειες
3. Ασύγχρονη Λειτουργία.

Οι κανόνες της μορφής ECA θεωρούνται χρήσιμες ιδιαίτερα σε περιπτώσεις που:

- Ένα σύστημα πρέπει να αντιδρά στην παρουσία ενός συνόλου συμβάντων.
- Η αντίδραση του συστήματος υπόκειται, εκτός από την παρουσία κάποιου(ων) συμβάντος(ντων), και στην ικανοποίηση κάποιων συνθηκών.
- Η λογική της επεξεργασίας των συμβάντων πρέπει να διαχωρίζεται από εκτέλεση των ενεργειών ως αντίδραση στην παρουσία των συμβάντων.

4.3 Αρχιτεκτονική του συστήματος

Η αρχιτεκτονική ενός συστήματος περιγράφει τη δομή και τη διασύνδεση μεταξύ των τμημάτων που την απαρτίζουν προκειμένου να ικανοποιεί τις λειτουργικές και μη-λειτουργικές απαιτήσεις αλλά και να παρέχει μια συνεκτική εννοιολογική περιγραφή της δομής αυτής. Δεν έχει σαν στόχο την λεπτομερή περιγραφή της υλοποίησης, όπως τους αλγόριθμους επεξεργασίας ή τις δομές δεδομένων, αλλά τη συνολική συμπεριφορά του συστήματος. Αποτελεί με άλλα λόγια ένα προσχέδιο για την ανάπτυξη ενός συστήματος καταδεικνύοντας τα βασικά συστατικά του στοιχεία και τις αλληλεπιδράσεις μεταξύ των στοιχείων αυτών.

Ένας αποτελεσματικός τρόπος απεικόνισης μιας αρχιτεκτονικής είναι μέσω επιπέδων. Η δόμηση της πολυπλοκότητας και της λειτουργικότητας ενός συστήματος σε επίπεδα υποστηρίζει διάφορες επιθυμητές ιδιότητες μιας σχεδίασης όπως προσαρμοστικότητα, συνεργασιμότητα και απόκριση. Η βασική ιδέα είναι η δόμηση της λειτουργικότητας σε 2 ή περισσότερα ιεραρχικά επίπεδα τα οποία αλληλεπιδρούν προκειμένου να επιτύχουν μια συνεκτική συμπεριφορά.

Η σχεδίαση του συστήματος σε επίπεδα εξασφαλίζει ευελιξία. Δηλαδή η αλλαγή κάποιου στοιχείου σε κάποιο επίπεδο της αρχιτεκτονικής δεν μεταβάλλει τη συνολική συμπεριφορά του συστήματος. Οπότε κάθε στιγμή επιτυγχάνεται η βελτιστοποίηση του συστήματος με την αντικατάσταση κάποιου στοιχείου (component) χωρίς κάποια μεγάλη μεταβολή στην αρχιτεκτονική του. Μια καλή αρχιτεκτονική διασφαλίζει ότι το σύστημα ικανοποιεί τόσο τις λειτουργικές όσο και τις μη-λειτουργικές απαιτήσεις.

Στην προτεινόμενη αρχιτεκτονική διακρίνουμε 4 βασικά επίπεδα:

- Πηγές Συμβάντων
- Κανάλι Κορμού
- Επεξεργασία Συμβάντων
    - Υποσύστημα Συντονισμού
        - Προτυποποίηση των Συμβάντων
        - Μηχανή κανόνων
- Διεπαφή με το χρήστη

Η αρχιτεκτονική αυτή αποτυπώνεται στο σχήμα που ακολουθεί:

Σχήμα 20: Αρχιτεκτονική Ε.Σ με EDA

4.4 Ανάλυση των επιπέδων της Αρχιτεκτονικής

4.4.1 Πηγές Συμβάντων

Οι Πηγές Συμβάντων αποτελούν το σύνολο των συστημάτων, εφαρμογών, αισθητήρων τα οποία παράγουν συμβάντα. Τα συμβάντα αυτά δεν έχουν υποστεί κανενός είδους επεξεργασία. Μπορεί να προέρχονται από βάσεις δεδομένων, αισθητήρες, web services, δεδομένα από άλλες εφαρμογές σε πχ XML μορφή κ.α.

4.4.2 Κανάλι Κορμού

Στο επόμενο επίπεδο τα συμβάντα διοχετεύονται σε ένα Κανάλι Κορμού. Το βασικό χαρακτηριστικό του είναι ότι λειτουργεί ασύγχρονα. Κάθε χρονική στιγμή μπορεί να λάβει ένα συμβάν και αμέσως να το προωθήσει στο επόμενο επίπεδο. Το κανάλι κορμού λειτουργεί ως ένα κεντρικό σημείο συγκέντρωσης όλων των συμβάντων. Διασφαλίζεται με αυτό τον τρόπο ότι δεν θα έχουμε απώλεια πληροφορίας. Σε αυτό το επίπεδο έχουμε να κάνουμε κυρίως με απλά συμβάντα. Δηλαδή με πρωτογενή συμβάντα τα οποία δεν έχουν ακόμα συναθροιστεί σε σύνθετα. Με το κανάλι συμβάντων επιτυγχάνεται λοιπόν η μη-απώλεια πληροφορίας και η άμεση απόκριση σε αυτά. Η ασύγχρονη λειτουργία επιτυγχάνεται μέσω ενός μηχανισμού publish/subscribe, όπου οι πηγές συμβάντων παράγουν τα συμβάντα και τα υπόλοιπα στοιχεία της αρχιτεκτονικής «καταναλώνουν» τα συμβάντα αυτά. Δεν απαιτείται λοιπόν τα συμβάντα να μεταδίδονται απευθείας από υποσύστημα σε υποσύστημα, αλλά «εγγράφονται» στο κανάλι για να λαμβάνουν τα συμβάντα για τα οποία ενδιαφέρονται.

Για να μπορούν τα συμβάντα να επικοινωνούν μεταξύ τους θα πρέπει να υπάρχει μια κοινή κατανόηση ως προς τη φύση και τη λειτουργία τους. Δηλαδή θα πρέπει να ξεφεύγουν από τον σκοπό που επιτελούν στην εφαρμογή από την οποία προέρχονται. Επειδή η πηγή του

συμβάντος δεν μπορεί να γνωρίζει ποια υποσυστήματα έχουν «εγγραφεί» στο κανάλι κορμού για να το επεξεργαστεί, θα πρέπει να υπάρχει μια δομή σε υψηλότερο επίπεδο που να διασφαλίζει την επικοινωνία. Το επόμενο επίπεδο λοιπόν θα πρέπει καταρχήν να «προτυποποιεί» τα συμβάντα σε μια μορφή που θα είναι κατανοητή καθολικά στα υποσυστήματα.

4.4.3 Επεξεργασία Συμβάντων

Αυτό το επίπεδο αποτελεί τον πυρήνα της Αρχιτεκτονικής. Προκειμένου να κατανοήσουμε τη λειτουργικότητά του έχουμε προχωρήσει σε λεπτομερέστερη ανάλυση σε σχέση με τα άλλα επίπεδα υποδιαιρώντας τον στα βασικά επίπεδα από τα οποία αποτελείται. Καταρχήν σε μια «καθαρή» EDA αρχιτεκτονική τα υποσυστήματα επικοινωνούν με συμβάντα και άρα τα συμβάντα προωθούνται στα διάφορα υποσυστήματα κατά τη διαδικασία επεξεργασίας τους και για το χρόνο ζωής τους. Δημιουργείται έτσι ένα σύστημα το οποίο να αποτελείται από αλυσίδα υποσυστημάτων τα οποία μπορούν να θεωρηθούν ως ένα κατανεμημένο EDA σύστημα ή ως συνεργατικά EDA υποσυστήματα που λειτουργούν με ένα τέτοιο τρόπο ώστε να επιδεικνύουν τελικά μια λογική συμπεριφορά. Η δυσκολία στην επίτευξη μιας συνεκτικής συμπεριφοράς έγκειται στη διασφάλιση της επικοινωνίας μεταξύ των υποσυστημάτων. Για αυτό το λόγο θα πρέπει καταρχήν τα events να «προτυτοποιούνται» δηλαδή να έχουν μια ελάχιστη δομή προκειμένου να γίνονται αντιληπτά και κατανοητά από όλα τα υποσυστήματα και επίσης πρέπει να υπάρχει και ένα κεντρικό υποσύστημα του οποίου ο ρόλος θα είναι ο συντονιστικός.

Προτυποποίηση των συμβάντων

Από το κανάλι κορμού τα συμβάντα προωθούνται στο επόμενο επίπεδο που είναι η επεξεργασίας συμβάντων. Αρχικά πρέπει να υποστούν μια πρώτη επεξεργασία προκειμένου να καταστούν κατανοητά στα επόμενα επίπεδα. Θα πρέπει με αυτή την έννοια να οριστεί πιο συγκεκριμένα ο τρόπος αναπαράστασής τους.

Αναπαράσταση των συμβάντων

Για να επιτευχθεί η μέγιστη διαλειτουργικότητα των συμβάντων στο σύνολο της αρχιτεκτονικής θα πρέπει τα συμβάντα να είναι ανεξάρτητα από συγκεκριμένες πλατφόρμες επεξεργασίας. Επειδή τα συμβάντα προωθούνται μεταξύ των διαφορετικών τμημάτων της αρχιτεκτονικής, θα πρέπει στη δομή τους να αντικατοπτρίζεται η διαδρομή τους στα διάφορα υποσυστήματα. Για να επιτευχθεί αυτό, το κάθε υποσύστημα θα μπορούσε να προσθέτει πληροφορίες στο συμβάν οι οποίες μπορεί να είναι χρήσιμες στα υποσυστήματα που θα «παραλάβουν» το συμβάν αυτό.

Ταξινόμηση των συμβάντων

Σε ένα σύστημα παραγωγής του οποίου η λειτουργικότητα επεκτείνεται και σε ECA κανόνες το σύστημα θα πρέπει να έχει τη δυνατότητα να αναγνωρίζει πολλούς και διαφορετικούς τύπους ή κλάσεις συμβάντων. Για την προτεινόμενη αρχιτεκτονική θα μπορούσαμε να υιοθετήσουμε μια πρόταση που επιλύει αυτό το ζήτημα μέσω μιας οντολογίας συμβάντος, όπου ως οντολογία ορίζεται ένας μοναδικός ορισμός μιας κοινής αντίληψης. Στην φιλοσοφία μια οντολογία θεωρείται ένας συστηματικός λογαριασμός μιας ΄Υπαρξης. Στα Έμπειρα Συστήματα ως ύπαρξη ορίζεται οτιδήποτε μπορεί να αναπαρασταθεί. Σε ένα σύστημα με ECA κανόνες τα εισερχόμενα συμβάντα θα πρέπει να αντιστοιχηθούν σε ένα σύνολο κανόνων.

Μια τυπική αντιμετώπιση είναι η ταξινόμηση των συμβάντων βάσει των δεδομένων τους ή βάσει του τρόπου με τον οποίο εμφανίστηκαν και στην συνέχεια η αξιολόγηση των κανόνων που εφαρμόζονται στον συγκεκριμένο τύπο συμβάντων. Το κάθε συμβάν μπορεί να χαρακτηρίζεται από ένα δείκτη, συνήθως είναι στον header της δομής ενός συμβάντος, ο οποίος υποδηλώνει τον συγκεκριμένο τύπο συμβάντος. Εναλλακτικά οι κανόνες μπορούν να αντιστοιχηθούν βάσει άλλων ιδιοτήτων των συμβάντων, όπως για παράδειγμα βάσει της πηγής από την οποία προέρχονται (πχ από ένα RFID αισθητήρα).

Κανονικά τα συμβάντα αναγνωρίζονται από το γεγονός ότι ανήκουν σε ένα πλαίσιο το οποίο αφορά τις συνθήκες ή τις ενέργειες ενός ECA κανόνα. Το πλαίσιο μέσα στο οποίο λειτουργεί ένας κανόνας έχει διάφορα χαρακτηριστικά όπως:

- Χαρακτηριστικά που αφορούν το χρόνο: μια υπηρεσία παρέχεται για ένα χρονικό διάστημα.
- Χωρικά χαρακτηριστικά: όταν το μήνυμα φθάνει στον τελικό προορισμό του.
- Χαρακτηριστικά που αφορούν μια κατάσταση: Κανένα εισερχόμενο μήνυμα.
- Σημασιολογικά χαρακτηριστικά: άτομα τα οποία ανήκουν πχ στην ίδια εταιρεία.

Μηχανή Κανόνων

Η μηχανή κανόνων αποτελεί το βασικό συστατικό στοιχείο στην επεξεργασία συμβάντων. Παραλαμβάνει τα συμβάντα μετά την διαδικασία προτυποποίησης και εφαρμόζει τους κατάλληλους κανόνες. Σε αυτό το σημείο είναι αναγκαίο να αναλύσουμε περισσότερο τη λειτουργία της μηχανής κανόνων για να κατανοήσουμε πληρέστερα πως εντάσσεται στο σύνολο της αρχιτεκτονικής.

Καταρχήν η μηχανή κανόνων εκτός από την επεξεργασία των κανόνων υποστηρίζει και τη συγγραφή τους. Συνήθως υποστηρίζονται διάφοροι τρόποι σύνταξης των κανόνων που μπορεί να περιλαμβάνει από κάποια μορφή κώδικα μέχρι μορφές που είναι κατανοητές σε απλούς χρήστες ή ειδικούς. Σε αυτές τις μεθόδους περιλαμβάνονται οι πίνακες απόφασης, χρήση guided editor, DSL κ.α. Είναι σημαντική η δυνατότητα έκφρασης των κανόνων ακόμα και σε φυσική γλώσσα διότι ένα βασικό χαρακτηριστικό των έμπειρων συστημάτων είναι ο εμπλουτισμός της Βάσης Γνώσης με νέους κανόνες από τους ειδικούς του τομέα που εφαρμόζεται το έμπειρο σύστημα (Ε.Σ.). Εμπεριέχει λοιπόν η μηχανή κανόνων και ένα μεταγλωττιστή (compiler) ο οποίος μεταφράζει τη φυσική γλώσσα σε κώδικα.

Όπως αναλύσαμε στην προηγούμενη παράγραφο τα συμβάντα πρέπει να αποκτήσουν μια συγκεκριμένη δομή για να καταστούν επεξεργάσιμα. Η ύπαρξη αυτής της δομής είναι σημαντική γιατί ορίζει τα δεδομένα που μπορούμε να χρησιμοποιήσουμε ως μεταβλητές στη συγγραφή των κανόνων. Αυτή η δομή βάσει της οποίας δημιουργούνται οι κανόνες καλείται fact model.

Επιγραμματικά σε μια τυπική μηχανή κανόνων:

- Γίνεται συγγραφή των κανόνων
- Περιέχεται η μνήμη εργασίας
- Εκτελείται αναγνώριση προτύπου βάσει κάποιου αλγόριθμου (πχ RETE)
- Σε περίπτωση ενεργοποίησης ταυτόχρονα πολλών κανόνων εφαρμόζονται στρατηγικές επίλυσης συγκρούσεων (πχ salience, LIFO)

Πεδίο Γνώσης του Ε.Σ.

Το συγκεκριμένο επίπεδο όπως απεικονίζεται και στο σχήμα αποτελεί ένα ενδιάμεσο ανεξάρτητο επίπεδο μεταξύ της «Προτυποποίησης» και της μηχανής κανόνων. Το επίπεδο αυτό περιέχει τη Γνώση που εμπεριέχεται στο Ε.Σ.. Εξειδικεύει δηλαδή τον τομέα εφαρμογής του συστήματος. Η προτυποποίηση των συμβάντων, δηλαδή η επιλογή τους από ένα πλήθος δεδομένων και η μετατροπή τους σε μια κατανοητή δομή εξαρτώνται από το τι πρόβλημα θέλουμε να αντιμετωπίσουμε. Αλλιώς δεν θα μπορούμε να διαχειριστούμε τον όγκο των δεδομένων. Επιπροσθέτως, και οι κανόνες που εφαρμόζονται καλύπτουν ένα συγκεκριμένο πεδίο γνώσης. Ο καθορισμός αυτού του πεδίου εξαρτάται από το «Πεδίο Γνώσης» του συστήματος. Διακρίνεται στο σχήμα με διακεκομμένη γραμμή γιατί θέλουμε καταρχήν να τονίσουμε την ιδιαίτερη αλληλεπίδραση που έχει με τα άλλα δυο επίπεδα όπου καθορίζει καίρια τη συμπεριφορά τους και ταυτόχρονα η αλλαγή του περιεχομένου του Πεδίου Γνώσης μεταβάλλει συνολικά τη συμπεριφορά του συστήματος ως προς τη φύση του προβλήματος που καλείται να αντιμετωπίσει.

Εάν αλλάξουμε το πεδίο γνώσης με κάποιο άλλο τότε η ίδια αρχιτεκτονική μπορεί να εφαρμοστεί χωρίς καμία άλλη αλλαγή για την αντιμετώπιση άλλου τύπου προβλημάτων. Είναι και αυτό άλλο ένα πλεονέκτημα της loose-coupled αρχιτεκτονικής που προτείνουμε και είναι βασικό χαρακτηριστικό των Ε.Σ.

Διεπαφή με τον Χρήστη

Η διεπαφή με τον χρήστη εκπληρώνει έναν διττό σκοπό. Καταρχήν παρέχει στον απλό χρήστη την παρουσίαση των αποτελεσμάτων της επεξεργασίας. Ταυτόχρονα μπορεί να χρησιμοποιηθεί και από τον μηχανικό γνώσης προκείμενου να εμπλουτίσει ή να μεταβάλλει τη Γνώση του συστήματος. Αναφέραμε τους πολλούς τρόπους που παρέχει η μηχανή κανόνων για την εύκολη προσθήκη νέων κανόνων. Επίσης μπορεί να χρησιμοποιηθεί για την επαλήθευση των αποτελεσμάτων και τον έλεγχο της συλλογιστικής. Σε σχέση με ένα «καθαρό» έμπειρο σύστημα η διεπαφή δεν χρησιμοποιείται για την εισαγωγή δεδομένων με τη μορφή ερωτο-απαντήσεων, απαντήσεων αλλά το δεδομένα ανιχνεύονται από αισθητήρες και προωθούνται προς επεξεργασία.

4.5 Αντιστοιχία Ε.Σ. με την προτεινόμενη αρχιτεκτονική

Η συγκεκριμένη αρχιτεκτονική αποτελεί μια προσπάθεια βελτίωσης της Αρχιτεκτονικής ενός Ε.Σ με στοιχεία από EDA αρχιτεκτονική. Το αποτέλεσμα είναι ένα σύστημα το οποίο να μπορεί αν αντιδρά σε συμβάντα σε πραγματικό χρόνο. Η αρχιτεκτονική στην οποία βασιστήκαμε είναι εκείνη ενός τυπικού συστήματος παραγωγής. Η αλλαγές που επιφέραμε στη συμπεριφορά του και στην λειτουργία αναφέρονται συνοπτικά στον πίνακα που ακολουθεί:

Συστήματα Παραγωγής	Προτεινόμενη Αρχιτεκτονική
If – Then	Event – Condition - Action
Διαχωρισμός Βάσης Γνώσης και κελύφους	Διαχωρισμός event από action η οποία υλοποιείται μέσω Loosed Coupled αρχιτεκτονικής
Δεδομένα	Συμβάντα
Ερωτο-απαντήσεις μέσω διεπαφής με τον χρήστη	Συμβάντα τροφοδοτούνται από αισθητήρες και άλλες πηγές
Δεδομένα από βάσεις μέσω polling	Τα δεδομένα προωθούνται (push)
Statefull	Stateless

Πίνακας 3: Αντιστοίχηση χαρακτηριστικών ενός Συστήματος Παραγωγής και της Προτεινόμενης Αρχιτεκτονικής

Από τον πίνακα παρατηρούμε ότι σε ένα έμπειρο σύστημα επεξεργαζόμαστε δεδομένα τα οποία στις περισσότερες περιπτώσεις δίδονται μέσω ερωτοαπαντήσεων μεταξύ του χρήστη και του συστήματος, ενώ στην προτεινόμενη αρχιτεκτονική επεξεργαζόμαστε συμβάντα τα οποία προέρχονται από πολλές και ετερογενείς πηγές δεδομένων. Διατηρείται και στις δύο αρχιτεκτονικές ο διαχωρισμός μεταξύ Βάσης Γνώσης και κελύφους. Στην προτεινόμενη αρχιτεκτονική η αλλαγή στο Πεδίο Γνώσης επιτρέπει την αντιμετώπιση διαφόρων προβλημάτων. Τέλος, η επικοινωνία μεταξύ των υποσυστημάτων βασίζεται σε push-based επικοινωνίες για να εκτελείται η επεξεργασία σε πραγματικό χρόνο. Μια μετεξέλιξη της αρχιτεκτονικής θα ήταν με SOA όπου οι ενέργειες θα μπορούσαν να πραγματοποιούνται μέσω web services.

Για την ανάπτυξη μιας εφαρμογής σε ένα πραγματικό επιχειρησιακό περιβάλλον η τυπική προσέγγιση θα μπορούσε να ακολουθεί τα παρακάτω βήματα:

- Αναβάθμιση των συστημάτων της εταιρείας προκειμένου να αναγνωρίζει συμβάντα.
- Συσχέτιση αυτών των συμβάντων με δομές/ αντικείμενα(objects) των επιχειρησιακών διαδικασιών.
- Δυνατότητα αναγνώρισης προτύπων συμβάντων.
- Η αναγνώριση προτύπων επεκτείνεται και στην αναγνώριση πιο πολύπλοκων δομών συμβάντων, δηλαδή σύνθετων συμβάντων.

- Μοντελοποίηση των επιχειρησιακών διαδικασιών με τέτοιο τρόπο ώστε να αναπαριστούν κατανοητές επιχειρησιακές μεταβολές ορισμένες σε αυστηρά χρονικά πλαίσια.
- Δημιουργία αυτόματων μηχανισμών αντίδρασης σε αυτές τις μεταβολές είτε για την επίλυση προβλημάτων ή για την εκμετάλλευση επιχειρηματικών ευκαιριών.

Η εφαρμογή ενός έμπειρου συστήματος το οποίο να μπορεί να διαχειρίζεται συμβάντα προσφέρεται σε τομείς όπου τα προβλήματα είναι αυστηρά καθορισμένα και η γνώση μπορεί να αναπαρασταθεί με πρότυπα και κανόνες. Υπάρχουν σήμερα τεχνολογίες που υποστηρίζουν την επεξεργασία σύνθετων συμβάντων μέσω μηχανών κανόνων που χρησιμοποιούν κανόνες της μορφής Event-Condition-Acton. Στη συνέχεια αναλύουμε μια πιο τεχνολογική προσέγγιση στην επεξεργασία συμβάντων.

5 Τεχνολογίες και Εργαλεία Υλοποίησης

Η Επεξεργασία Συμβάντων αποτελεί έναν ταχαίος αναπτυσσόμενο κλάδο που επιχειρεί να προσφέρει λύσεις στις σύγχρονες επιχειρήσεις. Στον τομέα αυτό προσπαθούν να δραστηριοποιηθούν και συστήματα ανοικτού κώδικα. Στην προσπάθειά μας να διερευνήσουμε τη λειτουργικότητα ενός τέτοιου συστήματος, καταλήξαμε στο Drools της Jboss.

Ένα από τα βασικότερα πλεονεκτήματα του συστήματος είναι το Σύστημα Διαχείρισης των Κανόνων Παραγωγής (BRMS – Business Rules Management System) ως ένα κεντρικό σημείο αποθήκευσης και διαχείρισης των κανόνων. Επιπλέον το Drools είναι μια από τις ελάχιστες πλατφόρμες ανοιχτού κώδικα που μέσω του Fusion, ενός ανεξάρτητου module, επεκτείνει τη λειτουργικότητά στην επεξεργασία συμβάντων. Αποτελεί με αυτή την έννοια ιδανική επιλογή προκειμένου να παρουσιαστεί μια τεχνολογία η οποία προσφέρει τη δυνατότητα συνδυασμού των κανόνων ενός συστήματος παραγωγής με την επεξεργασία συμβάντων.

Η λειτουργικότητα του Drools βασίζεται στο JSR 94, το οποίο αποτελεί μία προσπάθεια προτυποποίησης των μηχανών κανόνων παραγωγής για την τεχνολογία της Java. Το JSR 94 παρέχει καθοδήγηση για τη διαχείριση των κανόνων, καθώς επίσης και τα API (Application Programming Interface) που πρέπει να υλοποιεί μία μηχανή για διαδικασίες όπως η εκτέλεση των κανόνων. Ο εξυπηρετητής εφαρμογών στο οποίο έχει εγκατασταθεί το Drools είναι ο Jboss AS.

5.1 Ο εξυπηρετητής εφαρμογών JBoss AS

Ο JBoss Application Server είναι ένας πιστοποιημένος J2EE εξυπηρετητής εφαρμογών υλοποιημένος από την κοινότητα Open Source Software (OSS) και αποτελεί τη σημαντικότερη υλοποίηση από τη σουίτα των OSS εφαρμογών που συνθέτουν το σύστημα JBoss Enterprise Middleware System (JEMS) [3w8]. Η ισχυρή αλλά ταυτόχρονα ευέλικτη αρχιτεκτονική του JBoss AS, η υψηλή του απόδοση και αξιοπιστία καθώς και το μηδενικό κόστος εγκατάστασης και χρήσης του, τον έχουν καταστήσει μία από τις πιο δημοφιλείς επιλογές των προγραμματιστών.

- Στην ανάλυση των μελετών χρήσης δεν χρησιμοποιήθηκε το εύρος των δυνατοτήτων που προσφέρει ο Jboss AS. Μπορεί όμως να χρησιμοποιηθούν στη βάση των επεκτάσεων που προτείνονται στο τελευταίο κεφάλαιο. Η λειτουργικότητα που προσφέρει σε σχέση με τις προτεινόμενες επεκτάσεις αφορούν:
- Την ενσωμάτωση του Apache Tomcat 5, που αποτελεί την υλοποίηση αναφοράς του προτύπου JSP 2.0 & Servlet 2.4 Container. Η ενοποίηση των υπηρεσιών του Tomcat με τις υπηρεσίες του JBoss AS παρέχει ένα περιβάλλον υψηλής αξιοπιστίας, εύκολης κλιμάκωσης και απεριόριστης διαθεσιμότητας για την υποστήριξη των διαδικτυακών εφαρμογών.
- Την ενσωμάτωση με την τεχνολογία Hibernate, που αποτελεί την πιο πρωτοπόρα και επιτυχημένη λύση στην αντιστοίχηση σχεσιακών δεδομένων με δομές του αντικειμενοστραφούς προγραμματισμού.
- Την υποστήριξη Web Services καθώς και τη δυνατότητα πιο ολοκληρωμένης επεξεργασίας των XML αρχείων. Ο JBoss AS ενσωματώνει απόλυτα τις προδιαγραφές των Web Service και δίνει τη δυνατότητα στις εφαρμογές που χτίζονται πάνω του να αλληλεπιδρούν δυναμικά με Web Services χρησιμοποιώντας πρωτόκολλα επικοινωνίας όπως το Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), και XML.

Τέλος, ισχυρό κίνητρο για την επιλογή του JBoss AS είναι ότι αποτελεί μέλος των συστημάτων της JBoss Enterprise Middleware, μίας ολοκληρωμένης σουίτας προϊόντων που στοχεύει στο να παρέχει μία πλατφόρμα ανάπτυξης επιχειρησιακών εφαρμογών ανεξαρτήτως λειτουργικού συστήματος και να διευκολύνει την ανάπτυξη των εφαρμογών αναλαμβάνοντας το χειρισμό διαδικασιών όπως αποθήκευση, ολοκληρωμένη εκτέλεση διεργασιών, επικοινωνία και καταμερισμό του φόρτου κατά μήκος ενός δικτύου. Κάποια από τα συστήματα αναλαμβάνουν

την υλοποίηση προτυποποιημένων διαδικασιών, ενώ άλλα υλοποιούν εξειδικευμένες υπηρεσίες όπως το JBoss Messaging και Hibernate. Επιπλέον, μέλος της σουίτας JEMS αποτελεί το Σύστημα Διαχείρισης Επιχειρησιακών Κανόνων JBoss Drools, που θα χρησιμοποιηθεί ως ο πυρήνας του έμπειρου συστήματος. Η μηχανή Jboss Drools συνεργάζεται άψογα με τον JBoss AS, ο οποίος θεωρείται ιδανικός για εφαρμογές με αυξημένες απαιτήσεις. απόδοσης και διαθεσιμότητας.

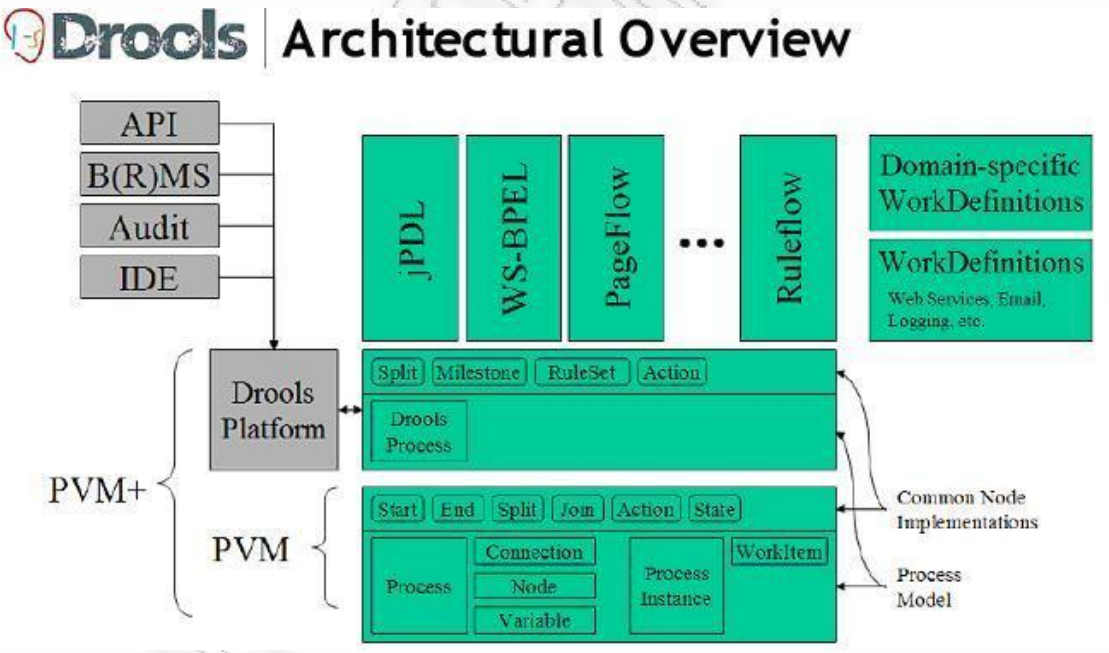
5.2 Η μηχανή κανόνων (Drools 5.0)

Το Drools είναι ένα σύστημα διαχείρισης κανόνων (Business Rule Management System – BRMS) που βασίζεται σε μια μηχανή κανόνων ορθής αλυσίδωσης που βασίζεται μια εξελιγμένη υλοποίηση του αλγόριθμου Rete που λέγεται ReteOO (Rete Object Oriented). Το Drools βασίζεται στο JSR94 (Java Rule Engine API) πρότυπο.

Το Drools από τη έκδοση 5.0 και μετά έχει μετεξελιχθεί σε μια σουίτα υποπρογραμμάτων τα οποία είναι τα εξής [3w7]:

- Drools Expert (η μηχανή κανόνων) – Ο πυρήνας του συστήματος
- Drools Guvnor – Web-based σύστημα αποθήκευσης και διαχείρισης των εφαρμογών του Drools
- Drools Flow - Υποσύστημα για τη δημιουργία διαδικασιών και ροών εργασιών
- Drools Fusion (event processing/temporal reasoning) – Υποσύστημα για την επεξεργασία σύνθετων συμβάντων (CEP)
- Drools Planner – Υποσύστημα για την βελτιστοποίηση προβλημάτων σχεδιασμού συμπεριλαμβανομένου NP-hard προβλήματα.

Η Αρχιτεκτονική του Drools όπως παρουσιάζεται στο site του Jboss Drools φαίνεται στο επόμενο σχήμα [3w6]:



Σχήμα 21: Αρχιτεκτονική του Drools

Ακολουθεί μια συνοπτική περιγραφή των βασικών χαρακτηριστικών του Drools

5.2.1 Το JSR 94 API

Οι περισσότερες μηχανές κανόνων παραγωγής υλοποιούν εξειδικευμένα API, που καθιστά δύσκολη της ενσωμάτωσή τους με τα υπάρχοντα πληροφοριακά συστήματα της επιχείρησης. Επιπλέον, αν μία μηχανή πάψει να υποστηρίζεται τεχνικά και η επιχείρηση αποφασίσει να υιοθετήσει άλλη μηχανή, το μεγαλύτερο μέρος του κώδικα της εφαρμογής θα πρέπει να υλοποιηθεί ξανά.

Το JSR 94 [3w14] αποτελεί μία προσπάθεια προτυποποίησης των μηχανών κανόνων παραγωγής για την τεχνολογία της Java. Το JSR 94 παρέχει καθοδήγηση για τη διαχείριση των κανόνων, καθώς επίσης και τα API (Application Programming Interface) που πρέπει να υλοποιεί μία μηχανή για διαδικασίες όπως η εκτέλεση των κανόνων, αλλά δεν προβάλλει κανένα περιορισμό για τη γλώσσα στην οποία θα εκφραστούν οι κανόνες και οι ενέργειες. Ωστόσο, υπάρχουν προσπάθειες προτυποποίησης μίας κοινής γλώσσας κανόνων, συμπεριλαμβανομένης της Rule Markup Language (RuleML).

Το JSR 94 ορίζει ένα απλό API για την πρόσβαση σε μία μηχανή κανόνων παραγωγής από μία Java EE ή SE εφαρμογή. Παρέχει API για [6]:

- Την εγκατάσταση και την απεγκατάσταση κανόνων
- Τη διαπέραση των κανόνων
- Τον έλεγχο των μεταδεδομένων των κανόνων
- Την εκτέλεση των κανόνων
- Την ανάκτηση των αποτελεσμάτων
- Το φιλτράρισμα των αποτελεσμάτων

Το JSR 94 δεν προτυποποιεί:

- Την ίδια τη μηχανή κανόνων παραγωγής
- Τη ροή εκτέλεσης των κανόνων
- Τη γλώσσα που θα χρησιμοποιηθεί για την περιγραφή των κανόνων

5.2.2 Αναπαράσταση Γνώσης

Το Drools είναι μια μηχανή κανόνων η οποία μπορεί να αποτελέσει τον πυρήνα για την υλοποίηση ενός Έμπειρου Συστήματος το οποίο θα ανήκει στην κατηγορία των Συστημάτων Παραγωγής. Ένα Σύστημα Παραγωγής είναι Turing Complete όπου η αναπαράσταση γνώσης πραγματοποιείται μέσω κανόνων. Οι κανόνες αυτοί ακολουθούν την FOL (First Order Logic) ή κατηγορική, η οποία επεκτείνει την Προτασιακή Λογική. Κάθε πρόταση είναι μια δήλωση η οποία μπορεί να χαρακτηριστεί ως αληθής ή ψευδής. Η δομή των κανόνων χωρίζεται σε δύο μέρη:

When

<Συνθήκες>

Then

<Ενέργειες>

Ο «πυρήνας» του συστήματος είναι ο μηχανισμός εξαγωγής συμπερασμάτων που αντιστοιχίζει τους κανόνες που είναι αποθηκευμένοι στη Βάση Γνώσης με τα Δεδομένα που τροφοδοτούνται στη μνήμη εργασίας. Από τη διαδικασία αντιστοίχησης η μηχανή εξάγει συμπεράσματα τα οποία εκκινούν τις κατάλληλες ενέργειες.

5.2.3 Μηχανισμός εξαγωγής συμπερασμάτων

Το Drools χαρακτηρίζεται ως μια μηχανή κανόνων ορθής αλυσίδωσης. Τα δεδομένα εισάγονται στη μνήμη εργασίας όπου ενεργοποιούν έναν ή περισσότερους κανόνες οι οποίοι στη συνέχεια εκτελούνται από την Ατζέντα. Η διαδικασία ξεκινά από τα δεδομένα και καταλήγει σε κάποιο συμπέρασμα και για αυτό το λόγο ονομάζεται οδηγούμενη-από-τα-δεδομένα (data-driven). Ο αλγόριθμος που υλοποιεί αυτή τη διαδικασία βασίζεται στον Rete.

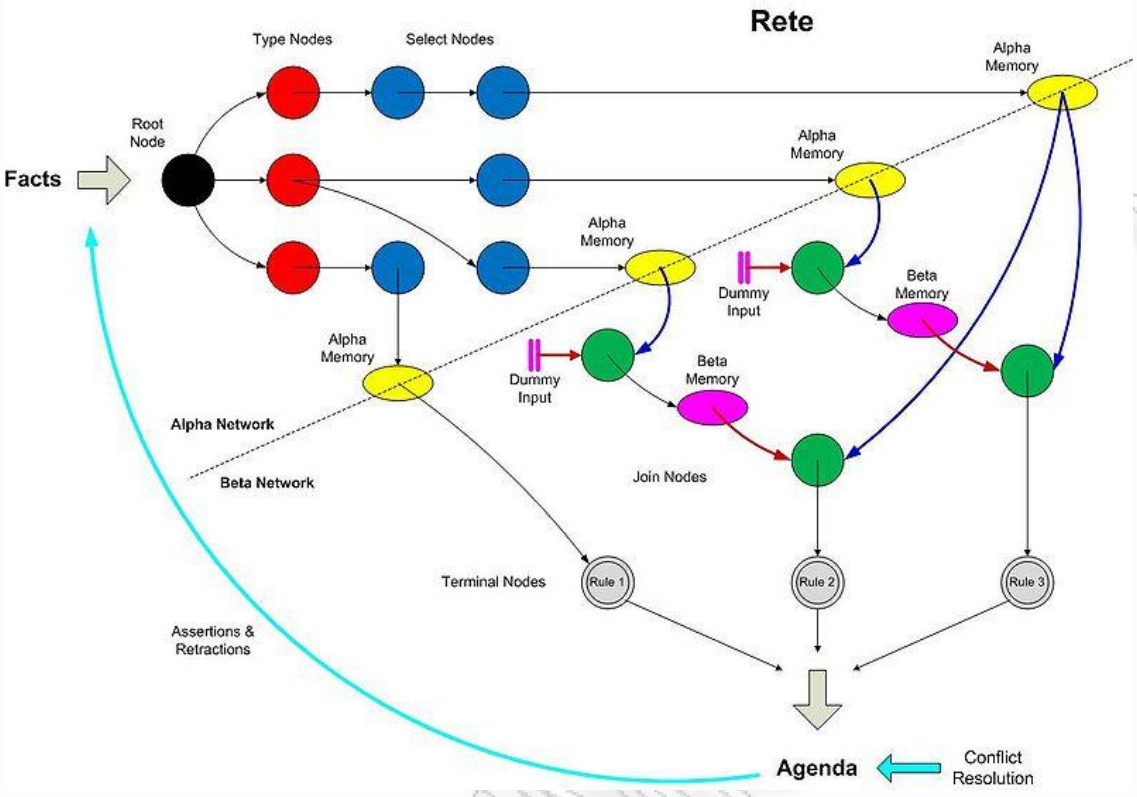
5.2.4 Ο αλγόριθμος Rete

Ο αλγόριθμος Rete είναι ένας αποτελεσματικός αλγόριθμος αντιστοίχησης προτύπων για την υλοποίηση συστημάτων κανόνων παραγωγής [7]. Είναι σχεδιασμένος να «θυσιάζει» μνήμη με σκοπό την αυξημένη ταχύτητα. Η υλοποίηση του Rete στο DROOLS καλείται ReteOO [5], η οποία αποτελεί μια εκσυγχρονισμένη και βελτιστοποιημένη έκδοση του απλού Rete και αφορά στα αντικειμενοστραφή συστήματα.

Ο αλγόριθμος Rete παρουσιάζει τα ακόλουθα βασικά χαρακτηριστικά [3w5]:

- Μειώνει ή εξαλείφει ορισμένους τύπους πλεονασμού με τη δυνατότητα διαμοίρασης κόμβων.
- Αποθηκεύει ενδιάμεσες τμηματικές αντιστοιχίες όταν εκτελεί ενώσεις μεταξύ διαφορετικών τύπων γεγονότων.
- Παρέχει τη δυνατότητα αποτελεσματικής αφαίρεσης στοιχείων μνήμης, όταν τα γεγονότα αφαιρούνται από τη λειτουργική μνήμη.

Ο αλγόριθμος Rete μπορεί να χωριστεί σε δύο μέρη: Μεταγλώττιση και εκτέλεση. Κατά τη φάση της μεταγλώττισης ο αλγόριθμος κατανέμει τους κανόνες στη μνήμη εργασίας με τέτοιο τρόπο ώστε να δημιουργήσει ένα αποτελεσματικό δίκτυο. Η ιδέα του δικτύου είναι να φιλτράρει τα δεδομένα όσο προωθούνται στους επόμενους κόμβους του δικτύου. Οι αρχικοί κόμβοι θα έχουν πολλές αντιστοιχίσεις και όσο προχωρούν στο δίκτυο θα γίνονται λιγότερες ώσπου να καταλήξουν στο τέλος του δικτύου στους τελικούς κόμβους. Για αυτό το λόγο και ο Rete χαρακτηρίζεται ως αλγόριθμος καθοδηγούμενος από τα δεδομένα. Γιατί ξεκινά με πολλά δεδομένα τα οποία προσπαθεί να αντιστοιχίσει σε κάποιον από τους ελάχιστους τελικούς κόμβους. Ένα Rete δίκτυο αποτυπώνεται στο σχήμα που ακολουθεί [3w5]:



Σχήμα 22: Δίκτυο Rete

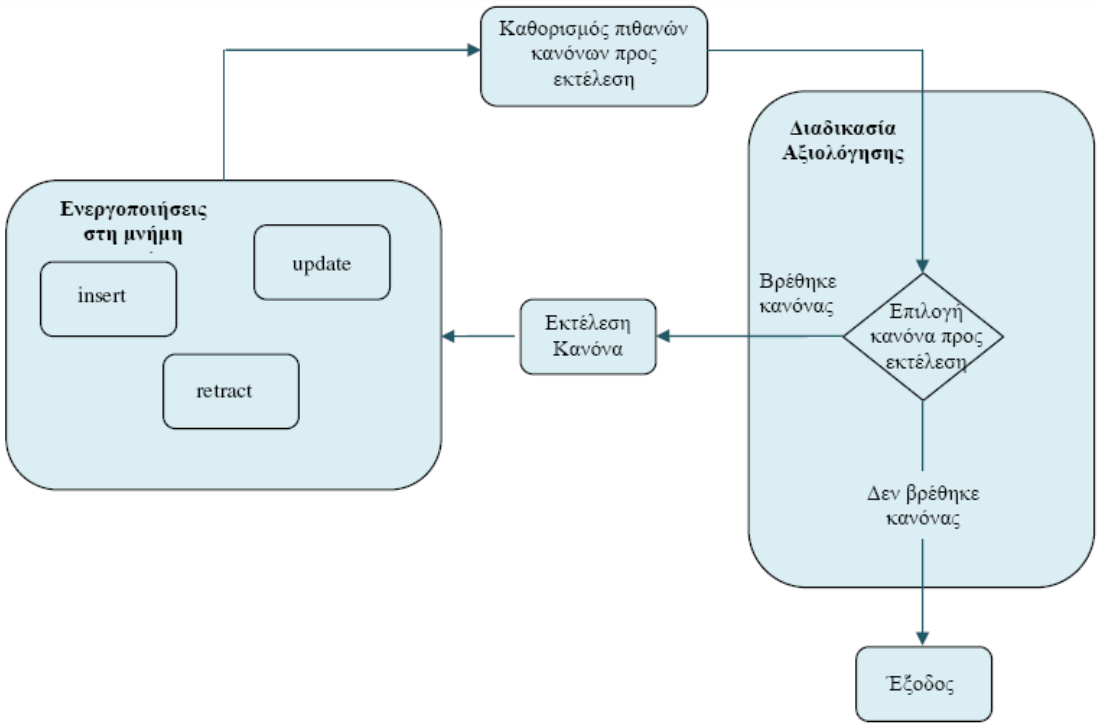
5.2.5 Ατζέντα

Η Agenda βασίζεται στον αλγόριθμο RETE. Κατά τη διαδικασία που η μνήμη επεξεργάζεται τα συμβάντα, κάποια από αυτά μπορεί να ικανοποιούν τις παραμέτρους κάποιων κανόνων. Αυτοί οι κανόνες είναι επιλεγμένοι προς εκτέλεση. Για κάθε ένα κανόνα που επιλέγεται προς εκτέλεση δημιουργείται και μια διαδικασία που πρέπει να εκτελέσει τον κανόνα και καλείται ενεργοποίηση (activation). Στο τέλος της διαδικασία επεξεργασίας των συμβάντων μπορεί να υπάρχουν πολλές ενεργοποιήσεις που πρέπει να εκτελεστούν. Η σειρά με την οποία θα εκτελεστούν καθορίζεται από την Agenda η οποία επιλέγει κάποια στρατηγική επίλυσης συγκρούσεων.

Ο μηχανισμός λειτουργεί σε δύο φάσεις οι οποίες επαναλαμβάνονται περιοδικά:

Στην πρώτη φάση πραγματοποιείται η επεξεργασία των συμβάντων στη μνήμη και δημιουργούνται, όπως προαναφέραμε οι ενεργοποιήσεις.

Στη δεύτερη φάση, η οποία καλείται φάση αξιολόγησης, εκτελούνται οι κανόνες που έχουν ενεργοποιηθεί. Εάν υπάρχει κάποιος κανόνας τότε εκτελείται, στην αντίθετη περίπτωση επιστρέφει στην πρώτη φάση. Η διαδικασία αυτή των δύο φάσεων εκτελείται μέχρι το σημείο που η Ατζέντα θα είναι κενή.



Σχήμα 23: Εκτέλεση σε δύο φάσεις

Στρατηγικές επίλυσης συγκρούσεων

Η επίλυση συγκρούσεων απαιτείται όταν πολλαπλοί κανόνες είναι προς εκτέλεση στην Ατζέντα. Η εκτέλεση ενός κανόνα μπορεί να προκαλέσει παρενέργειες στη μνήμη εργασίας, όπως για παράδειγμα ότι η εκτέλεση ενός κανόνα μπορεί να αφαιρέσει έναν άλλο κανόνα από τη μνήμη. Για αυτό το λόγο θα πρέπει ο μηχανισμός κανόνων να γνωρίζει με ποια σειρά πρέπει να εκτελεί τους κανόνες.

Οι στρατηγικές επίλυσης συγκρούσεων που εφαρμόζει το Drools είναι οι [5]:

Salience

Με την Salience ο χρήστης καθορίζει την προτεραιότητα με την οποία θα εκτελεστούν οι κανόνες. Ο κάθε κανόνας χαρακτηρίζεται με έναν βαθμό προτεραιότητας. Οι κανόνες με τους μεγαλύτερους αριθμούς εκτελούνται πρώτοι.

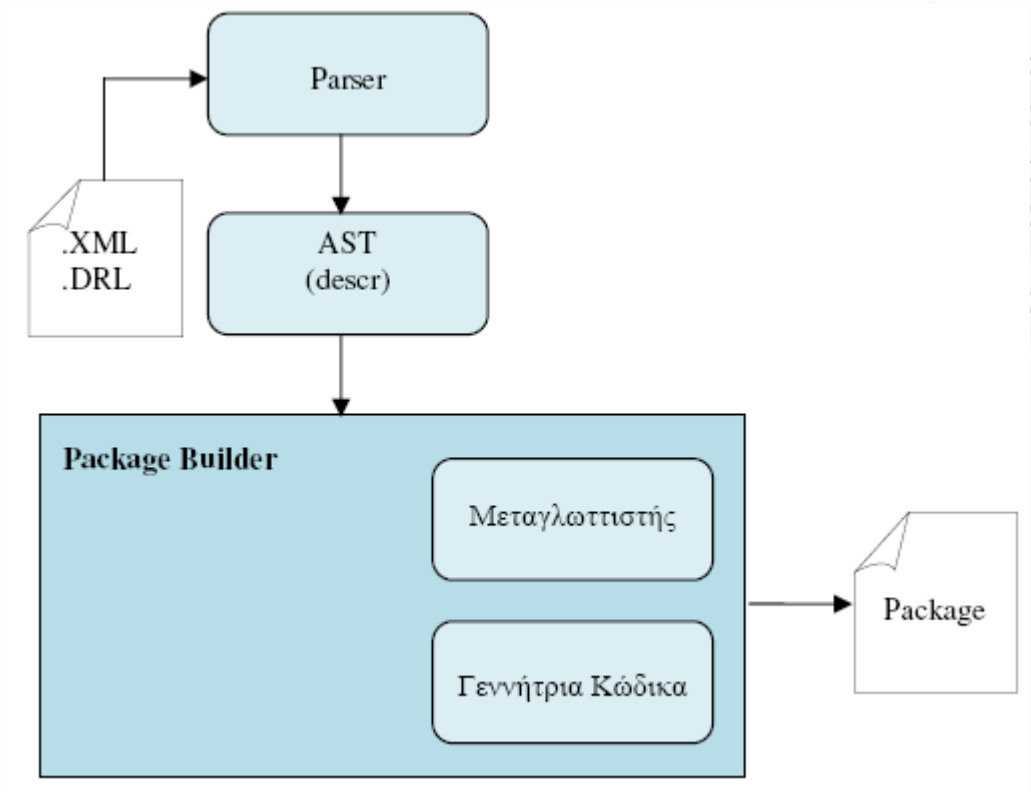
LIFO (Last In – First Out)

Στην περίπτωση αυτή η μνήμη καθορίζει τη σειρά εκτέλεσης των κανόνων. Σε κάθε ενέργεια της μνήμης επαληθεύονται κάποιοι κανόνες. Οι κανόνες αυτοί λαμβάνουν έναν αριθμό. Οι κανόνες με τον μεγαλύτερο αριθμό εκτελούνται πρώτοι. Οι κανόνες που «ανήκουν» σε μια ενέργεια παίρνουν τον ίδιο αριθμό και εκτελούνται χωρίς καθορισμένη σειρά.

5.3 Λειτουργία του Drools

Η λειτουργία του Drools χωρίζεται σε 2 διακριτές φάσεις: Συγγραφή (Authoring) και Runtime.

Η διαδικασία της Συγγραφής περιλαμβάνει τη δημιουργία των DRL ή XML αρχείων τα οποία τροφοδοτούν τον parser [5]. Ο parser στη συνέχεια ελέγχει εάν έχει ακολουθηθεί η σωστή γραμματική (Antlr 3) και παράγει μια ενδιάμεση δομή η οποία λέγεται descr και καθορίζει το AST που περιγράφει τους κανόνες. Κατόπιν, το AST προωθείται στον Package Builder ο οποίος παράγει τα Packages. Το Package αποτελεί ένα αντικείμενο το οποίο μπορεί να περιέχει έναν ή περισσότερους κανόνες, δηλαδή ένα η περισσότερα .drl αρχεία.

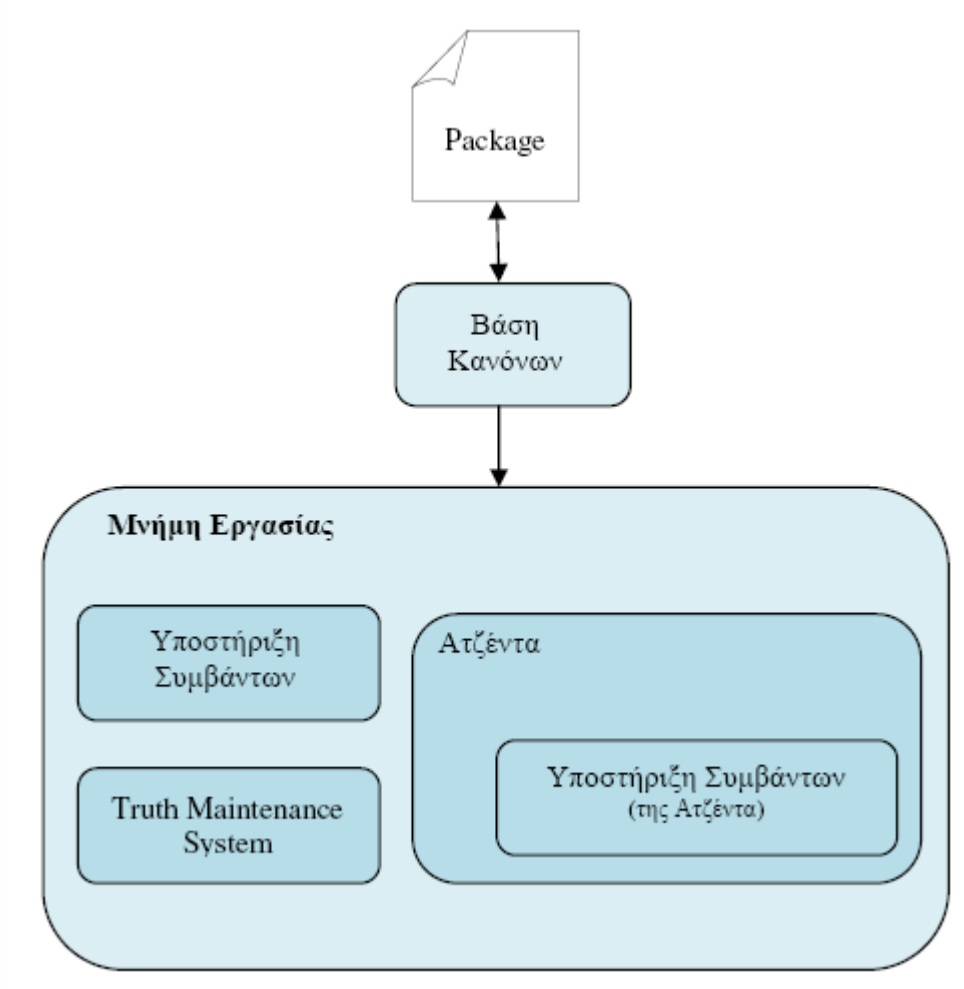


Σχήμα 24: Φάση Συγγραφής Κανόνων (Authoring)

Μετά τη συγγραφή των κανόνων που καταλήγει στην δημιουργία των packages, όπως είδαμε και στο προηγούμενο σχήμα, το package τροφοδοτεί τη RuleBase. Η Rulebase αποτελεί τμήμα της επόμενης διαδικασίας, δηλαδή της Runtime, και περιέχει ένα ή περισσότερα packages. Πακέτα μπορούν να προστεθούν ή να αφαιρεθούν από την Rulebase οποιαδήποτε στιγμή. Η μνήμη εργασίας (working memory) αποτελείται από υποτμήματα από τα οποία τα πιο βασικά είναι τα εξής:

- Working Memory Event Support
- Truth Maintenance System
- Agenda
- Agenda Event Support

Κατά την διαδικασία του Runtime μπορεί να εκκινηθούν μια ή περισσότερες ενέργειες. Για την εκτέλεση των ενεργειών αυτών υπεύθυνη είναι η Agenda [5].



Σχήμα 25: Φάση Εκτέλεσης Κανόνων (Runtime)

5.3.1 Facts

Τα Facts είναι αντικείμενα (beans) τα οποία ορίζονται και δημιουργούνται από την εφαρμογή και εισέρχονται στη μνήμη εργασίας. Τα Facts είναι οποιαδήποτε Java Objects στα οποία έχουν πρόσβαση οι κανόνες και αποτελούν τα δεδομένα της εφαρμογής. Για να μπορεί η εφαρμογή να επεξεργαστεί αυτά τα Facts θα πρέπει αυτά να ακολουθούν το πρότυπο των Javabean όσον αφορά τον καθορισμό των getters και setters για να αλληλεπιδρούν με το object. Κλάσεις χωρίς getters και setters δεν θεωρούνται «έγκυρα» facts.

5.3.2 Shadow Facts

Η λειτουργικότητα του Drools στη διαχείριση των Facts επεκτείνεται με τα Shadow Facts τα οποία αποτελούν ένα αντίγραφο των πραγματικών Facts, ως μηχανισμός συντήρησης της αλήθειας (truth maintenance) [5]. Κάθε έμπειρο σύστημα θα πρέπει με κάποιο τρόπο να διασφαλίζει ότι τα αποτελέσματα που παράγει είναι έγκυρα. Το Drools το εξασφαλίζει αυτό μέσω των shadow facts.

Κατά την διαδικασία επεξεργασίας των facts από την μηχανή εξαγωγής συμπερασμάτων, μπορεί κάποια από αυτά να μεταβληθούν από την ίδια τη διαδικασία. Η μηχανή κανόνων θα πρέπει κάθε στιγμή να γνωρίζει ποια facts έχουν «επηρεαστεί» κατά τη διαδικασία επεξεργασίας. Υπάρχουν δύο τρόποι αντιμετώπισης του ζητήματος αυτού. Ο ένας είναι να «κλειδώνουν» τα facts κατά τη διαδικασία εξαγωγής συμπερασμάτων. Ο δεύτερος τρόπος είναι

να δημιουργείται ένα αντίγραφο ενός object και όλες οι αλλαγές να γίνονται μέσω της μηχανής κανόνων. Τα shadow facts είναι σημαντικά, ιδιαιτέρα σε περιβάλλοντα όπου πολλές διεργασίες διαμοιράζονται μια μηχανή κανόνων. Εκτός από επιβεβαίωση των αποτελεσμάτων που παράγει η μηχανή κανόνων, τα shadow facts διευκολύνουν τους προγραμματιστές κατά τη διαδικασία ανάπτυξης μιας εφαρμογής διότι τους επιτρέπουν να διατηρούν ένα αρχείο των αλλαγών που υφίσταται ένα fact.

5.4 Δημιουργία κανόνων

Το Drools υποστηρίζει μια δικιά του γλώσσα κανόνων. Η μορφή της δεν είναι αυστηρή όσον αφορά τη σύνταξη και τη γραμματική της και μέσω “expanders” μπορεί να μοντελοποιήσει τη φυσική γλώσσα σε μορφή DSL (Domain Specific Languages). Η γραμματική πάνω στην οποία βασίζεται η γλώσσα του Drools είναι η Antlr3 [5].

Ένα αρχείο κανόνα

Ένα αρχείο κανόνα είναι ένα αρχείο με επέκταση .drl. Ένα αρχείο .drl μπορεί να περιέχει πολλούς κανόνες, ερωτήματα και συναρτήσεις, καθώς επίσης μεταβλητές και imports τα οποία χρησιμοποιούνται από τους κανόνες και τα ερωτήματα. Σε μια εφαρμογή που μπορεί να περιέχει πολλούς κανόνες μπορούν οι κανόνες αυτοί να κατανεμηθούν σε πολλά .drl αρχεία προκειμένου να μπορούν να είναι διαχειρίσιμα με αποτελεσματικότερο τρόπο. Η δομή ενός drl αρχείου είναι [5]:

package package-name
imports
globals
functions
queries
rules

Η σειρά με την οποία έχουν οριστεί τα διάφορα στοιχεία που περιέχονται στο αρχείο των κανόνων δεν έχει σημασία. Ο μόνος περιορισμός είναι να ορίζεται πάντα πρώτο το όνομα του πακέτου (package name). Δεν είναι υποχρεωτικό κάθε αρχείο κανόνων να περιέχει όλα τα στοιχεία που έχουν αναφερθεί, παρά μόνο αυτά που απαιτούνται για την συγκεκριμένη λειτουργία.

Οι ίδιοι οι κανόνες ακολουθούν τη δομή που έχουμε προαναφέρει:

```
rule "name"
attributes
when
        LHS
then
        RHS
end
```

Για τη δημιουργία των κανόνων το Drools παρέχει 4 επιλογές. Ο πιο άμεσος τρόπος συγγραφής κανόνων είναι σε έναν απλό επεξεργαστή κειμένου. Είναι όμως και ο πιο δύσχρηστος τρόπος εφόσον δεν παρέχει καμία ευκολία προς τον προγραμματιστή πχ για έλεγχο σφαλμάτων στον κώδικα.

Το Σύστημα Διαχείρισης Επιχειρησιακών Κανόνων (BRMS – Business Rules Management System), το οποίο ονομάζεται Guvnor, αποτελεί μια web-based εφαρμογή η πρόσβαση στην

οποία πραγματοποιείται μέσω browser και είναι εύκολη στη χρήση. Μέσω του browser επιτρέπεται η ταυτόχρονη πρόσβαση από μέλη ομάδων που εκτελούν το ίδιο έργο. Υποστηρίζει περίπου το σύνολο της λειτουργικότητας του Drools και αποτελεί τον πιο εύχρηστο τρόπο δημιουργίας κανόνων.

Ο τρόπος με τον οποίο εξασφαλίζεται η εκμετάλλευση του συνόλου της λειτουργικότητας που παρέχει το Drools είναι μέσω του IDE (**Integrated Development Environment**) του Eclipse. Το Drools εγκαθίσταται ως plug-in, δίνοντας πρόσβαση στον προγραμματιστή στις δυνατότητες του Eclipse για συγγραφή και απασφαλμάτωση των κανόνων. Μάλιστα παρέχεται και η δυνατότητα ταυτόχρονης χρήσης του Eclipse μαζί με το BRMS. Κανόνες που γράφονται στο ένα περιβάλλον μπορούν να εισαχθούν και στο άλλο για περαιτέρω επεξεργασία ή ολοκλήρωση με άλλα project.

Ένας ακόμη τρόπος συγγραφής κανόνων είναι με το Excel. Μέσω ενός κατάλληλα διαμορφωμένου προτύπου όπου οι γραμμές, σε ένα υπολογιστικό φύλλο, αναπαριστούν τους κανόνες και οι στήλες τις παραμέτρους, μπορούν να οριστούν κανόνες ιδιαίτερα στις περιπτώσεις που οι ίδιοι κανόνες επαναλαμβάνονται συνεχώς. Τα ειδικά διαμορφωμένα υπολογιστικά φύλλα ονομάζονται πίνακες απόφασης (Decision Tables).

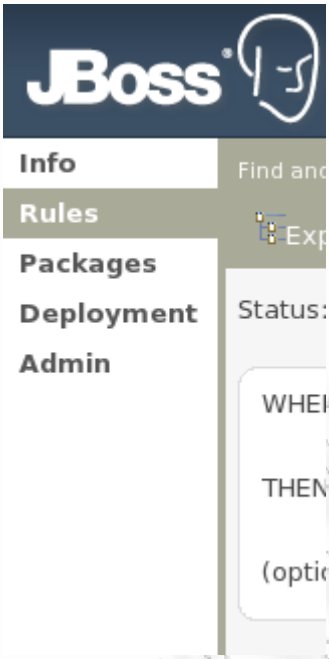
Στη συνέχεια γίνεται αναλυτική περιγραφή των τρόπων δημιουργίας κανόνων που προαναφέρθηκαν.

5.4.1 BRMS (Guvnor)

Όπως αναφέραμε το BRMS αποτελεί ένα ολοκληρωμένο σύστημα διαχείρισης κανόνων. Στο Drools το BRMS έχει μια διττή έννοια. Αποτελεί ένα τρόπο συγγραφής κανόνων αλλά και ένα σύστημα το οποίο μπορεί να ολοκληρωθεί και με άλλα συστήματα προκειμένου να αποτελέσει ένα κεντρικό σημείο διαχείρισης όλων των επιχειρησιακών κανόνων μιας επιχείρησης. Στη συγκεκριμένη παράγραφο θα ασχοληθούμε με το BRMS ως τρόπο για τη συγγραφή των κανόνων.

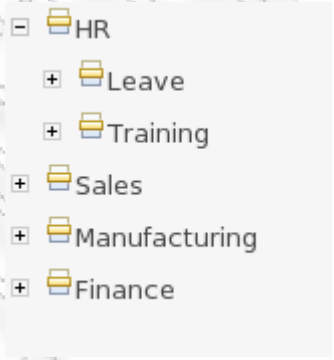
Το Guvnor υποστηρίζει τη συγγραφή, αποθήκευση, διαχείριση και ανάπτυξη των κανόνων. Από το screenshot που ακολουθεί διακρίνονται οι βασικές λειτουργίες του Guvnor:

- Info: Η αρχική οθόνη
- Rules: Οι κανόνες
- Package: Τα packages που αποτελούν ομαδοποιημένοι κανόνες.
- Deployment: Εφαρμογές που έχουν ολοκληρωθεί και εγκατασταθεί.
- Admin: Διάφορες λειτουργίες διαχείρισης



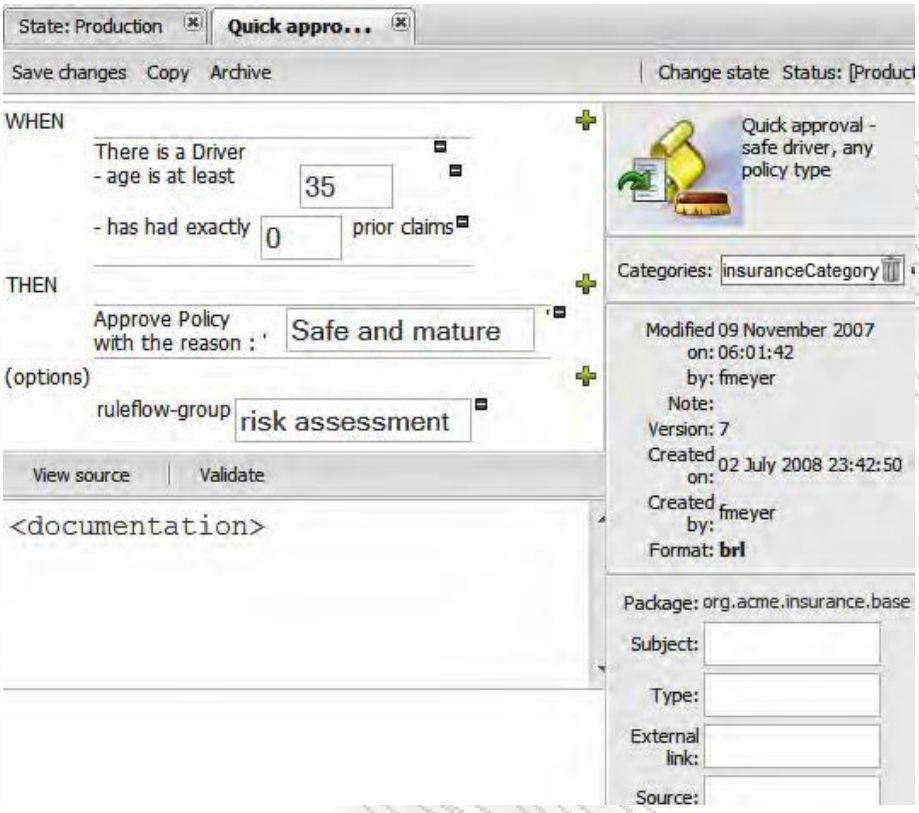
Συγγραφή κανόνων

Στη λογική του BRMS όλα χαρακτηρίζονται ως Assets και τα διαχειρίζεται με τον ίδιο τρόπο. Ως Assets χαρακτηρίζονται κανόνες, project, κλάσεις, categories. Οι κανόνες μπορεί να είναι σε πολλές μορφές όπως σε DRL, DSL, πίνακες απόφασης οι οποίες χαρακτηρίζονται ενιαία ως assets. Η κατηγορία1 περιέχει το object model βάσει του οποίου, ο κειμενογράφος κανόνων ορίζει τις μεταβλητές, συνθήκες και ενέργειες που υποστηρίζει το επιλεγμένο object model.



Οι κανόνες οι οποίοι δημιουργούνται από τον guided editor είναι της μορφής .BRL (Business Rules). Ο guided editor περιέχει πτυσσόμενα υπο-μενού και έτοιμες συνθήκες βάσει των οποίων δημιουργούνται σχηματικά οι κανόνες. Παράδειγμα ενός κανόνα φαίνεται στο screenshot που ακολουθεί:

[1] η κατηγορία είναι ένα είδος ομαδοποίησης όποιων κλάσεων οι οποίες μπορεί να βρίσκονται σε διαφορετικά packages



Σχήμα 26: Guvnor Guided Editor

Παρατηρούμε τον διαχωρισμό του κανόνα σε When και Then τμήμα.

5.4.2 Eclipse

Το Eclipse παρέχει ένα ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών. Το Drools ενσωματώνεται στο Eclipse ως plug-in από όπου και μπορεί να πραγματοποιείται η δημιουργία και διαχείριση των κανόνων καθώς και η ολοκλήρωση με το σύνολο της εφαρμογής. Το Eclipse είναι μια προγραμματιστική πλατφόρμα η οποία με την προσθήκη plug-in επεκτείνει τη λειτουργικότητά της. Με αυτή την έννοια μπορεί να χρησιμοποιηθεί όχι μόνο για τη συγγραφή κανόνων αλλά προσφέρει διασύνδεση με βάσεις, σχεδιασμό workflow κ.α. Τα βασικά χαρακτηριστικά που καλύπτει το IDE του Eclipse είναι:

- Γραφικό περιβάλλον για τη συγγραφή των κανόνων
  - Ο κειμενογράφος έχει επίγνωση της σύνταξης της DRL και παρέχει επιπλέον βοήθεια στη συγγραφή των κανόνων
  - Ο κειμενογράφος έχει επίγνωση και της DSL και παρέχει και βοήθεια στη συγγραφή των κανόνων σε αυτή τη μορφή.
- Οδηγοί (wizards)
  - Για τη δημιουργία νέων project
  - Δημιουργία DSL
  - Δημιουργίας πινάκων απόφασης

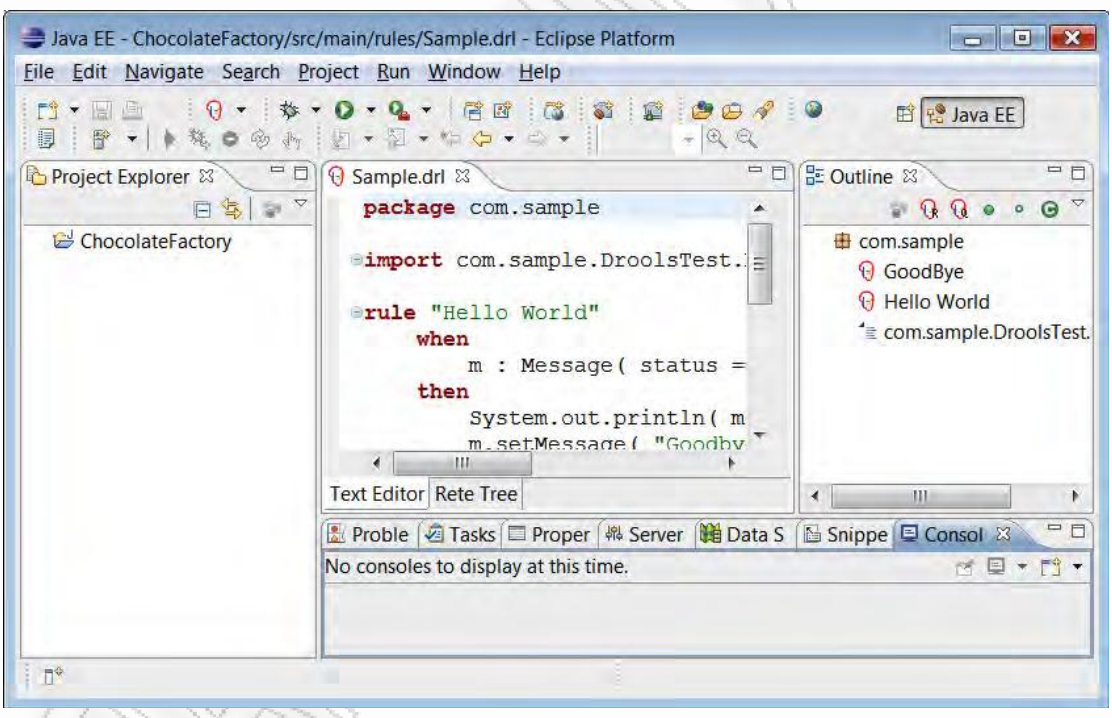
- Κειμενογράφο για DSL
    - Δημιουργία και διαχείριση της αντιστοίχησης μεταξύ τη φυσικής γλώσσας και της γλώσσας κανόνων
- Επαλήθευση των κανόνων

Το Eclipse υποστηρίζει δύο τρόπους συγγραφής κανόνων:

- Με κειμενογράφο (Textual Rule Editor)
- Με γραφικά καθοδηγούμενο περιβάλλον (Guided Editor)

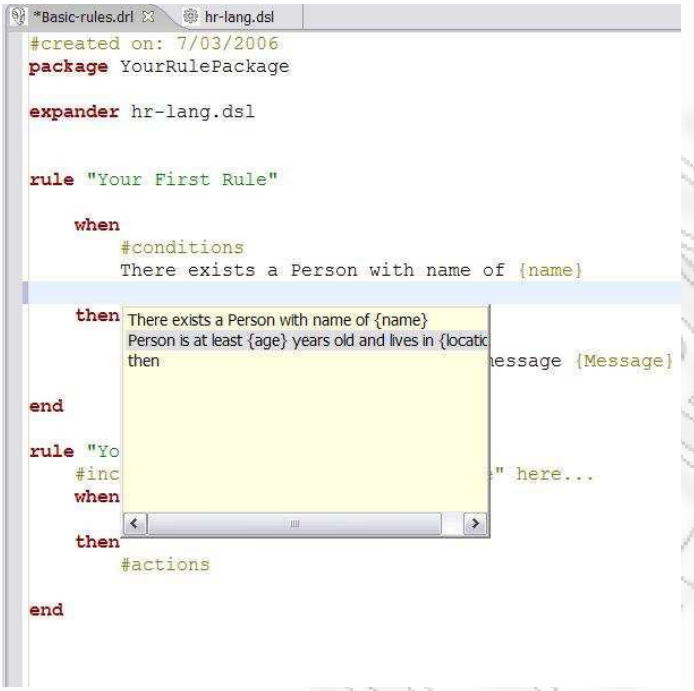
Ο Κειμενογράφος (Textual Rule Editor)

Ο κειμενογράφος στο Eclipse παρέχει όλες τις ευκολίες που παρέχουν στον προγραμματιστή και άλλα προγραμματιστικά περιβάλλοντα όπως την υπογράμμιση των λαθών και pop-up content assistant. Ο κειμενογράφος του Drools επεξεργάζεται αρχεία τα οποία έχουν επέκταση .drl. Στο screenshot που ακολουθεί, στο κεντρικό παράθυρο διακρίνεται ο βασικός κειμενογράφος.) Στην αριστερή κονσόλα εμφανίζονται τα projects και στην δεξιά εμφανίζεται ένα περίγραμμα των στοιχείων του αρχείου του οποίου η επεξεργασία γίνεται στο κεντρικό παράθυρο. Στο κάτω μέρος της οθόνης βρίσκεται η κονσόλα μηνυμάτων του Eclipse όπου εμφανίζονται πχ μηνύματα λάθους, διεργασίες παρασκηνίου κ.α.



Σχήμα 27: Eclipse Editor

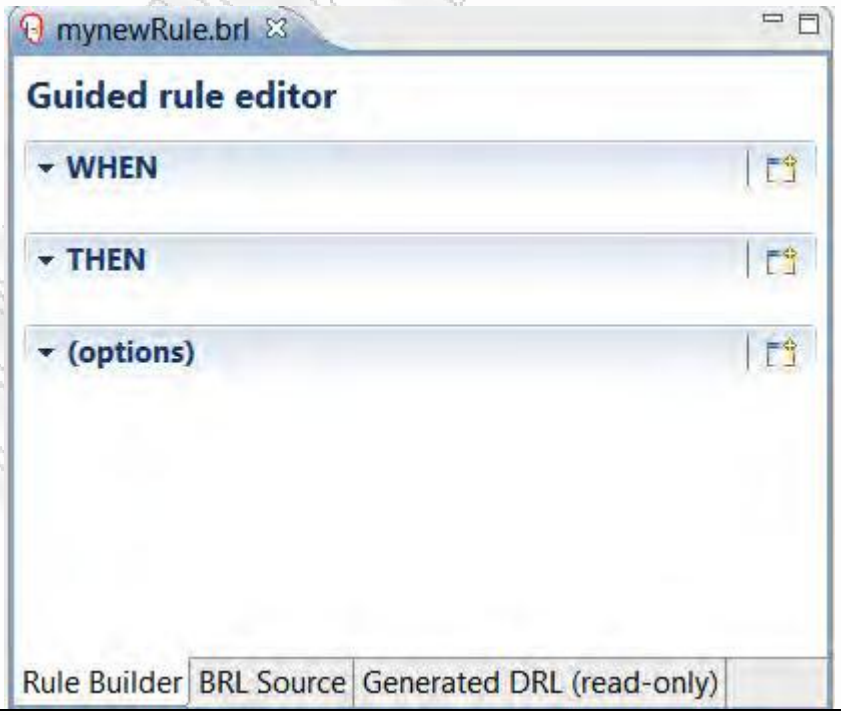
Κατά τη διαδικασία συγγραφής των κανόνων το Eclipse παρέχει διευκολύνσεις όπως pop up content assistance:



Σχήμα 28: Eclipse Editor Content Assistant

Το καθοδηγούμενο γραφικά περιβάλλον (Guided Editor)

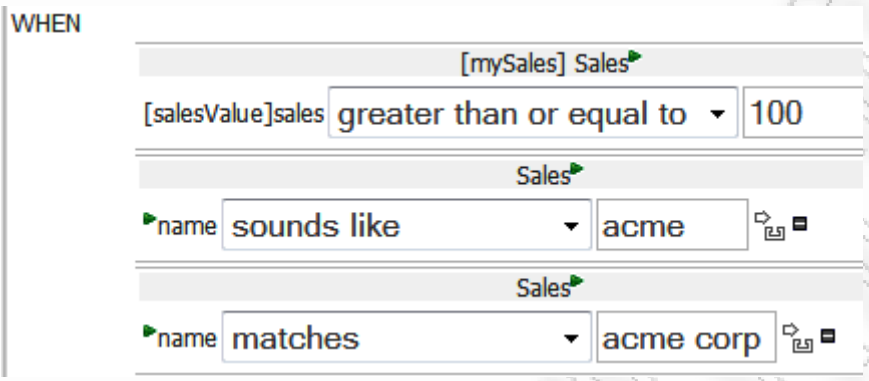
Ένας πιο εύχρηστος τρόπος για τη δημιουργία κανόνων που υποστηρίζει το Drools είναι μέσω ενός γραφικού περιβάλλοντος που καλείται Guided Editor. Η ίδια λειτουργικότητα υποστηρίζεται και στο BRMS. Η σύνταξη των κανόνων γίνεται μέσω drop-down όπου κατευθύνεται ουσιαστικά ο χρήστης στο να συμπληρώσει τις παραμέτρους. Όπως παρατηρούμε και στο screenshot που ακολουθεί υπάρχει από το γραφικό περιβάλλον ο διαχωρισμός σε When-Then.



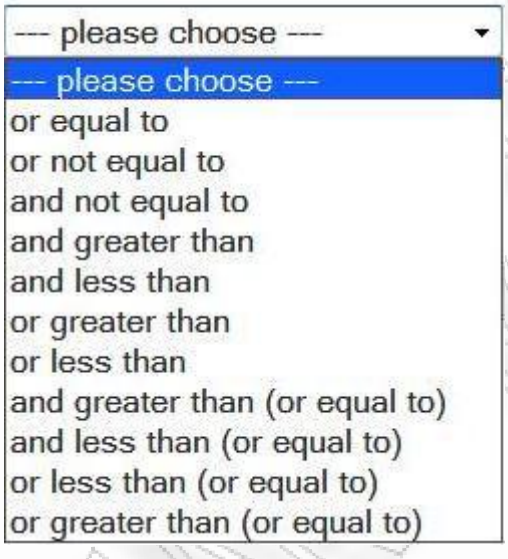
Σχήμα 29: Eclipse Guided Editor

Στη συνέχεια ο χρήστης συμπληρώνει στο κάθε τμήμα τις συνθήκες και τις ενέργειες καθοδηγούμενος από το γραφικό περιβάλλον οι παράμετροι του οποίου βασίζονται στο Object model το οποίο έχει δηλωθεί. Ακολουθεί ένα παράδειγμα από τη συμπλήρωση των τμημάτων When – Then:

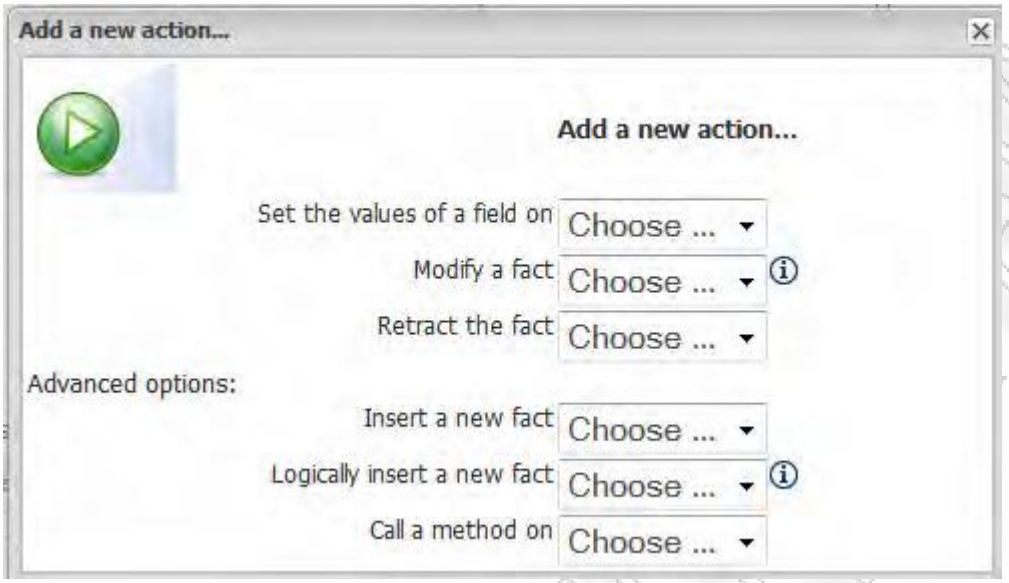
Το τμήμα When:



Για τη σύγκριση των μεταβλητών δίνονται οι επιλογές που ακολουθούν:



Το τμήμα Then:



5.4.3 Πίνακες Απόφασης

Οι πίνακες απόφασης (Decision Tables) αποτελούν μια ακριβή και συμπαγή μέθοδο αναπαράστασης της λογικής η οποία μπορεί να μοντελοποιήσει επιχειρησιακούς κανόνες.

Για αυτό το λόγο το Drools υποστηρίζει τη διαχείριση κανόνων σε μορφή υπολογιστικών φύλλων. Οι τύποι που υποστηρίζονται αυτή τη στιγμή είναι .xls και .csv. Οι πίνακες απόφασης στο Drools είναι ένας εύκολος τρόπος δημιουργίας κανόνων από τα δεδομένα τα οποία εισάγονται στο υπολογιστικό φύλλο. Εκτός από την ευκολία που παρέχει η μέθοδος αυτή, ένας ακόμη σημαντικός λόγος είναι ότι οι επιχειρήσεις χρησιμοποιούν υπολογιστικά φύλλα για να διαχειρίζονται τα δεδομένα τους και να εκτελούν υπολογισμούς, οπότε μπορούν σε αυτά τα υπολογιστικά φύλλα να ορίσουν και κανόνες.

Είναι ιδανική μέθοδος σε περιπτώσεις που θέλουμε να ελέγξουμε τις παραμέτρους των κανόνων χωρίς να αλλάξουμε απευθείας τους κανόνες αυτούς. Παράδειγμα ενός πίνακα απόφασης παρουσιάζεται στο σχήμα που ακολουθεί:

	A	B	C	D	E
1					
2			**Chocolate Trading Rules**		
3			RuleSet	net.firstpartners.chap8	
4			Notes	This decision table is to regulate the buying and selling of chocolate beans	
5			Import	net.firstpartners.exceldata.Cell,net.firstpartners.exceldata.Range	
6					
7					
8			RuleTable Evaluate the Buy Trades that we are interested in		
9			CONDITION	CONDITION	CONDITION
10			$r:Range		
11			eval($r.getRangeName().equals("$1"))	eval($r.getCell(1).getBooleanValue()==$1)	eval($r.getCell(4).isModified() == $1)
12		Note: This rule table checks broker specific prices (Buy)	Check the broker name (against the range)	Check if this is a buy offer (true)	And our cell as not yet been modified

Σχήμα 30: Πίνακες Απόφασης

Οι πίνακες απόφασης αποτελούν ουσιαστικά ένα εργαλείο που παράγει αυτόματα κανόνες της μορφής DRL. Κάθε γραμμή εκφράζει ένα κανόνα και κάθε στήλη αποτελεί είτε το μέρος της συνθήκης είτε το μέρος της ενέργειας του κανόνα:

RuleTable Evaluate the Buy Trades that we are interested in			
CONDITION	CONDITION	CONDITION	CONDITION
$r:Range			
eval($r.getRangeName().equals("$1"))	eval($r.getCell(1).getBooleanValue()==$1)	eval($r.getCell(4).isModified() == $1)	eval($r.getCell(3).getIntValue() < $1)
Check the broker name (against the range)	**Check if this is a buy offer (true)**	**And our cell as not yet been modified**	**As long as the price is less than**
A_Broker	true	false	300
C_Broker	true	false	400
E_Broker	true	false	500

Σχήμα 31: Μέρος της Συνθήκης (LHS)

ACTION	ACTION	ACTION	ACTION
$r.getCell(4).setValue("$1");	log.info("$1");	update($r.getCell(4));	update($r);
Update the status to _	**Log what we just did with the message _**	**Mark Cell as updated**	**Mark Range as updated**
Buy This	Plan to Buy from A Broker	blank	blank
Buy This	Plan to Buy from C Broker	blank	blank
Buy This	Plan to Buy from E Broker	blank	blank

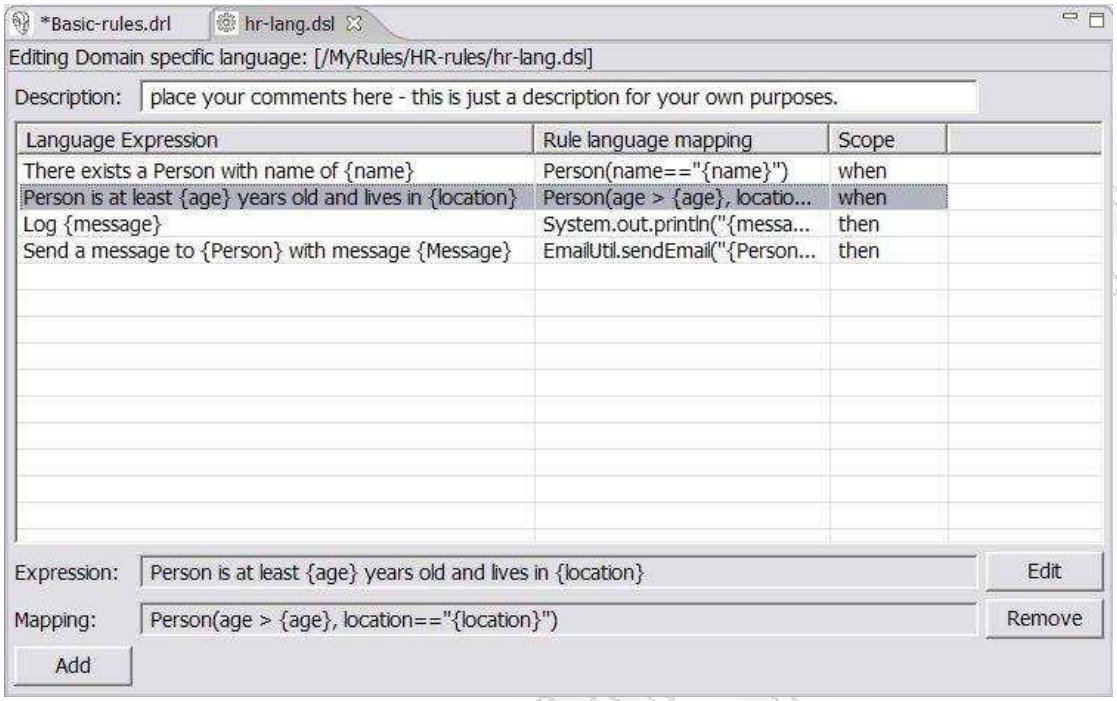
Σχήμα 32: Μέρος των Ενεργειών (RHS)

Ένα πλεονέκτημα από τη χρήση πινάκων απόφασης είναι η δυνατότητα ελέγχου των παραμέτρων των κανόνων χωρίς να επηρεάζονται άμεσα οι κανόνες. Για αυτό και αυτή η μέθοδος συνίσταται στις περιπτώσεις που έχουμε μεγάλους σε πλήθους κανόνες οι οποίοι ακολουθούν έναν ενιαίο τρόπο σύνταξης, δηλαδή παρουσιάζουν μια ομοιογένεια.

5.4.4 Γλώσσα Ειδικού Σκοπού (Domain Specific Languages)

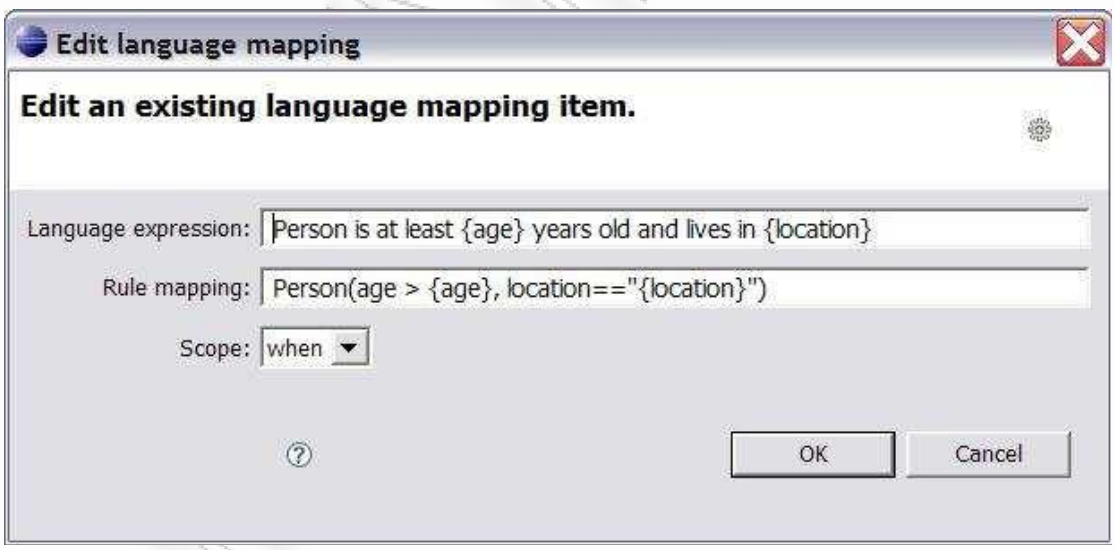
Όπως έχει αναφερθεί, οι γλώσσες ειδικού σκοπού (DSL) αποτελούν μια ακόμη μέθοδο γραφής κανόνων. Ένας πιο ακριβής ορισμός είναι ότι οι DSL αποτελούν μια επέκταση μιας γλώσσας κανόνα. Επιτρέπουν ουσιαστικά την έκφραση των κανόνων σε φυσική γλώσσα. Ένα βασικό χαρακτηριστικό των έμπειρων συστημάτων είναι η δυνατότητά τους να εμπλουτίζεται η βάση γνώσης απευθείας από τον ειδικό. Οι DSL παρέχουν ακριβώς αυτή τη δυνατότητα σε έναν μη-προγραμματιστή να μπορεί να δημιουργεί κανόνες οι οποίοι είναι εκφρασμένοι σε φυσική γλώσσα με τη μορφή προτάσεων. Αποτελεί επίσης και μια καλή μέθοδο απόκρυψης του τρόπου υλοποίησης ενός κανόνα και εμφάνισης μόνο της λογικής του. Ο ορισμός της DSL στο Drools μπορεί να γίνει τόσο στο IDE του Eclipse όσο και στο BRMS και τα αρχεία αυτά έχουν την επέκταση .dsl.

Η DSL προσθέτει ένα ενδιάμεσο στάδιο μεταξύ της φάσης σύνταξης ενός κανόνα και της φάσης εκτέλεσης του από την rule engine. Η ρυθμίσεις του DSL αποθηκεύονται σε μορφή κειμένου. Στην περίπτωση που για τη δημιουργία του DSL χρησιμοποιηθεί το IDE τότε υποστηρίζεται και η επαλήθευση της ορθότητας του DSL, αν και η δομή του αρχείου είναι σχετικά απλή. Το Drools επιτρέπει την προσαρμογή οποιουδήποτε τμήματος μιας γλώσσας συμπεριλαμβανομένων και των λέξεων-κλειδιών. Κανονικές εκφράσεις μπορεί να χρησιμοποιηθούν για την αντιστοίχηση με λέξεις/προτάσεις. Ακολουθεί παράδειγμα από δημιουργία .dsl στο Eclipse.



Σχήμα 33: Παράδειγμα ενός DSL

Παρατηρούμε την αντιστοίχηση μεταξύ των κανονικών εκφράσεων και των κανόνων. Αυτή η αντιστοίχηση χρησιμοποιείται από τον μεταγλωττιστή της μηχανής κανόνων για την μετατροπή της πρότασης σε κανόνα. Κατά τη σύνταξη του κανόνα καθορίζεται αν η πρόταση αναφέρεται στο τμήμα της συνθήκης ή στο τμήμα των ενεργειών. Εάν επιλεγεί στο πεδίο scope η επιλογή “When” τότε αναφέρεται στο LHS μέρος του κανόνα, εάν επιλεγεί το “Then” τότε αναφέρεται στο RHS μέρος του κανόνα.



Στο παραπάνω παράδειγμα, στον τύπο δεδομένων “Person” αντιστοιχίζονται οι μεταβλητές “age” και “location” οι τιμές των οποίων ανακτώνται από την αρχική πηγή των κανόνων.

5.4.5 Οι κανόνες σε μορφή XML

Το Drools υποστηρίζει επίσης τη δημιουργία και διαχείριση των κανόνων σαν XML δεδομένα [5]. Όπως ακριβώς και στη γλώσσα κανόνων του Drools (.DRL), και η μορφή των κανόνων σε XML αντιστοιχίζεται στην εσωτερική AST αναπαράσταση. Το Drools και στην XML μορφή υποστηρίζει το σύνολο της λειτουργικότητας που παρέχει και στην DRL μορφή.

Η δημιουργία κανόνων σε μορφή XML αποτελεί ίσως τον πιο δύσκολο τρόπο σε σχέση με τις επιλογές που έχουμε προαναφέρει όσον αφορά στην προγραμματιστική προσπάθεια. Ενώ από τους διάφορους τρόπους που περιγράψαμε υπάρχει μια τάση για την όλο και πιο εύκολη δημιουργία των κανόνων, με την XML επιστρέφουμε σε έναν αμιγώς προγραμματιστικό τρόπο.

Η XML χρησιμοποιείται στις περιπτώσεις που απαιτείται ο συνδυασμός της γλώσσας του Drools με κάποια άλλη γλώσσα κανόνων η οποία και αυτή να βασίζεται στην XML (είναι σχετικά εύκολη η μετατροπή μέσω XSLT από ένα XML format σε ένα άλλο). Επίσης είναι χρήσιμη και στην περίπτωση που το Drools θα πρέπει να ενσωματωθεί σε μια εφαρμογή όπου η επικοινωνία μεταξύ των διαφόρων λειτουργικών μονάδων της εφαρμογής μπορεί να γίνεται μέσω XML. Η χρήση της XML ως γλώσσα κανόνων έχει όλα τα πλεονεκτήματα της XML γενικότερα, δηλαδή ως μορφή διασύνδεσης με διαφορετικά συστήματα. Στη δυνατότητα αυτή έκφρασης των κανόνων σε XML βασίζεται και η μετεξέλιξη τόσο του Drools όσο και πολύ περισσότερο της EDA αρχιτεκτονικής στην ολοκλήρωση με τη SOA αρχιτεκτονική.

Το XSD το οποίο υποστηρίζεται και το οποίο περιγράφει την XML, είναι πλήρως συμβατό με το πρότυπο W3C. Ακολουθεί ένα σχετικά αναλυτικό παράδειγμα ενός κανόνα σε XML μορφή:

```
<rule name="simple_rule">
<rule-attribute name="salience" value="10" />
<rule-attribute name="no-loop" value="true" />
<rule-attribute name="agenda-group" value="agenda-group" />
<rule-attribute name="activation-group" value="activation-group" />
<lhs>
        <pattern identifier="cheese" object-type="Cheese">
                <from>
                        <accumulate>
                        <pattern object-type="Person"></pattern>
                                <init>
                                        int total = 0;
                                <init>
                                        <action>
                                                total += $cheese.getPrice();
                                        </action>
                                        <result>
                                                new Integer( total ) );
                                        </result>
                        </accumulate>
                </from>
        </pattern>
<pattern identifier="max" object-type="Number">
        <from>
                <accumulate>
                        <pattern identifier="cheese" object-type="Cheese"></pattern>
                        <external-function evaluator="max" expression="$price"/>
                </accumulate>
        </from>
</pattern>
</lhs>
<rhs>
        list1.add( $cheese );
</rhs>
</rule>
```

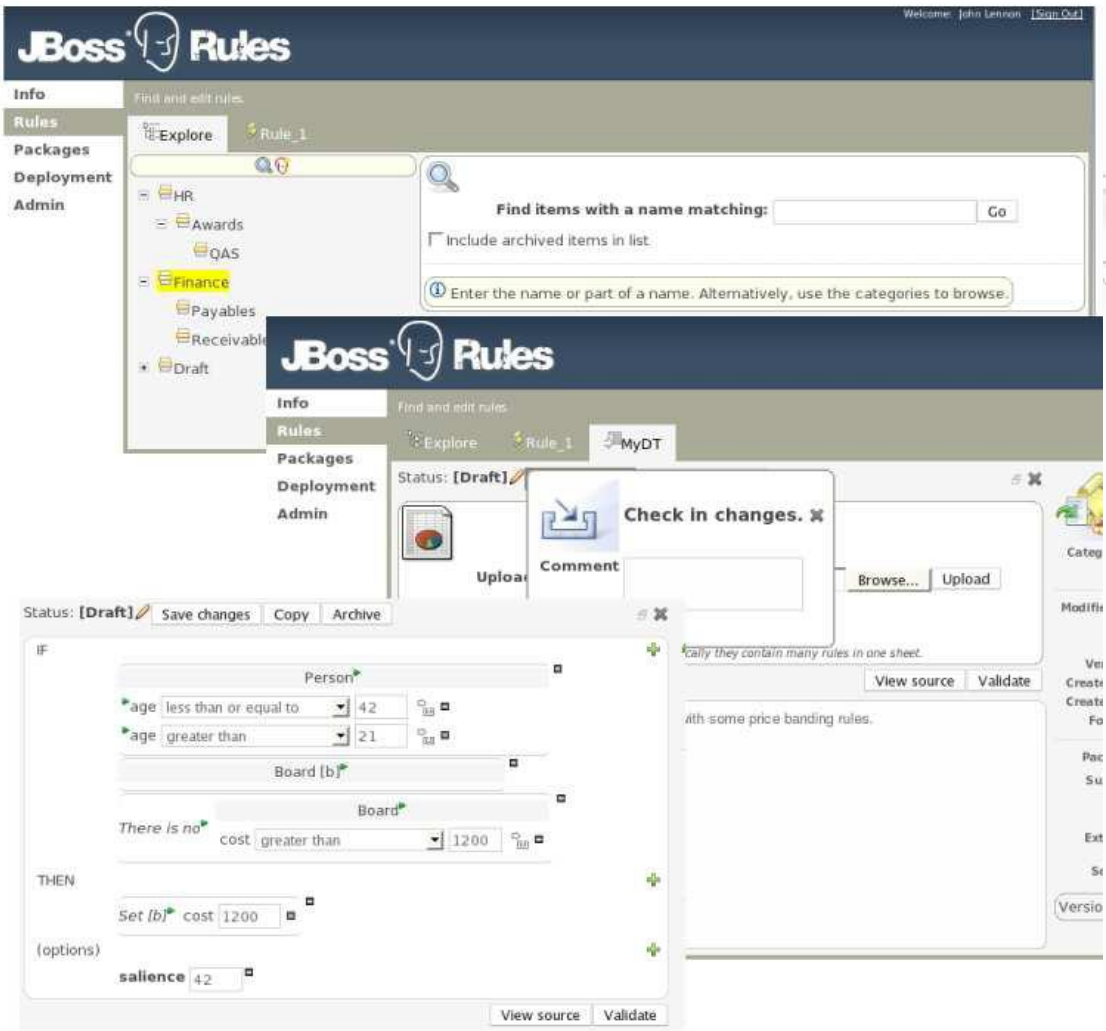
Αναλύοντας τον παραπάνω κανόνα παρατηρούμε ότι περιέχει ένα δεξί (<rhs>, </rhs>) και ένα αριστερό τμήμα (<lhs>, </lhs>). Το δεξί κομμάτι είναι αυτό που περιγράφει τις ενέργειες που πρέπει να εκτελεστούν όταν ικανοποιηθούν οι όροι που ορίζονται από το αριστερό τμήμα το οποίο είναι το τμήμα των συνθηκών

Το βασικό στοιχείο του LHS είναι το <pattern>. Αυτό καθορίζει την κλάση και πιθανώς και τις μεταβλητές οι οποίες σχετίζονται με το στιγμιότυπο (instance) μιας κλάσης. Μέσα στο <pattern> περιέχονται οι συνθήκες και οι περιορισμοί που πρέπει να ικανοποιηθούν.

5.5 Αρχιτεκτονική του Συστήματος (BRMS)

Ένα σύστημα διαχείρισης επιχειρησιακών κανόνων αποτελεί μια ολοκληρωμένη εφαρμογή που στον πυρήνα της έχει τη μηχανή κανόνων. Εκτός από τη δυνατότητα δημιουργίας κανόνων υποστηρίζει ταυτόχρονα τη διαχείριση και ανάπτυξη τους σε ένα επιχειρησιακό περιβάλλον καθώς επίσης και την ολοκλήρωση με άλλα συστήματα, την ανάλυση των αποτελεσμάτων και τη χρήση εργαλείων από τον τελικό χρήστη. Αποτελεί ένα πρόσθετο επίπεδο «πάνω» από τη μηχανή κανόνων που εξασφαλίζει επιπλέον λειτουργικότητα.

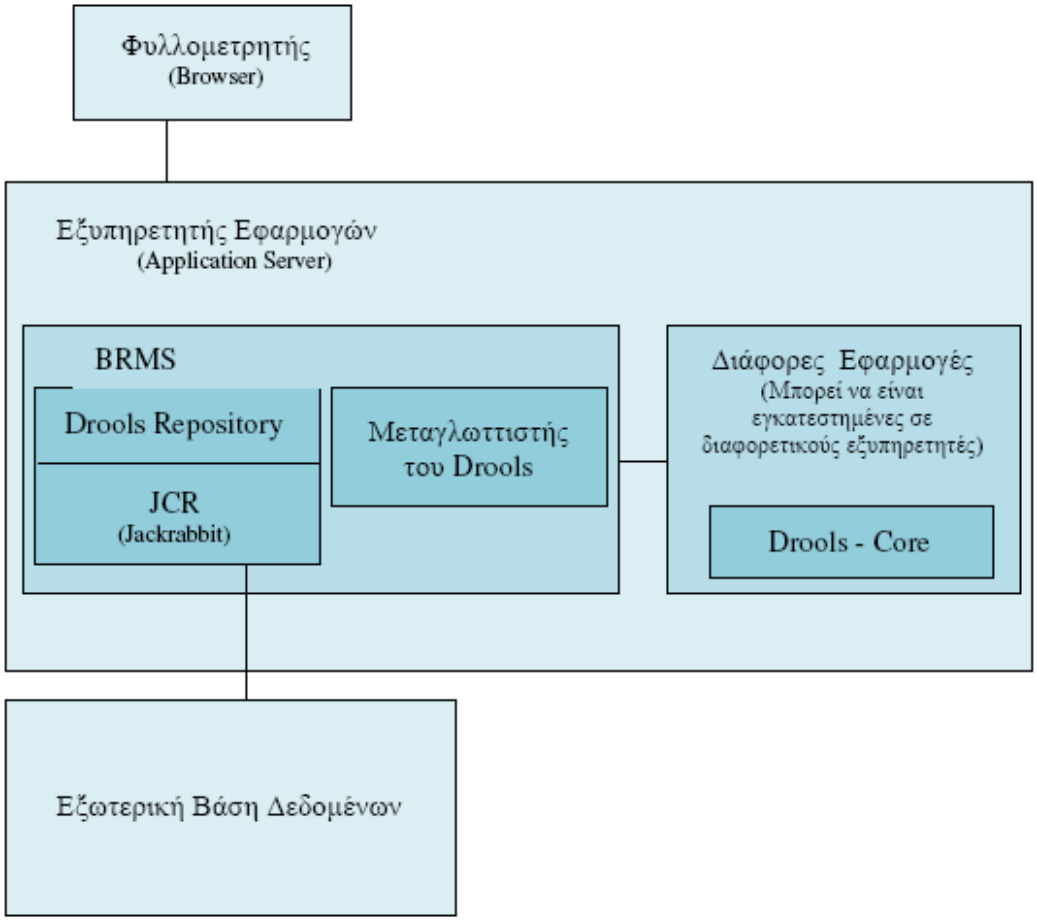
Ένα BRMS είναι ένα ολοκληρωμένο σύστημα ικανό να αναπτύσσει, εκτελεί, ελέγχει και συντηρεί ένα πλήθος πολύπλοκων λειτουργιών που υλοποιούν την λογική στη λήψη αποφάσεων από τα επιχειρησιακά συστήματα του οργανισμού ή της επιχείρησης. Η λογική αυτή υλοποιείται μέσω επιχειρησιακών κανόνων και περιλαμβάνει πολιτικές, απαιτήσεις και υπό συνθήκη εντολές οι οποίες καθορίζουν τις ενέργειες που πρέπει κατά περίπτωση να εκτελέσουν τα υπολογιστικά συστήματα. Επιπροσθέτως, μέσω του web interface, παρέχει πρόσβαση σε ένα ευρύτερο πλήθος χρηστών που περιλαμβάνει εκτός από τους προγραμματιστές και τους τελικούς χρήστες. Οι τελικοί χρήστες έχουν ένα κοινό σημείο αναφοράς το οποίο περιέχει το σύνολο των επιχειρησιακών κανόνων με απευθείας πρόσβαση και σε εύχρηστη διεπαφή.



Ένα BRMS θα πρέπει τουλάχιστον να περιλαμβάνει [5]:

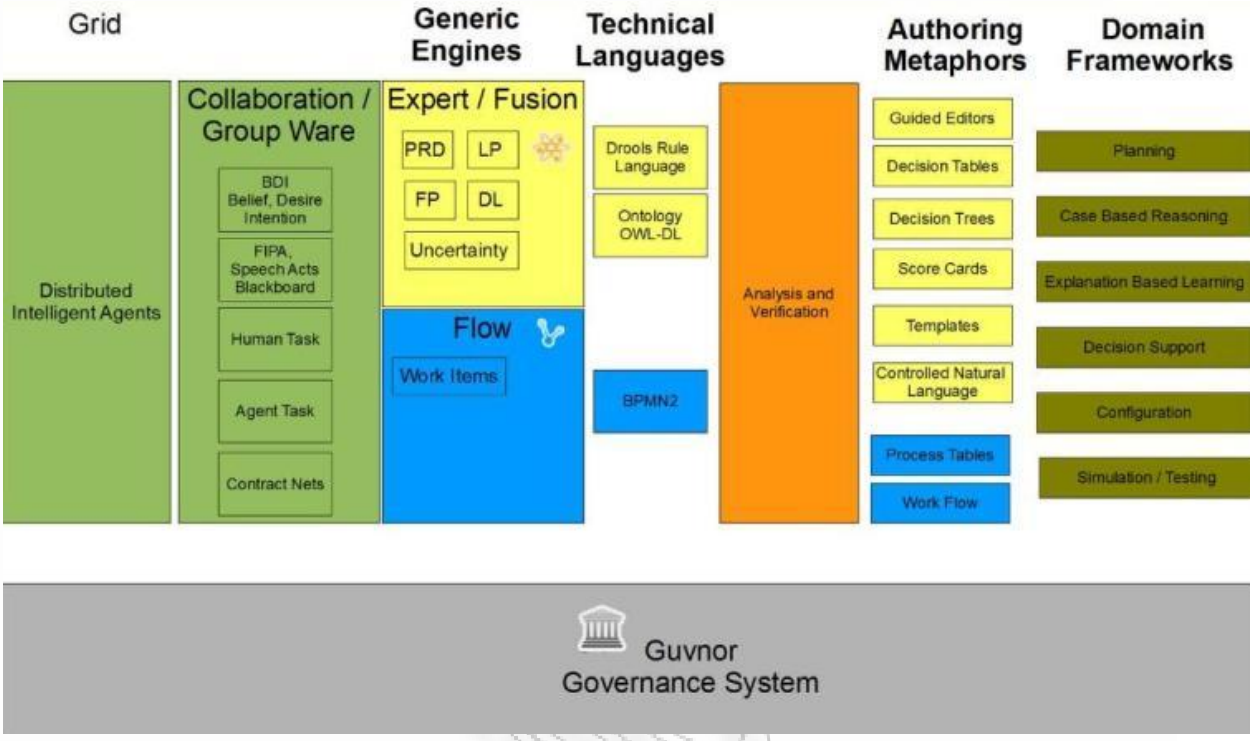
- Μια αποθήκη δεδομένων (repository) που να επιτρέπει την εξαγωγή της επιχειρησιακής λογικής η οποία είναι με τη μορφή κώδικα στα επιχειρησιακά συστήματα.
- Εργαλεία (Λογισμικό) βάσει των οποίων προγραμματιστές και αναλυτές θα χρησιμοποιήσουν για τον καθορισμό, υλοποίηση και διαχείριση της επιχειρησιακής λογικής.
- Runtime περιβάλλον, προκειμένου οι εφαρμογές να μπορούν να «ανασύρουν» την επιχειρησιακή λογική που διαχειρίζεται το BRMS και να την εκτελούν χρησιμοποιώντας τη μηχανή κανόνων.

Στο διάγραμμα που ακολουθεί [5] αποτυπώνονται οι βασικές λειτουργικές μονάδες που απαρτίζουν το σύστημα και ο τρόπος με τον οποίο διασυνδέονται μεταξύ τους. Το BRMS εγκαθίσταται στον application server ως αρχείο .war και η πρόσβαση επιτυγχάνεται μέσω φυλλομετρητή. Η διεπαφή είναι ajax based και βασίζεται στο GWT.



Σχήμα 34: Αρχιτεκτονική του BRMS

Για να κατανοήσουμε τις δυνατότητες του BRMS αρκεί να επισημάνουμε ότι στο μέλλον το Drools φιλοδοξεί, βασισμένο στο Guvnor, να δημιουργήσει ένα σύστημα το οποίο θα αποτελεί ένα κεντρικό σημείο αποθήκευσης και διαχείρισης όλων των assets μιας επιχείρησης όπως κανόνες, services, βάσεις δεδομένων, ροές εργασιών κ.α. Η αποθήκευση όλων των δεδομένων και των μεταδεδομένων που περιγράφουν όλα τα προαναφερθέντα assets, η διαχείριση τους σε όλο τον κύκλο ζωής τους, από την ανάλυση μέχρι την υλοποίηση και λειτουργία τους, καθώς και η πρόσβαση σε αυτά μέσω διαφόρων διεπαφων (eclipse, web) είναι η τελική μορφή του BRMS. Για το λόγο ότι θα ενσωματώνει όλες αυτές τις λειτουργίες η ομάδα του Drools ονομάζει αυτό το project ως Guvnor Governance System [3w4] και η αρχιτεκτονική του απεικονίζεται στο screenshot που ακολουθεί:



Σχήμα 35: Το Guvnor ως Σύστημα Διακυβέρνησης

5.6 Drools Fusion

Το Fusion είναι ένα module του Drools το οποίο επεκτείνει τη λειτουργικότητά του στην επεξεργασία σύνθετων συμβάντων (CEP) και στην Επεξεργασία Ροών Συμβάντων (ESP). Το Drools από τη έκδοση 5.0 και μετά υποστηρίζει sliding windows, temporal operators και type declarations προκειμένου να επεξεργαστεί νέφη και ροές συμβάντων [1]. Αν και μπορεί να λειτουργήσει ως ανεξάρτητο module είναι ολοκληρωμένο με όλα τα υπόλοιπα υποσυστήματα του Drools. Τα βασικά χαρακτηριστικά του, τα οποία επιτρέπουν την επεξεργασία συμβάντων, είναι συνοπτικά τα εξής:

- Αναγνώριση και επεξεργασία των συμβάντων
- Επιλογή των συμβάντων που έχουν μεγάλες πιθανότητες να αντιστοιχίζονται σε κάποιο πρότυπο από νέφη ή ροές συμβάντων
- Ανίχνευση των σχέσεων που υποκρύπτουν κάποιο πρότυπο μεταξύ των συμβάντων αυτών.
- Επιλογή των κατάλληλων ενεργειών ανάλογα με το πρότυπο που έχει αναγνωρισθεί.

Στο επόμενο κεφάλαιο γίνεται αναλυτική περιγραφή της λειτουργικότητας του Fusion.

6 Μελέτες Περίπτωσης

Στην παράγραφο αυτή θα μελετήσουμε συγκεκριμένα παραδείγματα εφαρμογής της τεχνολογίας επεξεργασίας συμβάντων μέσω της πλατφόρμας του Drools. Περιπτώσεις που συνήθως χρησιμοποιούνται μηχανές κανόνων είναι όταν έχουμε [2]:

- Συχνές αλλαγές στην επιχειρησιακή λογική.
- Οι domain experts (ή οι αναλυτές) είναι άμεσα διαθέσιμοι, αλλά όχι τεχνικά.
- Το πρόβλημα είναι πέρα από κάθε προφανή λύση βασισμένη σε αλγόριθμο.
- Το πρόβλημα μπορεί να μην είναι πολύπλοκο, αλλά δεν υπάρχει και κάποιος προφανής τρόπος επίλυσης.

Ορισμός ενός επιχειρησιακού κανόνα (business rule)

Επειδή στις επόμενες σελίδες θα αναφερθούμε στα «business rules», κρίνεται σκόπιμο να παραθέσουμε την περιγραφή τους. Ο όρος «επιχειρησιακός κανόνας» χρησιμοποιείται για τον προσδιορισμό ενός μη τεχνικού κανόνα, συνεπώς οι επιχειρησιακοί κανόνες θα μπορούσαν αντίστοιχα να καλούνται «ιατρικοί», «οικονομικοί», «ασφαλιστικοί», «καταβολής παροχών», κλπ. Ο όρος εξαρτάται από τον οργανισμό στον οποίο εργάζεται κανείς καθώς και από την εξειδίκευση που έχει σε αυτόν.

Ένας επιχειρησιακός κανόνας είναι κάθε κομμάτι γνώσης που μπορεί να εκφραστεί με την ακόλουθη μορφή: when ‘something' is true, then do 'this' [3w3]. Όταν «κάτι» ισχύει, τότε κάνε «αυτό»

Όλες οι εταιρείες και οι οργανισμοί έχουν ήδη θεσπίσει επιχειρησιακούς κανόνες, ακόμη και αν υπονοούνται ή καλύπτονται κάτω από χιλιάδες γραμμές κώδικα. Παραδείγματα αυτών των κανόνων μπορεί να είναι:

Όταν ο μισθός κάποιου είναι λιγότερος από 40.000 Ευρώ, τότε φορολόγησέ τον με 20%.

Όταν ένας υπάλληλος παίρνει προαγωγή, τότε δώσε του 10% αύξηση μισθού.

Στις επόμενες παραγράφους θα περιγράψουμε τη διαδικασία εγκατάστασης της πλατφόρμας Jboss Drools καθώς και δυο χαρακτηριστικές μελέτες περίπτωσης.

6.1 Εγκατάσταση της Πλατφόρμας Jboss Drools

Το λογισμικό που απαιτείται να εγκατασταθεί είναι ανοικτού κώδικα (open source) και είναι διαθέσιμο στο διαδίκτυο. Η εγκατάσταση του λογισμικού περιλαμβάνει τα εξής στοιχεία (με χρονική σειρά):

- Java JDK, SE Edition 6
- Jboss AS
- BRMS/Guvnor
- Eclipse
- Drools Plug-in

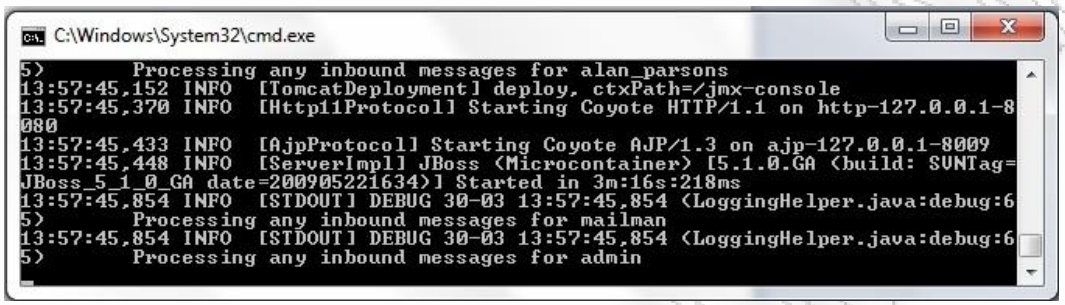
Η ανάλυση του παραπάνω λογισμικού έχει παρουσιαστεί στο προηγούμενο κεφάλαιο. Ακολουθεί η περιγραφή της διαδικασίας εγκατάστασης βήμα προς βήμα:

Java SE

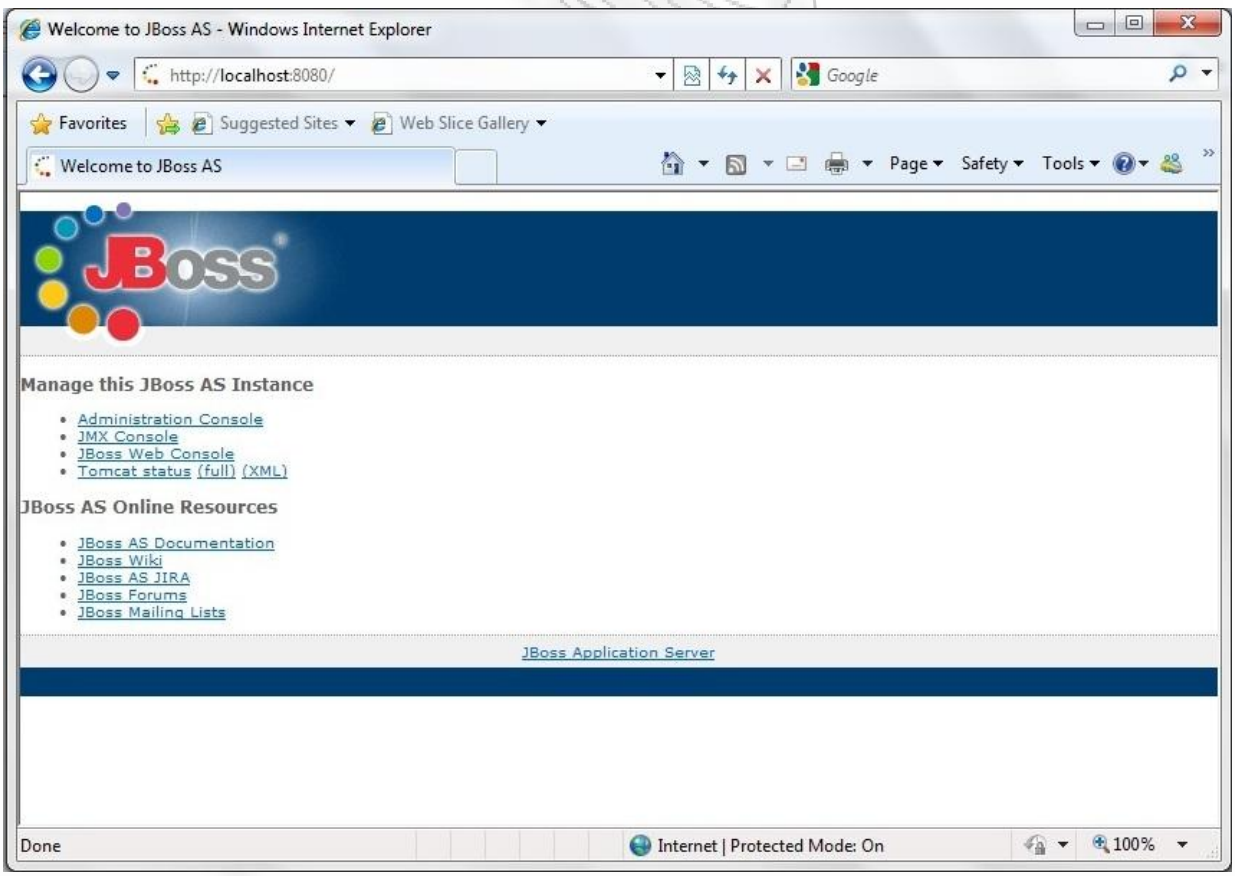
Καταρχήν πρέπει να εγκαταστήσουμε το περιβάλλον ανάπτυξης της Java. Από τη διεύθυνση http://www.oracle.com/technetwork/java/javase/downloads/index.html επιλέξαμε και εγκαταστήσαμε τη version 6 του JDK SE. Η εγκατάσταση του JDK περιλαμβάνει και το jre, το runtime environment της Java και η version του είναι η τελευταία (edition 6 update 24).

Jboss AS

Το Guvnor αποτελεί ένα web-based σύστημα διαχείρισης κανόνων. Μπορεί να εγκατασταθεί σε οποιονδήποτε Java-based Application/Web server όπως στους Websphere, Weblogic, Tomcat, αλλά στη συγκεκριμένη εγκατάσταση επιλέξαμε τον Aplication Server της Jboss κυρίως για το λόγο ότι το Drools αποτελεί προϊόν της ίδιας εταιρείας οπότε και αναμέναμε να υπάρχει πλήρης συμβατότητα και αξιοπιστία. Ο Jboss AS υπάρχει στη διεύθυνση: http://www.jboss.org/jbossas/downloads/ και η version την οποία εγκαταστήσαμε είναι η 5.1.0.GA.



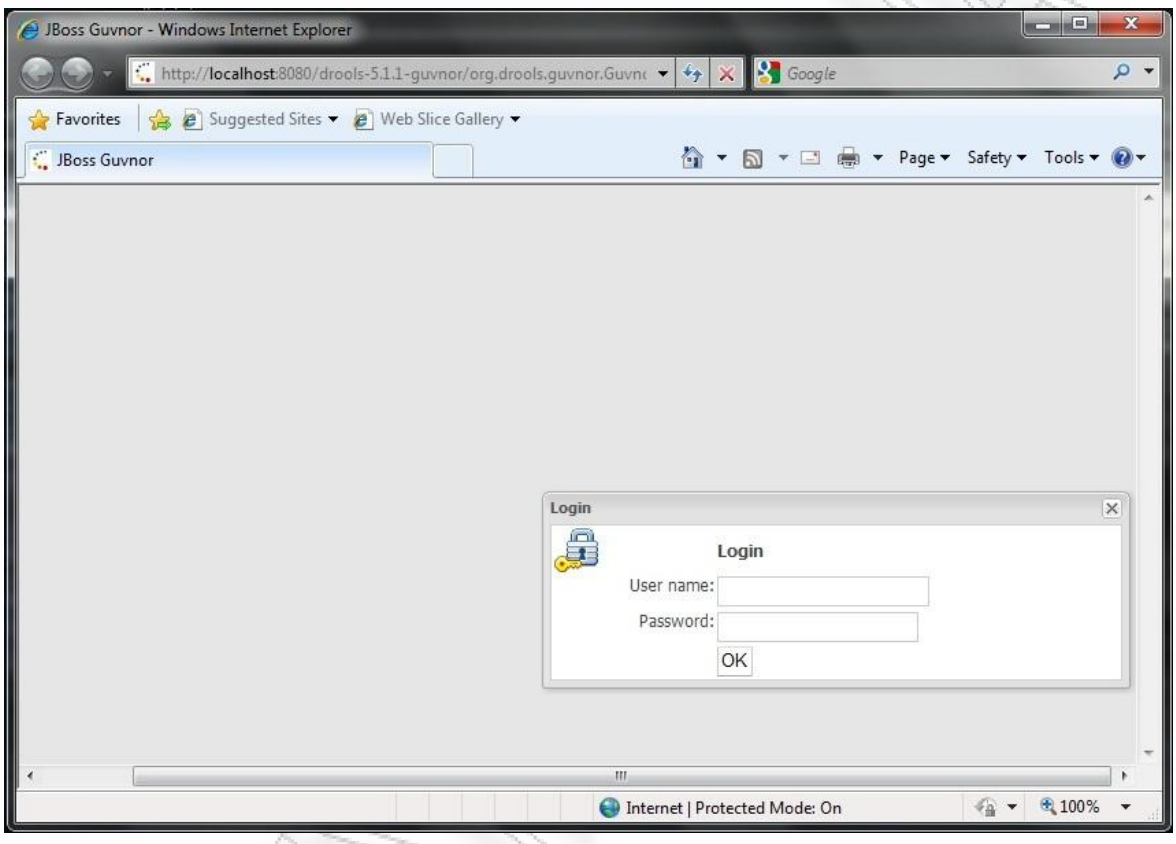
Για να εξακριβώσουμε ότι η εγκατάσταση έγινε σωστά θα πρέπει σε ένα browser να πληκτρολογήσουμε τη διεύθυνση: http://localhost:8080 και να εμφανιστεί η παρακάτω οθόνη:



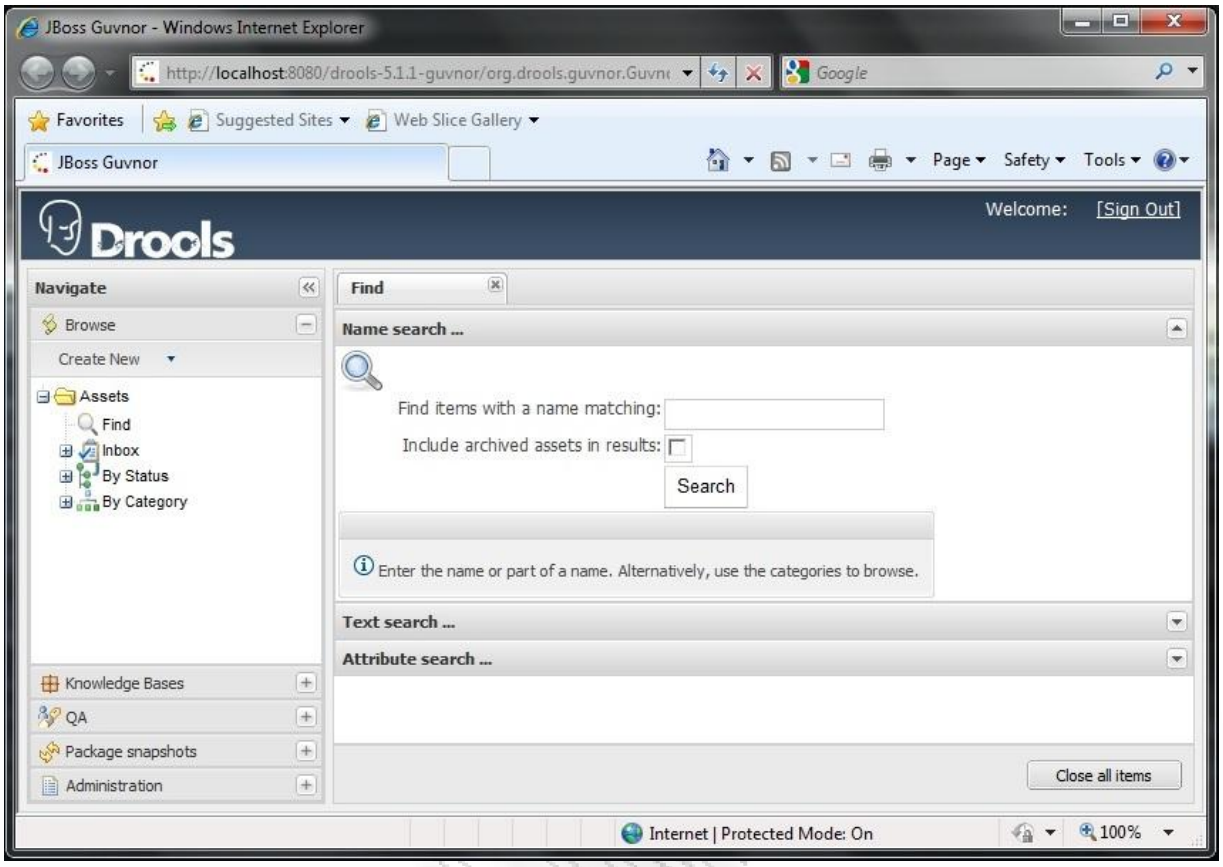
Guvnor

Εφόσον εγκαταστήσαμε τον application server μπορούμε να προχωρήσουμε με την εγκατάσταση του Gunvor. Το BRMS βρίσκεται στο site του Drools και συγκεκριμένα στη

διεύθυνση: http://www.jboss.org/drools/downloads.html. Στη λίστα των downloads υπάρχει η επιλογή για το Drools Guvnor. Επιλέγουμε το .war αρχείο, το κατεβάζουμε και το αντιγράφουμε στο φάκελο όπου έχουμε εγκαταστήσει τον application server, δηλαδή τον Jboss AS στην περίπτωσή μας. Η version που εγκαταστήσαμε είναι η 5.1.1. Στη συνέχεια πληκτρολογούμε τη διεύθυνση: http://localhost:8080/drools-5.1.1-guvnor και θα πρέπει να εμφανιστεί η οθόνη που ακολουθεί:



Μετά την επιλογή «OK» εμφανίζεται η αρχική οθόνη του Gunvor που είναι:

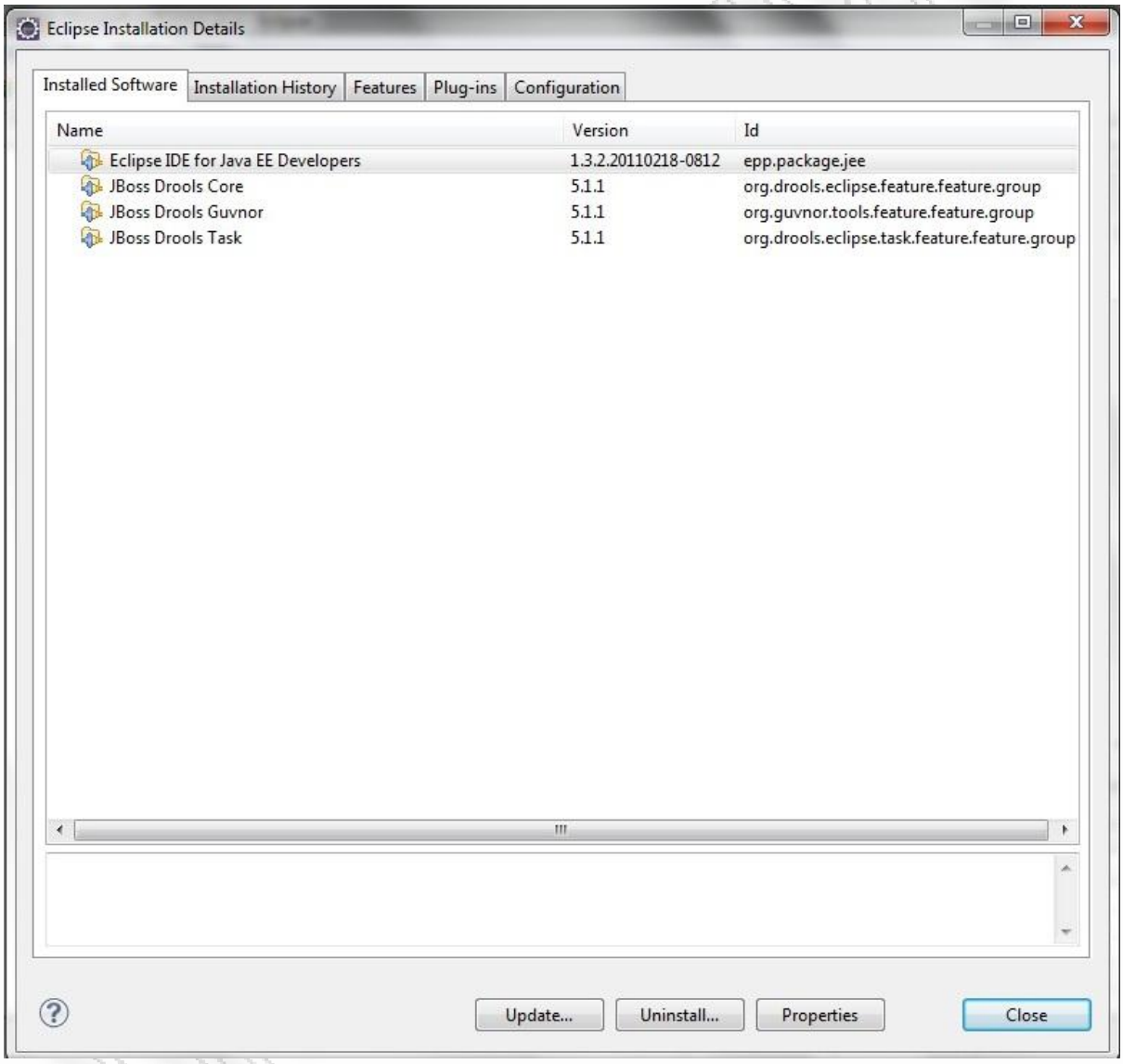


Eclipse

Το eclipse αποτελεί την πλατφόρμα η οποία παρέχει το γραφικό περιβάλλον προκειμένου να γίνει η συγγραφή των κανόνων καθώς επίσης και η εισαγωγή των δεδομένων. Το eclipse είναι ένα project ανοιχτού κώδικα και είναι διαθέσιμο στη διεύθυνση: http://www.eclipse.org/downloads/. Από τις επιλογές που προσφέρονται ενδιαφερόμαστε για το Eclipse IDE for Java EE Developers. Η version που εγκαταστήσαμε είναι η Helios Service Release 2, όπως διακρίνουμε και στο ακόλουθο screenshot:

Drools Plug-in

Όπως έχουμε προαναφέρει το Eclipse δεν είναι ένας απλός editor αλλά μια πλατφόρμα η οποία μπορεί να επεκτείνει τη λειτουργικότητά της. Με το Drools plug-in μπορούμε να γράψουμε και να απασφαλματώσουμε επιχειρησιακούς κανόνες. Το eclipse παρέχει έναν update manager από όπου παρέχει τη δυνατότητα εγκατάστασης εργαλείων ως plug-in. Το plug-in για το Drools είναι στη διεύθυνση: http://www.jboss.org/drools/downloads.html και από τις προσφερόμενες επιλογές αναζητείται το Drools IDE Update Site το οποίο είναι ουσιαστικά ένα url από όπου το Eclipse λαμβάνει το Drools. Το plug-in που εγκαταστήσαμε είναι της version 5.1.1 όπως φαίνεται και στο παρακάτω screenshot:



Το Drools Fusion

Όπως προαναφέραμε το Drools Fusion είναι το Module το οποίο προσφέρει τη δυνατότητα επεξεργασίας συμβάντων. Είναι σημαντικό να υπενθυμίσουμε ότι το Drools Fusion αποτελεί επέκταση του Drools Expert δηλαδή της core rule engine του Drools από τη version 5.0 και

μετά. Δηλαδή επεκτείνει τη λειτουργικότητα του Drools στη διαχείριση events χωρίς να απαιτείται ξεχωριστή εγκατάσταση. Τα βασικά χαρακτηριστικά του Fusion είναι [3w2]:

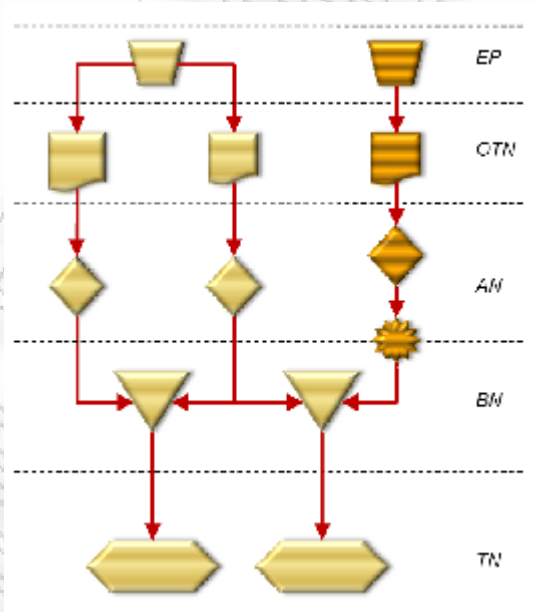
Αναγνώριση Συμβάντων

Τα συμβάντα είναι μια ειδική «οντότητα» δεδομένων τα οποία ανιχνεύονται όταν παρουσιάζεται μια σημαντική μεταβολή στην κατάσταση ενός συστήματος. Τα συμβάντα έχουν διακριτά χαρακτηριστικά όπως συγκεκριμένους χρονικούς περιορισμούς και σχέσεις. Το Fusion, αναγνωρίζοντας το είδος του συμβάντος, επιτρέπει στους προγραμματιστές να δημιουργήσουν επιχειρησιακούς κανόνες και διαδικασίες που θα ενεργούν στην παρουσία ή απουσία των συμβάντων.

```
17 # tells the engine that a StockTick instance will assume the
18 # role (semantics) of events and that the default retention
19 # policy will be 2 minutes
20 declare StockTick
21     @role( event )
22     @expires( 2m )
23 end
```

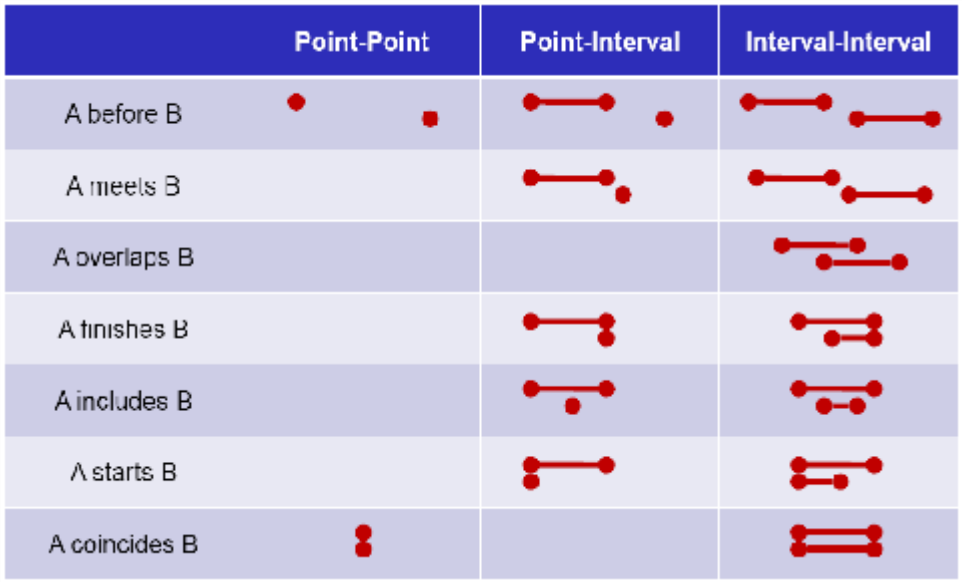
Υποστήριξη ασύγχρονων πολλαπλών ροών δεδομένων

Τα συμβάντα μπορεί να εμφανίζονται οποιαδήποτε στιγμή από πολλές πηγές οι οποίες μπορεί να είναι σε μορφή ροών ή νεφών. Το Fusion επεξεργάζεται τόσο ροές όσο και νέφη συμβάντων.



Υποστήριξη για χρονική συλλογιστική (temporal reasoning)

Τα συμβάντα έχουν ισχυρές χρονικές σχέσεις και περιορισμούς. Το Fusion επεκτείνει το Drools με μια σειρά χρονικών τελεστών που επιτρέπουν τη μοντελοποίηση και επεξεργασία των χρονικών συσχετίσεων μεταξύ των συμβάντων.



Υποστηρίζει garbage collection για τα συμβάντα

Τα συμβάντα κάποια στιγμή δεν χρειάζονται και πρέπει να αποσυρθούν. Ιδιαίτερα όταν αναφερόμαστε σε νέφη συμβάντων ο όγκος των δεδομένων αυξάνεται εκθετικά. Είναι κρίσιμο να «αποσύρονται» τα συμβάντα που πλέον δεν χρειάζονται προκειμένου να απελευθερώνονται υπολογιστικοί πόροι.

Υποστηρίζει συλλογιστική και στην απουσία συμβάντων

Όσο αναγκαία είναι η δημιουργία κανόνων και διαδικασιών σαν αντίδραση στην παρουσία συμβάντων, τόσο αναγκαίοι είναι και οι κανόνες και οι διαδικασίες στις περιπτώσεις που δεν υπάρχουν συμβάντα. Για παράδειγμα, ο κανόνας «Στην περίπτωση που η θερμοκρασία ξεπεράσει την καθορισμένη τιμή και για 10 λεπτά δεν έχουν παρθεί μέτρα, τότε σήμανε συναγερμό».

```
rule "reasoning on events over time"
when
   $a : A( )
   $b : B( this after[-2,2] $a )
   $c : C( this after[-3,4] $a )
   $d : D( this after[1,2] $b, this after[2,3] $c )
   not E( this after[1,10] $d )
then
   // do something
end
```

Υποστηρίζει Συρόμενα Παράθυρα (Sliding Windows)

Υπάρχουν περιπτώσεις όπου στην επεξεργασία συμβάντων απαιτούνται υπολογισμοί σε χρονικά παράθυρα. Επιπλέον, υποστηρίζει και συναρτήσεις ορισμένες από τους χρήστες.

Εφαρμογές του Drools Fusion

Το Fusion επεκτείνει τη λειτουργικότητα του Drools Expert με δυνατότητες επεξεργασίας συμβάντων και στοιχείων χρονικής συλλογιστικής. Επίσης παράσχει στο Drools την δυνατότητα να λειτουργήσει ως μια πλατφόρμα μοντελοποίησης CEP σεναρίων. Τα σενάρια αυτά συναντώνται συχνά σε ένα επιχειρηματικό περιβάλλον και απαιτούν ταχύτατη και αξιόπιστη αντιμετώπιση. Τέτοια σενάρια είναι [3w2]:

- Ανίχνευση Απάτης (Freud Detection)
- Έγκριση Πιστώσεων
- Αξιολόγηση Κινδύνου
- Τιμολόγηση Ασφαλίσεων
- Χρηματιστηριακές Συναλλαγές
- κ.α.

6.2 Μελέτη περίπτωσης για Fraud Detection

Ένα από τα πιο χαρακτηριστικά παραδείγματα όπου βρίσκει εφαρμογή το event processing είναι σε ένα τραπεζικό σύστημα ανίχνευσης απάτης (fraud detection). Σε ένα τέτοιο σύστημα θα πρέπει να διαχειριστούμε και να λάβουμε αποφάσεις πάνω σε ένα μεγάλο πλήθος συμβάντων τα οποία συμβαίνουν οποιαδήποτε χρονική στιγμή και πρέπει να τα επεξεργαστούμε άμεσα. Το σύστημα λοιπόν θα πρέπει να είναι σε θέση να αναλύσει τα συμβάντα αυτά και ανάλογα με το σύνολο των κανόνων που έχουμε ορίσει θα πρέπει να εκτελέσει τον κατάλληλο συναγερμό.

Καταρχήν θα πρέπει να ορίσουμε τους κατάλληλους επιχειρησιακούς κανόνες βάσει των οποίων ανιχνεύονται οι προσπάθειες «εξαπάτησης» του συστήματος και ταυτόχρονα να προσδιορίσουμε τις κατάλληλες ενέργειες για την αντιμετώπιση της απειλής. Μερικοί από τους κανόνες που θα εκτελεί το σύστημα είναι [1]:

1. Σε περίπτωση που λάβουμε ειδοποίηση από πελάτη ότι έχει κλαπεί η κάρτα του τότε μπλοκάρετε την ανάληψη χρημάτων από αυτό το λογαριασμό
2. Εάν δύο συναλλαγές αποσύρουν ποσό μεγαλύτερο από το 300% σε σχέση με το μέσο όρο των ποσών που αποσύρονται τις τελευταίες 30 μέρες τότε η κατάσταση των συναλλαγών αυτών χαρακτηρίζεται ως Minor απειλή.
3. Εάν ο αριθμός των συναλλαγών σε μια μέρα είναι μεγαλύτερος από 500% σε σχέση με το μέσο όρο των συναλλαγών και το υπόλοιπο του λογαριασμού είναι 10% λιγότερο από το μέσο όρο του υπόλοιπου τότε οι συναλλαγές χαρακτηρίζονται ως Minor απειλή.

Πριν προχωρήσουμε στην υλοποίηση των κανόνων αυτών θα πρέπει να ορίσουμε τις βασικές προδιαγραφές βάσει των προαναφερθέντων κανόνων. Καταρχήν θα πρέπει να ορίσουμε τα events τα οποία πρέπει να επεξεργαστούμε. Τα events αφορούν στον Λογαριασμό (account) και στη Συναλλαγή (transaction).

Όσον αφορά στα events της συναλλαγής πρέπει να έχουμε ένα event για την έναρξη της συναλλαγής (TransactionCreatedEvent) και ένα για τον τερματισμό της (TransactionCompletedEvent). Το κάθε transaction event καθορίζεται από έναν identifier, τον αριθμό λογαριασμού που χρεώνεται, τον αριθμό λογαριασμού που πιστώνεται και το ποσό της συναλλαγής. Θα πρέπει επίσης να υπάρχει και ένα event που να ορίζει μια συναλλαγή ως ύποπτη (SuspiciousTransaction) η οποία θα περιέχει το identifier της συναλλαγής και το βαθμό επικινδυνότητάς της (SeverityLevel).

Το βασικό event για τη Συναλλαγή (TransactionCreatedEvent) ορίζεται ως εξής:

```
public TransactionCreatedEvent(BigDecimal amount,
   Long fromAccountNumber, Long toAccountNumber,
   UUID tranasctionUuid) {
  super(amount, fromAccountNumber, toAccountNumber, tranasctionUuid);
}
```

Παρατηρούμε ότι καθορίζονται:

- Το ποσό - amount (BigDecimal),
- Ο αριθμός λογαριασμού του καταθέτη - fromAccountNumber (Long),
- Ο αριθμός του λογαριασμού του παραλήπτη - toAccountNumber(Long),
- Το αναγνωριστικό της συναλλαγής - UUID (transactionUuid)

Για τα events του λογαριασμού (account) θα πρέπει αν υπάρχει ένα event το οποίο να ανανεώνει το ποσό του λογαριασμού μετά από μια συναλλαγή και το οποίο είναι το AccountUpdatedEvent το οποίο περιέχει τον αριθμό λογαριασμού, το υπόλοιπο και το identifier της συναλλαγής που μετέβαλλε την κατάσταση του λογαριασμού. Επίσης πρέπει να υπάρχει και εδώ ένα event που θα χαρακτηρίζει ως ύποπτο τον λογαριασμό (SuspiciousAccount) αυτή τη φορά με τιμές επικινδυνότητας MINOR και MAJOR. Τέλος θα πρέπει να ορίσουμε και ένα event που θα προσδιορίζει μια κάρτα ότι έχει χαθεί (LostCardEvent), και αυτό το event θα περιέχει μόνο τον αριθμό λογαριασμού. Το event για την ενημέρωση του λογαριασμού ορίζεται ως εξής:

```
public class AccountUpdatedEvent implements Serializable {

 public final Long accountNumber;
 public final BigDecimal balance;
 public final UUID tranasctionUuid;

 public AccountUpdatedEvent(Long accountNumber,
    BigDecimal amount, BigDecimal balance,
    UUID tranasctionUuid) {
  super();
  this.accountNumber = accountNumber;
  this.balance = balance;
  this.tranasctionUuid = tranasctionUuid;
 }

 public Long getAccountNumber() {
  return accountNumber;
 }

 public BigDecimal getBalance() {
  return balance;
 }

 public UUID getTranasctionUuid() {
  return tranasctionUuid;
 }


 private transient String toString;

 @Override
 public String toString() {
  if (toString == null) {
    toString = new ToStringBuilder(this).appendSuper(
       super.toString()).append("accountNumber",
       accountNumber).append("balance", balance).toString();
  }
  return toString;
 }

}
```

Η rule engine του Drools, δηλαδή το Drools Expert, διαχειρίζεται τα δεδομένα ως facts και όχι ως events. Όταν ένα event εισέρχεται στη μνήμη εργασίας η μηχανή κανόνων «επιστρέφει» ένα FactHandle το οποίο επιτρέπει την διαχείριση του Object (δηλαδή των τιμών του Fact) κατά τη διαδικασία επεξεργασίας. Το Fusion επεκτείνει αυτή τη λειτουργικότητα με το EnentFactHandle προσθέτοντας επιπλέον πεδία, όπως το startTimestamp και το duration. Η δήλωση των events στο Drools γίνεται μέσω του συμβόλου «@». Για να δηλώσουμε για παράδειγμα ότι TransactionCreatedEvent είναι event γράφουμε:

```
Declare TransactionCreatedEvent
    @role (event)
End
```

Εάν το event χαρακτηρίζεται από διάρκεια (duration) τότε αντιστοιχίζεται με το πεδίο duration (durationProperty) του EventFactHandle με την εντολή:

```
@duration (durationProperty)
```

Πρέπει να αναφέρουμε ότι αρχικοποιούμε και τον τρόπο με τον οποίο δημιουργούμε τη Knowledgebase. Στο Fusion πρέπει να ορίσουμε ότι η Knowledgebase θα πρέπει να λειτουργεί σε event mode και να υποδέχεται events. Το Drools παρέχει δύο τρόπους διαμόρφωσης του τρόπου λειτουργίας της Knowledgebase για event processing. Είτε σε STREAM mode ή σε CLOUD mode. Στη περίπτωση του STREAM mode έχουμε:

```
KnowledgeBaseConfiguration config = KnowledgeBaseFactory
    .newKnowledgeBaseConfiguration();
config.setOption( EventProcessingOption.STREAM );
```

Κανόνας Ειδοποίησης (notification rule)

Ο πρώτος κανόνας που θα υλοποιήσουμε είναι η αναστολή ενός λογαριασμού όταν το σύστημα δεχθεί ένα LostCardEvent. Στο συγκεκριμένο κανόνα έχουμε αντιστοίχηση μεταξύ ενός fact (Account) και ενός event (LostCardEvent). Στην περίπτωση που έχουμε αντιστοίχηση τότε μεταβάλλεται το status του Account σε «Blocked». Ακολουθεί ο κώδικας [1]:

```
rule notification

    when
        $account : Account( status != Account.Status.BLOCKED )
        LostCardEvent( accountNumber == $account.number )
            from entry-point LostCardStream
    then
        modify($account) {
            setStatus(Account.Status.BLOCKED)
        };
end
```

Παρατηρούμε στη δομή του κανόνα ότι χωρίζεται σε δύο τμήματα. Στο τμήμα When όπου γίνεται η αντιστοίχιση μεταξύ των Account και των LostCardEvent, στη γραμμή LostCardEvent(accountNumber == $account.number), και στο τμήμα των ενεργειών Then όπου, σε περίπτωση που βρεθεί αντιστοίχιση, ο κώδικας μεταβάλλει την κατάσταση του account σε BLOCKED (setStatus(Account.Status.BLOCKED)).

Προκειμένου να εξακριβώσουμε τη σωστή λειτουργία του κανόνα θα δημιουργήσουμε ένα test case. Για όλους τους κανόνες που θα αναφέρουμε θα δημιουργήσουμε διάφορα test cases. Για τα test cases θα πρέπει να δημιουργήσουμε μια κλάση για την αρχικοποίηση των παραμέτρων της εφαρμογής. Η κλάση καλείται CepTest και είναι [1]:

```
public class CepTest {
    static KnowledgeBase knowledgeBase;
    StatefulKnowledgeSession session;
    Account account;
    FactHandle accountHandle;
    SessionPseudoClock clock;
    TrackingAgendaEventListener trackingAgendaEventListener;
    WorkingMemoryEntryPoint entry;
    @Before
    public void initialize() throws Exception {
        KnowledgeSessionConfiguration conf =
            KnowledgeBaseFactory.newKnowledgeSessionConfiguration();
        conf.setOption( ClockTypeOption.get( "pseudo" ) );
        session = knowledgeBase.newStatefulKnowledgeSession(conf,
            null);
        clock = (SessionPseudoClock) session.getSessionClock();
        trackingAgendaEventListener =
            new TrackingAgendaEventListener();
        session.addEventListener(trackingAgendaEventListener);
        account = new Account();
        account.setNumber(123456l);
        account.setBalance(BigDecimal.valueOf(1000.00));
        accountHandle = session.insert(account);
```

Στο συγκεκριμένο κομμάτι του κώδικα έχουμε αρχικοποίηση:

- Του ρολογιού (Clock):

Το κάθε event χαρακτηρίζεται από ένα timestamp. Το Drools έχει υλοποιήσει δύο τύπους ρολογιών, το Real Time Clock που χρησιμοποιεί το ρολόι του συστήματος, και ένα Pseudo Clock το οποίο ρυθμίζεται από την εφαρμογή. Για λόγους testing χρησιμοποιείται το Pseudo Clock για να μπορεί να υπάρχει έλεγχος του χρόνου ενώ το Real Time clock χρησιμοποιείται σε πραγματικό επιχειρησιακό περιβάλλον. Στο συγκεκριμένο test case η εντολή που ορίζει τον τύπο του ρολογιού είναι: conf.setOption(ClockTypeOption.get("pseudo")

- Της κλάσης Account:

Ορίζεται ο αριθμός του λογαριασμού ως 1234561 (account.setNumber(123456l), και ισοζύγιο στα 1000 account.setBalance(BigDecimal.valueOf(1000.00)).

΄Ελεγχος του κανόνα - notification rule

Αρχικά το τεστ ελέγχει εάν ο λογαριασμός έχει ανασταλθεί λόγω λάθους. Στη συνέχεια εισέρχονται τα events από το LostCardStream (session.getWorkingMemoryEntryPoint("LostCardStream"). και συνεχίζει με το επόμενο event (entry.insert(new LostCardEvent(account.getNumber())). Τέλος ανιχνεύει ότι η κατάσταση του event είναι Blocked (Account.Status.BLOCKED)

```
@Test
public void notification() throws Exception {
    session.fireAllRules();
    assertNotSame(Account.Status.BLOCKED,account.getStatus());
    entry = session
        .getWorkingMemoryEntryPoint("LostCardStream");
```

```
    entry.insert(new LostCardEvent(account.getNumber()));
    session.fireAllRules();
    assertSame(, account.getStatus());
}
```

Κανόνας Δύο μεγάλες αναλήψεις (rule twoLargeWithdrawals)

Εάν δύο συναλλαγές αποσύρουν ποσό μεγαλύτερο από το 300% σε σχέση με το μέσο όρο των ποσών που αποσύρονται τις τελευταίες 30 μέρες τότε η κατάσταση των συναλλαγών αυτών χαρακτηρίζεται ως Minor απειλή.

Το αξιοσημείωτο στον συγκεκριμένο κανόνα είναι ότι ενδιαφερόμαστε μόνο για συναλλαγές ενός λογαριασμού τις 30 τελευταίες μέρες. Σε αυτή την περίπτωση θα χρησιμοποιήσουμε τη δυνατότητα του Fusion για *sliding time windows*. Δηλαδή ενδιαφερόμαστε για τα events του συγκεκριμένου χρονικού διαστήματος που επεξεργαζόμαστε. Ένα επίσης πλεονέκτημα από τον περιορισμό των events που πρέπει να επεξεργαστούμε είναι ότι όταν τα events ξεπεράσουν αυτό το χρονικό διάστημα και δεν είναι πια απαραίτητα, το Drools τα αφαιρεί απελευθερώνοντας υπολογιστικούς πόρους.

Ο μέσος όρος των χρημάτων που έχει γίνει ανάληψη υπολογίζεται ως ο μέσος όρος των ποσών των TransactionCompletedEvent events οι οποίες πραγματοποιήθηκαν τις 30 τελευταίες μέρες (over window:time(30d) from entry-point TransactionStream). Στη συνέχεια απομένει η εύρεση 2 συναλλαγών οι οποίες πραγματοποιήθηκαν τα τελευταία 90 sec με ποσό μεγαλύτερο κατά 300% από το μέσο όρο. Στην περίπτωση αυτή χρησιμοποιούμε το TransactionCreatedEvent. Η υλοποίηση έχει ως εξής [1]:

```
rule twoLargeWithdrawals
dialect "mvel"
    when
    $account : Account( )
    Number( $averageAmount : doubleValue ) from accumulate(
        TransactionCompletedEvent( fromAccountNumber ==
        $account.number, $amount : amount )
        over window:time( 30d ) from entry-point
        TransactionStream, average( $amount ) )
    $t1 : TransactionCreatedEvent( fromAccountNumber ==
        $account.number, amount > ($averageAmount * 3.00) ) over
        window:time(90s) from entry-point TransactionStream
    $t2 : TransactionCreatedEvent( this != $t1,
        fromAccountNumber == $account.number,
        amount > ($averageAmount * 3.00) ) over
        window:time(90s) from entry-point TransactionStream
then
    insert(new SuspiciousAccount($account.number,
        SuspiciousAccountSeverity.MINOR));
    insert(new SuspiciousTransaction($t1.transactionUuid,
        SuspiciousTransactionSeverity.MINOR));
    insert(new SuspiciousTransaction($t2.transactionUuid,
        SuspiciousTransactionSeverity.MINOR));
end
```

Στον κώδικα παρατηρούμε τον υπολογισμό της μέσης τιμής ενός λογαριασμού average($amount) τις τελευταίες 30 μέρες (over window:time(30d)). Στη συνέχεια γίνεται αντιστοίχιση αυτού το λογαριασμού με δύο διαφορετικές συναλλαγές (TransactionCreatedEvents) όπου το ποσό είναι κατά 300% μεγαλύτερο από το μέσο όρο (amount > ($averageAmount * 3.00) και πραγματοποιήθηκαν σε χρονικό διάστημα 90 sec (over window: time(90s) from entry-point TransactionStream).

Στην περίπτωση που επαληθευθούν οι παραπάνω προϋποθέσεις τότε δημιουργούνται τρία νέα events, SuspiciousAccount, SuspiciousTransaction(για το $t1) και SuspiciousAccount (για το $t2) όπου χαρακτηρίζονται και τα τρία ως ύποπτα με βαθμό MINOR.

Έλεγχος του κανόνα - twoLargeWithdrawals rule

Στη συνέχεια δημιουργούμε ένα test case που ενεργοποιεί τον κανόνα. Δημιουργούμε έξι events. Δύο events (TransactionCompletedEvent) που ορίζουν το μέσο όρο του ποσού ((400+600)/2 = 500) και τα υπόλοιπα τέσσερα είναι τα πιθανά events (TransactionCreatedEvent) που μπορεί να χαρακτηριστούν ως ύποπτα. Το πρώτο δεν έχει ποσό τριπλάσιο από το μέσο όρο (100< 1500) και το δεύτερο εισέρχεται μετά από 91 sec οπότε δεν ενεργοποιούν τον κανόνα. Τα δύο τελευταία πληρούν και τις δύο προϋποθέσεις οπότε ο κανόνας ενεργοποιείται και χαρακτηρίζονται ως «ύποπτοι». Η υλοποίηση του test είναι ως εξής [1]:

```
@Test
public void twoLargeWithdrawals() throws Exception {
    entry = session
        .getWorkingMemoryEntryPoint("TransactionStream");
    transactionCompletedEvent(400);
    clock.advanceTime(5, TimeUnit.DAYS);
    transactionCompletedEvent(600);
    clock.advanceTime(11, TimeUnit.DAYS);

    transactionCreatedEvent(100);
    clock.advanceTime(30, TimeUnit.SECONDS);
    transactionCreatedEvent(1600);
    assertNotFired("twoLargeWithdrawals");

    clock.advanceTime(91, TimeUnit.SECONDS);
    transactionCreatedEvent(2100);
    assertNotFired("twoLargeWithdrawals");

    clock.advanceTime(30, TimeUnit.SECONDS);
    transactionCreatedEvent(1700);
    assertFired("twoLargeWithdrawals");
}
```

Κανόνας Υψηλής Δραστηριότητας (rule highActivity)

Εάν ο αριθμός των συναλλαγών σε μια μέρα είναι μεγαλύτερος από 500% σε σχέση με το μέσο όρο των συναλλαγών και το υπόλοιπο του λογαριασμού είναι 10% λιγότερο από το μέσο όρο του υπόλοιπου τότε οι συναλλαγές χαρακτηρίζονται ως Minor απειλή [1]:

```
rule highActivity
    when
        $account : Account( )
        $accountInfo : AccountInfo( number == $account.number,
            numberOfTransactions1Day > (averageNumberOfTransactions.
            multiply(BigDecimal.valueOf(5.00))), averageBalance <
            ($account.getBalance().multiply(
            BigDecimal.valueOf(10.00))) )
    then
        insert(new SuspiciousAccount($account.getNumber(),
            SuspiciousAccountSeverity.MINOR));
    End
```

Οι δύο βασικές συνθήκες είναι ο αριθμός των ημερήσιων συναλλαγών να είναι μεγαλύτερος κατά 500% σε σχέση με τον μέσο όρο:

numberOfTransactions1Day > (averageNumberOfTransactions.multiply(BigDecimal.valueOf(5.00)))

και το υπόλοιπο του λογαριασμού έχει μείνει με ποσό μικρότερο κατά 10% από το μέσο όρο:

averageBalance < ($account.getBalance().multiply(BigDecimal.valueOf(10.00))

τότε χαρακτήρισε το λογαριασμό ως ύποπτο:

```
then
insert(new SuspiciousAccount($account.getNumber(),SuspiciousAccountSeverity.MINOR));
```

Έλεγχος του κανόνα - highActivity rule

Το test case για τον έλεγχο λειτουργίας του κανόνα χωρίζεται σε 4 μέρη:

- Το πρώτο ελέγχει ένα μικρό αριθμό συναλλαγών με χαμηλό υπόλοιπο
- Το δεύτερο ελέγχει περιπτώσεις με μικρό αριθμό συναλλαγών
- Το τρίτο ελέγχει περιπτώσεις με χαμηλό υπόλοιπο
- Το τέταρτο ελέγχει τη σωστή εκτέλεση του κανόνα

```
@Test
public void highActivity() throws Exception {
    accountInfoFactType.set(accountInfo,
        "averageNumberOfTransactions",BigDecimal.valueOf(10));
    accountInfoFactType.set(accountInfo,
        "numberOfTransactions1Day", 40l);
    accountInfoFactType.set(accountInfo, "averageBalance",
        BigDecimal.valueOf(9000));
    session.update(accountInfoHandle, accountInfo);
    assertNotFired("highActivity");

    accountInfoFactType.set(accountInfo, "averageBalance",
        BigDecimal.valueOf(11000));
    session.update(accountInfoHandle, accountInfo);
    assertNotFired("highActivity");

    accountInfoFactType.set(accountInfo,
        "numberOfTransactions1Day", 60l);
    accountInfoFactType.set(accountInfo, "averageBalance",
        BigDecimal.valueOf(6000));
    session.update(accountInfoHandle, accountInfo);
    assertNotFired("highActivity");

    accountInfoFactType.set(accountInfo, "averageBalance",
        BigDecimal.valueOf(11000));
    session.update(accountInfoHandle, accountInfo);
    assertFired("highActivity");
}
```

6.3 Μελέτη περίπτωσης για Stock Broker

Μια ακόμη χαρακτηριστική μελέτη περίπτωσης επεξεργασίας συμβάντων είναι η διαχείριση μετοχών η οποία παρουσιάζεται μέσα από το παράδειγμα του Stock Broker του Fusion [3w1]. Και σε αυτή την περίπτωση ο όγκος των δεδομένων είναι πολύ μεγάλος και οι αποφάσεις πρέπει να λαμβάνονται άμεσα. Για το λόγο αυτό η εφαρμογή επεξεργασίας συμβάντων είναι ιδανική. Στη συγκεκριμένη υλοποίηση το πλεονέκτημα είναι ότι τα συμβάντα έχουν μια

καθορισμένη μορφή. Δηλαδή ακολουθούν μια συγκεκριμένη δομή δεδομένων και το fact model είναι εύκολο να προσδιοριστεί. Τα δεδομένα περιέχουν το timestamp, την επωνυμία της μετοχής και την τιμή της. Η μεταβολή των μετοχών δημιουργεί μια ροή συμβάντων. Στη ροή αυτή ενεργούν δύο επιχειρησιακοί κανόνες.

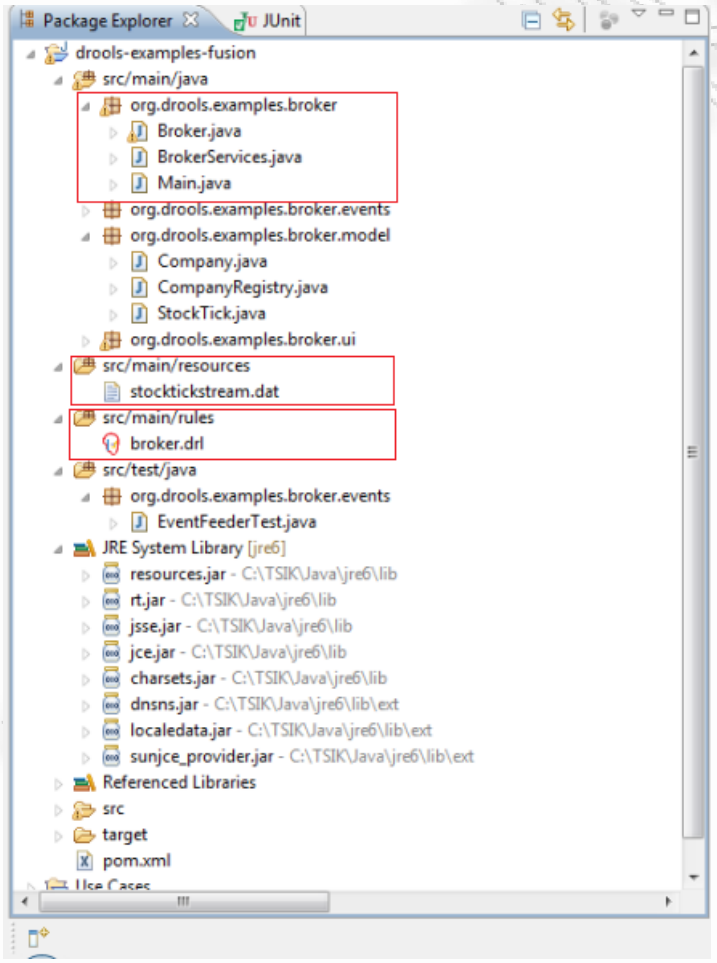
Ο πρώτος κανόνας αφορά στην ανανέωση της τιμής της μετοχής σε σχέση με τη ροή των events και ταυτόχρονα μεταβάλει την τιμή της μετοχής ποσοστιαία.

Ο δεύτερος κανόνας ανιχνεύει πότε η τιμή μιας μετοχής πέφτει κατά 5 cents και τότε ενημερώνει με μήνυμα στο user interface για τις μετοχές που έχουν υποστεί αυτή τη μεταβολή.

Τα βασικά τμήματα που καθορίζουν τη λειτουργία της εφαρμογής είναι:

- Οι κλάσεις:
    - Main
    - Broker
- Το rule file:
    - broker.drl
- Και η γεννήτρια συμβάντων:
    - `stocktickstream.dat`

όπως διακρίνουμε και στην παρακάτω εικόνα:



Η κλάση main

Η main class εκκινεί την εφαρμογή. Δημιουργεί το user interface (UIManager.setLookAndFeel(new Plastic3DLookAndFeel()), διαβάζει τιμές για συγκεκριμένες εταιρείες, δημιουργεί ένα BrokerService και τροφοδοτεί αυτό το BrokerService με events.

```
public class Main {

    public static void main(String[] args) throws Exception {
        // set up and show main window
        UIManager.setLookAndFeel( new Plastic3DLookAndFeel() );
        CompanyRegistry registry = new CompanyRegistry();
        BrokerWindow window = new BrokerWindow(
registry.getCompanies() );
        window.show();

        Broker broker = new Broker( window, registry );

        TimerService clock = new JDKTimerService(1);
        StockTickPersister source = new StockTickPersister();
        source.openForRead( new InputStreamReader(
Main.class.getResourceAsStream( "/stocktickstream.dat" ) ),
                            System.currentTimeMillis() );

        EventFeeder feeder = new EventFeeder(clock, source, broker );
        feeder.feed();

    }
}
```

Η κλάση Broker

Η Broker Class δημιουργεί την Knowledgebase στην οποία «φορτώνει» τους κανόνες και καθορίζει επίσης τον τρόπο λειτουργίας της rule engine. Και σε αυτή την μελέτη περίπτωσης η Knowledgebase λειτουργεί σε STREAM mode:

```
KnowledgeBaseConfiguration conf =
KnowledgeBaseFactory.newKnowledgeBaseConfiguration();
        conf.setOption( EventProcessingOption.STREAM );
```

Στην κλάση αυτή ορίζεται και η μέθοδος receive. Η receive λαμβάνει τα events τα αντιστοιχεί σε stocktick objects και στην συνέχεια αντιστοιχεί αυτά τα stocktick objects στη ροή των events (tick.Stream). Καλεί στην συνέχεια την fireAllRules (this.session.fireAllRules()) και ανάλογα με τους κανόνες που έχουν ενεργοποιηθεί ανανεώνει τις τιμές των μετοχών των εταιρειών στο user interface.

```
    public void receive(Event<?> event) {
        try {
            StockTick tick = ((Event<StockTick>) event).getObject();
            Company company = this.companies.getCompany(
tick.getSymbol() );
            this.tickStream.insert( tick );
            this.session.getAgenda().getAgendaGroup( "evaluation"
).setFocus();
            this.session.fireAllRules();
```

```
            window.updateCompany( company.getSymbol() );
            window.updateTick( tick );
        } catch ( Exception e ) {

System.err.println("================================================
==========");
            System.err.println("Unexpected exception caught:
"+e.getMessage() );
            e.printStackTrace();
        }
    }
```

Το rule file (broker.drl)

Στο rule file ορίζονται τέσσερις κανόνες:

- rule "evaluation done"
- rule "Setup statistics"
- rule "Update stock price"
- rule "sudden drop"

Οι πρώτοι δύο κανόνες ουσιαστικά εφαρμόζουν μορφοποιήσεις στα εισερχόμενα events. Οι πραγματικά επιχειρησιακοί κανόνες που έχουν ενδιαφέρον να αναλύσουμε είναι οι δύο τελευταίοι.

Στο αρχείο αυτό καθορίζεται ότι το StockTick δεν είναι fact άλλα event, όπως είχαμε κάνει και στην προηγούμενη μελέτη περίπτωσης. Μάλιστα για το συγκεκριμένο event καθορίζουμε ότι ο χρόνος ζωής του είναι τα 2 λεπτά (@expires(2m)):

```
declare StockTick
    @role( event )
    @expires( 2m )
end
```

Ο κανόνας - rule "Update stock price"

Με τον κανόνα αυτό γίνεται η ανανέωση των τιμών των μετοχών των εταιρειών ανάλογα με το event stream. Στο τμήμα Then παρατηρούμε μια μέθοδο αλλαγής των δεδομένων ενός event ανανεώνοντας την τιμή με την νέα τιμή (modify($cp) { currentPrice = $pr }) ενώ εμπλουτίζουμε το event υπολογίζοντας μια ποσοστιαία μεταβολή της τρέχουσας τιμής με τη συνάρτηση Δέλτα (modify($st) { delta = $cp.delta }).

```
rule "Update stock price"
    agenda-group "evaluation"
    lock-on-active
when
    $cp : Company( $sb : symbol )
    $st : StockTick( symbol == $sb, $pr : price ) from entry-point
"StockTick stream"
then
    modify( $cp ) { currentPrice = $pr }
    modify( $st ) { delta = $cp.delta }
end
```

Παρατηρούμε επίσης τον καθορισμό του entry point, του σημείου εισόδου δηλαδή των συμβάντων:

```
(from entry-point "StockTick stream)
```

Ο κανόνας - rule "sudden drop"

Με τον κανόνα αυτό ενημερώνουμε το user interface με τις μετοχές αυτές οι οποίες έχουν υποστεί μείωση κατά 5 cents. Συγκρίνουμε την επόμενη τιμή της μετοχής με την προηγούμενη και αν είναι μικρότερη του 0,05 ($sb : symbol, $ts : timestamp, $pr : price, $dt : delta < -0.05), τότε καλείται η globally shared broker service services και δημιουργεί ένα μήνυμα ("Drop >5%: "+$sb+" delta: $"+percent($dt)+" price: $"+$pr), που ενημερώνει ότι η μετοχή έχει υποστεί απότομη πτώση:

```
rule "sudden drop"
when
    $st : StockTick( $sb : symbol, $ts : timestamp, $pr : price, $dt :
delta < -0.05 ) from entry-point "StockTick stream"
    not( StockTick( symbol == $sb, timestamp > $ts ) from entry-point
"StockTick stream" )
then
    services.log( "Drop >5%: "+$sb+" delta: $"+percent($dt)+" price:
$"+$pr );
end
```

Η γεννήτρια συμβάντων

Για τον έλεγχο των κανόνων πρέπει να τροφοδοτήσουμε τη rule engine με events. Η γεννήτρια συμβάντων καθορίζεται από την κλάση EventGenerator. Στην κλάση αυτή ορίζεται η πηγή των συμβάντων:

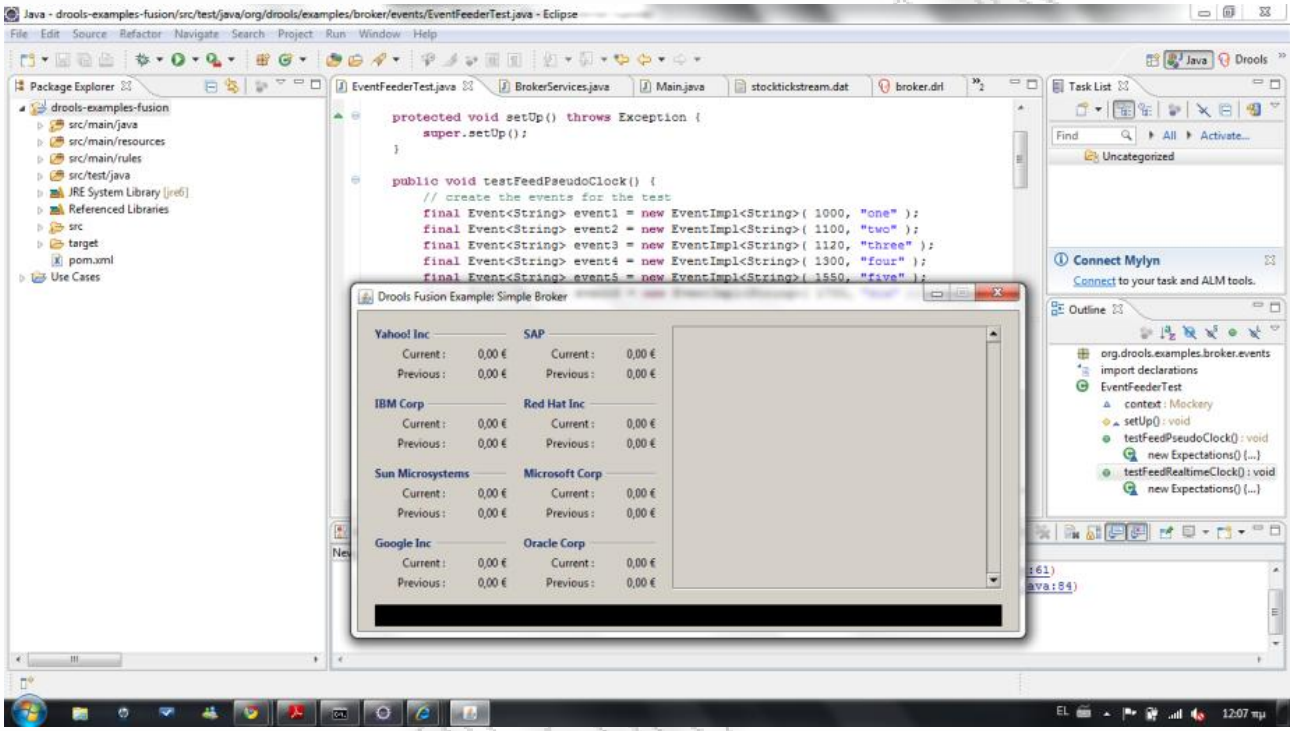
```
public class EventGenerator {
    private static final String DATA_FILE =
"src/main/resources/stocktickstream.dat";
```

που όπως παρατηρούμε είναι ένα .dat αρχείο καθώς επίσης και η μορφή των συμβάντων. Ακολουθεί ένα μικρό δείγμα των τιμών του αρχείου:

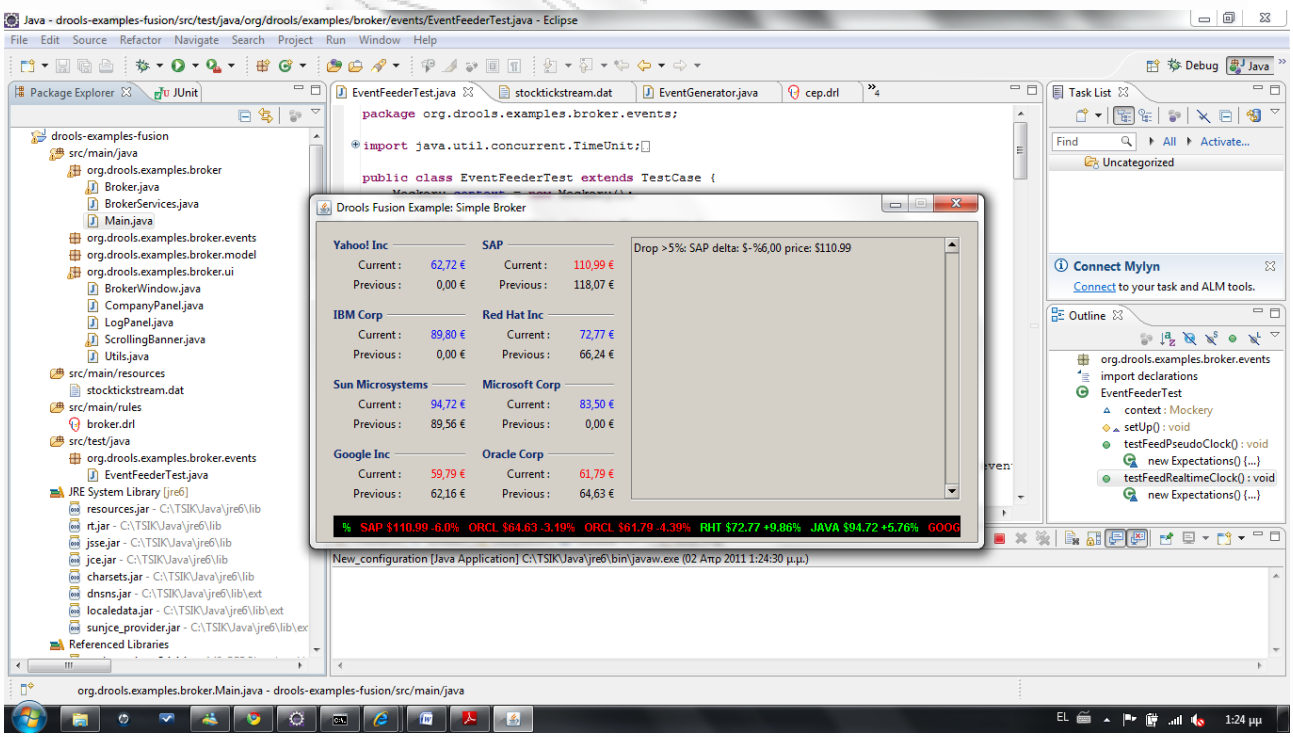
```
0;YHOO;62,72 €
0;SAP;118,07 €
0;IBM;89,80 €
0;RHT;66,24 €
0;JAVA;89,56 €
0;MSFT;83,50 €
0;GOOG;62,16 €
0;ORCL;66,76 €
0;SAP;110,99 €
527;ORCL;64,63 €
2450;ORCL;61,79 €
4400;RHT;72,77 €
5458;JAVA;94,72 €
6681;GOOG;59,79 €
8269;SAP;101,35 €
```

Εκκίνηση της εφαρμογής

Τρέχουμε την κλάση main ως java application και αρχικά εμφανίζεται το user interface με αρχικοποιημένες τις τιμές με την τιμή 0:



Στη συνέχεια αρχίζει η «τροφοδοσία» με events:

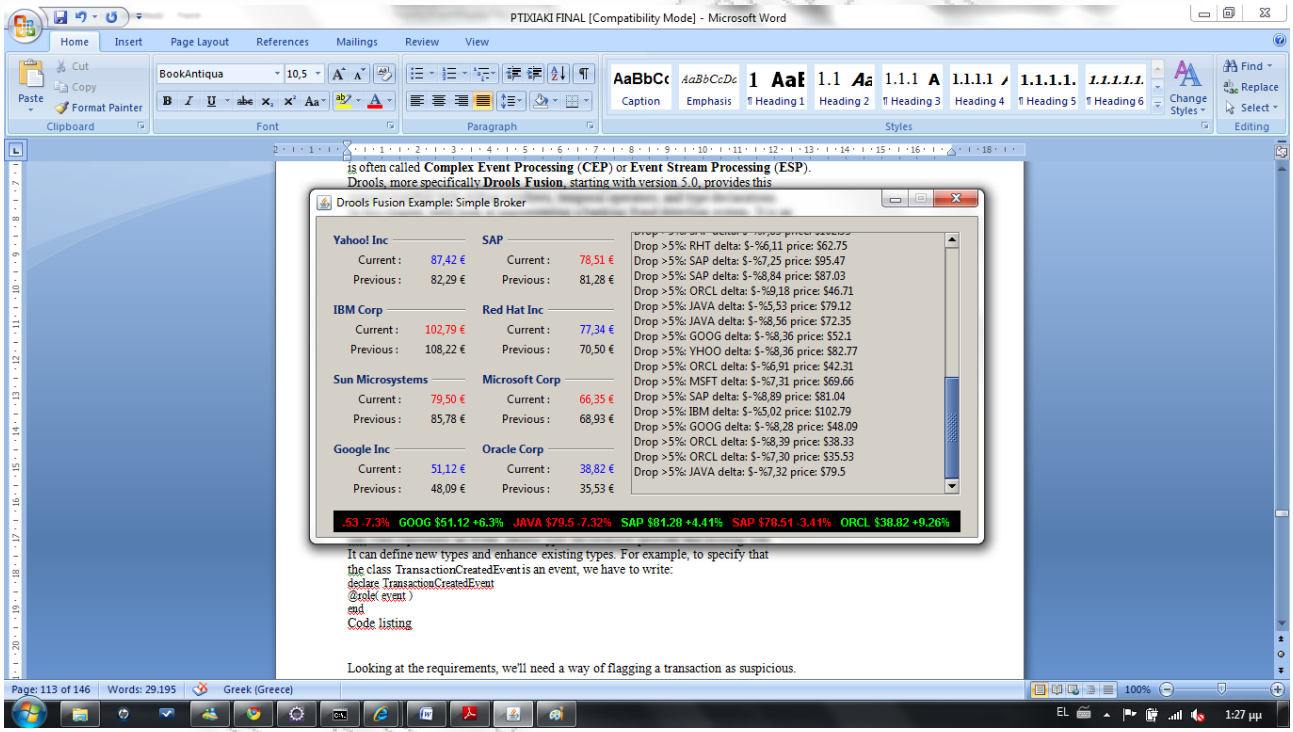


Από την ακολουθία των events ανανεώνονται οι τιμές των μετοχών ανάλογα με το event stockstream και ενεργοποιείται και ο δεύτερος κανόνας (rule "sudden drop") όπου η τιμή της μετοχής της SAP έχει υποστεί μείωση μεγαλύτερη από 5%.

Ένα χαρακτηριστικό του user interface είναι ότι οι τιμές των μετοχών που παρουσιάζουν πτώση είναι με χρώμα κόκκινο:

```
if ( tick.getDelta() < 0 ) {
    g.setColor( Color.red );
} else {
    g.setColor( Color.green );
}
```

Η εφαρμογή εξακολουθεί να τροφοδοτείται με δεδομένα με αποτέλεσμα να παρατηρούμε καινούργια μηνύματα από την απότομη πτώση της τιμής των μετοχών:



7 Επίλογος

Στο κεφάλαιο αυτό θα κάνουμε μια επισκόπηση της διπλωματικής εργασίας. Θα αναφέρουμε τα συμπεράσματα στα οποία καταλήξαμε από την ανάλυση της επεξεργασίας συμβάντων, τις τεχνολογίες που εφαρμόσαμε και τις μελέτες περίπτωσης που διερευνήσαμε. Θα κάνουμε επίσης μια αναφορά στις δυσκολίες που συναντήσαμε κατά την εκπόνηση της εργασίας. Τέλος, θα προτείνουμε βελτιώσεις τόσο όσον αφορά γενικότερα στην επεξεργασία συμβάντων όσο και συγκεκριμένα στην εφαρμογή του Drools.

7.1 Σύνοψη – Συμπεράσματα

Στην εργασία αυτή προσπαθήσαμε να αναλύσουμε την τεχνολογία της επεξεργασίας συμβάντων, η οποία επανέρχεται δυναμικά στο προσκήνιο, προσπαθώντας να δώσει λύσεις αφενός στα προβλήματα διαχείρισης του τεράστιου όγκου πληροφοριών που σωρεύουν οι επιχειρήσεις και αφετέρου στην ταχεία λήψη αποφάσεων. Είδαμε ότι η EDA αρχιτεκτονική διασφαλίζει την επεξεργασία ενός νέφους συμβάντων και την αντιστοίχηση των συμβάντων αυτών με πρότυπα προκειμένου να ενεργοποιηθούν οι κατάλληλοι μηχανισμοί αντιμετώπισης τους.

Διερευνήσαμε επίσης το κατά πόσο μπορεί να συνδυαστεί η EDA αρχιτεκτονική με την αρχιτεκτονική των έμπειρων συστημάτων και πιο συγκεκριμένα με τα συστήματα παραγωγής, προκειμένου να εμπλουτίσουμε τη λειτουργικότητά τους με δυνατότητες επεξεργασίας ECA κανόνων.

Αναλύσαμε διεξοδικά, μέσω της πλατφόρμας ανοικτού κώδικα Drools της Jboss, τεχνολογίες που προσφέρονται σήμερα για την επεξεργασία συμβάντων. Επικεντρωθήκαμε στην αρχιτεκτονική της και στους τρόπους που προσφέρει στη συγγραφή κανόνων. Ιδιαίτερα ως προς το τελευταίο, είδαμε ότι παρέχει δυνατότητες εμπλουτισμού της βάσης γνώσης με κανόνες που εκφράζονται ακόμα και με φυσική γλώσσα (.dsl) την οποία μπορεί να χρησιμοποιήσει ο οποιοδήποτε ειδικός ενός τομέα γνώσης χωρίς τη διαμεσολάβηση προγραμματιστών.

Τέλος αναλύσαμε δύο χαρακτηριστικές μελέτες περίπτωσης που εφαρμόζουν μέσω του Drools Fusion την επεξεργασία συμβάντων. Παραθέσαμε κώδικα που υλοποιεί την επεξεργασία συμβάντων, όπως τη δήλωση ενός object ως event, τον καθορισμό entry-point για την τροφοδοσία της μηχανής κανόνων με event streams και τη διαμόρφωση λειτουργίας της Knowledgebase σε Stream ή Cloud mode. Μέσα από την παρουσίαση του Stock Broker demo του fusion, είδαμε πώς ένα stream από τιμές μετοχών μπορεί να ενεργοποιήσει κανόνες που ενημερώνουν τον χρηματιστή για την διακύμανση συγκεκριμένων μετοχών.

Σήμερα οι επιχειρήσεις καλούνται να αναπτύξουν μεθοδολογίες και τεχνολογίες που θα τους επιτρέψουν να διαχειριστούν όλο το πλήθος των δεδομένων που διακινούνται στις επιχειρησιακές διαδικασίες και προέρχονται από πολλές πηγές από όλο φάσμα των δραστηριοτήτων τους. Με την παρούσα διπλωματική εργασία δείξαμε ότι η επεξεργασία συμβάντων αποτελεί μια αποτελεσματική απάντηση σε αυτή την αναγκαιότητα και ότι ακόμα και συστήματα ανοικτού κώδικα, όπως το Drools, μπορούν να ανταπεξέλθουν σε αυτή την αποστολή.

7.2 Δυσκολίες

Τα προβλήματα που παρουσιάστηκαν κατά την εκπόνηση της εργασίας αυτής αντιμετωπίστηκαν περισσότερο ως προκλήσεις.

Καταρχήν για την επεξεργασία συμβάντων, δεν έχει ακόμα αναπτυχθεί ικανοποιητική βιβλιογραφία. Μάλιστα, αντιμετωπίσαμε δυσκολία και στην εύρεση αντίστοιχης ελληνικής ορολογίας. Υπάρχουν αρκετές βασικές έννοιες όπως αυτές του συμβάντος (event) και των δεδομένων (facts) που οι διαφορές τους δεν έχουν ακόμα αποσαφηνιστεί πλήρως.

Ένα δεύτερο πρόβλημα είναι ότι σε καθαρά τεχνολογικό επίπεδο ελάχιστες εταιρείες προσφέρουν εμπορικά προϊόντα για την επεξεργασία συμβάντων διότι είναι ένας τομέας που, αν και αναπτύσσεται ταχύτητα, μόλις τα τελευταία χρόνια έχει αυτονομηθεί προκειμένου να μιλάμε για CEP συστήματα. Ήταν λοιπόν δύσκολο να διερευνήσουμε την λειτουργικότητα που πρέπει να υποστηρίζεται καθώς επίσης και την εύρεση λύσεων που έχουν υλοποιηθεί για την επιλογή των μελετών περίπτωσης.

Αρκετός χρόνος δαπανήθηκε επίσης στην εξεύρεση του κατάλληλου λογισμικού. Δυστυχώς δεν υπάρχουν πολλές επιλογές ανοικτού κώδικα για την επεξεργασία συμβάντων ή κάποια freeware έκδοση. Το Fusion του Drools που τελικά επιλέχθηκε αποτελεί ακόμα και σήμερα ένα "work in progress" και δεν έχει τα χαρακτηριστικά ενός «ώριμου» εργαλείου για event processing. Αυτό είχε σαν αποτέλεσμα να συναντήσουμε δυσκολίες ακόμα και κατά τη φάση της εγκατάστασης του λογισμικού, ενώ το documentation αν και μπορεί να χαρακτηριστεί ως επαρκές, σε συγκεκριμένα προβλήματα ήταν δύσκολο να βρεθεί μια άμεση απάντηση.

7.3 Περαιτέρω Ανάπτυξη

Στην παράγραφο αυτή θα προτείνουμε πιθανές βελτιώσεις και επεκτάσεις τόσο όσον αφορά στο Drools Fusion όσο και γενικότερα στην επεξεργασία συμβάντων.

Βελτιώσεις στις μελέτες περίπτωσης του Drools Fusion

Όσον αφορά στις μελέτες περίπτωσης καταρχήν μπορούν αν εμπλουτιστούν με περισσότερους κανόνες προκείμενου να επεκτείνουν τη λειτουργικότητά τους σε ένα μεγαλύτερο φάσμα δεδομένων αλλά και στην εκτέλεση περισσότερων ενεργειών. Οι κανόνες αυτοί μπορεί να είναι σε μορφή φυσικής γλώσσας, εφόσον οριστεί κάποιο .dsl, προκειμένου να παρουσιαστεί και αυτή η δυνατότητα του Drools όπου ο εμπλουτισμός των μελετών περίπτωσης με νέους κανόνες να μην απαιτεί τη διαμεσολάβηση προγραμματιστή αλλά του ειδικού του συγκεκριμένου πεδίου όπου εφαρμόζεται η μελέτη περίπτωσης.

Προκειμένου να υποστηριχθεί ένα μεγάλο πλήθος κανόνων αλλά και ένας μεγάλος όγκος δεδομένων, η μηχανή κανόνων θα πρέπει να διασυνδεθεί με μια εξωτερική βάση δεδομένων. Μια προσέγγιση για αυτό το ζήτημα θα μπορούσε να είμαι μια MySQL βάση μέσω Hibernate.

Μια επίσης ιδέα που κυριαρχεί γενικότερα όσον αφορά στην EDA Αρχιτεκτονική είναι η ολοκλήρωση με SOA. Έτσι, η τροφοδοσία των entry-points με events θα μπορούσε να γίνει μέσω web service. Ιδιαίτερα στο παράδειγμα του Stock Broker, ένα web service θα μπορούσε να ενημερώνει την εφαρμογή με τις πραγματικές τιμές των μετοχών και στη συνέχεια να ενεργοποιούνται οι κατάλληλοι κανόνες. Και οι ενέργειες μπορεί να είναι ένα web service το οποίο να προκαλεί πχ. την αγορά ή πώληση μετοχών.

Εξέλιξη της Επεξεργασίας Συμβάντων

Τέλος, θεωρούμε σκόπιμη μια σύντομη αναφορά στις τελευταίες εξελίξεις που αφορούν στην επεξεργασία συμβάντων και ιδιαίτερα την ολοκλήρωση της EDA αρχιτεκτονικής με την SOA (Service Oriented Architecture) Αρχιτεκτονική.

- Η SOA Αρχιτεκτονική αποτελείται από τρία βασικά στοιχεία [3]:
- Υπηρεσίες (Services)
- Επιχειρησιακό Κανάλι Υπηρεσιών (Enterprise Service Bus), για την επικοινωνία μεταξύ πολλαπλών συστημάτων που χρησιμοποιούν διαφορετικές και ετερογενείς τεχνολογίες.
- Χαλαρή σύζευξη (loose-coupling), μεταξύ των υποσυστημάτων της Αρχιτεκτονικής.

Από τα παραπάνω διακρίνουμε κοινά χαρακτηριστικά μεταξύ της EDA και SOA Αρχιτεκτονικής. Η βασική διαφορά είναι ότι στα τμήματα μιας EDA αρχιτεκτονικής μπορεί να μην γνωρίζει το ένα την ύπαρξη του άλλου, εφόσον η επικοινωνία μεταξύ τους πραγματοποιείται μέσω events τα οποία μεταδίδονται μέσω ενός βασικού καναλιού. Η αλληλεπίδραση μεταξύ υπηρεσιών (services) και συμβάντων (events) χαρακτηρίζεται ως SOA καθοδηγούμενη από συμβάντα (Event Driven SOA – EDSOA [4]. Στην Αρχιτεκτονική αυτή η ύπαρξη κάποιου συμβάντος μπορεί να προκαλέσει την εκκίνηση μίας ή περισσοτέρων υπηρεσιών. Οι υπηρεσίες αυτές μπορεί να εκτελούν από απλές λειτουργίες όπως την αποστολή ενός email έως και πιο πολύπλοκες όπως την εκκίνηση κάποιας επιχειρησιακής διαδικασίας ή την παραγωγή νέων

συμβάντων. Στα πλαίσια της σύγχρονης επιχειρησιακής ανάγκης για ευελιξία, αυτονομία και ταχύτητα στη λήψη αποφάσεων, η παραπάνω προσέγγιση διασφαλίζει βασικές απαιτήσεις όπως την συντηρισιμότητα, την επεκτασιμότητα του λογισμικού καθώς και την επικοινωνία μεταξύ ετερογενών συστημάτων.

Βιβλιογραφία - Αναφορές

- **Βιβλιογραφικές Αναφορές**

[1] Drools JBoss Rules 5.0 Developer's Guide, Packt Publishing

[2] JBoss Enterprise BRMS Platform 5.0 JBoss Rules 5 Reference Guide, Edition 1. Mark Proctor, Michael Neale, Edson Tirelli, Red Hat, Inc

[3] Mike P. Papazoglou, Willem-Jan van den Heuvel (2007), "*Service oriented architectures: approaches, technologies and research issues"*, The VLDB Journal

[4] Seeley, R. (2006), "*Oracle´s Debnath on making an Event-Driven SOA"*

[5] Mark Proctor, Michael Neale, Michael Frandsen, Sam Griffith Jr., Edson Tirelli, Fernando Meyer, Kris Verlaenen, "*Drools Documentation vv4.0.4"*

[6] Q. H. Mahmoud (2005), "*Getting Started With the Java Rule Engine API (JSR 94): Toward Rule-Based Applications"*, SDN Technical Articles and Tips

[7] Charles L. Forgy (1979), "*On the efficient implementation of production systems"*, PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA

[8] Charles L. Forgy. (1982), "*Rete: a fast algorithm for the many pattern/many object pattern match problem*", Artificial Intelligence, v.19, pp 17–37

[9] Chakravarty, Singh, (2008), "*Incorporating Events into Cross-Organizational Business Processes",* IEEE Internet Computing

[10] Jürgen Dunkel, Alberto Fernández, Rubén Ortiz, Sascha Ossowski (2008), "*Event-Driven Architecture for Decision Support in Traffic Management Systems",* Proceedings of the 11th International IEEE, Conference on Intelligent Transportation Systems, Beijing, China, October 12-15

[11] Jeroen van Bemmel, Partricia Dockhorn and Ing Widya (2004), "*Paradigm: Event-driven Computing"*, Lucent Technologies, CTIT

[12] Ι. Βλαχάβας - Π. Κεφάλας - Ν. Βασιλειάδης - Φ. Κόκκορας – Η. Σακελλαρίου (2002), "*Τεχνητή Νοημοσύνη",* Εκδόσεις Γαργατάνη

[13] Luckham, D. (2002), "*The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems"*, Addison-Wesley Professional, 1st edition

[14] Luckham, D. (2005), "*Why we need a new technology to manage Event Driven Systems"*,

[15] Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides (1995), "*Design Patterns: Elements of Reusable Object-Oriented Software"*, Addison-Wesley Professional

[16] Barros A., Decker G., Grosskopf A. (2007), "*Complex Events in Business Processes, Business Information Systems"*, Springer

[17] Decker G., Grosskopf A., Barros A. (2007), "*A Graphical Notation for Modeling Complex Events in Business Processes"*, IEEE computer society

[18] Korherr, B. and List, B. (2006), "*A UML 2 Profile for Event Driven Process Chains"*, Proceedings of the 1st IFIP International Conference on Research and Practical Issues of Enterprise Information Systems, Springer Verlag

[19] Jonas Rommelspacher (2008), "*Modelling Complex Events with Event-Driven Process Chains"*, SIGSAND-EUROPE

[20] Σ.Τζαφέστας (1988), *Εισαγωγή στην Τεχνητή Νοημοσύνη και τα Έμπειρα Συστήματα*

[21] Steinberg, A.N., Bowman, C.L., and White, F.E. (1998), *"Revisions to the JDL Model"*, Joint NATO/IRIS Conference Proceedings, Quebec,

[22] Thomas J. Owens (2007), *"Survey of Event Processing",* AIR FORCE RESEARCH LAB ROME NY INFORMATION DIRECTORATE

[23] Dave L. Hall and James Llinas (1997), “Introduction to Multisensor Data Fusion”, Proc. of IEEE , Vol. 85

[24] Erik Blasch, Ivan Kadar, John Salerno, Mieczyslaw Kokar, Subrata Dase, Gerald Powell, Daniel Corkill, and E. Euspini (2006), “Issues and Challenges in Situation Assessment (Level 2 Fusion)”, Journal of Advances in Information Fusion, Vol 1, No 2

[25] Brenda M. Michelson (2006), “*Event-Driven SOA Is Just Part of the EDA Story*”, Patricia Seybold Group

[26] M. Minsky (1975), “*A Framework for Representing Knowledge. The Psychology of Computer Vision”*, P. H. Winston (ed.), McGraw-Hill

[27] Newell, A., & Simon, H. A. (1972), “*Human problem solving*”. Englewood Cliffs, NJ: Prentice-Hall

[28] McCarthy, John, Marvin Minsky, Nathan Rochester, and Claude Shannon, "*A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence*." Formal Reasoning Group Stanford University, 31 Aug. 1955.

[29] G. Luger, W. Stubblefield “Artificial Intelligence. Structures and Strategies for Complex Problem Solving”, Addison-Wesley, 1998

[30] Ramesh Jain, *«EventWeb: Developing a Human-Centered Computing System»*, February 2008 (vol. 41 no. 2)

[31] David C. Luckham and Benoit A. Gennart. *«Event patterns: a language construct for hierarchical design of concurrent systems»*. Technical Report: CSL-TR-90-453, 1990

[32] Matias Alvarado, Marisol Vázquez., *«Decision Making automation based on fuzzy event-condition-action rules»*

[33] Y.H. Zhang, Q.Y. Dai, and R.Y. Zhong, *«An Extensible Event-Driven Manufacturing Management with ComplexEvent Processing Approach»,* International Journal of Control and Automation. Vol.2, No.3, September 2009

[34] Papamarkos, G., Poulovassilis, A., & Wood, P., *«RDFTL: An event-condition-action language for RDF»*. Web Dynamics Workshop, at WWW'2004.

- **Διαδικτυακές Αναφορές**

[3w1] http://www.jboss.org/drools/downloads.html

[3w2] http://www.jboss.org/drools/drools-fusion

[3w3] http://firstpartners.net/whitepapers/Drools-Business-Rules-Management-System-BRMS-Guide.pdf

[3w4] http://blog.athico.com/2010/06/ai-research-overview.html

[3w5] http://en.wikipedia.org/wiki/Rete_algorithm

[3w6]http://www.jbossworld.com/2008/downloads/pdf/thursday/New_Introduction_to_JBoss_Drools_and_The_Business_Rules_Management_System_Mark_Proctor_JBoss.pdf

[3w7] http://www.jboss.org/drools

[3w8] http://www.jboss.org/jbossas

[3w9] http://en.wikipedia.org/wiki/Backward_chaining

[3w10] http://en.wikipedia.org/wiki/Forward_chaining

[3w11] http://www.computing.surrey.ac.uk/ai/PROFILE/mycin.html

[3w12] http://en.wikipedia.org/wiki/Complex_event_processing

[3w13] http://en.wikipedia.org/wiki/Pattern_matching

[3w14] http://www.jcp.org/en/jsr/detail?id=94

[3w15] http://www.javarules.org/api_doc/api/index.html

[3w16] http://java.sun.com/developer/technicalArticles/J2SE/JavaRule.html

[3w17] http://complexevents.com/
[3w18] http://en.wikipedia.org/wiki/Event_stream_processing
[3w19] http://en.wikipedia.org/wiki/Data_Stream_Management_System
[3w20] http://en.wikipedia.org/wiki/Event-driven_architecture
[3w21]http://www.chris-kimble.com/Courses/World_Med_MBA/Types-of-Information-System.html
[3w22] http://www.duke.edu/~mccann/mwb/15semnet.htm
[3w23] http://en.wikipedia.org/wiki/Drools
[3w24] http://blog.athico.com/
[3w25] http://www.jboss.com/products/rules
[3w26] http://www.onjava.com/pub/a/onjava/2005/08/03/drools.html
[3w27] http://blog.athico.com/2007/12/drools-solver-javapolis-2007-slides.html
[3w28] http://adcalves.wordpress.com/
[3w29] http://osdir.com/ml/java.drools.user/2006-02/msg00008.html
[3w30] http://www.jboss.com/pdf/JBossBRMSDataSheet.pdf
[3w31] http://www.bcs-sges.org/

Παράρτημα

Ανάλυση Έμπειρων Συστημάτων

Διαχωρισμός των Ε.Σ βάσει της Αναπαράστασης Γνώσης

Ένα Έμπειρο Σύστημα, διακρίνεται από την ευφυή συμπεριφορά του. Κάθε τέτοιο σύστημα περιλαμβάνει δύο δομικά στοιχεία που είναι η Βάση Γνώσης και ο Μηχανισμός Εξαγωγής Συμπερασμάτων όπως αναλύσαμε και στην παράγραφο για την Αρχιτεκτονική των συστημάτων αυτών. Η αναπαράσταση γνώσης αφορά στο τμήμα της Βάσης Γνώσης εφόσον και ουσιαστικά αποτελεί τον τρόπο βάσει του οποίου θα αποθηκεύσουμε τη γνώση. Η γνώση δεν προγραμματίζεται αυστηρά, αλλά περιγράφεται ρητά στη Βάση Γνώσης με τη βοήθεια κάποιας τυπικής γλώσσας που ονομάζεται γλώσσα αναπαράστασης της γνώσης (knowledge representation language). Δηλαδή η Βάση Γνώσης αποτελείται από ένα σύνολο εκφράσεων που περιγράφουν την ενσωματωμένη στο σύστημα γνώση. Ο Μηχανισμός Εξαγωγής Συμπερασμάτων επεξεργάζεται το σύνολο αυτών των εκφράσεων και παράγει ευφυή συμπεριφορά. Αυτή ουσιαστικά είναι και η αρχή λειτουργίας των έμπειρων συστημάτων. Υπάρχουν τεχνικές αναπαράστασης, όπως τα πλαίσια, οι κανόνες παραγωγής και τα σημασιολογικά δίκτυα που έχουν προέλθει από θεωρίες που αφορούν τον τρόπο διαχείρισης της γνώσης από τον άνθρωπο. Για να είναι χρήσιμη μια γλώσσα αναπαράστασης θα πρέπει να είναι επαρκής, δηλαδή να μπορεί να ανταποκριθεί στις ανάγκες της αναπαράστασης της γνώσης των εφαρμογών που χρησιμοποιείται. Τα τρία θεμελιώδη κριτήρια αξιολόγησης της επάρκειας μιας γλώσσας αναπαράστασης είναι η εκφραστικότητα, η αποδοτικότητα και η φυσικότητα. Οι σημαντικότερες από τις τεχνικές αυτές αναλύονται στη συνέχεια.

Σημασιολογικά Δίκτυα (Semantic Networks)

Τα Σημασιολογικά Δίκτυα είναι σχήματα που αναπαριστούν γνώση. Το νόημα και η σημασία ενός σημασιολογικού κειμένου εκπορεύεται από την δομή του. Τα Σημασιολογικά Δίκτυα αποτελούνται από κόμβους (nodes) και δεσμούς (links) ανάμεσά τους. Οι κόμβοι υποδηλώνουν κλάσεις αντικειμένων (classes), αντικείμενα (objects), έννοιες (concepts) και τιμές ιδιοτήτων (values). Οι δεσμοί υποδηλώνουν σχέσεις (relations) μεταξύ αντικειμένων ή ιδιότητες που συνδέουν αντικείμενα με τιμές. Οι κυριότεροι τύποι δεσμών είναι:

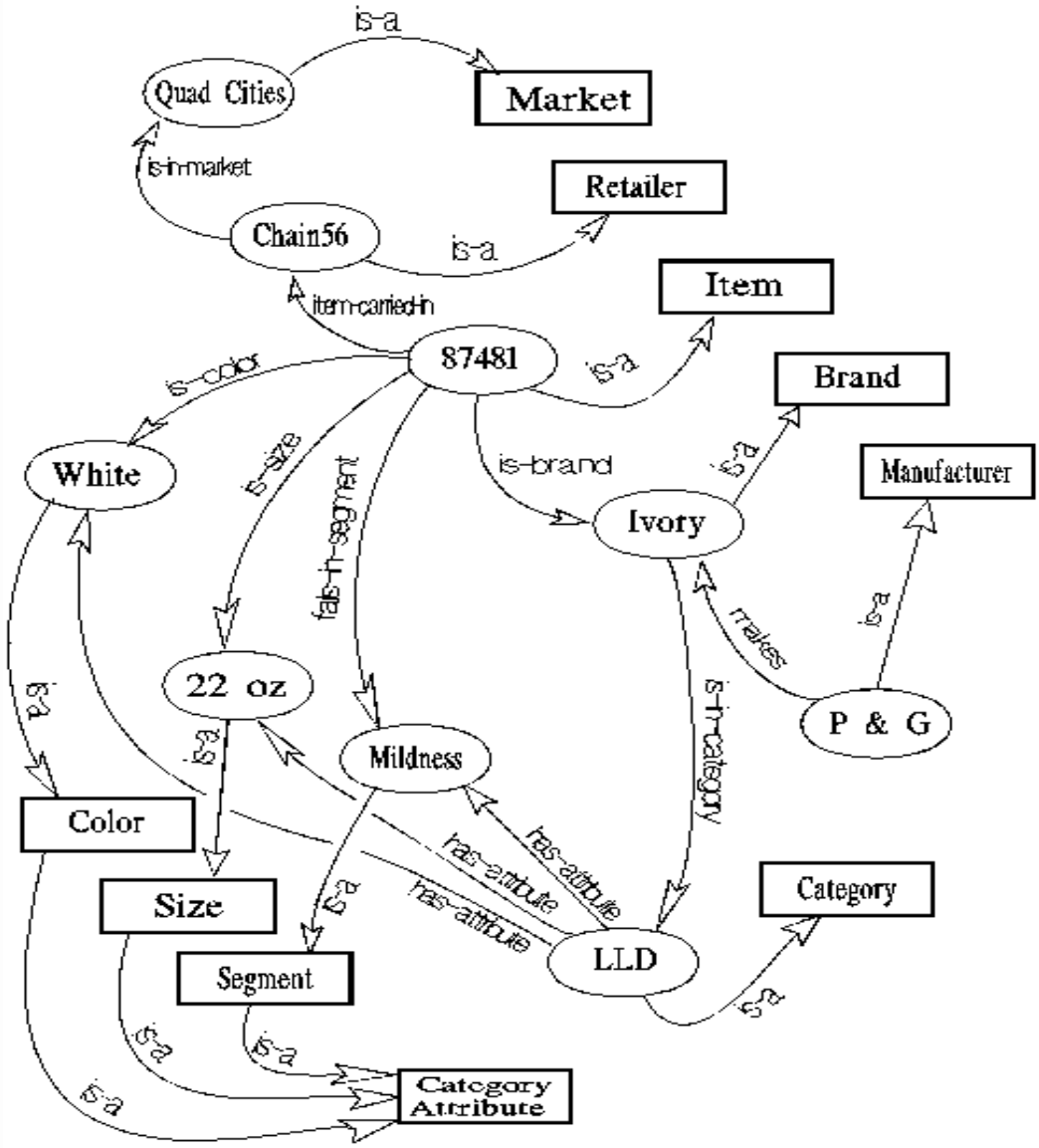
- isa: ο σύνδεσμος σχετίζει ένα αντικείμενο με την γενικότερη κλάση που ανήκει
- ako: ο σύνδεσμος σχετίζει μια κλάση με μία υπερκλάση
- connects: ο σύνδεσμος αποκαλύπτει μία συμμετρική σχέση μεταξύ των κόμβων

Οι σύνδεσμοι isa και ako επιτρέπουν την εξαγωγή συμπερασμάτων και την κληρονομικότητα.

Ένα αντικείμενο κληρονομεί ιδιότητες από μία υψηλότερη ιεραρχικά κλάση. Οι βασικότεροι τύποι σημασιολογικών δικτύων είναι οι εξής:

- Δίκτυα ορισμών (definitional networks)
- Δίκτυα ισχυρισμών (assertional networks)
- Δίκτυα συνεπαγωγής (implicational networks)
- Εκτελέσιμα δίκτυα (executable networks)
- Μαθησιακά δίκτυα (learning networks)
- Υβριδικά δίκτυα (hybrid networks)

Ένα χαρακτηριστικό Σημασιολογικό Δίκτυο περιγράφεται στην εικόνα που ακολουθεί [3w22]:



Σχήμα 36: Σημασιολογικό Δίκτυο

Ένα από τα σημαντικότερα πλεονεκτήματα των Σημασιολογικών Δικτύων, ως μηχανισμοί αναπαράστασης γνώσης, είναι καταρχήν το γεγονός ότι παρέχουν γραφικά και με συμπιεσμένο τρόπο μια πολύ απλή αναπαράσταση. Παρέχουν επίσης έναν απλό μηχανισμό εξαγωγής συμπερασμάτων και προσεγγίζουν το σύνθετο τρόπο που είναι οργανωμένη η ανθρώπινη σκέψη. Στα μειονεκτήματα θα μπορούσαμε να αναφέρουμε ότι για κάθε κατάσταση υπάρχουν άπειροι διαφορετικοί τρόποι για να αναπαρασταθούν. Η μέθοδος εξαγωγής συμπερασμάτων για ένα σημασιολογικό δίκτυο πρέπει να είναι ξεχωριστή για κάθε ιδιαίτερο τύπο συνδέσμου σε αντίθεση με τις αναπαραστάσεις που βασίζονται στην λογική. Είναι επίσης αμφίβολη και η αποδοτικότητά τους γιατί για την απάντηση ερωτημάτων και κυρίως αυτά που επιζητούν αρνητικές απαντήσεις, είναι συχνά απαραίτητο να γίνει αναζήτηση στο μεγαλύτερο μέρος του σημασιολογικού δικτύου.

Πλαίσια (Frames)

Ο Minsky το 1975 [26] πρότεινε έναν πρωτοποριακό τρόπο αναπαράστασης της γνώσης χρησιμοποιώντας σαν βασικά στοιχεία τα πλαίσια (frames). Την πρόταση αυτή την εμπνεύστηκε από το γεγονός ότι το ανθρώπινο μυαλό φαίνεται ότι οργανώνει τη γνώση σε μεγάλα κομμάτια

τα οποία παρουσιάζουν ορισμένη εσωτερική δομή. Ως πλαίσιο ορίζεται μια δομή δεδομένων η οποία αναπαριστά μια στερεότυπη κατάσταση. Μια σημαντική παρατήρηση του Minsky στο ίδιο άρθρο είναι ότι οι δηλωτικές και διαδικαστικές όψεις ενός κομματιού γνώσης πρέπει να είναι πιο στενά συνδεμένες σε μια αναπαράσταση. Αν και απαιτούν επιδεξιότητα και επίπονη εργασία, εξελίχθηκαν σε έναν σημαντικό τρόπο αναπαράστασης γνώσης.

Τα πλαίσια χαρακτηρίζονται από το όνομά τους, τις σχισμές τους (slots) που συνδέονται άμεσα με τις τιμές τους (fillers). Επιπλέον, οι σχισμές μπορεί να περιλαμβάνουν και επισυναπτόμενες μεθόδους, οι οποίες είναι γενικά τριών τύπων. Οι μέθοδοι if-needed εκτελούνται όταν χρειάζεται μία τιμή πλήρωσης η οποία δεν υπάρχει, ή η προκαθορισμένη τιμή δεν μπορεί να εφαρμοστεί. Οι μέθοδοι if-added εκτελούνται όταν πρόκειται να προστεθεί μία τιμή πλήρωσης σε μία σχισμή, ενώ οι μέθοδοι if-removal εκτελούνται όταν πρόκειται να αφαιρεθεί μία τιμή από μία σχισμή.

Δεδομένου ότι έχουμε ήδη αναφέρει τα Σημασιολογικά Δίκτυα αξίζει να αναφέρουμε ότι τα πλαίσια μπορούν να αποτελέσουν αντικείμενα-κόμβους ενός τέτοιου δικτύου και να συνδεθούν με μία ιεραρχία. Επίσης τα πλαίσια διαθέτουν κληρονομικότητα, όπως και τα σημασιολογικά δίκτυα, αλλά πλεονεκτούν σε σχέση με αυτά στο ότι η ιεραρχία τους είναι πιο ξεκάθαρη και επιπροσθέτως ένα πλαίσιο περιέχει όλη την πληροφορία για τη συγκεκριμένη έννοια που αναπαριστά ενώ ένας κόμβος σε ένα σημασιολογικό δίκτυο αναπαριστά μόνο την έννοια, ενώ οι ιδιότητές της περιγράφονται σε άλλους κόμβους που συνδέονται με αυτόν.

Αντικείμενα (Objects)

Τα Αντικείμενα αποτελούν μια μέθοδο αναπαράστασης γνώσης εμπνευσμένη από την έρευνα για γλώσσες προσομοίωσης. Οι μέχρι τώρα μεθοδολογίες παρουσιάζουν πολυπλοκότητα είτε όσον αφορά στην υλοποίηση τους είτε όσον αφορά στον τρόπο περιγραφής των διαφόρων εννοιών. Την ανάγκη δημιουργίας εύκολα αντιληπτών μοντέλων ήρθαν να καλύψουν τα αντικείμενα. Η κεντρική ιδέα είναι ότι οι έννοιες, οντότητες, στον φυσικό κόσμο μπορούν να αποδομηθούν (decomposed) σε αντικείμενα.

Το κάθε αντικείμενο χαρακτηρίζεται από την κατάστασή του, τις μεθόδους και τα συμβάντα (events). Οι μέθοδοι καθορίζουν τη συμπεριφορά του αντικειμένου, το οποίο αντιδρά σε συγκεκριμένα γεγονότα που παράγονται από το περιβάλλον. Τα αντικείμενα θυμίζουν τον ορισμό των αντικειμένων που συναντάμε στον αντικειμενοστραφή προγραμματισμό. Έτσι τα αντικείμενα οργανώνονται σε κλάσεις, σύμφωνα με τις ιδιότητες τις μεθόδους και τα συμβάντα στα οποία αντιδρά. Υποστηρίζεται επίσης η κληρονομικότητα και ο πολυμορφισμός, ενώ παρέχεται και η δυνατότητα διαβαθμισμένης απόκρυψης της εσωτερικής πληροφορίας και της εσωτερικής πολυπλοκότητας.

Διαφορές Αντικειμένων – Πλαισίων

Για την καλύτερη κατανόηση των Αντικειμένων ως μέθοδο αναπαράστασης γνώσης θα τα συγκρίνουμε με μία άλλη μέθοδο που έχουμε ήδη αναφέρει και είναι αυτή των πλαισίων. Στις ομοιότητες καταρχήν μπορούμε να αναφέρουμε τη δομημένη περιγραφή, την κληρονομικότητα, τις ιδιότητες και τις μεθόδους. Στον πίνακα που ακολουθεί αντιστοιχίζονται οι βασικές διαφορές τους:

Αντικείμενα	Πλαίσια
Οι κλάσεις λειτουργούν ως τύποι δεδομένων	Οι κλάσεις έχουν το ρόλο προτύπου
Η αυστηρά καθορισμένη δομή των αντικειμένων, διευκολύνει τον έλεγχο ορθότητας των προγραμμάτων και επιτρέπει την παραγωγή αποδοτικότερου κώδικα	Τα στιγμιότυπα στις κλάσεις δεν είναι υποχρεωτικό να ακολουθήσουν αυστηρά τις προδιαγραφές που ορίζει η κλάση τους
Η πρόσβαση στις ιδιότητες και στις μεθόδους	Οι τιμές των ιδιοτήτων (slots) είναι πάντα

είναι διαβαθμισμένη	προσβάσιμες
Τα αντικείμενα εμπεριέχουν τον κώδικα ελέγχου μέσα τους με τη μορφή μεθόδων	Στα πλαίσια ο υπόλοιπος κώδικας είναι αποθηκευμένος εκτός των πλαισίων
Οι μέθοδοι ενεργοποιούνται με την εκούσια αποστολή μηνυμάτων από τους χρήστες ή από μεθόδους άλλων αντικειμένων	Οι δαίμονες ενεργοποιούνται αυτόματα όταν γίνει πρόσβαση στις ιδιότητες
Υπάρχουν ενσωματωμένα ολόκληρα προγράμματα υπό τη μορφή μεθόδων.	Προσκολλώνται μικρές διαδικασίες (δαίμονες) σε κάποιες ιδιότητες

Πίνακας 4: Βασικές Διαφορές Μεταξύ Πλαισίων και Αντικειμένων

Κανόνες Παραγωγής (Production Rules)

Στη δεκαετία του ΄70, οι A. Newell και H.A. Simon [27] προσπάθησαν να δημιουργήσουν ένα σύστημα επίλυσης γενικών προβλημάτων και κατέληξαν στο GPS (General Problem Solver). Αυτό το σύστημα εισήγαγε την έννοια του συστήματος κανόνων παραγωγής. Η ιδέα αυτού του συστήματος βασίστηκε στη μελέτη των δύο ερευνητών πάνω στην ανθρώπινη συλλογιστική. Οι αισθήσεις αντιλαμβάνονται ερεθίσματα τα οποία μεταδίδουν στον εγκέφαλο, ο οποίος ανάλογα με τα ερεθίσματα αυτά ενεργοποιεί τους κατάλληλους κανόνες μνήμης που τελικά οδηγούν στην ανάλογη αντίδραση του ατόμου. Σύμφωνα και με έρευνες στο πεδίο της Γνωστικής Ψυχολογίας η ανθρώπινη μνήμη οργανώνεται σε μικρές συλλογές γνώσεων. Ο κάθε κανόνας λοιπόν θα μπορούσε να εκφράζει μια τέτοια μικρή συλλογή μνήμης.

Οι κανόνες αυτοί είναι της μορφής IF…THEN, γιατί όπως προαναφέραμε η διαδικασία συλλογισμού διακρίνεται από δύο τμήματα, το τμήμα της μνήμης και το τμήμα της αντίδρασης. Έτσι και οι κανόνες παραγωγής έχουν την εξής γενική μορφή:

```
Rule: <Rule Name>
IF
   <LHS>
THEN
   <RHS>
```

Εξετάζοντας την παραπάνω μορφή παρατηρούμε ότι κάθε κανόνας χαρακτηρίζεται μοναδικά από ένα όνομα (Rule Name) και στη συνέχεια ακολουθεί το IF…THEN μέρος του κανόνα. Το εάν-τμήμα (IF-Part) ή τμήμα συνθήκη (Condition-part) ή αριστερό τμήμα (Left-Hand Side) και το τότε-τμήμα ή τμήμα-ενέργεια ή δεξιό τμήμα (Right-Hand-Side). Οι συνθήκες μπορεί να αποτελούν μια σειρά συνθηκών που συνδέονται μεταξύ τους μέσω λογικών τελεστών. Στην περίπτωση που το σύνολο των συνθηκών ικανοποιηθεί τότε εκτελείται η ακολουθία των ενεργειών.

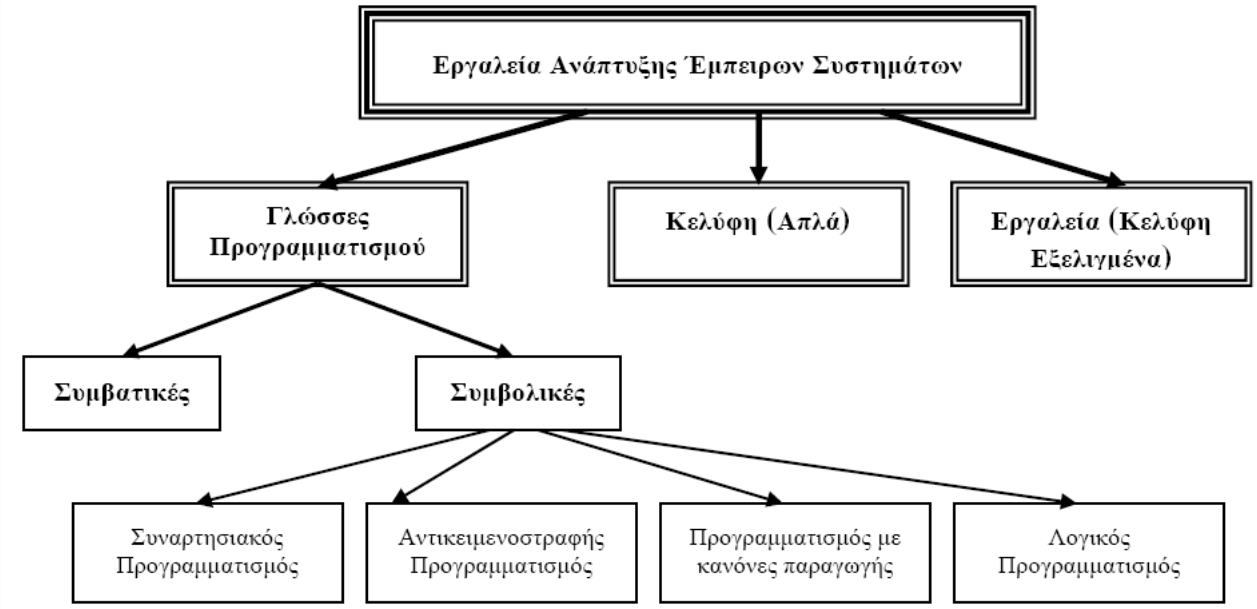
Επιγραμματικά στα πλεονεκτήματα των κανόνων παραγωγής είναι η ευκολία που παρέχουν στην αναπαράσταση γνώσης. Μπορεί πολύ εύκολα να εμπλουτιστεί η βάση γνώσης προσθέτοντας συνεχώς και νέους κανόνες. Ο κάθε κανόνας λειτουργεί αυτόνομα και μπορεί να αλλάξει ανά πάσα στιγμή χωρίς να επηρεάσει τη λειτουργία των άλλων. Ο τρόπος λειτουργίας του είναι κοντά στον ανθρώπινο τρόπο σκέψης.

Κανόνες παραγωγής, αν και αρχικά είχαν χρησιμοποιηθεί σε μοντέλα ψυχολογίας, επί του παρόντος εξακολουθούν να αποτελούν τον πιο ευρέως διαδεδομένο φορμαλισμό

αναπαράστασης γνώσης σε έμπειρα συστήματα. Χρησιμοποιούνται κυρίως σε προβλήματα κατηγοριοποίησης (classification). Μεγάλη επιτυχία των κανόνων παραγωγής είναι σε έμπειρα συστήματα διάγνωσης ασθενειών και ιδιαίτερα στα συστήματα MYCIN και EMYCIN.

Εργαλεία Ανάπτυξης Έμπειρων Συστημάτων

Τα εργαλεία υλοποίησης των Έμπειρων Συστημάτων χωρίζονται σε τρεις κατηγορίες, τις Γλώσσες Προγραμματισμού, τα Κελύφη και τα Εργαλεία. Σύμφωνα με άλλες απόψεις ο διαχωρισμός γίνεται σε γλώσσες προγραμματισμού και κελύφη τα οποία διακρίνονται σε απλά και εξελιγμένα, όπου με τα εξελιγμένα υπονοούνται τα εργαλεία. Σήμερα η επικρατούσα άποψη είναι τα εξελιγμένα κελύφη να αναφέρονται ως εργαλεία διότι διαθέτουν πολλές δυνατότητες παραμετροποίησης του συστήματος και όχι μόνο τη Βάση Γνώσης όπως στα κελύφη. Ακολουθεί η αναλυτική παρουσίαση των τριών αυτών ομάδων εργαλείων.



Σχήμα 37: Εργαλεία Ανάπτυξης Έμπειρων Συστημάτων

Γλώσσες Προγραμματισμού

Ένα έμπειρο σύστημα μπορεί να υλοποιηθεί σε οποιαδήποτε γλώσσα προγραμματισμού. Οι γλώσσες προγραμματισμού διακρίνονται σε Συμβατικές και Συμβολικές. Οι συμβατικές γλώσσες έχουν το πλεονέκτημα ότι υποστηρίζονται από τις υπάρχουσες υπολογιστικές υποδομές, όπως εργαλεία ανάπτυξης λογισμικού, διασύνδεση με βάσεις δεδομένων, συνεργασία με βοηθητικά προγράμματα. Βασικό τους μειονέκτημα είναι η δυσκολία στην μοντελοποίηση της γνώσης, δηλαδή των εννοιών και των εκφράσεων που αναπαρίστανται στη Βάση Γνώσης. Μια ακόμη δυσκολία έγκειται και στο γεγονός ότι οι γλώσσες προγραμματισμού υψηλού επιπέδου απαιτούν μεγάλο βαθμό εξειδίκευσης και μακρόχρονη εμπειρία. Τέτοιες γλώσσες είναι οι JAVA, FORTRAN, C/C++, κ.α.

Αντίθετα οι συμβολικές γλώσσες, όπως οι LISP, PROLOG και OPS5, είναι προσανατολισμένες στη χρήση συμβόλων και παρέχουν εξελιγμένους μηχανισμούς χειρισμού συμβολικών εκφράσεων, αυτόματη διαχείριση μνήμης και ευέλικτες δομές ελέγχου. Οι γλώσσες αυτές είναι ευκολότερες στην εκμάθηση από τα εργαλεία τεχνολογίας της γνώσης, γιατί υποστηρίζουν έναν μόνο τρόπο αναπαράστασης γνώσης και συλλογιστικής. Οι συμβολικές γλώσσες ή γλώσσες Τεχνητής Νοημοσύνης (ΤΝ) διακρίνονται στις εξής κατηγορίες:

- Συναρτησιακός προγραμματισμός (π.χ. LISP)

- Λογικός προγραμματισμός (π.χ. PROLOG)
- Αντικειμενοστραφής προγραμματισμός (π.χ. SMALLTALK)
- Προγραμματισμός με κανόνες παραγωγής (π.χ. OPS5)

Χρησιμοποιώντας μία γλώσσα προγραμματισμού ο μηχανικός γνώσης έχει τη δυνατότητα όχι μόνο να αναπαραστήσει τη γνώση με βάση τις δομές δεδομένων που ορίζει η γλώσσα, αλλά και να αναπτύξει το μηχανισμό εξαγωγής συμπερασμάτων για το έμπειρο σύστημα. Αποτελούν ένα εργαλείο για γρήγορη κατασκευή πρωτοτύπου του έμπειρου συστήματος. Δυστυχώς η διεπαφή με τον χρήστη δεν είναι αρκετά εξελιγμένη και φιλική.

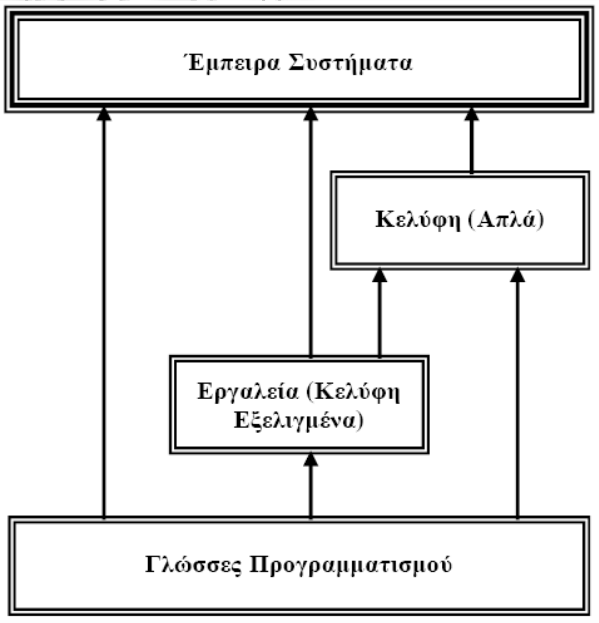
Κελύφη

Τα κελύφη αποτελούν Έμπειρα Συστήματα στα οποία απουσιάζει η Βάση Γνώσης. Το πρώτο έμπειρο σύστημα το οποίο εισήγαγε την έννοια του κελύφους ήταν το Mycin. Τα επόμενα συστήματα βασίστηκαν στο Mycin και με διαφορετική Βάση Γνώσης προσπάθησαν να επιλύσουν προβλήματα σε διάφορους τομείς. Μερικά από αυτά τα συστήματα είναι το LITHIO (γεωλογία), το CLOT (ασθένειες πήξης του αίματος), το PUFF (πνευμονικές ασθένειες) κ.α.

Επειδή βασίζονται σε προϋπάρχοντα συστήματα, δεν παρέχονται δυνατότητες ριζικών αλλαγών στην υλοποίηση των νέων συστημάτων. Εκτός από τη Βάση Γνώσης δεν μπορούν να γίνουν ριζικές αλλαγές, όπως για παράδειγμα στον τρόπο αναπαράστασης της γνώσης, στη συλλογιστική κ.α.. Το γεγονός αυτό περιορίζει και το εύρος των προβλημάτων που μπορούν να αντιμετωπίσουν. Για παράδειγμα το Mycin και τα συστήματα που το διαδέχθηκαν χρησιμοποιήθηκαν σε προβλήματα διάγνωσης. Ορισμένα χαρακτηριστικά κελύφη είναι τα KAS, EXPERT, EMYCIN κ.α.

Εργαλεία

Τα εργαλεία (tools ή toolkits) ή εξελιγμένα κελύφη, διαφοροποιούνται από τα απλά κελύφη ως προς το ότι παρέχουν ένα ολοκληρωμένο περιβάλλον ανάπτυξης του συστήματος, με δυνατότητες υποστήριξης πολλών τρόπων αναπαράστασης γνώσης και συλλογιστικής. Μέσω των εργαλείων τα έμπειρα συστήματα μπορούν να υλοποιηθούν από μηδενικής βάσης χωρίς να βασίζονται σε ένα προϋπάρχον σύστημα. Μπορούν με αυτό τον τρόπο να επεκτείνουν το εύρος των εφαρμογών που μπορούν να αντιμετωπίσουν. Για τους παραπάνω λόγους προτείνουμε την ορολογία των εργαλείων κα όχι των εξελιγμένων κελυφών. Η ανάπτυξη ενός έμπειρου συστήματος με τους τρεις διαφορετικούς τρόπους και οι αλληλεξαρτήσεις τους παρουσιάζεται στο επόμενο σχήμα:



Σχήμα 38: Τρόποι Ανάπτυξης ενός Έμπειρου Συστήματος

Τα εργαλεία ως ολοκληρωμένα περιβάλλοντα ανάπτυξης εφαρμογών παρέχουν περισσότερες δυνατότητες στην υλοποίηση, υποστήριξη και εξέλιξη των συστημάτων. Σημαντική είναι και η καλύτερη διεπαφή του συστήματος τόσο με τον μηχανικό γνώσης όσο και με τον τελικό χρήστη.

Υπάρχουν όμως και μειονεκτήματα όπως ότι συνήθως είναι δυσκολότερα στην εκμάθηση γιατί περιέχουν πολλές και ετερογενείς μεταξύ τους προγραμματιστικές έννοιες. Η ανάμειξη διαφόρων προγραμματιστικών μοντέλων γίνεται πειραματικά από τους προγραμματιστές, γιατί δεν υπάρχουν θεωρίες υβριδικών τρόπων αναπαράστασης της γνώσης. Χαρακτηριστικά έμπειρα συστήματα τα οποία αναπτύχθηκαν από εργαλεία είναι τα PERSONAL CONSULTANT, ART, KEE, NEXPERT, CLIPS κ.α.

Πλεονεκτήματα Έμπειρων Συστημάτων

Η ταχεία εξέλιξη των έμπειρων συστημάτων και η ευρεία χρήση τους οφείλεται σε μια σειρά πλεονεκτημάτων που προσφέρουν σε σχέση τόσο με τα συμβατικά συστήματα αλλά πολύ περισσότερο σε σχέση με έναν άνθρωπο-ειδικό. Επιγραμματικά αναφέρουμε τους πιο σημαντικούς από αυτούς τους λόγους.

Μονιμότητα γνώσης

Τα έμπειρα συστήματα δεν ξεχνούν, σε αντίθεση με τους ανθρώπους.

Αναπαραγωγή

Η δημιουργία πολλαπλών αντιγράφων ενός έμπειρου συστήματος είναι μια εύκολη και φθηνή διαδικασία. Αντίθετα η εκπαίδευση ανθρώπινου προσωπικού για την ανάλογη εξειδίκευση είναι μια χρονοβόρα και δαπανηρή διαδικασία.

Αποδοτικότητα-Κόστος

Τα έμπειρα συστήματα μπορούν να αυξήσουν την παραγωγικότητα και να μειώσουν το κόστος προσωπικού. Τα έμπειρα συστήματα έχουν ένα υψηλό κόστος παραγωγής άλλα ένα χαμηλό κόστος λειτουργίας. Εξάλλου το συνολικό κόστος παραγωγής και συντήρησης μπορεί να μοιραστεί σε πολλούς χρήστες.

Συνέπεια

Τα έμπειρα συστήματα συμπεριφέρονται με παρόμοιο τρόπο σε παρόμοια προβλήματα. Οι άνθρωποι επηρεάζονται από πολλούς παράγοντες οι οποίοι μειώνουν την αντικειμενικότητά τους.

Τεκμηρίωση γνώσης

Ένα έμπειρο σύστημα παρέχει μια μόνιμη τεκμηρίωση της γνώσης και των κανόνων του γνωστικού πεδίου. Σε πολλές περιπτώσεις άλλωστε χρησιμοποιούνται τα έμπειρα συστήματα για την εκπαίδευση ανθρώπων.

Πληρότητα

Ένα έμπειρο σύστημα μπορεί σε μικρό χρόνο να εξετάσει πολλές πιθανές λύσεις και να λάβει υπόψη του όλα τα δεδομένα εισόδου.

Έγκαιρη πληροφορία

Η έγκαιρη πληροφορία μπορεί να βοηθήσει στην αποφυγή λαθών.

Εύρος γνώσης

Σε ένα έμπειρο σύστημα συνδυάζεται η γνώση και εμπειρία πολλών έμπειρων ανθρώπων. Η γνώση που αποθηκεύεται έχει μεγαλύτερο εύρος από τη γνώση ενός μεμονωμένου ειδικού-πεδίου.

Μειονεκτήματα Έμπειρων Συστημάτων

Όπως και σε κάθε άλλο υπολογιστικό σύστημα τα έμπειρα συστήματα έχουν και μειονεκτήματα που αφορούν στην καταλληλότητά τους ως προς το είδος των προβλημάτων

που καλούνται να αντιμετωπίσουν, στα μειονεκτήματα που παρουσιάζουν ως υπολογιστικά συστήματα, αλλά πολύ περισσότερο στην αδυναμία τους να λειτουργήσουν στο επίπεδο συλλογισμού ενός ανθρώπου.

Αναφέρουμε επιγραμματικά και σε αυτή την παράγραφο το πιο χαρακτηριστικά μειονεκτήματά τους.

Έλλειψη κοινής λογικής

Οι ειδικοί- πεδίου συνδυάζουν πολλές φορές μαζί με την γνώση και την εμπειρία τους, σοφία ή κοινή λογική για την αποδοχή ή απόρριψη λύσεων. Τα έμπειρα συστήματα μπορούν μεν να δικαιολογήσουν τις λύσεις που προτείνουν αλλά δεν μπορούν να τις αξιολογήσουν.

Έλλειψη Δημιουργικότητας

Οι άνθρωποι μπορούν να αντιδρούν δημιουργικά (αν και μερικές φορές λανθασμένα) σε αναπάντεχες και ασυνήθιστες καταστάσεις.

Έλλειψη Εκμάθησης

Οι άνθρωποι μπορούν να προσαρμόζονται σε περιβάλλοντα που αλλάζουν συνεχώς μέσω της επικοινωνίας με το περιβάλλον και της εκμάθησης. (Τα νευρωνικά δίκτυα προσφέρονται για αυτόν το σκοπό, έχουν όμως το μειονέκτημα ότι δεν μπορούν να δικαιολογήσουν τα αποτελέσματά τους.)

Έλλειψη Αισθήσεων

Η δυνατότητα αίσθησης του περιβάλλοντος και η πολλαπλή ανάκτηση πληροφοριών μέσω των αισθητήριων οργάνων επιτρέπει στους ανθρώπους μια συνεχή ανανέωση της γνώσης. Τα έμπειρα συστήματα περιορίζονται στην εισαγωγή της γνώσης μέσω της συμβολικής της αναπαράστασης.

Περιορισμένο γνωστικό αντικείμενο

Τα έμπειρα συστήματα δεν είναι καλά στο να αντιμετωπίζουν ασαφείς ή νέες καταστάσεις, ούτε στο να παρέχουν απαντήσεις σε προβλήματα εκτός του γνωστικού τους πεδίο.

Σύγκριση Συμβατικών συστημάτων και Έμπειρων Συστημάτων

Εκτός από τα πλεονεκτήματα και μειονεκτήματα των έμπειρων συστημάτων ως προς την αντιμετώπιση των προβλημάτων που καλούνται να επιλύσουν αλλά και πολύ περισσότερο ως προς τη σχέση τους με τον άνθρωπο-ειδικό, ενδιαφέρον παρουσιάζει και η σύγκριση τους με τα συμβατικά συστήματα. Ο παρακάτω πίνακας παρουσιάζει τις διαφορές αυτές και βοηθά στο να αποσαφηνίσουμε περαιτέρω τη φύση και τη λειτουργία τους.

Συμβατικά Συστήματα	Έμπειρα Συστήματα
Επεξεργασία και γνώση συνδυάζονται σε ένα σειριακό πρόγραμμα	Η Βάση Γνώσης είναι χωριστή από τον μηχανισμό επεξεργασίας
Τα προγράμματα δεν κάνουν λάθη	Το πρόγραμμα μπορεί να κάνει λάθος
Συνήθως δεν εξηγούν γιατί τα δεδομένα εισόδου χρειάζονται ή πώς συνάγονται τα συμπεράσματα	Η επεξήγηση είναι μέρος των περισσοτέρων έμπειρων συστημάτων
Η τροποποίηση της γνώσης είναι κοπιώδης	Η τροποποίηση των κανόνων είναι εύκολη
Η εκτέλεση είναι βήμα-προς-βήμα	Η εκτέλεση βασίζεται σε ευρεστικούς κανόνες και την λογική
Χρειάζονται ολοκληρωμένη πληροφορία	Διαχειρίζονται ελλιπή ή ασαφή πληροφορία
Αποτελεσματική διαχείριση μεγάλων βάσεων δεδομένων	Αποτελεσματική διαχείριση μεγάλων βάσεων δεδομένων

Αναπαράσταση και χρήση δεδομένων	Αναπαράσταση και χρήση γνώσης

Πίνακας 5: Σύγκριση Συμβατικών Συστημάτων και Έμπειρων Συστημάτων

Γνωστά Έμπειρα Συστήματα

Τα Έμπειρα Συστήματα χωρίζονται σε 2 κατηγορίες ανάλογα με την περίοδο που αναπτύχθηκαν. Στην πρώιμη περίοδο των πρώτων προσπαθειών από το τέλος της δεκαετίας του '60 μέχρι το τέλος της δεκαετίας του '70 ανήκουν τα συστήματα πρώτης γενιάς. Στη συνέχεια και προκειμένου να επιλυθούν οι αδυναμίες που παρουσίασαν τα συστήματα της πρώτης γενιάς δημιουργήθηκαν τα έμπειρα συστήματα δεύτερης γενιάς. Ακολουθεί μια παρουσίαση των πιο αντιπροσωπευτικών συστημάτων της κάθε γενιάς και των ιδιαίτερων χαρακτηριστικών τους.

Έμπειρα Συστήματα Πρώτης Γενιάς

Τα τρία αντιπροσωπευτικά συστήματα της πρώτης γενιάς τα οποία θα εξετάσουμε είναι το MYCIN, το PROSPECTOR και το INTERNIST-I/CADUCEUS τα οποία αναπτύχθηκαν παράλληλα γύρω στα μέσα της δεκαετίας του 70. Η αρχική προσέγγιση επικεντρωνόταν στο επίπεδο αναπαράστασης (representation level) δίνοντας έμφαση στην ομοιομορφία ως προς την αναπαράσταση γνώσης και κατά συνέπεια στην απλότητα συλλογισμού.

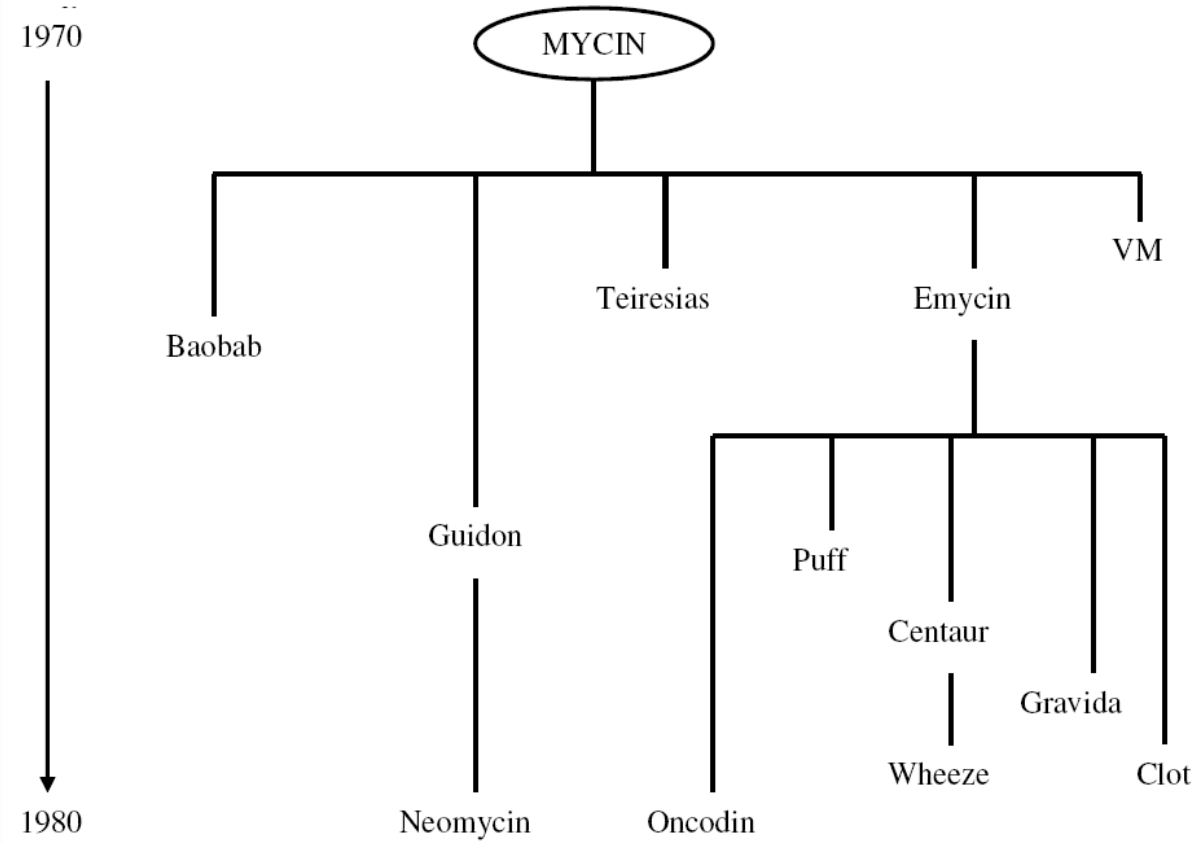
MYCIN

Το MYCIN αποτελεί μια από τις πρώτες προσπάθειες δημιουργίας έμπειρων συστημάτων και αναπτύχθηκε τη δεκαετία του '70 στο Πανεπιστήμιο του Stanford. Πρόκειται για ένα ιατρικό έμπειρο σύστημα το οποίο έκανε διαγνώσεις μολύνσεων από βακτήρια και συνέστηνε αντιβιοτική θεραπεία. Το διαγνωστικό του μέρος ήταν ένα σύστημα παραγωγής το οποίο εφάρμοζε ανάστροφη αλυσίδωση για την εξαγωγή συμπερασμάτων. Η Βάση Γνώσης του συστήματος περιλάμβανε περίπου 450 κανόνες. Εκτός από το μηχανισμό εξαγωγής συμπερασμάτων εισήγαγε και άλλες καινοτομίες όπως συντελεστές βεβαιότητας, δηλαδή βάρη, ως ένα νέο τρόπο μοντελοποίησης ασαφούς λογικής. Από το MYCIN προήλθε και η έννοια του συστήματος κελύφους που αποτέλεσε την πρώτη προσέγγιση στην υλοποίηση έμπειρων συστημάτων.

Η συμβολή του στην περαιτέρω εξέλιξη των Έμπειρων Συστημάτων ήταν καθοριστική όχι μόνο ως προς το καθαυτό σύστημα αλλά και σε σχέση με ένα σύνολο υποσυστημάτων τα οποία αναπτύχθηκαν υποστηρικτικά προς αυτό. Σημαντικά υποσυστήματα ήταν:

- Το σύστημα επεξηγήσεων
- Το σύστημα ημιαυτόματης απόκτησης γνώσης (TEIRESIAS)
- Σύστημα Εκμάθησης (GUIDON)
- Σύστημα κελύφους (EMYCIN)

Η συμβολή του MYCIN και των υποσυστημάτων του στην εξέλιξη των έμπειρων συστημάτων αποτυπώνεται στο σχήμα που ακολουθεί [3w11]:

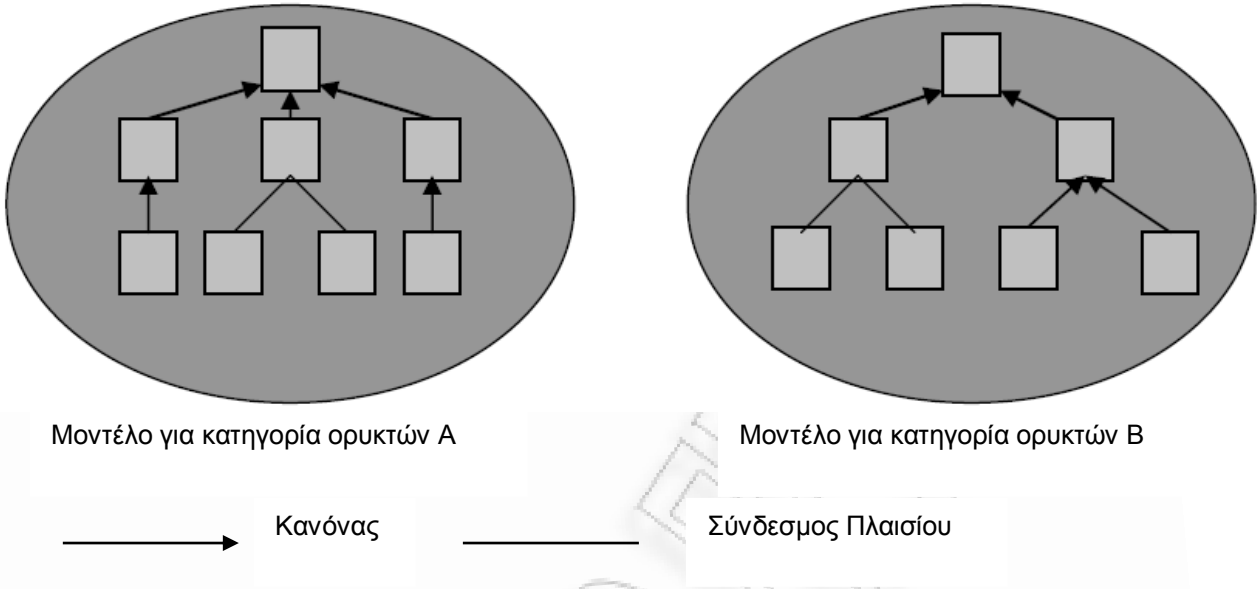


Σχήμα 39: Εξέλιξη του Mycin

Παρά το σημαίνοντα ρόλο που έπαιξε στην πορεία των έμπειρων συστημάτων, δεν χρησιμοποιήθηκε ποτέ σε πραγματικές συνθήκες, καθώς χρειάζονταν περίπου 48 ώρες για να βγάλει αποτελέσματα, χρονικό διάστημα που μπορούσε να αποβεί μοιραίο για τον ασθενή.

PROSPECTOR

Ένα ακόμη έμπειρο σύστημα το οποίο αναπτύχθηκε στο πανεπιστήμιο του Stanford είναι το Prospector. Δημιουργήθηκε προκειμένου να χρησιμοποιηθεί από γεωλόγους στην αναζήτηση και αξιολόγηση κοιτασμάτων ορυκτών. Το Prospector είναι όπως και το Mycin ένα σύστημα παραγωγής το οποίο χρησιμοποιεί μια υβριδική αναπαράσταση την γνώσης μέσω κανόνων και σημασιολογικών δικτύων. Συνολικά, η Βάση Γνώσης περιέχει γύρω στα 15 μοντέλα ορυκτών, το καθένα αποτελούμενο από πέραν των 150 κανόνων και 200 κόμβων.



Σχήμα 40: Prospector

Τα συμπεράσματα των κανόνων αναπαρίστανται από διαμερίσεις παρέχοντας υψηλότερη δύναμη εκφρασιμότητας όπως πχ ποσοτικοποιημένες εκφράσεις, δυνατότητες που δεν υπάρχουν στο Mycin. Η μηχανή εξαγωγής συμπεράσματος υποστηρίζει μεικτή αλυσίδωση, δηλαδή ορθή και ανάστροφη, η οποία υλοποιεί υποθετικο–συμπερασματικό συλλογισμό που βασίζεται στη χρήση συντελεστών βεβαιότητας και στον καθορισμό πιθανοτήτων όσον αφορά τα δεδομένα. Το σύστημα υλοποιήθηκε σε Interlisp.

INTERNIST-1

Το INTERNIST–1, αναπτύχθηκε στο Πανεπιστήμιο του Pittsburgh με στόχο την υποβοήθηση ιατρών στη διεξαγωγή πολύπλοκων διαγνώσεων που αφορούσαν στον τομέα της γενικής παθολογίας. Λόγω του μεγάλου εύρους εφαρμογής του συστήματος, αποτέλεσε ένα από τα μεγαλύτερα έμπειρα συστήματα που είχαν αναπτυχθεί, περιέχοντας προφίλ για περισσότερες από 500 ασθένειες οι οποίες περιγράφονται σε παραπάνω από 3500 ενδείξεις.

Το σύστημα INTERNIST–1, το οποίο έχει μία από τις πιο εκτενείς βάσεις γνώσης (ο τομέας του είναι γενική παθολογία) διαφέρει από τα άλλα δύο συστήματα ως προς τη «μεθοδολογία» που ακολουθήθηκε για την ανάπτυξή του. Στην περίπτωση των MYCIN και PROSPECTOR αποφασίσθηκε καταρχάς ο τρόπος αναπαράστασης της γνώσης, συγκεκριμένα η χρήση κανόνων παραγωγής ως η κύρια μορφή αναπαράστασης, και στη συνέχεια η προσπάθεια εστιάστηκε στην απόσπαση της γνώσης των εν λόγω εμπείρων σε αυτή τη μορφή. Στο INTERNIST–1 το πρώτο βήμα αφορούσε την κατανόηση της εμπειρογνωμοσύνης και το σχεδιασμό ενός ιδεατού μοντέλου για αυτήν. Αναπτύχθηκε έτσι ως ένα σχετικά απλό σύστημα πλαισίων, όπου τα πλαίσια (για τις ασθένειες) οργανώνονται με ιεραρχικό τρόπο, και εσωτερικά τα πλαίσια περιέχουν διαφόρων ειδών συσχετίσεις (associations), αλλά όχι συλλογιστική γνώση. Ο συλλογισμός είναι σε καθολικό επίπεδο και μάλιστα υλοποιείται, όχι σε μορφή πλαισίων ελέγχου, αλλά σε μορφή διαδικασιών. Ο συλλογισμός του INTERNIST–1 είναι κυρίως συμπερασματικής μορφής με στόχο την απόσπαση νέων παρατηρήσεων από το χρήστη, οι οποίες αφορούν προβλεπόμενες ενδείξεις των ανταγωνιζόμενων υποθέσεων. Υπάρχουν τρεις στρατηγικές (ευρετικά): Αποκλεισμός (Ruleout), Διαχωρισμός (Discriminate) και Επιδίωξη (Pursue).

Ουσιαστικά το INTERNIST–1 είναι ένα σύστημα ταξινόμησης, αφού το βασικό πρόβλημα που επιλύει είναι η κατάταξη των ανεξήγητων θετικών ενδείξεων σε μία από τις προτεινόμενες ασθένειες. Διάδοχοι του INTERNIST–1 ήταν το σύστημα α CADUCEUS, το οποίο στόχευε σε μία πιο πλούσια οργάνωση της βάσης γνώσης κυρίως σε σχέση με αιτιολογικές και ταξινομικές

σχέσεις, και πιο πρόσφατα το σύστημα QMR, το οποίο ανάμεσα σε άλλα στόχευε στη μοντελοποίηση της διάστασης του χρόνου. Το σύστημα υλοποιήθηκε σε LISP.

Έμπειρα Συστήματα Δεύτερης Γενιάς

Στόχος της δεύτερης γενιάς έμπειρων συστημάτων είναι η απαλοιφή των αδυναμιών της πρώτης γενιάς. Στην πραγματικότητα ο όρος «συστήματα δεύτερης γενιάς» είναι κάπως ασαφής, διότι δεν υπάρχει ξεκάθαρη διαχωριστική γραμμή ανάμεσα στα συστήματα «πρώτης» και «δεύτερης» γενιάς. Μια διαφορετική αλλά αντίστοιχη ταξινόμηση των έμπειρων συστημάτων είναι σε «ρηχά» και «βαθιά» υπονοώντας κυρίως τη διαφοροποίηση τους ως προς την αναπαράσταση της γνώσης.

Η δεύτερη γενιά έμπειρων συστημάτων περιλαμβάνει όλα τα συστήματα από τις αρχές της δεκαετίας του ΄80 μέχρι σήμερα. Ακολουθεί μια επιλεκτική ανάλυση 3 συστημάτων προκειμένου να εντοπιστούν οι κύριες διαφορές μεταξύ των 2 γενιών. Τα συστήματα που θα αναφέρουμε είναι το Neomycin το οποίο αποτελεί εξέλιξη του Mycin, το Caduceus πού είναι εξέλιξη του INTERNIST–1 και το MDX το οποίο παρουσιάζει μια ιδιαίτερη αρχιτεκτονική.

Η κύρια διάκριση των 2 γενιών έγκειται σε τρία σημεία:

1. Στη χρήση πολλαπλών μοντέλων συλλογισμού. Συνήθως υπάρχουν δύο μοντέλα, το ευρετικό μοντέλο (heuristic model) και κάποιο «βαθύτερο», αιτιολογικό κυρίως μοντέλο (causal model).
2. Στο σχεδιασμό που είναι προσανατολισμένος στο επίπεδο γνώσης. Είναι πιο κατανοητό ο σχεδιασμός να αναφέρεται σε επίλυση με ταξινόμηση ή τη χρήση απαγωγικού συλλογισμού, παρά να αναφέρεται σε ανάστροφη αλυσίδωση κανόνων, αφού έτσι παρέχεται ένας πιο ιδεατός χαρακτηρισμός της συμπεριφοράς του επιδιωκόμενου συστήματος.
3. Τέλος δίνεται μεγαλύτερη έμφαση στην ικανότητα μάθησης και αυτοβελτίωσης εκ μέρους του συστήματος, όπου για παράδειγμα το ευρετικό μοντέλο μπορεί να παραχθεί σταδιακά από το «βαθύτερο» μοντέλο, με (ημι)αυτόματο τρόπο.

Σαν γενικό συμπέρασμα θα μπορούσαμε να αναφέρουμε ότι η δεύτερη γενιά σαφώς επικεντρώνεται στο διαχωρισμό και στη ρητή μοντελοποίηση και αναπαράσταση περισσότερων ειδών γνώσης, σε σύγκριση με τα συστήματα πρώτης γενιάς. Είναι σημαντικό να αναφέρουμε ότι στα συστήματα δεύτερης γενιάς έγινε σημαντική προσπάθεια στην επαναχρησιμοποίηση του τρόπου αναπαράστασης της γνώσης. Στην πρώτη γενιά εισήχθη η έννοια του κελύφους όπου αφορούσε την επαναχρησιμοποίηση του συμβολικού τρόπου αναπαράστασης της γνώσης. Στη δεύτερη γενιά εξελίχθηκε η η έννοια της επαναχρησιμοποίησης, κυρίως σε επίπεδο γνώσης, π.χ. όπως επαναχρησιμοποίηση βάσεων γνώσης ή συμβολικών μονάδων, δηλαδή κώδικα. Αυτές οι μονάδες δεν αποτελούν κατ' ανάγκη ολοκληρωμένες απόψεις ενός συστήματος, όπως συμβαίνει με τα συστήματα κελύφους, αλλά απλώς αποτελούν βασικά στοιχεία (building blocks), τα οποία χρειάζεται να συντεθούν για την ολοκλήρωση του συστήματος.

NEOMYCIN

Το NEOMYCIN αναπτύχθηκε, όπως και το MYCIN στο Πανεπιστήμιο του Stanford. Το βασικό κίνητρο για τη δημιουργία του NEOMYCIN οφείλεται στις διάφορες ελλείψεις και τα εγγενή προβλήματα του GUIDON, του διδακτικού συστήματος του MYCIN. Η αποτυχία του GUIDON έγκειται στο γεγονός ότι ο συλλογισμός του MYCIN, με άλλα λόγια η ανάστροφη αλυσίδωση κανόνων, που το GUIDON προσπαθούσε να διδάξει, δεν ταυτιζόταν με τον ανθρώπινο συλλογισμό. Ο κεντρικός στόχος πίσω από τη δημιουργία του NEOMYCIN ήταν η ρητή μοντελοποίηση της γνώσης ως βασική προϋπόθεση για την αποδοτική διδασκαλία διαγνωστικού συλλογισμού. Εκτός από τη ρητή μοντελοποίηση της στρατηγικής γνώσης, η γνώση αναφορικά με τις μολύνσεις περιγράφεται με δύο διαφορετικούς τρόπους, με βάση δύο ξεχωριστά μοντέλα, του ταξινομικού μοντέλου, το οποίο αποτελεί τη γνώση δόμησης (structural knowledge), και του αιτιολογικού μοντέλου, το οποίο αποτελεί τη γνώση υποστήριξης (support knowledge).

Η λογική του συστήματος είναι η κάθε τελική διάγνωση να υποδιαιρείται σε ενδιάμεσα στάδια δημιουργώντας έτσι ένα δένδρο απόφασης. Η κάθε διεργασία χωρίζεται σε υπο-εργασίες οι οποίες εκφράζονται με τη μορφή μετα-κανόνων. Οι μετα–κανόνες της υπό εκτέλεση εργασίας διερμηνεύονται με ορθή αλυσίδωση. Οι μετα–κανόνες αναφέρονται σε αφηρημένες έννοιες, όπως «ένδειξη» και «αιτία», και όχι σε συγκεκριμένες έννοιες, όπως «πονοκέφαλος» και «μηνιγγίτιδα».

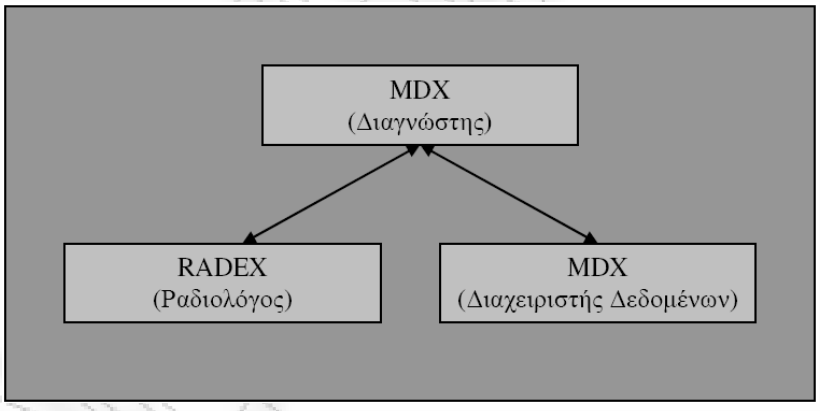
CADUCEUS

Το CADUCEUS αποτελεί ένα διαγνωστικό έμπειρο σύστημα το οποίο δημιουργήθηκε στο πανεπιστήμιο του Pittsburgh. Αποτέλεσε την μετεξέλιξη του INTERNIST-I και όταν τελικά ολοκληρώθηκε μπορούσε να διαγνώσει πάνω από 1000 ασθένειες. Το INTERNIST-I ως σύστημα δεν έχει τρόπο αναίρεσης προηγούμενων συμπερασμάτων και επέτρεπε ένα λάθος να διαδοθεί και σε μεταγενέστερα συμπεράσματα. Το CADUCEUS υλοποιήθηκε προκειμένου να απαληφθεί ο ακολουθιακός τρόπος κατασκευής σύνθετων λύσεων. Η μηχανή εξαγωγής συμπερασμάτων βασίστηκε σε αυτή του Mycin ενσωματώνοντας επιπλέον χαρακτηριστικά όπως απαγωγική συλλογιστική που επέτρεπε την αντιμετώπιση πιο σύνθετων προβλημάτων διάγνωσης.

MDX

Το σύστημα MDX αναπτύχθηκε στο Πανεπιστήμιο του Ohio State με κύριους ερευνητές τους B. Chandrasekaran και S. Mittal. Στόχος του συστήματος είναι η διάγνωση του συνδρόμου του ήπατος, γνωστού ως χολέσταση. Όπως αναφέρθηκε και στην εισαγωγή η γνώση διακρίνεται σε ρηχή και βαθιά. Στο MDX προτείνεται η μεταγλωτισμένη γνώση η οποία τοποθετείται ανάμεσα στο φάσμα μεταξύ της ρηχής και βαθιάς γνώσης προσπαθώντας να αποκτήσει τα θετικά χαρακτηριστικά και των δύο.

Το σύστημα MDX αποτελείται από τρεις ξεχωριστές μονάδες, την κυρίως μονάδα, τον διαγνώστη, που επίσης ονομάζεται MDX, και δύο βοηθητικές μονάδες, το διαχειριστή δεδομένων (PATREC) και το ραδιολόγο (RADEX) όπως φαίνεται και στο σχήμα που ακολουθεί:



Σχήμα 41: MDX

Το καθένα από τα υποσυστήματα αυτά λειτουργεί αυτόνομα παρέχοντας την απαραίτητη γνώση. Η μεταξύ τους επικοινωνία πραγματοποιείται μέσω συγκεκριμένου πρωτοκόλλου. Το τμήμα του διαγνώστη είναι υπεύθυνο για το συντονισμό των υποσυστημάτων. Το MDX υλοποιήθηκε σε γλώσσα LISP.

Η Αρχιτεκτονική του MDX εξελίχθηκε και γενικεύθηκε από τον B. Chandrasekaran ο οποίος πρότεινε την Αρχιτεκτονική Γενικευμένων Εργασιών (Generic Tasks Architecture) και η οποία παρέχει μια πιο ιδεατή ερμηνεία του όρου της επαναχρησιμοποίησης, από την ερμηνεία που αποδόθηκε στα πλαίσια της πρώτης γενιάς. Μία γενικευμένη εργασία συνδέεται με συγκεκριμένο στόχο και χαρακτηρίζεται από τα εξής:

- Τις δομές γνώσης που χρησιμοποιεί, με άλλα λόγια το μοντέλο γνώσης της.
- Τους τύπους των πληροφοριών που αποτελούν την είσοδο και έξοδό της.

- Το μηχανισμό ελέγχου.

Οι βασικές γενικευμένες εργασίες μπορούν να συντεθούν προς κατασκευή σύνθετων γενικευμένων εργασιών. Μία σύνθετη εργασία χαρακτηρίζεται από:

- Τις γενικευμένες εργασίες που την απαρτίζουν.
- Τις διαδρομές και τις συνθήκες, οι οποίες διέπουν τη μεταβίβαση πληροφοριών ανάμεσα σε αυτές.

Εμπορικά Συστήματα Διαχείρισης Κανόνων Παραγωγής

Τρεις βασικοί λόγοι οδηγούν την αγορά λογισμικού στη χρήση συστημάτων επιχειρηματικών κανόνων:

- Πρώτον, οι προμηθευτές λογισμικού επιμένουν στην παραγωγή λογισμικού για τη μοντελοποίηση επιχειρησιακών διαδικασιών οι οποίοι θα παρέχουν τη δυνατότητα σχεδιασμού μέσω γραφικού περιβάλλοντος από τους ειδικούς του κάθε τομέα, χωρίς τη διαμεσολάβηση ή υλοποίηση από προγραμματιστές. Οι επιχειρησιακοί κανόνες θα υλοποιούνται και συντηρούνται απευθείας από τους ανθρώπους που κατέχουν την επιχειρησιακή γνώση και λογική.
- Δεύτερον, οι ανεξάρτητοι προμηθευτές παρέχουν συστήματα κανόνων που εμπεριέχουν υλοποιημένους κανόνες, με σκοπό τη διαφοροποίηση τους από τις IBM, Microsoft, Oracle, SAP, και άλλους προμηθευτές τέτοιων συστημάτων που προσφέρουν επιχειρηματικούς κανόνες.
- Τρίτον, ένα νέο κύμα συγχωνεύσεων ξεκινά καθώς οι κύριοι μεγάλοι προμηθευτές λογισμικού στοιχηματίζουν σε αυτή την κρίσιμη τεχνολογία.

Για τους προγραμματιστές, αυτές οι τάσεις θα αποφέρουν καλύτερα εργαλεία συνεργασίας με τους επιχειρηματίες, αλλά και μεγαλύτερο εύρος επιλογών από τους προμηθευτές.

Ενδεικτικά μπορούμε να αναφέρουμε τις παρακάτω σύγχρονες μηχανές παραγωγής που έχουν αναπτυχθεί με τα παρακάτω συγκριτικά αποτελέσματα:

	Jess	Drools	Blaze Advisor	ILOG JRules	OpenRules	VisualRules	QuickRules	OPSJ
Άδεια Χρήσης	Δωρεάν για Ακαδημαϊκή χρήση	Λογισμικό Ανοιχτού Κώδικα	Το κόστος υπολογίζεται ανά προγραμματιστή	(Starter Pack)	Το κόστος υπολογίζεται ανά εγκατάσταση	Ο τιμ/γος είναι διαθέσιμος κατόπιν αίτησης	Ο τιμ/γος είναι διαθέσιμος κατόπιν αίτησης	Το κόστος υπολογίζεται ανά προγραμματιστή
Περιβάλλον Ανάπτυξης	Eclipse IDE	Eclipse IDE	IDE	IDE	Eclipse IDE	IDE	Eclipse IDE / NetBeans IDE	Eclipse IDE
Υλοποίηση του Προτύπου JSR 94	Ναι	Ναι	Ναι	Ναι	Ναι	Ναι	Ναι	Ναι
BRMS	Όχι	Ναι	Εφαρμογές Συντήρησης Κανόνων	Rule Studio	MS Excel	Ναι	Rule Builder IDE	Παραθυρικό Περιβάλλον Ανάπτυξης
Ορθή Αλυσίδωση	Ναι	Ναι	Ναι	Ναι	Ναι	Ναι	Ναι	Ναι
Ανάστροφη Αλυσίδωση	Ναι	Όχι	Ναι	Όχι	Όχι	Όχι	Όχι	Ναι
Αλγόριθμος Ταυτοποίησης	Enhanced Rete	Rete OO	Rete III	Rete Sequential	XML Based	Όχι	Rete	Rete II
Fuzzy Logic	Ναι	Όχι	Διαχείριση Ελλιπών Δεδομένων	Όχι	Όχι	Όχι	Όχι	Όχι
Δυνατότητα Αποσφαλμάτωσης	Όχι	Ναι	Ναι	Ναι	Rule Validator	Visual Rule Modeler	Ναι	Όχι
Υποστήριξη Μηχανικής Μάθησης	Όχι	Όχι	Όχι	Όχι	Ναι	Όχι	Όχι	Όχι

Πίνακας 6: Σύγκριση μηχανών κανόνων

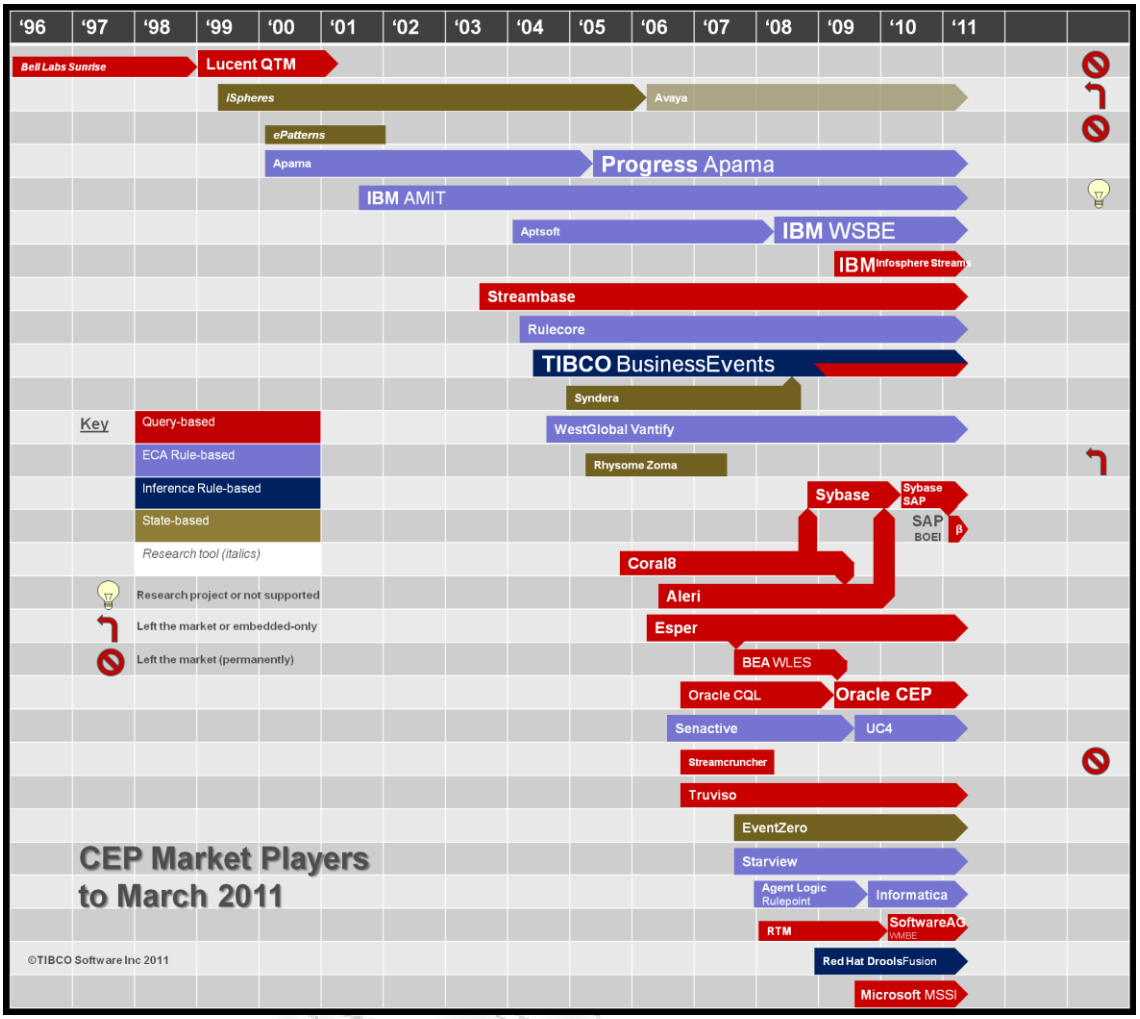
- Jess
  http://www.jessrules.com/
- Drools

http://www.jboss.org/drools

- Blaze Advisor
  http://www.fico.com/en/Products/DMTools/Pages/FICO-Blaze-Advisor-System.aspx
- ILOG
  http://www-01.ibm.com/software/websphere/ilog/
- OpenRules
  http://openrules.com/
- VisualRules
  http://www.visual-rules.com/technology.html
- QuickRules
  http://www.yasutech.com/products/quickrules/features.htm
- OPSJ
  http://www.pst.com/opsj.htm

Εμπορικά Συστήματα Επεξεργασίας Σύνθετων Συμβάντων

Από τις αρχές της δεκαετίας του 2000 δημιουργήθηκαν τα πρώτα ολοκληρωμένα εμπορικά συστήματα που υποστήριξαν την Επεξεργασία Σύνθετων Συμβάντων. Προς τα τέλη της δεκαετίας υπήρξε μια μεγάλη αύξηση των προσφερόμενων λύσεων, όπου άρχισαν να εισέρχονται και μεγάλες εταιρείες όπως η IBM, ORACLE κ.α. κυρίως μέσω εξαγοράς εταιρειών που είχαν αρχίσει να αναπτύσσουν από νωρίτερα CEP πλατφόρμες. Ακολουθεί από παρουσίαση της TIBCO (http://tibcoblogs.com/cep/wp-content/uploads/2011/03/cep-market-mar2011.png) μια επισκόπηση των προϊόντων που προσφέρονται σήμερα με την χρονική σειρά με την οποία εμφανίστηκαν:



Σχήμα 42: CEP Συστήματα

Ακολουθεί λίστα με τις συγκεκριμένες εταιρείες που αναφέρει το παραπάνω γράφημα:

- Progress Apama
  http://web.progress.com/en/apama/index.html
- StreamBase
  http://www.streambase.com/
- ruleCore CEP Server
  http://www.rulecore.com/
- WebSphere Business Events (WBE)
  http://www-01.ibm.com/software/solutions/soa/business_event_processing.html
- Tibco BusinessEvents
  http://www.tibco.com/products/business-optimization/complex-event-processing/default.jsp
- Aleri Sybase Event Streaming Platform
  http://www.sybase.com/products/financialservicessolutions/aleristreamingplatform
- Senactive (Εξαγοράστηκε από την uc4)
  http://www.uc4.com/

- EsperTech
  http://www.espertech.com/
- Oracle
  http://www.oracle.com/technetwork/middleware/complex-event-processing/overview/index.html
- Microsoft StreamInsight
  http://www.microsoft.com/sqlserver/2008/en/us/R2-complex-event.aspx
- Esper (Open Source)
  http://esper.codehaus.org/
- JBoss – Drools Fusion (Open Source)
  http://www.jboss.org/drools/drools-fusion.html

Τέλος παρατίθενται συστήματα τα οποία αναφέρονται αλλά δεν συνεχίστηκε η ανάπτυξή τους:

iSpheres – Αποτέλεσε πρωτοπόρα εταιρεία στην CEP αλλά τελικά πτώχευσε.
Syndera – Η εταιρεία πτώχευσε.
StreamCruncher – Open source πρόγραμμα το οποίο τελικά εγκαταλείφθηκε.
Coral8 – Εξαγοράστηκε από την Aleri.
Aleri – Εξαγοράστηκε από την Sybase
Senactive – Εξαγοράστηκε από την UC4