ΑΝΑΠΤΥΞΗ ΕΝΔΙΑΜΕΣΟΥ ΛΟΓΙΣΜΙΚΟΥ ΕΝΗΜΕΡΟΥ ΠΕΡΙΕΧΟΜΕΝΟΥ ΒΑΣΙΣΜΕΝΟ ΣΤΗΝ ΑΣΑΦΗ ΛΟΓΙΚΗ

55

ΜΠΛΕ

Η
ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

Υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης
του Τμήματος Πληροφορικής
Εξεταστική Επιτροπή

από τη

Ελένη Γεώργα

ως μέρος των Υποχρεώσεων

για τη λήψη

του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΙΣ ΤΕΧΝΟΛΟΓΙΕΣ-ΕΦΑΡΜΟΓΕΣ

Ιούνιος 2008

ΑΦΙΕΡΩΣΗ

Στον φίλο μου Σταύρο.

ΕΥΧΑΡΙΣΤΙΕΣ

Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Δημήτριο Ι. Φωτιάδη. Τον ευχαριστώ ιδιαίτερα για τις πολύτιμες συμβουλές που μου έδωσε και για την στήριξη και βοήθεια στην προσπάθειά μου να ολοκληρώσω την διατριβή μου.

Επίσης θα ήθελα να ευχαριστήσω την οικογένεια μου για την ηθική, ψυχολογική και οικονομική υποστήριξη που μου παρείχαν όλα αυτά τα χρόνια.

Ένα μεγάλο ευχαριστώ οφείλω στον φίλο μου Σταύρο Κατσαούνη για την αμέριστη συμπαράσταση και την υπομονή του σε όλη την διάρκεια των σπουδών μου.

Τέλος ευχαριστώ όλους όσους με βοήθησαν, ο καθένας με τον δικό του τρόπο, σε όλη την διάρκεια της προσπάθειάς μου. Ιδιαίτερα ευχαριστώ τον Τάσο Κοντογιώργη, τον Μάρκο Τσίπουρα, την Μαρία Βάββα, την Εύη Τριπολίτη, τον Νίκο Κατερτσίδη και τον Στέφανο Πέτσιο.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ

ΠΕΡΙΛΗΨΗ

Ελένη Γεώργα του Ιωάννου και της Κανέλλας. MSc, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Ιούνιος, 2008. Ανάπτυξη Ενδιάμεσου Λογισμικού Ενήμερου Περιεχομένου Βασισμένο στην Ασαφή Λογική. Επιβλέποντας: Δημήτριος Ι. Φωτιάδης.

Οι εφαρμογές που λειτουργούν σε κινητά συστήματα πρέπει να παρακολουθούν το περιβάλλον λειτουργίας τους και να προσαρμόζονται κατάλληλα στις αλλαγές που πραγματοποιούνται. Το σύνολο των παραγόντων που επηρεάζουν την λειτουργία μίας κινητής εφαρμογής ονομάζεται *context*, ενώ οι εφαρμογές που διαχειρίζονται το *context* ονομάζονται *context – aware* εφαρμογές. Ένα σημαντικό θέμα στην ανάπτυξη ρεαλιστικών *context – aware* συστημάτων είναι η διαχείριση της ποιότητας του *context*. Τα *context – aware* συστήματα πρέπει να μοντελοποιούν την αβεβαιότητα και την ασάφεια στο *context* και να χρησιμοποιούν τεχνικές λήψης αποφάσεων που διαχειρίζονται αυτά τα θέματα, ώστε να είναι περισσότερο αξιόπιστα, προσαρμόσιμα στις μεταβολές του περιβάλλοντος και στις απαιτήσεις των χρηστών, και να παρέχουν ποιοτικότερες υπηρεσίες στους χρήστες.

Σκοπός της παρούσας εργασίας είναι η ανάπτυξη μίας υπηρεσίας ενδιάμεσου λογισμικού για την υποστήριξη *context – aware* εφαρμογών σε κινητές συσκευές που λαμβάνει υπόψη τα χαρακτηριστικά της πληροφορίας του *context*. Το μοντέλο αναπαράστασης και διαχείρισης του *context* που χρησιμοποιεί η υπηρεσία βασίζεται στην ασαφή λογική. Η ασαφής λογική προσομοιώνει τον τρόπο που ο άνθρωπος αντιλαμβάνεται το *context*, ενώ ταυτόχρονα επιλύει θέματα που σχετίζονται με την αβεβαιότητα των δεδομένων. Η υπηρεσία ενδιάμεσου λογισμικού παρέχει την δυνατότητα ορισμού ενός ασαφούς μοντέλου για ένα *context – aware* σύστημα. Η αναπαράσταση του *context* γίνεται με βάση τα μοντέλα ασαφούς λογικής: *Tsukamoto*, *Mamdani* και *Sugeno*. Ο μηχανισμός λήψης αποφάσεων για την προσαρμογή των λειτουργιών της εφαρμογής στηρίζεται στις μεθόδους ασαφούς συλλογιστικής που παρέχουν

τα μοντέλα αυτά. Η επιλογή του μοντέλου γίνεται τόσο με βάση την γνώση για το *context*, όσο και με βάση τα χαρακτηριστικά της εκάστοτε εφαρμογής και της κινητής συσκευής όπου εκτελείται.

Όπως έδειξαν οι πειραματικές μετρήσεις, η χρήση των υπολογιστικών πόρων είναι ικανοποιητικά μικρή, ώστε να είναι εφικτή η λειτουργία της υπηρεσίας ενδιάμεσου λογισμικού σε κινητές συσκευές. Επιπλέον, η χρονική απόκριση της υπηρεσίας είναι αρκετά μικρή, ώστε να μην γίνεται αντιληπτή η λειτουργία της από τον χρήστη της εφαρμογής.

EXTENDED ABSTRACT IN ENGLISH

Georga, Eleni. MSc, Computer Science Department, University of Ioannina, Greece. June 2008. Development of Context Aware Middleware Based on Fuzzy Logic. Thesis Supervisor: Dimitrios I. Fotiadis.

The whole number of factors which affect the operation of a mobile system is called context and the applications which manage the context are called context – aware applications. The management of the quality of the context is important for the development of realistic context – aware systems. The context – aware systems should model the uncertainty and the fuzziness on the context, and use decision support techniques in order to be more reliable, meet the user's demands, and provide more qualitative services.

The objective of the present work is the development of a middleware to support context – aware applications on mobile devices. The architecture of the middleware and the provided services are based on a previous study [33]. In the present work we mainly study the problem of the management of the quality of the context. We analyze the specific characteristics of the information provided by the context, such as the uncertainty and the fuzziness, and we detect the supplementary demands that create those features for the middleware. According to the resulting specifications we designed and implemented the model of the representation and the management of the context which uses the middleware.

The model of the representation and the management of the context which is used by the middleware service is based on the fuzzy logic. The fuzzy logic simulates the way that man realizes the context and is concerned with issues related to the fuzziness of the data. The middleware service provides the capability of defining a fuzzy model for a context – aware system. The representation of the context is based on fuzzy logic models: Tsukamoto,

Mamdani and Sugeno. The reasoning mechanism is based on fuzzy logic techniques which are provided by those models. The model to be used in a prospective application is chosen according to the knowledge for the context and to the characteristics of the mobile application and the mobile device on which is performed.

As it was shown by experimental measurements the use of computational resources is sufficient to permit the operation of the middleware service on mobile devices. Additionally the response time of the service is as small as its operation not to be tangible by the user.

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ

1.1 Context – Aware Συστήματα

Τα κινητά συστήματα έχουν ένα σύνολο ιδιαίτερων χαρακτηριστικών, τα οποία τα διαφοροποιούν από τα σταθερά κατανεμημένα συστήματα. Οι κινητές συσκευές έχουν περιορισμένους υπολογιστικούς πόρους, όπως υπολογιστική ισχύ, μνήμη και μόνιμο αποθηκευτικό χώρο, ενώ η λειτουργία τους εξαρτάται απόλυτα από την μπαταρία τους. Επιπλέον, οι κινητές συσκευές επικοινωνούν μεταξύ τους ασύρματα, σχηματίζοντας δίκτυα με ακαθόριστη δομή και συχνές συνδέσεις και αποσυνδέσεις κόμβων. Το εύρος ζώνης στην επικοινωνία των κινητών συσκευών είναι περιορισμένο, ενώ η συνεχής διαθεσιμότητα του δικτύου δεν είναι δεδομένη. Τέλος, τα κινητά συστήματα λειτουργούν σε ένα εξαιρετικά δυναμικό *context*. Με τον όρο *context* εννοούμε οτιδήποτε μπορεί να επηρεάσει την συμπεριφορά μίας κινητής εφαρμογής, είτε αυτό αφορά πόρους της συσκευής, είτε αφορά πόρους εκτός συσκευής, όπως η θέση του χρήστη, το είδος της δικτυακής σύνδεσης, οι διαθέσιμες υπηρεσίες στο δίκτυο κ.α.

Οι εφαρμογές που λειτουργούν σε κινητά συστήματα πρέπει να προσαρμόζονται σε μεταβαλλόμενες συνθήκες. Για να το επιτύχουν αυτό, απαιτείται να έχουν γνώση του περιβάλλοντος στο οποίο λειτουργούν και των παραγόντων που επηρεάζουν την λειτουργία τους (*context*). Έτσι, οι κινητές εφαρμογές χρειάζεται να ενημερώνονται για τις αλλαγές που συμβαίνουν στο περιβάλλον εκτέλεσής τους και να προσαρμόζονται σύμφωνα με αυτές στις

αλλαγές, ώστε να παρέχουν ποιοτικότερες υπηρεσίες στους χρήστες. Οι εφαρμογές που διαχειρίζονται το *context*, ονομάζονται *context – aware*.

Για να αποκτήσει μία εφαρμογή *context – aware* χαρακτηριστικά, πρέπει να αναπτυχθεί πρόσθετο λογισμικό. Το πρόσθετο λογισμικό αφορά την εύρεση, την διαχείριση και την χρησιμοποίηση του *context* για την προσαρμογή των λειτουργιών της εφαρμογής. Το ζητούμενο είναι η σχεδίαση και η ανάπτυξη μίας κοινής πλατφόρμας, πάνω στην οποία να μπορούν εύκολα και ευέλικτα να χτιστούν *context – aware* εφαρμογές. Μία κοινή βάση ανάπτυξης εξασφαλίζει την επαναχρησιμοποίηση λογισμικού και ανεξαρτητοποιεί την εφαρμογή από το λογισμικό που αφορά την διαχείριση του *context*. Το λογισμικό υποστήριξης *context – aware* εφαρμογών παίζει το ρόλο ενδιάμεσου λογισμικού κάτω από την εφαρμογή και περιλαμβάνει ένα σύνολο έτοιμων και διαμορφώσιμων υπηρεσιών ή εργαλείων για την υλοποίηση μίας *context – aware* εφαρμογής.

Η ανάπτυξη ενδιάμεσου λογισμικού για *context – aware* εφαρμογές έχει προσελκύσει το ενδιαφέρον της ερευνητικής κοινότητας. Η κατάλληλη αρχιτεκτονική, το μοντέλο αναπαράστασης και διαχείρισης του *context*, οι υπηρεσίες που παρέχονται αλλά και ο τρόπος αλληλεπίδρασης με το επίπεδο εφαρμογής είναι τα βασικά θέματα έρευνας.

Στην συνέχεια του κειμένου, με τον όρο εφαρμογή θα αναφερόμαστε στο λογισμικό που τρέχει στο επίπεδο εφαρμογής στην κινητή συσκευή. Με τον όρο σύστημα θα αναφερόμαστε σε όλο το σύστημα ενδιάμεσου λογισμικού – εφαρμογής, καθώς και των υπόλοιπων συστατικών, υλικού ή λογισμικού, που σχετίζονται με αυτά.

1.2 Αντικείμενο της Μεταπτυχιακής Εργασίας

Σκοπός της παρούσας εργασίας είναι η ανάπτυξη ενός ενδιάμεσου λογισμικού για την υποστήριξη *context – aware* εφαρμογών σε κινητές συσκευές. Η αρχιτεκτονική του ενδιάμεσου λογισμικού και οι υπηρεσίες που παρέχει βασίζονται στην εργασία [33]. Στην παρούσα εργασία, μελετάμε, κυρίως, το πρόβλημα της διαχείρισης της ποιότητας του *context*. Αναλύουμε τα ιδιαίτερα χαρακτηριστικά της πληροφορίας του *context*, όπως η αβεβαιότητα και η ασάφεια, και εντοπίζουμε τις επιπλέον απαιτήσεις που δημιουργούν τα χαρακτηριστικά αυτά για το ενδιάμεσο λογισμικό. Με βάση τις προδιαγραφές που προέκυψαν, σχεδιάσαμε

και υλοποιήσαμε το μοντέλο αναπαράστασης και διαχείρισης του *context* που χρησιμοποιεί το ενδιάμεσο λογισμικό.

Η μοντελοποίηση του *context* που προτείνουμε βασίζεται στην ασαφή λογική. Η ασαφής λογική προσομοιώνει την ανθρώπινη θεώρηση για το *context*, ενώ ταυτόχρονα επιλύει θέματα που σχετίζονται με την αβεβαιότητα των δεδομένων. Συγκεκριμένα, το ενδιάμεσο λογισμικό που αναπτύξαμε παρέχει την δυνατότητα αναπαράστασης του *context* με βάση τα μοντέλα ασαφούς λογικής *Tsukamoto*, *Mamdani* και *Sugeno*. Τα μοντέλα αυτά θα περιγραφούν λεπτομερώς στο Κεφάλαιο 4. Ο μηχανισμός λήψης αποφάσεων που χρησιμοποιεί το ενδιάμεσο λογισμικό, για την προσαρμογή των λειτουργιών της εφαρμογής, στηρίζεται στις μεθόδους ασαφούς συλλογιστικής που παρέχουν τα μοντέλα αυτά. Η επιλογή του κατάλληλου μοντέλου εξαρτάται από τα χαρακτηριστικά του *context*, της εφαρμογής και της συσκευής.

1.3 Διάρθρωση της Μεταπτυχιακής Εργασίας

Στο Κεφάλαιο 2 αναφέρονται βασικές έννοιες και ορισμοί και περιγράφονται τα βασικότερα θέματα που αφορούν την ανάπτυξη *context – aware* συστημάτων. Επίσης, περιγράφονται τα λειτουργικά χαρακτηριστικά του ενδιάμεσου λογισμικού για *context – aware* εφαρμογές και αναφέρονται, ενδεικτικά, κάποια *context – aware* συστήματα ενδιάμεσου λογισμικού. Στο Κεφάλαιο 3 παρουσιάζεται το ενδιάμεσο λογισμικό που σχεδιάστηκε και αναπτύχθηκε στην εργασία [33]. Στο Κεφάλαιο 4 ορίζονται τα βασικά χαρακτηριστικά της πληροφορίας του *context* και περιγράφονται οι πιο χαρακτηριστικές εργασίες που ασχολούνται με θέματα ποιότητας του *context*. Επίσης, γίνεται μία εκτενής παρουσίαση της θεωρίας των ασαφών συνόλων και της ασαφούς λογικής. Στο Κεφάλαιο 5 περιγράφεται το μοντέλο αναπαράστασης και διαχείρισης του *context* που αναπτύξαμε. Στο Κεφάλαιο 6 αναφέρονται θέματα υλοποίησης και περιγράφεται η κινητή εφαρμογή που αναπτύχθηκε πάνω από το ενδιάμεσο λογισμικό. Στο Κεφάλαιο 7 παρουσιάζονται τα πειράματα που έγιναν και γίνεται σχολιασμός των αποτελεσμάτων. Τέλος, στο Κεφάλαιο 8 συνοψίζονται τα συμπεράσματα και παρουσιάζονται μελλοντικές επεκτάσεις της παρούσας έρευνας.

ΚΕΦΑΛΑΙΟ 2. CONTEXT – AWARE ΣΥΣΤΗΜΑΤΑ

Σε αυτό το κεφάλαιο αναφέρονται βασικές έννοιες και ορισμοί και περιγράφονται τα βασικότερα θέματα που αφορούν την ανάπτυξη *context – aware* συστημάτων. Επίσης, περιγράφονται τα λειτουργικά χαρακτηριστικά του ενδιάμεσου λογισμικού για *context – aware* εφαρμογές και αναφέρονται, ενδεικτικά, κάποια *context – aware* συστήματα ενδιάμεσου λογισμικού.

2.1 Η Έννοια του Context

Το *context* συνδέεται με πολλά πεδία της επιστήμης των υπολογιστών. Στην εργασία αυτή επικεντρωνόμαστε στο *context*, όπως αυτό χρησιμοποιείται στον τομέα των κινητών υπολογιστικών συστημάτων.

2.1.1 Ορισμός του Context

Στην βιβλιογραφία αναφέρονται αρκετοί ορισμοί για το *context*. Πολλοί ερευνητές προσπάθησαν να ορίσουν το *context* περιγραφικά μέσω παραδειγμάτων ή με την χρήση συνωνύμων, ενώ άλλοι χρησιμοποιώντας τυπικούς ορισμούς. Ένας από τους πιο ακριβείς ορισμούς δόθηκε από τους *Dey* και *Abowd* [13]. Αυτοί αναφέρονται στο *context* ως «κάθε πληροφορία που μπορεί να χρησιμοποιηθεί για να χαρακτηρίσει την κατάσταση των

οντοτήτων που θεωρούνται σχετικές με την αλληλεπίδραση ενός χρήστη και μίας εφαρμογής, συμπεριλαμβανομένου του χρήστη και της εφαρμογής».

2.1.2 Κατηγορίες Context

Η πληροφορία του *context* κατηγοριοποιείται μέσω της διάκρισης διαφόρων διαστάσεων. Ο *Schilit* [43] κατηγοριοποιεί το *context* σε τρεις κλάσεις με βάση την πηγή προέλευσης των παραμέτρων του:

Context Συστήματος

Κάθε κινητή εφαρμογή πρέπει να είναι ενήμερη για το *context* τόσο του υπολογιστικού συστήματος στο οποίο εκτελείται, όσο και του συστήματος επικοινωνίας που χρησιμοποιεί. Έτσι, το *context* του συστήματος περιλαμβάνει δεδομένα που σχετίζονται με τα χαρακτηριστικά της συσκευής (π.χ. μπαταρία, μνήμη, ταχύτητα *CPU*, αποθηκευτικός χώρος), την ασύρματη δικτυακή σύνδεση (π.χ. είδος σύνδεσης, εύρος ζώνης), τους υπολογιστικούς πόρους στο περιβάλλον λειτουργίας της (π.χ. εκτυπωτές, τερματικά, υπηρεσίες, χρήστες), το επίπεδο εφαρμογής κ.α.

Context Χρήστη

Αναφέρεται σε κάθε είδος πληροφορίας που σχετίζεται με τον χρήστη και τον χαρακτηρίζει. Το *context* χρήστη μπορεί να είναι η ηλικία του, η θέση του, το ιατρικό του ιστορικό, κλπ. Επίσης, βιομετρικά δεδομένα όπως τα δακτυλικά αποτυπώματα, η μορφή της ίριδας ή του προσώπου, αποτελούν μέρος του συγκεκριμένου *context*. Το *context* χρήστη συνδέεται και με την συναισθηματική του κατάσταση, παρόλο που είναι δύσκολη η απόκτηση αυτού του είδους πληροφορίας στα υπολογιστικά συστήματα. Γενικά, ως *context* χρήστη θεωρείται κάθε στοιχείο που αφορά τις δραστηριότητες του χρήστη, στην κοινωνική, προσωπική και χωροχρονική του κατάσταση.

Φυσικό Context

Περιλαμβάνει πληροφορία σχετική με το φυσικό περιβάλλον, η οποία δεν καλύπτεται από το *context* του συστήματος και του χρήστη. Η διάσταση αυτή αφορά τα στοιχεία που μπορούν να μετρηθούν μέσω αισθητήρων (π.χ. φωτεινότητα, θερμοκρασία, κ.α.).

Το *context* παρουσιάζει αρκετά χρονικά χαρακτηριστικά. Στην εργασία [12] επισημαίνεται ότι ο χρόνος είναι μία σημαντική παράμετρος *context* για τις κινητές εφαρμογές. Για τον λόγο αυτό, στην παραπάνω κατηγοριοποίηση προστίθεται και η χρονική πληροφορία (π.χ. ώρα, ήμερα, ημερομηνία).

Με βάση την χρονική διάσταση το *context* χαρακτηρίζεται ως στατικό και δυναμικό [29]. Το στατικό *context* περιγράφει όλα τα στοιχεία ενός κινητού συστήματος που διατηρούνται σταθερά, όπως για παράδειγμα το είδος της κινητής συσκευής που χρησιμοποιείται. Λόγω της μεταβαλλόμενης δομής των κινητών συστημάτων, μεγάλη πλειοψηφία της πληροφορίας σε αυτά είναι δυναμική.

Μία επιπλέον διάκριση του *context* γίνεται με βάση τον τρόπο που το διαχειρίζεται η εφαρμογή. Διακρίνονται δύο κατηγορίες του *context*, όπως αυτό χρησιμοποιείται στα κινητά συστήματα [12]. Η μια κατηγορία του *context* περιλαμβάνει κάθε πληροφορία που καθορίζει και επηρεάζει την συμπεριφορά μίας κινητής εφαρμογής. Η άλλη κατηγορία του *context* είναι σχετική με την εφαρμογή, αλλά όχι κρίσιμη για την λειτουργία της. Έτσι, το *context* διακρίνεται σε ενεργό και παθητικό, αντίστοιχα. Στο ενεργό *context*, οι υπηρεσίες της εφαρμογής παρέχονται στον χρήστη προσαρμοσμένες στο τρέχον *context* ή συγκεκριμένο *context* μπορεί αυτόματα να οδηγήσει στην εκτέλεση κάποιων ενεργειών. Στο παθητικό *context*, η εφαρμογή είτε παρουσιάζει το *context* στον χρήστη, είτε το αποθηκεύει, ώστε να χρησιμοποιηθεί από τις υπηρεσίες της στο μέλλον.

2.1.3 Ιδιότητες του Context

Ωστόσο, η κατηγοριοποίηση του *context* δεν είναι επαρκής για την πλήρη κατανόηση αυτού ως έννοια και για την αποδοτική χρήση του στα κινητά συστήματα. Η πληροφορία του *context* έχει ένα σύνολο ιδιαίτερων χαρακτηριστικών που επηρεάζουν τον σχεδιασμό του μοντέλου διαχείρισης του *context* σε ένα σύστημα [29].

Ο χρόνος είναι ένα βασικό γνώρισμα που θα πρέπει να συμπεριληφθεί στο μοντέλο αναπαράστασης και αποθήκευσης του *context*. Αυτό γιατί, το *context* παρουσιάζει αρκετές χρονικές εξαρτήσεις. Δηλαδή, οι παλιές τιμές του *context* μπορεί να είναι χρήσιμες για την

εξαγωγή συμπερασμάτων ή την αποτίμηση του τρέχοντος *context*. Επίσης, η περιγραφή του *context* σε πολλές περιπτώσεις περιέχει καταστάσεις που ίσχυσαν ή που θα ισχύσουν.

Ένα δεύτερο χαρακτηριστικό του *context* είναι ότι πρόκειται για ατελή πληροφορία. Συγκεκριμένα, η πληροφορία του *context* είναι λανθασμένη αν αποτυγχάνει στο να απεικονίσει την αληθινή κατάσταση των οντοτήτων που μοντελοποιεί, ασυνεπής όταν περιέχει αντιφατικά στοιχεία και ελλιπής όταν κάποια στοιχεία δεν είναι γνωστά. Τα παραπάνω προβλήματα οφείλονται σε αρκετούς παράγοντες. Αρχικά, λόγω του ότι τα κινητά περιβάλλοντα υπολογισμού είναι αρκετά δυναμικά, η πληροφορία που τα περιγράφει μπορεί να γίνει ασυνεπής χρονικά. Επίσης, οι προμηθευτές του *context*, όπως οι αισθητήρες, οι αλγόριθμοι παραγωγής δεδομένων και οι χρήστες, μπορεί να παράγουν λανθασμένα δεδομένα. Τέλος, είναι πιθανή η απώλεια δεδομένων, λόγω των ασταθών συνδέσεων. Το θέμα της ατέλειας του *context* και η αντιμετώπιση του προβλήματος περιγράφονται στο Κεφάλαιο 4.

Το είδος του *context* μπορεί να κυμαίνεται από χαμηλού εννοιολογικού επιπέδου μέχρι υψηλότερου επιπέδου *context* με σημασιολογική πληροφορία. Το μεγαλύτερο ποσοστό πληροφορίας στα κινητά συστήματα παράγεται από αισθητήρες. Συνήθως, υπάρχει μεγάλο χάσμα μεταξύ της εξόδου των αισθητήρων και του επιπέδου πληροφορίας που είναι χρήσιμη στις εφαρμογές. Αυτό το χάσμα μπορεί να γεφυρωθεί μέσω διαφόρων τρόπων επεξεργασίας του *context*. Επίσης, οι απαιτήσεις διαφοροποιούνται μεταξύ διαφορετικών εφαρμογών. Έτσι, το μοντέλο του *context* πρέπει να υποστηρίζει πολλαπλές αναπαραστάσεις της ίδιας πληροφορίας σε διαφορετικές μορφές και σε διαφορετικά επίπεδα αφαίρεσης και να είναι δυνατό να βρίσκει την σχέση μεταξύ των διαφόρων αναπαραστάσεων.

Τέλος, υπάρχει αλληλοσυσχέτιση μεταξύ των παραμέτρων του *context*. Οι παράμετροι του *context* συνδέονται συνήθως μέσω κανόνων που περιγράφουν την παραγωγή πληροφορίας από άλλη πληροφορία. Δηλαδή, πληροφορία που παρέχεται από διαφορετικές πηγές μπορεί να συνδυαστεί και να εξαχθούν συμπεράσματα για το *context*. Επίσης, πληροφορία που αφορά διάφορες οντότητες μπορεί να συσχετιστεί για την σύνθεση επιπλέον πληροφορίας. Αυτό το είδος σχέσης, όπου τα χαρακτηριστικά της παραγόμενων δεδομένων συνδέονται στενά με τις ιδιότητες των δεδομένων από τα οποία προκύπτουν, αναφέρεται ως εξάρτηση.

2.2 Context-Aware Εφαρμογές

Σε αντίθεση με τα σταθερά κατανεμημένα συστήματα, τα κινητά συστήματα λειτουργούν σε ένα εξαιρετικά δυναμικό *context*. Σύμφωνα με τα παραπάνω, με τον όρο *context* εννοούμε οτιδήποτε μπορεί να επηρεάσει την συμπεριφορά μιας εφαρμογής, είτε αυτό αφορά πόρους της συσκευής, όπως μνήμη, μπαταρία, μέγεθος οθόνης, είτε αυτό αφορά πόρους εκτός συσκευής, όπως τοποθεσία, είδος δικτυακής σύνδεσης και κατανεμημένες διαθέσιμες υπηρεσίες στο δίκτυο. Οι κινητές εφαρμογές χρειάζεται να ενημερώνονται για τις αλλαγές που συμβαίνουν στο περιβάλλον εκτέλεσής τους και να προσαρμόζονται σύμφωνα με αυτές τις αλλαγές, ώστε να παρέχουν ποιοτικότερες υπηρεσίες στους χρήστες. Οι εφαρμογές που διαχειρίζονται το *context*, ονομάζονται *context – aware* εφαρμογές.

2.2.1 Ορισμός των Context – Aware Εφαρμογών

Ο όρος *context – aware* χρησιμοποιήθηκε για πρώτη ορά από τους *Schilit* και *Theimer* το 1994 [44] για την περιγραφή των εφαρμογών που «προσαρμόζονται σύμφωνα με την θέση του χρήστη, την ύπαρξη ατόμων και αντικειμένων στο περιβάλλον λειτουργίας τους και με βάση τις αλλαγές σε αυτά τα αντικείμενα στον χρόνο». Από τότε δόθηκαν αρκετοί ορισμοί για τον όρο *context – aware* σε σχέση πάντοτε με κινητά περιβάλλοντα υπολογισμού. Όμως, οι περισσότεροι από αυτούς ήταν αρκετά ειδικοί και στηρίζονταν στην λειτουργικότητα συγκεκριμένων εφαρμογών που έκαναν χρήση του *context*.

Οι *Dey* και *Abowd* [13] προτείνουν ότι «ένα σύστημα είναι *context – aware* εάν χρησιμοποιεί το *context* για να παρέχει σχετική πληροφορία ή υπηρεσίες στον χρήστη». Ο ορισμός αυτός είναι αρκετά γενικός και μπορεί εύκολα να εφαρμοστεί για τον χαρακτηρισμό μίας εφαρμογής. Επιπλέον, είναι ο πρώτος ορισμός που θεωρεί ότι μία *context – aware* εφαρμογή μόνο αποκρίνεται στο *context*, ενώ η απόκτηση και η επεξεργασία του αφορούν άλλες υπολογιστικές οντότητες.

2.2.2 Ανάπτυξη Context – Aware Εφαρμογών

Η πληροφορία του *context* έχει κάποια ιδιαίτερα χαρακτηριστικά, όπως οι χρονικές εξαρτήσεις, η έλλειψη αξιοπιστίας, η απαίτηση για διαφορετικές αναπαραστάσεις και η αλληλεξάρτηση των παραμέτρων της, που κάνουν δύσκολο το χειρισμό της [29]. Το

αποτέλεσμα είναι η επεξεργασία του *context* να αποτελεί σύνθετη διαδικασία, που είναι δύσκολο να υλοποιηθεί σε ένα υπολογιστικό σύστημα. Επιπλέον απαιτείται ένα σύνολο υποσυστημάτων ανεξάρτητων της εφαρμογής που υλοποιούν τις διαδικασίες συλλογής και διαχείρισης του *context*. Η ανάπτυξη αυτών των υποσυστημάτων για κάθε εφαρμογή είναι μία δύσκολη και χρονοβόρα διαδικασία. Για τον λόγο αυτό, είναι επιθυμητό η υλοποίηση των παραπάνω λειτουργιών να γίνεται ανεξάρτητα από την εφαρμογή. Επίσης, απαιτείται να υπάρχει διαλειτουργικότητα στις *context – aware* εφαρμογές. Αυτή η ανάγκη δημιουργεί την απαίτηση για ένα κοινό λογισμικό που βοηθάει στην ανάπτυξη και υποστηρίζει *context – aware* εφαρμογές.

Ειδικότερα, για την ανάπτυξη μίας *context – aware* εφαρμογής, απαιτείται να αντιμετωπιστούν τα ακόλουθα θέματα. Αρχικά, χρειάζεται να αναπτυχθούν μηχανισμοί επικοινωνίας με τους προμηθευτές για την συλλογή της πληροφορίας του *context*. Το είδος επικοινωνίας μπορεί να διαφέρει σημαντικά μεταξύ διαφόρων προμηθευτών (π.χ. η πληροφορία για την τοποθεσία του χρήστη συλλέγεται με χρήση *GPS* για εξωτερικούς χώρους, ενώ για εσωτερικούς χώρους με χρήση υπέρυθρων ή ραδιοσυχνοτήτων). Επίσης, χρειάζεται επεξεργασία της πληροφορίας ώστε να μεταφραστεί σωστά και να χρησιμοποιηθεί από την εκάστοτε εφαρμογή. Στην συνέχεια, η συμπεριφορά της εφαρμογής πρέπει να διαμορφωθεί με βάση το διαθέσιμο *context*. Αυτό συνεπάγεται την ύπαρξη μηχανισμών ανίχνευσης των αλλαγών που είναι κρίσιμες για την λειτουργία της εφαρμογής και μηχανισμών εκτέλεσης των διαδικασιών προσαρμογής της εφαρμογής. Ωστόσο, αυτοί οι μηχανισμοί δεν είναι δυνατό να βασίζονται σε μία στατική αντιστοίχιση σεναρίων *context* και κατάλληλων ενεργειών προσαρμογής. Αυτό διότι, τόσο οι σχεδιαστές των εφαρμογών δεν είναι δυνατό να προβλέψουν όλα τα πιθανά περιβάλλοντα εκτέλεσης της εφαρμογής, αλλά και διότι οι ανάγκες των χρηστών μεταβάλλονται συχνά.

Η κατασκευή και υποστήριξη *context – aware* εφαρμογών θα ήταν μία δύσκολη διαδικασία, αν οι σχεδιαστές είχαν να αντιμετωπίσουν όλα αυτά τα θέματα. Για αυτό τον λόγο οι ερευνητές οδηγήθηκαν σε μία αρχιτεκτονική προσέγγιση όπου, η διαχείριση του *context* γίνεται από ενδιάμεσο λογισμικό. Το ενδιάμεσο λογισμικό για *context – aware* εφαρμογές περιλαμβάνει ένα σύνολο έτοιμων και διαμορφώσιμων υπηρεσιών ή εργαλείων, που μπορεί να χρησιμοποιήσει ο προγραμματιστής για την υλοποίηση της εφαρμογής.

Το ενδιάμεσο λογισμικό για την ανάπτυξη *context – aware* εφαρμογών συμβάλλει στην επίλυση προβλημάτων ανομοιογένειας των προμηθευτών του *context*, στην επεξεργασία του context και στον εύκολο καθορισμό του κρίσιμου *context* για την λειτουργία της εφαρμογής. Επίσης, παρέχει μηχανισμούς για δυναμικό καθορισμό των στρατηγικών προσαρμογής της εφαρμογής ανάλογα με τις αλλαγές στο *context*, με βάση τις τρέχουσες ανάγκες των χρηστών. Δηλαδή, το ενδιάμεσο λογισμικό είναι αυτό που αλληλεπιδρά με τους προμηθευτές, συλλέγει την πληροφορία του *context*, την επεξεργάζεται σύμφωνα με τις απαιτήσεις της εφαρμογής, ελέγχει αν οι αλλαγές στο *context* επηρεάζουν την συμπεριφορά της εφαρμογής και εκτελεί τις ενέργειες προσαρμογής που απαιτούνται.

2.3 Ενδιάμεσο Λογισμικό

Το ενδιάμεσο λογισμικό χρησιμοποιείται στην ανάπτυξη κατανεμημένων συστημάτων. Είναι ένα επίπεδο λογισμικού, το οποίο βρίσκεται μεταξύ του λειτουργικού συστήματος και της κατανεμημένης εφαρμογής και καθιστά ικανή την διαφανή ενοποίηση κατανεμημένων αντικειμένων [2]. Το ενδιάμεσο λογισμικό διαχειρίζεται την επικοινωνία μεταξύ των συστατικών ενός κατανεμημένου συστήματος παρέχοντας στους σχεδιαστές εφαρμογών κατάλληλες αφαιρέσεις που κρύβουν την πολυπλοκότητα των κατανεμημένων συστημάτων. Για παράδειγμα, οι προγραμματιστές δεν χρειάζεται να γνωρίζουν την θέση μίας απομακρυσμένης υπηρεσίας για να πραγματοποιήσουν κλήση σε αυτή, ούτε χρειάζεται να φροντίσουν για ενδεχόμενες αποτυχίες στο δίκτυο. Δηλαδή, το ενδιάμεσο λογισμικό κάνει την κατανομή διαφανή ως προς τους σχεδιαστές και τους χρήστες παρουσιάζοντας το κατανεμημένο σύστημα ως μία ενοποιημένη μονάδα [18].

2.3.1 Ενδιάμεσο Λογισμικό για Κινητά Συστήματα

Οι ήδη υπάρχουσες τεχνολογίες ενδιάμεσου λογισμικού έχουν σχεδιαστεί για την υποστήριξη εφαρμογών που εκτελούνται σε στατικά κατανεμημένα συστήματα. Όμως, έχουν περιορισμένη εφαρμογή στα κινητά συστήματα. Για τον προσδιορισμό των απαιτήσεων των συστημάτων ενδιάμεσου λογισμικού για κινητές εφαρμογές, χρησιμοποιείται ένα μοντέλο αναφοράς που διαχωρίζει τα συστήματα αυτά ως προς το είδος επικοινωνίας που υποστηρίζουν, τον τρόπο που χρησιμοποιούν το *context* και το υπολογιστικό κόστος που απαιτούν για να εκτελεστούν [6].

Αρχικά, τα παραδοσιακά κατανεμημένα συστήματα χαρακτηρίζονται από μία σταθερή δικτυακή υποδομή, όπου οι χρήστες είναι μονίμως συνδεδεμένοι στο δίκτυο μέσω συνδέσεων μεγάλου εύρους ζώνης. Αυτή η υποδομή ευνοεί σύγχρονους μηχανισμούς επικοινωνίας, όπως αιτήσεις αντικειμένων και απομακρυσμένες κλήσεις διαδικασιών. Ωστόσο, οι κινητές συσκευές επικοινωνούν μεταξύ τους ασύρματα, σχηματίζοντας δίκτυα με ακαθόριστη δομή και συχνές συνδέσεις και αποσυνδέσεις κόμβων. Το εύρος ζώνης είναι περιορισμένο, ενώ η συνεχής διαθεσιμότητα του δικτύου δεν είναι δεδομένη. Έτσι, το ενδιάμεσο λογισμικό για κινητές εφαρμογές χρειάζεται να χρησιμοποιεί ασύγχρονους μηχανισμούς επικοινωνίας.

Επιπλέον, η αρχή της διαφάνειας που χρησιμοποιείται στο ενδιάμεσο λογισμικό για στατικά κατανεμημένα συστήματα δεν μπορεί να εφαρμοστεί σε κινητές εφαρμογές. Τα κινητά συστήματα λειτουργούν σε ένα εξαιρετικά δυναμικό περιβάλλον. Επομένως, δεν είναι δυνατό οι σχεδιαστές να προβλέψουν εκ των προτέρων όλα τα δυνατά περιβάλλοντα εκτέλεσης και να ενημερώσουν το ενδιάμεσο λογισμικό για το πώς θα συμπεριφερθεί σε κάθε κατάσταση. Από την στιγμή που δεν υπάρχει στατική γνώση, το ενδιάμεσο λογισμικό δεν μπορεί να λαμβάνει αποφάσεις ανεξάρτητα από την εφαρμογή και ταυτόχρονα να εγγυάται την καλύτερη ποιότητα υπηρεσιών. Αντιθέτως, το ενδιάμεσο λογισμικό πρέπει να αλληλεπιδρά με το επίπεδο εφαρμογής, να ενημερώνει την εφαρμογή για το τρέχον *context* και να προσαρμόζει την λειτουργία του χρησιμοποιώντας την πληροφορία που του δίνει η εφαρμογή.

Τέλος, οι κινητές εφαρμογές λειτουργούν σε συσκευές με περιορισμένους υπολογιστικούς πόρους. Έτσι, δεν μπορούν να υποστηρίξουν συστήματα ενδιάμεσου λογισμικού που έχουν υψηλό υπολογιστικό κόστος. Το υπολογιστικό κόστος εξαρτάται από το σύνολο των μη-λειτουργικών απαιτήσεων που πρέπει να ικανοποιεί το ενδιάμεσο λογισμικό. Για τον λόγο αυτό, είναι απαραίτητο να βρεθεί μία μέση λύση, μεταξύ των υπηρεσιών που παρέχει το ενδιάμεσο λογισμικό στην εφαρμογή και στην ποσότητα των πόρων που χρησιμοποιεί.

Εφόσον οι απαιτήσεις για την ανάπτυξη κινητών εφαρμογών είναι σημαντικά διαφορετικές από αυτές που επιβάλλουν οι στατικά κατανεμημένες εφαρμογές, οι ερευνητές οδηγήθηκαν στην σχεδίαση συστημάτων ενδιάμεσου λογισμικού αποκλειστικά για κινητά συστήματα. Οι λύσεις που έχουν αναπτυχθεί μέχρι σήμερα διακρίνονται σε τρεις κατηγορίες, η κάθε μια από τις οποίες αντιστοιχεί σε μία προσπάθεια ικανοποίησης μίας εκ των τριών απαιτήσεων που

αναφέρθηκαν παραπάνω [6]. Συγκεκριμένα, οι λύσεις που στηρίζονται στην ιδιότητα της ανάκλασης [7] αποτελούν μία προσπάθεια για δημιουργία ενδιάμεσου λογισμικού που χρειάζεται λίγους υπολογιστικούς πόρους και που διαμορφώνει αυτόματα την εσωτερική του δομή με βάση το *context*. Τα *TupleSpace* συστήματα αντιμετωπίζουν το πρόβλημα της μη-συνεχούς διαθεσιμότητας του δικτύου υποστηρίζοντας ασύγχρονη επικοινωνία. Τέλος, το ενδιάμεσο λογισμικό για *context – aware* εφαρμογές παρέχει στις εφαρμογές ενημερότητα για το *context*.

2.4 Ενδιάμεσο Λογισμικό για Context – Aware Εφαρμογές

Οι υπηρεσίες διαχείρισης του *context* σε ένα *context – aware* ενδιάμεσο λογισμικό μπορεί να εκτελούνται στην κινητή συσκευή που λειτουργεί η εφαρμογή ή κεντρικά σε στατικούς εξυπηρέτες με αυξημένες υπολογιστικές δυνατότητες [9]. Στην πρώτη περίπτωση η απόκτηση και η διαχείριση του *context*, καθώς και οι διαδικασίες προσαρμογής της εφαρμογής στις αλλαγές του *context* εκτελούνται από το ενδιάμεσο λογισμικό στην κινητή συσκευή. Στην δεύτερη περίπτωση ένας ή περισσότεροι κεντρικοί εξυπηρέτες παρέχουν ενημερότητα για το *context* σε διάφορες εφαρμογές σε ένα κατανεμημένο περιβάλλον. Η αρχιτεκτονική πελάτη – εξυπηρέτη δεν επιφέρει επιπρόσθετο φόρτο στην κινητή συσκευή, εφόσον το ενδιάμεσο λογισμικό δεν δεσμεύει υπολογιστικούς πόρους της συσκευής για την παροχή των υπηρεσιών του. Όμως, οι δυνατότητες κλιμάκωσης είναι περιορισμένες και δεν είναι ανεκτική σε σφάλματα, εφόσον εξαρτάται από ένα κεντρικό συστατικό.

Σχήμα 2.1: Αφαιρετική αρχιτεκτονική ενός context – aware συστήματος.

Τα *context – aware* συστήματα ενδιάμεσου λογισμικού που έχουν προταθεί τα τελευταία χρόνια εμφανίζουν κάποια κοινά λειτουργικά χαρακτηριστικά. Στο Σχήμα 2.1 φαίνεται αφαιρετικά η αρχιτεκτονική ενός *context – aware* συστήματος. Η αρχιτεκτονική κατανέμει στα τρία επίπεδα, μεταξύ των προμηθευτών και της εφαρμογής, την λειτουργικότητα του ενδιάμεσου λογισμικού [1, 15]. Το βασικό στοιχείο της αρχιτεκτονικής αυτής είναι ο διαχωρισμός των διαδικασιών συλλογής και διαχείρισης του *context* από την χρήση αυτού στο επίπεδο εφαρμογής, με σκοπό την βελτίωση της επεκτασιμότητας και επαναχρησιμοποίησης του συστήματος.

Το πρώτο επίπεδο αποτελείται από μία συλλογή από διαφορετικούς προμηθευτές *context*. Οι προμηθευτές μπορεί να είναι οποιαδήποτε πηγή υλικού ή λογισμικού που παρέχει τιμές για τις παραμέτρους του *context*. Στην βιβλιογραφία [4] χρησιμοποιείται ο όρος αισθητήρας για όλους τους τύπους προμηθευτών, από απλές συσκευές υλικού μέχρι σύνθετες υπολογιστικές μονάδες.

Στο δεύτερο επίπεδο γίνεται η συλλογή της πληροφορίας. Στο επίπεδο αυτό χρησιμοποιούνται κατάλληλα προγράμματα για την επικοινωνία με τους προμηθευτές. Το λογισμικό επικοινωνίας με τους προμηθευτές υλοποιείται με επαναχρησιμοποιήσιμα τμήματα που κρύβουν τα τεχνικά χαρακτηριστικά των προμηθευτών και αποκτούν την πληροφορία με διαφάνεια προς τα ανώτερα επίπεδα.

Το επίπεδο επεξεργασίας του συλλεγόμενου *context* από τους προμηθευτές δεν υλοποιείται σε κάθε *context – aware* σύστημα, αλλά παρέχει χρήσιμη πληροφορία για εφαρμογές που απαιτούν υψηλού εννοιολογικού επιπέδου πληροφορία. Το επίπεδο επεξεργασίας είναι υπεύθυνο για την ερμηνεία / μετάφραση του *context* και την εξαγωγή συμπερασμάτων και νέας πληροφορίας συνδυάζοντας πληροφορία από διαφορετικούς προμηθευτές ή πληροφορία που αφορά διαφορετικές οντότητες. Για την διαχείριση του *context* απαιτούνται είτε μηχανισμοί, όπως κανόνες, γραμμένοι σε διάφορους τύπους λογικής (π.χ. λογική πρώτης τάξης, χρονική και ασαφής λογική κ.α.), είτε μηχανισμοί εκπαίδευσης (π.χ. μπεϋζιανα ή νευρωνικά δίκτυα κ.α.). Επίσης στο επίπεδο αυτό διευθετούνται προβλήματα που σχετίζονται με την ασάφεια και την ανακρίβεια της πληροφορίας που παράγουν οι προμηθευτές καθώς και με την ασφάλεια και εμπιστευτικότητα του *context*.

Το επίπεδο αποθήκευσης οργανώνει και αποθηκεύει το *context* με βάση το μοντέλο αναπαράστασης που χρησιμοποιείται. Επίσης υλοποιεί διεπαφή για την παροχή της πληροφορίας στο επίπεδο εφαρμογής. Η πρόσβαση της εφαρμογής στα δεδομένα μπορεί να γίνει με δύο τρόπους, σύγχρονα και ασύγχρονα. Στην σύγχρονη επικοινωνία, η εφαρμογή απευθύνει ερωτήσεις για τα δεδομένα που την αφορούν και παραμένει ανενεργή μέχρι να λάβει απάντηση. Η ασύγχρονη επικοινωνία γίνεται μέσω εγγραφών και ειδοποιήσεων. Η εφαρμογή εγγράφεται στην λίστα ειδοποίησης για κάποια παράμετρο του *context* και όταν συμβεί κάποιο γεγονός ειδοποιείται, λαμβάνοντας την νέα τιμή. Στις περισσότερες περιπτώσεις η ασύγχρονη επικοινωνία είναι πιο κατάλληλη, λόγω των συχνών αλλαγών στις τιμές του *context*. Η σύγχρονη τεχνική έχει μεγαλύτερες απαιτήσεις σε υπολογιστικούς πόρους καθώς η εφαρμογή χρειάζεται να υποβάλλει συχνά ερωτήσεις για το *context* και να ανιχνεύει για αλλαγές σε αυτό, μέσω κάποιου είδους ιστορικού.

Η εφαρμογή βρίσκεται στο πέμπτο επίπεδο. Στο επίπεδο αυτό υλοποιείται ο τρόπος που η εφαρμογή αντιδρά στις αλλαγές του *context* και η χρήση της πληροφορίας από τις υπηρεσίες της εφαρμογής.

Τα υπάρχοντα συστήματα ενδιάμεσου λογισμικού για *context – aware* εφαρμογές χρησιμοποιούν διάφορες προσεγγίσεις για την συνολική διαχείριση της πληροφορίας του *context*. Όλα τα συστήματα παρέχουν μεθόδους για την συλλογή του *context* και για την παροχή του στο επίπεδο της εφαρμογής. Ωστόσο, στο σύνολό τους έχουν αρκετά διαφορετικά λειτουργικά και τεχνικά χαρακτηριστικά.

Παρουσιάζουμε κάποιες αντιπροσωπευτικές υλοποιήσεις συστημάτων ενδιάμεσου λογισμικού και αναλύουμε τις προσεγγίσεις που χρησιμοποιούνται στο κάθε σύστημα. Εστιάζουμε σε συστήματα που καλύπτουν πολλαπλά επίπεδα της αρχιτεκτονικής που περιγράψαμε σε αυτή την ενότητα.

Context Toolkit

Θεμελιώδης για την έρευνα στα *context – aware* συστήματα θεωρείται η εργασία του *Dey* από το *GeorgiaTech*. Ο *Dey* στο πλαίσιο της διδακτορικής του διατριβής πρότεινε μία γενική αρχιτεκτονική για την υποστήριξη και ανάπτυξη *context – aware* εφαρμογών [14, 15, 42].

Υλοποίησε μία υποδομή βασισμένη σε αυτή την αρχιτεκτονική και ένα εργαλείο με το όνομα *Context Toolkit*.

Η αρχιτεκτονική του *Context Toolkit* βασίζεται σε μία αντικειμενοστραφή προσέγγιση και περιλαμβάνει τρεις τύπους αντικειμένων: *widgets*, *servers* ή *aggregators* και *interpreters*. Το *context widget* είναι μία μονάδα λογισμικού που παρέχει στην εφαρμογή πληροφορία *context* από το περιβάλλον εκτέλεσής της. Ο ρόλος των *widgets* είναι να απομονώσουν την εφαρμογή από τις διαδικασίες συλλογής του *context*. Κρύβουν την πολυπλοκότητα των αισθητήρων παρέχοντας αφαιρετικά την πληροφορία, ώστε να μπορεί να χρησιμοποιηθεί από την εφαρμογή.

Οι *context servers* είναι υπεύθυνοι για την συλλογή της πληροφορίας του *context* που αφορά μία συγκεκριμένη οντότητα, όπως για παράδειγμα ένας χρήστης. Αποτελούν υποκλάσεις των *widgets* και κληρονομούν όλες τις ιδιότητες και τις μεθόδους ενός *widget*. Ο *context server* κάνει εγγραφή στο *widget* που τον ενδιαφέρει, αυτό δηλαδή που παρέχει την πληροφορία που σχετίζεται με κάποια οντότητα και λειτουργεί σαν ένα *proxy* μεταξύ *widget* και εφαρμογής.

Οι context *interpreters* μεταφράζουν το context σε κατανοητή μορφή για την εφαρμογή. Μπορούν να κάνουν μετατροπές της πληροφορίας σε διάφορους τύπους αναπαράστασης και να συνδυάσουν διαφορετικούς τύπους πληροφορίας συνθέτοντας και περαιτέρω *context*.

Κάθε ένα από τα παραπάνω αντικείμενα μπορούν να εκτελεστούν αυτόνομα. Τα αντικείμενα μπορούν να δημιουργηθούν όλα σε ένα κόμβο ή σε πολλούς διαφορετικούς. Για την επικοινωνία μεταξύ των αντικειμένων χρησιμοποιείται *http* και *XML*. Ωστόσο και άλλες τεχνολογίες μπορούν να χρησιμοποιηθούν. Η βασική υλοποίηση έχει γίνει σε *Java*, αλλά οι μηχανισμοί που χρησιμοποιούνται είναι ανεξάρτητοι από την γλώσσα προγραμματισμού.

Context Broker Architecture (CoBra)

Το πλαίσιο ενδιάμεσου λογισμικού *CoBra* χρησιμοποιεί μία αρχιτεκτονική βασισμένη σε πράκτορες για την υποστήριξη *context – aware* συστημάτων σε ευφυείς χώρους [11]. Οι ευφυείς χώροι είναι φυσικοί χώροι, όπως αίθουσες συναντήσεων, γραφεία, εργαστήρια και οχήματα, που περιλαμβάνουν έξυπνα υπολογιστικά συστήματα για την παροχή υπηρεσιών προς τους χρήστες.

Το σύστημα περιλαμβάνει μία κεντρική μονάδα διαχείρισης του *context* εκ μέρους των πρακτόρων. Η κεντρική μονάδα είναι ένας ευφυής *broker*, ενώ οι πράκτορες μπορεί να είναι κινητές εφαρμογές, υπηρεσίες που παρέχονται από συσκευές σε ένα χώρο ή υπηρεσίες διαδικτύου. Η κεντρική μονάδα αποτελείται από μία βάση δεδομένων *context*, μία υπομονάδα για την συλλογή του *context* και μία υπομονάδα για την διαχείριση της ασφάλειας των δεδομένων. Στο *CoBra*, χρησιμοποιείται οντολογία υλοποιημένη σε *OWL* για την αναπαράσταση της πληροφορίας [10]. Η εξαγωγή συμπερασμάτων γίνεται με βάση την οντολογία.

Service-oriented Context – Aware Middleware (SOCAM)

Το *SOCAM* είναι άλλο ένα σύστημα ενδιάμεσου λογισμικού για την κατασκευή *context – aware* εφαρμογών [23]. Αποτελείται από ένα κεντρικό εξυπηρέτη που ονομάζεται *Context Interpreter* και συλλέγει πληροφορία από τους προμηθευτές του *context*. Την πληροφορία αυτή την παρέχει στις εφαρμογές σε επεξεργασμένη μορφή. Οι *context – aware* εφαρμογές βρίσκονται στην κορυφή της αρχιτεκτονικής και κάνοντας χρήση διαφόρων επιπέδων *context* προσαρμόζουν την συμπεριφορά τους.

Συγκεκριμένα, το *SOCAM* αποτελείται από τα ακόλουθα συστατικά:

- *Context Providers*: Αυτοί συλλέγουν πληροφορία από διαφόρων ειδών προμηθευτές και την αναπαριστούν με χρήση *OWL*. Διαχωρίζονται σε εξωτερικούς και εσωτερικούς. Οι πρώτοι συλλέγουν πληροφορία από εξωτερικές πηγές, ενώ οι δεύτεροι από αισθητήρες τοποθετημένους σε ένα εσωτερικό χώρο. Κάθε προμηθευτής *context* είναι απαραίτητο να εγγράφεται στην *Locating Service*, ώστε να είναι δυνατό να χρησιμοποιηθεί από τους πελάτες του συστήματος.
- *Context Database*: Η μοντελοποίηση του *context* γίνεται με χρήση οντολογιών. Συγκεκριμένα, ο χώρος πληροφοριών διαμερίζεται σε μικρούς τομείς και ορίζεται για κάθε τέτοιο τομέα μία οντολογία για μείωση της πολυπλοκότητας επεξεργασίας του *context*. Κάθε μία από αυτές τις οντολογίες υλοποιείται σε *OWL*. Επίσης, ορίζεται και μία γενική οντολογία που περιγράφει τις γενικές έννοιες για την διασύνδεση των επιμέρους οντολογιών. Έτσι, στην βάση δεδομένων αποθηκεύονται οι οντολογίες για κάθε διαφορετικό τομέα και τα στιγμιότυπά τους.
- *Context Interpreter*: Λειτουργεί και ως *Context Provider*, εφόσον παίρνει τα δεδομένα από αυτούς, τα επεξεργάζεται και στην συνέχεια τα παρέχει στους χρήστες. Ο *Context*

Interpreter εξάγει *context* υψηλότερου επιπέδου, επιλύει αντιφάσεις στην *context* πληροφορία και διατηρεί σε συνεπή μορφή τα δεδομένα στην βάση. Περιλαμβάνει *RDFS*, *OWL* και βασισμένες σε κανόνες μηχανές εξαγωγής συμπερασμάτων με βάση το *context*. Επίσης περιέχει ένα σύνολο από *APIs* έτσι ώστε τα συστατικά του συστήματος να ρωτάνε, να διαγράφουν και γενικώς να τροποποιούν την πληροφορία στην βάση δεδομένων.

- *Service Locating Service*: Χρησιμοποιείται για την ανακάλυψη διαφόρων προμηθευτών *context*. Όταν έρθει μία ερώτηση προς αυτή την υπηρεσία, τότε θα επιστρέψει μία αναφορά στον προμηθευτή που παρέχει την ζητούμενη πληροφορία.

Hydrogen

Το πλαίσιο ενδιάμεσου λογισμικού *Hydrogen* υιοθετεί μία *peer – to – peer* αρχιτεκτονική, όπου όλα τα τμήματα για την συλλογή και διαχείριση του *context* βρίσκονται πάνω στην κινητή συσκευή [30]. Το σύστημα είναι ανεξαρτητοποιημένο από κεντρικούς κόμβους παροχής υπηρεσιών και παρέχει την δυνατότητα στην εφαρμογή να λειτουργεί, ακόμα και όταν η συσκευή δεν είναι συνδεδεμένη σε κάποιο ασύρματο δίκτυο. Παρέχει υπηρεσίες για συλλογή του *context* από κατανεμημένους προμηθευτές και άλλους κινητούς κόμβους.

Στο *Hydrogen*, το *context* διακρίνεται σε απομακρυσμένο και τοπικό. Δηλαδή, η πληροφορία μπορεί να μεταφερθεί μεταξύ εφαρμογών μέσω μίας υπηρεσίας διαμοιρασμού του *context*. Όταν οι συσκευές βρεθούν σε κοντινή απόσταση πραγματοποιείται ασύρματα ανταλλαγή πληροφορίας. Η επικοινωνία μεταξύ των εφαρμογών γίνεται με χρήση *TCP/IP*, με *XML* μηνύματα. Για την διαχείριση του *context* μίας συσκευής χρησιμοποιείται αντικειμενοστραφές μοντέλο με μία υπερκλάση που ονομάζεται *ContextObject*. Η κλάση αυτή παρέχει μία διεπαφή για την μετατροπή του context από *XML* αναπαραστάσεις σε αντικείμενα και το αντίστροφο. Το *Hydrogen* διαχειρίζεται πέντε είδη *context* που σχετίζονται με τον χρόνο, την θέση, την συσκευή, τον χρήστη και το δίκτυο. Όμως, μπορεί εύκολα να προστεθούν και άλλα είδη *context* εξειδικεύοντας την κλάση *ContextObject*.

Η αρχιτεκτονική του πλαισίου *Hydrogen* αποτελείται από τρία επίπεδα, τα οποία τοποθετούνται στην κινητή συσκευή που εκτελείται η εφαρμογή. Το πρώτο επίπεδο είναι υπεύθυνο για την απόκτηση των δεδομένων από τους προμηθευτές. Αυτό το επίπεδο επιτρέπει ταυτόχρονη χρήση των προμηθευτών από διαφορετικές εφαρμογές. Το δεύτερο

επίπεδο συλλέγει την πληροφορία από το πρώτο και παρουσιάζει το *context* στις εφαρμογές με ένα ασύγχρονο ή σύγχρονο τρόπο. Η επεξεργασία του *context* πραγματοποιείται στο επίπεδο εφαρμογής. Επίσης, το *Hydrogen* δεν αποθηκεύει το *context*, λόγω του ότι οι κινητές συσκευές έχουν περιορισμένη μνήμη. Στην κορυφή της αρχιτεκτονικής βρίσκονται ο εφαρμογές, οι οποίες προσαρμόζονται στις αλλαγές του *context*. Η επικοινωνία μεταξύ των διαφορετικών επιπέδων βασίζεται σε *XML*.

Context-awareness-substructure (CASS)

Το *CASS* είναι ένα ακόμα πλαίσιο που υιοθετεί την αρχιτεκτονική με ένα κεντρικό κόμβο για την διαχείριση του *context* [20]. Οι εφαρμογές – πελάτες είναι ανεξαρτητοποιημένες από την επεξεργασία του *context* και επικοινωνούν με τον κεντρικό κόμβο λαμβάνοντας την επεξεργασμένη πληροφορία. Το ενδιάμεσο λογισμικό περιλαμβάνει υπηρεσίες για επικοινωνία με κατανεμημένους προμηθευτές, μία μηχανή εξαγωγής συμπερασμάτων, διερμηνέα του *context* και μια βάση δεδομένων για αποθήκευση και ανάκτηση της πληροφορίας. Οι εφαρμογές περιλαμβάνουν τμήματα για επικοινωνία με τις υπηρεσίες ενδιάμεσου λογισμικού.

Η αρχιτεκτονική του συστήματος *CASS* περιλαμβάνει τα εξής τμήματα: τον *SensorListener*, τον *ContextRetriever*, τον *Interpreter*, την *RuleEngine* και μία βάση δεδομένων. Το τμήμα *SensorListener* ανιχνεύει αλλαγές του *context* που προέρχεται από κατανεμημένους προμηθευτές. Στην συνέχεια, η συλλεγόμενη πληροφορία αποθηκεύεται από τον *SensorListener* στην βάση δεδομένων. Το τμήμα *ContextRetriever* είναι υπεύθυνο για την ανάκτηση των αποθηκευμένων δεδομένων. Και οι δύο κλάσεις μπορούν να χρησιμοποιήσουν τις υπηρεσίες που παρέχει ο διερμηνέας του *context*, (*Interpreter*). Για την ανάκτηση και την διαχείριση της πληροφορίας από την βάση δεδομένων χρησιμοποιείται *SQL*. Η βάση δεδομένων εκτός από τα δεδομένα του *context* περιέχει και κανόνες. Οι κανόνες χρησιμοποιούνται για την μετάφραση του *context* και την εξαγωγή γνώσης που αφορά την προσαρμογή της εφαρμογής.

Οι κλάσεις στην πλευρά της εφαρμογής χρησιμοποιούνται για την επικοινωνία της εφαρμογής με το σύστημα ενδιάμεσου λογισμικού. Οι εφαρμογές συνδέονται ασύρματα στο σύστημα *CASS*. Για την μείωση της επίδρασης των ασταθών συνδέσεων, το *context* αποθηκεύεται προσωρινά στην κινητή συσκευή όπου εκτελείται η εφαρμογή.

CORTEX

Στο πλαίσιο *CORTEX* [3] δεν υπάρχει κάποιος κεντρικός εξυπηρέτης για την διαχείριση του *context*. Το ενδιάμεσο λογισμικό τοποθετείται κάτω από κάθε ξεχωριστή εφαρμογή. Περιλαμβάνει υπηρεσίες για συλλογή, μετάφραση και εξαγωγή νέας πληροφορίας *context*.

Η αρχιτεκτονική του συστήματος *CORTEX* βασίζεται σε ένα μοντέλο αντικειμένων (*Sensient Object Model*). Το αντικειμενοστραφές αυτό μοντέλο σχεδιάστηκε για την ανάπτυξη *context – aware* εφαρμογών σε *ad – hoc* κινητά περιβάλλοντα. Η καταλληλότητα του μοντέλου για κινητές εφαρμογές βασίζεται στην χρήση μίας υπηρεσίας ενδιάμεσου λογισμικού που παρέχει ενημερότητα θέσης και χρησιμοποιεί γεγονότα για την επικοινωνία μεταξύ των υποσυστημάτων (*STEAM*) [38].

Ένα *Sentient Object* είναι μία οντότητα που σε αυτή ενσωματώνεται η διαχείριση του *context* για κάθε αντικείμενο. Τα αντικείμενα αυτά επικοινωνούν μέσω διεπαφών με τα αντικείμενα που παράγουν και καταναλώνουν γεγονότα. Τα ίδια είναι δυνατό να λειτουργήσουν και ως προμηθευτές και ως καταναλωτές γεγονότων. Για την ανάπτυξη αυτών των αντικειμένων παρέχεται ένα εργαλείο γραφικών.

Κάθε Sentient Object αποτελείται από τρία κύρια μέρη. Η μονάδα *Sensory Capture* χρησιμοποιεί μπεϋζιανά δίκτυα για την διαχείριση της αβεβαιότητας των δεδομένων από τους αισθητήρες και εξάγει *context* με σημασιολογική πληροφορία. Η μονάδα *Context Hierarchy* διατηρεί το *context* που αφορά το αντίστοιχο αντικείμενο καθώς και τις ενέργειες που πρέπει να εκτελεστούν, όταν συμβεί ένα σύνολο γεγονότων. Η πληροφορία αυτή αναπαρίσταται με βάση ένα ιεραρχικό μοντέλο [21]. Τέλος, η μονάδα εξαγωγής συμπερασμάτων είναι υπεύθυνη για την προσαρμογή της εφαρμογής με βάση τους κανόνες που υπάρχουν στο τμήμα *Context Hierarchy*. Η μονάδα αυτή είναι υλοποιημένη σε *CLIPS*.

Middleware for Context Aware Agents (Gaia Project)

Οι *Ranganathan et al.* [40] έχουν κατασκευάσει ένα ενδιάμεσο λογισμικό για την ανάπτυξη *context – aware* εφαρμογών. Αυτό το ενδιάμεσο λογισμικό ενσωματώθηκε στην υποδομή για ευφυείς χώρους *Gaia* [41]. Βασίζεται σε ένα κατηγορηματικό μοντέλο για την αναπαράσταση του *context*, το οποίο επιτρέπει στους πράκτορες του συστήματος να χρησιμοποιούν προσεγγίσεις που βασίζονται σε κανόνες ή σε μηχανική μάθηση, ώστε να αποφασίζουν την

συμπεριφορά τους σε διαφορετικά περιβάλλοντα εκτέλεσης. Επίσης, το ενδιάμεσο λογισμικό χρησιμοποιεί οντολογίες για να εξασφαλίσει διαλειτουργικότητα μεταξύ των διαφορετικών πρακτόρων σε διάφορα περιβάλλοντα λειτουργίας.

Το κύριο χαρακτηριστικό του συγκεκριμένου ενδιάμεσου λογισμικού είναι ότι παρέχει πράκτορες με μία ποικιλία από μηχανισμούς για την εξαγωγή συμπερασμάτων με βάση το context. Οι πράκτορες περιλαμβάνουν τόσο τα συστατικά του ενδιάμεσου λογισμικού, όσο και τις κατανεμημένες εφαρμογές. Χρησιμοποιώντας αυτούς τους μηχανισμούς, οι πράκτορες μπορούν να συμπεραίνουν διάφορα χαρακτηριστικά του τρέχοντος *context*, να απαντούν σε λογικές ερωτήσεις για το *context* και να προσαρμόζουν τον τρόπο που συμπεριφέρονται ανάλογα με το *context*. Οι πράκτορες μπορούν να κάνουν συλλογισμούς με βάση το *context* χρησιμοποιώντας κανόνες που είναι γραμμένοι σε διάφορους τύπους λογικής, όπως λογική πρώτης τάξης, χρονική λογική, περιγραφική λογική, ασαφής λογική κ.α. Επίσης, χρησιμοποιούν διάφορες τεχνικές μηχανικής μάθησης για την διαχείριση του *context*. Οι τεχνικές μηχανικής μάθησης που χρησιμοποιούνται περιλαμβάνουν μπεϋζιανή μάθηση, νευρωνικά δίκτυα, κλπ.

Στην συνέχεια περιγράφεται η αρχιτεκτονική του συστήματος *Middleware for Context Aware Agents*. Ένα από τα βασικά στοιχεία της αρχιτεκτονικής του ενδιάμεσου αυτού λογισμικού είναι οι *Context Providers*. Οι *Context Providers* είναι αισθητήρες ή άλλες πηγές παραγωγής του *context*. Άλλοι πράκτορες μπορούν μέσω ερωτήσεων ή μέσω εγγραφών και ειδοποιήσεων να αποκτούν την πληροφορία του *context* από τους *Context Providers*. Οι *Context Providers* διαφημίζουν την πληροφορία που παρέχουν χρησιμοποιώντας την υπηρεσία *Context Provider Lookup Service*. Η υπηρεσία *Context History Service* επιτρέπει σε άλλους πράκτορες να έχουν πρόσβαση στο ιστορικό του *context*. Διαφορετικοί *Context Providers* χρησιμοποιούν διαφορετικούς μηχανισμούς για την επεξεργασία του *context* και για την απάντηση των ερωτήσεων που λαμβάνουν. Οι *Context Synthesizers* λαμβάνουν το *context* από τους *Context Providers*, συμπεραίνουν *context* υψηλότερου επιπέδου και το παρέχουν σε άλλους πράκτορες. Τέλος, οι *Context Consumers* είναι πράκτορες που λαμβάνουν διάφορους τύπους *context* και με βάση αυτή την πληροφορία προσαρμόζουν τις υπηρεσίες τους.

Όπως αναφέρθηκε παραπάνω, το σύστημα *Middleware for Context Aware Agents* χρησιμοποιεί οντολογίες για να ορίσει την σημασιολογία των διαφόρων ειδών *context*. Οι οντο-

λογίες ορίζουν την σημασιολογία και τις ιδιότητες των διαφόρων ειδών πληροφορίας του *context*. Διαφορετικοί πράκτορες επιτρέπεται να έχουν μία κοινή σημασιολογική κατανόηση του *context*. Οι οντολογίες που χρησιμοποιούνται είναι γραμμένες σε *DAML+OIL*. Η χρήση πρότυπων τεχνολογιών για την σημασιολογία του *context* επιτρέπει διαλειτουργικότητα μεταξύ πρακτόρων σε διαφορετικά περιβάλλοντα. Για την υποστήριξη της διαλειτουργικότητας, απαιτείται να αναπτυχθούν αντιστοιχίσεις μεταξύ των εννοιών που ορίζει η κάθε οντολογία. Το τμήμα *Ontology Server* διατηρεί τις πληροφορίες που περιγράφουν τους διαφορετικούς τύπους του *context* και παρέχει μία διεπαφή για την προσθήκη νέων οντολογιών στις ήδη υπάρχουσες. Έτσι, επιτρέπει νέους τύπους εννοιών να εισάγονται και να χρησιμοποιούνται όταν αυτό απαιτείται.

CARISMA

Το *CARISMA* [8] είναι ένα πλαίσιο ενδιάμεσου λογισμικού για *context – aware* εφαρμογές, το οποίο λειτουργεί εξολοκλήρου στην κινητή συσκευή που εκτελείται η εφαρμογή. Χρησιμοποιεί την αρχή της ανάκλασης για την παροχή υπηρεσιών προσαρμοσμένων στο *context* [7]. Το σύστημα βασίζεται στην διατήρηση πολλαπλών υλοποιήσεων της ίδιας υπηρεσίας, ώστε να παρέχεται με διαφορετικό τρόπο στην εφαρμογή ανάλογα με το *context*. Οι διαφορετικοί τρόποι με τους οποίους παρέχεται μία υπηρεσία ονομάζονται πολιτικές. Η συμπεριφορά του ενδιάμεσου λογισμικού, σε σχέση με μία εφαρμογή, περιγράφεται σαν ένα σύνολο συσχετίσεων μεταξύ των παρεχόμενων υπηρεσιών, των πολιτικών που μπορούν να εφαρμοστούν για την εκτέλεσή τους και τις συνθήκες του *context* που πρέπει να ισχύουν για την εφαρμογή μίας πολιτικής. Η πληροφορία αυτή κωδικοποιείται σε έγγραφα *XML* σχηματίζοντας το προφίλ της εφαρμογής.

Όταν εκτελείται μία υπηρεσία της εφαρμογής, το ενδιάμεσο λογισμικό ελέγχει το προφίλ της εφαρμογής και το συγκρίνει με το τρέχον *context*. Ανάλογα, επιλέγει μία κατάλληλη πολιτική για την εκτέλεση της υπηρεσίας. Το προφίλ της εφαρμογής μπορεί να αλλάξει κατά την διάρκεια της εκτέλεσής της. Επιπλέον, χρησιμοποιείται ένας μηχανισμός για την επίλυση συγκρούσεων στην επιλογή της κατάλληλης πολιτικής.

Context – Aware Middleware for Ubiquitous computing Systems (CAMUS)

Το *CAMUS* [45, 46] είναι ένα πλαίσιο ενδιάμεσου λογισμικού για την υποστήριξη και ανάπτυξη *context – aware* εφαρμογών με τα παρακάτω χαρακτηριστικά:

- Παρέχει ένα ενιαίο πλαίσιο για την συλλογή των δεδομένων του *context* από τους προμηθευτές. Παράλληλα, επιτρέπει δυναμική αναδιαμόρφωση των προμηθευτών στο σύστημα.
- Υποστηρίζει διαφορετικούς μηχανισμούς εξαγωγής συμπερασμάτων με βάση το *context* και περιλαμβάνει κανόνες γραμμένους σε διαφορετικούς τύπους λογικής καθώς και μηχανισμούς μηχανικής μάθησης.
- Χρησιμοποιεί οντολογίες για την μοντελοποίηση του context, παρέχοντας σημασιολογική και συντακτική διαλειτουργικότητα και διαμοιρασμό γνώσης μεταξύ διαφορετικών τομέων.
- Διευκολύνει τους σχεδιαστές στον καθορισμό της συμπεριφοράς των εφαρμογών, των πολιτικών εμπιστευτικότητας των δεδομένων και των μηχανισμών ασφαλείας.

Η αρχιτεκτονική του συστήματος δομείται σε επίπεδα. Στην συνέχεια περιγράφουμε τα βασικά λειτουργικά συστατικά του *CAMUS*. Το χαμηλότερο επίπεδο της αρχιτεκτονικής του συστήματος περιλαμβάνει τους *Feature Extraction Agents* (*FX Agents*). Αυτοί επικοινωνούν με τους προμηθευτές και συλλέγουν το *context*. Χρησιμοποιώντας ειδικούς αλγορίθμους, εξάγούν τα χαρακτηριστικά του *context* και διαχειρίζονται την αβεβαιότητα της πληροφορίας. Η πληροφορία αυτή διαβιβάζεται στα συστατικά *Feature – Context Mapping*. Εκεί, εκτελούνται οι απαιτούμενες αντιστοιχίσεις για να προσδιοριστεί η σημασιολογική έννοια του *context*, με βάση την οντολογία που χρησιμοποιείται. Το επεξεργασμένο *context* και οι οντολογίες διατηρούνται στο τμήμα *Ontology Repository*. Οι οντολογίες υλοποιούνται σε *OWL*.

Ο μηχανισμός *Reasoning Engine* έχει μία συλλογή από διαφορετικά τμήματα που διαχειρίζονται το *context* και τις οντολογίες και παράγουν πιο σύνθετο *context*. Περιλαμβάνει το *Ontology Reasoning Module* και το *Context Reasoning Module*. Το πρώτο τμήμα συνθέτει υψηλότερου επιπέδου context με χρήση των οντολογιών. Το δεύτερο τμήμα χρησιμοποιεί διάφορα είδη λογικής και μηχανισμούς μηχανικής μάθησης για την εξαγωγή συμπερασμάτων.

Στην κορυφή της αρχιτεκτονικής του συστήματος βρίσκεται ο *Context Aggregator*, ο οποίος είναι υπεύθυνος για την εκτέλεση των αιτήσεων των εφαρμογών και την παροχή του *context*

σε αυτές, μέσω των *Context Delivery Services*. Στο στάδιο αυτό, εκτελούνται τεχνικές ελέγχου πρόσβασης στην πληροφορία του *context*.

PACE

Στην εργασία [26] εξετάζονται θέματα που αφορούν την σχεδίαση και ανάπτυξη ενός *context – aware* συστήματος από την πλευρά της τεχνολογίας λογισμικού. Προτείνεται ένα πλαίσιο ανάπτυξης *context – aware* συστημάτων που στοχεύει στην απλοποίηση και τυποποίηση των διαδικασιών ανάπτυξης *context – aware* λογισμικού.

Το πλαίσιο περιλαμβάνει μία γραφική προσέγγιση μοντελοποίησης του *context*, την *Context Modeling Language* (*CML*). Η *CML* παρέχει εργαλεία για την περιγραφή των τύπων του *context*, των κατηγοριών του, πληροφορίας που αφορά την ποιότητα του *context* καθώς και των εξαρτήσεων μεταξύ διαφόρων τύπων του *context*. Σε συνδυασμό με την *CML*, παρέχεται μία διαδικασία μετατροπής του γραφικού μοντέλου σε σχεσιακό και ένας τρόπος περιγραφής του *context* με χρήση κατηγορηματικής λογικής. Επιπλέον, το πλαίσιο περιλαμβάνει μία τεχνική για μοντελοποίηση των προτιμήσεων των χρηστών και δύο συμπληρωματικά προγραμματιστικά μοντέλα.

Συνέχεια της εργασίας αυτής αποτελεί η ενσωμάτωση του πλαισίου σε μία υποδομή ενδιάμεσου λογισμικού, το σύστημα *PACE* [28]. Το ενδιάμεσο λογισμικό *PACE* αποτελείται από:

- Ένα σύστημα διαχείρισης του *context*.
- Ένα σύστημα διαχείρισης των προτιμήσεων των χρηστών.
- Ένα προγραμματιστικό εργαλείο που διευκολύνει την αλληλεπίδραση μεταξύ των εφαρμογών και του ενδιάμεσου λογισμικού.
- Βοηθητικά εργαλεία για την δημιουργία συστατικών που μπορούν να χρησιμοποιηθούν από τα παραπάνω συστήματα.

Το σύστημα διαχείρισης του *context* παρέχει ομαδοποίηση και αποθήκευση της πληροφορίας του *context*, καθώς και εκτέλεση των ερωτημάτων από το επίπεδο εφαρμογής. Χρησιμοποιεί το μοντέλο του *context* που περιγράφτηκε στην εργασία [26]. Αποτελείται από ένα κατανεμημένο σύνολο αποθηκευτικών χώρων. Καθένας από αυτούς διαχειρίζεται ένα σύνολο από μοντέλα του *context*. Οι εφαρμογές μπορούν να προσδιορίσουν τα δικά τους μοντέλα *context* ή να τα μοιράζονται με άλλες εφαρμογές. Οι *context – aware* εφαρμογές δεν είναι

στατικά συνδεδεμένες με ένα μοναδικό αποθηκευτικό χώρο, αλλά μπορούν να ανακαλύπτουν δυναμικά νέους. Μπορούν να χρησιμοποιηθούν διάφορες μέθοδοι αλληλεπίδρασης με ένα αποθηκευτικό χώρο. Αντίστοιχα, μπορεί να χρησιμοποιηθεί μία ποικιλία από μηχανισμούς ανακάλυψης αποθηκευτικών χώρων.

Κάθε αποθηκευτικός χώρος έχει την δυνατότητα εκτέλεσης ελέγχου πρόσβασης σε αυτόν. Ο μηχανισμός ελέγχου πρόσβασης επιτρέπει στους χρήστες να ορίζουν πολιτικές που καθορίζουν πότε το *context* που διατηρείται σε αυτούς μπορεί να προσπελάζεται. Οι πολιτικές αυτές αποθηκεύονται και αποτιμώνται από το σύστημα διαχείρισης των προτιμήσεων των χρηστών.

Το σύστημα διαχείρισης των προτιμήσεων των χρηστών βοηθάει τις *context – aware* εφαρμογές να λαμβάνουν αποφάσεις με βάση το τρέχον *context* προς όφελος των χρηστών. Ο κύριος ρόλος του είναι να αποθηκεύει την πληροφορία που εκφράζει τις προτιμήσεις των χρηστών και να προσδιορίζει ποιες ενέργειες πρέπει να εκτελεστούν σύμφωνα με αυτές. Οι εφαρμογές μπορούν να αποθηκεύουν τις προτιμήσεις σε ένα ή περισσότερους αποθηκευτικούς χώρους. Οι προτιμήσεις περιγράφονται και χρησιμοποιούνται σύμφωνα με το μοντέλο που έχουν αναπτύξει οι ερευνητές.

Το προγραμματιστικό εργαλείο συμπληρώνει την λειτουργικότητα των προηγούμενων συστημάτων εκτελώντας τις ενέργειες προσαρμογής των εφαρμογών. Ένα σημαντικό πλεονέκτημα του εργαλείου είναι ότι κάνει διαφανή στις εφαρμογές την διαδικασία επικοινωνίας με το σύστημα διαχείρισης του *context* και των προτιμήσεων των χρηστών. Επιπλέον, χρησιμοποιείται ένα ευέλικτο πλαίσιο, το οποίο στηρίζεται σε μηνύματα, για την επικοινωνία μεταξύ των συστατικών του *context – aware* συστήματος.

Το τελευταίο τμήμα του ενδιάμεσου λογισμικού *PACE* είναι ένα σύνολο εργαλείων, τα οποία παράγουν συστατικά που βοηθούν στον σχηματισμό και στην ανάπτυξη *context – aware* συστημάτων, όπως είναι τα διάφορα συστατικά που απαιτούνται για την μοντελοποίηση του context.

ΚΕΦΑΛΑΙΟ 3. CONTEXT – AWARE ΕΝΔΙΑΜΕΣΟ ΛΟΓΙΣΜΙΚΟ ΓΙΑ ΚΙΝΗΤΕΣ ΣΥΣΚΕΥΕΣ

Σε αυτό το κεφάλαιο περιγράφεται η υπηρεσία ενδιάμεσου λογισμικού για *context – aware* εφαρμογές σε κινητές συσκευές που σχεδιάστηκε και αναπτύχθηκε στην μεταπτυχιακή εργασία [33]. Αναλύονται η αρχιτεκτονική της υπηρεσίας ενδιάμεσου λογισμικού και το μοντέλο που χρησιμοποιείται για την αναπαράσταση και την διαχείριση του *context*.

3.1 Γενική Περιγραφή του Συστήματος

Στην μεταπτυχιακή εργασία [33] προτείνεται μία υπηρεσία ενδιάμεσου λογισμικού που διαχειρίζεται το *context* με διαφάνεια προς την εφαρμογή και προσαρμόζει αυτόματα την λειτουργία της εφαρμογής στις αλλαγές του *context*. Η υπηρεσία ενδιάμεσου λογισμικού σχεδιάστηκε και υλοποιήθηκε για να λειτουργεί σε κινητές συσκευές, για την υποστήριξη των κινητών εφαρμογών. Μπορεί να ενσωματωθεί στο λογισμικό της κινητής εφαρμογής, λειτουργώντας αυτόνομα ή ως τμήμα ενός ευρύτερου *context – aware* ενδιάμεσου λογισμικού.

Η υπηρεσία ενδιάμεσου λογισμικού αναλαμβάνει όλη την διαχείριση της πληροφορίας που λαμβάνεται από τα χαμηλότερα επίπεδα και προσαρμόζει τις λειτουργίες της εφαρμογής,

χωρίς την παρέμβαση του χρήστη. Παρέχει ένα πλαίσιο για την υλοποίηση της επικοινωνίας με τους εξωτερικούς προμηθευτές. Εκεί, ο προγραμματιστής μπορεί να υλοποιήσει την επικοινωνία με τις πηγές του *context* ή με άλλα τμήματα ενδιάμεσου λογισμικού, τα οποία παρέχουν επεξεργασμένη την πληροφορία (π.χ. εξυπηρέτες *context*).

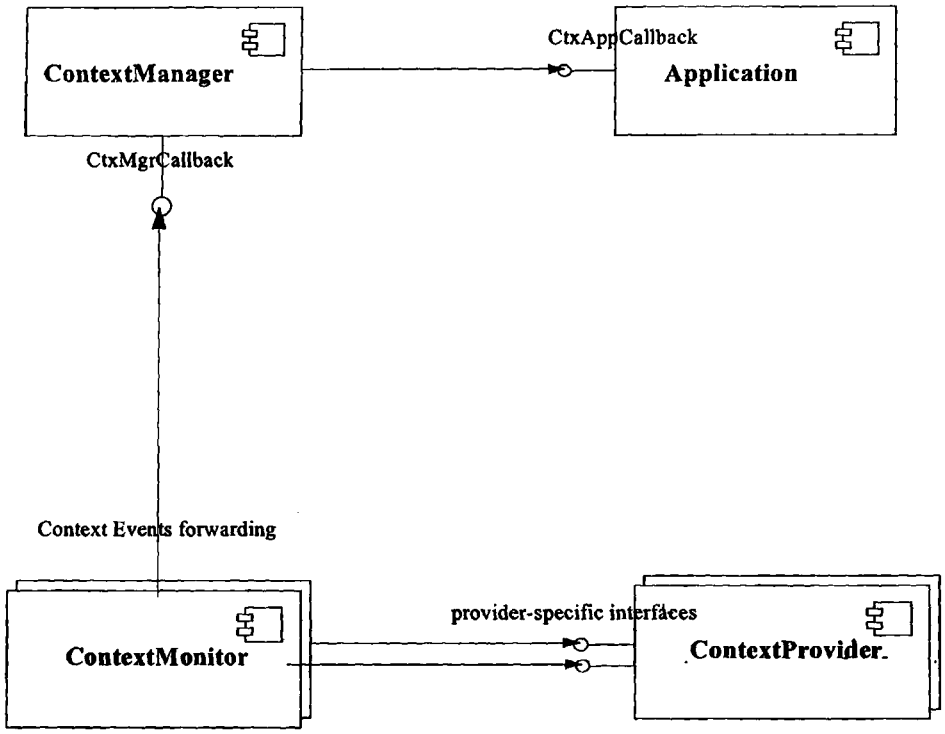
Για την διαχείριση του *context* και την προσαρμογή της εφαρμογής, προτείνεται ένα μοντέλο βασισμένο στην *MTL* πρώτης τάξης [37]. Η συμπεριφορά της εφαρμογής σε σχέση με τις αλλαγές του *context* μοντελοποιείται με ένα σύνολο λογικών κανόνων συνθήκης – ενέργειας (*ECA*) [35]. Οι κανόνες αυτοί καθορίζουν, τις ενέργειες που πρέπει να εκτελέσει η υπηρεσία ενδιάμεσου λογισμικού, αν ισχύσουν συγκεκριμένες συνθήκες για το *context*. Οι κανόνες περιλαμβάνουν ένα σύνολο χρονικών τελεστών για τον χρονικό προσδιορισμό των συνθηκών και των ενεργειών. Για την αποτίμηση των κανόνων και την επιλογή των ενεργειών προσαρμογής, υλοποιείται μία μέθοδος που χειρίζεται τις χρονικές εξαρτήσεις στις παραμέτρους του *context*, κατά την διαδικασία αποτίμησης. Οι κανόνες ελέγχονται σύμφωνα με τα γεγονότα και αποφασίζεται αν πρέπει να πραγματοποιηθούν ενέργειες προσαρμογής.

Η υπηρεσία ενδιάμεσου λογισμικού μπορεί να διαμορφώνει την συμπεριφορά της, βελτιώνοντας την απόδοση του συστήματος. Η διαμόρφωση δεν πραγματοποιείται άμεσα σαν ενέργεια κάποιου κανόνα, αλλά μπορεί να προκύψει έμμεσα, σαν αποτέλεσμα των ενεργειών ενός κανόνα. Επιπλέον, προσδίδονται στην υπηρεσία ανακλαστικά χαρακτηριστικά, δίνοντας την δυνατότητα στον χρήστη να διαμορφώνει δυναμικά την συμπεριφορά της. Συγκεκριμένα, η υπηρεσία ενδιάμεσου λογισμικού παρέχει μεθόδους για την δυναμική διαμόρφωση των κανόνων, κατά την λειτουργία του συστήματος.

3.2 Αρχιτεκτονική της Υπηρεσίας Ενδιάμεσου Λογισμικού

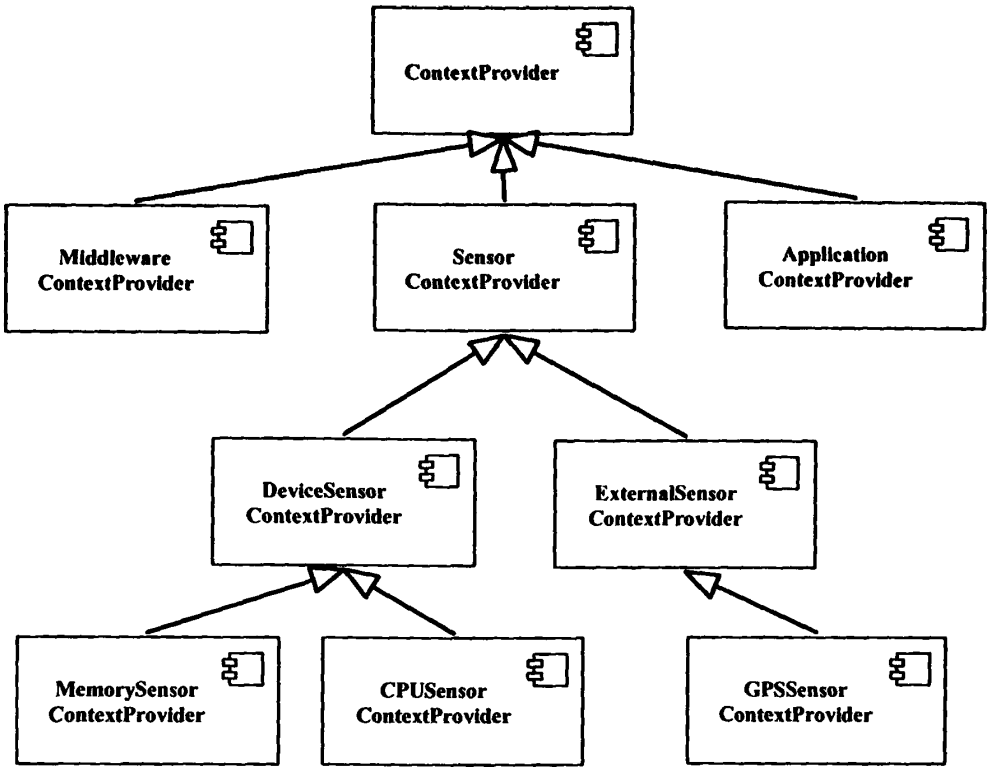
Στο Σχήμα 3.1 φαίνεται ο τρόπος με τον οποίο, μία *context – aware* εφαρμογή αντιλαμβάνεται το περιβάλλον που εκτελείται και ο ρόλος της προτεινόμενης υπηρεσίας ενδιάμεσου λογισμικού σε αυτό το περιβάλλον. Το τμήμα *ContextManager* είναι η κεντρική μονάδα της υπηρεσίας ενδιάμεσου λογισμικού. Χρησιμοποιεί την πληροφορία που απεικονίζει τις αλλαγές στο *context* και προσαρμόζει τις λειτουργίες της εφαρμογής σύμφωνα με αυτές τις αλλαγές.

Από την πλευρά της εφαρμογής, το περιβάλλον περιλαμβάνει ένα σύνολο προμηθευτών που παρέχουν την πληροφορία του *context*. Οι προμηθευτές του *context* μπορεί να ποικίλουν από αισθητήρες, τοποθετημένους στην κινητή συσκευή ή στο περιβάλλον που κινείται ο χρήστης, έως τμήματα ενδιάμεσου λογισμικού που σχετίζονται με την εφαρμογή (Σχήμα 3.2). Η ίδια η εφαρμογή μπορεί να θεωρηθεί ως ένας προμηθευτής *context*. Γενικά, το *context* της εφαρμογής αποτελείται από ένα πεπερασμένο σύνολο παραμέτρων. Κάθε μία από αυτές τις παραμέτρους παίρνει τιμές ενός συγκεκριμένου τύπου. Θεωρείται ότι οι παράμετροι του *context* είναι ακέραιοι ή πραγματικοί αριθμοί, καθώς και συμβολοσειρές. Η διαχείριση πιο σύνθετων τύπων δεδομένων απαιτεί αυξημένο ποσοστό υπολογιστικών πόρων.



Σχήμα 3.1: Γενική αρχιτεκτονική της υπηρεσίας.

Οι τιμές των παραμέτρων του *context* προωθούνται από τους μηχανισμούς παρακολούθησης των προμηθευτών στο μηχανισμό εξαγωγής συμπερασμάτων του *ContextManager*. Ο τρόπος με τον οποίο ο μηχανισμός εξαγωγής συμπερασμάτων προσαρμόζει τις λειτουργίες της εφαρμογής, με βάση αυτά τα γεγονότα, καθορίζεται από ένα σύνολο κανόνων που δίνονται ως είσοδος στον *ContextManager*. Ένας κανόνας αποτελείται από το τμήμα συνθηκών και ενεργειών. Το τμήμα συνθηκών προσδιορίζει τις χρονικά συσχετιζόμενες αλλαγές στις τιμές των παραμέτρων του *context* που ενεργοποιούν χρονικά συσχετιζόμενες ενέργειες προσαρμογής, οι οποίες καθορίζονται στο τμήμα ενεργειών.



Σχήμα 3.2: Ιεραρχία προμηθευτών context.

Έστω, για παράδειγμα, μία εφαρμογή που χρησιμοποιεί ένα μηχανισμό εισαγωγής δεδομένων που μπορεί να ενεργοποιηθεί ή να απενεργοποιηθεί καλώντας μία συγκεκριμένη υπηρεσία που προσφέρεται από τον μηχανισμό. Το μοτίβο λειτουργίας του μηχανισμού εισαγωγής δεδομένων είναι μία *context* παράμετρος που μπορεί να χρησιμοποιηθεί στο τμήμα ενεργειών ενός κανόνα. Θέτοντας την τιμή αυτής της παραμέτρου σε κατάσταση ανενεργό σε μία ενέργεια του κανόνα, συνεπάγεται την κλήση της υπηρεσίας που πραγματικά απενεργοποιεί τον μηχανισμό εισαγωγής δεδομένων. Η διαδικασία αυτή πραγματοποιείται, μέσω της χρήσης μίας προκαθορισμένης *callback* διεπαφής από τον μηχανισμό εξαγωγής συμπερασμάτων, κατά την εκτέλεση του τμήματος ενεργειών ενός κανόνα. Η *callback* διεπαφή πρέπει να υλοποιηθεί από τον προγραμματιστή της εφαρμογής, κατά την ενσωμάτωση της προτεινόμενης υπηρεσίας ενδιάμεσου λογισμικού στην εφαρμογή.

Η *callback* διεπαφή παρέχει μία προκαθορισμένη λειτουργία προσαρμογής που δέχεται ως είσοδο τις τιμές των παραμέτρων του *context*, όπως αυτές προσδιορίζονται στο τμήμα ενεργειών ενός κανόνα. Στο παράδειγμά μας, αυτό θα μπορούσε να είναι η τιμή της παραμέτρου που αναπαριστά το μοτίβο λειτουργίας του μηχανισμού εισαγωγής δεδομένων. Με βάση τις τιμές εισόδου, η λειτουργία προσαρμογής καλεί τις αντίστοιχες λειτουργίες της εφαρμογής ή της υπηρεσίας ενδιάμεσου λογισμικού για την προσαρμογή της εφαρμογής. Στο παράδειγμα

μας, αυτό θα μπορούσε να είναι η λειτουργία που ενεργοποιεί ή απενεργοποιεί τον μηχανισμό εισαγωγής δεδομένων.

Έτσι, για την ενοποίηση της προτεινόμενης υπηρεσίας ενδιάμεσου λογισμικού με μία *context – aware* εφαρμογή εκτελούνται τα παρακάτω βήματα:

1. Ο προγραμματιστής της εφαρμογής δίνει ως είσοδο στην υπηρεσία ενδιάμεσου λογισμικού το σύνολο των παραμέτρων του *context* και των κανόνων.
2. Ο προγραμματιστής συνδέει την υπηρεσία ενδιάμεσου λογισμικού με:
    I. Την *callback* διεπαφή που εκτελεί τις ενέργειες προσαρμογής της εφαρμογής.
    II. Τους μηχανισμούς παρακολούθησης των προμηθευτών που καταγράφουν τις αλλαγές στις τιμές των παραμέτρων του *context*.

3.3 Μοντελοποίηση του Context

Το κύριο θέμα στην μοντελοποίηση του *context* είναι ο προσδιορισμός των κανόνων του *context*, που καθοδηγούν τον μηχανισμό εξαγωγής συμπερασμάτων της προτεινόμενης υπηρεσίας ενδιάμεσου λογισμικού. Ένας κανόνας είναι ένας τύπος χρονικής λογικής που αποτελείται από ένα τμήμα συνθηκών και ένα τμήμα ενεργειών. Το τμήμα συνθηκών περιλαμβάνει απλές συνθήκες που σχετίζονται με συζεύξεις ή διαζεύξεις, σχηματίζοντας σύνθετες συνθήκες. Πιο συγκεκριμένα, το τμήμα συνθηκών ορίζεται είτε σε Κανονική Συζευκτική Μορφή (*CNF*) [19] είτε σε Κανονική Διαζευκτική Μορφή (*DNF*) [19] (Πίνακας 3.1). Μία απλή συνθήκη συνδέει μία παράμετρο του *context* με μία τιμή, μέσω της χρήσης ενός τυπικού τελεστή σύγκρισης (Πίνακας 3.2). Το τμήμα συνθηκών συνδέεται με το τμήμα ενεργειών μέσω μίας λογικής συνεπαγωγής. Έτσι, όταν το τμήμα συνθηκών ισχύει, ενεργοποιείται το τμήμα ενεργειών.

Συγκεκριμένα, μία απλή συνθήκη ικανοποιείται την χρονική στιγμή της αποτίμησης ενός κανόνα, εάν η παράμετρος του *context* που υπάρχει στην συνθήκη έχει την τρέχουσα χρονική στιγμή, τιμή που ικανοποιεί την συνθήκη. Παρόμοια, μία σύνθετη συνθήκη ικανοποιείται την χρονική στιγμή της αποτίμησης ενός κανόνα, εάν η λογική σύζευξη ή διάζευξη των επιμέρους απλών συνθηκών είναι αληθής εκείνη την στιγμή.

Πίνακας 3.1: Συντακτικό των κανόνων του context.

<rule>	::=	<conditional_part> ⇒ <action_part>
<conditional _part>	::=	<c_statement> | <d_statement>
<c_statement>	::=	<past_operator> (<c_condition>) && <c_statement>
	|	<past_operator> (<c_condition>)
<d_statement>	::=	<past_operator> (<d_condition>) || <d_statement>
	|	<past_operator> (<d_condition>)
<c_condition>	::=	<op_condition> || <c_condition> | <op_condition>
<d_condition>	::=	<op_condition> && <d_condition> | <op_condition>
<op_condition>	::=	(<past_operator> <condition>)
<condition>	::=	<context_property> <comparison_operator> <value>
<past_operator>	::=	[previous] | [once] | [always_been] | ε
<action_part>	::=	<a_statement>
<a_statement>	::=	<action> && <a_statement>
	|	<action> && [next] (<a_statement>)
	|	<action>
<action>	::=	(<future_op> <context_property> = <value>)
	|	<context_property> = <value> [until] <condition>
<future_op>	::=	[next] | [henceforth] | ε
<comparison_operator>	::=	> | < | = | >= | <= | !=

Μία απλή ή σύνθετη συνθήκη μπορεί να συνοδεύεται από ένα χρονικό λογικό παρελθοντικό τελεστή (Πίνακας 3.2). Οι παρελθοντικοί τελεστές επιβάλουν επιπλέον χρονικούς περιορισμούς στην ικανοποίηση μίας συνθήκης. Ειδικότερα, για μία συνθήκη A:

- Η έκφραση $[previous]A$ είναι αληθής την χρονική στιγμή t, αν η συνθήκη A ήταν αληθής την χρονική στιγμή $t-1$. Έτσι, ο τελεστής χρησιμοποιείται σε μία συνθήκη για να καθορίσει ότι μία παράμετρος του *context* πρέπει να πάρει μία τιμή που ικανοποιεί την A ακριβώς πριν το γεγονός που ενεργοποίησε την αποτίμηση του κανόνα, ο οποίος περιλαμβάνει αυτή την έκφραση.
- Η έκφραση $[once]A$ είναι αληθής την χρονική στιγμή t, αν η συνθήκη A ικανοποιήθηκε κάποια χρονική στιγμή $t_1 : 0 < t_1 < t$. Αυτός ο τελεστής χρησιμοποιείται σε μία συνθήκη για να καθορίσει ότι μία παράμετρος του *context* πρέπει να πάρει τιμή που ικανοποιεί την A τουλάχιστον μία φορά πριν το γεγονός που ενεργοποίησε την αποτίμηση του κανόνα, ο οποίος περιλαμβάνει αυτή την έκφραση.
- Η έκφραση $[always\_been]A$ είναι αληθής την χρονική στιγμή t, αν η συνθήκη A ικανοποιήθηκε την χρονική στιγμή $t_1 : 0 < t_1 < t$ και στο χρονικό διάστημα $[t_1, t]$ δεν έγινε ποτέ ψευδής. Έτσι, αυτός ο τελεστής χρησιμοποιείται για να καθορίσει ότι σε κάποια στιγμή στο παρελθόν μία παράμετρος του *context* πρέπει να πάρει τιμή που συνεχώς ικανοποιεί την A μέχρι το γεγονός που ενεργοποίησε την αποτίμηση του κανόνα, ο οποίος περιλαμβάνει αυτή την έκφραση.

Το τμήμα ενεργειών ενός κανόνα αποτελείται από την λογική σύζευξη ενεργειών. Κάθε ενέργεια συσχετίζει μία παράμετρο του *context* με μία τιμή μέσω του τελεστή ισότητας. Αυτό σημαίνει ότι, η παράμετρος πρέπει να τεθεί στην συγκεκριμένη τιμή, εκτελώντας την κατάλληλη λειτουργία προσαρμογής της εφαρμογής. Η λειτουργία προσαρμογής καλείται από την υπηρεσία ενδιάμεσου λογισμικού μέσω της *callback* διεπαφής, που υλοποιείται στην εφαρμογή. Διαζεύξεις μεταξύ των ενεργειών δεν υποστηρίζονται, παρόλο που είναι πιθανό να συμβούν καταστάσεις που συνεπάγονται περισσότερες από μία εναλλακτικές ενέργειες.

Οι ενέργειες μπορεί να συνοδεύονται από χρονικούς λογικούς μελλοντικούς τελεστές (Πίνακας 3.2), οι οποίοι επιβάλλουν την σειρά με την οποία οι αντίστοιχες ενέργειες πρέπει να εκτελεστούν. Συγκεκριμένα:

- Ο τελεστής $[next]$ καθορίζει την σειρά εκτέλεσης των ενεργειών. Για παράδειγμα, αν το τμήμα ενεργειών ενός κανόνα είναι:

$$A \,\&\&\, [next]\big([next]\big([next]C \,\&\&\, D\big) \,\&\&\, B\big)$$

τότε, η διάταξη των ενεργειών A, B, C και D είναι: A, B, C, D.

- Η έκφραση $A[until]B$ δηλώνει ότι, τα αποτελέσματα της ενέργειας A θα ισχύουν μέχρι την χρονική στιγμή που η συνθήκη B θα γίνει αληθής. Πρακτικά, αυτό σημαίνει ότι, η παράμετρος του *context* που υπάρχει στην ενέργεια A δεν μπορεί να πάρει μία νέα τιμή από άλλη ενέργεια μέχρι να ικανοποιηθεί η συνθήκη B. Μπορεί, όμως, να αλλάξει τιμή απ΄ ευθείας από τον χρήστη.
- Η έκφραση $[henceforth]A$ δηλώνει ότι, τα αποτελέσματα της ενέργειας A θα ισχύουν συνεχώς στο μέλλον. Πρακτικά, αυτό σημαίνει ότι, η παράμετρος του *context* που υπάρχει στην ενέργεια A θα πάρει μία νέα τιμή, η οποία δεν θα αλλάξει ποτέ στο μέλλον, μέσω άλλης ενέργειας από την υπηρεσία ενδιάμεσου λογισμικού. Μπορεί, όμως, να αλλάξει τιμή από επιλογή του χρήστη.

Πίνακας 3.2: Τελεστές που χρησιμοποιούνται στους κανόνες του context.

Τελεστές	
Λογικοί τελεστές	||, &&, ⇒
Τελεστές σύγκρισης	<, <, =, >=, <=, !=
Παρελθοντικοί τελεστές	[previous]
	[once]
	[always_been]
Μελλοντικοί τελεστές	[next]
	[until]
	[henceforth]

3.4 Εξαγωγή Συμπερασμάτων με Βάση το Context

Ο *ContextManager* είναι η κεντρική μονάδα της προτεινόμενης υπηρεσίας ενδιάμεσου λογισμικού που μεσολαβεί στην αλληλεπίδραση μεταξύ της εφαρμογής και των άλλων υποσυστημάτων της υπηρεσίας. Ειδικότερα, η εφαρμογή δημιουργεί ένα αντικείμενο *ContextManager*, το οποίο αναλαμβάνει την αρχικοποίηση και τον συντονισμό των διάφορων

διαδικασιών της υπηρεσίας ενδιάμεσου λογισμικού, που παρέχονται από τα αντικείμενα *RulesParser*, *EventHandler* και *RulesEvaluator* (Σχήμα 3.3). Ο *ContextManager* δέχεται ως είσοδο από την εφαρμογή την δήλωση των παραμέτρων του *context*, που αφορούν την εφαρμογή, και τους κανόνες. Η πληροφορία αυτή προωθείται στα υπόλοιπα υποσυστήματα για περαιτέρω επεξεργασία και χρήση.

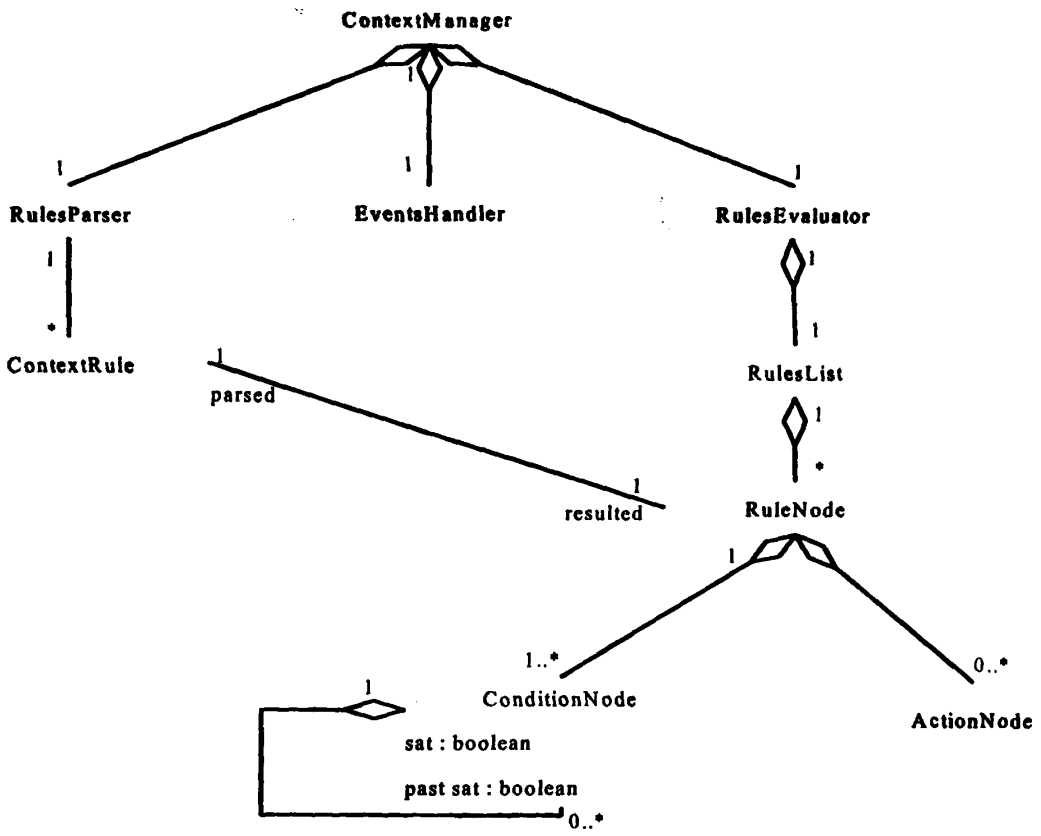
Για διευκόλυνση της εργασίας του προγραμματιστή, παρέχονται διάφοροι εναλλακτικοί τρόποι φόρτωσης των παραμέτρων του *context* και των κανόνων. Αρχικά, είναι δυνατό να δοθούν στον κώδικα της εφαρμογής, μέσω της χρήσης μίας διεπαφής που παρέχεται από τον *ContextManager*. Εναλλακτικά, οι παράμετροι και οι κανόνες μπορεί να δοθούν σε ένα αρχείο που αποθηκεύεται στον μόνιμο αποθηκευτικό χώρο της κινητής συσκευής. Μία τρίτη επιλογή είναι ο δυναμικός καθορισμός των παραμέτρων και των κανόνων, μέσω μίας γραφικής διεπαφής που παρέχεται από την υπηρεσία ενδιάμεσου λογισμικού. Η γραφική αυτή διεπαφή μπορεί εύκολα να προστεθεί στην γραφική διεπαφή της εφαρμογής.

Το Σχήμα 3.3 συνοψίζει την εσωτερική δομή του *ContextManager*. Στην συνέχεια, περιγράφεται λεπτομερώς η συμπεριφορά των αντικειμένων που αποτελούν τον *ContextManager*.

3.4.1 Συντακτική Ανάλυση Κανόνων

Το αντικείμενο *RulesParser* είναι υπεύθυνο για την συντακτική ανάλυση των κανόνων. Δέχεται ως είσοδο τους κανόνες από τον *ContextManager* (Σχήμα 3.4). Κατά την διάρκεια της ανάλυσης, αποσυνθέτει τους κανόνες και αποθηκεύει την καθορισμένη πληροφορία σε μία δομή δεδομένων στην μνήμη.

Για κάθε κανόνα, ο *RulesParser* κατασκευάζει ένα νέο κόμβο (Σχήμα 3.3) σε αυτή την δομή. Ο κόμβος δείχνει σε δύο λίστες, οι οποίες περιέχουν πληροφορία για το τμήμα συνθηκών και ενεργειών του κανόνα, αντίστοιχα (Σχήμα 3.3). Η λίστα των συνθηκών περιέχει ένα κόμβο για κάθε διαφορετική συνθήκη. Για κάθε σύνθετη συνθήκη δημιουργείται μια νέα λίστα που περιέχει τις απλές συνθήκες. Η λίστα των ενεργειών περιέχει κόμβους που αντιστοιχούν στις ενέργειες του κανόνα. Οι κόμβοι αυτοί διατάσσονται σύμφωνα με την σειρά εκτέλεσης των ενεργειών, όπως αυτή καθορίζεται από τελεστές [*next*] που χρησιμοποιούνται στον κανόνα.



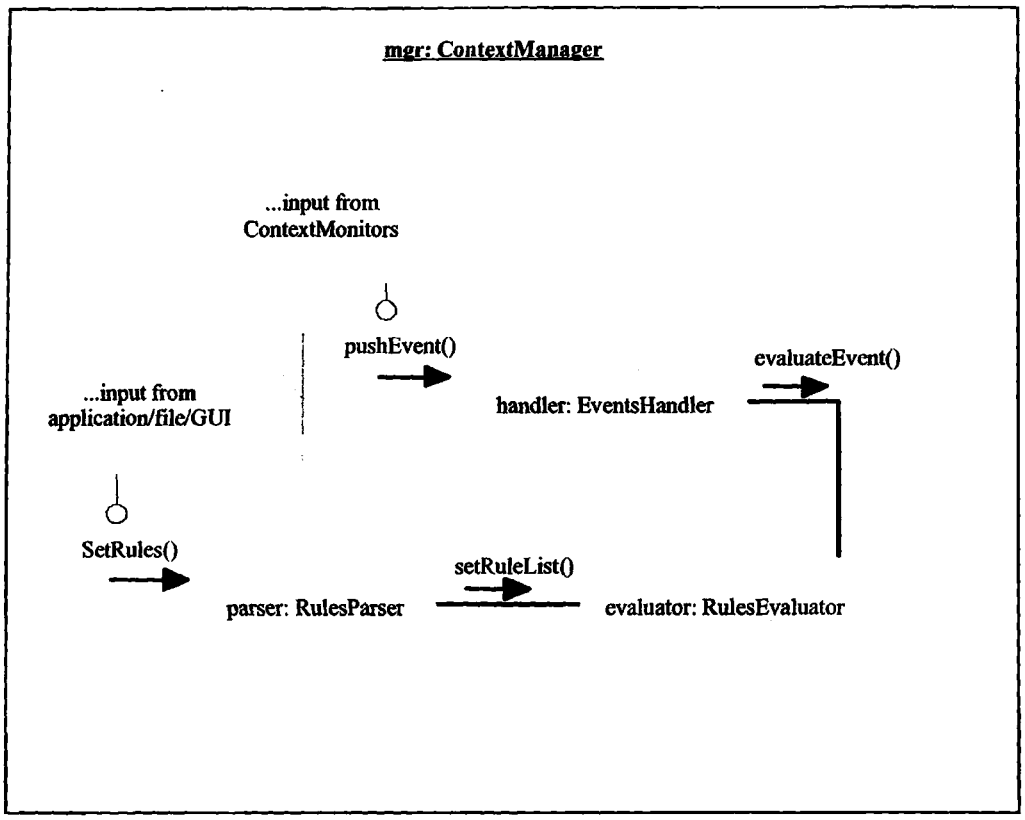
Σχήμα 3.3: Αρχιτεκτονική του ContextManager.

3.4.2 Διαχείριση Γεγονότων

Το αντικείμενο *EventHandler* δέχεται ως είσοδο γεγονότα από τους μηχανισμούς παρακολούθησης των προμηθευτών του *context* (Σχήμα 3.4). Τα γεγονότα περιέχουν τις τρέχουσες τιμές των παραμέτρων του *context* που ενδιαφέρουν την εφαρμογή. Αυτές οι τιμές προωθούνται στον μηχανισμό εξαγωγής συμπερασμάτων για επεξεργασία και αποτίμηση.

Ο *EventHandler* περιέχει μία ουρά, όπου αποθηκεύονται τα γεγονότα του *context*. Η ουρά χρησιμοποιείται από δύο διαφορετικά νήματα. Το πρώτο νήμα είναι υπεύθυνο για την άφιξη των νέων γεγονότων. Δεδομένου ότι, η άφιξη των νέων γεγονότων διαχειρίζεται από ένα μόνο νήμα, όλα τα γεγονότα διατάσσονται συνολικά με βάση τον χρόνο επεξεργασίας τους από αυτό το νήμα. Συγκεκριμένα, στο μοντέλο του *context* ενσωματώνεται η έννοια του σχετικού και όχι του πραγματικού χρόνου. Στην έναρξη λειτουργίας της υπηρεσίας ενδιάμεσου λογισμικού, ισχύει $t = 0$. Όταν ένα νέο γεγονός αποθηκεύεται στην ουρά, ο χρόνος αυξάνεται κατά μία μονάδα, δηλαδή $t = t + 1$. Αυτή η χρονική ταμπέλα συνδέεται με το γεγονός και χρησιμοποιείται κατά την διάρκεια αποτίμησης των κανόνων με βάση το

γεγονός. Το δεύτερο νήμα σχετίζεται με την εξαγωγή των γεγονότων από την ουρά και την προώθησή τους στο υποσύστημα ελέγχου ικανοποιησιμότητας των κανόνων. Η αποτίμηση των κανόνων εκτελείται σε ένα μόνο νήμα, έτσι ώστε να διατηρείται η χρονική συνέπεια της διαδικασίας εξαγωγής συμπερασμάτων.



Σχήμα 3.4: Τα συστατικά του ContextManager κατά την εκτέλεση της υπηρεσίας.

Η χρονική συνεπεία ορίζεται ως εξής: Η αποτίμηση των κανόνων με βάση ένα γεγονός *e* πρέπει να πραγματοποιηθεί μετά την εκτέλεση των ενεργειών προσαρμογής που προκύπτουν από την αποτίμηση των κανόνων με βάση το γεγονός που προηγήθηκε του γεγονότος.

3.4.3 Αποτίμηση Κανόνων

Το αντικείμενο *RulesEvaluator* πραγματοποιεί την διαδικασία αποτίμησης των κανόνων στην υπηρεσία ενδιάμεσου λογισμικού. Συγκεκριμένα, καλείται από τον *EventHandler* για κάθε γεγονός *e* που εξάγεται από την ουρά γεγονότων. Στην συνέχεια, ο *RulesEvaluator* ελέγχει τα τμήματα συνθηκών των κανόνων που περιέχουν την παράμετρο του *context*, της οποίας η

νέα τιμή δίνεται από αυτό το γεγονός. Η αποτίμηση ενός κανόνα βασίζεται στην πληροφορία που υπάρχει στην δομή δεδομένων, μετά την συντακτική ανάλυση των κανόνων.

Κάθε κόμβος που αντιστοιχεί σε μία συνθήκη περιέχει μία λογική μεταβλητή, που ονομάζεται *sat* (Σχήμα 3.3). Η λογική αυτή μεταβλητή δηλώνει την ικανοποιησιμότητα της αντίστοιχης συνθήκης την τρέχουσα χρονική στιγμή που γίνεται ο έλεγχος. Αν μία συνθήκη συνοδεύεται από παρελθοντικό τελεστή, ο αντίστοιχος κόμβος περιέχει μία επιπλέον λογική μεταβλητή, που ονομάζεται *past_sat* (Σχήμα 3.3). Η τιμή αυτής της λογικής μεταβλητής εξαρτάται από τον τύπο του παρελθοντικού τελεστή που συνοδεύει την συνθήκη. Πιο συγκεκριμένα:

- Για τον χρονικό τελεστή $[previous]$, η τιμή της λογικής μεταβλητή *past_sat* τίθεται σε «αληθής», αν η συνθήκη ίσχυε μετά την εκτέλεση της διαδικασίας αποτίμησης των κανόνων, με βάση ένα γεγονός e', που έφτασε ακριβώς πριν το γεγονός e.
- Για τον χρονικό τελεστή $[once]$, η τιμή της λογικής μεταβλητή *past_sat* τίθεται σε «αληθής», αν η συνθήκη ίσχυε μετά την εκτέλεση της διαδικασίας αποτίμησης των κανόνων, με βάση ένα γεγονός e', που έφτασε κάποια χρονική στιγμή πριν το γεγονός e.
- Για τον χρονικό τελεστή $[always\_been]$, η τιμή της λογικής μεταβλητή *past_sat* τίθεται σε «αληθής», αν η συνθήκη ίσχυε μετά την εκτέλεση της διαδικασίας αποτίμησης των κανόνων, με βάση ένα γεγονός e', που έφτασε κάποια χρονική στιγμή πριν το γεγονός e και δεν άλλαξε ποτέ σε «ψευδής» από τότε.

Με βάση τα παραπάνω, η διαδικασία αποτίμησης των κανόνων, με βάση ένα γεγονός e, πραγματοποιείται ως εξής:

1. Η δομή των κανόνων διατρέχεται:
    I. Για κάθε κανόνα, ελέγχεται αν περιέχει συνθήκες που περιλαμβάνουν την παράμετρο του *context*, στην οποία αναφέρεται το γεγονός e. Αν υπάρχουν τέτοιες εκφράσεις, ενημερώνεται η τιμή του πεδίου *sat* με βάση την νέα τιμή της παραμέτρου.
    II. Για κάθε κανόνα, επιπλέον, ελέγχεται αν περιέχει συνθήκες που συνοδεύονται από παρελθοντικούς τελεστές. Ενημερώνονται τα πεδία όλων των κόμβων, που αντιστοιχούν σε αυτές τις συνθήκες. Αυτό το βήμα πρέπει να εκτελεστεί

ακόμα και για κόμβους που δεν σχετίζονται με την παράμετρο του *context*, που περιέχεται στο γεγονός e. Οι τιμές των μεταβλητών *sat* αυτών των κόμβων, μπορεί να έχουν αλλάξει κατά την διάρκεια της διαδικασίας αποτίμησης των κανόνων, με βάση ένα προηγούμενο γεγονός. Συνεπώς, οι μεταβλητές *past_sat* αυτών των κόμβων, πρέπει να ενημερωθούν κατάλληλα, εφόσον η εφαρμογή βρίσκεται σε μία νέα χρονική κατάσταση.

2. Ελέγχεται, το τμήμα συνθηκών των κανόνων, των οποίων οι κόμβοι ενημερώθηκαν. Αν ισχύει το τμήμα συνθηκών ενός κανόνα, οι ενέργειές του εκτελούνται. Δηλαδή, καλείται η *callback* διεπαφή που υπάρχει στην εφαρμογή, με είσοδο την ταξινομημένη λίστα του τμήματος ενεργειών. Επίσης, ο *RulesEvaluator* φροντίζει για την ικανοποίηση των περιορισμών που επιβάλλονται από τους χρονικούς τελεστές $[until]$ και $[henceforth]$, που μπορεί να υπάρχουν στο τμήμα ενεργειών ενός κανόνα. Έτσι:

   I. Αν μία ενέργεια είναι της μορφής $A[until]B$, ο *RulesEvaluator* εκτελεί τα ακόλουθα βήματα:

      i. Εκτελεί την ενέργεια A.

      ii. Προσθέτει στην δομή των κανόνων ένα ακόμα κόμβο με την συνθήκη B, χωρίς τμήμα ενεργειών.

      iii. Διατρέχει το τμήμα ενεργειών όλων των κανόνων και κλειδώνει τις ενέργειες που περιέχουν τις παραμέτρους της A. Έτσι, αν ικανοποιηθεί κάποιος κανόνας, οι ενέργειές του πραγματοποιούνται, εφόσον δεν είναι κλειδωμένες από κάποια συνθήκη $[until]$.

      iv. Η συνθήκη που προστέθηκε ελέγχεται μαζί με τους υπόλοιπους κανόνες, σε κάθε γεγονός που λαμβάνει η υπηρεσία ενδιάμεσου λογισμικού. Όταν ικανοποιηθεί η συνθήκη B, η δομή των κανόνων διατρέχεται και πάλι και οι ενέργειες που είχαν κλειδωθεί από την συγκεκριμένη συνθήκη ξεκλειδώνονται. Μετά, η συνθήκη αφαιρείται από την δομή των κανόνων

   II. Αν μία ενέργεια είναι της μορφής $[henceforth]A$, εκτελούνται διαδοχικά τα ακόλουθα βήματα:

i. Εκτελείται η ενέργεια A.

ii. Διατρέχεται το τμήμα ενεργειών κάθε κανόνα και αφαιρούνται οι ενέργειες που αναθέτουν τιμή στην παράμετρο της ενέργειας A. Οι κόμβοι που περιέχουν αυτές τις ενέργειες αφαιρούνται από την δομή των κανόνων και έτσι δεν ενεργοποιούνται ποτέ ξανά, μέχρι την επανέναρξη της εφαρμογής και το νέο φόρτωμα των κανόνων.

iii. Αν αφαιρεθούν όλες οι ενέργειες από κάποιο κανόνα, τότε αυτός διαγράφεται από την δομή των κανόνων. Έτσι, μειώνεται το μέγεθος της δομής και οι άσκοπες αναζητήσεις.

iv. Διατρέχεται η δομή των κανόνων και από τους κανόνες που έχουν ενέργειες με την παράμετρο A, αφαιρείται από τις συμβολοσειρές που συμβολίζουν το τμήμα ενεργειών η υποσυμβολοσειρά με την παράμετρο A. Αυτό γίνεται για βελτίωση της απόδοσης, αφού σε μελλοντική αναζήτηση ενεργειών για κλείδωμα, από κάποια συνθήκη $[until]$, η ενέργεια δεν θα βρεθεί στην συμβολοσειρά του κανόνα και δεν θα γίνει η άσκοπη αναζήτηση της ενέργειας στο τμήμα ενεργειών του κανόνα.

3.5 Συμπεράσματα

Στην εργασία [33] προτείνεται μία υπηρεσία ενδιάμεσου λογισμικού που επιτρέπει την εξαγωγή συμπερασμάτων, με βάση το χρονικά συσχετισμένο *context*, σε κινητά συστήματα. Η αρχιτεκτονική της προτεινόμενης υπηρεσίας δεν στηρίζεται στην χρήση μίας κεντρικής μονάδας διαχείρισης του *context* που εκτελείται σε ένα στατικό εξυπηρέτη. Για την αποφυγή των περιορισμών που εισάγονται από μία τέτοια προσέγγιση, η υπηρεσία σχεδιάστηκε και υλοποιήθηκε, ώστε να μπορεί να εκτελείται σε κινητές συσκευές. Όπως έδειξαν οι πειραματικές μετρήσεις, η κατανάλωση των υπολογιστικών πόρων είναι ικανοποιητικά μικρή, ώστε να είναι εφικτή η λειτουργία της σε κινητές συσκευές. Επιπλέον, η χρονική απόκριση της υπηρεσίας είναι αρκετά σύντομη, ώστε να μην γίνεται αντιληπτή η λειτουργία της από τον χρήστη της εφαρμογής.

Όμως, ένα σημαντικό μειονέκτημα της συγκεκριμένης προσέγγισης είναι ότι δεν διαχειρίζεται θέματα ποιότητας της πληροφορίας του *context*. Το μοντέλο αναπαράστασης

και διαχείρισης του *context*, όπως ορίζεται, δεν περιλαμβάνει βασικά γνωρίσματα του *context*, όπως είναι η αβεβαιότητα και η ασάφεια. Με τον όρο αβεβαιότητα εννοείται η έλλειψη ακριβούς πληροφορίας για την λήψη μίας απόφασης, ενώ η ασάφεια σχετίζεται με την αντίληψη που έχει ο καθένας για λεκτικούς προσδιορισμούς ποσοτικών μεγεθών. Αυτό έχει ως αποτέλεσμα, οι αποφάσεις που λαμβάνονται με βάση τα αρχικά δεδομένα να μην είναι αξιόπιστες.

Συγκεκριμένα, οι τιμές του *context*, στα περισσότερα *context – aware* συστήματα, παρέχονται κυρίως από αισθητήρες. Ωστόσο, τα δεδομένα από τους αισθητήρες είναι σχεδόν πάντα ατελή και ελλιπή, λόγω περιορισμών στις υπάρχουσες τεχνολογίες αισθητήρων. Επιπλέον, υπάρχει μεγάλη πιθανότητα να χαθεί μέρος των δεδομένων κατά την μετάδοση, λόγω των ασταθών συνδέσεων στο δίκτυο. Η δεδομένη αμφιβολία για την ακρίβεια του *context* αφορά ζητήματα ποιότητας της πληροφορίας που λαμβάνεται. Το πρόβλημα αυτό είναι γνωστό στην βιβλιογραφία με τον όρο ποιότητα του context (*Quality of Context* ή *QoC*) [26].

Επίσης, ο τρόπος με τον οποίο τα *context – aware* συστήματα προσαρμόζουν τις υπηρεσίες τους στις μεταβολές του περιβάλλοντος λειτουργίας τους, ουσιαστικά, προσομοιώνει τον τρόπο αντίληψης του *context* από τους ίδιους τους χρήστες και το ποιες ενέργειες αυτοί θα εκτελούσαν, υπό αυτές τις συνθήκες, για την βελτίωση της απόδοσης του συστήματος. Όμως, ο μηχανισμός λήψης αποφάσεων από τους χρήστες είναι συνήθως ασαφής. Δηλαδή, η ποσοτικοποίηση των παραμέτρων του *context* από τους χρήστες, για την εξαγωγή συμπερασμάτων για το σύστημα, δεν μπορεί να προσδιοριστεί ακριβώς. Έτσι, ο ακριβής και σαφής προσδιορισμός των τιμών του *context* που επηρεάζουν την λειτουργία του συστήματος είναι μία μη ρεαλιστική διαδικασία, η οποία εισάγει επιπρόσθετη ανακρίβεια στην διαδικασία εξαγωγής νέας γνώσης.

Για τους λόγους που αναφέρθηκαν παραπάνω, τροποποιήσαμε την συγκεκριμένη υπηρεσία ενδιάμεσου λογισμικού, ώστε να μοντελοποιεί την αβεβαιότητα και την ασάφεια στο *context*. Το νέο μοντέλο αναπαράστασης και διαχείρισης του *context* που αναπτύξαμε βασίζεται στην ασαφή λογική. Η θεωρία των ασαφών συνόλων και η ασαφής λογική παρέχουν εργαλεία διαχείρισης της ασάφειας και ένα πλαίσιο συλλογιστικής βασισμένο στην ασάφεια. Η θεωρία των ασαφών συνόλων εισήχθηκε από τον *Zadeh* [51] για την αναπαράσταση κλάσεων των οποίων τα όρια δεν είναι καλά προσδιορισμένα ή είναι ευέλικτα. Επιπλέον, η ασαφής

συλλογιστική επιλύει θέματα που αφορούν την αβεβαιότητα των δεδομένων, εφόσον διαχειρίζεται προσεγγιστικές τιμές στα δεδομένα, μέσω των λεκτικών προσδιορισμών.

ΚΕΦΑΛΑΙΟ 4. ΔΙΑΧΕΙΡΙΣΗ ΤΗΣ ΠΟΙΟΤΗΤΑΣ ΤΟΥ CONTEXT

Σε αυτό το κεφάλαιο περιγράφονται τα χαρακτηριστικά της πληροφορίας του *context* και αναφέρονται οι πιο χαρακτηριστικές εργασίες που ασχολούνται με θέματα ποιότητας του *context*. Η ποιότητα του *context* αναλύεται σε όρους αβεβαιότητας και ασάφειας. Η θεωρία των ασαφών συνόλων και η ασαφής λογική παρέχουν εργαλεία διαχείρισης της αβεβαιότητας και της ασάφειας στο *context*.

4.1 Χαρακτηριστικά του Context

Ένα σημαντικό θέμα στην ανάπτυξη ρεαλιστικών *context – aware* συστημάτων είναι η διαχείριση της ποιότητας του *context*. Τα *context – aware* συστήματα πρέπει να μοντελοποιούν την αβεβαιότητα και την ασάφεια στο *context* και να χρησιμοποιούν τεχνικές λήψης αποφάσεων που διαχειρίζονται αυτά τα θέματα, ώστε να είναι περισσότερα αξιόπιστα, προσαρμόσιμα στις μεταβολές του περιβάλλοντος και στις απαιτήσεις των χρηστών και να παρέχουν ποιοτικότερες υπηρεσίες στους χρήστες.

4.1.1 Είδη Αβεβαιότητας στο Context

Στην ενότητα αυτή, αναφέρονται τρία είδη αβεβαιότητας στο *context* που αφορούν τις παραμέτρους του [27]. Μία παράμετρος για ένα *context – aware* σύστημα είναι:

- **Άγνωστη** όταν καμία πληροφορία δεν είναι διαθέσιμη για αυτή.
- **Ανακριβής** όταν η αναφερόμενη κατάσταση είναι σωστή αλλά αντιστοιχεί σε ανακριβή προσέγγιση της αληθινής κατάστασης (π.χ. λόγω ύπαρξης θορύβου στα αρχικά δεδομένα).
- **Λανθασμένη** όταν υπάρχει διαφορά μεταξύ της πραγματικής και της αναφερόμενης κατάστασης.

Όσον αφορά την προέλευσή τους, οι άγνωστοι παράμετροι οφείλονται σε προβλήματα στις ασύρματες συνδέσεις ή σε αποτυχίες των αισθητήρων. Η ανακρίβεια είναι συνηθισμένη για πληροφορία που προέρχεται από αισθητήρες, ενώ το λανθασμένο *context* προκύπτει ως αποτέλεσμα ανθρώπινων λαθών ή από την χρήση ευριστικών μεθόδων παραγωγής νέας πληροφορίας από τα αρχικά δεδομένα.

Συγκεκριμένα, διακρίνουμε τρεις κύριες πηγές του *context*: τους **αισθητήρες**, τους **χρήστες** και τους **μηχανισμούς παραγωγής** *context* από άλλη πληροφορία. Το *context* που προέρχεται από τους αισθητήρες μεταβάλλεται συχνά και χαρακτηρίζεται από προβλήματα ανακρίβειας και χρονικής συνέπειας. Τα προβλήματα αυτά οφείλονται σε λάθη και αποτυχίες των αισθητήρων, σε διακοπές στο δίκτυο και σε καθυστερήσεις που εισάγονται από την κατανομή του *context*.

Το *context* που δίνεται από τον χρήστη είναι στατικό ή δυναμικό. Το στατικό *context* εισάγεται από τον σχεδιαστή του συστήματος στην φάση ανάπτυξής του και δεν αντιμετωπίζει κάποια ιδιαίτερα προβλήματα ποιότητας. Το δυναμικό *context* χρήστη λαμβάνεται απευθείας από τον χρήστη (π.χ. προφίλ του χρήστη) ή έμμεσα από τις εφαρμογές που χρησιμοποιεί (π.χ. από λογισμικό που διατηρεί ιστορικό των ενεργειών του χρήστη). Αυτό το είδος πληροφορίας είναι συχνά ασαφές, ελλιπές ή άγνωστο.

Τέλος, τα χαρακτηριστικά του παραγόμενου *context* προσδιορίζονται από εκείνα της αρχικής πληροφορίας. Ωστόσο, επιπλέον λάθη και ανακρίβεια εισάγονται από την διαδικασία

παραγωγής του νέου *context*. Αυτό συμβαίνει ιδιαίτερα, όταν δεν λαμβάνεται σε αυτή υπόψη η αβεβαιότητα της αρχικής πληροφορίας.

4.1.2 Ασάφεια στο Context

Η ασάφεια είναι έννοια που σχετίζεται με την ποσοτικοποίηση της πληροφορίας του *context* και οφείλεται κυρίως σε ανακριβή δεδομένα. Για παράδειγμα, η φράση «Η μπαταρία είναι αρκετά φορτισμένη», αν και δεν προσδιορίζει με ακρίβεια την φόρτιση της μπαταρίας, επιτρέπει να ληφθούν ορισμένες αποφάσεις για την λειτουργία μίας κινητής συσκευής. Το πρόβλημα σε αυτές τις περιπτώσεις δεν οφείλεται τόσο στις έννοιες που χρησιμοποιούνται, όσο στο ότι η αντίληψη που έχει ο καθένας για τέτοιους λεκτικούς προσδιορισμούς ποσοτικών μεγεθών είναι ασαφής. Αν και είναι δυνατό να αποδοθούν συγκεκριμένες τιμές σε λεκτικά προσδιοριζόμενα μεγέθη των παραμέτρων του *context*, για να περιοριστεί η ασάφεια, αυτό οδηγεί σε λάθος αποφάσεις.

Στα *context – aware* συστήματα ένα μεγάλο ποσοστό του *context* παρέχεται από αισθητήρες. Οι τιμές που λαμβάνονται από τους αισθητήρες διαβαθμίζονται με βάση κάποια κλίμακα, ώστε να λαμβάνονται αποφάσεις σχετικά με την λειτουργία του συστήματος. Η διαβάθμιση αυτή ορίζεται στο μοντέλο του *context*. Δεδομένου ότι το μοντέλο του *context* πρέπει να είναι αρκετά ρεαλιστικό, θα πρέπει η διαβάθμιση αυτή να προσομοιώνει τον ασαφή τρόπο ποσοτικοποίησης του *context*, από τους ίδιους τους ανθρώπους. Επομένως, είναι λογικό να έχουμε ασαφή αναπαράσταση και χρήση του *context* στην ανάπτυξη *context – aware* συστημάτων.

Η θεωρία των ασαφών συνόλων και η ασαφής λογική παρέχουν μηχανισμούς που μπορούν να χρησιμοποιηθούν για την αναπαράσταση του *context* και την εξαγωγή συμπερασμάτων με βάση αυτό. Επιπλέον, οι μέθοδοι ασαφούς λογικής διαχειρίζονται και θέματα που αφορούν την ανακρίβεια των τιμών του *context*. Αυτό είναι ένα βασικό θέμα στην ανάπτυξη *context – aware* συστημάτων, εφόσον οι τιμές που παρέχονται από τους αισθητήρες αντιστοιχούν, σχεδόν πάντα, σε ανακριβή προσέγγιση των πραγματικών τιμών του *context*. Έτσι, ο ακριβής προσδιορισμός των τιμών του *context* που επηρεάζουν την λειτουργία του συστήματος, εμπεριέχει τον κίνδυνο λήψης λανθασμένων αποφάσεων. Το πρόβλημα αυτό επιλύεται στην ασαφή λογική, μέσω της χρήσης λεκτικών προσδιορισμών για το *context*.

4.2 Μέθοδοι Διαχείρισης της Αβεβαιότητας στο Context

Μέχρι σήμερα έχει γίνει σημαντική έρευνα σχετικά με την αβεβαιότητα του *context*. Στις πρώτες προσπάθειες, η αβεβαιότητα μοντελοποιήθηκε χρησιμοποιώντας διάφορους δείκτες, όπως ατέλεια, ανακρίβεια, κλπ [16, 22, 31, 34]. Οι προσεγγίσεις αυτές έχουν σημαντικά μειονεκτήματα. Έχουν την δυνατότητα να εκφράσουν μόνο απλούς τύπους *context*. Επιπλέον, δεν χρησιμοποιούν μηχανισμούς εξαγωγής νέας και πιο σύνθετης πληροφορίας που λαμβάνουν υπόψη την αβεβαιότητα στο αρχικό *context*. Ένα άλλο μειονέκτημά τους είναι το ότι επιβαρύνουν τον χρήστη, καθώς ο χρήστης μεσολαβεί στην διαδικασία προσδιορισμού των χαρακτηριστικών ποιότητας του *context*. Για τους λόγους αυτούς, σε πρόσφατες έρευνες χρησιμοποιούνται πιθανοτικά μοντέλα, όπως Μπεϋζιανά δίκτυα, για την μοντελοποίηση του *context* και την εξαγωγή συμπερασμάτων με βάση την αβεβαιότητα σε αυτό.

4.2.1 Πιθανοτική Λογική και Μπεϋζιανά Δίκτυα

Για την διαχείριση της αβεβαιότητας στο *context*, μία από τις πιο χρήσιμες προσεγγίσεις είναι το θεώρημα του *Bayes*. Η εξαγωγή συμπερασμάτων με βάση το θεώρημα του *Bayes* στηρίζεται στην ιδέα ότι, τα παρατηρούμενα γεγονότα (δηλ. οι παράμετροι του *context*) μπορούν να χρησιμοποιηθούν για τον υπολογισμό των πιθανοτήτων των αγνώστων γεγονότων. Όμως, οι πιθανοτικές εξαρτήσεις σε συστήματα με πολλές παραμέτρους είναι πολύπλοκες στην διαχείρισή τους (*π.χ.* για n δυαδικές μεταβλητές, χρειάζεται ο προσδιορισμός 2^n πιθανοτήτων). Για τον λόγο αυτό, συνήθως χρησιμοποιούνται Μπεϋζιανά δίκτυα.

Ένα Μπεϋζιανό δίκτυο είναι ένα πιθανοτικό μοντέλο που αποτελείται από ένα πεπερασμένο σύνολο τυχαίων μεταβλητών. Οι κόμβοι του γραφήματος αναπαριστούν τις τυχαίες μεταβλητές και οι ακμές τις πιθανοτικές εξαρτήσεις μεταξύ αυτών. Οι εξαρτήσεις ποσοτικοποιούνται μέσω ενός συνόλου κατανομών πιθανότητας υπό-συνθήκη. Το γινόμενο όλων των κατανομών πιθανότητας υπό-συνθήκη είναι η από κοινού πιθανότητα του Μπεϋζιανού δικτύου. Η πιθανότητα αυτή χρησιμοποιείται για τον υπολογισμό οποιαδήποτε άλλης πιθανότητας, όπως η εκ των υστέρων πιθανότητα κάθε μεταβλητής, δεδομένου ενός συνόλου παρατηρήσεων.

Βέβαια, προκύπτουν αρκετά θέματα στην χρήση Μπεϋζιανών τεχνικών για την επεξεργασία του *context*:

1. Οι ειδικοί στον τομέα εφαρμογής πρέπει να καθορίσουν το πλήθος και το είδος των μεταβλητών του *context*, καθώς και κάποιες αρχικές εξαρτήσεις μεταξύ αυτών. Επιπλέον, είναι αυτοί που επιβεβαιώνουν την δομή και τις πιθανότητες που υπολογίζονται στην φάση της εκπαίδευσης του μοντέλου.
2. Πρέπει να χρησιμοποιηθούν τεχνικές εκμάθησης της δομής και των παραμέτρων (δηλ. αρχικές πιθανότητες, πιθανότητες υπό-συνθήκη) του δικτύου. Στα *context – aware* συστήματα, η εκπαίδευση πρέπει να είναι μία συνεχής διαδικασία καθώς συλλέγονται διαρκώς νέα δεδομένα.
3. Με βάση το μέγεθος και την δομή του δικτύου, πρέπει να επιλεγούν κατάλληλοι αλγόριθμοι για την εξαγωγή των εκ των υστέρων και από κοινού πιθανοτήτων.

Πρόσφατα, κάποιες ερευνητικές προσεγγίσεις έχουν χρησιμοποιήσει Μπεϋζιανά δίκτυα για την μοντελοποίηση της αβεβαιότητας στο *context* και την εξαγωγή συμπερασμάτων για την κατάσταση του συστήματος με βάση αυτή. Στις εργασίες [24, 39, 49] με χρήση κατάλληλων εργαλείων λογισμικού κατασκευάζεται η δομή και εκπαιδεύεται το Μπεϋζιανό δίκτυο για να υπολογιστούν οι παράμετροί του. Η γνώση για το πεδίο της εκάστοτε εφαρμογής λαμβάνεται από μία οντολογία που αναπαριστά το *context*. Στην συνέχεια, οι διάφορες υπηρεσίες της κάθε εφαρμογής χρησιμοποιούν το Μπεϋζιανό δίκτυο για τον υπολογισμό των πιθανοτήτων των γεγονότων του *context*.

4.3 Θεωρία Ασαφών Συνόλων και Ασαφής Λογική

Η ασαφής λογική είναι ένα υπερσύνολο της κλασικής λογικής, η οποία έχει επεκταθεί ώστε να μπορεί να διαχειριστεί τιμές αλήθειας μεταξύ του «απολύτως αληθούς» και του «απολύτως ψευδούς». Βασίζεται στη θεωρία των ασαφών συνόλων [51]. Η κύρια ιδέα της θεωρίας αυτής είναι η ότι η «διαδικασία μετατροπής διακριτών μεγεθών σε ασαφή επιτρέπει την γενίκευση μίας διακριτής θεωρίας σε συνεχή».

4.3.1 Βασικές Έννοιες Ασαφών Συνόλων

Ένα ασαφές σύνολο A ορίζεται ως ένα σύνολο διατεταγμένων ζευγών $\left(x, u_A(x)\right)$, όπου $x \in X$ και $u_A(x) \in [0,1]$. Το σύνολο X αποτελεί ένα ευρύτερο σύνολο αναφοράς που περιλαμβάνει όλα τα αντικείμενα στα οποία μπορεί να γίνει αναφορά. Συνήθως, το X είναι υποσύνολο του συνόλου των πραγματικών αριθμών. Η τιμή $u_A(x)$ λέγεται βαθμός αλήθειας και εκφράζει τον βαθμό συμμετοχής του x στο ασαφές σύνολο A. Τέλος, η συνάρτηση u_A ονομάζεται συνάρτηση συμμετοχής. Η συνάρτηση συμμετοχής μπορεί να προέρχεται από: (α) αντικειμενικές εκτιμήσεις, (β) προκαθορισμένες μορφές συναρτήσεων, (γ) συχνότητες εμφανίσεων και πιθανότητες, (δ) φυσικές μετρήσεις και (ε) διαδικασίες μάθησης και προσαρμογής.

Η διαφορά των ασαφών συνόλων συγκριτικά με την κλασική θεωρία συνόλων είναι ότι στην δεύτερη ισχύει ότι $u_A(x) \in \{0,1\}$. Δηλαδή, η θεωρία ασαφών συνόλων μεταπίπτει στην αντίστοιχη κλασική, όταν οι δυνατές τιμές της συνάρτησης συμμετοχής είναι μόνο 0 και 1.

4.3.2 Ιδιότητες Ασαφών Συνόλων

Για τα ασαφή σύνολα ορίζονται πράξεις και ισχύουν ιδιότητες ανάλογες με αυτές που ισχύουν στα κλασικά σύνολα. Ορισμένες από αυτές, όπως η ένωση και η τομή ορίζονται μέσω των τελεστών *max* και *min* που συμβολίζονται με $\vee$ και $\wedge$, αντίστοιχα. Το σύμβολο του εκάστοτε τελεστή μπορεί να γραφεί και στην αρχή των στοιχείων του συνόλου (προθεματική μορφή) ως εξής: $u = \wedge(u_1, u_2, ..., u_n) = \bigwedge_{k=1}^{n}(u_k)$.

Ισχύουν οι ακόλουθες ιδιότητες:

- Ένα ασαφές σύνολο ονομάζεται κανονικό αν υπάρχει τουλάχιστον ένα σημείο x_0 του πεδίου ορισμού του για το οποίο ισχύει $u(x_0) = 1$.
- Δύο ασαφή σύνολα A και B ορισμένα στο ευρύτερο σύνολο αναφοράς X είναι ίσα αν οι συναρτήσεις συμμετοχής τους είναι ίσες σε όλο το πεδίο ορισμού τους (Εξίσωση (4.1)).

$$A = B \text{ αν } u_A(x) = u_B(x), \quad \forall x \in X \qquad \text{Εξ. 4.1}$$

- Το συμπληρωματικό ενός ασαφούς συνόλου A είναι το $\bar{A}$. Η συνάρτηση συμμετοχής του δίνεται από την Εξίσωση (4.2).

$$u_{\bar{A}}(x) = 1 - u_A(x) \qquad \text{Εξ. 4.2}$$

- Για δύο ασαφή σύνολα A και B ορισμένα στο X, το A είναι υποσύνολο του B ($A \subseteq B$) αν ισχύει η Εξίσωση (4.3).

$$u_A(x) \le u_B(x), \quad \forall x \in X \qquad \text{Εξ. 4.3}$$

- Η ένωση δύο ασαφών συνόλων A και B ορισμένων στο X είναι ένα νέο ασαφές σύνολο $A \cup B$ ή $A\,OR\,B$ ορισμένο επίσης στο X. Η συνάρτηση συμμετοχής του δίνεται από την Εξίσωση (4.4).

$$A \cup B : u_{A \cup B}(x) = \vee\left(u_A(x), u_B(x)\right) = \max\left(u_A(x), u_B(x)\right), \quad \forall x \in X \qquad \text{Εξ. 4.4}$$

- Η τομή δύο ασαφών συνόλων A και B ορισμένων στο X είναι ένα νέο ασαφές σύνολο $A \cap B$ ή $A\,AND\,B$ ορισμένο επίσης στο X. Η συνάρτηση συμμετοχής του δίνεται από την Εξίσωση (4.5).

$$A \cap B : u_{A \cap B}(x) = \wedge\left(u_A(x), u_B(x)\right) = \min\left(u_A(x), u_B(x)\right), \quad \forall x \in X \qquad \text{Εξ. 4.5}$$

- Κενό ασαφές σύνολο $\varnothing$, είναι αυτό για το οποίο η συνάρτηση συμμετοχής έχει την τιμή 0 σε όλο το πεδίο ορισμού του.

- Το γινόμενο δύο ασαφών συνόλων A και B ορισμένων στο X είναι ένα νέο ασαφές σύνολο του οποίου η συνάρτηση συμμετοχής ισούται με το αλγεβρικό γινόμενο των αντίστοιχων συναρτήσεων των A και B (Εξίσωση (4.6)).

$$u_{A\cdot B}(x)=u_A(x)u_B(x),\quad \forall x\in X \qquad \text{Εξ. 4.6}$$

- Το γινόμενο ενός πραγματικού αριθμού a με ένα ασαφές σύνολο A, δίνει ένα νέο ασαφές σύνολο $a\cdot A$. Η συνάρτηση συμμετοχής του δίνεται από την Εξίσωση (4.7).

$$u_{a\cdot A}(x)=au_A(x),\quad \forall x\in X \qquad \text{Εξ. 4.7}$$

- Ένα ασαφές σύνολο μπορεί να υψωθεί στην δύναμη a (θετικός πραγματικός αριθμός), υψώνοντας στο a την συνάρτηση συμμετοχής του. Προκύπτει έτσι ένα νέο ασαφές σύνολο με συνάρτηση συμμετοχής που δίνεται από την Εξίσωση (4.8).

$$u_{A^a}(x)=\left(u_A(x)\right)^a,\quad \forall x\in X \qquad \text{Εξ. 4.8}$$

Η ύψωση ενός ασαφούς συνόλου στην δεύτερη δύναμη $a=2$, λαμβάνεται συνήθως ως μεταβολή της λεκτικής τιμής που αντιστοιχεί στο ασαφές σύνολο με τον όρο «πολύ».

Η ένωση και η τομή δύο ασαφών συνόλων ορίστηκαν μέσω των τελεστών *max* και *min* που συμβολίζονται με $\vee$ και $\wedge$, αντίστοιχα. Οι τελεστές αυτοί ορίστηκαν από τον *Zadeh* [51] και ανήκουν στην κλάση των $S-norm$ και $T-norm$ συναρτήσεων, αντίστοιχα. Συγκεκριμένα, η ένωση δύο ασαφών συνόλων A και B ορισμένων στο X προσδιορίζεται μέσω της κλάσης συναρτήσεων $S:[0,1]\times[0,1]\to[0,1]$ με βάση την Εξίσωση (4.9).

$$u_{A\cup B}(x)=S\left(u_A(x),u_B(x)\right),\quad \forall x\in X \qquad \text{Εξ. 4.9}$$

Η κλάση αυτών των συναρτήσεων αναφέρεται ως $S-norm$ [17]. Στην συνέχεια, δίνονται τρεις συναρτήσεις αυτής της κλάσης (Εξισώσεις (4.10 – 4.12)) που χρησιμοποιούνται για τον προσδιορισμό της συνάρτησης συμμετοχής της ένωσης δύο ασαφών συνόλων.

$$u_{A\cup B}(x)=\max\left(u_A(x),u_B(x)\right)=u_A(x)\vee u_B(x),\quad \forall x\in X \qquad \text{Εξ. 4.10}$$

$$u_{A\cup B}(x)=u_A(x)+u_B(x)-u_A(x)u_B(x),\quad \forall x\in X \qquad \text{Εξ. 4.11}$$

$$u_{A\cup B}(x)=1\wedge(u_A(x)+u_B(x)),\quad \forall x\in X \qquad \text{Εξ. 4.12}$$

Ομοίως, η τομή δύο ασαφών συνόλων A και B ορισμένων στο X προσδιορίζεται μέσω της κλάσης συναρτήσεων $T:[0,1]\times[0,1]\to[0,1]$ με βάση την Εξίσωση (4.13).

$$u_{A\cap B}(x)=T\left(u_A(x),u_B(x)\right),\quad \forall x\in X \qquad \text{Εξ. 4.13}$$

Η κλάση αυτών των συναρτήσεων αναφέρεται ως $T-$ *norm* [17]. Στις Εξισώσεις (4.14 – 4.16) φαίνονται τρία ενδεικτικά παραδείγματα συναρτήσεων αυτής της κλάσης που χρησιμοποιούνται ως τελεστές για τον προσδιορισμό της τομής δύο ασαφών συνόλων.

$$u_{A\cap B}(x)=\min\left(u_A(x),u_B(x)\right)=u_A(x)\wedge u_B(x),\quad \forall x\in X \qquad \text{Εξ. 4.14}$$

$$u_{A\cap B}(x)=u_A(x)u_B(x),\quad \forall x\in X \qquad \text{Εξ. 4.15}$$

$$u_{A\cap B}(x)=0\vee(u_A(x)+u_B(x)-1),\quad \forall x\in X \qquad \text{Εξ. 4.16}$$

Όπως και στην κλασική θεωρία συνόλων, έτσι και στην αντίστοιχη ασαφή θεωρία, ισχύουν: (α) ο νόμος της διπλής άρνησης, (β) η ταυτοδυναμία, (γ) η αντιμεταθετική ιδιότητα, (δ) η προσεταιριστική ιδιότητα, (ε) η επιμεριστική ιδιότητα, (στ) η απορροφητική ιδιότητα και (ζ) ο νόμος του *De Morgan*. Όλες οι παραπάνω ιδιότητες, μπορούν να εκφραστούν χρησιμοποιώντας τις συναρτήσεις συμμετοχής των ασαφών συνόλων αντί για τα ίδια τα ονόματά τους.

Υπάρχουν μερικές ιδιότητες της κλασικής θεωρίας συνόλων που δεν ισχύουν στην περίπτωση των ασαφών συνόλων. Για παράδειγμα, η σχέση $A\cap\overline{A}=\varnothing$ που ισχύει στην κλασική θεωρία συνόλων, στην θεωρία των ασαφών συνόλων γίνεται: $A\cap\overline{A}\neq\varnothing$. Παρόμοια, η σχέση

$A \cup \bar{A} = X$ που στην κλασική θεωρία συνόλων αποτελεί τον νόμο του αποκλειόμενου μέσου, στην θεωρία των ασαφών συνόλων γίνεται: $A \cup \bar{A} \neq X$.

4.3.3 Συναρτήσεις Συμμετοχής

Ένα ασαφές σύνολο χαρακτηρίζεται πλήρως από μία συνάρτηση συμμετοχής. Επειδή τα περισσότερα ασαφή σύνολα, στην πράξη, ορίζονται στο σύνολο των πραγματικών αριθμών, οι συναρτήσεις συμμετοχής ορίζονται με βάση μαθηματικές συναρτήσεις. Στην συνέχεια, αναφέρονται οι κλάσεις συναρτήσεων που χρησιμοποιούνται, συνήθως, για τον ορισμό των συναρτήσεων συμμετοχής.

Τριγωνική Συνάρτηση

Η τριγωνική συνάρτηση ορίζεται από την Εξίσωση (4.17). Οι παράμετροι $\{a,b,c\}$ με $a \le b \le c$ προσδιορίζουν τις συντεταγμένες στον άξονα x των τριών γωνιών της τριγωνικής συνάρτησης.

$$trimf(x;a,b,c) = \begin{cases} 0, & x \le a. \\ \dfrac{x-a}{b-a}, & a \le x \le b. \\ \dfrac{c-x}{c-b}, & b \le x \le c. \\ 0, & c \le x. \end{cases} \qquad \text{Εξ. 4.17}$$

Τραπεζοειδής Συνάρτηση

Η τραπεζοειδής συνάρτηση ορίζεται από την Εξίσωση (4.18).

$$trapmf(x;a,b,c,d) = \begin{cases} 0, & x \le a. \\ \dfrac{x-a}{b-a}, & a \le x \le b. \\ 1, & b \le x \le c. \\ \dfrac{d-x}{d-c}, & c \le x \le d. \\ 0, & d \le x. \end{cases} \qquad \text{Εξ. 4.18}$$

Οι παράμετροι $\{a,b,c,d\}$ με $a \le b \le c \le d$ προσδιορίζουν τις συντεταγμένες στον άξονα x των τεσσάρων γωνιών της τραπεζοειδούς συνάρτησης.

Gaussian Συνάρτηση

Η συνάρτηση Gaussian ορίζεται από την Εξίσωση (4.19). Η Γκαουσιανή συνάρτηση προσδιορίζεται πλήρως από τις παραμέτρους c και σ. Η παράμετρος c αναπαριστά το κέντρο, ενώ η παράμετρος σ το πλάτος της συνάρτησης.

$$gaussmf(x;c,\sigma) = e^{-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2} \qquad \text{Εξ. 4.19}$$

Συνάρτηση Cauchy

Η συνάρτηση *Cauchy* ορίζεται από την Εξίσωση (4.20), όπου η παράμετρος b είναι συνήθως θετική.

$$gbellmf(x;a,b,c) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}} \qquad \text{Εξ. 4.20}$$

Σιγμοειδής Συνάρτηση

Η σιγμοειδής συνάρτηση ορίζεται από την Εξίσωση (4.21).

$$sigmf(x;a,c) = \frac{1}{1+e^{-a(x-c)}} \qquad \text{Εξ. 4.21}$$

4.3.4 Ασαφείς Σχέσεις

Οι ασαφείς σχέσεις είναι ασαφή σύνολα ορισμένα σε πεδία αναφοράς μεγαλύτερης διάστασης (π.χ. $X \times X$, $X \times Y \times Z$, κλπ). Οι ασαφείς σχέσεις μπορούν να συνδυαστούν μεταξύ τους μέσω της διαδικασίας της σύνθεσης. Αν για παράδειγμα συνδυαστεί η ασαφής σχέση $R_1(x,y)$ ορισμένη στο $X \times Y$, με την ασαφή σχέση $R_2(y,z)$ ορισμένη στο $Y \times Z$, τότε θα προκύψει μία ασαφής σχέση $R(x,z)$, η οποία θα ορίζεται στο σύνολο $X \times Z$ και θα συσχετίζει άμεσα στοιχεία των συνόλων X και Z. Βέβαια, είναι απαραίτητο να

προσδιοριστεί επακριβώς η συνάρτηση συμμετοχής $u_R(x,z)$ της R, με χρήση των συναρτήσεων συμμετοχής των R_1 και R_2.

Η σύνθεση είναι πολύ σημαντική διαδικασία καθώς οι κανόνες της μορφής *if – then* αντιστοιχούν σε ασαφείς σχέσεις και το πρόβλημα της ασαφούς συλλογιστικής είναι μαθηματικά ισοδύναμο με την σύνθεση. Οι περισσότερο γνωστές μέθοδοι σύνθεσης ασαφών σχέσεων είναι η σύνθεση *max – min* [51] και η σύνθεση *max – product* [51]. Αν $R_1(x,y)$ και $R_2(y,z)$ είναι δύο ασαφείς σχέσεις ορισμένες στα σύνολα $X \times Y$ και $Y \times Z$, αντίστοιχα, τότε η σύνθεση τους δίνει μία νέα σχέση $R_1 \circ R_2$ ορισμένη στο $X \times Z$. Η συνάρτηση συμμετοχής της για την περίπτωση της *max – min* σύνθεσης δίνεται από την Εξίσωση (4.22), ενώ για την περίπτωση της *max – product* σύνθεσης δίνεται από την Εξίσωση (4.23).

$$u_{R_1 \circ R_2}(x,z) = \vee_y \left(u_{R_1}(x,y) \wedge u_{R_2}(y,z) \right) \qquad \text{Εξ. 4.22}$$

$$u_{R_1 \circ R_2}(x,z) = \vee_y \left(u_{R_1}(x,y) u_{R_2}(y,z) \right) \qquad \text{Εξ. 4.23}$$

4.3.5 Ασαφείς Μεταβλητές, Προτάσεις και Κανόνες

Μία μεταβλητή της οποίας οι τιμές ορίζονται με χρήση ασαφών συνόλων ονομάζεται ασαφής μεταβλητή. Τα ασαφή σύνολα αποτελούν το πεδίο τιμών της ασαφούς μεταβλητής, για αυτό ονομάζονται και λεκτικές τιμές, ενώ η ασαφής μεταβλητή ονομάζεται και λεκτική μεταβλητή. Οι ασαφείς μεταβλητές χρησιμοποιούνται σε ασαφείς κανόνες και στην ασαφή διαδικασία εξαγωγής συμπερασμάτων.

Ένας ασαφής κανόνας είναι μία υπό συνθήκη έκφραση που συσχετίζει δύο ή περισσότερες ασαφείς προτάσεις. Μία ασαφής πρόταση θέτει μία τιμή σε μία ασαφή μεταβλητή. Στην πιο απλή εκδοχή, ένας ασαφής κανόνας έχει την μορφή: *if* X *is* A *then* Y *is* B, όπου A, B ασαφή σύνολα ορισμένα στο X και Y, αντίστοιχα. Η αναλυτική περιγραφή ενός ασαφούς κανόνα είναι μία ασαφής σχέση $R(x,y)$ ορισμένη στο $X \times Y$ που ονομάζεται σχέση επαγωγής και η οποία προκύπτει με κατάλληλο συνδυασμό των συναρτήσεων συμμετοχής των ασαφών συνόλων του τμήματος συνθηκών και ενεργειών του κανόνα.

Στην γενική της μορφή, η σχέση επαγωγής ορίζεται από την Εξίσωση (4.24). Η συνάρτηση φ ονομάζεται τελεστής επαγωγής και υποδεικνύει τον ακριβή τρόπο με τον οποίο πρέπει να συνδυαστούν οι συναρτήσεις συμμετοχής του τμήματος συνθηκών και ενεργειών ενός ασαφούς κανόνα, ώστε να προκύψει η αναλυτική του έκφραση. Ο Πίνακας 4.1 περιέχει τις πιο σημαντικές από τις εκφράσεις που έχουν προταθεί για τον τελεστή επαγωγής. Η σχέση επαγωγής χρησιμοποιείται κατά την διαδικασία εξαγωγής συμπερασμάτων από ασαφείς κανόνες.

$$R(x,y) \equiv u_R(x,y) = \varphi\left(u_A(x), u_B(y)\right) \qquad \text{Εξ. 4.24}$$

Πίνακας 4.1: Παραδείγματα ασαφών τελεστών επαγωγής.

Ονομασία Τελεστή	Αναλυτική Έκφραση
φ_m : Zadeh Max - Min	$\left(u_A(x) \wedge u_B(y)\right) \vee \left(1 - u_A(x)\right)$
φ_c : Mamdani Min	$u_A(x) \wedge u_B(y)$
φ_p : Larsen Product	$u_A(x) u_B(y)$
φ_a : Arithmetic	$1 \wedge \left(1 - u_A(x) + u_B(y)\right)$
φ_b : Boolean	$\left(1 - u_A(x)\right) \vee u_B(y)$

4.3.6 Ασαφής Συλλογιστική

Ο όρος ασαφής συλλογιστική αφορά την εξαγωγή συμπερασμάτων από ασαφείς κανόνες. Η χρήση ασαφούς συλλογιστικής απαιτεί την ύπαρξη μίας ασαφούς λεκτικής περιγραφής του προβλήματος. Δηλαδή, την περιγραφή του προβλήματος με ασαφείς μεταβλητές, ασαφείς τιμές και ασαφείς κανόνες.

Ένα πρόβλημα ασαφούς συλλογιστικής στην πιο απλή του μορφή ορίζεται ως εξής:

if X is A then Y is B,
X is A' Y is B'(?)

Τα ασαφή σύνολα A, A' έχουν το ίδιο πεδίο ορισμού, όπως και τα ασαφή σύνολα B, B'.Το ζητούμενο είναι ο υπολογισμός της τιμής B' της ασαφούς μεταβλητής y μέσω του ασαφούς κανόνα και της γνωστής τιμής A' της ασαφούς μεταβλητής x. Το πρόβλημα αυτό διευθετείται μέσω της συλλογιστικής διαδικασίας *Generalized Modus Ponens* (*GMP*) [52] (Εξίσωση (4.25)). Η διαδικασία *GMP* εφαρμόζεται και στην γενική περίπτωση προβλημάτων ασαφούς συλλογιστικής. Στα προβλήματα αυτά υπάρχει ένα σύνολο κανόνων, των οποίων το τμήμα συνθηκών αποτελείται από ασαφείς προτάσεις που σχετίζονται με λογικές διαζεύξεις ή συζεύξεις. Συγκεκριμένα, το τμήμα συνθηκών ορίζεται σε *DNF* ή σε *CNF* μορφή. Το τμήμα ενεργειών αποτελείται από μία η περισσότερες ενέργειες.

$$B' = A' \circ R(x, y) \qquad \text{Εξ. 4.25}$$

Η διαδικασία *GMP* μπορεί να διαιρεθεί στα ακόλουθα τέσσερα στάδια για τον υπολογισμό της τιμής κάθε άγνωστης ασαφούς μεταβλητής:

1. Για κάθε ασαφή πρόταση του τμήματος συνθηκών κάθε κανόνα υπολογίζεται η σύνθεση της τιμής της ασαφούς μεταβλητής με την γνωστή τιμή της. Λόγω του ότι πρόκειται για σύνθεση μονοδιάστατων σχέσεων, το αποτέλεσμα της σύνθεσης είναι μία αριθμητική τιμή.
2. Για κάθε κανόνα συνδυάζονται οι αριθμητικές τιμές του τμήματος συνθηκών που προέκυψαν από το προηγούμενο στάδιο με χρήση των τελεστών *S – norm* και *T – norm*. Ο τελεστής *S – norm* χρησιμοποιείται όταν οι αντίστοιχες ασαφείς προτάσεις σχετίζονται με λογική διάζευξη, ενώ ο τελεστής *T – norm* χρησιμοποιείται όταν οι αντίστοιχες ασαφείς προτάσεις σχετίζονται με λογική σύζευξη. Έτσι, προκύπτει μία αριθμητική τιμή για το τμήμα συνθηκών κάθε κανόνα. Η τιμή αυτή ονομάζεται βάρος του κανόνα.
3. Για κάθε ασαφή πρόταση του τμήματος ενεργειών κάθε κανόνα υπολογίζεται η σχέση επαγωγής μεταξύ της τιμής της ασαφούς μεταβλητής και του βάρους του κανόνα. Δηλαδή, μεταβάλλεται η συνάρτηση συμμετοχής του αντίστοιχου ασαφούς συνόλου με βάση το βάρος του κανόνα.

4. Για κάθε ασαφή μεταβλητή των ενεργειών όλων των κανόνων υπολογίζεται η ένωση των τιμών της, που προέκυψαν από το προηγούμενο βήμα, με χρήση ενός τελεστή της *S – norm* κλάσης.

4.3.7 Συστήματα Ασαφούς Λογικής

Ένα σύστημα ασαφούς λογικής είναι ένα υπολογιστικό πλαίσιο που βασίζεται στην θεωρία των ασαφών συνόλων, στους ασαφείς κανόνες και στην ασαφή συλλογιστική. Η βασική δομή ενός συστήματος ασαφούς λογικής περιλαμβάνει τρία κύρια συστατικά: (α) μία βάση δεδομένων με τους ασαφείς κανόνες, (β) μία βάση δεδομένων που περιέχει τις συναρτήσεις συμμετοχής των ασαφών συνόλων που χρησιμοποιούνται στους κανόνες και (γ) ένα μηχανισμό ασαφούς συλλογιστικής που εκτελεί την διαδικασία εξαγωγής συμπερασμάτων με χρήση των κανόνων και των δοθέντων γεγονότων.

Η είσοδος ενός συστήματος ασαφούς λογικής είναι είτε ασαφής είτε διακριτή, αλλά η έξοδός του είναι πάντα ένα ή περισσότερα ασαφή σύνολα. Ωστόσο, μερικές φορές είναι χρήσιμο να δίνει ως έξοδο διακριτές τιμές. Απαιτείται όμως μεθοδολογία αποασαφοποίησης, η οποία εξάγει την τιμή που αναπαριστά καλύτερα ένα ασαφές σύνολο. Με μη – ασαφείς εισόδους και εξόδους, ένα σύστημα ασαφούς λογικής εκτελεί μία μη – γραμμική απεικόνιση από τον χώρο των δεδομένων εισόδου στο χώρο των δεδομένων εξόδου. Η απεικόνιση αυτή πραγματοποιείται με βάση τους ασαφείς κανόνες, καθένας από τους οποίους περιγράφει την τοπική συμπεριφορά της απεικόνισης.

Στην συνέχεια, περιγράφονται τρία μοντέλα συστημάτων ασαφούς λογικής που έχουν χρησιμοποιηθεί σε αρκετές εφαρμογές. Οι διαφορές μεταξύ αυτών των συστημάτων βρίσκονται στο τμήμα ενεργειών των κανόνων.

Το Μοντέλο Ασαφούς Λογικής Mamdani

Ένα σύστημα ασαφούς λογικής που βασίζεται στο μοντέλο *Mamdani* [36] ακολουθεί την διαδικασία ασαφούς συλλογιστικής *GMP*. Όμως, πριν την εφαρμογή αυτής της διαδικασίας χρειάζεται να οριστούν οι συναρτήσεις για τους τελεστές που χρησιμοποιούνται σε κάθε στάδιο. Όσον αφορά τους τελεστές της ένωσης και της τομής, χρησιμοποιείται η κατάλληλη

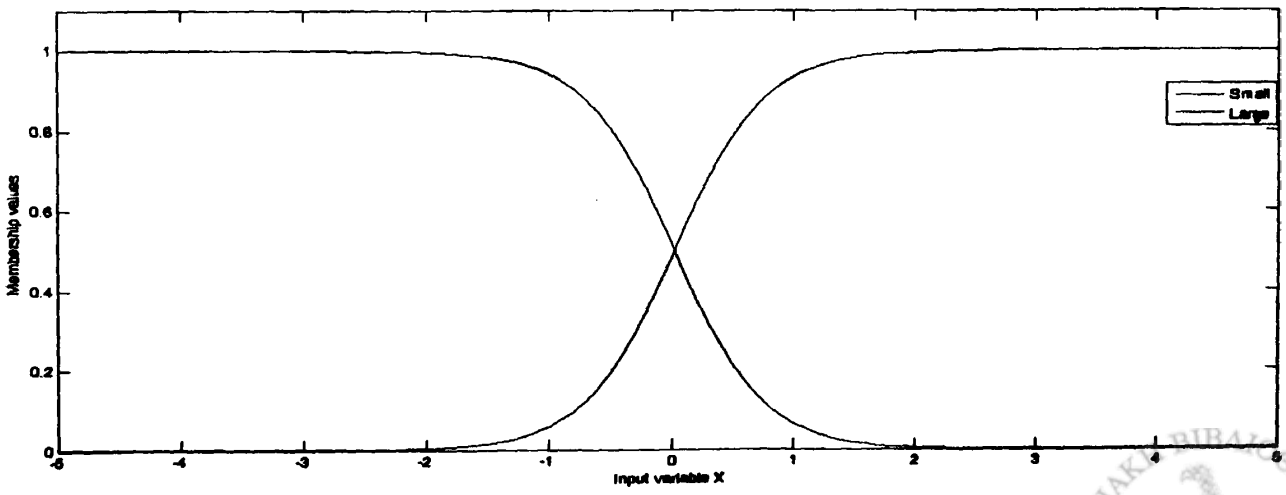
S – norm και *T – norm* συνάρτηση, αντίστοιχα. Για τον τελεστή επαγωγής χρησιμοποιείται, συνήθως, *T – norm* συνάρτηση αντί για κάποια από τις συναρτήσεις επαγωγής.

Επιπλέον, ένα σύστημα ασαφούς λογικής τύπου *Mamdani* περιλαμβάνει το στάδιο της αποασαφοποίησης. Στο στάδιο αυτό, υπολογίζεται μία διακριτή τιμή για κάθε ασαφές σύνολο που δίνεται ως έξοδος από την διαδικασία *GMP*. Αν και υπάρχουν αρκετές μέθοδοι αποασαφοποίησης, περισσότερο γνωστές είναι οι μέθοδοι *maximum* και *centroid*. Σύμφωνα με την μέθοδο αποασαφοποίησης *maximum*, η διακριτή τιμή είναι αυτή που αντιστοιχεί στην μέγιστη τιμή του ασαφούς συνόλου. Αν υπάρχουν περισσότερες από μία τέτοιες τιμές, τότε λαμβάνεται ο μέσος όρος τους. Στην μέθοδο αποασαφοποίησης *centroid*, η διακριτή τιμή είναι αυτή που προκύπτει από το κέντρο βάρους του ασαφούς συνόλου.

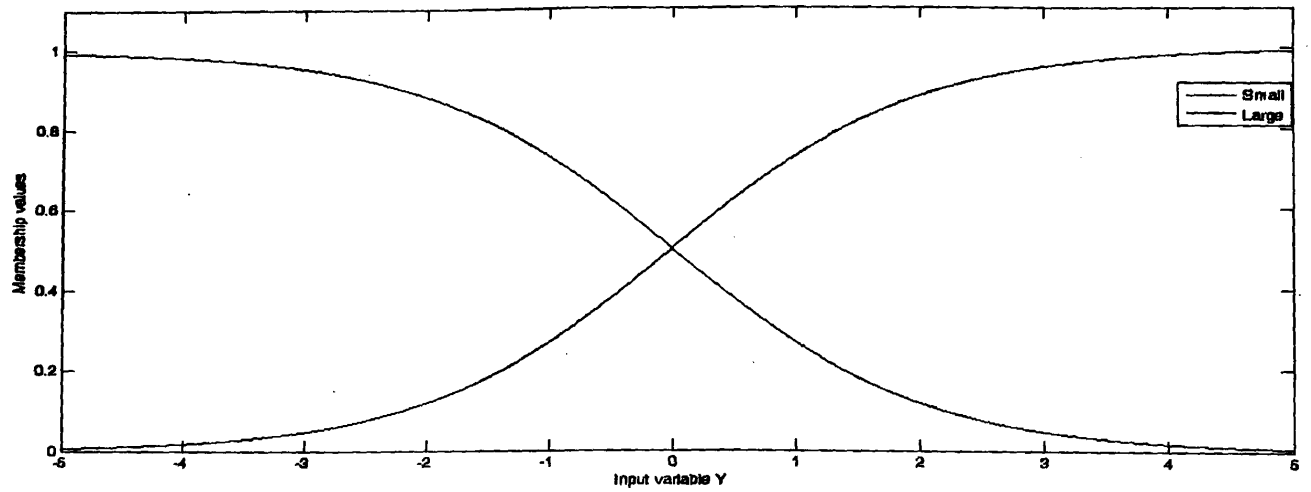
Για παράδειγμα, έστω το παρακάτω σύνολο ασαφών κανόνων:

if X is Small AND Y is Small then Z is NegativeLarge,
if X is Small AND Y is Large then Z is NegativeSmall,
if X is Large AND Y is Small then Z is PositiveSmall,
if X is Large AND Y is Large then Z is PositiveLarge.

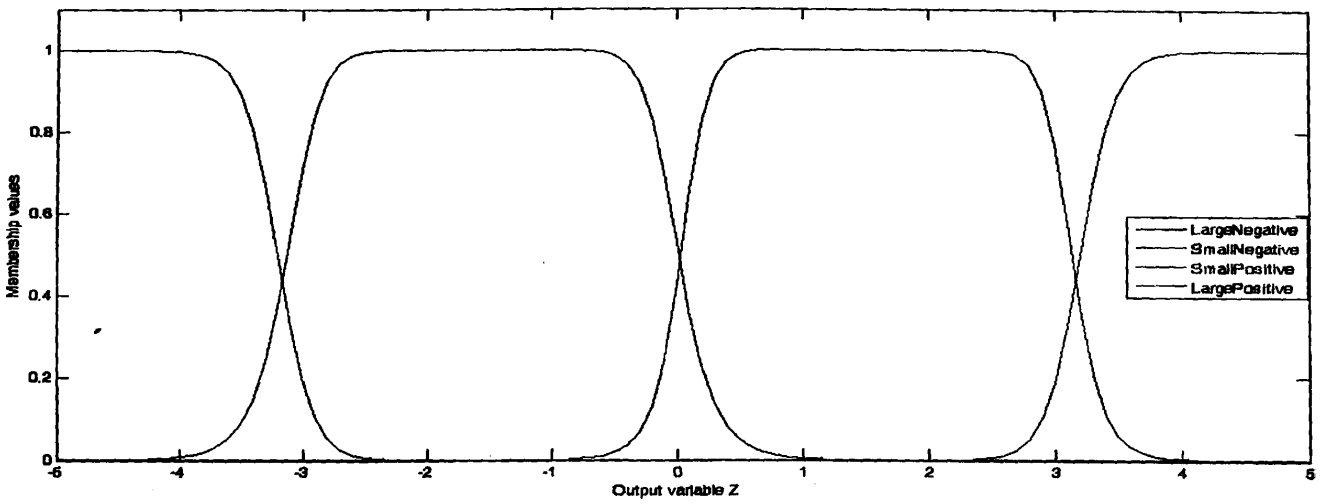
Στα Σχήματα 4.1 – 4.3 φαίνονται οι γραφικές παραστάσεις των συναρτήσεων συμμετοχής των ασαφών συνόλων που υπάρχουν στους ασαφείς κανόνες. Στο Σχήμα 4.4 φαίνεται η έξοδος του συστήματος. Έχει επιλεγεί ως τελεστής ένωσης ο *max*, ως τελεστής τομής ο *min*, ως τελεστής επαγωγής ο *min* και ως μέθοδος αποασαφοποίησης η *centroid*.



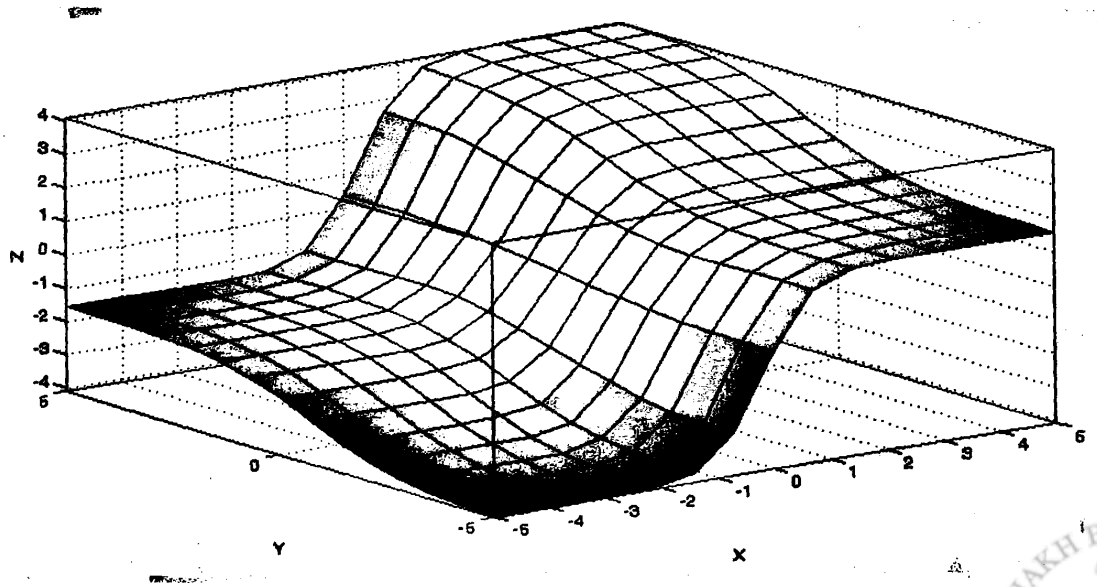
Σχήμα 4.1: Συναρτήσεις συμμετοχής της ασαφούς μεταβλητής X.



Σχήμα 4.2: Συναρτήσεις συμμετοχής της ασαφούς μεταβλητής Y.



Σχήμα 4.3: Συναρτήσεις συμμετοχής της ασαφούς μεταβλητής Z.



Σχήμα 4.4: Η έξοδος του συστήματος Mamdani.

Το Μοντέλο Ασαφούς Λογικής Sugeno

Το μοντέλο ασαφούς λογικής *Sugeno* προτάθηκε από τους *Takagi*, *Sugeno* και *Kang* [47]. Ένας τυπικός κανόνας σε ένα σύστημα ασαφούς λογικής τύπου *Sugeno* έχει την μορφή:

$$if\ X\ is\ A\ AND\ Y\ is\ B\ then\ Z = f(X,Y),$$

όπου τα A και B είναι ασαφή σύνολα και η $f(X,Y)$ είναι γραμμικός συνδυασμός των ασαφών μεταβλητών εισόδου. Γενικά, το τμήμα συνθηκών ενός κανόνα, σε ένα σύστημα ασαφούς λογικής τύπου *Sugeno*, ορίζεται όπως και στο μοντέλο *Mamdani*. Όμως, το τμήμα ενεργειών αποτελείται από μία ή περισσότερες ενέργειες που ορίζονται ως γραμμικός συνδυασμός των ασαφών μεταβλητών του τμήματος συνθηκών του κανόνα. Όταν η $f(X,Y)$ είναι σταθερή το μοντέλο Sugeno ονομάζεται μηδενικής τάξης.

Η διαδικασία ασαφούς συλλογιστικής στο μοντέλο ασαφούς λογικής *Sugeno* διαφέρει από την *GMP*. Συγκεκριμένα, σε ένα σύστημα ασαφούς λογικής τύπου *Sugeno* υπολογίζεται το βάρος κάθε κανόνα με το ίδιο τρόπο όπως και στην διαδικασία *GMP*. Στην συνέχεια, υπολογίζεται η τιμή των ενεργειών κάθε κανόνα. Δηλαδή, η τιμή της συνάρτησης που έχει οριστεί για την κάθε μεταβλητή εξόδου του κανόνα. Η τιμή αυτή πολλαπλασιάζεται με το βάρος του κανόνα. Η τελική τιμή κάθε μεταβλητής εξόδου υπολογίζεται από την Εξίσωση (4.26), όπου Z είναι μία μεταβλητή εξόδου, w_i είναι το βάρος του κανόνα i, Z_i είναι η έξοδος του κανόνα i για την μεταβλητή Z και N είναι ο αριθμός των κανόνων. Αντί για τον σταθμικό μέσο όρο, μπορεί να χρησιμοποιηθεί το σταθμικό άθροισμα (Εξίσωση (4.27)).

$$Z = \frac{\sum_{i=1}^{N} w_i Z_i}{\sum_{i=1}^{N} w_i} \qquad \text{Εξ. 4.26}$$

$$Z = \sum_{i=1}^{N} w_i Z_i \qquad \text{Εξ. 4.27}$$

Στην συνέχεια δίνεται ένα παράδειγμα μοντέλου ασαφούς λογικής *Sugeno* πρώτης τάξης. Το μοντέλο αποτελείται από τέσσερις κανόνες:

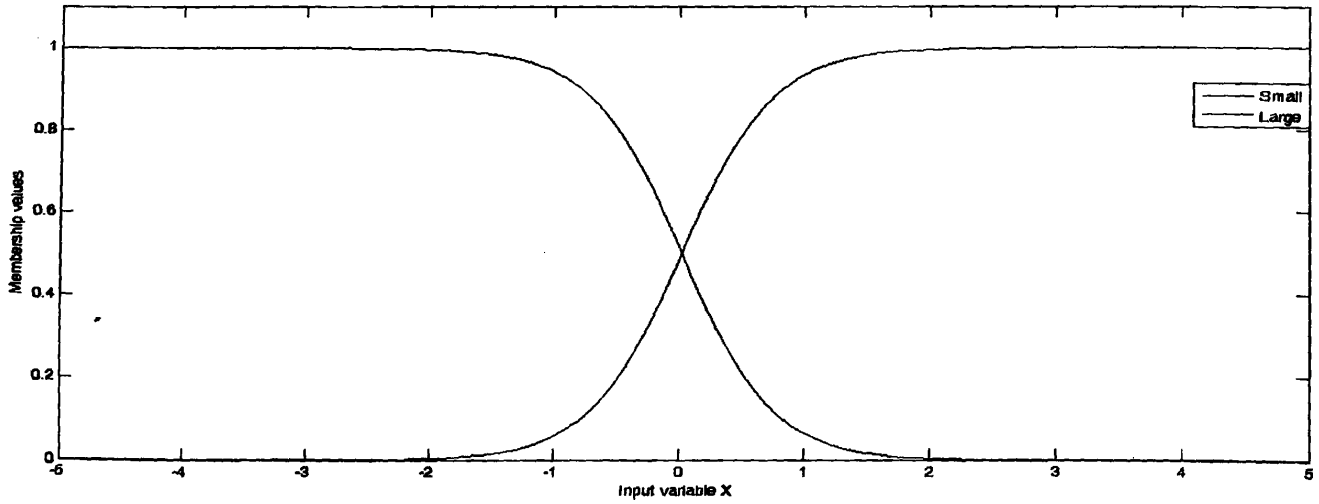
if X is Small AND Y is Small then Z=-X+Y+1,

if X is Small AND Y is Large then Z=-Y+3,

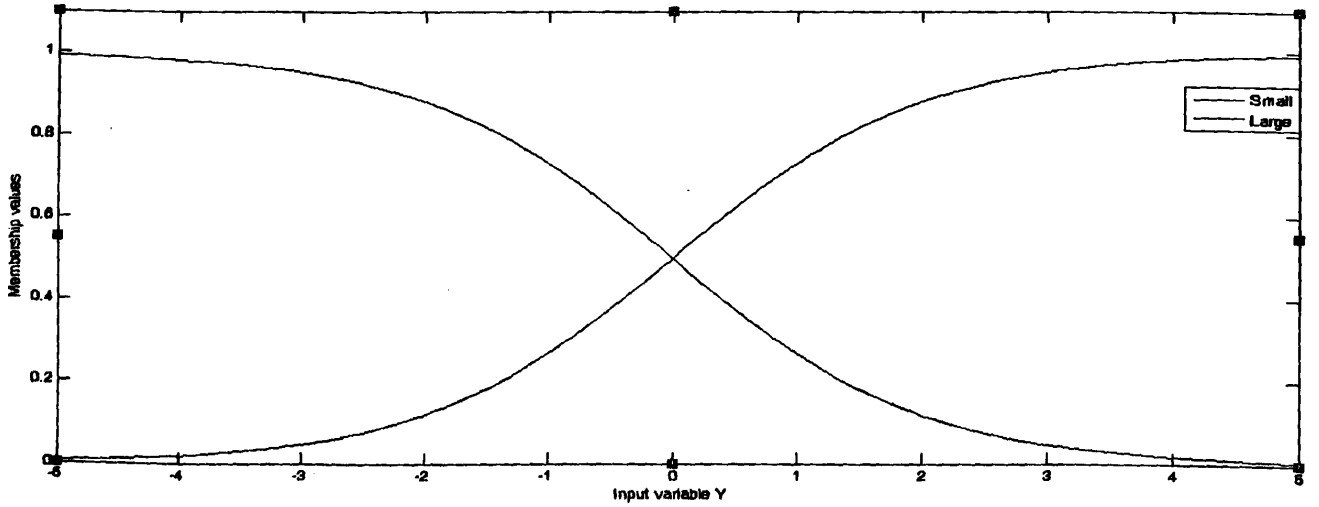
if X is Large AND Y is Small then Z=-X+3,

if X is Large AND Y is Large then Z=X+Y+2.

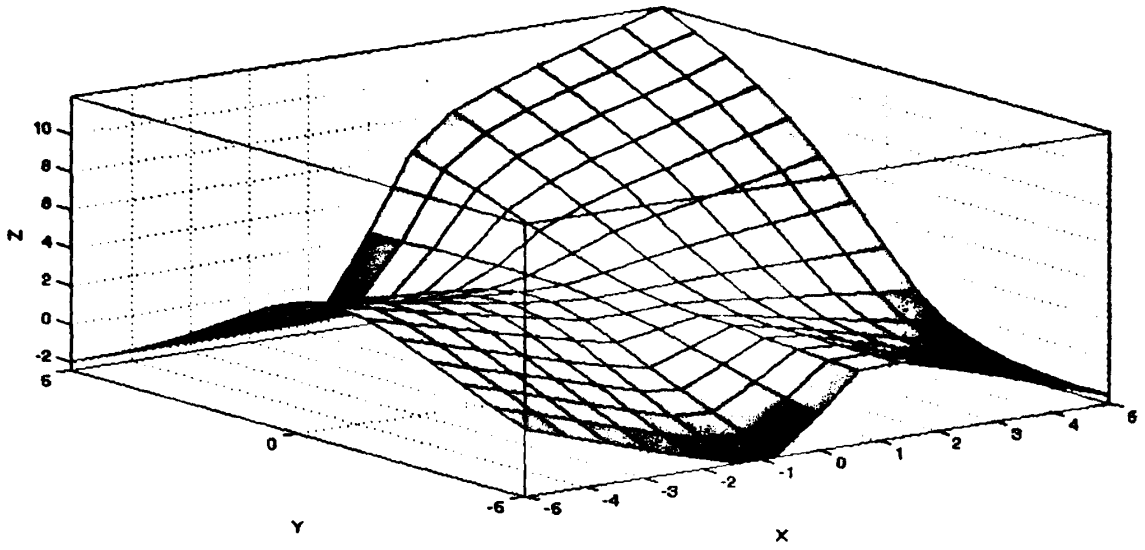
Στα Σχήματα 4.5 και 4.6 φαίνονται οι γραφικές παραστάσεις των συναρτήσεων συμμετοχής των ασαφών συνόλων που υπάρχουν στις συνθήκες των κανόνων. Στο Σχήμα 4.7 φαίνεται η έξοδος του συστήματος. Έχει επιλεγεί ως τελεστής ένωσης ο *max* και ως τελεστής τομής ο *min*. Φαίνεται ότι, η επιφάνεια αποτελείται από τέσσερα επίπεδα, καθένα από τα οποία προσδιορίζεται από την συνάρτηση εξόδου κάθε κανόνα.



Σχήμα 4.5: Συναρτήσεις συμμετοχής της ασαφούς μεταβλητής X.



Σχήμα 4.6: Συναρτήσεις συμμετοχής της ασαφούς μεταβλητής Y.



Σχήμα 4.7: Η έξοδος του συστήματος Sugeno.

Σε αντίθεση με το μοντέλο ασαφούς λογικής *Mamdani*, το μοντέλο *Sugeno* δεν ακολουθεί αυστηρά την ασαφή συλλογιστική διαδικασία *GMP*. Αυτό προκαλεί προβλήματα όταν η είσοδος είναι ασαφής και όχι διακριτή. Στην περίπτωση αυτή, μπορεί να εφαρμοστεί η ίδια τεχνική, όπως και στην *GMP*, για τον υπολογισμό των βαρών των κανόνων. Όμως, η έξοδος θα είναι διακριτή, είτε εφαρμοστεί η μέθοδος του σταθμικού μέσου όρου, είτε η μέθοδος του σταθμικού αθροίσματος. Δηλαδή, στο μοντέλο *Sugeno* η ασάφεια δεν μεταφέρεται από τις εισόδους στις εξόδους. Από την άλλη πλευρά, το μοντέλο *Sugeno* έχει μικρότερο υπολογιστικό κόστος σε σχέση με το *Mamdani*. Επίσης, είναι κατάλληλο όταν οι μεταβλητές εξόδου του συστήματος δεν είναι δυνατό να περιγραφούν ως ασαφή σύνολα ή όταν δεν υπάρχει εκ των προτέρων γνώση για αυτές. Στην δεύτερη περίπτωση, χρησιμοποιείται το μοντέλο *Sugeno* μηδενικής τάξης.

Το Μοντέλο Ασαφούς Λογικής Tsukamoto

Το μοντέλο ασαφούς λογικής *Tsukamoto* [50] έχει παρόμοια σύνταξη κανόνων με το μοντέλο *Mamdani*. Δηλαδή, τα τμήματα συνθηκών και ενεργειών ενός κανόνα αποτελούνται από ασαφείς προτάσεις. Όμως, υπάρχει ο περιορισμός ότι τα ασαφή σύνολα του τμήματος ενεργειών πρέπει να περιγράφονται από μονότονες συναρτήσεις συμμετοχής.

Κατά την εξαγωγή συμπερασμάτων σε ένα σύστημα ασαφούς λογικής τύπου Tsukamoto, αρχικά, υπολογίζεται το βάρος κάθε κανόνα με την τεχνική της διαδικασίας *GMP*. Στην συνέχεια, υπολογίζεται η τιμή των ασαφών μεταβλητών εξόδου των κανόνων. Η τιμή είναι

αυτή που αντιστοιχεί στο βάρος του κανόνα για το κάθε ασαφές σύνολο. Η τελική τιμή για κάθε μεταβλητή εξόδου προκύπτει από τον σταθμικό μέσο όρων των εξόδων όλων των κανόνων για αυτή. Το μοντέλο ασαφούς λογικής *Tsukamoto* δεν ακολουθεί την τυπική ασαφή συλλογιστική διαδικασία και έτσι μπορεί να δίνει διακριτές εξόδους. Το μοντέλο *Tsukamoto* δεν χρησιμοποιείται συχνά, επειδή δεν είναι τόσο διαφανές όσο τα άλλα δύο μοντέλα.

4.4 Context – Aware Συστήματα Ασαφούς Λογικής

Στην ενότητα αυτή περιγράφεται ένα σύνολο *context – aware* συστημάτων ενδιάμεσου λογισμικού που μοντελοποιούν το *context* και εξάγουν συμπεράσματα για την λειτουργία της εφαρμογής με χρήση ασαφούς λογικής.

Στην εργασία [5] προτείνεται ένα μοντέλο προσαρμογής των υπηρεσιών ενός *context – aware* συστήματος, το οποίο βασίζεται στην ασαφή λογική. Συγκεκριμένα, περιγράφεται ένας μηχανισμός για την επιλογή της πιο κατάλληλης πολιτικής για την εκτέλεση μίας υπηρεσίας, σύμφωνα με το τρέχον *context* του συστήματος.

Στο μοντέλο *FSAM*, κάθε παράμετρος του *context* αναπαρίσταται από μία λεκτική μεταβλητή, η οποία αντιστοιχίζεται σε ένα σύνολο λεκτικών τιμών. Κάθε λεκτική τιμή αναπαρίσταται από μία προκαθορισμένη συνάρτηση συμμετοχής. Επιπλέον, θεωρείται ότι κάθε υπηρεσία σε ένα *context – aware* σύστημα μπορεί να μεταφερθεί με χρήση διαφορετικών πολιτικών. Κάθε πολιτική σχετίζεται με μία συγκεκριμένη κατάσταση *context*. Κατά την διαδικασία εξαγωγής συμπερασμάτων, για κάθε πολιτική των αιτούμενων υπηρεσιών υπολογίζεται ένας δείκτης (*fitness degree*) που προσδιορίζει τον βαθμό καταλληλότητας της πολιτικής για την τρέχουσα κατάσταση *context*. Ο δείκτης *fitness degree* προσδιορίζεται μέσω μίας συνάρτησης που υπολογίζει την ασαφή απόσταση μεταξύ της πιο κατάλληλης κατάστασης *context* για μία πολιτική και της τρέχουσας κατάστασης *context* (*fitness function*). Ο δείκτης *fitness degree* μειώνεται καθώς αυξάνεται η ασαφής απόσταση. Έτσι, για μία αιτούμενη υπηρεσία, επιλέγεται η πολιτική με το μεγαλύτερο *fitness degree* ως η καλύτερη για την εκτέλεση της υπηρεσίας, με βάση τις τρέχουσες συνθήκες στο *context*.

Η πλήρης περιγραφή του μοντέλου *FSAM* δίνεται από τους παρακάτω ορισμούς:

- Ορισμός 1 (Υπηρεσία): Μία υπηρεσία αναπαριστά μία λειτουργία που παρέχεται από το *context – aware* σύστημα. Έστω $S = \{s_1, s_2, ..., s_q\}$ το σύνολο των υπηρεσιών στο σύστημα. Ως S_{need} ορίζεται το σύνολο των αιτούμενων υπηρεσιών από την εφαρμογή.
- Ορισμός 2 (Πολιτική): Μία πολιτική αναπαριστά μία μέθοδο για την εκτέλεση της υπηρεσίας, η οποία έχει συγκεκριμένες απαιτήσεις σε πόρους και παρέχει συγκεκριμένη ποιότητα υπηρεσιών (*QoS*). Έστω $P_i = \{p_i^1, p_i^2, ..., p_i^{mi} \mid i \in [1, q]\}$ το σύνολο των υπηρεσιών για την μεταφορά της υπηρεσίας $s_i \in S$.
- Ορισμός 3 (*Context*): Έστω $C = \{c_1, c_2, ..., c_n\}$ το σύνολο των παραμέτρων του *context* που παρακολουθούνται από το *context – aware* σύστημα.
- Ορισμός 4 (Κατάσταση *Context*): Μία κατάσταση *context* είναι ένας συνδυασμός πληροφορίας του *context*. Έστω $LV = \{lv_1, lv_2, ..., lv_k\}$ το σύνολο των λεκτικών τιμών. Η κατάσταση του *context* μία χρονική στιγμή t δηλώνεται ως $SI(t)$ και αναπαρίσταται από το σύνολο που δίνεται στην Εξίσωση (4.28), όπου $u_{lv_b}(value\_of(c_a, t))$ είναι ο βαθμός συμμετοχής της τιμής της παραμέτρου c_a την χρονική στιγμή t στο ασαφές σύνολο lv_b.

$$SI(t) = \left\{ \begin{matrix} (c_a, lv_b, u_{lv_b}(value\_of(c_a, t))), \\ c_a \in C, a \in [1, n], lv_b \in LV, b \in [1, k] \end{matrix} \right\} \qquad \text{Εξ. 4.28}$$

- Ορισμός 5 ($SR(p_i^j)$): Δεδομένης μίας υπηρεσίας s_i ($s_i \in S_{need}$), κάθε πολιτική p_i^j ($p_i^j \in P_i$) συνδέεται με μία κατάσταση *context* που είναι η πιο κατάλληλη για αυτή ($SR(p_i^j)$). Το σύνολο $SR(p_i^j)$ ορίζεται από την Εξίσωση (4.29).

$$SI(t) = \left\{ \begin{matrix} (c_a, lv_b, u_{lv_b}(best\_value\_of(c_a))), \\ c_a \in C, a \in [1, n], lv_b \in LV, b \in [1, k] \end{matrix} \right\} \qquad \text{Εξ. 4.29}$$

Η Εξίσωση (4.29) είναι ίδια με την Εξίσωση (4.28) με την διαφορά ότι η συνάρτηση $value\_of(c_a, t)$ αντικαθίσταται από την $best\_value\_of(c_a)$, η οποία δηλώνει την

πιο κατάλληλη τιμή της παραμέτρου c_a του *context*, όταν χρησιμοποιείται η πολιτική p_i^j της υπηρεσίας s_i.

- Ορισμός 6 (*Fuzzy – Based Service Adaptation Process (FSAP)*): Η διαδικασία *FSAP* είναι μία αντιστοίχιση από την τρέχουσα κατάσταση *context* $SI(t)$ σε ένα σύνολο κατάλληλων πολιτικών $P_{suitable}$. Κάθε στοιχείο στο σύνολο αυτό είναι η πιο κατάλληλη πολιτική για μία συγκεκριμένη υπηρεσία s_i $(s_i \in S_{need})$.
- Ορισμός 7 (*Fitness Function*): Έστω $FD(p_i^j)$ ο δείκτης *fitness degree* για την πολιτική p_i^j για την τρέχουσα κατάσταση *context*. Δεδομένης μίας υπηρεσίας s_i $(s_i \in S_{need})$, η συνάρτηση *Fitness Function* $(FF): SI(current) \times SR(p_i^j) \rightarrow FD(p_i^j)$ είναι μία αντιστοίχιση από το σύνολο $SI(t)$ και $SR(p_i^j)$ στον δείκτη $FD(p_i^j)$. Η συνάρτηση *Fitness Function* δίνεται από την Εξίσωση (4.30).

$$FF\left(SI(current), SR\left(p_i^j\right)\right) = \frac{1}{\sum_{i=1}^{size\_of\left(SR\left(p_i^j\right)\right)} \left| u\left(best\_value\_of\left(c_i\right)\right) - u\left(value\_of\left(c_i, current\right)\right) \right|^{l_i}} \qquad \text{Εξ. 4.30}$$

όπου $u(x)$ είναι η συνάρτηση συμμετοχής για το i -οστό στοιχείο των συνόλων $SI(t)$ και $SR(p_i^j)$. Η τιμή l_i προσαρμόζει την επίδραση μίας παραμέτρου του *context* στην επιλογή μίας πολιτικής για μία υπηρεσία.

Έτσι, κάθε φορά που μεταβάλλονται οι τιμές των παραμέτρων του *context*, για κάθε πολιτική κάθε υπηρεσίας που ανήκει στο σύνολο S_{need} υπολογίζεται ο δείκτης *fitness degree*. Τελικά, επιλέγεται, για κάθε αιτούμενη υπηρεσία, η πολιτική που έχει το μεγαλύτερο *fitness degree*. Η πολιτική αυτή, συνεπώς, ικανοποιεί περισσότερο τις απαιτήσεις της τρέχουσας κατάστασης *context* και παρέχει την καλύτερη δυνατή ποιότητα υπηρεσιών στους χρήστες.

Στην εργασία [32] προτείνεται ένα *context – aware* σύστημα ενδιάμεσου λογισμικού, του οποίου η αρχιτεκτονική βασίζεται σε πράκτορες. Το σύστημα αυτό παρέχει ένα σύνολο βασικών υπηρεσιών για την απόκτηση, επεξεργασία και κατανομή του *context*. Ο μηχανισμός

εξαγωγής συμπερασμάτων για την κατάσταση του συστήματος, με βάση το τρέχον *context*, στηρίζεται στην ασαφή λογική. Επιπλέον, χρησιμοποιείται η τεχνική των δέντρων απόφασης [48] για την παραγωγή των κανόνων προσαρμογής των υπηρεσιών του συστήματος. Το κύριο χαρακτηριστικό των δέντρων απόφασης είναι ότι μετατρέπονται εύκολα σε κανόνες, αλλά είναι ευαίσθητα στον θόρυβο. Για τον λόγο αυτό, στην εργασία αυτή γίνεται χρήση μίας τεχνικής που στηρίζεται σε ασαφή δέντρα απόφασης.

Αρχικά, κάθε παράμετρος του *context* αντιστοιχίζεται σε ένα σύνολο λεκτικών τιμών. Για κάθε λεκτική τιμή, καθορίζεται μία συνάρτηση συμμετοχής που αναπαριστά καλύτερα τον βαθμό συμμετοχής των τιμών του *context*. Ως συνάρτηση συμμετοχής χρησιμοποιείται η τραπεζοειδής, αφού θεωρείται γενική συνάρτηση συμμετοχής από την οποία μπορεί να παραχθούν άλλες, όπως η κανονική συνάρτηση.

Μετά τον καθορισμό των συναρτήσεων συμμετοχής κατασκευάζεται το ασαφές δέντρο απόφασης. Χρησιμοποιείται ένα σύνολο εκπαίδευσης που προκύπτει είτε από ειδικούς στο πεδίο της εφαρμογής, είτε από αισθητήρες και ενέργειες προηγούμενων χρηστών. Αρχικά, γίνεται ασαφοποίηση του *context* στο σύνολο εκπαίδευσης και στην συνέχεια χρησιμοποιείται ο αλγόριθμος *ID3* [48], ο οποίος εφαρμόζεται στις ασαφείς τιμές και χρησιμοποιεί το μέτρο της εντροπίας και το *Information Gain* [48] για την εκτέλεση της διαδικασίας δημιουργίας του δέντρου.

Οι ασαφείς κανόνες που παράγονται από το ασαφές δέντρο απόφασης χρησιμοποιούνται στην διαδικασία λήψης αποφάσεων για την λειτουργία του συστήματος. Στην εργασία [32], χρησιμοποιείται το μοντέλο ασαφούς λογικής *Mamdani*. Ως τελεστής ένωσης και τομής χρησιμοποιείται ο *max* και ο *min* αντίστοιχα, ως τελεστής επαγωγής ο *min* και για την διαδικασία αποασαφοποίησης χρησιμοποιείται η μέθοδος *centroid*.

Στην εργασία [25] παρουσιάζεται μία διαδικασία κατασκευής ενός ασαφούς δέντρου απόφασης για την διαχείριση της ασάφειας στο *context*. Το *context* διακρίνεται σε *context* χαμηλού επιπέδου και σε *context* υψηλού επιπέδου. Η πρώτη κατηγορία περιλαμβάνει, ουσιαστικά, το *context* που προέρχεται από τους αισθητήρες, την εφαρμογή, καθώς και από τον χρήστη. Ενώ, η δεύτερη κατηγορία περιγράφει το *context* που παράγεται από την διαδικασία εξαγωγής συμπερασμάτων και αφορά τις αποφάσεις για την λειτουργία του

συστήματος. Για κάθε παράμετρο του *context* της πρώτης κατηγορίας ορίζεται ένα σύνολο λεκτικών τιμών για τις οποίες προσδιορίζονται οι αντίστοιχες συναρτήσεις συμμετοχής. Οι συναρτήσεις συμμετοχής για το *context* της δεύτερης κατηγορίας υπολογίζονται ως το γινόμενο των συναρτήσεων συμμετοχής των παραμέτρων του *context* από τις οποίες εξαρτάται. Στην συνέχεια, για την κατασκευή του δέντρου απόφασης χρησιμοποιείται το μέτρο της εντροπίας και το *Information Gain* για κάθε παράμετρο του *context*, με την διαφορά ότι σε αυτά τα μεγέθη χρησιμοποιούνται οι τιμές των συναρτήσεων συμμετοχής για το *context* του συνόλου εκπαίδευσης.

ΚΕΦΑΛΑΙΟ 5. ΠΕΡΙΓΡΑΦΗ ΥΠΗΡΕΣΙΑΣ ΕΝΔΙΑΜΕΣΟΥ ΛΟΓΙΣΜΙΚΟΥ

Σε αυτό το κεφάλαιο περιγράφεται η υπηρεσία ενδιάμεσου λογισμικού που αναπτύχθηκε. Αναλύονται η αρχιτεκτονική της υπηρεσίας και το μοντέλο που χρησιμοποιείται για την αναπαράσταση του *context*. Επίσης, περιγράφεται η χρήση του μοντέλου στην διαδικασία εξαγωγής συμπερασμάτων για την προσαρμογή μίας *context – aware* εφαρμογής. Στο τέλος, επισημαίνονται τα χαρακτηριστικά της υπηρεσίας ενδιάμεσου λογισμικού που αφορούν την δυναμική διαμόρφωση της λειτουργίας της.

5.1 Γενική Περιγραφή της Υπηρεσίας Ενδιάμεσου Λογισμικού

Στην παρούσα εργασία, τροποποιήθηκε η υπηρεσία ενδιάμεσου λογισμικού που παρουσιάστηκε στο Κεφάλαιο 3, ώστε να διαχειρίζεται την αβεβαιότητα και την ασάφεια στο *context*. Η αρχιτεκτονική της υπηρεσίας διατηρείται, αλλά η μοντελοποίηση του *context* βασίζεται, πλέον, στην ασαφή λογική. Συγκεκριμένα, η υπηρεσία ενδιάμεσου λογισμικού αναπαριστά και διαχειρίζεται το *context* με χρήση ενός από τα μοντέλα ασαφούς λογικής που περιγράφηκαν στο Κεφάλαιο 4. Ο προσδιορισμός του μοντέλου που θα χρησιμοποιηθεί

γίνεται από τον προγραμματιστή της *context – aware* εφαρμογής, σε μία μέθοδο που ορίζεται από την προγραμματιστική διεπαφή διασύνδεσης της εφαρμογής με την υπηρεσία.

Όμως, η λειτουργικότητα των συγκεκριμένων μοντέλων περιορίζεται στην επεξεργασία παραμέτρων του *context* που μπορούν να αναπαρασταθούν ως ασαφή σύνολα και που γενικά παίρνουν τιμές σε ένα συνεχές διάστημα. Όμως, σε ένα *context – aware* σύστημα υπάρχουν παράμετροι του *context* που παίρνουν λίγες διακριτές τιμές. Για τις παραμέτρους αυτές, δεν έχει νόημα ο ορισμός ασαφών συνόλων. Έτσι, τα μοντέλα ασαφούς λογικής Mamdani, Sugeno και Tsukamoto επεκτάθηκαν, ώστε να μπορούν να διαχειριστούν και τέτοιου είδους πληροφορία.

5.2 Αρχιτεκτονική της Υπηρεσίας Ενδιάμεσου Λογισμικού

Η υπηρεσία ενδιάμεσου λογισμικού παρέχεται με την μορφή ενός υποσυστήματος λογισμικού που λειτουργεί παράλληλα με την εφαρμογή. Περιλαμβάνει ένα σύνολο δομικών μονάδων που υλοποιούν τις διαδικασίες διαχείρισης του *context*. Η αρχιτεκτονική της υπηρεσίας έχει ήδη δοθεί στο Κεφάλαιο 3. Στην συνέχεια, περιγράφονται συνοπτικά τα συστατικά της υπηρεσίας ενδιάμεσου λογισμικού:

- *ContextManager*: Είναι η κεντρική μονάδα της υπηρεσίας ενδιάμεσου λογισμικού. Μεσολαβεί μεταξύ της εφαρμογής και των υπόλοιπων υποσυστημάτων και αρχικοποιεί την υπηρεσία.
- *RulesParser*: Αναλύει συντακτικά τις παραμέτρους του *context* και τους κανόνες και αποθηκεύει τα περιεχόμενά τους. «Κατασκευάζει» την γνώση του συστήματος και την κάνει διαθέσιμη στα υπόλοιπα υποσυστήματα.
- *EventHandler*: Συλλέγει τις τιμές του *context* από τους μηχανισμούς παρακολούθησης των προμηθευτών και κατασκευάζει γεγονότα. Μεσολαβεί μεταξύ των προμηθευτών και της μονάδας επεξεργασίας των γεγονότων.
- *RulesEvaluator*: Επεξεργάζεται τα γεγονότα και αποφασίζει την εκτέλεση ενεργειών. Προσαρμόζει την συμπεριφορά της εφαρμογής με βάση την γνώση του συστήματος.

Εκτός από τα υποσυστήματα που συμμετέχουν στην διαχείριση του *context*, η υπηρεσία ενδιάμεσου λογισμικού περιλαμβάνει και δύο συστατικά που αφορούν την δυναμική διαμόρφωσή της. Ο *ContextRecordManager* και ο *RulesRecordManager* είναι τα

υποσυστήματα που παρέχουν στην εφαρμογή μεθόδους και γραφικές διεπαφές χρήστη (*GUIs*), για την τροποποίηση του *context* και των κανόνων κατά την λειτουργία του συστήματος, αντίστοιχα.

Η υπηρεσία ενδιάμεσου λογισμικού παρέχει επιπλέον μία προγραμματιστική διεπαφή για την διασύνδεσή της με την εφαρμογή. Η διεπαφή υλοποιείται από τον προγραμματιστή της εφαρμογής και εγκαθίστανται οι σύνδεσμοι για την επικοινωνία υπηρεσίας – εφαρμογής. Η διεπαφή αυτή περιγράφεται με λεπτομέρεια στο Κεφάλαιο 6.

5.3 Μοντελοποίηση του Context

Το μοντέλο αναπαράστασης και διαχείρισης του *context* που χρησιμοποιούμε βασίζεται στην ασαφή λογική. Το *context* περιγράφεται μέσω ασαφών συνόλων, ασαφών κανόνων και γεγονότων. Ένας ασαφής κανόνας αποτελείται από εκφράσεις λογικά συνδεδεμένες μεταξύ τους.

Γεγονότα

Τα γεγονότα αναπαριστούν τις αλλαγές στο *context*. Ένα γεγονός αντικατοπτρίζει την νέα τιμή σε μία παράμετρο του *context*. Τα γεγονότα μεταφέρουν την νέα γνώση στο σύστημα από τους προμηθευτές του *context*.

Εκφράσεις

Στην γενική περίπτωση, μία έκφραση αναπαριστά μία ασαφή πρόταση. Το είδος των εκφράσεων εξαρτάται από το μοντέλο ασαφούς λογικής που χρησιμοποιείται. Οι εκφράσεις αποτελούν το βασικό δομικό στοιχείο του τμήματος συνθηκών και ενεργειών των κανόνων.

Κανόνες

Περιγράφουν τον τρόπο με τον οποίο το σύστημα θα πρέπει να αντιδράσει στις αλλαγές του *context*. Αποτελούνται από την σύνδεση συνθηκών και ενεργειών.

Οι κανόνες αποτελούν την γνώση που έχει η υπηρεσία ενδιάμεσου λογισμικού, προκειμένου να γνωρίζει πώς θα προσαρμόσει αυτόματα την εφαρμογή στις αλλαγές του *context*. Οι αλλαγές του *context* αναπαρίστανται με γεγονότα, τα οποία συλλέγονται από την υπηρεσία

ενδιάμεσου λογισμικού. Στην συνέχεια, περιγράφεται η αναπαράσταση του *context* για καθένα από τα τρία μοντέλα ασαφούς λογικής.

5.3.1 Αναπαράσταση του Context στο Μοντέλο Mamdani

Στο μοντέλο ασαφούς λογικής *Mamdani* για κάθε παράμετρο του *context* προσδιορίζεται ένας αριθμός λεκτικών τιμών. Κάθε λεκτική τιμή περιγράφεται από μία συνάρτηση συμμετοχής, η οποία αναπαριστά καλύτερα τον βαθμό συμμετοχής των τιμών της παραμέτρου του *context* στο συγκεκριμένο ασαφές σύνολο. Στον Πίνακα 5.1 φαίνεται το συντακτικό για την περιγραφή των παραμέτρων του *context* με χρήση ασαφών συνόλων.

Πίνακας 5.1: Συντακτικό του context στο μοντέλο Mamdani.

<context_description>	::=	context_parameter: <domain>, <linguistic_values>
<domain>	::=	[value, value]
<linguistic_values>	::=	linguistic_value (<membership_function>, <parameters>), <linguistic_values> | linguistic_value (<membership_function>, <parameters>)
<membership_function>	::=	trimf| trapmf | gaussmf | gbellmf | sigmf
<parameters>	::=	parameter = value, <parameters> | parameter = value

$$CNF: R = \wedge_{i=1}^{n} C_i, \quad C_i = \left(\vee_{j=1}^{m} c_j\right)$$
$$DNF: R = \vee_{i=1}^{n} D_i, \quad D_i = \left(\wedge_{j=1}^{m} c_j\right)$$
$$c_j =< simple\_condition >$$

Σχήμα 5.1: Συντακτικό του τμήματος συνθηκών των κανόνων.

Ένας ασαφής κανόνας συσχετίζει δύο ή περισσότερες ασαφείς προτάσεις. Μία ασαφής πρόταση θέτει μία λεκτική τιμή σε μία παράμετρο του *context*. Το τμήμα των συνθηκών ενός κανόνα, μπορεί να ακολουθεί Κανονική Συζευκτική Μορφή (*CNF*) ή Κανονική Διαζευκτική Μορφή (*DNF*). Το συντακτικό του τμήματος συνθηκών φαίνεται στο Σχήμα 5.1. Το τμήμα των ενεργειών ενός κανόνα είναι λιγότερο σύνθετο, αφού αποτελείται μόνο από συζεύξεις ενεργειών. Η κάθε ενέργεια περιγράφεται μέσω μίας απλής ασαφούς πρότασης.

Το τμήμα συνθηκών ενός κανόνα είναι σε *CNF* μορφή, όταν αποτελείται από την σύζευξη ενός ή περισσότερων σύνθετων συνθηκών (C_i), όπου κάθε σύνθετη συνθήκη αποτελείται από την διάζευξη ενός ή περισσότερων απλών συνθηκών (c_j). Τα C_i ονομάζονται *clauses* του κανόνα σε Κανονική Συζευκτική Μορφή. Το τμήμα συνθηκών ενός κανόνα είναι σε *DNF* μορφή, όταν αποτελείται από την διάζευξη ενός ή περισσότερων σύνθετων συνθηκών (D_i), όπου κάθε σύνθετη συνθήκη αποτελείται από την σύζευξη ενός ή περισσότερων απλών συνθηκών (c_j). Τα D_i ονομάζονται *implicants* του κανόνα σε Κανονική Διαζευκτική Μορφή. Τα c_j ονομάζονται *literals* του κανόνα σε *CNF* ή *DNF* μορφή και είναι ασαφείς προτάσεις.

Σύμφωνα με θεμελιώδες θεώρημα της λογικής, οποιαδήποτε λογική έκφραση μπορεί να γραφεί ισοδύναμα σε μία σε Κανονική Συζευκτική Μορφή και σε μία σε Κανονική Διαζευκτική Μορφή [19]. Το θεώρημα αυτό μας εξασφαλίζει ότι το συντακτικό που χρησιμοποιούμε μπορεί να περιγράψει οποιαδήποτε λογική έκφραση.

Πίνακας 5.2: Συντακτικό των κανόνων στο μοντέλο Mamdani.

<rule>	::=	<conditional_part> ⇒ <action_part>
<conditional _part>	::=	<c_statement> | <d_statement>
<c_statement>	::=	(<c_condition>) && <c_statement> | (<c_condition>)
<d_statement>	::=	(<d_condition>) || <d_statement> | (<d_condition>)
<c_condition>	::=	<condition> || <c_condition> | <condition>
<d_condition>	::=	<condition> && <d_condition> | <condition>
<condition>	::=	(context_parameter = linguistic_value)
<action_part>	::=	<a_statement>
<a_statement>	::=	<action> && <a_statement> | <action>
<action>	::=	(context_parameter = linguistic_value)

$$R_1 = (X = HIGH \;\&\&\; Y = LOW) \,||\, (Z = HIGH) \Rightarrow E = HIGH$$
$$R_2 = (Z = LOW \,||\, Y = LOW) \;\&\&\; (X = MEDIUM) \Rightarrow E = MEDIUM$$

Σχήμα 5.2: Παραδείγματα κανόνων στο μοντέλο Mamdani.

Στον Πίνακα 5.2 φαίνεται το συντακτικό των κανόνων. Η σύνθεση των κανόνων γίνεται σύμφωνα με το συγκεκριμένο συντακτικό. Η υπηρεσία ενδιάμεσου λογισμικού δεν μπορεί να ερμηνεύσει κανόνες αυθαίρετης μορφής καθώς κάτι τέτοιο θα έκανε πολύ δύσκολη και χρονοβόρα την συντακτική ανάλυση και την διαδικασία αποτίμησης. Στο Σχήμα 5.2 φαίνονται δύο γενικά παραδείγματα κανόνων που ακολουθούν το συντακτικό. Περισσότερα παραδείγματα κανόνων περιγράφονται στο Κεφάλαιο 6, όπου αναλύεται η κινητή *context – aware* εφαρμογή που υλοποιήσαμε πάνω από την υπηρεσία ενδιάμεσου λογισμικού και οι συγκεκριμένοι κανόνες που χρησιμοποιήθηκαν.

5.3.2 Αναπαράσταση του Context στο Μοντέλο Sugeno

Το μοντέλο ασαφούς λογικής *Sugeno* επιβάλει τον διαχωρισμό του *context* σε αυτό που οι τιμές του λαμβάνονται από τους προμηθευτές και σε αυτό που οι τιμές του προσδιορίζουν την λειτουργία της εφαρμογής. Το *context* της πρώτης κατηγορίας ονομάζεται *context* εισόδου, ενώ της δεύτερης κατηγορίας ονομάζεται *context* εξόδου. Το μοντέλο ασαφούς λογικής *Sugeno* περιορίζει την ασάφεια στο *context* εισόδου. Το *context* εξόδου προσδιορίζεται ως γραμμικός συνδυασμός των παραμέτρων του *context* εισόδου. Έτσι, για κάθε παράμετρο του *context* εισόδου καθορίζεται ένας αριθμός λεκτικών τιμών. Το ασαφές σύνολο που αντιστοιχεί σε κάθε λεκτική τιμή περιγράφεται από μία συνάρτηση συμμετοχής. Αντίστοιχα, για κάθε παράμετρο του *context* εξόδου προσδιορίζεται ένα σύνολο συναρτήσεων που περιγράφουν τον τρόπο μεταβολής της αντίστοιχης παραμέτρου με βάση τις τιμές του *context* εισόδου από το οποίο εξαρτάται. Η περιγραφή του *context* εισόδου γίνεται με βάση το συντακτικό του Πίνακα 5.1. Στον Πίνακα 5.3 φαίνεται το συντακτικό για την περιγραφή του *context* εξόδου.

Πίνακας 5.3: Συντακτικό του context εξόδου στο μοντέλο Sugeno.

<context_description>	::=	context_parameter (<dependencies>): <domain>, <functions>
<dependencies>	::=	context_parameter, < dependencies > | context_parameter
<domain>	::=	[value, value]
< functions>	::=	function_name (<parameters>), <functions> | function_name (<parameters>)
<parameters>	::=	parameter = value, <parameters> | parameter = value

Το συντακτικό των κανόνων διαφέρει από το συντακτικό του μοντέλου *Mamdani*. Συγκεκριμένα, το τμήμα συνθηκών ενός κανόνα, σε ένα σύστημα ασαφούς λογικής τύπου *Sugeno*, ορίζεται όπως και στο μοντέλο *Mamdani*. Όμως, το τμήμα ενεργειών αποτελείται από την σύζευξη εκφράσεων που αναθέτουν τιμές στις παραμέτρους του context εξόδου με χρήση των συναρτήσεων. Οι συναρτήσεις είναι είτε γραμμικός συνδυασμός του context εισόδου, είτε σταθερές. Στον Πίνακα 5.4 φαίνεται το συντακτικό των κανόνων στο μοντέλο ασαφούς λογικής *Sugeno*. Στο Σχήμα 5.3 φαίνονται δύο γενικά παραδείγματα κανόνων που ακολουθούν το συγκεκριμένο συντακτικό.

Πίνακας 5.4: Συντακτικό των κανόνων στο μοντέλο Sugeno.

<rule>	::=	<conditional_part> ⇒ <action_part>
<conditional _part>	::=	<c_statement> | <d_statement>
<c_statement>	::=	(<c_condition>) && <c_statement> | (<c_condition>)
<d_statement>	::=	(<d_condition>) || <d_statement> | (<d_condition>)
<c_condition>	::=	<condition> || <c_condition> | <condition>
<d_condition>	::=	<condition> && <d_condition> | <condition>
<condition>	::=	(context_parameter = linguistic_value)
<action_part>	::=	<a_statement>
<a_statement>	::=	<action> && <a_statement> | <action>
<action>	::=	(context_parameter = function_name)

$$R_1 = (X = HIGH \;\&\&\; Y = LOW) \,||\, (Z = HIGH) \Rightarrow E = E_1$$
$$R_1 = (Z = LOW \,||\, Y = LOW) \;\&\&\; (X = MEDIUM) \Rightarrow E = E_2$$
$$E_1 = 3X + 2Y + Z - 1$$
$$E_2 = X + Y + Z$$

Σχήμα 5.3: Παραδείγματα κανόνων στο μοντέλο Sugeno.

5.3.3 Αναπαράσταση του Context στο Μοντέλο Tsukamoto

Το μοντέλο ασαφούς λογικής *Tsukamoto* αναπαριστά το *context* με τον ίδιο τρόπο, όπως και το μοντέλο *Mamdani*. Δηλαδή, για κάθε παράμετρο του *context* ορίζεται ένας αριθμός ασαφών συνόλων και το κάθε ασαφές σύνολο περιγράφεται μέσω μίας συνάρτησης συμμετοχής. Επιπλέον, τα τμήματα συνθηκών και ενεργειών ενός κανόνα αποτελούνται από ασαφείς προτάσεις και ακολουθούν την ίδια σύνταξη με το μοντέλο *Mamdani*. Όμως, τα

ασαφή σύνολα για τις παραμέτρους του *context* στο τμήμα ενεργειών των κανόνων, πρέπει να προσδιορίζονται μέσω μονότονων συναρτήσεων συμμετοχής.

5.3.4 Επέκταση του Μοντέλου του Context

Η αναπαράσταση του *context*, όπως ορίστηκε στα τρία μοντέλα ασαφούς λογικής, αφορά μόνο παραμέτρους του *context* που ορίζονται σε ένα συνεχές διάστημα (π.χ. ένα υποσύνολο του R). Όμως, σε ένα *context – aware* σύστημα υπάρχουν παράμετροι του *context* που λαμβάνουν τιμές από κάποιο μικρό διακριτό διατεταγμένο ή μη – διατεταγμένο σύνολο. Οι τιμές των παραμέτρων αυτών δεν έχει νόημα να περιγραφούν με ασαφή σύνολα. Στην παρούσα εργασία επεκτείναμε το μοντέλο του *context*, ώστε να μπορεί να διαχειριστεί και αυτού του είδους την πληροφορία. Έτσι, οι εκφράσεις στο τμήμα συνθηκών και ενεργειών των κανόνων είναι δυνατό να αποτελούνται από μία παράμετρο του *context* συγκριτικά συσχετισμένη με μία τιμή. Ο τύπος της τιμής εξαρτάται από την παράμετρο και μπορεί να είναι ακέραιος, δεκαδικός ή συμβολοσειρά. Ο Πίνακας 5.5 δίνει το συντακτικό για τις μη ασαφείς παραμέτρους του *context*. Στον Πίνακα 5.6 φαίνεται το συντακτικό των όρων *< condition >* και *< action >* για το συγκεκριμένο είδος *context*.

Πίνακας 5.5: Συντακτικό των μη – ασαφών παραμέτρων του context.

<context_description>	::=	context_parameter: <domain>
<domain>	::=	[value, value]

Πίνακας 5.6: Επέκταση του μοντέλου του context.

<condition>	::=	(context_parameter <comparison_operator> value)
<action>	::=	(context_parameter = value) | context_parameter
<comparison_operator>	::=	> | < | >= | <= | =

5.4 Συντακτική Ανάλυση Κανόνων

Ο *RulesParser* είναι το υποσύστημα που εκτελεί συντακτική ανάλυση στο *context* και στους κανόνες και εξάγει την πληροφορία που χρησιμοποιεί η υπηρεσία ενδιάμεσου λογισμικού. Καλείται από τον *ContextManager* και διαβάζει την πληροφορία για το *context* και τους κανόνες από δύο προκαθορισμένα αρχεία. Κατά την διάρκεια της συντακτικής ανάλυσης,

αποσυνθέτει το *context* και τους κανόνες στα δομικά τους συστατικά και δημιουργεί μία δομή στην μνήμη, όπου αποθηκεύει το περιεχόμενο τους.

Το πέρασμα αυτής της πληροφορίας στην μνήμη επιτρέπει την γρήγορη προσπέλαση των δεδομένων και κοστίζει λιγότερο υπολογιστικά από την αποθήκευση σε αρχείο. Η δομή είναι ένα δέντρο αυθαίρετης μορφής και η αποθήκευση γίνεται με κατάλληλο τρόπο, ώστε να διευκολύνεται η προσπέλαση των δεδομένων κατά την διάρκεια του ελέγχου των κανόνων. Η κατασκευή του δέντρου γίνεται ως εξής:

- Κάθε κανόνας αποθηκεύεται σε ένα κόμβο μίας απλά συνδεδεμένης λίστας.
- Στον κόμβο του κάθε κανόνα συνδέονται δύο ακόμα λίστες στις οποίες αποθηκεύεται το τμήμα συνθηκών και το τμήμα ενεργειών.
- Στην λίστα των συνθηκών, σε κάθε κόμβο αποθηκεύεται μία συνθήκη. Αν η συνθήκη είναι σύνθετη, δηλαδή αποτελείται από δύο ή παραπάνω απλές συνθήκες, συνδέεται μία ακόμα λίστα που κάθε κόμβος της περιέχει μία απλή συνθήκη.
- Στην λίστα των ενεργειών σε κάθε κόμβο αποθηκεύεται μία ενέργεια.

Αφού ΄ολοκληρωθεί η συντακτική ανάλυση, το δέντρο γίνεται διαθέσιμο μέσω του *Contextmanager* στα υπόλοιπα υποσυστήματα της υπηρεσίας ενδιάμεσου λογισμικού. Συγκεκριμένα, ο *RulesEvaluator* χρησιμοποιεί τα δεδομένα του δέντρου κατά την επεξεργασία των γεγονότων.

Οι κόμβοι του δέντρου περιέχουν την απαραίτητη πληροφορία, ώστε να γίνονται όσο το δυνατόν λιγότερες αναζητήσεις. Συγκεκριμένα, κάθε κόμβος περιέχει δεδομένα για τις λίστες που συνδέονται σε αυτόν, ώστε κατά την αναζήτηση να αποφεύγονται άσκοπες προσπελάσεις μονοπατιών. Για παράδειγμα ο κόμβος κάθε κανόνα περιέχει όλη την συμβολοσειρά του κανόνα, ώστε να ανιχνεύονται όλες οι υποσυμβολοσειρές. Όμως, η χρήση ασαφούς λογικής για την αναπαράσταση του *context* και την αποτίμηση των κανόνων επιβάλει την αποθήκευση επιπλέον πληροφορίας στους κόμβους του δέντρου. Στην συνέχεια, περιγράφονται τα δεδομένα που αποθηκεύονται στους κόμβους του δέντρου σε καθένα από τα τρία μοντέλα ασαφούς λογικής.

5.4.1 Διαχείριση Γνώσης Συστήματος στο Μοντέλο Mamdani

Στο μοντέλο ασαφούς λογικής *Mamdani* απαιτείται η αποθήκευση πληροφορίας τόσο για τα ασαφή σύνολα που παρουσιάζονται στους κανόνες, όσο και για τα μεγέθη που υπολογίζονται κατά την διαδικασία ασαφούς συλλογιστικής *GMP*. Έτσι, στους κόμβους του δέντρου διατηρούνται, μεταξύ άλλων, τα εξής πεδία:

- *domain*: Το πεδίο *domain* αποθηκεύεται στους κόμβους των απλών συνθηκών και των ενεργειών. Δηλώνει το πεδίο ορισμού της παραμέτρου του *context* στην απλή συνθήκη ή ενέργεια.
- *membership_function*: Το πεδίο *membership_function* αποθηκεύεται στους κόμβους των απλών συνθηκών και των ενεργειών. Δηλώνει το είδος της συνάρτησης συμμετοχής για το ασαφές σύνολο που δίνεται στην απλή συνθήκη ή ενέργεια.
- *mf_parameters*: Το πεδίο *mf_parameters* αποθηκεύεται στους κόμβους των απλών συνθηκών και των ενεργειών. Δηλώνει τις τιμές των παραμέτρων της συνάρτησης συμμετοχής για το ασαφές σύνολο που δίνεται στην απλή συνθήκη ή ενέργεια.
- *membership_value*: Το πεδίο *membership_value* αποθηκεύεται στους κόμβους των απλών συνθηκών. Δηλώνει τον βαθμό συμμετοχής της τρέχουσας τιμής της παραμέτρου του *context* στο ασαφές σύνολο που καθορίζει η συνθήκη.
- *truth_value*: Το πεδίο *truth_value* αποθηκεύεται στους κόμβους των σύνθετων συνθηκών. Δηλώνει την τιμή ικανοποίησης της σύνθετης συνθήκης και υπολογίζεται με βάση τους βαθμούς συμμετοχής των απλών συνθηκών που την αποτελούν.
- *rule_weight*: Το πεδίο *rule_weight* αποθηκεύεται στον κόμβο κάθε κανόνα. Δηλώνει το βάρος του κανόνα, το οποίο υπολογίζεται κατά την συλλογιστική διαδικασία *GMP*.

Ο υπολογισμός και η χρήση των μεγεθών αυτών θα περιγραφεί στην επόμενη ενότητα, όπου αναλύεται η αποτίμηση των κανόνων με βάση το μοντέλο ασαφούς λογικής *Mamdani*.

5.4.2 Διαχείριση Γνώσης Συστήματος στο Μοντέλο Sugeno

Το μοντέλο ασαφούς λογικής *Sugeno* διαφοροποιείται από το μοντέλο *Mamdani* ως προς την πληροφορία που αποθηκεύεται στους κόμβους των ενεργειών. Έτσι, στον κόμβο κάθε ενέργειας διατηρούνται τα εξής πεδία:

- *function_parameters*: Το πεδίο *function_parameters* δηλώνει τις τιμές των παραμέτρων της συνάρτησης μέσω της οποίας περιγράφεται η συγκεκριμένη ενέργεια.
- *context_parameters*: Το πεδίο *context_parameters* δηλώνει τις παραμέτρους του *context* από τις οποίες εξαρτάται η συγκεκριμένη ενέργεια. Το πεδίο αυτό δεν χρειάζεται όταν η συνάρτηση είναι σταθερή.
- *context_values*: Το πεδίο *context_values* δηλώνει τις τρέχουσες τιμές των παραμέτρων του *context* από τις οποίες εξαρτάται η συγκεκριμένη ενέργεια. Ομοίως, το πεδίο *context_values* δεν χρειάζεται όταν η συνάρτηση είναι σταθερή.

5.4.3 Διαχείριση Γνώσης Συστήματος στο Μοντέλο Tsukamoto

Στο μοντέλο ασαφούς λογικής *Tsukamoto*, αποθηκεύεται η ίδια πληροφορία στους κόμβους του δέντρου με το μοντέλο *Mamdani*. Όμως, στους κόμβους των ενεργειών αποθηκεύονται οι αντίστροφες συναρτήσεις των συναρτήσεων συμμετοχής των ασαφών συνόλων που περιέχουν. Αυτό διότι, οι αντίστροφες συναρτήσεις είναι αυτές που χρησιμοποιούνται, κατά την διαδικασία εξαγωγής συμπερασμάτων, σε ένα σύστημα ασαφούς λογικής τύπου *Tsukamoto*.

5.4.4 Διαχείριση Γνώσης Συστήματος για Μη – Ασαφείς Προτάσεις

Οι κανόνες, σε καθένα από τα τρία μοντέλα ασαφούς λογικής, μπορεί να περιέχουν εκφράσεις που ακολουθούν το συντακτικό του Πίνακα 5.6. Όταν μία τέτοια έκφραση περιέχεται στο τμήμα συνθηκών ενός κανόνα, τότε στον κόμβο της συνθήκης αποθηκεύεται το πεδίο *truth_value*. Το μέγεθος αυτό έχει την τιμή 1 ή 0, ανάλογα με το αν ικανοποιείται η συνθήκη ή όχι. Παρόμοια, όταν μία τέτοια έκφραση περιέχεται στο τμήμα ενεργειών ενός κανόνα, τότε χρησιμοποιείται η ίδια η παράμετρος του *context* ή η τιμή της, όταν πρόκειται να εκτελεστεί η συγκεκριμένη ενέργεια.

5.5 Αποτίμηση Κανόνων

Ο *RulesEvaluator* καλείται από τον *EventHandler* και ελέγχει την ικανοποιησιμότητα των κανόνων. Αποτελεί ουσιαστικά ένα έμπειρο σύστημα, το οποίο λαμβάνει αποφάσεις με βάση τους κανόνες και την γνώση που παρέχουν τα γεγονότα που συλλέγει η υπηρεσία ενδιάμεσου

λογισμικού. Ο *RulesEvaluator* λαμβάνει τα γεγονότα από την ουρά του *EventHandler* και ελέγχοντας τους κανόνες αποφασίζει αν πρέπει να εκτελεστεί το τμήμα ενεργειών κάποιου κανόνα. Επιπλέον, είναι υπεύθυνος να διατηρήσει την συνέπεια της πληροφορίας στο δέντρο των κανόνων κάνοντας τις απαραίτητες ενημερώσεις.

Η λειτουργία του *RulesEvaluator* προσδιορίζεται πλήρως σε μία μέθοδο που υλοποιείται από τον προγραμματιστή της εφαρμογής. Στην μέθοδο αυτή καθορίζεται το μοντέλο ασαφούς λογικής καθώς και οι συναρτήσεις για τους τελεστές που χρησιμοποιούνται σε κάθε στάδιο της διαδικασίας ασαφούς συλλογιστικής.

5.5.1 Αποτίμηση Κανόνων στο Μοντέλο Mamdani

Το μοντέλο ασαφούς λογικής *Mamdani*, όπως αναφέρθηκε στο Κεφάλαιο 4 ακολουθεί την διαδικασία ασαφούς συλλογιστικής *GMP*. Η διαδικασία *GMP* διαιρείται στα εξής στάδια: (α) ασαφοποίηση της εισόδου, (β) υπολογισμός των βαρών των κανόνων, (γ) υπολογισμός επιμέρους αποτελεσμάτων και (δ) σύνθεση επιμέρους αποτελεσμάτων. Επιπλέον, ένα σύστημα ασαφούς λογικής τύπου *Mamdani* περιλαμβάνει το στάδιο της αποασαφοποίησης. Στην συνέχεια, περιγράφεται αναλυτικά η εκτέλεση των σταδίων αυτών από τον *RulesEvaluator*.

Ασαφοποίηση Εισόδου

Για κάθε νέο γεγονός, διατρέχεται η λίστα των κανόνων και ελέγχεται αν η παράμετρος του *context* στο γεγονός συμμετέχει σε κάποιο κανόνα. Αν συμμετέχει, διατρέχεται η λίστα συνθηκών του κανόνα και ελέγχεται αν συμμετέχει σε κάποια απλή ή σύνθετη συνθήκη. Αν συμμετέχει σε μία απλή συνθήκη, υπολογίζεται ο βαθμός συμμετοχής της τιμής της παραμέτρου του *context* στο ασαφές σύνολο που προσδιορίζει η συνθήκη, με βάση τις τιμές των πεδίων *membership_function* και *mf_parameters* της απλής συνθήκης. Έτσι, ενημερώνεται η τιμή του πεδίου *membership_value* της απλής συνθήκης. Αν συμμετέχει σε μία σύνθετη συνθήκη, διατρέχεται η λίστα των απλών υποσυνθηκών και ενημερώνεται το πεδίο *membership_value* της απλής συνθήκης στην οποία υπάρχει η συγκεκριμένη παράμετρος του *context*. Στην συνέχεια, ενημερώνεται το πεδίο *truth_value* της σύνθετης συνθήκης. Συγκεκριμένα, αν η σύνθετη συνθήκη αποτελείται από την σύζευξη απλών συνθηκών, τότε υπολογίζεται η *T-norm* συνάρτηση των τιμών *membership_value* των απλών

συνθηκών. Ενώ, αν η σύνθετη συνθήκη αποτελείται από την διάζευξη απλών συνθηκών, τότε υπολογίζεται η *S-norm* συνάρτηση των τιμών *membership_value* των απλών συνθηκών. Οι τύποι των *T-norm* και *S-norm* συναρτήσεων που χρησιμοποιούνται προσδιορίζονται στην μέθοδο περιγραφής του μοντέλου ασαφούς λογικής.

Υπολογισμός του Βάρους των Κανόνων

Στο στάδιο αυτό υπολογίζεται το νέο βάρος των κανόνων στους οποίους συμμετέχει η παράμετρος του *context*. Αν ο κανόνας είναι σε *CNF* μορφή, τότε υπολογίζεται η *T-norm* συνάρτηση των τιμών *membership_value* των απλών συνθηκών και των τιμών *truth_value* των σύνθετων συνθηκών του κανόνα. Αν ο κανόνας είναι σε *DNF* μορφή, τότε υπολογίζεται η *S-norm* συνάρτηση των συγκεκριμένων τιμών των απλών συνθηκών και των σύνθετων συνθηκών του κανόνα. Η τιμή αυτή αποθηκεύεται στο πεδίο *rule_weight* του κανόνα.

Υπολογισμός Επιμέρους Αποτελεσμάτων

Για κάθε κανόνα του οποίου το βάρος έχει μεταβληθεί, υπολογίζεται η νέα τιμή για τις παραμέτρους του *context* που υπάρχουν στο τμήμα ενεργειών του κανόνα. Πιο αναλυτικά, έστω E μία παράμετρος του context και $LV = \{lv_1, lv_2, ..., lv_k\}$ το σύνολο των ασαφών συνόλων που έχει οριστεί για αυτή την παράμετρο. Για την ενέργεια $E = lv_j$ σε ένα κανόνα, όπου $lv_j \in LV$, εκτελούνται τα εξής βήματα:

1. Υπολογίζεται η σχέση επαγωγής μεταξύ της τιμής lv_j της παραμέτρου του *context* E και του βάρους του κανόνα. Δηλαδή, μεταβάλλεται η συνάρτηση συμμετοχής του αντίστοιχου ασαφούς συνόλου με βάση το βάρος του κανόνα. Ως τελεστής επαγωγής, χρησιμοποιείται, συνήθως, μία $T - norm$ συνάρτηση αντί για κάποια από τις συναρτήσεις επαγωγής.
2. Το Βήμα 1 επαναλαμβάνεται για όλες τις ενέργειες που μεταβάλλουν την τιμή της παραμέτρου E στους υπόλοιπους κανόνες. Δηλαδή για όλες τις ενέργειες της μορφής $E = lv_i$, όπου $i \in [1, k]$, υπολογίζεται η σχέση επαγωγής μεταξύ της τιμής lv_i και του βάρους του κανόνα στον οποίο εμφανίζεται η ενέργεια.

Με αυτή την διαδικασία, για κάθε παράμετρο του *context* E στο τμήμα ενεργειών των κανόνων που έχει μεταβληθεί το βάρος τους, υπολογίζεται ένα νέο σύνολο ασαφών συνόλων

$LV' = \{lv_1', lv_2', ..., lv_k'\}$. Ο συναρτήσεις συμμετοχής των ασαφών συνόλων lv_i', όπου $i \in [1,k]$, έχουν προκύψει με μεταβολή των αντίστοιχων συναρτήσεων των ασαφών συνόλων lv_i με βάση το βάρος των κανόνων.

Σύνθεση Επιμέρους Αποτελεσμάτων

Για κάθε παράμετρο του *context* E στο τμήμα ενεργειών των κανόνων που έχει μεταβληθεί το βάρος τους, υπολογίζεται η ένωση των ασαφών συνόλων του συνόλου LV'. Η *S-norm* συνάρτηση που προσδιορίζεται στην μέθοδο περιγραφής του ασαφούς μοντέλου χρησιμοποιείται ως τελεστής ένωσης.

Αποασαφοποίηση Αποτελεσμάτων

Για κάθε παράμετρο του *context* E στο τμήμα ενεργειών των κανόνων που έχει μεταβληθεί το βάρος τους, υπολογίζεται μία διακριτή τιμή για το ασαφές σύνολο που προέκυψε από το προηγούμενο στάδιο. Για την διαδικασία αποασαφοποίησης χρησιμοποιείται είτε η μέθοδος *centroid*, είτε η μέθοδος *maximum*.

5.5.2 Αποτίμηση Κανόνων στο Μοντέλο Sugeno

Στο μοντέλο ασαφούς λογικής *Sugeno*, ο υπολογισμός των βαρών των κανόνων γίνεται ακριβώς με τον ίδιο τρόπο, όπως και στο μοντέλο *Mamdani*. Στην συνέχεια, για κάθε κανόνα που έχει μεταβληθεί το βάρος του, υπολογίζεται η νέα τιμή για τις παραμέτρους του *context* που υπάρχουν στο τμήμα ενεργειών του κανόνα. Έστω, λοιπόν, E μία παράμετρος του *context* και $F = \{f_1, f_2, ..., f_k\}$ το σύνολο των συναρτήσεων που έχει οριστεί για αυτή την παράμετρο. Για την ενέργεια $E = f_j$ σε ένα κανόνα, όπου $f_j \in F$, εκτελούνται τα ακόλουθα βήματα:

1. Υπολογίζεται η τιμή της συνάρτησης f_j της παραμέτρου του *context* E με βάση τις τρέχουσες τιμές των παραμέτρων του *context* από τις οποίες εξαρτάται η παράμετρος E. Για τον υπολογισμό χρησιμοποιούνται οι τιμές των εξής πεδίων της ενέργειας $E = f_j$: *function_parameters*, *context_parameters*, *context_values*. Το πεδίο *context_values* ενημερώνεται για κάθε νέο γεγονός που αφορά τις παραμέτρους που δίνονται στο πεδίο *context_parameters*.

2. Το Βήμα 1 επαναλαμβάνεται για όλες τις ενέργειες που μεταβάλλουν την τιμή της παραμέτρου E στους υπόλοιπους κανόνες. Δηλαδή για όλες τις ενέργειες της μορφής $E = f_i$, όπου $i \in [1,k]$, υπολογίζεται η τιμή της συνάρτησης f_i της παραμέτρου του *context* E.
3. Με βάση τα Βήματα 1 και 2, για κάθε παράμετρο του *context* E στο τμήμα ενεργειών των κανόνων που έχει μεταβληθεί το βάρος τους, έχει υπολογιστεί ένα σύνολο τιμών $E = \{E_1, E_2, ..., E_k\}$. Η τελική τιμή της παραμέτρου E υπολογίζεται από την Εξίσωση (5.1) ή από την Εξίσωση (5.2), όπου w_i το βάρος του κανόνα στον οποίο υπολογίστηκε η τιμή E_i.

$$E = \frac{\sum_{i=1}^{k} w_i E_i}{\sum_{i=1}^{k} w_i} \qquad \text{Εξ. 5.1}$$

$$E = \sum_{i=1}^{k} w_i E_i \qquad \text{Εξ. 5.2}$$

5.5.3 Αποτίμηση Κανόνων στο Μοντέλο Tsukamoto

Παρόμοια, στο μοντέλο ασαφούς λογικής *Tsukamoto*, ο υπολογισμός των βαρών των κανόνων γίνεται ακριβώς με τον ίδιο τρόπο, όπως και στο μοντέλο *Mamdani*. Στην συνέχεια, για κάθε κανόνα του οποίου το βάρος έχει μεταβληθεί, υπολογίζεται η νέα τιμή για τις παραμέτρους του *context* που υπάρχουν στο τμήμα ενεργειών του κανόνα. Συγκεκριμένα, έστω E μία παράμετρος του *context* και $LV = \{lv_1, lv_2, ..., lv_k\}$ το σύνολο των ασαφών συνόλων που έχει οριστεί για αυτή την παράμετρο. Για την ενέργεια $E = lv_j$ σε ένα κανόνα, όπου $lv_j \in LV$, εκτελούνται τα εξής βήματα:

1. Υπολογίζεται η τιμή του στοιχείου του ασαφούς συνόλου lv_j που ο βαθμός συμμετοχής του ισούται με το βάρος του κανόνα. Η τιμή αυτή υπολογίζεται μέσω της αντίστροφης συνάρτησης της συνάρτησης συμμετοχής του ασαφούς συνόλου lv_j. Η τιμή αυτή αποτελεί την τιμή της παραμέτρου του *context* E για αυτόν τον κανόνα.

2. Το Βήμα 1 επαναλαμβάνεται για όλες τις ενέργειες που μεταβάλλουν την τιμή της παραμέτρου E στους υπόλοιπους κανόνες. Δηλαδή για όλες τις ενέργειες της μορφής $E = lv_i$, όπου $i \in [1,k]$, υπολογίζεται η τιμή της παραμέτρου του *context* E, μέσω της αντίστροφης συνάρτησης της συνάρτησης συμμετοχής του ασαφούς συνόλου lv_i και του βάρους του κάθε κανόνα.
3. Με βάση τα Βήματα 1 και 2, για κάθε παράμετρο του *context* E στο τμήμα ενεργειών των κανόνων που έχει μεταβληθεί το βάρος τους, έχει υπολογιστεί ένα σύνολο τιμών $E = \{E_1, E_2, ..., E_k\}$. Η τελική τιμή της παραμέτρου E υπολογίζεται από την Εξίσωση (5.1) ή από την Εξίσωση (5.2).

5.5.4 Αποτίμηση Κανόνων για Μη – Ασαφείς Προτάσεις

Η ύπαρξη εκφράσεων στους κανόνες που ακολουθούν το συντακτικό του Πίνακα 5.6 επηρεάζει με συγκεκριμένο τρόπο την λειτουργία των τριών μοντέλων. Όταν μία τέτοια έκφραση περιέχεται στο τμήμα συνθηκών ενός κανόνα, τότε το πεδίο *truth_value* της αντίστοιχης απλής συνθήκης ενημερώνεται σε 1 ή 0, ανάλογα με το αν ισχύει η συνθήκη ή όχι. Δηλαδή, κάθε γεγονός, για την παράμετρο του *context* που υπάρχει στην απλή συνθήκη, μεταβάλει την τιμή του πεδίου *truth_value* με βάση το αν ικανοποιείται η συνθήκη. Στην συνέχεια, η τιμή του πεδίου *truth_value* χρησιμοποιείται κανονικά, όπως και οι τιμές των πεδίων *membership_value* και *truth_value* των απλών και σύνθετων συνθηκών αντίστοιχα, για τον υπολογισμό του βάρους κάθε κανόνα.

Όταν στο τμήμα ενεργειών ενός κανόνα, που έχει μεταβληθεί το βάρος του, υπάρχει μία ενέργεια που ακολουθεί το συντακτικό του Πίνακα 5.6, τότε αρχικά ελέγχεται αν η ενέργεια έχει ως αποτέλεσμα την ανάθεση μίας τιμής σε μία παράμετρο του *context*. Αν δεν ισχύει αυτό απλά εκτελείται η ενέργεια, αλλιώς εκτελείται η εξής διαδικασία:

1. Διατρέχεται η λίστα των κανόνων και βρίσκονται όλοι οι κανόνες που μεταβάλλουν την τιμή της παραμέτρου του *context* που υπάρχει στην συγκεκριμένη ενέργεια.
2. Ως τελική τιμή, για αυτή την παράμετρο του *context*, επιλέγεται αυτή που δίνεται από τον κανόνα με το μέγιστο βάρος. Αν υπάρχουν παραπάνω από ένας κανόνες με το ίδιο μέγιστο βάρος, που δίνουν διαφορετική τιμή στην παράμετρο του *context*, τότε

υπάρχει «σύγκρουση». Στην περίπτωση αυτή, επιλέγεται τυχαία μία από τις τιμές για την παράμετρο του context.

5.5.5 Εκτέλεση Ενεργειών

Η τελική τιμή που υπολογίζεται για κάθε παράμετρο του *context* αποστέλλεται στην εφαρμογή, μέσω μίας μεθόδου που υλοποιείται από τον προγραμματιστή, κατά την διασύνδεση της εφαρμογής με την υπηρεσία ενδιάμεσου λογισμικού. Η μέθοδος αυτή καλείται από τον *RulesEvaluator* με ορίσματα την παράμετρο του *context* και την τιμή της. Στην περίπτωση, όμως, που μία ενέργεια δεν αναθέτει κάποια τιμή σε μία παράμετρο αλλά έχει ως αποτέλεσμα την κλήση μίας μεθόδου της εφαρμογής χωρίς ορίσματα, τότε το δεύτερο όρισμα είναι *null*. Η μέθοδος αντιστοιχεί τα ορίσματα εισόδου με την κλήση μίας μεθόδου της εφαρμογής που εκτελεί την ενέργεια. Η μέθοδος που καλεί ο *RulesEvaluator* ονομάζεται *ActionResolve* και αποτελεί ένα *callback* που χρησιμοποιεί η υπηρεσία ενδιάμεσου λογισμικού για να ειδοποιήσει τη εφαρμογή για τις ενέργειες προσαρμογής. Η μέθοδος περιγράφεται αναλυτικότερα στο Κεφάλαιο 6.

5.6 Δυναμική Διαμόρφωση της Υπηρεσίας Ενδιάμεσου Λογισμικού

Τα υποσυστήματα *ContextRecordManager* και *RulesRecordManager* περιλαμβάνουν ένα σύνολο μεθόδων για την δυναμική διαμόρφωση της υπηρεσίας ενδιάμεσου λογισμικού από τον χρήστη και παρέχουν δύο γραφικές διεπαφές χρήστη (*GUIs*) που μπορεί να χρησιμοποιήσει ο χρήστης για να προσπελάσει αυτές τις μεθόδους: μία για την διαμόρφωση του *context* και μία για την διαμόρφωση των κανόνων.

5.6.1 Διαμόρφωση Context

Το *GUI* για την διαμόρφωση του context περιλαμβάνει επιλογές για τις εξής λειτουργίες:

Εμφάνιση Context

Ο χρήστης μπορεί να δει στην οθόνη της συσκευής του τις παραμέτρους του *context* και την πληροφορία που σχετίζεται με κάθε παράμετρο. Για παράδειγμα για μία ασαφή παράμετρο μπορεί να δει τα ασαφή σύνολα που έχουν οριστεί για αυτή και τις συναρτήσεις συμμετοχής

τους. Οι παράμετροι του *context* εμφανίζονται με την σειρά εισαγωγής τους, συνοδευόμενοι από ένα αύξοντα *id*.

Εισαγωγή Context

Ο χρήστης μπορεί να εισάγει νέα παράμετρο *context* και να ορίσει την απαιτούμενη πληροφορία για αυτή.

Διαγραφή Context

Ο χρήστης μπορεί να διαγράψει μία παράμετρο *context*. Η παράμετρος επιλέγεται με το *id* της. Βέβαια στην περίπτωση αυτή χρειάζεται να αλλάξει ή να διαγράψει τους κανόνες που την περιέχουν.

Αλλαγή Context

Ο χρήστης μπορεί να αλλάξει την πληροφορία που έχει οριστεί για μία παράμετρο του *context*. Για παράδειγμα μπορεί να μεταβάλλει το είδος της συνάρτησης συμμετοχής ενός ασαφούς συνόλου. Μετά την αλλαγή καλείται ο *RulesParser*, ο οποίος κάνει τις κατάλληλες ενημερώσεις στο δέντρο των κανόνων.

Αποθήκευση Context

Ο χρήστης μπορεί να αποθηκεύσει τις τρέχουσες παραμέτρους του *context* στο αρχείο με το *context*, αν επιθυμεί να αποθηκευτούν οι αλλαγές που έχει κάνει. Αν το αρχείο υπάρχει, σβήνεται και δημιουργείται νέο.

Διαγραφή Αρχείου Context

Ο χρήστης μπορεί να διαγράψει το αρχείο του *context* είτε για εξοικονόμηση χώρου, είτε για να προσθέσει νέες παραμέτρους του *context* στην συνέχεια.

5.6.2 Διαμόρφωση Κανόνων

Το *GUI* για την διαμόρφωση των κανόνων περιλαμβάνει επιλογές για τις εξής λειτουργίες:

Εμφάνιση Κανόνων

Ο χρήστης μπορεί να δει στην οθόνη της συσκευής του, όλους τους κανόνες που βρίσκονται εκείνη την στιγμή στο δέντρο. Οι κανόνες εμφανίζονται με την σειρά εισαγωγής τους, συνοδευόμενοι από αύξοντα *id*.

Εισαγωγή Νέου Κανόνα

Ο χρήστης μπορεί να γράψει ένα νέο κανόνα στην οθόνη της συσκευής του. Μετά την εισαγωγή καλείται ο *RulesParser*, ο οποίος αποσυνθέτει τον κανόνα και τον αποθηκεύει στο δέντρο.

Διαγραφή Κανόνα

Ο χρήστης μπορεί να διαγράψει ένα κανόνα. Ο κανόνας επιλέγεται με το *id* του και διαγράφεται από το δέντρο.

Αλλαγή Κανόνα

Ο χρήστης μπορεί να αλλάξει ένα κανόνα. Μετά την αλλαγή καλείται ο *RulesParser*, ο οποίος αποθηκεύει τον κανόνα στο δέντρο.

Αποθήκευση Κανόνων

Ο χρήστης μπορεί να αποθηκεύσει την τρέχουσα μορφή των κανόνων στο αρχείο με τους κανόνες, αν επιθυμεί να αποθηκευτούν οι αλλαγές που έχει κάνει. Αν το αρχείο υπάρχει, σβήνεται και δημιουργείται νέο.

Διαγραφή Αρχείου Κανόνων

Ο χρήστης μπορεί να διαγράψει το αρχείο με τους κανόνες αν το επιθυμεί. Αυτό μπορεί να γίνει είτε για εξοικονόμηση χώρου, αν δεν υπάρχει κάποιος κανόνας που το κάνει αυτόματα, είτε για να σβήσει άμεσα όλους τους κανόνες και να προσθέσει άλλους στην συνέχεια.

ΚΕΦΑΛΑΙΟ 6. ΥΛΟΠΟΙΗΣΗ ΥΠΗΡΕΣΙΑΣ ΕΝΔΙΑΜΕΣΟΥ ΛΟΓΙΣΜΙΚΟΥ

Στο κεφάλαιο αυτό περιγράφεται η υλοποίηση της υπηρεσίας ενδιάμεσου λογισμικού και δίνονται προγραμματιστικές λεπτομέρειες σχετικά με την ανάπτυξη του ενδιάμεσου λογισμικού. Προκειμένου να ελέγξουμε την χρηστικότητα και την ορθή λειτουργία της υπηρεσίας ενδιάμεσου λογισμικού υλοποιήθηκε μία κινητή εφαρμογή πάνω από την υπηρεσία. Επιπλέον, περιγράφεται η διασύνδεση της εφαρμογής με την υπηρεσία και η επικοινωνία μεταξύ τους.

6.1 Πλατφόρμα Υλοποίησης

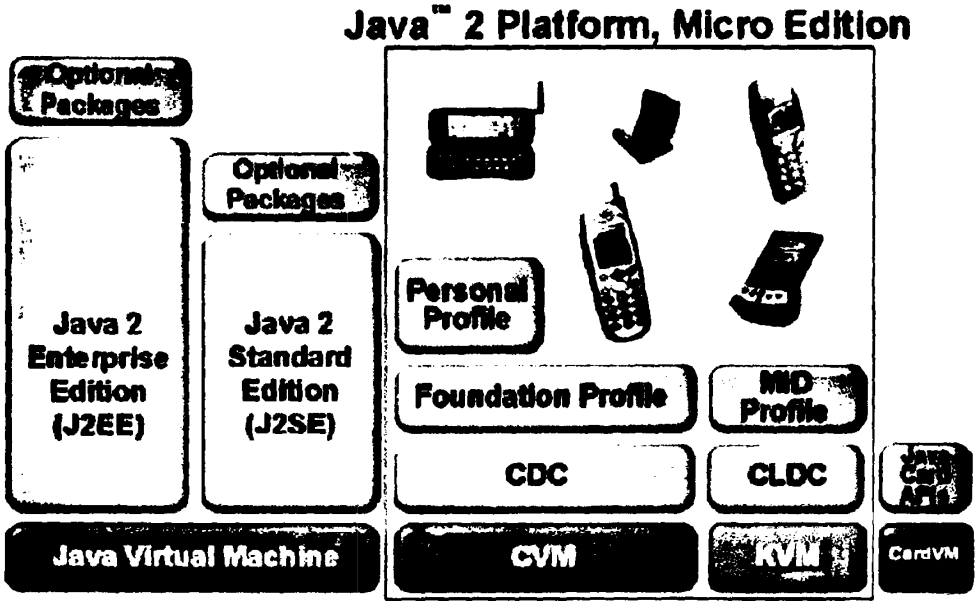
Η υλοποίηση της υπηρεσίας έγινε με χρήση της πλατφόρμας *J2ME* (*Java 2 Micro Edition*) της *Java*. Οι τεχνολογίες αυτής της πλατφόρμας χρησιμοποιούνται από ένα πλήθος κινητών συσκευών, όπως κινητά τηλέφωνα, *PDAs*, φορέσιμες συσκευές, ενσωματωμένα συστήματα αυτοκινήτων κ.α. Αποτελείται από ένα σύνολο από καθορισμένα πρότυπα, *Java APIs* και προαιρετικά πακέτα. Στο Σχήμα 6.1 φαίνεται η αρχιτεκτονική της πλατφόρμας *J2ME*.

Η πλατφόρμα περιλαμβάνει ένα σύνολο συστατικών που επιτρέπουν την ανάπτυξη εφαρμογών για κινητές συσκευές. Τα συστατικά αυτά αφορούν διαφορετικές συσκευές με πολλές δυνατότητες και χαρακτηριστικά. Η *J2ME* υποστηρίζεται από δύο κατηγορίες

κινητών συσκευών, με βάση το μέγεθος των πόρων που διαθέτουν. Για κάθε μία παρέχει ξεχωριστό περιβάλλον εκτέλεσης. Ακολουθεί μία σύντομη περιγραφή των συστατικών της πλατφόρμας *J2ME*.

CDC

Το *CDC* (*Connected Device Configuration*) αποτελεί την προδιαγραφή για το ένα από τα δύο περιβάλλοντα εκτέλεσης της *J2ME* (*CDC* και *CLDC*). Παρέχει μία εικονική μηχανή (*CVM*) και μαζί με το *FP* (*Foundation Profile*) αποτελεί ένα πλήρες περιβάλλον εκτέλεσης *J2ME*. Αφορά κινητές συσκευές μεσαίου μεγέθους, όπως μεγάλα *PDAs*, συστήματα πλοήγησης αυτοκινήτων κ.α. Οι συσκευές αυτές έχουν σχετικά περιορισμένους υπολογιστικούς πόρους.



Σχήμα 6.1: Αρχιτεκτονική της πλατφόρμας J2ME.

FP

Το *FP* είναι το βασικό προφίλ του *CDC*. Περιλαμβάνει τα *APIs* για την ανάπτυξη εφαρμογών πάνω από το *CDC*. Το *FP* προορίζεται για εφαρμογές που δεν απαιτούν γραφικές διεπαφές. Άλλα προφίλ, όπως το *Personal Basis Profile* και το *Personal Profile* είναι χτισμένα πάνω στο *FP* παρέχοντας γραφικές διεπαφές και άλλες επιπλέον δυνατότητες.

CLDC

Το *CLDC* (*Connected Limited Device Configuration*) είναι η βάση του περιβάλλοντος εκτέλεσης της *J2ME* που στοχεύει σε μικρές κινητές συσκευές με ιδιαίτερα περιορισμένους

υπολογιστικούς πόρους. Οι συσκευές αυτές περιλαμβάνουν κυρίως απλά κινητά τηλέφωνα, μικρούς ψηφιακούς βοηθούς κ.α. Το *CLDC* περιλαμβάνει δύο εικονικές μηχανές για την εκτέλεση των εφαρμογών, την *KVM* και την *CLDC HotSpot*. Η *KVM* σχεδιάστηκε για συσκευές με εξαιρετικά μικρές υπολογιστικές δυνατότητες που μπορούν να υποστηρίξουν υποτυπώδεις κινητές εφαρμογές. Η *CLDC HotSpot* είναι σχεδιασμένη για συσκευές νέας γενιάς με περισσότερη διαθέσιμη μνήμη. Οι βασικές συσκευές που υλοποιούν το *CLDC* για την ανάπτυξη εφαρμογών είναι τα κινητά τηλέφωνα.

MIDP

Το *MIDP* (*Mobile Information Device Profile*) μαζί με το *CLDC* συνθέτουν το περιβάλλον εκτέλεσης *Java* για κινητές συσκευές περιορισμένων πόρων. Το *MIDP* περιλαμβάνει ένα σύνολο από *APIs* που παρέχουν την βασική λειτουργικότητα για κινητές εφαρμογές, όπως γραφικές διεπαφές χρηστών, δικτυακή επικοινωνία, τοπικό αποθηκευτικό χώρο και διαχείριση του κύκλου ζωής των εφαρμογών.

Προαιρετικά Πακέτα

Η πλατφόρμα *J2ME* περιλαμβάνει ένα σύνολο προαιρετικών πακέτων που επεκτείνουν την λειτουργικότητα των κινητών εφαρμογών. Τα πιο γνωστά από αυτά είναι το *MMA* (*Mobile Media API*) για υπηρεσίες πολυμέσων, το *WMA* (*Wireless Messaging API*) για ασύρματη αποστολή μηνυμάτων, το *WSA* (*Web Services API*) για προσπέλαση υπηρεσιών διαδικτύου από την κινητή εφαρμογή κ.α. Διάφορα άλλα *APIs* παρέχονται από συγκεκριμένους κατασκευαστές κινητών συσκευών και είναι συμβατά μόνο με συγκεκριμένες συσκευές, όπως το *Location API* της *NOKIA*.

Επιλογή Πλατφόρμας

Η άλλη επιλογή, ανταγωνιστική της τεχνολογίας *J2ME* είναι το *.Net Compact Framework* της *Microsoft*. Και οι δύο τεχνολογίες υποστηρίζουν ένα πλήθος προγραμματιστικών εργαλείων που διευκολύνουν την αποτελεσματική ανάπτυξη κινητών εφαρμογών. Η βασική διαφορά τους είναι η ανεξαρτησία από *hardware* πλατφόρμες.

Οι τεχνολογίες *J2ME* είναι συμβατές με ένα μεγάλο εύρος συσκευών. Οι προδιαγραφές της *J2ME* έχουν υλοποιηθεί από όλους τους βασικούς κατασκευαστές κινητών τηλεφώνων, όπως *Ericsson, Motorola, Nokia, Sieemens* κ.α. Οποιαδήποτε εφαρμογή σε *MIDP* μπορεί να τρέξει

σε όλες τις συσκευές που υποστηρίζουν εφαρμογές *J2ME*. Συνεπώς, το λογισμικό *MIDP* ικανοποιεί σε μεγάλο βαθμό την βασική απαίτηση της μεταφερσιμότητας.

Αντίθετα, το *.Net Compact Framework* τρέχει μόνο στις συσκευές που έχουν την άδεια *Pocket PC* της *Microsoft*. Προτιμήθηκε η πλατφόρμα με την μεγαλύτερη διαθεσιμότητα, η οποία επιπλέον παρέχονταν δωρεάν.

6.2 Κινητές Εφαρμογές

Οι εφαρμογές που υλοποιούνται σε *MIDP* ονομάζονται *MIDlets*. Το *MIDlet* είναι το κωδικό όνομα για τις εφαρμογές που έχουν αναπτυχθεί με τεχνολογίες *Java* και τρέχουν σε ασύρματες και κινητές συσκευές. Τα *MIDlets* είναι το ανάλογο όνομα των *Applets* για κινητές συσκευές. Όπως και τα *Applets*, τα *MIDlets* ελέγχονται από το λογισμικό που τα τρέχει. Στην περίπτωση των *Applets*, το λογισμικό που τα τρέχει είναι ένας *browser* ή κάποιο ανάλογο εργαλείο. Τα *MIDlets* τρέχουν στην υλοποίηση του *CLDC* σε μία κινητή συσκευή.

Η υπηρεσία ενδιάμεσου λογισμικού υλοποιήθηκε αποκλειστικά σε *MIDP*, ώστε να μπορεί να τρέξει κάτω από *MIDlets*. Για την υλοποίηση και τον έλεγχο του λογισμικού χρησιμοποιήθηκαν προσομοιωτές κινητών συσκευών. Συγκεκριμένα, το λογισμικό αναπτύχθηκε και ελέγχθηκε με το εργαλείο *J2ME Wireless Toolkit* (*WTK*) *2.5* της *SUN*. Διάφορα άλλα εργαλεία προσομοίωσης παρέχονται από κατασκευαστές όπως η *Ericsson*, η *Motorola*, η *Nokia* και η *Siemens*.

Οι προσομοιωτές στοχεύουν τόσο στην προσομοίωση της λειτουργίας μία συσκευής, όσο και στα ιδιαίτερα χαρακτηριστικά κάποιου πραγματικού μοντέλου. Παρέχουν σύνολο επιλογών για την ακριβέστερη προσομοίωση στην εκτέλεση της εφαρμογής. Επιπλέον, παρέχουν εργαλεία για την παρακολούθηση των βασικών παραμέτρων στην εκτέλεση της εφαρμογής, όπως η κατανάλωση μνήμης, η χρήση του επεξεργαστή κ.α.

6.3 Υλοποίηση της Υπηρεσίας Ενδιάμεσου Λογισμικού

Η υπηρεσία ενδιάμεσου λογισμικού παρέχεται με την μορφή ενός *MIDP* πακέτου. Κάθε υποσύστημα της υπηρεσίας υλοποιήθηκε ως ξεχωριστή κλάση με μεθόδους και παραμέτρους

που υλοποιούν τις λειτουργίες του υποσυστήματος. Επίσης, το πακέτο περιλαμβάνει και κάποιες βοηθητικές κλάσεις. Σχεδιαστικά, ίσως ήταν προτιμότερο οι λειτουργίες των υποσυστημάτων να κατανεμηθούν σε περισσότερες κλάσεις επιτυγχάνοντας μία περισσότερο δομημένη αρχιτεκτονική. Οι κλάσεις που υλοποιήθηκαν είναι οι ελάχιστες απαραίτητες, αφού μεγάλος αριθμός κλάσεων μειώνει την χρονική απόδοση του συστήματος, ενώ αυξάνει το μέγεθος του τελικού εκτελέσιμου αρχείου (*JAR*) που θα φορτωθεί στην μνήμη της κινητής συσκευής.

6.3.1 Πακέτο contextManager

Συνοπτικά, η υπηρεσία ενδιάμεσου λογισμικού αποτελείται από τα εξής:

Package
contextManager

Interface
ContextManagement

Classes
ContextManager
RulesParser
EventHandler
RulesEvaluator
ContextRecordManagement
RulesRecordManagement

Οι κλάσεις του πακέτου υλοποιούν τα αντίστοιχα υποσυστήματα που περιγράφηκαν στο Κεφάλαιο 5. Η διεπαφή *ContextManagement* ορίζει τις μεθόδους που συνδέουν μία εφαρμογή με την υπηρεσία ενδιάμεσου λογισμικού και περιγράφεται στην συνέχεια.

Αν κάποιος θέλει να υλοποιήσει μία *context – aware* εφαρμογή, μπορεί να το κάνει χρησιμοποιώντας το πακέτο *contextManager*. Για να χρησιμοποιηθεί η υπηρεσία ενδιάμεσου λογισμικού από μία εφαρμογή, ο προγραμματιστής της εφαρμογής θα πρέπει να προσθέσει

κώδικα διασύνδεσης. Η διασύνδεση εφαρμογής – υπηρεσίας συνοψίζεται στις ακόλουθες ενέργειες:

1. Φορτώνεται (*import*) στην εφαρμογή το πακέτο *contextManager* κάνοντας διαθέσιμες τις ορατές κλάσεις του.
2. Η βασική κλάση (*public*) της εφαρμογής που επεκτείνει την κλάση *MIDlet*, πρέπει να υλοποιεί την διεπαφή *ContextManagement*. Η διεπαφή αυτή περιλαμβάνει τέσσερις μεθόδους που καθορίζουν την αλληλεπίδραση της εφαρμογής με την υπηρεσία.
3. Κατά την έναρξη της εφαρμογής (είτε στον κατασκευαστή, είτε στην μέθοδο *startApp()* που ξεκινά η ροή του *MIDlet*) δημιουργείται ένα αντικείμενο της κλάσης *ContextManager*. Ο κατασκευαστής καλείται με όρισμα το *context*, τους κανόνες και το μοντέλο ασαφούς λογικής που θα χρησιμοποιηθεί και αυτόματα γίνονται οι διαδικασίες αρχικοποίησης της υπηρεσίας.
4. Στην μέθοδο *destroyApp()* που τερματίζει το *MIDlet* της εφαρμογής καλείται η μέθοδος του *ContextManager*, *stopMonitor()*, που τερματίζει το νήμα παρακολούθησης των προμηθευτών.

6.3.2 Διεπαφή ContextManagement

Η διεπαφή *ContextManagement* ορίζει τέσσερις μεθόδους για την διασύνδεση με μία εφαρμογή. Ο προγραμματιστής της εφαρμογής πρέπει απαραίτητα να υλοποιήσει την διεπαφή συμπληρώνοντας το σώμα των μεθόδων. Η διεπαφή περιλαμβάνει τις ακόλουθες μεθόδους:

Interface ContextManagement

- *public String [] defineContext (String [ ] context)*
- *public String [] defineContextRules (String [] rules)*
- *public String [] defineFuzzyModel (String [] model)*
- *public void actionResolve (String parameter, String value)*

Ακολουθεί η περιγραφή των μεθόδων της διεπαφής:

public String [] defineContext (String [ ] context)

Στην μέθοδο αυτή ο προγραμματιστής καταχωρεί τις παραμέτρους του *context*. Δημιουργεί ένα πίνακα από συμβολοσειρές, του οποίου κάθε στοιχείο είναι μία παράμετρος του *context*.

Η περιγραφή των παραμέτρων του *context* γίνεται με βάση το συντακτικό που ορίστηκε στο Κεφάλαιο 5. Η μέθοδος επιστρέφει τον πίνακα στον *ContextManager* κατά την έναρξη της υπηρεσίας ενδιάμεσου λογισμικού. Αν υπάρχει ήδη το αρχείο με το *context* στην μνήμη της συσκευής, η υπηρεσία θα το διαβάσει από εκεί. Διαφορετικά, η υπηρεσία θα διαβάσει τον πίνακα και θα αποθηκεύσει τις παραμέτρους του *context* στο αρχείο. Αν ο πίνακας δεν οριστεί μέσα στην μέθοδο, το αρχείο με το *context* δεν θα πρέπει να σβηστεί ποτέ από την μνήμη της συσκευής.

Η μέθοδος δεν περιορίζει τον προγραμματιστή στον τρόπο κατασκευής του πίνακα. Μέσα στην μέθοδο μπορεί να υλοποιηθεί οποιαδήποτε διαδικασία απόκτησης του *context*. Για παράδειγμα, η μέθοδος μπορεί να υλοποιεί δικτυακή σύνδεση σε ένα κόμβο προσπελάσιμο μέσω δικτύου και να λαμβάνει ένα αρχείο με το *context*. Το αρχείο μπορεί να είναι σε μορφή εγγραφών και να βρίσκεται σε κάποια άλλη κινητή συσκευή, οπότε μπορεί να αποκτηθεί ασύρματα μέσω *Bluetooth*, *SMS*, κ.α. Μπορεί όμως να είναι και απλό αρχείο κειμένου (*text*) και να βρίσκεται σε κάποιο γνωστό εξυπηρέτη στο διαδίκτυο, από όπου μπορούν να διαβαστούν τα περιεχόμενά του μέσω *http* σύνδεσης.

public String [] defineContextRules (String [ ] rules)

Στην μέθοδο αυτή ο προγραμματιστής κατασκευάζει ένα πίνακα με τους κανόνες του *context*. Οι κανόνες ακολουθούν το συντακτικό που δόθηκε στο Κεφάλαιο 5. Ο χειρισμός της μεθόδου είναι ο ίδιος με την μέθοδο *defineContext*.

public String [] defineFuzzyModel (String [] model)

Στην μέθοδο αυτή ο προγραμματιστής προσδιορίζει το μοντέλο ασαφούς λογικής και τις συναρτήσεις για τους τελεστές που χρησιμοποιούνται σε κάθε στάδιο της διαδικασίας ασαφούς εξαγωγής συμπερασμάτων. Στο μοντέλο *Mamdani*, ο προγραμματιστής ορίζει τον τύπο των $S - norm$ και $T - norm$ συναρτήσεων, τον τύπο της συνάρτησης επαγωγής και την μέθοδο αποασαφοποίησης. Στα μοντέλα *Sugeno* και *Tsukamoto*, ο προγραμματιστής ορίζει τον τύπο των $S - norm$ και $T - norm$ συναρτήσεων και την μέθοδο αποασαφοποίησης. Ο χειρισμός της μεθόδου είναι ο ίδιος με τις μεθόδους *defineContext* και *defineContextRules*.

public void actionResolve (String parameter, String value)

Η μέθοδος αυτή ερμηνεύει τις ενέργειες που προκύπτουν από την ικανοποίηση των κανόνων, σε ενέργειες προσαρμογής της εφαρμογής. Η μέθοδος αποτελεί *callback* και καλείται μόνο από την υπηρεσία ενδιάμεσου λογισμικού. Συγκεκριμένα, καλείται από τον *RulesEvaluator* για την πραγματοποίηση των ενεργειών που αποφασίζει. Για κάθε ενέργεια, ο *RulesEvaluator* καλεί την μέθοδο με ορίσματα μία παράμετρο του *context* και την τιμή που της αναθέτει η ενέργεια. Αν η ενέργεια δεν αναθέτει τιμή σε μία παράμετρο, αλλά εκτελεί μία μέθοδο χωρίς ορίσματα, το δεύτερο όρισμα είναι *null*.

Ο προγραμματιστής υλοποιεί στο σώμα της μεθόδου μία αντιστοίχιση των παραμέτρων του *context* με μεθόδους της εφαρμογής, οι οποίες υλοποιούν τις ενέργειες προσαρμογής. Οι τιμές των παραμέτρων, αν δεν είναι *null*, περνούν σαν ορίσματα στις μεθόδους της εφαρμογής. Αν η τιμή είναι null σημαίνει ότι, η συγκεκριμένη μέθοδος που πρέπει να εκτελεστεί δεν έχει ορίσματα.

Οι ενέργειες που αποφασίζονται σε ένα κανόνα μπορεί να μην αφορούν μόνο την εφαρμογή, αλλά και την υπηρεσία ενδιάμεσου λογισμικού. Για παράδειγμα, η ικανοποίηση ενός κανόνα μπορεί να έχει σαν αποτέλεσμα το σβήσιμο του αρχείου των κανόνων ή τον τερματισμό του νήματος παρακολούθησης των προμηθευτών. Οι ενέργειες αυτές υλοποιούνται με κλήσεις στις αντίστοιχες μεθόδους του *ContextManager*. Συνεπώς, οι μέθοδοι που καλούνται μέσα από την *actionResolve* μπορεί να είναι και μέθοδοι της υπηρεσίας.

Η υπηρεσία ενδιάμεσου λογισμικού καλώντας αυτή την μέθοδο προσαρμόζει αυτόματα την λειτουργία της εφαρμογής, αλλά και την δική της, στις αλλαγές του *context*. Η υλοποίηση της μεθόδου πρέπει να αντιστοιχίζει όλες τις πιθανές ενέργειες που επιστρέφει η υπηρεσία σε μεθόδους που υλοποιούν τις ζητούμενες ενέργειες.

6.4 Χρήση της Υπηρεσίας Ενδιάμεσου Λογισμικού από Μία Κινητή Εφαρμογή

Για να γίνει επίδειξη της υπηρεσίας ενδιάμεσου λογισμικού και να ελεγχθεί η λειτουργία της σε πραγματικές συνθήκες, υλοποιήθηκε μία *MIDP* εφαρμογή για κινητές συσκευές. Υλοποιήθηκε η διασύνδεση της εφαρμογής με την υπηρεσία ακολουθώντας τα βήματα που περιγράφηκαν παραπάνω και πραγματοποιήθηκαν διάφορα σενάρια λειτουργίας. Η

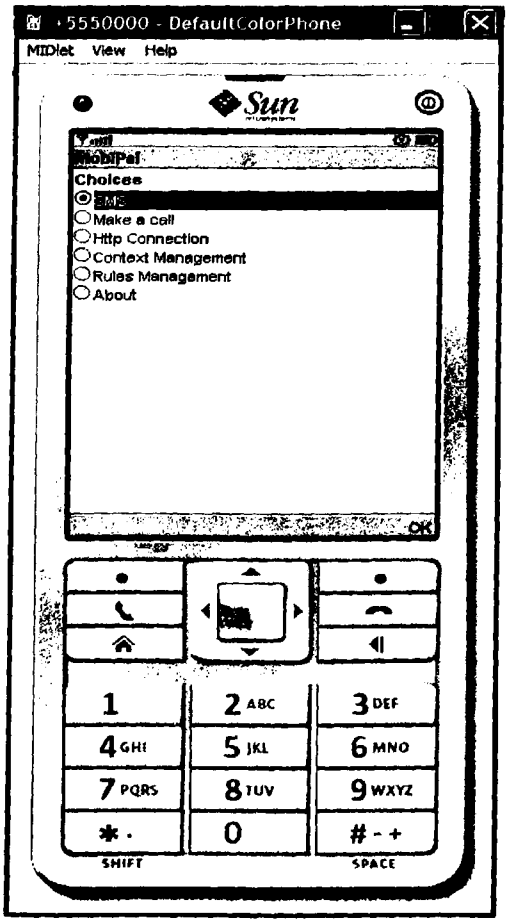
εφαρμογή, όπως και η υπηρεσία, υλοποιήθηκε και εκτελέστηκε με τον προσομοιωτή του *WTK*.

Η υλοποίηση της εφαρμογής δεν είχε στόχο την κατασκευή ενός σύνθετου *MIDlet* με πολλές λειτουργικές δυνατότητες. Το ζητούμενο ήταν να υλοποιηθεί μία εφαρμογή, η οποία να εξαρτάται από το πραγματικό *context* και μέσω της υπηρεσίας να χρησιμοποιεί το *context* για να προσαρμόζει τις λειτουργίες της.

6.4.1 Περιγραφή Εφαρμογής

Το MIDlet που υλοποιήσαμε παρέχει στον χρήστη βασικές υπηρεσίες όπως:

- Αποστολή και λήψη *SMS*.
- Χειρισμό εισερχόμενων / εξερχόμενων κλήσεων.
- Δυνατότητες αναπαραγωγής πολυμέσων.
- Σύνδεση στο διαδίκτυο.



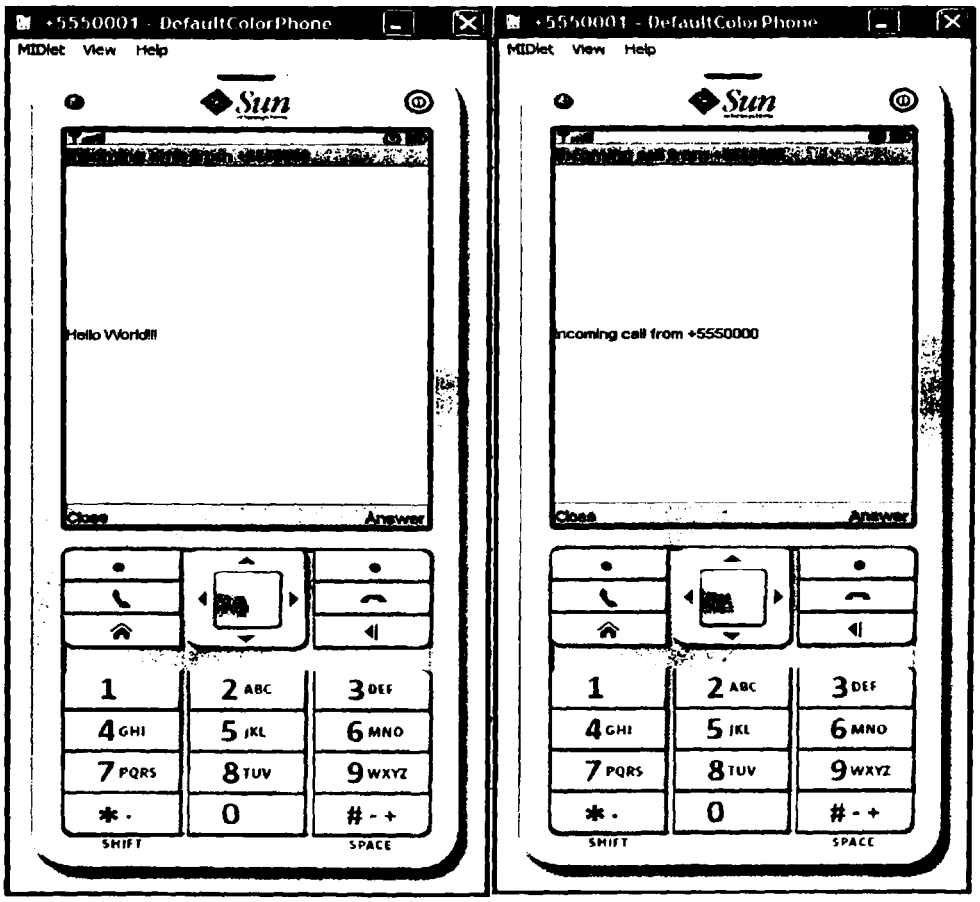
Σχήμα 6.2: Αρχική γραφική διεπαφή της εφαρμογής.

Επιπλέον, στο μενού επιλογών της εφαρμογής προστέθηκαν επιλογές εμφάνισης των γραφικών διεπαφών που παρέχει η υπηρεσία ενδιάμεσου λογισμικού, για την διαμόρφωση του *context* και των κανόνων. Στο Σχήμα 6.2 φαίνεται η αρχική γραφική διεπαφή της εφαρμογής.

Στην συνέχεια, περιγράφονται οι λειτουργίες της εφαρμογής.

Αποστολή και Λήψη SMS

Η εφαρμογή μπορεί να στέλνει και να δέχεται γραπτά μηνύματα, τα οποία εμφανίζει στην οθόνη του κινητού τηλεφώνου (Σχήμα 6.3 (α)). Η αποστολή των μηνυμάτων γίνεται στον αριθμό που αντιστοιχεί σε κάθε προσομοιωτή (π.χ. +5550000). Για την υλοποίηση αυτής λειτουργίας χρησιμοποιήθηκε επιπρόσθετα το *WMA 2.0*.



Σχήμα 6.3: (α) Εισερχόμενο SMS. (β) Εισερχόμενη κλήση.

Χειρισμός Κλήσεων

Η εφαρμογή μπορεί να δέχεται και να κάνει κλήσεις σε άλλες κινητές συσκευές. Το *MIDP* παρέχει μία μέθοδο για πραγματοποίηση εξερχόμενης κλήσης σε άλλη συσκευή, η οποία

όμως δεν λειτουργεί στους προσομοιωτές. Για τον λόγο αυτό, η κλήση υλοποιήθηκε έμμεσα στέλνοντας ένα *SMS* με το περιεχόμενο *call*. Αυτό αποτελεί μία αρκετά ρεαλιστική λύση, αφού η διαδικασία με την οποία μία εφαρμογή «ακούει» εισερχόμενες κλήσεις και *SMSs* είναι παρόμοια (Σχήμα 6.3 (β)).

Η εισερχόμενη κλήση μπορεί να συνοδεύεται από ήχο ή να είναι αθόρυβη. Το αρχείο ήχου μπορεί να είναι σε διάφορους τύπους (*wav*, *mp3*, κ.α.) και βρίσκεται αποθηκευμένο στην μνήμη της κινητής συσκευής, όπου αποθηκεύονται τα εκτελέσιμα αρχεία της εφαρμογής. Για την αναπαραγωγή ήχου χρησιμοποιήθηκε το *MMA 2.0*.

Σύνδεση στο διαδίκτυο

Η εφαρμογή παρέχει την δυνατότητα σύνδεσης στο διαδίκτυο μέσω *http*. Συγκεκριμένα ο χρήστης μπορεί:

1. Να ανοίξει από τον *browser* του κινητού τηλεφώνου κάποια ιστοσελίδα.
2. Να κατεβάσει και να δει στην οθόνη του κινητού τηλεφώνου αρχείο κειμένου που βρίσκεται σε κάποιον εξυπηρέτη του διαδικτύου.

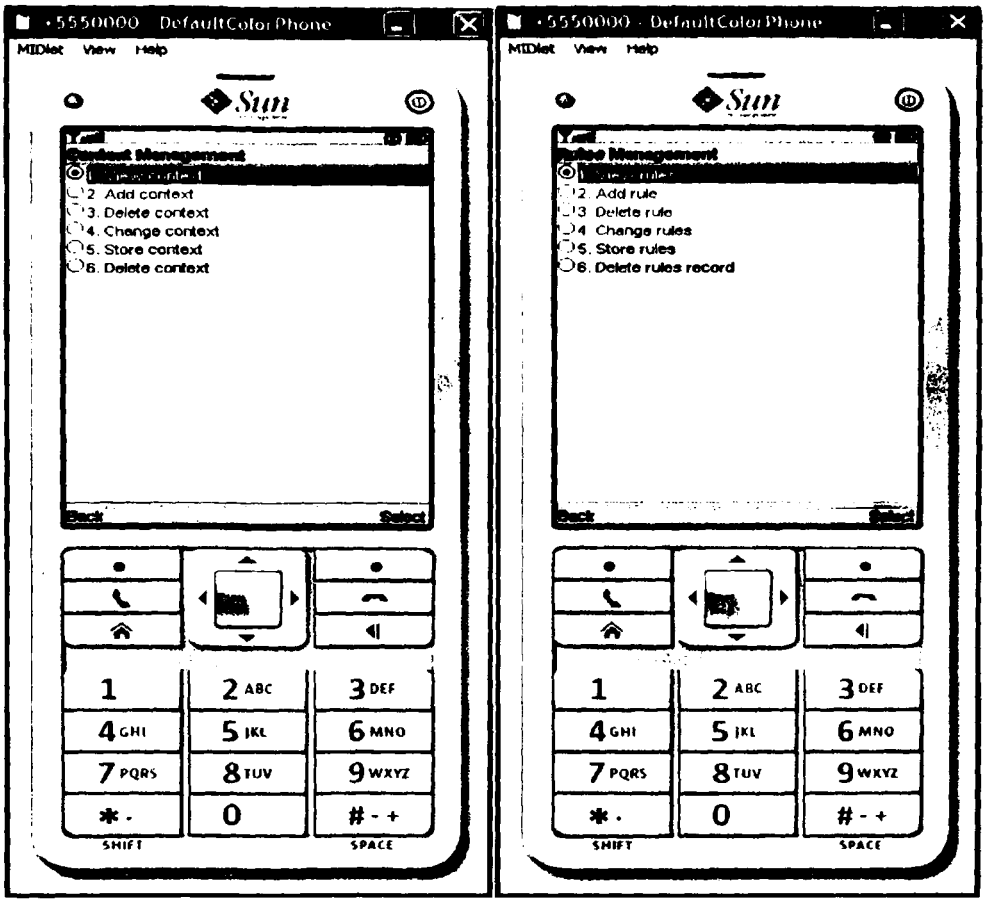
Η μέθοδος, που υλοποιεί την πρώτη λειτουργία, απαιτεί η εφαρμογή να υποστηρίζει κάποια τεχνολογία προσπέλασης ιστοσελίδων και *browser* που επιτρέπει την σύνδεση σε ειδικές σελίδες του διαδικτύου για κινητά τηλέφωνα. Η μέθοδος αναζητά κάποιο διαθέσιμο *browser* στο λογισμικό που εκτελείται η εφαρμογή και ανοίγει εκεί την ιστοσελίδα.

Η δεύτερη λειτουργία υλοποιήθηκε με *http* σύνδεση και μεταφορά των δεδομένων του αρχείου που βρίσκεται στον εξυπηρέτη του διαδικτύου. Τα δεδομένα εμφανίζονται στην οθόνη του κινητού τηλεφώνου. Επίσης, χρησιμοποιήθηκε η μέθοδος για σύνδεση με *https*, αν το υποστηρίζει ο εξυπηρέτης.

6.4.2 Διαμόρφωση της Υπηρεσίας από την Εφαρμογή

Στο μενού επιλογών της εφαρμογής προσθέσαμε δύο επιλογές που επιτρέπουν την διαμόρφωση της υπηρεσίας ενδιάμεσου λογισμικού απ' ευθείας από τον χρήστη. Η μία επιλογή (*Context Management*, Σχήμα 6.2) αφορά την διαμόρφωση των παραμέτρων του *context*. Συγκεκριμένα, ενεργοποιείται το GUI που παρέχει η υπηρεσία, μέσω του οποίου ο

χρήστης μπορεί να δει τις παραμέτρους του *context*, να προσθέσει, να αφαιρέσει και να αλλάξει τις παραμέτρους, να αποθηκεύσει το *context* που υπάρχει στην μνήμη της συσκευής και να σβήσει το αρχείο με το *context*. Στο Σχήμα 6.4 (α) φαίνεται το *GUI* διαμόρφωσης των παραμέτρων του *context*.



Σχήμα 6.4: (α) Το GUI για την διαμόρφωση του context. (β) Το GUI για την διαμόρφωση των κανόνων.

Η άλλη επιλογή που έχει ο χρήστης είναι η διαμόρφωση των κανόνων, που έχουν οριστεί, για την προσαρμογή της εφαρμογής με βάση τις αλλαγές στο *context*. Αυτή η επιλογή (*Rules Management*, Σχήμα 6.2) ενεργοποιεί ένα άλλο *GUI*, το οποίο περιέχει αντίστοιχες επιλογές, με το παραπάνω *GUI*, για την διαχείριση των κανόνων. Στο Σχήμα 6.4 (β) φαίνεται το *GUI* για τον έλεγχο των κανόνων.

6.4.3 Context – Aware Σχεδίαση της Εφαρμογής

Αφού υλοποιήσαμε την εφαρμογή, το επόμενο βήμα ήταν να την συνδέσουμε με την υπηρεσία ενδιάμεσου λογισμικού και να της ενσωματώσουμε *context – aware* συμπεριφορά.

Αρχικά, καθορίστηκε η επιθυμητή συμπεριφορά της εφαρμογής. Τα χαρακτηριστικά της εφαρμογής που μεταβάλλονται με βάση τις αλλαγές στο *context* που λαμβάνεται από τους προμηθευτές είναι: (α) η ένταση του ήχου, (β) το μέγεθος της γραμματοσειράς και (γ) η φωτεινότητα της οθόνης. Οι παράμετροι του *context* που καθορίζουν τις τιμές των παραπάνω χαρακτηριστικών είναι: (α) η ταχύτητα του χρήστη, (β) η ένταση του θορύβου στο περιβάλλον και (γ) η ένταση της φωτεινότητας στο περιβάλλον. Στον Πίνακα 6.1 φαίνονται οι παράμετροι που συνθέτουν το context για την συγκεκριμένη εφαρμογή και το πεδίο τιμών τους.

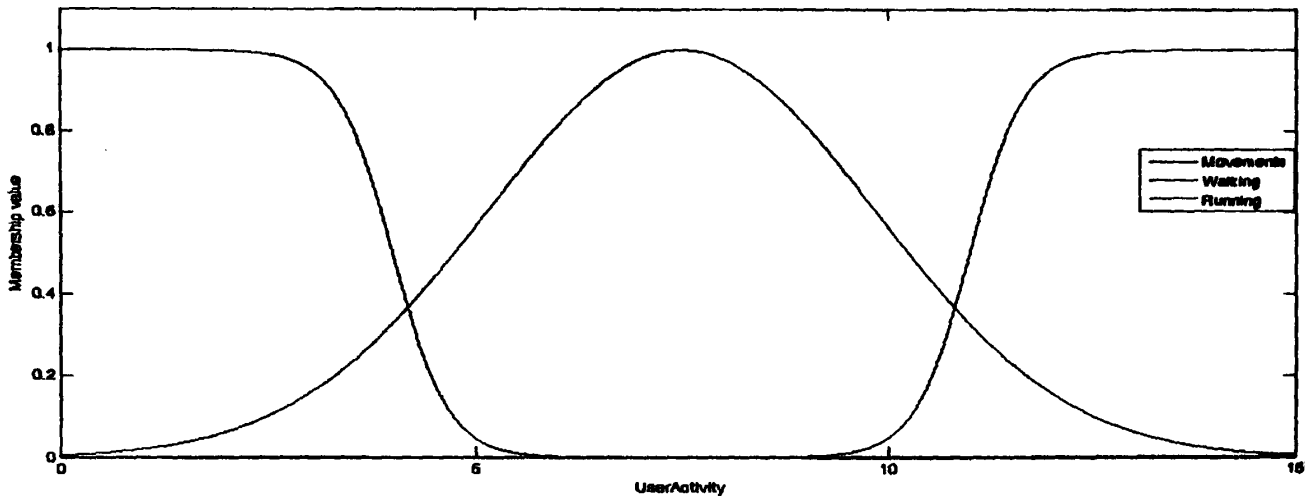
Πίνακας 6.1: Παράμετροι του context.

Παράμετρος	Πεδίο Τιμών
UserActivity	*Float*, $[0-15]Km/h$
LoudnessofEnvironment	*Float*, $[0-100]dB$
EnvironmentIlluminationLevel	*Float*, $[0-1500]lx$
VolumeofOperatingTunes	*Float*, $[0-5]$
FontSizeLevel	*Float*, $[0-5]$
DisplayIlluminationLevel	*Float*, $[0-5]$

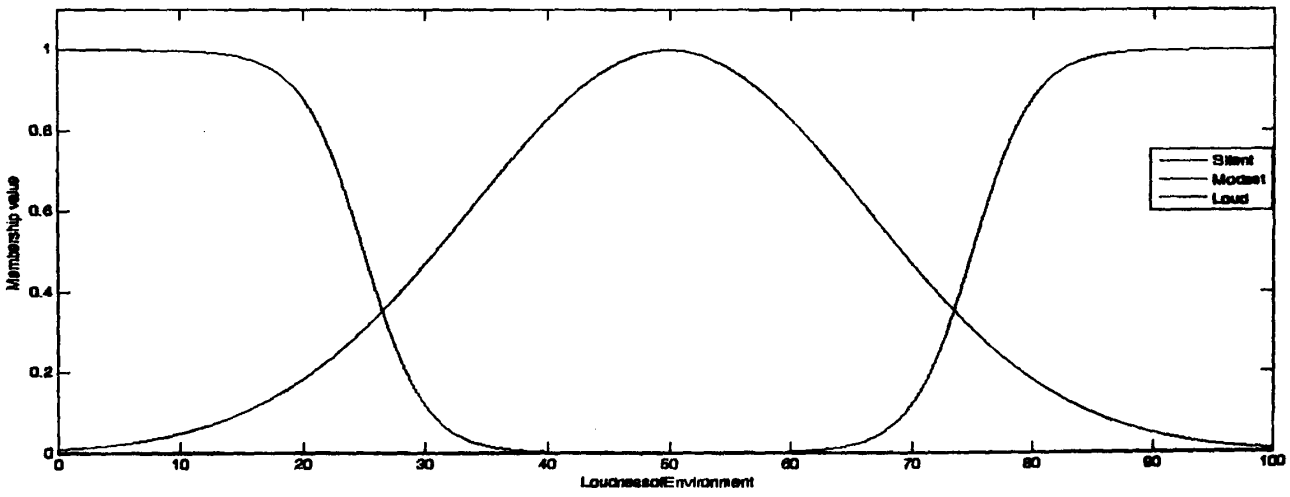
Η ένταση του ήχου για λειτουργίες της εφαρμογής, όπως οι εισερχόμενες κλήσεις και τα γραπτά μηνύματα, προσδιορίζεται με βάση την ταχύτητα του χρήστη και την ένταση του θορύβου στο περιβάλλον που κινείται ο χρήστης. Η ταχύτητα του χρήστη περιγράφεται μέσω της ασαφούς μεταβλητής *UserActivity*, ενώ η ένταση του θορύβου μέσω της ασαφούς μεταβλητής *LoudnessofEnvironment*. Για την ασαφή μεταβλητή *UserActivity* ορίστηκαν τα εξής ασαφή σύνολα: *Movements*, *Walking* και *Running*. Για την ασαφή μεταβλητή *LoudnessofEnvironment* ορίστηκαν τα ασαφή σύνολα: *Silent*, *Modest* και *Loud*. Στα Σχήματα 6.5 και 6.6 φαίνονται οι συναρτήσεις συμμετοχής που περιγράφουν τα ασαφή σύνολα για τις δύο αυτές παραμέτρους του *context*.

Η ένταση του ήχου της κινητής συσκευής προσδιορίζεται από την ασαφή μεταβλητή *VolumeofOperatingTunes*. Όπως φαίνεται στον Πίνακα 6.1, η μεταβλητή αυτή λαμβάνει τιμές

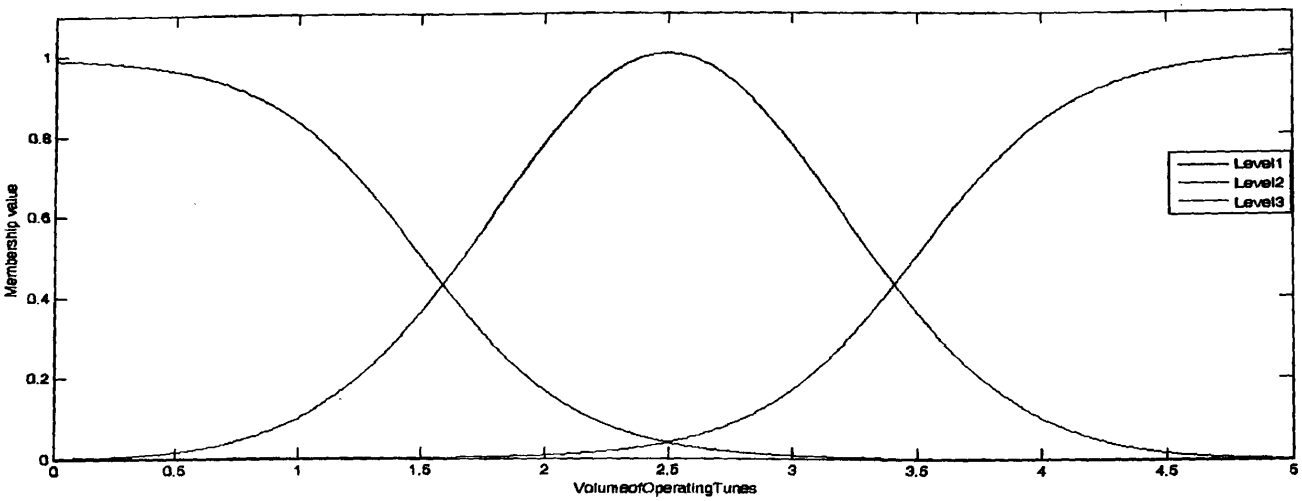
στο πεδίο των πραγματικών αριθμών στο διάστημα $[0-5]$. Ο καθορισμός του πεδίου τιμών έγινε ενδεικτικά, εφόσον η ένταση του ήχου εξαρτάται από τα χαρακτηριστικά της εκάστοτε κινητής συσκευής που εκτελείται η εφαρμογή. Η ασαφή μεταβλητή *VolumeofOperatingTunes* περιγράφεται από τα ασαφή σύνολα: *Level1*, *Level2*, *Level3* (Σχήμα 6.7). Ο Πίνακας 6.2 περιέχει τους ασαφείς κανόνες με βάση τους οποίους μεταβάλλεται η ένταση του ήχου στην κινητή συσκευή. Η σύνταξη των κανόνων γίνεται με βάση το μοντέλο ασαφούς λογικής *Mamdani*.



Σχήμα 6.5: Συναρτήσεις συμμετοχής της ασαφούς μεταβλητής UserActivity.



Σχήμα 6.6: Συναρτήσεις συμμετοχής της ασαφούς μεταβλητής LoudnessofEnvironment.



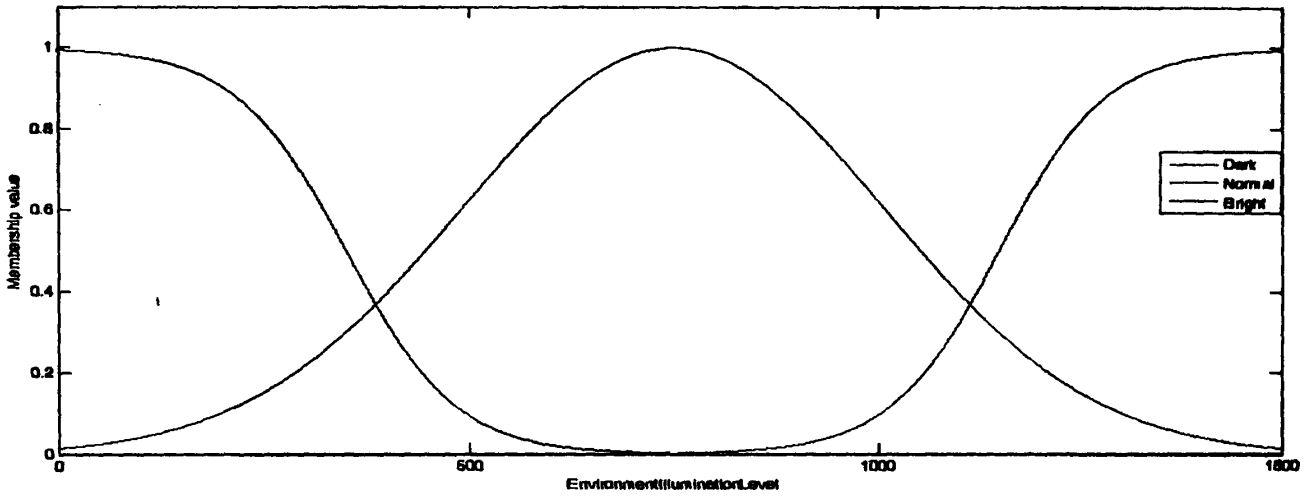
Σχήμα 6.7: Συναρτήσεις συμμετοχής της ασαφούς μεταβλητής VolumeofOperatingTunes.

Πίνακας 6.2: Κανόνες context για την ασαφή μεταβλητή VolumeofOperatingTunes.

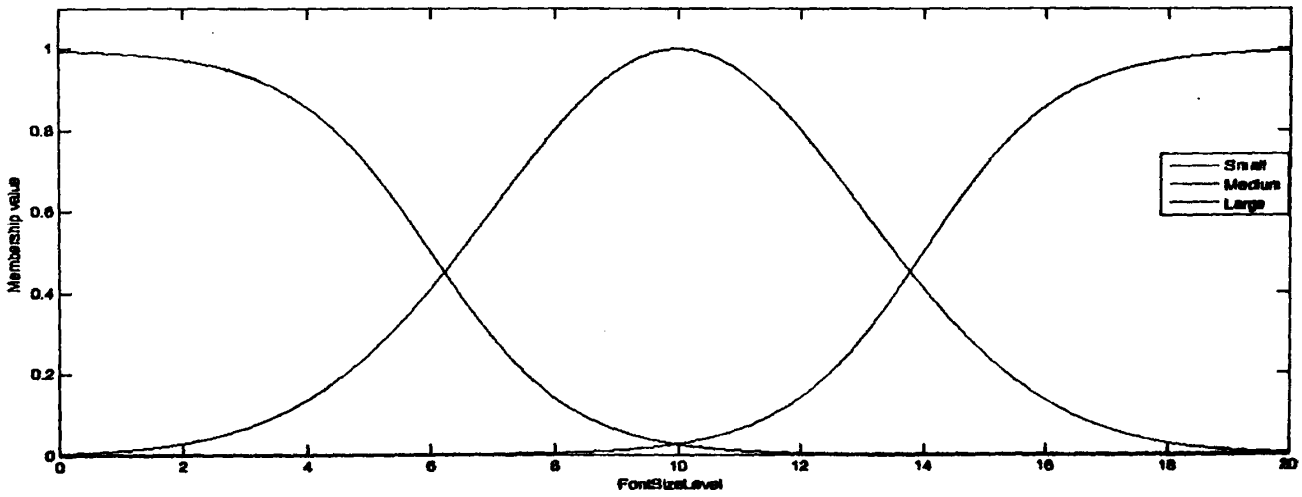
(UserActivity=Movements) && (LoudnessofEnvironment=Silent) => (VolumeofOperatingTunes=Level1)
(UserActivity=Movements) && (LoudnessofEnvironment=Modest) => (VolumeofOperatingTunes=Level2)
(UserActivity=Movements) && (LoudnessofEnvironment=Loud) => (VolumeofOperatingTunes=Level3)
(UserActivity=Walking) && (LoudnessofEnvironment=Silent) => (VolumeofOperatingTunes=Level2)
(UserActivity=Walking) && (LoudnessofEnvironment=Modest) => (VolumeofOperatingTunes=Level2)
(UserActivity=Walking) && (LoudnessofEnvironment=Loud) => (VolumeofOperatingTunes=Level3)
(UserActivity=Running) && (LoudnessofEnvironment=Silent) => (VolumeofOperatingTunes=Level3)
(UserActivity=Running) && (LoudnessofEnvironment=Modest) => (VolumeofOperatingTunes=Level3)
(UserActivity=Running) && (LoudnessofEnvironment=Loud) => (VolumeofOperatingTunes=Level3)

Το μέγεθος της γραμματοσειράς και η φωτεινότητα της οθόνης εξαρτώνται από τις τιμές της ταχύτητας του χρήστη και της έντασης της φωτεινότητας στο περιβάλλον του χρήστη. Η ένταση της φωτεινότητας στο περιβάλλον περιγράφεται με την ασαφή μεταβλητή *EnvironmentIlluminationLevel* και το μέγεθος της γραμματοσειράς και η φωτεινότητα της

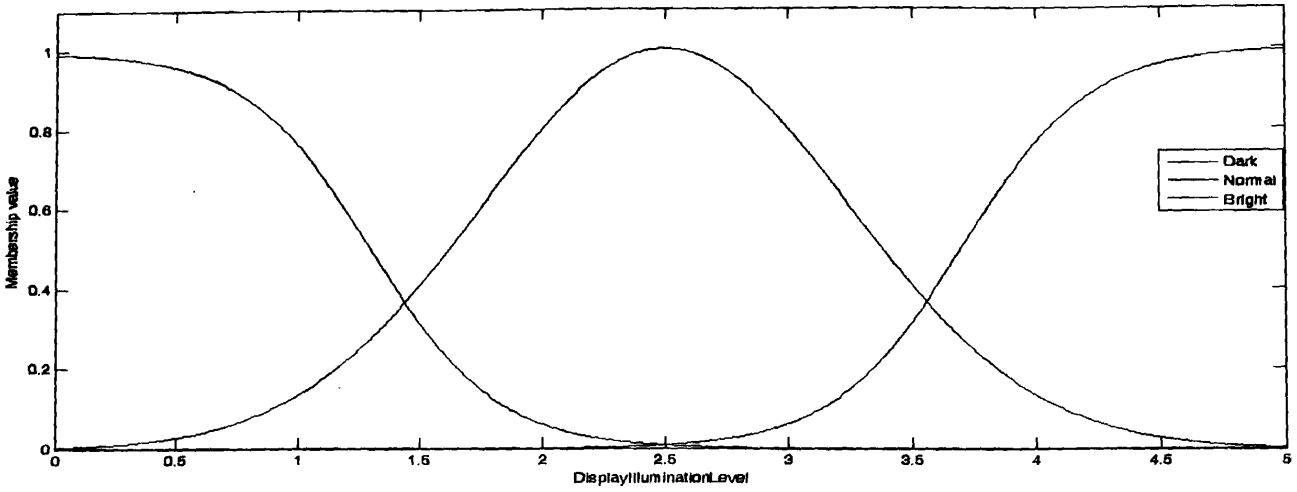
οθόνης με τις ασαφείς μεταβλητές *FontSizeLevel* και *DisplayIlluminationLevel*, αντίστοιχα. Ο ορισμός του πεδίου τιμών για τις ασαφείς μεταβλητές *FontSizeLevel* και *DisplayIlluminationLevel* (Πίνακας 6.1) έγινε με τον ίδιο τρόπο, όπως και για την μεταβλητή *VolumeofOperatingTunes*. Τα ασαφή σύνολα που περιγράφουν την μεταβλητή *EnvironmentIlluminationLevel* είναι τα εξής: *Dark*, *Normal*, *Bright*. Τα ίδια ασαφή σύνολα, ονομαστικά, περιγράφουν και την μεταβλητή *DisplayIlluminationLevel*, αλλά ορίζονται με διαφορετικές συναρτήσεις συμμετοχής. Τέλος, για την ασαφή μεταβλητή *FontSizeLevel* ορίστηκαν τα ασαφή σύνολα: *Small*, *Medium*, *Large*.



Σχήμα 6.8: Συναρτήσεις συμμετοχής της ασαφούς μεταβλητής EnvironmentIlluminationLevel.



Σχήμα 6.9: Συναρτήσεις συμμετοχής της ασαφούς μεταβλητής FontSizeLevel.



Σχήμα 6.10: Συναρτήσεις συμμετοχής της ασαφούς μεταβλητής DisplayIlluminationLevel.

Πίνακας 6.3: Κανόνες context για την ασαφή μεταβλητή FontSizeLevel.

(UserActivity=Movements) && (EnvironmentIlluminationLevel=Dark) => (FontSizeLevel=Medium)
(UserActivity=Movements) && (EnvironmentIlluminationLevel=Normal) => (FontSizeLevel=Small)
(UserActivity=Movements) && (EnvironmentIlluminationLevel=Bright) => (FontSizeLevel=Small)
(UserActivity=Walking) && (EnvironmentIlluminationLevel=Dark) => (FontSizeLevel=Large)
(UserActivity=Walking) && (EnvironmentIlluminationLevel=Normal) => (FontSizeLevel=Medium)
(UserActivity=Walking) && (EnvironmentIlluminationLevel=Bright) => (FontSizeLevel=Medium)
(UserActivity=Running) && (EnvironmentIlluminationLevel=Dark) => (FontSizeLevel=Large)
(UserActivity=Running) && (EnvironmentIlluminationLevel=Normal) => (FontSizeLevel=Large)
(UserActivity=Running) && (EnvironmentIlluminationLevel=Bright) => (FontSizeLevel=Large)

Στα Σχήματα 6.8 – 6.10 φαίνονται τα ασαφή σύνολα και οι συναρτήσεις συμμετοχής που ορίστηκαν για τις παραμέτρους *EnvironmentIlluminationLevel*, *FontSizeLevel* και *DisplayIlluminationLevel* του *context*. Οι Πίνακες 6.3 και 6.4 περιέχουν τους ασαφείς

κανόνες που προσδιορίζουν την επιθυμητή λειτουργία της εφαρμογής με βάση αυτές τις παραμέτρους του context. Συγκεκριμένα, οι κανόνες στον Πίνακα 6.3 ελέγχουν την μεταβλητή *FontSizeLevel*, ενώ οι κανόνες στον Πίνακα 6.4 αφορούν την μεταβλητή *DisplayIlluminationLevel*. Οι κανόνες και στους δύο πίνακες ακολουθούν το συντακτικό του μοντέλου ασαφούς λογικής *Mamdani*.

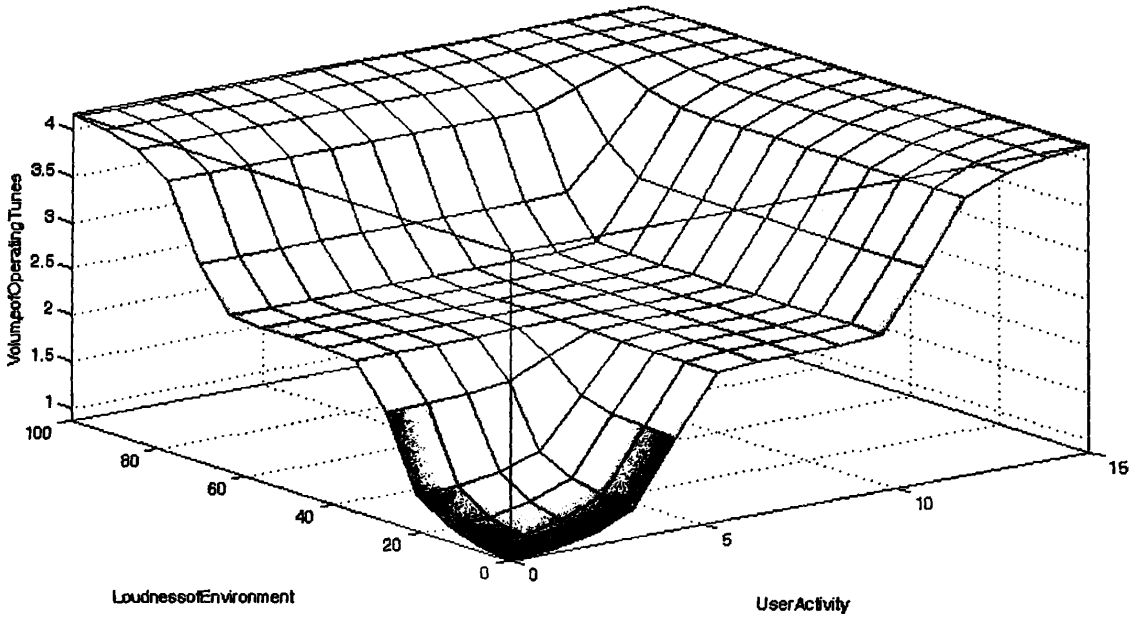
Πίνακας 6.4: Κανόνες context για την ασαφή μεταβλητή DisplayIlluminationLevel.

(UserActivity=Movements) && (EnvironmentIlluminationLevel=Dark) => (DisplayIlluminationLevel=Bright)
(UserActivity=Movements) && (EnvironmentIlluminationLevel=Normal) => (DisplayIlluminationLevel=Normal)
(UserActivity=Movements) && (EnvironmentIlluminationLevel=Bright) => (DisplayIlluminationLevel=Dark)
(UserActivity=Walking) && (EnvironmentIlluminationLevel=Dark) => (DisplayIlluminationLevel=Bright)
(UserActivity=Walking) && (EnvironmentIlluminationLevel=Normal) => (DisplayIlluminationLevel=Normal)
(UserActivity=Walking) && (EnvironmentIlluminationLevel=Bright) => (DisplayIlluminationLevel=Dark)
(UserActivity=Running) && (EnvironmentIlluminationLevel=Dark) => (DisplayIlluminationLevel=Bright)
(UserActivity=Running) && (EnvironmentIlluminationLevel=Normal) => (DisplayIlluminationLevel=Bright)
(UserActivity=Running) && (EnvironmentIlluminationLevel=Bright) => (DisplayIlluminationLevel=Normal)

Η αποτίμηση των κανόνων, για κάθε νέο γεγονός που λαμβάνει η υπηρεσία ενδιάμεσου λογισμικού, γίνεται με βάση το μοντέλο ασαφούς λογικής *Mamdani*. Οι τελεστές ένωσης και τομής, που χρησιμοποιούνται, προσδιορίζονται από τις Εξισώσεις (4.11) και (4.15), αντίστοιχα. Από την Εξίσωση (4.11) προσδιορίζεται και ο τελεστής επαγωγής. Τέλος, για την διαδικασία αποασαφοποίησης χρησιμοποιείται η μέθοδος *centroid*. Έτσι, για κάθε νέα τιμή που λαμβάνει η υπηρεσία από τους προμηθευτές για τις παραμέτρους *UserActivity*, *LoudnessofEnvironment* και *EnvironmentIlluminationLevel* εκτελείται η διαδικασία ασαφούς συλλογιστικής *Mamdani*. Ανάλογα με τους κανόνες που ενεργοποιούνται, αποφασίζονται οι

νέες τιμές για τις παραμέτρους *VolumeofOperatingTunes*, *FontSizeLevel* και *DisplayIlluminationLevel*.

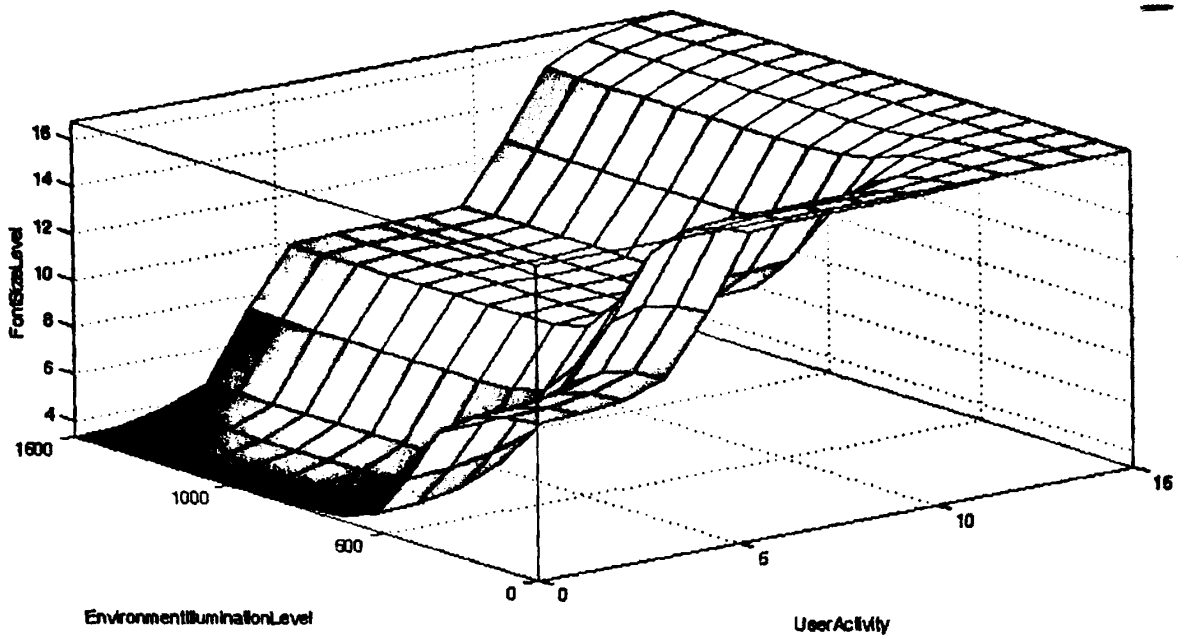
Στο Σχήμα 6.11 φαίνεται η έξοδος των κανόνων, που δίνονται στον Πίνακα 6.2, για όλους τους δυνατούς συνδυασμούς των τιμών των παραμέτρων *UserActivity* και *LoudnessofEnvironment*. Οι κανόνες αυτοί ελέγχουν την τιμή της παραμέτρου *VolumeofOperatingTunes*. Είναι προφανές, ότι η ένταση του ήχου στην κινητή συσκευή είναι μέγιστη, όταν η ταχύτητα του χρήστη ή η ένταση του θορύβου στο περιβάλλον είναι μέγιστη. Επιπλέον, η ένταση του ήχου λαμβάνει την ελάχιστη τιμή της, όταν η ταχύτητα του χρήστη και η ένταση του θορύβου είναι ελάχιστη.



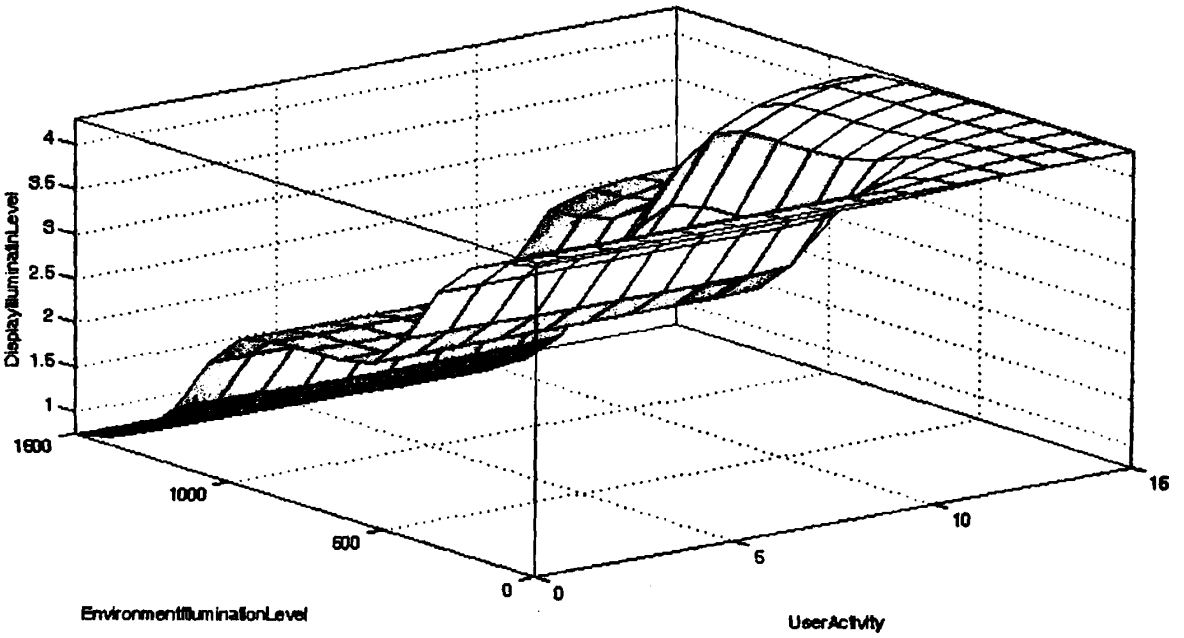
Σχήμα 6.11: Η έξοδος για την ασαφή μεταβλητή VolumeofOperatingTunes.

Στα Σχήματα 6.12 και 6.13 φαίνεται η έξοδος των κανόνων που δίνονται στους Πίνακες 6.3 και 6.4. Οι κανόνες αυτοί καθορίζουν τις τιμές των παραμέτρων *FontSizeLevel* και *DisplayIlluminationLevel*, αντίστοιχα. Όταν η ταχύτητα του χρήστη είναι μέγιστη, το μέγεθος της γραμματοσειράς είναι μεγάλο. Ενώ, όταν η ταχύτητα του χρήστη είναι ελάχιστη και η ένταση της φωτεινότητας στο περιβάλλον είναι μέγιστη, τότε το μέγεθος της γραμματοσειράς είναι μικρό. Όσον αφορά την ένταση της φωτεινότητας στην κινητή συσκευή, αυτή λαμβάνει την μέγιστη τιμή της όταν η ένταση της φωτεινότητας στο περιβάλλον είναι μικρή και όταν η ταχύτητα του χρήστη είναι μέγιστη. Η ένταση της φωτεινότητας στην οθόνη είναι πάντα

ικρή, όταν η ένταση της φωτεινότητας στο περιβάλλον είναι υψηλή, εκτός από την τερίπτωση που η ταχύτητα του χρήστη είναι μέγιστη.



Σχήμα 6.12: Η έξοδος για την ασαφή μεταβλητή FontSizeLevel.



Σχήμα 6.13: Η έξοδος για την ασαφή μεταβλητή DisplayIlluminationLevel.

τον έλεγχο της ορθής λειτουργίας της υπηρεσίας ενδιάμεσου λογισμικού εκτελέστηκαν ρορα σενάρια λειτουργίας της εφαρμογής. Σε κάθε σενάριο, μεταβάλλονται με κεκριμένο τρόπο οι τιμές των παραμέτρων *UserActivity*, *LoudnessofEnvironment* και *ironmentIlluminationLevel* και ελέγχεται η έξοδος της υπηρεσίας για τις παραμέτρους

VolumeofOperatingTunes, *FontSizeLevel* και *DisplayIlluminationLevel*. Παρατηρήθηκε ότι, η έξοδος εξαρτάται από τις συναρτήσεις συμμετοχής που περιγράφουν τα ασαφή σύνολα των παραμέτρων του *context*. Οι συναρτήσεις συμμετοχής που χρησιμοποιούνται στην συγκεκριμένη εφαρμογή ορίστηκαν εμπειρικά.

6.4.4 Προγραμματιστική Διασύνδεση

Αφού καθορίστηκε το *context* και οι κανόνες που περιγράφουν την προσαρμογή της εφαρμογής, υλοποιήθηκε η διασύνδεση της εφαρμογής με την υπηρεσία ενδιάμεσου λογισμικού. Η διασύνδεση περιλαμβάνει τις ενέργειες που περιγράφηκαν στην παράγραφο 6.3.1. Στην συνέχεια, περιγράφεται η υλοποίηση της διεπαφής *ContextManagement* από την εφαρμογή.

Η διεπαφή *ContextManagement* υλοποιήθηκε συμπληρώνοντας το σώμα των τεσσάρων μεθόδων που ορίζονται στην διεπαφή.

public String [] defineContext (String [ ] context)

Στην μέθοδο αυτή κατασκευάζεται ένας πίνακας συμβολοσειρών με τις παραμέτρους του *context*. Η περιγραφή κάθε παραμέτρου γίνεται με βάση το συντακτικό του μοντέλου ασαφούς λογικής *Mamdani*. Η υλοποίηση της μεθόδου φαίνεται στην συνέχεια:

```
public String [] defineContext(String [] context) {

            context = new String [] {

"UserActivity: [0,15], Movements (sigmf, a=-3, c=4), Walking (gaussmf,
s=2.342, c=7.5), Running (sigmf, a=3, c=11)",

"LoudnessofEnvironment: [0,100], Silent (sigmf, a=-0.4, c=25), Modest
(gaussmf, s=16.29, c=50), Loud (sigmf, a=0.4, c=75)",

"EnvironmentIlluminationLevel: [0,1500], Dark (sigmf, a=-0.015, c=350),
Normal (gaussmf, s=257, c=750), Bright (sigmf, a=0.015, c=1150)",

"VolumeofOperatingTunes: [0,5], Level1 (sigmf, a=-3.2, c=1.5), Level2
(gaussmf, s=0.7, c=2.5), Level3 (sigmf, a=3.2, c=3.5)",

"FontSizeLevel: [0,20], Small (sigmf, a=-.9, c=6), Medium(gaussmf, s=3,
c=10), Large (sigmf, a=0.9, c=14)",
```

```
"DisplayIlluminationLevel: [0,5], Dark (sigmf, a=-3.9, c=1.3), Normal
(gaussmf, s=0.7, c=2.5), Bright (sigmf, a=3.9, c=3.7)"

            );

            return context; }
```

public String [] defineContextRules (String [ ] rules)

Στην μέθοδο αυτή ο προγραμματιστής κατασκευάζει ένα πίνακα με τους κανόνες του *context*. Οι κανόνες ακολουθούν το συντακτικό του μοντέλου *Mamdani*. Η υλοποίηση της μεθόδου φαίνεται στην συνέχεια:

```
public String [] defineContextRules(String [] rules) {

            rules = new String [] {

            ..... // Δηλώσεις κανόνων
            .....
            };
            return rules; }
```

Οι κανόνες, που ορίστηκαν στην μέθοδο *defineContextRules*, είναι αυτοί που έχουν δοθεί στους Πίνακες 6.2 – 6.4. Όμως, οι κανόνες που έχουν το ίδιο τμήμα ενεργειών και διαφορετικό τμήμα συνθηκών συνδυάζονται σε ένα κανόνα. Το τμήμα συνθηκών αυτού του κανόνα αποτελείται από την διάζευξη των συνθηκών των επιμέρους κανόνων. Επίσης, οι κανόνες με το ίδιο τμήμα συνθηκών και διαφορετικό τμήμα ενεργειών συνδυάζονται σε ένα κανόνα. Το τμήμα ενεργειών του αποτελείται από την σύζευξη των ενεργειών των επιμέρους κανόνων.

public String [] defineFuzzyModel (String [] model)

Στην μέθοδο αυτή κατασκευάζεται ο πίνακας συμβολοσειρών που προσδιορίζει το μοντέλο ασαφούς λογικής που χρησιμοποιείται. Η υλοποίηση της μεθόδου γίνεται ως εξής:

```
public String [] defineFuzzyModel(String [] model) {

            model = new String [] {
                  "FuzzyModel = Mamdani",
                  "AndMethod = prod",
                  "OrMethod = probor",
                  "Implication = prod",
                  "Aggregation = probor",
                  "Defuzzification = centroid"
            };
            return model; }
```

public void actionResolve (String parameter, String value)

Στην μέθοδο αυτή γίνεται η αντιστοίχιση των ενεργειών σε μεθόδους της εφαρμογής. Για μία ενέργεια, < *parameter* > = < *value* >, εκτελείται η μέθοδος που σχετίζεται με την παράμετρο < *parameter* > του *context* με όρισμα την τιμή < *value* >. Η υλοποίηση της μεθόδου, για το σύνολο των κανόνων που περιγράφηκαν παραπάνω, δίνεται παρακάτω:

```
public void actionResolve(String parameter, String svalue) {

        float value = 0;

        if(IsFloat(svalue))
            value = Float.parseFloat(svalue);

        if("VolumeofOperatingTunes".equals(parameter))
            setVolumeofOperatingTunes(value);

        else if("FontSizeLevel".equals(parameter))
            setFontSizeLevel(value);

        else if("DisplayIlluminationLevel".equals(parameter))
            setDisplayIlluminationLevel(value);

        else
            display.setCurrent(Error_Message); }
```

Οι μέθοδοι *setVolumeofOperatingTunes*, *setFontSizeLevel* και *setDisplayIlluminationLevel* δεν υλοποιούν πραγματικά τις αντίστοιχες λειτουργίες. Τυπώνουν την τιμή της αντίστοιχης παραμέτρου του *context*

ΚΕΦΑΛΑΙΟ 7. ΑΞΙΟΛΟΓΗΣΗ ΥΠΗΡΕΣΙΑΣ ΕΝΔΙΑΜΕΣΟΥ ΛΟΓΙΣΜΙΚΟΥ

Μία βασική απαίτηση για το λογισμικό που αναπτύξαμε είναι να κάνει ορθολογική χρήση των υπολογιστικών πόρων που παρέχουν οι κινητές συσκευές. Για να κάνουμε αποτίμηση της απόδοσης της υπηρεσίας ενδιάμεσου λογισμικού, υλοποιήθηκε μία σειρά πειραμάτων. Στα πειράματα μετρήθηκε η χρήση των κρίσιμων υπολογιστικών πόρων καθώς και η χρονική απόκριση των βασικών λειτουργιών της υπηρεσίας. Στο κεφάλαιο αυτό, περιγράφουμε τα πειράματα που εκτελέστηκαν και παρουσιάζουμε τα αποτελέσματα που προέκυψαν από τις μετρήσεις.

7.1 Προσομοίωση της Κινητής Εφαρμογής

Ο σκοπός των πειραμάτων ήταν να μετρήσουμε τους πόρους που καταναλώνει η υπηρεσία ενδιάμεσου λογισμικού, κατά την λειτουργία της, και την χρονική της απόδοση. Τα πειράματα εκτελέστηκαν στον προσομοιωτή του *WTK*, ώστε να είναι αντιπροσωπευτικά της λειτουργίας μίας πραγματικής κινητής συσκευής. Οι παράμετροι που μετρήθηκαν στα πειράματα είναι οι ακόλουθες: (α) η συνολική μνήμη που απαιτεί η υπηρεσία και (β) η χρονική διάρκεια των βασικών λειτουργιών της υπηρεσίας.

Για κάθε μία από τις παραπάνω παραμέτρους, εκτελέστηκε ένα σύνολο πειραμάτων. Οι τιμές μετρήθηκαν κατά την διάρκεια των εκτελέσεων μεταβάλλοντας τις μεταβλητές που τις επηρεάζουν. Οι παράμετροι αυτές επιλέχθηκαν, διότι αποτελούν τους πιο κρίσιμους υπολογιστικούς πόρους για μία κινητή συσκευή και επιπλέον, τα εργαλεία για την μέτρηση της επίδοσης ήταν διαθέσιμα. Η κατανάλωση της μπαταρίας θα είχε ενδιαφέρον να μετρηθεί, αλλά στην πλατφόρμα *J2ME* δεν υπάρχει εργαλείο που να προσομοιώνει την μπαταρία κινητής συσκευής.

7.1.1 Περιβάλλον Προσομοίωσης

Για την εκτέλεση των πειραμάτων, υλοποιήσαμε ένα απλό *MIDlet* και το συνδέσαμε με την υπηρεσία ενδιάμεσου λογισμικού. Για την προσομοίωση της λειτουργίας της εφαρμογής, προσθέσαμε στο *MIDlet* ένα σύστημα παραγωγής δεδομένων. Το σύστημα αποτελείται από τρεις γεννήτριες: μία γεννήτρια παραμέτρων του *context*, μία γεννήτρια κανόνων και μία γεννήτρια γεγονότων. Οι γεννήτριες παρέχουν τις εισόδους που θα έπαιρνε η υπηρεσία, κατά την λειτουργία της, σε ένα *context – aware* σύστημα.

Γεννήτρια Context

Η γεννήτρια *context* παράγει τις παραμέτρους του *context* που χρησιμοποιούνται σε κάθε πείραμα. Η δημιουργία των παραμέτρων γίνεται με βάση το συντακτικό του μοντέλου του *context* που δόθηκε στο Κεφάλαιο 5. Για το σύνολο πειραμάτων που αφορούν το μοντέλο ασαφούς λογικής *Mamdani*, η γεννήτρια παράγει τις παραμέτρους του *context* σύμφωνα με το συντακτικό του μοντέλου *Mamdani* (Πίνακες 5.1, 5.5). Το συντακτικό του μοντέλου *Sugeno* (Πίνακες 5.3, 5.5) χρησιμοποιείται στα πειράματα που εκτελέστηκαν για αυτό το μοντέλο. Η γεννήτρια *context* εκτελείται πριν από την έναρξη της υπηρεσίας. Οι παράμετροι του *context* εισάγονται σε καθορισμένο αρχείο και ο *ContextManager* τις διαβάζει από εκεί.

Γεννήτρια Κανόνων

Η γεννήτρια κανόνων παράγει τυχαίους κανόνες. Οι κανόνες ακολουθούν το συντακτικό του μοντέλου του *context* που δόθηκε στο Κεφάλαιο 5, ανάλογα με το μοντέλο ασαφούς λογικής που χρησιμοποιείται (Πίνακες 5.2, 5.4, 5.6). Οι παράμετροι του *context*, που συμμετέχουν σε κάθε κανόνα, επιλέγονται από το σύνολο που παράγει η γεννήτρια *context*. Παρόμοια, η

γεννήτρια κανόνων εκτελείται πριν από την έναρξη της υπηρεσίας. Οι κανόνες εισάγονται σε καθορισμένο αρχείο και ο *ContextManager* τους διαβάζει από εκεί.

Γεννήτρια Γεγονότων

Η γεννήτρια γεγονότων παράγει τυχαία γεγονότα για την υπηρεσία. Η γεννήτρια τοποθετείται στο νήμα του *SensorsMonitor* και προσομοιώνει τον ρόλο των προμηθευτών. Εκτελεί ένα βρόχο στον οποίο παράγει καθορισμένο αριθμό γεγονότων, για τις παραμέτρους του *context* που υπάρχουν στον τμήμα συνθηκών των κανόνων.

7.1.2 Ρυθμίσεις Πειραμάτων

Η λειτουργία της υπηρεσίας ενδιάμεσου λογισμικού μπορεί να χωριστεί σε δύο φάσεις: την φάση της συντακτικής ανάλυσης και την φάση της επεξεργασίας των γεγονότων. Οι μετρήσεις των παραμέτρων έγιναν χωριστά για τις δύο φάσεις, αφού αναμένουμε η κατανάλωση των πόρων να είναι διαφορετική για κάθε φάση.

7.2 Μνήμη της Κινητής Εφαρμογής

Η μνήμη είναι ένας από τους βασικούς υπολογιστικούς πόρους για μία εφαρμογή, ενώ ταυτόχρονα η διαθέσιμη ποσότητά της στις κινητές συσκευές είναι περιορισμένη. Στα πειράματα που εκτελέστηκαν, μετρήθηκε η μνήμη που απαιτεί η υπηρεσία ενδιάμεσου λογισμικού για να εκτελεστεί σε μία κινητή συσκευή. Πριν από την περιγραφή των πειραμάτων, κρίνεται σκόπιμο να περιγραφεί η δομή της μνήμης που παρέχουν οι κινητές συσκευές και πως αυτή χρησιμοποιείται από μία κινητή εφαρμογή.

7.2.1 Μνήμη του MIDlet

Η μνήμη των κινητών εφαρμογών, που ακολουθούν το πρότυπο του *MIDlet*, αποτελείται από τρία συστατικά: την μνήμη του προγράμματος, τον σωρό και τον μόνιμο αποθηκευτικό χώρο.

- Μνήμη προγράμματος: Είναι η μνήμη της συσκευής, όπου θα εγκατασταθούν τα εκτελέσιμα αρχεία του *MIDlet*. Τα αρχεία συμπιέζονται σε ένα αρχείο *JAR* και περιλαμβάνουν ένα πολύ μικρότερο αρχείο *JAD* που περιγράφει τα περιεχόμενα του *JAR*.

- Σωρός: Είναι το τμήμα της μνήμης που χρησιμοποιεί το *MIDlet* κατά την εκτέλεσή του. Όλες οι μεταβλητές και τα αντικείμενα που δημιουργεί και χρησιμοποιεί η εφαρμογή αποθηκεύονται στον σωρό.
- Μόνιμος αποθηκευτικός χώρος: Είναι η μνήμη που χρησιμοποιείται στις κινητές συσκευές για μόνιμη αποθήκευση δεδομένων.

Η ποσότητα της διαθέσιμης μνήμης, για κάθε ένα από τους παραπάνω τύπους, εξαρτάται από τον κατασκευαστή της κάθε κινητής συσκευής. Ωστόσο, η προδιαγραφή για την ανάπτυξη κινητών εφαρμογών *JTWI* (*Java Technology for the Wireless Industry*), που χρησιμοποιεί το *MIDP 2.0*, καθορίζει τις ελάχιστες τιμές για κάθε τύπο μνήμης που θα πρέπει να έχουν οι συσκευές που υποστηρίζουν τα *MIDlets*. Οι τιμές αυτές είναι: 64 *KB* για την μνήμη προγράμματος, 256 *KB* για τον σωρό και 30 *KB* για τον μόνιμο αποθηκευτικό χώρο.

7.2.2 Ρυθμίσεις Προσομοίωσης

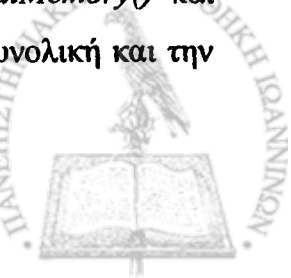
Το *WTK* παρέχει επιλογές για τον καθορισμό της μνήμης του σωρού και του μόνιμου αποθηκευτικού χώρου. Για την εκτέλεση των πειραμάτων, χρησιμοποιήσαμε 2 *MB* μνήμη στον σωρό και 2 *MB* μόνιμο αποθηκευτικό χώρο. Όπως θα δούμε στην συνέχεια, οι τιμές αυτές αποδείχτηκαν υπεραρκετές για την εκτέλεση των πειραμάτων.

7.3 Κατανάλωση Μνήμης

Σε αυτό το σύνολο πειραμάτων, μετρήθηκε η μνήμη που απαιτεί η υπηρεσία ενδιάμεσου λογισμικού για τις τρεις κατηγορίες μνήμης που αναφέρθηκαν παραπάνω. Η μνήμη του σωρού και του μόνιμου αποθηκευτικού χώρου μετρήθηκαν ως συνάρτηση των παραμέτρων που την επηρεάζουν. Επιπλέον, παρουσιάζεται το μέγεθος των εκτελέσιμων αρχείων που εγκαθίστανται στην συσκευή.

7.3.1 Μνήμη Σωρού

Στο πρώτο σύνολο πειραμάτων, μετρήθηκε η μνήμη που δεσμεύει η υπηρεσία, κατά την λειτουργία της, από τον σωρό. Η μνήμη μετρήθηκε με τις μεθόδους *totalMemory()* και *freeMemory()* της κλάσης *java.lang.Runtime*. Οι μέθοδοι επιστρέφουν την συνολική και την διαθέσιμη μνήμη στην *JVM*.



Οι παράμετροι που μεταβάλαμε σε αυτό το σύνολο πειραμάτων είναι:

- Ο αριθμός των κανόνων.
- Ο αριθμός των συνθηκών σε κάθε κανόνα.
- Ο αριθμός των ενεργειών σε κάθε κανόνα.

Αυτό που φυσιολογικά αναμένουμε είναι, η μνήμη να αυξάνεται γραμμικά καθώς αυξάνονται οι παραπάνω παράμετροι. Η αύξηση της δεσμευμένης μνήμης αναμένεται να υπάρχει, μόνο κατά την διάρκεια της συντακτικής ανάλυσης, όπου δημιουργείται το δέντρο των κανόνων. Στην συνέχεια, κατά την επεξεργασία των γεγονότων, οι καταχωρήσεις στην μνήμη είναι ελάχιστες και προσωρινές και αναμένουμε η μνήμη να παραμένει σχεδόν σταθερή.

Στην συνέχεια, περιγράφονται τα πειράματα που αφορούν την μνήμη. Σε κάθε πείραμα μετρήθηκε η μέγιστη τιμή της δεσμευμένης μνήμης. Η δεσμευμένη μνήμη παίρνει την μέγιστη τιμή της στο τέλος της συντακτικής ανάλυσης. Για να εξαλείψουμε την εξάρτηση από το σύνολο του *context* και των κανόνων, εκτελέσαμε κάθε πείραμα 100 φορές για 100 διαφορετικά σύνολα παραμέτρων του *context* και κανόνων. Στην συνέχεια, πήραμε τον μέσο όρο των τιμών των 100 εκτελέσεων.

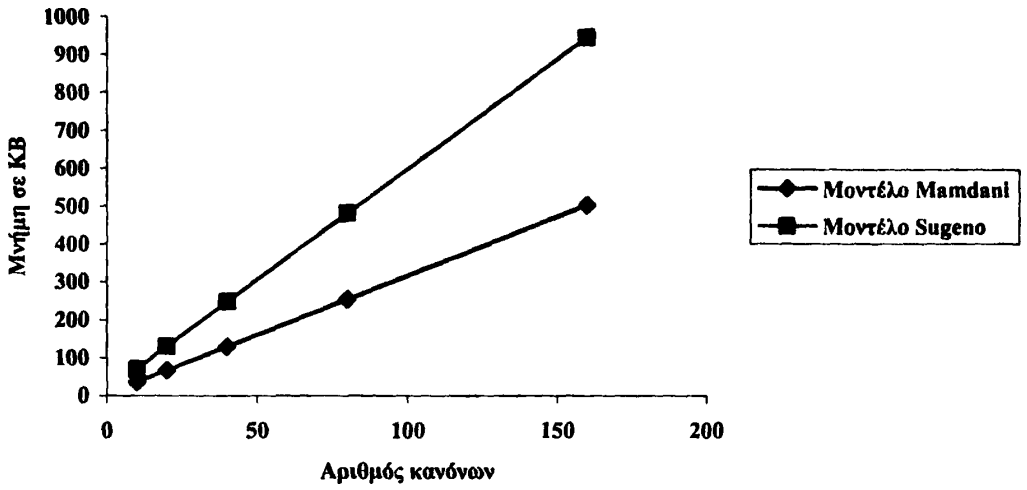
Τα πειράματα εκτελέστηκαν για τα μοντέλα ασαφούς λογικής *Mamdani* και *Sugeno*. Η χρήση της μνήμης στο μοντέλο *Tsukamoto* είναι ίδια με το μοντέλο *Mamdani*. Και στα δύο μοντέλα ασαφούς λογικής χρησιμοποιήθηκε ένα σύνολο με 20 παραμέτρους του *context*. Στο μοντέλο *Mamdani*, για κάθε παράμετρο του *context* ορίστηκαν 3 ασαφή σύνολα. Στο μοντέλο *Sugeno*, για κάθε παράμετρο του *context* εισόδου ορίστηκαν 3 ασαφή σύνολα, ενώ για κάθε παράμετρο του *context* εξόδου ορίστηκαν 3 συναρτήσεις.

Μεταβολή Αριθμού Κανόνων

Μετρήθηκε η μέγιστη μνήμη που δεσμεύει η υπηρεσία όταν αυξάνεται ο αριθμός των κανόνων. Κάθε κανόνας έχει σταθερό αριθμό συνθηκών και ενεργειών ίσο με 6 και 2, αντίστοιχα. Στο Σχήμα 7.1 φαίνεται το διάγραμμα της χρησιμοποιούμενης μνήμης σε συνάρτηση με τον αριθμό των κανόνων για τα δύο μοντέλα ασαφούς λογικής.

Για λογικό αριθμό κανόνων, η απαιτούμενη μνήμη είναι μικρή και εντός των ορίων που τίθενται από το πρότυπο *JTWI*. Το μοντέλο *Sugeno* δεσμεύει περισσότερη μνήμη, σε σχέση

με το μοντέλο *Mamdani*, επειδή για κάθε ενέργεια ενός κανόνα αποθηκεύεται περισσότερη πληροφορία. Η επιπλέον πληροφορία αφορά τα πεδία που αποθηκεύουν τις παραμέτρους του *context* από τις οποίες εξαρτάται η κάθε ενέργεια και τις τιμές αυτών των παραμέτρων.



Σχήμα 7.1: Μεταβολή της χρησιμοποιούμενης μνήμης όταν αυξάνεται ο αριθμός των κανόνων.

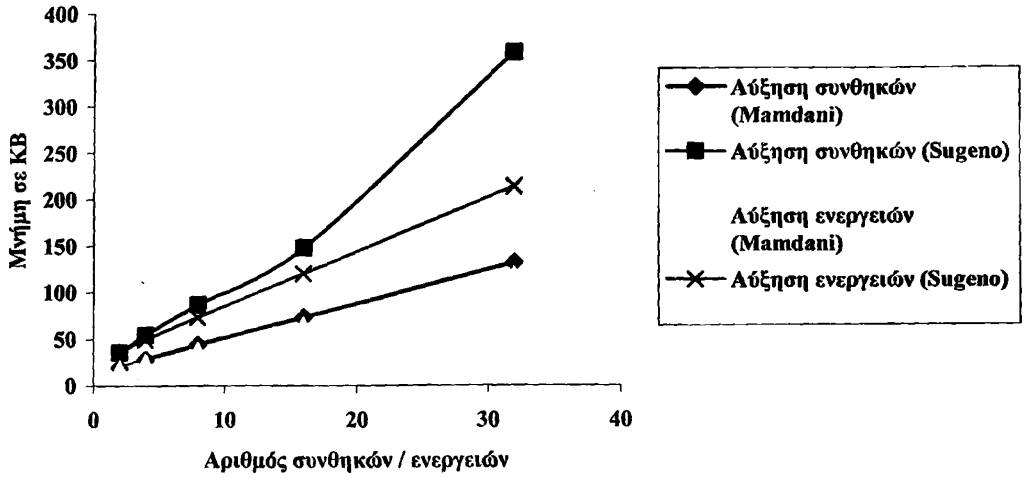
Μεταβολή Αριθμού Συνθηκών

Μετρήθηκε η μεταβολή της μνήμης όταν αυξάνεται ο αριθμός των συνθηκών σε κάθε κανόνα. Ο αριθμός των κανόνων ήταν 10 και ο αριθμός ενεργειών σε κάθε κανόνα ήταν 2.

Μεταβολή Αριθμού Ενεργειών

Μετρήθηκε η μεταβολή της μνήμης όταν μεταβάλλεται ο αριθμός των ενεργειών σε κάθε κανόνα. Ο αριθμός των κανόνων ήταν 10 και ο αριθμός των συνθηκών σε κάθε κανόνα 2.

Στο Σχήμα 7.2 φαίνεται το διάγραμμα της μεταβολής της μνήμης σε συνάρτηση με τον αριθμό των συνθηκών και των ενεργειών για τα μοντέλα *Mamdani* και *Sugeno*. Παρατηρούμε, ότι η χρησιμοποιούμενη μνήμη αυξάνεται περισσότερο, όσο αυξάνονται οι συνθήκες ενός κανόνα. Αυτό συμβαίνει διότι στους κόμβους των συνθηκών κρατάμε περισσότερη πληροφορία από αυτούς των ενεργειών. Τα βασικά επιπλέον πεδία είναι ο λογικός τελεστής (στις ενέργειες είναι πάντα ο &&) και η συμβολοσειρά των υποσυνθηκών που υπάρχει στις σύνθετες εκφράσεις.



Σχήμα 7.2: Μεταβολή της χρησιμοποιούμενης μνήμης όταν αυξάνεται ο αριθμός των συνθηκών / ενεργειών.

Στο μοντέλο *Sugeno*, η αύξηση των συνθηκών σε κάθε κανόνα έχει ως αποτέλεσμα οι ενέργειες, στο τμήμα ενεργειών κάθε κανόνα, να εξαρτώνται από περισσότερες παραμέτρους του *context*. Συγκεκριμένα, η αύξηση των συνθηκών σε κάθε κανόνα επιφέρει γραμμική αύξησή στο μέγεθος της πληροφορίας που αποθηκεύεται για κάθε ενέργεια στους κόμβους του δέντρου. Το γεγονός αυτό δικαιολογεί το ότι, η μνήμη αυξάνεται περισσότερο στο μοντέλο *Sugeno*, σε σχέση με το μοντέλο *Mamdani*. Επίσης, η αύξηση των ενεργειών έχει μεγαλύτερη επίδραση στο μοντέλο *Sugeno*, διότι για κάθε ενέργεια ενός κανόνα αποθηκεύεται περισσότερη πληροφορία.

7.3.2 Μνήμη Μόνιμου Αποθηκευτικού Χώρου

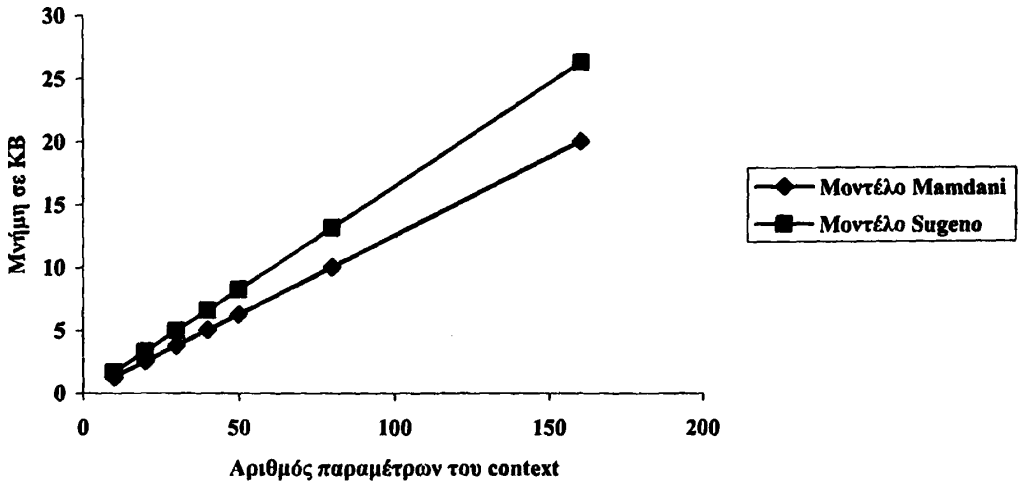
Μετρήθηκε η μνήμη που χρησιμοποιεί η υπηρεσία ενδιάμεσου λογισμικού στο μόνιμο αποθηκευτικό χώρο. Η μνήμη αυτή καταλαμβάνεται από τα αρχεία των παραμέτρων του *context* και των κανόνων που παίρνει η υπηρεσία σαν είσοδο. Τα αρχεία αυτά μπορεί να σβηστούν ή να μεταβληθεί το μέγεθός τους, κατά την διάρκεια λειτουργίας της υπηρεσίας.

Τα πειράματα εκτελέστηκαν για τα μοντέλα ασαφούς λογικής *Mamdani* και *Sugeno*. Τα περιεχόμενα των αρχείων του *context* και των κανόνων στο μοντέλο *Tsukamoto* είναι ίδια με του μοντέλου *Mamdani*. Στο μοντέλο *Mamdani* κάθε παράμετρος του *context* περιγράφεται μέσω 3 ασαφών συνόλων. Το ίδιο ισχύει και για κάθε παράμετρο του *context* εισόδου στο

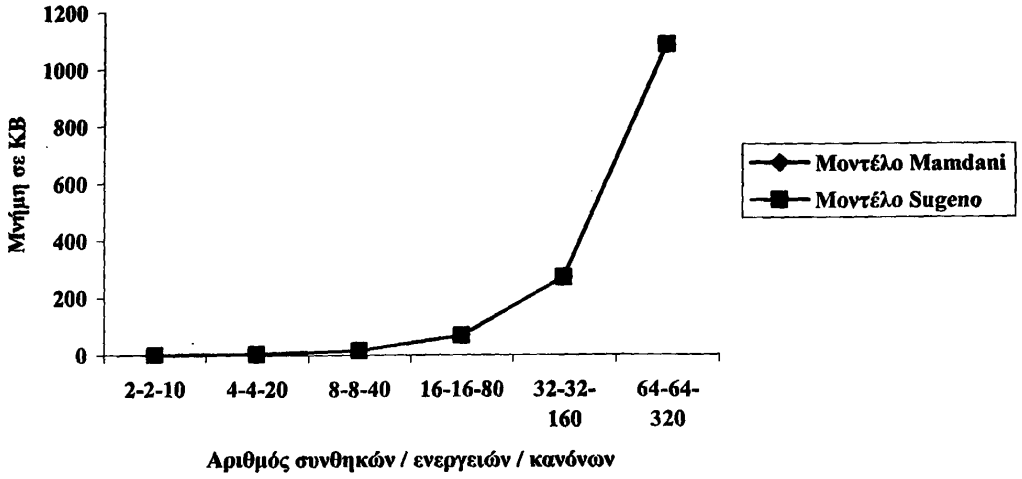
μοντέλο *Sugeno*. Στο μοντέλο *Sugeno*, κάθε παράμετρος του *context* εξόδου περιγράφεται μέσω 3 συναρτήσεων πρώτου βαθμού. Επίσης, θεωρήσαμε ότι, κάθε παράμετρος του *context* εξόδου εξαρτάται από 6 παραμέτρους του *context* εισόδου.

Στα πειράματα που εκτελέσαμε, μετρήσαμε το μέγεθος των αρχείων όταν αυτά δημιουργούνται. Οι παράμετροι που μεταβάλαμε ήταν τα περιεχόμενα των αρχείων. Στην περίπτωση του αρχείου του *context*, η παράμετρος που μεταβάλαμε ήταν το πλήθος των παραμέτρων του *context*. Όσον αφορά το αρχείο των κανόνων, οι παράμετροι που μεταβάλαμε ήταν ο αριθμός των κανόνων, των συνθηκών και των ενεργειών κάθε κανόνα. Μεταβάλαμε ταυτόχρονα τις παραμέτρους που σχετίζονται με το αρχείο των κανόνων, γιατί μεμονωμένες μεταβολές τους δεν προκαλούν ιδιαίτερη αύξηση στο μέγεθος του αρχείου.

Όπως φαίνεται στο Σχήμα 7.3, το μέγεθος του αρχείου του *context* αυξάνεται ανάλογα με τον αριθμό των παραμέτρων του *context*. Το μοντέλο *Sugeno* απαιτεί περισσότερο αποθηκευτικό χώρο για το αρχείο του *context*, εφόσον για κάθε παράμετρο του *context* εξόδου αποθηκεύονται και οι παράμετροι του *context* εισόδου από τις οποίες εξαρτάται. Επιπλέον, παρατηρούμε (Σχήμα 7.4) ότι, η μνήμη που καταλαμβάνει το αρχείο των κανόνων και στα δύο μοντέλα ασαφούς λογικής είναι μικρή, όσο ο αριθμός των κανόνων και των συστατικών τους παραμένουν σε λογικά επίπεδα.



Σχήμα 7.3: Μεταβολή του μεγέθους του αρχείου του context όταν αυξάνεται ο αριθμός των παραμέτρων του context.



Σχήμα 7.4: Μεταβολή του μεγέθους του αρχείου των κανόνων όταν αυξάνεται ο αριθμός των συνθηκών / ενεργειών / κανόνων.

7.3.3 Μνήμη Προγράμματος

Η μεταγλώττιση των αρχείων του *MIDlet* δημιουργεί το εκτελέσιμο *JAR* αρχείο του *MIDlet* και την περιγραφή του, το *JAD*. Τα αρχεία της εφαρμογής συμπιέζονται στο *JAR* μειώνοντας το μέγεθος των εκτελέσιμων κλάσεων. Το αρχείο *JAR* είναι αυτό που θα εγκατασταθεί στην κινητή συσκευή καταλαμβάνοντας χώρο από την μνήμη προγράμματος.

Το μέγεθος του *JAR* αρχείου μπορεί να ελαττωθεί περαιτέρω με χρήση ειδικών εργαλείων συμπίεσης. Τα εργαλεία αυτά ονομάζονται *obfuscators* και ελαχιστοποιούν το μέγεθος του εκτελέσιμου κώδικα. Οι *obfuscators* μετατρέπουν τις κλάσεις του *MIDlet* απομακρύνοντας αχρησιμοποίητες μεθόδους και κλάσεις και μετονομάζοντας μεθόδους και μεταβλητές. Τα εργαλεία αυτά διανέμονται ελεύθερα προς χρήση σε διάφορες εκδόσεις και εγκαθίστανται στον *J2ME* μεταγλωττιστή. Το *JAR* που παράγουν μπορεί να εκτελεστεί από την υλοποίηση του *CLDC* της συσκευής χωρίς διαφοροποιήσεις από το κανονικό.

Το μέγεθος των εκτελέσιμων κλάσεων της υπηρεσίας, που προκύπτουν από την μεταγλώττιση, είναι 67 *KB*. Το αρχείο *JAR* που δημιουργείται κατά την μεταγλώττιση, χωρίς χρήση *obfuscator*, έχει μέγεθος 29*KB*, ενώ το *JAD* 471 *Bytes*. Ο λόγος που επιτυγχάνεται τόσο μεγάλη συμπίεση είναι επειδή στα αρχεία της υπηρεσίας περιλαμβάνεται μόνο κώδικας

και όχι αρχεία ήχου ή εικόνες, όπως συμβαίνει στα περισσότερα *MIDlets*. Συνεπώς, η μνήμη προγράμματος που καταλαμβάνει η υπηρεσία είναι, ακόμα και χωρίς επιπλέον συμπίεση.

7.4 Χρονική Απόκριση

Μετρήθηκε η χρονική απόκριση της υπηρεσίας ενδιάμεσου λογισμικού στις δύο φάσεις λειτουργίας της. Στην πρώτη φάση, μετρήθηκε ο χρόνος της συντακτικής ανάλυσης των κανόνων. Στην δεύτερη φάση, μετρήθηκε ο μέσος χρόνος αποτίμησης ανά γεγονός. Δηλαδή πόσο χρόνο, κατά μέσο όρο, διαρκεί η επεξεργασία ενός γεγονότος από τον *RulesEvaluator*.

Για την μέτρηση του χρόνου, χρησιμοποιήθηκε η μέθοδος *System.currentTimeMillis()* της *Java*. Οι παράμετροι που μεταβάλαμε ήταν ο αριθμός των κανόνων και ο αριθμός των συνθηκών και των ενεργειών κάθε κανόνα. Φυσιολογικά, αναμένουμε ότι η χρονική απόκριση θα αυξάνεται όσο μεγαλώνουν οι παραπάνω παράμετροι, σε άλλες περιπτώσεις περισσότερο και σε άλλες λιγότερο.

Ο χρόνος αποτίμησης ενός γεγονότος εξαρτάται από το αν υπάρχουν κανόνες που επηρεάζονται από το γεγονός και πόσοι. Αν το γεγονός δεν βρεθεί σε κανένα κανόνα, δεν θα γίνει ενημέρωση των πεδίων σε κανένα κόμβο του δέντρου. Ενώ, κάθε κανόνας που περιέχει την παράμετρο του *context* του γεγονότος διατρέχεται, για να βρεθεί η συνθήκη ή οι συνθήκες και να γίνουν οι απαραίτητες ενημερώσεις. Επιπλέον ρόλο παίζει και ο αριθμός των κανόνων που ικανοποιούνται κάθε χρονική στιγμή, καθώς πρέπει να προσπελαστεί η λίστα των ενεργειών κάθε κανόνα.

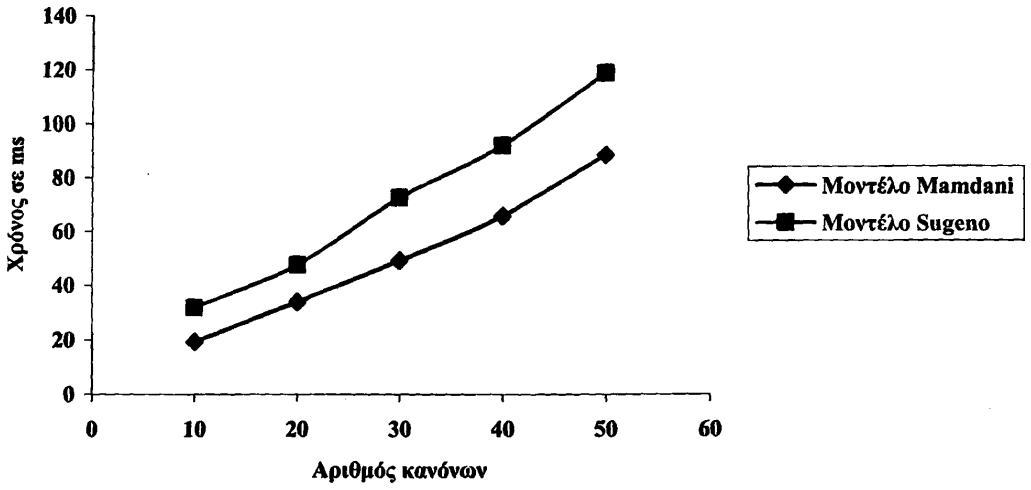
Κάθε πείραμα εκτελέστηκε 100 φορές για 100 διαφορετικά σύνολα παραμέτρων του *context* και κανόνων. Σε κάθε εκτέλεση, μετρήσαμε τον χρόνο της συντακτικής ανάλυσης και τον συνολικό χρόνο για 1000 γεγονότα και πήραμε τον μέσο χρόνο επεξεργασίας ανά γεγονός. Στην συνέχεια, πήραμε τον μέσο όρο των τιμών των 100 εκτελέσεων.

Τα πειράματα εκτελέστηκαν για τα μοντέλα ασαφούς λογικής *Mamdani* και *Sugeno*. Ο χρόνος της συντακτικής ανάλυσης στο μοντέλο *Tsukamoto* είναι ίσος με τον αντίστοιχο χρόνο στο μοντέλο *Mamdani*. Ενώ, ο χρόνος αποτίμησης ενός γεγονότος είναι μικρότερος ή ίσος από τον αντίστοιχο χρόνο στο μοντέλο *Sugeno*. Στο μοντέλο *Mamdani*, για κάθε

παράμετρο του *context* ορίστηκαν 3 ασαφή σύνολα. Στο μοντέλο *Sugeno*, για κάθε παράμετρο του *context* εισόδου ορίστηκαν 3 ασαφή σύνολα, ενώ για κάθε παράμετρο του *context* εξόδου ορίστηκαν 3 συναρτήσεις.

Μεταβολή Αριθμού Κανόνων

Μετρήσαμε την χρονική απόκριση της υπηρεσίας όταν αυξάνεται ο αριθμός των κανόνων. Ο αριθμός των συνθηκών σε κάθε κανόνα είναι 6 και των ενεργειών 2. Στο Σχήμα 7.5 φαίνεται η αύξηση του χρόνου της συντακτικής ανάλυσης, όταν αυξάνεται ο αριθμός των κανόνων. Στο Σχήμα 7.6 φαίνεται ο μέσος χρόνος επεξεργασίας ανά γεγονός, όταν αυξάνεται ο αριθμός των κανόνων.

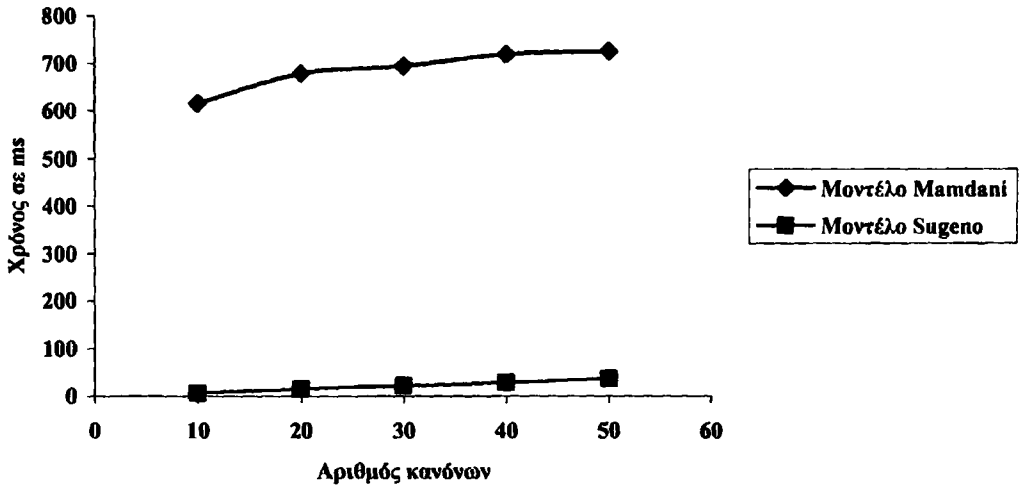


Σχήμα 7.5: Χρονική διάρκεια της συντακτικής ανάλυσης όταν αυξάνεται ο αριθμός των κανόνων.

Ο χρόνος που απαιτείται για την συντακτική ανάλυση αυξάνεται γραμμικά με τον αριθμό των κανόνων και στα δύο μοντέλα ασαφούς λογικής. Σε όλες τις περιπτώσεις, ο απαιτούμενος χρόνος δεν είναι σημαντικός, ειδικά λαμβάνοντας υπόψη ότι, η συντακτική ανάλυση εκτελείται μόνο μία φορά, κατά την αρχικοποίηση της υπηρεσίας. Ο χρόνος της συντακτικής ανάλυσης είναι μεγαλύτερος στο μοντέλο *Sugeno*, λόγω των επιπλέον πεδίων που αποθηκεύονται στους κόμβους των ενεργειών κάθε κανόνα.

Η αύξηση του αριθμού των κανόνων δεν επηρεάζει σχεδόν καθόλου τον χρόνο επεξεργασίας κάθε γεγονότος, όπως φαίνεται στο Σχήμα 7.6. Αυτό γιατί, ο χρόνος επεξεργασίας ενός

γεγονότος εξαρτάται από τον αριθμό των κανόνων που επηρεάζει αυτό το γεγονός και από το σύνολο των ενεργειών που τελικά θα εκτελεστούν. Για παράδειγμα, αν η παράμετρος του *context* ενός γεγονότος βρεθεί σε 4 κανόνες, όπου κάθε κανόνας έχει 2 ενέργειες, τότε το πολύ να εκτελεστούν 8 ενέργειες. Δηλαδή, το γεγονός ότι, μία παράμετρος του *context* υπάρχει στο τμήμα των συνθηκών ενός κανόνα, δεν συνεπάγεται ότι θα εκτελεστεί το τμήμα ενεργειών του. Απαραίτητη προϋπόθεση για να γίνει αυτό είναι να αλλάξει το βάρος του κανόνα με βάση το νέο γεγονός. Στα πειράματα που εκτελέσαμε, διατηρήσαμε σταθερό, κατά μέσο όρο, τον αριθμό των κανόνων που επηρεάζει ένα γεγονός και τον αριθμό των ενεργειών που εκτελούνται. Έτσι, θέσαμε τον αριθμό των κανόνων ίσο με 6 και τον αριθμό των ενεργειών ίσο με 3.



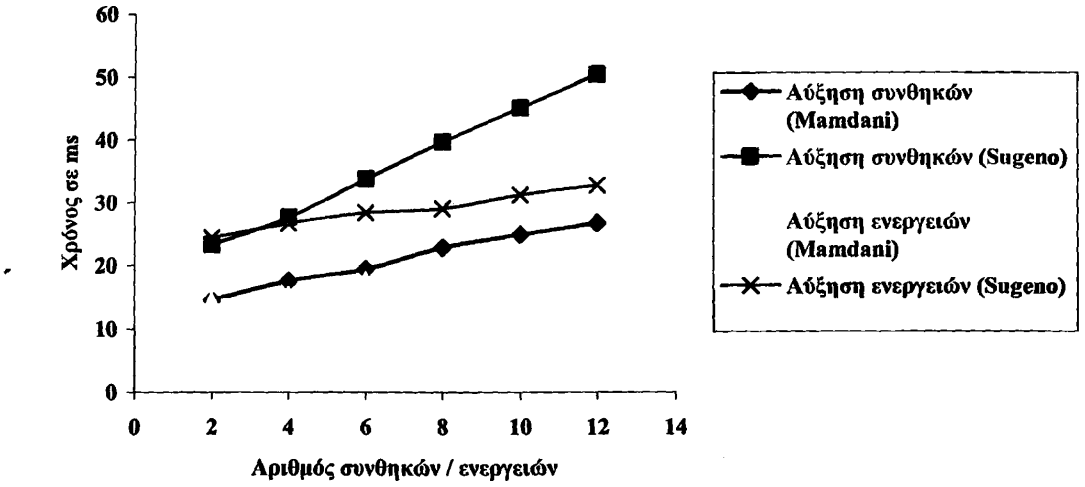
Σχήμα 7.6: Μέσος χρόνος επεξεργασίας ανά γεγονός όταν αυξάνεται ο αριθμός των κανόνων.

Ο χρόνος αποτίμησης ενός γεγονότος είναι πολύ μεγαλύτερος στο μοντέλο *Mamdani*. Ο επιπλέον χρόνος οφείλεται στα βήματα του υπολογισμού και της σύνθεσης των επιμέρους αποτελεσμάτων, όπως αυτά περιγράφηκαν στην παράγραφο 5.5.1. Επιπλέον χρόνο προσθέτει και το βήμα της αποασαφοποίησης των τελικών αποτελεσμάτων. Συγκεκριμένα, σημαντική χρονική επιβάρυνση προσθέτουν οι ακόλουθες διαδικασίες: (α) ο υπολογισμός της συνάρτησης επαγωγής μεταξύ του βάρους του κάθε κανόνα, του οποίου το βάρος έχει αλλάξει, και του κάθε ασαφούς συνόλου στο τμήμα ενεργειών του, (β) η ένωση των ασαφών συνόλων στο τμήμα ενεργειών των κανόνων, που αφορούν την ίδια ασαφή μεταβλητή, σε ένα ασαφές σύνολο που περιγράφει αυτή την μεταβλητή και (γ) η αποασαφοποίηση κάθε

ασαφούς συνόλου του προηγούμενο βήματος, για τον υπολογισμό της διακριτής τιμής της αντίστοιχης μεταβλητής.

Μεταβολή Αριθμού Συνθηκών και Ενεργειών

Σε αυτό το σύνολο πειραμάτων μεταβάλαμε τον αριθμό των συνθηκών και των ενεργειών στους κανόνες και μετρήσαμε την χρονική απόκριση της υπηρεσίας. Στην περίπτωση αύξησης του αριθμού των συνθηκών, ο αριθμός των ενεργειών διατηρείται σταθερός ίσος με 2. Παρόμοια, όταν αυξάνεται ο αριθμός των ενεργειών, ο αριθμός των συνθηκών διατηρείται σταθερός ίσος με 2. Το μέγεθος του συνόλου των κανόνων είναι ίσο με 10 και στις δύο περιπτώσεις.

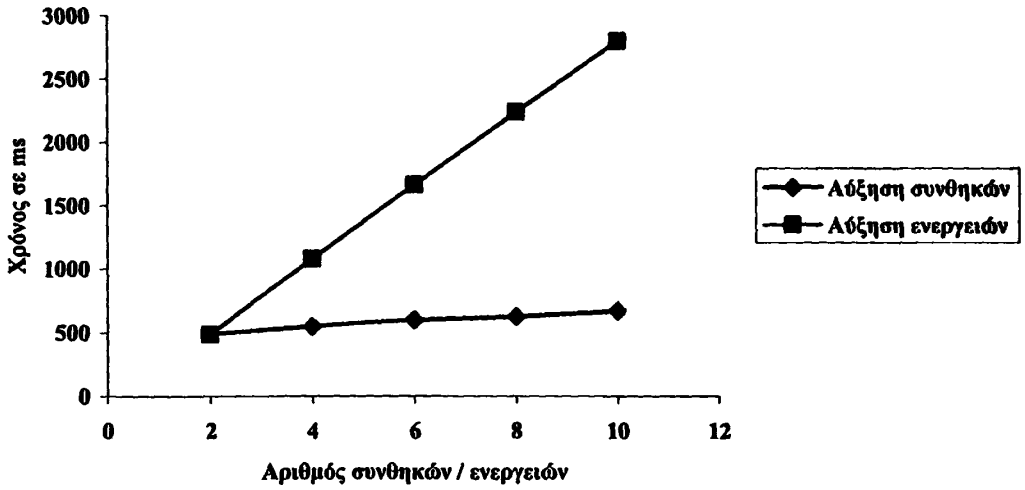


Σχήμα 7.7: Χρονική διάρκεια της συντακτικής ανάλυσης όταν αυξάνεται ο αριθμός των συνθηκών / ενεργειών.

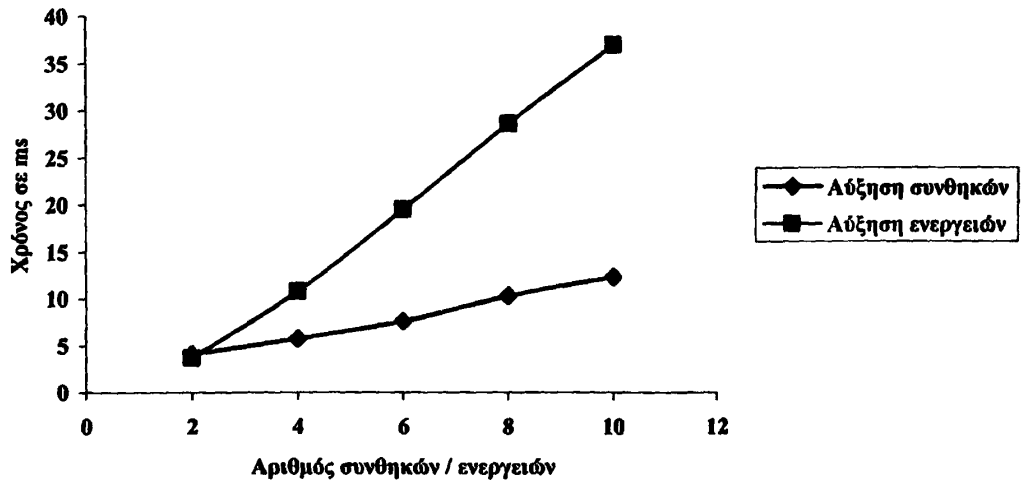
Όπως φαίνεται στο Σχήμα 7.7, ο χρόνος της συντακτικής ανάλυσης αυξάνεται περισσότερο, όταν αυξάνεται ο αριθμός των συνθηκών των κανόνων. Αυτό συμβαίνει, διότι το τμήμα των συνθηκών είναι πιο σύνθετο από αυτό των ενεργειών και προστίθενται περισσότεροι κόμβοι στο δέντρο. Επιπλέον, η αύξηση του αριθμού των συνθηκών επηρεάζει περισσότερο το μοντέλο *Sugeno*, διότι οδηγεί στην αύξηση των παραμέτρων του *context* από τις οποίες εξαρτάται κάθε παράμετρος του *context* στο τμήμα ενεργειών των κανόνων.

Στα Σχήματα 7.8 και 7.9 φαίνεται ο μέσος χρόνος επεξεργασίας ανά γεγονός για τα μοντέλα *Mamdani* και *Sugeno* αντίστοιχα, όταν αυξάνεται ο αριθμός των συνθηκών και των

ενεργειών. Παρατηρούμε ότι, η αύξηση των συνθηκών στους κανόνες επηρεάζει ελάχιστα την χρονική απόκριση της υπηρεσίας. Αντίθετα, η αύξηση των ενεργειών στους κανόνες αυξάνει γραμμικά τον μέσο χρόνο επεξεργασίας ανά γεγονός. Το γεγονός αυτό είναι απόλυτα λογικό, εφόσον το μεγαλύτερο ποσοστό του χρόνου αποτίμησης των κανόνων, για ένα γεγονός, καταναλώνεται στην επεξεργασία του τμήματος ενεργειών των κανόνων που έχει μεταβληθεί το βάρος τους. Η πολυπλοκότητα του τμήματος συνθηκών επηρεάζει μόνο το χρόνο υπολογισμού του βάρους των κανόνων.



Σχήμα 7.8: Μέσος χρόνος επεξεργασίας ανά γεγονός όταν αυξάνεται ο αριθμός των συνθηκών / ενεργειών στο μοντέλο Mamdani.



Σχήμα 7.9: Μέσος χρόνος επεξεργασίας ανά γεγονός όταν αυξάνεται ο αριθμός των συνθηκών / ενεργειών στο μοντέλο Sugeno.

Κατά την διάρκεια της συντακτικής ανάλυσης, για το μοντέλο *Sugeno* απαιτείται περισσότερος χρόνος, λόγω του ότι αποθηκεύει περισσότερη πληροφορία για το τμήμα ενεργειών των κανόνων. Αντίθετα, για το μοντέλο *Mamdani* απαιτείται περισσότερος χρόνος για την αποτίμηση του κάθε γεγονότος, επειδή εκτελεί πιο σύνθετους υπολογισμούς, όπως αναφέρθηκε και παραπάνω.

ΚΕΦΑΛΑΙΟ 8. ΣΥΜΠΕΡΑΣΜΑΤΑ – ΜΕΛΛΟΝΤΙΚΗ ΕΡΓΑΣΙΑ

8.1 Συμπεράσματα

Το θέμα της παρούσας εργασίας είναι η ανάπτυξη ενδιάμεσου λογισμικού για *context – aware* εφαρμογές σε κινητές συσκευές, το οποίο λαμβάνει υπόψη τα χαρακτηριστικά της πληροφορίας του *context*. Για τον λόγο αυτό, ενσωματώσαμε στην υπηρεσία ενδιάμεσου λογισμικού, που παρουσιάστηκε στην εργασία [33], ένα μοντέλο αναπαράστασης και διαχείρισης του *context* με βάση την ασαφή λογική. Η ασαφής λογική εξασφαλίζει την περιγραφή και χρήση του *context* με τρόπο ανάλογο της ανθρώπινης λογικής. Το στοιχείο αυτό είναι πολύ σημαντικό στην ανάπτυξη ενός *context – aware* συστήματος, εφόσον ο βασικός στόχος του είναι η παροχή ποιοτικότερων υπηρεσιών στους χρήστες. Επιπλέον, η χρήση ασαφών συνόλων, για την περιγραφή του *context*, συμβάλλει στην αντιμετώπιση προβλημάτων που προκύπτουν από την παροχή ανακριβών τιμών του *context* από τους προμηθευτές.

Η υπηρεσία ενδιάμεσου λογισμικού που αναπτύξαμε παρέχει υλοποιημένα τα τρία μοντέλα ασαφούς λογικής: *Tsukamoto, Mamdani* και *Sugeno*. Ανάλογα με το μοντέλο, διαφοροποιείται ο τρόπος περιγραφής του *context*, η σύνταξη των κανόνων και ο μηχανισμός εξαγωγής συμπερασμάτων, για την λειτουργία της εφαρμογής, με βάση τις αλλαγές στο *context*. Η επιλογή του μοντέλου γίνεται τόσο με βάση την γνώση για το *context*, όσο και με βάση τα χαρακτηριστικά της εκάστοτε εφαρμογής και της κινητής συσκευής που εκτελείται. Το κάθε μοντέλο ασαφούς λογικής δεν είναι στατικά υλοποιημένο, αλλά οι τελεστές που

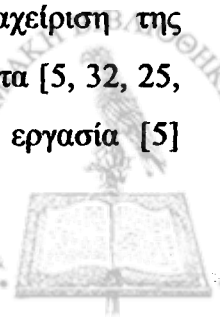
χρησιμοποιούνται στα διάφορα στάδια λειτουργίας του προσδιορίζονται από τον προγραμματιστή της εφαρμογής.

Τα μοντέλα αυτά, όπως είναι ορισμένα στην βιβλιογραφία, διαχειρίζονται μόνο παραμέτρους του *context* που παίρνουν τιμές σε ένα συνεχές διάστημα του συνόλου των πραγματικών αριθμών. Το γεγονός αυτό, μας οδήγησε στην επέκταση του μοντέλου του *context*, ώστε να μπορεί να διαχειρίζεται και διακριτό *context*. Έτσι, δίνουμε την δυνατότητα στον προγραμματιστή της εφαρμογής να ορίσει και να χρησιμοποιήσει στους κανόνες παραμέτρους του *context* που ορίζονται σε ένα μικρό σύνολο τιμών. Έτσι, το σύστημά μας αποκτά μεγαλύτερη ευελιξία ως προς το είδος του *context* που μπορεί να επεξεργαστεί.

Επιπλέον, προσδώσαμε στην υπηρεσία ενδιάμεσου λογισμικού ανακλαστικά χαρακτηριστικά, δίνοντας την δυνατότητα στον χρήστη να διαμορφώσει δυναμικά την συμπεριφορά της. Η υπηρεσία παρέχει ένα *API*, με μεθόδους για την δυναμική διαμόρφωση των παραμέτρων του *context* και των κανόνων και συνεπώς του ενδιάμεσου λογισμικού, κατά την λειτουργία του συστήματος. Χρησιμοποιώντας το συγκεκριμένο *API*, υλοποιήσαμε ένα *GUI* για κινητές συσκευές, με τις λειτουργίες διαμόρφωσης της υπηρεσίας, ώστε να διευκολυνθεί το έργο του προγραμματιστή.

Η χρηστικότητα και η ορθή λειτουργία του μοντέλου του *context* ελέγχθηκε μέσω της ανάπτυξης μίας απλής *context – aware* εφαρμογής. Επιπλέον, εκτελέσαμε ένα σύνολο πειραμάτων, στο οποία μετρήσαμε την χρήση της μνήμης, καθώς και την χρονική απόκριση των βασικών λειτουργιών της υπηρεσίας. Όπως έδειξαν οι πειραματικές μετρήσεις, η χρήση μνήμης είναι ικανοποιητικά μικρή και στα τρία μοντέλα ασαφούς λογικής, ώστε να είναι εφικτή η λειτουργία της υπηρεσίας εξολοκλήρου πάνω σε κινητές συσκευές. Η χρονική απόκριση της υπηρεσίας είναι αρκετά μικρή στα μοντέλα *Sugeno* και *Tsukamoto*. Το μοντέλο *Mamdani* απαιτεί περισσότερο χρόνο για την λήψη αποφάσεων, αλλά ο χρόνος αυτός είναι ικανοποιητικά μικρός, ώστε να μην γίνεται αντιληπτή η λειτουργία της υπηρεσίας από τον χρήστη της εφαρμογής.

Στην βιβλιογραφία έχουν αναφερθεί διάφορες προσεγγίσεις για την διαχείριση της αβεβαιότητας και της ασάφειας της πληροφορίας στα *context – aware* συστήματα [5, 32, 25, 24, 39, 49]. Το μοντέλο διαχείρισης του *context* που περιγράφεται στην εργασία [5]



χρησιμοποιεί ασαφή σύνολα για την αναπαράσταση των παραμέτρων του *context*, ενώ ο μηχανισμός εξαγωγής συμπερασμάτων δεν ακολουθεί κάποιο από τα μοντέλα ασαφούς λογικής. Στην εργασία [32] οι ασαφείς κανόνες παράγονται με την εκπαίδευση ενός ασαφούς δέντρου απόφασης και για την λήψη αποφάσεων χρησιμοποιείται το μοντέλο *Mamdani*. Ωστόσο, η λειτουργία του είναι άμεσα εξαρτημένη από την ύπαρξη κάποιων στατικών κόμβων διαχείρισης του *context*. Το τμήμα του ενδιάμεσου λογισμικού που τρέχει πάνω στις κινητές συσκευές υλοποιεί μόνο την επικοινωνία μεταξύ της κινητής εφαρμογής και του εξυπηρετών για την απόκτηση της πληροφορίας. Στην εργασία [25] περιγράφεται μία τεχνική κατασκευής ενός ασαφούς δέντρου απόφασης για *context – aware* συστήματα. Μία διαφορετική προσέγγιση για την αντιμετώπιση της αβεβαιότητας στο context είναι χρήση Μπεϋζιανών δικτύων [24, 39, 49]. Ωστόσο, τα Μπεϋζιανά δίκτυα βασίζονται στην πιθανοτική λογική και δεν μπορούν να αναπαραστήσουν ασαφή γνώση.

Το σύστημά μας μπορεί να λειτουργήσει αυτόνομα πάνω σε κινητές συσκευές και είναι ανεξάρτητο από κεντρικούς κόμβους παροχής υπηρεσιών. Παρέχει ένα γενικό πλαίσιο για τον ορισμό ενός ασαφούς μοντέλου για ένα *context – aware* σύστημα. Ειδικότερα, το μοντέλο *Sugeno* δεν έχει ενσωματωθεί σε κανένα από τα *context – aware* συστήματα ασαφούς λογικής που αναφέρονται στην βιβλιογραφία. Έτσι, δίνει την δυνατότητα στον προγραμματιστή της εφαρμογής να επιλέξει το κατάλληλο μοντέλο ασαφούς λογικής με βάση τα χαρακτηριστικά της εφαρμογής. Επιπλέον, το μοντέλο του *context* ενσωματώνει και σαφείς παραμέτρους του *context*, ώστε να είναι περισσότερο ευέλικτο. Συμπερασματικά, η υπηρεσία ενδιάμεσου λογισμικού που αναπτύχθηκε στην εργασία αυτή αποτελεί ένα πλήρες σύστημα ασαφούς λογικής για την διαχείριση του *context* και την αυτόματη προσαρμογή των λειτουργιών της εφαρμογής.

8.2 Μελλοντική Εργασία

Στα μελλοντικά μας σχέδια εντάσσεται η ενσωμάτωση στην υπηρεσία ενδιάμεσου λογισμικού ενός μηχανισμού για τον προσδιορισμό των παραμέτρων των συναρτήσεων συμμετοχής, που ορίζονται για κάθε ασαφές σύνολο, με βάση το context. Στην παρούσα εργασία, οι τιμές των παραμέτρων των συναρτήσεων συμμετοχής, ορίζονται εμπειρικά, από κάποιον ειδικό που έχει επίγνωση της συμπεριφοράς των τιμών των παραμέτρων του context. Ωστόσο, είναι σημαντική η ύπαρξη ενός μηχανισμού που να εξάγει αυτόματα τις τιμές αυτών

των παραμέτρων, χρησιμοποιώντας τεχνικές βελτιστοποίησης. Ειδικά, για το μοντέλο Sugeno υπάρχουν τεχνικές μηχανικής μάθησης που εξάγουν τις παραμέτρους των συναρτήσεων για το context εξόδου με βάση το context εισόδου.

Ενδιαφέρον επίσης παρουσιάζει η περιγραφή του context και των κανόνων με πιο τυπική μορφή. Συγκεκριμένα, θα μπορούσε να γίνει χρήση της τεχνολογίας XML για την περιγραφή του συντακτικού των παραμέτρων του context και των κανόνων.

Όσον αφορά την λειτουργικότητα της υπηρεσίας ενδιάμεσου λογισμικού, σκοπεύουμε μελλοντικά να την επεκτείνουμε, προσθέτοντας υπηρεσίες για εύρεση προμηθευτών και υπηρεσιών διαδικτύου. Εφόσον, η πλατφόρμα J2ME περιλαμβάνει το Web Services API, αυτό είναι εφικτό να γίνει χρησιμοποιώντας τις τεχνολογίες των Υπηρεσιών διαδικτύου.

ΑΝΑΦΟΡΕΣ

[1] H. Ailisto, P. Alahuhta, V. Haataja, V. Kylloenen and M. Lindholm. "Structuring Context Aware Applications: Five-Layer Model and Example Case", Workshop on Concepts and Models for Ubiquitous Computing, 2002.

[2] P. A. Bernstein. "Middleware: A Model for Distributed Services", Communications of the ACM, Vol. 39(2), pp 86 – 97, 1996.

[3] G. Biegel and V. Cahill. "A Framework for Developing Mobile, Context Aware Applications", IEEE Conference on Pervasive Computing and Communication, pp 361 – 365, 2004.

[4] K. Bob, A. Quigley, C. Johnson and R. Hexel. "Merino: Towards an Intelligent Environment Architecture for Multi-Granularity Context Description", Workshop on User Modeling for Ubiquitous Computing, 2003.

[5] J. Cao, N. Xing, A. T. S. Chan, Y. Feng and B. Jin. "Service Adaptation Using Fuzzy Theory in Context-Aware Mobile Computing Middleware", Conference on Embedded and Real-Time Computing Systems and Applications, pp 496 – 501, 2005.

[6] L. Capra, W. Emmerich and C. Mascolo. "Middleware for Mobile Computing", Research Note RN/30/01, University College London, 2001.

[7] L. Capra, W. Emmerich and C. Mascolo. "Reflective Middleware Solutions for Context-Aware Applications", Conference on Metalevel Architectures and Separation of Crosscutting Concerns, pp 126 – 133, 2001.

[8] L. Capra, W. Emmerich and C. Mascolo. "CARISMA: Context–Aware Reflective Middleware System for Mobile Applications", IEEE Transactions on Software Engineering, Vol. 29(10), pp 929 – 945, 2003.

[9] H. Chen. "An Intelligent Broker Architecture for Pervasive Context-Aware Systems", PhD thesis, University of Maryland, Baltimore County, 2004.

[10] H. Chen, T. Finin. "An Ontology for a Context-Aware Pervasive Computing Environment", Workshop on Ontologies and Distributed Systems, 2003.

[11] H. Chen, T. Finin and A. Joshi. "Semantic Web in the Context Broker Architecture", IEEE Conference on Pervasive Computing and Communications, pp 277 – 286, 2004.

[12] G. Chen and D. Kotz. "A survey of context-aware mobile computing research", ACM Technical Report: TR 2000 - 381, 2000.

[13] A. K. Dey and G. D. Abowd. "Towards a Better Understanding of Context and Context Awareness", Workshop on the What, Who, Where, When, and How of Context-Awareness, 2000.

[14] A. K. Dey and G. D. Abowd. "The Context Toolkit: Aiding the Development of Context–Aware Applications", Workshop of Software Engineering for Wearable and Pervasive Computing, 2000.

[15] A. K. Dey and G. D. Abowd. "A Conceptual Framework and a Toolkit for Supporting Rapid Prototyping of Context-Aware Applications", Human-Computer Interactions Journal, Vol. 16(2 – 4), pp 7 – 166, 2001.

[16] A. Dey, J. Manko and G. Abowd. "Distributed Mediation of Ambiguous Context in Aware Environments", ACM Symposium on User Interface Software and Technology, pp 121 – 130, 2002.

[17] D. Dubois and H. Prade. "Fuzzy Sets and Systems: Theory and Applications", Academic Press, New York, 1980.

[18] W. Emmerich. "Software Engineering and Middleware: A Roadmap", Conference on Software Engineering, pp 117 – 129, 2000.

[19] H. Enderton, "A Mathematical Introduction to Logic", Academic Press, 1972.

[20] P. Fahy and S. Clarke. "CASS – A Middleware for Mobile Context–Aware Applications", Workshop on Context Awareness, 2004.

[21] A. J. Gonzalez, R. Ahlers. "Context-Based Representation of Intelligent Behaviour in Training Simulations", Transactions of the Society for Computer Simulation International, Vol. 15(4), pp 153 – 166, 1999.

[22] P. Gray and D. Salber. "Modelling and Using Sensed Context in the Design of Interactive Applications", IFIP Conference on Engineering for Human-Computer Interaction, pp 317 – 336, 2001.

[23] T. Gu, H. K. Pung and D. Q. Zhang. "A Middleware for Building Context-Aware Mobile Services", IEEE Vehicular Technology Conference, 2004.

[24] T. Gu, H. K. Pung and D. Q. Zhang. "A Bayesian Approach for Dealing with Uncertain Contexts", Conference on Pervasive Computing, 2004.

[25] D. Guan, W. Yuan, A. Gavrilov, S. Lee, Y. Lee and S. Han. "Using Fuzzy Decision Tree to Handle Uncertainty in Context Deduction", Conference on Intelligent Computing, pp 63 – 72, 2006.

[26] K. Henricksen and J. Indulska. "A Software Engineering Framework for Context-Aware Pervasive Computing", IEEE Conference on Pervasive Computing and Communications, pp 77 – 86, 2004.

[27] K. Henricksen and J. Indulska. "Modelling and Using Imperfect Context Information", IEEE Conference on Pervasive Computing and Communications Workshops, pp 33 – 37, 2004.

[28] K. Henricksen, J. Indulska, T. McFadden and S. Balasubramaniam. "Middleware for Distributed Context-Aware Systems", Conference on Distributed Objects and Applications, 2005.

[29] K. Henricksen, J. Indulska and A. Rakotonirainy. "Modelling Context Information in Pervasive Computing Systems", Conference on Pervasive Computing, pp 167 – 180, 2002.

[30] T. Hofer, W. Schwinger, M. Pichler, G. Leonhartsberger and J. Altmann. "Context-Awareness on Mobile Devices – Hydrogen approach", Conference on System Sciences, pp 292 – 302, 2002.

[31] G. Judd and P. Steenkiste. "Providing Contextual Information to Pervasive Computing Applications", IEEE Conference on Pervasive Computing and Communications, pp 133 – 142, 2003.

[32] M. Khedr. "Enhancing Applicability of Context-Aware Systems Using Agent-Based Hybrid Inference Approach", Vehicular Technology Conference, pp 2785 – 2789, 2005.

[33] Α. Κοντογιώργης. "Σχεδίαση και Ανάπτυξη Ενδιάμεσου Λογισμικού Ενήμερου Περιεχομένου για Κινητές Συσκευές", Μεταπτυχιακή Εργασία, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, 2005.

[34] H. Lei, D. M. Sow, J. S. Davis, G.Banavar and M. R. Ebling. "The Design and Applications of a Context Service", ACM SIGMOBILE Mobile Computing and Communications, Vol. 6(4), pp 44 – 55, 2002.

[35] D. Lopez de Ipipa and E. Katsiri. "An ECA Rule-Matching Service for Simpler Development of Reactive Applications", IEEE Distributed Systems Online, Vol. 2(7), 2001.

[36] E. H. Mamdani and S. Assilian. "An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller", International Journal of Man-Machine Studies, Vol. 7(1), pp 1 – 13, 1975.

[37] Z. Manna and A. Pnueli. "The Temporal Logic of Reactive and Concurrent Systems", Springer – Verlag, 1992.

[38] R. Meier and V. Cahill. "Exploiting Proximity in Event-Based Middleware for Collaborative Mobile Applications", IFIP Conference on Distributed Applications and Interoperable Systems, pp 285 – 296, 2003.

[39] A. Ranganathan, J. Al-Muhtadi, R. H. Campbell. "Reasoning about Uncertain Contexts in Pervasive Computing Environments", IEEE Pervasive Computing, Vol. 3(2), pp 62 – 70, 2004.

[40] A. Ranganathan and R. H. Campbell. "A Middleware for Context-Aware Agents in Ubiquitous Computing Environments", Middleware 2003, pp 143 – 161, 2003.

[41] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell and K. Nahrstedt. "A Middleware Infrastructure for Active Spaces", IEEE Pervasive Computing, Vol. 1(4), pp 74 – 83, 2002.

[42] D. Salber, A. K. Dey and G. D. Abowd. "The Context Toolkit: Aiding the Development of Context-Aware Applications", Workshop on Software Engineering for Wearable and Pervasive Computing, 2000.

[43] B. N. Schilit, N. I. Adams and R. Want. "Context-Aware Computing Applications", Workshop on Mobile Computing Systems and Applications, pp 85 – 90, 1994.

[44] B. N. Schilit and M .M. Theimer. "Disseminating Active Map Information to Mobile Hosts", IEEE Networks, Vol. 8(5), pp 22 – 32, 1994.

[45] A. Shehzad, H .Q. Ngo and S. Y. Lee. "Developing Context-Aware Ubiquitous Computing Systems with a Unified Middleware Framework", Conference on Embedded and Ubiquitous Computing, 2004.

[46] A. Shehzad, H. Q. Ngo, S. Y. Lee and Y-K. Lee. "A Comprehensive Middleware Architecture for Context-Aware Ubiquitous Computing Systems", ACIS Conference on Computer and Information Science, pp 251 – 256, 2005.

[47] M. Sugeno and G. T. Kang. "Structure Identification of Fuzzy Model", Fuzzy Sets and Systems, Vol. 28, pp 15 – 33, 1988.

[48] P-N. Tan, M. Steinbach and V. Kumar. "Introduction to Data Mining", Addison Wesley, 2005.

[49] B. A. Truong, Y-K. Lee and S-U. Lee. "Modeling Uncertainty in Context-Aware Computing", ACIS Conference on Computer and Information Science, pp 676 – 681, 2005.

[50] Y. Tsukamoto. "An Approach to Fuzzy Reasoning Method", Advances in Fuzzy Set Theory and Applications, pp 137 – 149, 1979.

[51] L. A. Zadeh. "Fuzzy Sets", Information and Control, Vol. 8, pp 338 – 353, 1966.

[52] L.A. Zadeh. "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes", IEEE Transactions on Systems, Man and Cybernetics, Vol. (1), pp 28 – 44, 1973.

ΣΥΝΤΟΜΟ ΒΙΟΓΡΑΦΙΚΟ

Η Ελένη Γεώργα γεννήθηκε το 1981 στην Σπάρτη. Το 2000 ξεκίνησε τις σπουδές της στο Τμήμα Πληροφορικής του Πανεπιστημίου Ιωαννίνων, τις οποίες και ολοκλήρωσε το 2005. Το διάστημα 2005 – 2008 παρακολούθησε το μεταπτυχιακό πρόγραμμα σπουδών του ίδιου τμήματος.