ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Μοντελοποίηση Δεδομένων Περιβάλλοντος Προσωπικών Πληροφοριακών Συστημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΓΕΩΡΓΙΟΣ Ι. ΡΟΒΗΣ

Επιβλέπων : Ιωάννης Ψαρράς

Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2013

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Μοντελοποίηση Δεδομένων Περιβάλλοντος Προσωπικών Πληροφοριακών Συστημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΓΕΩΡΓΙΟΣ Ι. ΡΟΒΗΣ

Επιβλέπων : Ιωάννης Ψαρράς

Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την .....

.....	.....	.....
Ι. Ψαρράς	Β. Ασημακόπουλος	Γρ. Μέντζας
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2013

.....

ΓΕΩΡΓΙΟΣ Ι. ΡΟΒΗΣ

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

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Ιωάννη Ψαρρά για τη δυνατότητα που μου προσέφερε να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα, καθώς και για την καθοδήγηση που μου παρείχε καθ' όλη τη διάρκεια εκπόνησης της διπλωματικής μου εργασίας.

Η εργασία αυτή πραγματοποιήθηκε από κοινού με τον συνάδελφο κ. Παντελή Κακαβά, στον οποίον οφείλω να αποδώσω τα εύσημα για ένα μεγάλο κομμάτι της. Με τον κ. Κακαβά θα συνεχίσουμε τη στενή μας συνεργασία για την εκπόνηση της δικιάς του διπλωματικής εργασίας ως συνέχεια της επιστημονικής μας έρευνας πάνω στο θέμα του παρόντος,

Επίσης θα ήθελα να ευχαριστήσω τον κ. Χρήστο Ντάνο για τη βοήθεια και τις πολύτιμες συμβουλές που μου προσέφερε στα πλαίσια της ενασχόλησής μου με το θέμα.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου και τους φίλους μου που στάθηκαν δίπλα μου όλα αυτά τα χρόνια, και ιδιαίτερα τον Μάριο, τον Παντελή και τον Άρη για την βοήθεια τους κατά τα χρόνια μας στο ΕΜΠ.

Πίνακας Περιεχομένων

Πίνακας Περιεχομένων....7

Ευρετήριο Εικόνων....9

Περίληψη....10

Abstract....12

1 Εισαγωγή....15

2 Ορισμοί και Βασικές Έννοιες....20

2.1 Context – Context Awareness....21

2.2 Context Aware Προγραμματισμός....26

2.3 Διαχείριση Context....29

2.3.1 Συλλογή Context....30

2.3.2 Context Συλλογισμός....34

2.3.3 Context διαχείριση....36

2.4 Λογισμικό πλαίσιο – Πλατφόρμα λογισμικού....40

2.4.1 Πλαίσιο....40

2.4.2 Πλατφόρμα....41

2.5 Ενδιάμεσο Λογισμικό....42

2.5.1 Context Aware Ενδιάμεσο Λογισμικό....47

3 Ανασκόπηση Βιβλιογραφίας (State of the Art)....52

4 Webinos....61

4.1 Όραμα - Χαρακτηριστικά webinos....62

4.2 Λειτουργίες - Καινοτομίες webinos....63

4.3 Τα APIs του webinos....67

5 Webinos Context 78
5.1 Διαχειριστής Context webinos 78
5.1.1 Εισαγωγή 78
5.1.2 Context Λειτουργικότητα 81
5.1.3 Επισκόπηση Ροής Δεδομένων Διαχειριστή Context 84
5.1.4 Context Λεξικό 86
5.1.5 Διάγραμμα Ανάπτυξης 86
5.1.6 Υποκλοπή Μηνύματος και Αποθήκευση 88
5.1.7 Context Querying 89
5.1.8 Κανόνες Context 90
5.1.9 Προγραμματισμένες Κλήσεις API 91
5.2 Context Σενάρια & Περιπτώσεις Χρήσης 92
5.2.1 Εντοπισμός χαμένης συσκευής 92
5.2.2 Τοποθεσία απομακρυσμένης συσκευής 93
5.2.3 Συσκευές Πολλαπλών Χρηστών 94
5.2.4 Συλλογή και αναπαράσταση Context Δεδομένων 95
5.2.5 Συνδυασμός και context λογισμός 96
5.2.6 Φιλτράρισμα για αποφυγή διπλοεγγραφών 97
6 References 99

Ευρετήριο Εικόνων

Εικόνα I: Το context από την οπτική του χρήστη **22**

Εικόνα II : Παράδειγμα προμηθευτών Context Δεδομένων **32**

Εικόνα III: Παράδειγμα Context Συλλογισμού **35**

Εικόνα IV: Λειτουργία Ενδιάμεσου Λογισμικού **44**

Εικόνα V: Αφαιρετική Αρχιτεκτονική Ενός Context Aware Συστήματος **49**

Εικόνα VI: Λειτουργίες Context - Aware Ενδιάμεσου Λογισμικού **50**

Εικόνα VII: Βασική Ιδέα - Καινοτομίες webinos **65**

Εικόνα VIII: Αρχιτεκτονική webinos **77**

Εικόνα IX: Επισκόπηση Ροής Δεδομένων του Διαχειριστή Context **84**

Εικόνα X: Διάγραμμα Ανάπτυξης Διαχειριστή Context **87**

Εικόνα XI: Υποκλοπή μηνύματος και επισκόπηση αποθήκευσης **88**

Εικόνα XII: Διαδικασία Context Querying **89**

Εικόνα XIII: Επισκόπηση Context Κανόνων **90**

Εικόνα XIV: Προγραμματισμένες Κλήσεις API **91**

Περίληψη

Η ταχεία εξέλιξη της τεχνολογίας έχει οδηγήσει στην αλματώδη αύξηση τόσο της χρήσης και της ποικιλίας των κινητών/φορητών προσωπικών συσκευών, όσο και της πληροφορίας που αυτά δημιουργούν και διαχειρίζονται. Ένα από τα προβλήματα που αντιμετωπίζουν σήμερα οι μηχανικοί λογισμικού είναι η σχεδίαση και η ανάπτυξη εφαρμογών, οι οποίες θα παρακολουθούν το περιβάλλον τους και θα προσαρμόζονται κατάλληλα στις αλλαγές που πραγματοποιούνται. Τέτοιου είδους εφαρμογές είναι απαραίτητες για κινητές συσκευές και συστήματα, όπου το περιβάλλον λειτουργίας και οι παράγοντες που επηρεάζουν την εφαρμογή μεταβάλλονται συνεχώς. Η λύση σε αυτό το πρόβλημα μοιάζει να είναι η ανάπτυξη κατάλληλου ενδιάμεσου λογισμικού, για την υποστήριξη αυτών των εφαρμογών. Η ανάπτυξη ενδιάμεσου λογισμικού ικανού να διαχειρίζεται τα δεδομένα περιβάλλοντος και τις μεταβολές που προκύπτουν, παρουσιάζει σημαντικά προβλήματα που σχετίζονται με την πολυπλοκότητα του σχεδιασμού και τους περιορισμένους υπολογιστικούς πόρους των συσκευών.

Σε αυτήν την εργασία μελετάμε μία πλατφόρμα ενδιάμεσου λογισμικού, το webinos, το οποίο σχεδιάζεται για την υποστήριξη και την προσαρμογή εφαρμογών σε δυναμικά μεταβαλλόμενα περιβάλλοντα προσωπικών πληροφοριακών συστημάτων. Στόχος του είναι να προσφέρει μια πλατφόρμα που θα υποστηρίζει την γρήγορη δημιουργία καινοτόμων και ασφαλών δικτυακών εφαρμογών για κάθε είδους έξυπνες συσκευές, όπως κινητά τηλέφωνα, Η/Υ (φορητούς και μη), τηλεοράσεις και συσκευές ενσωματωμένες σε οχήματα. Αρχικά μελετάμε τις μέχρι τώρα προτεινόμενες αρχιτεκτονικές ενδιάμεσου λογισμικού και στην συνέχεια προχωράμε στην μοντελοποίηση του τρόπου λειτουργίας του διαχειριστή δεδομένων περιβάλλοντος της συγκεκριμένης πλατφόρμας, επιδιώκοντας την βελτιστοποίηση του τρόπου λειτουργίας της. Αυτό προσπαθούμε να το καταφέρουμε προτείνοντας σημαντικές τροποποιήσεις των υπαρχόντων μεθόδων, αλλά και με εναλλακτικές αρχιτεκτονικές, συμβατές πάντα με την πλατφόρμα. Τελικός μας στόχος είναι τόσο η ικανοποίηση των αναγκών του τελικού χρήστη της εφαρμογής που θα τρέχει πάνω στην πλατφόρμα, όσο και η δημιουργία της κατάλληλης υποδομής υποστήριξης προς τον χρήστη – προγραμματιστή, ώστε να διευκολύνεται η σχεδίαση και ανάπτυξη μιας εφαρμογής.

Λέξεις Κλειδιά: << **ενδιάμεσο λογισμικό, πλατφόρμα, πλαίσιο, διάχυτες εφαρμογές, κατανεμημένος προγραμματισμός, κινητές συσκευές, συλλογή, αποθήκευση, φιλτράρισμα, συλλογισμός**>>

Abstract

Rapid technological evolution has led to a huge increase in the use and variety of mobile personal devices, as well as in the information derived and processed by them. One of the major problems software engineers face, is the design and development of applications, which will be able to notice the changes that occur in their surrounding environment and adapt accordingly. Such applications, context – aware, are becoming more and more necessary for mobile devices and systems, where the factors influencing the application or device are instantly changing. One solution appears to be the development of the appropriate middleware platform that will support those applications. The main problems in developing such a middleware that will handle the unpredictably changing context data are the complexity its design requires and the limited amount of computing power that mobile devices have.

This thesis studies a middleware platform, webinos, which was designed for the support and adjustment of applications in dynamically changing personal information systems. Webinos' vision is the creation of a middleware platform which will support the quick development of innovating and secure web applications for every kind of smart devices, such as mobile phones, PCs (laptops, tablets etc), TVs and devices embedded in vehicles. To begin with we examine the proposed middleware architectures, modeling the functions and uses of the platform's context manager, aiming at optimizing them. We try to achieve that either by proposing adjustments to the currently used methods, or by suggesting new architectures, which of course are compatible with the platform. Our final goal is to satisfy the needs of the application's end user, while creating the best infrastructure for a programmer to use webinos, in order to help himself in designing and developing an application.

Keywords: <<context data, context awareness, context aware computing, webinos, middleware, platform, context reasoning, handling, context management, pervasive and ubiquitous computing >>

1 Εισαγωγή

Οι άνθρωποι είναι αρκετά επιτυχείς στο να μεταφέρουν τις ιδέες ο ένας στον άλλο και στο να τις αντιλαμβάνονται και να αντιδρούν κατάλληλα. Μερικοί από τους παράγοντες στους οποίους οφείλεται αυτό είναι: ο πλούτος της γλώσσας που μοιράζονται, η κοινή αντίληψη για το πώς λειτουργεί ο κόσμος, και μια έμμεση και σιωπηρή, κοινή κατανόηση των καθημερινών καταστάσεων [13]. Όταν οι άνθρωποι μιλάνε μεταξύ τους, έχουν την ικανότητα να χρησιμοποιούν έμμεσα πληροφορίες ανάλογα με την κάθε κατάσταση-συζήτηση για να αυξάνουν το εύρος της. Κατά αυτόν τον τρόπο δημιουργούν, μπορούμε να πούμε, ένα πλαίσιο (context) γύρω από κάθε επικοινωνιακή κατάσταση, θέτοντας τα όρια μέσα στα οποία κυμαίνονται οι πληροφορίες που ανταλλάσσονται. Το πλαίσιο αυτό τους επιτρέπει, όπως είπαμε και παραπάνω, να αντιδρούν κατάλληλα πάνω σε αυτά στα οποία συνδιαλέγονται. Είναι προφανές ότι διαφορετικά θα μιλήσεις για κάτι ή για κάποιον, εφόσον γνωρίζεις τη σχέση του συνομιλητή σου με το θέμα της συζήτησης. Π.χ. Αλλιώς θα αναφερθείς σε μία αρρώστια ή σε ένα ατυχές συμβάν, εφόσον γνωρίζεις ότι έχει βιώσει κάτι αντίστοιχο ο ακροατής σου.

Η διαδικασία καθορισμού αυτού του νοητού πλαισίου σε μια συζήτηση γίνεται αυτόματα, με ποικίλους τρόπους και χωρίς τις περισσότερες φορές να γίνεται αντιληπτή ούτε από τα ίδια τα μέλη που θέτουν το πλαίσιο. Ο τρόπος με τον οποίο τίθεται το θέμα μιας συζήτησης, αλλά και το εύρος των πληροφοριών που την πλαισιώνουν μπορεί να εξαρτώνται από πολλούς ανεξάρτητους παράγοντες. Κύριος τρόπος ανταλλαγής πληροφοριών που αφορούν το context μιας συζήτησης, αποτελεί η γλώσσα του σώματος, μέσω της οποίας είτε άμεσα είτε έμμεσα μπορεί ο ένας συνομιλητής να περάσει στον άλλο τις απαραίτητες πληροφορίες όσον αφορά τα λογικά όρια της συζήτησής τους. Π.χ. Οι κινήσεις των χεριών, ο ρυθμός, ο τόνος και η ένταση της ομιλίας καθορίζουν τον τρόπο μετάδοσης της πληροφορίας και εν προκειμένω επηρεάζουν και το ίδιο το περιεχόμενο της συζήτησης. Ακόμα, μια κοινή εμπειρία ή μία κοινή γνώση ενός γεγονότος ή αντικειμένου μπορούν να αποτελέσουν σημαντικούς παράγοντες καθορισμού του πλαισίου μιας συζήτησης. Για παράδειγμα, η από κοινού παρακολούθηση μίας ταινίας ή η κοινή εργασία σε έναν χώρο μπορούν να πλαισιώσουν μία συζήτηση χωρίς να αποτελούν απαραίτητα μέρος του κύριου περιεχομένου της. Όλα τα παραπάνω, καθώς και άλλα πολλά, αποτελούν τους λόγους για τους οποίους, σε οποιαδήποτε μορφή και αν είναι, η επικοινωνία ανάμεσα σε δύο ή και περισσότερους ανθρώπους είναι μοναδική και είναι δύσκολο να κατανοηθεί από έναν τρίτο που παρακολουθεί, είτε ζωντανά είτε μέσω οπτικοακουστικού

υλικού τη συζήτηση, χωρίς πιθανώς να βλέπει τους συνομιλητές ή χωρίς να γνωρίζει κάτι το οποίο αυτοί ξέρουν από κοινού.

Δυστυχώς δεν μπορούμε να πούμε το ίδιο για την επικοινωνία ανάμεσα σε έναν άνθρωπο και σε μία μηχανή, και συγκεκριμένα έναν Η/Υ. Η δυνατότητα ενός ανθρώπου να μεταφέρει ιδέες και σκέψεις σε έναν Η/Υ είναι ιδιαίτερα περιορισμένη και εξαρτάται κυρίως από τους μηχανισμούς εισαγωγής πληροφορίας του κάθε Η/Υ ενώ βασίζεται δευτερεύοντος στην συνειδητή προσπάθεια και φαντασία του ίδιου του προγραμματιστή. Στον παραδοσιακό διαδραστικό προγραμματισμό αυτό έχει οδηγήσει σε έναν ιδιαίτερα φτωχό τρόπο εισαγωγής πληροφοριών από τους χρήστες στους Η/Υ. Όπως είναι γνωστό οι τρόποι μετάδοσης πληροφορίας περιορίζονται στις γνωστές συσκευές που χρησιμοποιούμε και συνδέονται με έναν Η/Υ, στις διάφορες θύρες του, και είναι είτε εισόδου, είτε εξόδου, είτε εισόδου/εξόδου. Από ένα παράδειγμα για την κάθε μία αποτελούν το ποντίκι (εισόδου), τα ηχεία (εξόδου) και η οθόνη αφής (εισόδου/ εξόδου).

Από τα παραπάνω είναι προφανές ότι προς το παρόν δεν είναι δυνατό για έναν Η/Υ να αντιληφθεί και να χρησιμοποιήσει, σε οποιαδήποτε λειτουργία του, μεγάλο κομμάτι των πληροφοριών που προσπαθεί, έμμεσα κυρίως, να του περάσει ο άνθρωπος. Ως εκ τούτου, τα πλεονεκτήματα αυτών των πληροφοριών πλαισίου (context) της ανθρώπινης επικοινωνίας μένουν ανεκμετάλλευτα. Αν πετυχαίναμε τη βελτίωση στον τρόπο αντίληψης και πρόσβασης του Η/Υ σε τέτοιου είδους πληροφορίες θα καταφέρναμε να αυξήσουμε σημαντικά τον πλούτο της επικοινωνίας ανθρώπου- μηχανής, οπότε και να ενισχύσουμε την παραγωγικότητα και την χρησιμότητα των υπολογιστικών λειτουργιών και υπηρεσιών. Για το σκοπό αυτό πρέπει αρχικά να καταλάβουμε τι εννοούμε όταν λέμε πληροφορίες πλαισίου, τον τρόπο με τον οποίον αυτές μπορούν να χρησιμοποιηθούν, και την αρχιτεκτονική υποστήριξη που απαιτείται για την ερμηνεία και εκμετάλλευσή τους. Η κατανόηση του context από την πλευρά των σχεδιαστών των εφαρμογών, θα επιτρέψει την δυνατότητα επιλογής του είδους του context που θα χρησιμοποιήσει η εφαρμογή τους, καθώς και τις συμπεριφορές και αντιδράσεις ως προς αυτό που θα πρέπει να προβλέπονται και να ενσωματώνονται. Τέλος, η ανάπτυξη και χρήση της κατάλληλης αρχιτεκτονικής υποστήριξης θα διευκολύνει ~~σ~~την δημιουργία των επιθυμητών εφαρμογών διαχείρισης context. Η αρχιτεκτονική υποστήριξη θα αποτελείται από δύο μέρη: τις υπηρεσίες και τις αφαιρετικές έννοιες (αοριστίες).

Την τελευταία δεκαετία παρατηρείται μια ραγδαία εξάπλωση των κινητών συσκευών στην καθημερινή μας ζωή. Οι υπολογιστές αποκτούν όλο και μικρότερες διαστάσεις, ώστε να μεταφέρονται εύκολα και να μεσολαβούν σε περισσότερες διαδικασίες στην καθημερινότητα

του χρήστη. Οι κινητές συσκευές σήμερα περιλαμβάνουν φορητούς υπολογιστές (*laptops*), υπολογιστές παλάμης (*tablets*), έξυπνα κινητά τηλέφωνα (*smartphones*) και φορητές συσκευές, όπως ρολόγια. Η ευρεία χρήση κινητών υπολογιστικών συσκευών έχει ως αποτέλεσμα τη σταδιακή μετατροπή των σύγχρονων υπολογιστικών συστημάτων από *κατανεμημένα* σε *κινητά.* Οι υπολογιστές δεν συνδέονται πλέον, μόνο μέσω του *Internet*, αλλά σχηματίζουν και ασύρματα δίκτυα με κινητούς κόμβους. Οι κόμβοι αυτών των δικτύων αποτελούνται από σταθερούς υπολογιστές, αλλά και κινητές συσκευές, οι οποίες συνεχώς αλλάζουν θέση και περιβάλλον λειτουργίας. Τα κινητά συστήματα περιλαμβάνουν ένα μεγάλο αριθμό προσπελάσιμων και συχνά αόρατων υπολογιστικών συσκευών. Οι συσκευές αυτές μπορεί να βρίσκονται ενσωματωμένες στο περιβάλλον ή στο χρήστη. Συνδέονται σε μία, ταχύτατα και αυθαίρετα, μεταβαλλόμενη δικτυακή υποδομή, στην οποία βρίσκονται κατανεμημένες υπηρεσίες, διαθέσιμες προς τους χρήστες.

Οι κινητές συσκευές έχουν κάποια ιδιαίτερα χαρακτηριστικά, που τις διαφοροποιούν από τους σταθερούς ηλεκτρονικούς υπολογιστές, ως προς το λογισμικό που μπορούν να υποστηρίξουν:

- Έχουν περιορισμένους υπολογιστικούς πόρους όπως: υπολογιστική ισχύ, μνήμη και μόνιμο αποθηκευτικό χώρο.
- Επικοινωνούν μεταξύ τους και με άλλους υπολογιστές ασύρματα, σχηματίζοντας δίκτυα με ακαθόριστη δομή και συχνές συνδέσεις και αποσυνδέσεις κόμβων (*ad – hoc* δίκτυα). Το εύρος ζώνης στην επικοινωνία των συσκευών αυτών είναι συνήθως περιορισμένο, ενώ η συνεχής διαθεσιμότητα του δικτύου δεν είναι δεδομένη.
- Ο χρήστης έχει τη δυνατότητα να κινείται στο χώρο αλλάζοντας συνεχώς το περιβάλλον λειτουργίας της συσκευής. Σε κάποιες περιπτώσεις η γεωγραφική θέση του χρήστη και περιβαλλοντολογικοί παράγοντες μπορεί να επηρεάζουν την εκτέλεση της εφαρμογής που τρέχει στη συσκευή.
- Οι κινητές συσκευές έχουν περιορισμένη αυτονομία. Η λειτουργία τους εξαρτάται απόλυτα από τη μπαταρία τους.

Τα συστήματα με τα παραπάνω χαρακτηριστικά ονομάζονται *διάχυτα* (*pervasive ή ubiquitous*). Ο όρος *pervasive* αναφέρεται για πρώτη φορά από τον *Weiser* το *1991 [46]*. Ο όρος περιγράφει τη *διαφανή ενσωμάτωση των υπολογιστικών συσκευών στην καθημερινότητα του χρήστη, όπου η τεχνολογία χάνεται στο παρασκήνιο εστιάζοντας στο χρήστη και τις δραστηριότητές του.*

Αυτή η πρόσφατη διάδοση των φθηνών, μικρών, και όλο και πιο ακριβών τεχνολογιών αισθητήρων έχουν οδηγήσει σε μια επανάσταση στον τομέα της πληροφορίας, καθώς οι εφαρμογές που λειτουργούν σε διάχυτα συστήματα γίνονται ολοένα και πιο διαδεδομένες και πρέπει να μπορούν να αλληλεπιδρούν με το φυσικό περιβάλλον και να προσαρμόζονται σε μεταβαλλόμενες συνθήκες. Για να το επιτύχουν αυτό απαιτείται να έχουν *επίγνωση* του περιβάλλοντος στο οποίο λειτουργούν και των *παραγόντων* που επηρεάζουν τη λειτουργία τους. Το σύνολο των παραγόντων του οποιουδήποτε περιβάλλοντος μέσα στο οποίο τρέχει μια εφαρμογή και που επηρεάζουν τη λειτουργία της ονομάζεται *context*, π.χ. τοποθεσία, χρόνος, θερμοκρασία, φωτεινότητα, άλλοι άνθρωποι σε κοντινή απόσταση κ.α. Με τη χρήση του *context*, τα υπολογιστικά συστήματα μπορούν να γίνουν φιλικότερα προς τους χρήστες, πιο αποδοτικά και να παρέχουν ποιοτικότερες υπηρεσίες. Οι πληροφορίες αυτές, οι οποίες αφορούν κυρίως της συνθήκες κάτω από τις οποίες «τρέχει» μια εφαρμογή είναι ιδιαίτερα σημαντικές για συστήματα, όπου το *context* μεταβάλλεται συχνά και απρόβλεπτα και οι υπηρεσίες οφείλουν να προσαρμόζονται στις μεταβολές.

Αυτή η αντίληψη του φυσικού περιβάλλοντος από τις εφαρμογές, καθώς και η χρήση των πληροφοριών που συλλέγονται για την επίτευξη των στόχων τους, είναι γνωστή ως context - awareness και μια σειρά από πολλά υποσχόμενες εφαρμογές έχουν κάνει την εμφάνισή τους σε αυτόν τον τομέα. Η τεχνολογία του context – awareness αποκτά ιδιαίτερη σημασία σε περιβάλλοντα κινητών συσκευών όπου το λειτουργικό περιβάλλον αλλάζει συνεχώς εξαιτίας της φορητότητας των συσκευών και των χαρακτηριστικών της ασύρματης τεχνολογίας επικοινωνιών. Σε τέτοια περιβάλλοντα, μέσω του context – awareness μπορεί να επιτραπεί στις εφαρμογές να ανταποκρίνονται έξυπνα σε μεταβλητό εύρος ζώνης, αναξιόπιστες συνδέσεις και για την εξοικονόμηση διαφορετικών συνδέσεων.

Οι εφαρμογές αυτές που μπορούν να γνωρίζουν το *context* και να το χρησιμοποιήσουν κατάλληλα, ώστε να προσαρμόσουν τις λειτουργίες τους σε αυτό, χαρακτηρίζονται ως *Context - Aware (CA)*. Για να αποκτήσει μια εφαρμογή *CA* χαρακτηριστικά, πρέπει να αναπτυχθεί επιπρόσθετο λογισμικό. Το επιπρόσθετο λογισμικό αφορά, την εύρεση, τη διαχείριση, και τη χρησιμοποίηση του *context*, για την παροχή ποιοτικότερων υπηρεσιών στο χρήστη. Σύμφωνα με τις αρχές της τεχνολογίας λογισμικού τα *CA* συστήματα θα πρέπει να καλύπτουν τις απαιτήσεις που σχετίζονται με *ετερογένεια, κατανομή (distribution), διαλειτουργικότητα (interoperability) και επεκτασιμότητα (extensibility).* {webinos.org}. Επιπρόσθετα θα πρέπει να σχεδιαστούν ώστε να ανταποκρίνονται σε παράγοντες όπως, ξαφνικά ή απρόβλεπτα φαινόμενα,

μετακίνηση φυσικών ή λειτουργικών οντοτήτων, διαμόρφωση, αβεβαιότητα, κακόβουλες ενέργειες και αποτυχίες που σχετίζονται με το φυσικό περιβάλλον [43]. Η ανάπτυξη συστημάτων με αυτά τα χαρακτηριστικά είναι προφανώς, δύσκολη και χρονοβόρα διαδικασία για τους μηχανικούς λογισμικού.

Το ζητούμενο είναι η σχεδίαση και ανάπτυξη μιας κοινής πλατφόρμας, πάνω στην οποία να μπορούν εύκολα και ευέλικτα να χτιστούν *CA* εφαρμογές. Μία κοινή βάση ανάπτυξης εξασφαλίζει την επαναχρησιμοποίηση λογισμικού και ανεξαρτητοποιεί την εφαρμογή από το λογισμικό που αφορά τη διαχείριση του *context*. Το λογισμικό υποστήριξης *CA* εφαρμογών παίζει το ρόλο *ενδιάμεσου λογισμικού* κάτω από την εφαρμογή και παρέχει υπηρεσίες διαχείρισης του *context*. Η ανάπτυξη κατάλληλου ενδιάμεσου λογισμικού για *CA* εφαρμογές έχει προσελκύσει το ενδιαφέρον της ερευνητικής κοινότητας την τελευταία δεκαετία. Η κατάλληλη αρχιτεκτονική, οι υπηρεσίες που παρέχονται αλλά και ο τρόπος αλληλεπίδρασης με το επίπεδο εφαρμογής είναι τα βασικά θέματα έρευνας.

Μία από τις μεγαλύτερες προκλήσεις στον context – aware προγραμματισμό, και η οποία δεν έχει ακόμα γίνει προφανής, είναι ο σχεδιασμός και οι προδιαγραφές ενός ενδιάμεσου λογισμικού και υπηρεσιών για την υποστήριξη του προγραμματιστή τέτοιων εφαρμογών. Στην παρακάτω εργασία προσπαθούμε να προσδιορίσουμε τα κύρια γνωρίσματα που πρέπει να έχει ένα ενδιάμεσο λογισμικό, με προσανατολισμό τις CA εφαρμογές, ώστε να μπορεί να παρέχει τα κατάλληλα εργαλεία τόσο στον προγραμματιστή – χρήστη του ενδιάμεσου λογισμικού, όσο και στον τελικό χρήστη της εφαρμογής. Στην προσπάθεια μας αυτή δουλεύουμε πάνω στην τρέχουσα έκδοση του ενδιάμεσου λογισμικού webinos, του οποίου οι στόχοι, ως προς αυτή του την κατεύθυνση τουλάχιστον ταυτίζονται με τους δικούς μας. Εξετάζουμε τα πιθανά σενάρια χρήσης και από τις δύο πλευρές των δύο προαναφερθέντων χρηστών του ενδιάμεσου λογισμικού, ώστε να μπορέσουμε να αποσαφηνίσουμε και να ξεχωρίσουμε από την μία τις λειτουργικές και από την άλλη τις μη λειτουργικές απαιτήσεις που θα πρέπει να ικανοποιούνται από και προς το webinos, για την επίτευξη των στόχων μας. Τελική μας επιδίωξη είναι η δημιουργία ενός μοντέλου ενδεικτικής λειτουργίας και προδιαγραφών που θα πρέπει να τηρούνται ώστε να έχουμε τα ζητούμενα αποτελέσματα και την επιθυμητή συμπεριφορά του ενδιάμεσου λογισμικού, απέναντι σε κάθε χρήστη, με γνώμονα της πληροφορίες περιβάλλοντος και τις μεταβολές τους.

2 Ορισμοί και Βασικές Έννοιες

Context είναι «αυτό που περιβάλλει και δίνει σημασία σε κάτι άλλο». Αρκετοί τομείς της επιστήμης των Η/Υ έχουν προσπαθήσει να ερευνήσουν περαιτέρω την έννοια αυτή τα τελευταία 50 χρόνια, να συσχετίσουν πληροφορίες επεξεργασίας και επικοινωνίας με το πλαίσιο της κατάστασης, μέσα στο οποίο αυτές υφίστανται. Ειδικότερα το context παίζει κύριο ρόλο στην φυσική υπολογιστική γλώσσα και εν γένει στην αλληλεπίδραση ανθρώπου – μηχανής. Για παράδειγμα, γραφικές διεπαφές χρήστη χρησιμοποιούν context πληροφορίες για να προσαρμόσουν menus στις προτιμήσεις του χρήστη και την κατάσταση του διαλόγου. Ένας καινούργιος τομέας, στον οποίο το context αποκτά όλο και μεγαλύτερη σημασία, είναι οι κινητοί Η/Υ.

Όταν οι άνθρωποι μιλάνε μεταξύ τους, έχουν την ικανότητα να χρησιμοποιούν έμμεσα πληροφορίες ανάλογα με την κάθε κατάσταση-συζήτηση για να αυξάνουν το εύρος της. Κατά αυτόν τον τρόπο δημιουργούν, μπορούμε να πούμε, ένα πλαίσιο (context) γύρω από κάθε επικοινωνιακή κατάσταση, θέτοντας τα όρια μέσα στα οποία κυμαίνονται οι πληροφορίες που ανταλλάσσονται. Αυτό γίνεται αυτόματα, χωρίς τις περισσότερες φορές να γίνεται αντιληπτό από κανένα από τα μέλη π.χ. ενός διαλόγου. Δυστυχώς δεν μπορούμε να πούμε το ίδιο για την επικοινωνία ανάμεσα σε έναν άνθρωπο και σε μία μηχανή, έναν Η/Υ. Η δυνατότητα ενός ανθρώπου να μεταφέρει ιδέες και σκέψεις σε έναν Η/Υ είναι ιδιαίτερα περιορισμένη και εξαρτάται κυρίως από τους μηχανισμούς εισαγωγής πληροφορίας του κάθε Η/Υ. Στον παραδοσιακό διαδραστικό προγραμματισμό, οι χρήστες έχουν έναν ιδιαίτερα φτωχό τρόπο να εισάγουν πληροφορίες στους Η/Υ. Βελτιώνοντας την πρόσβαση του Η/Υ στο πληροφοριακό πλαίσιο (context), θέτουμε τα όρια της συνομιλίας του με τον χρήστη. Με αυτόν τον τρόπο αυξάνουμε τον πλούτο της επικοινωνίας ανθρώπου-μηχανής και γίνεται δυνατό να παραχθούν πιο χρήσιμες υπολογιστικές λειτουργίες. [13].

2.1 **Context – Context Awareness**

Για να αντιληφθούμε την έννοια του context και να δώσουμε έναν συγκεκριμένο και εύστοχο ορισμό θα πρέπει να δούμε αρχικά πως οι ερευνητές έχουν προσπαθήσει να ορίσουν την έννοια αυτή. Πολλοί ορισμοί έχουν δοθεί οι περισσότεροι εκ των οποίων έχουν γίνει είτε με την απαρίθμηση παραδειγμάτων είτε διαλέγοντας κάποια συνώνυμα ώστε να περιγράψουν το context.

Οι Schilit και Theimer από το 1994 [36] εισάγοντας τα context aware συστήματα είχαν αναφερθεί στο context ως τοποθεσία, ταυτότητες των κοντινών ανθρώπων και αντικειμένων και αλλαγές σε αυτά τα αντικείμενα. Άλλοι ορισμοί που έχουν δοθεί χρησιμοποιώντας συνώνυμα αναφέρονται στο context ως το περιβάλλον μιας κατάστασης. Ο Schilit υποστηρίζει πως τρεις σημαντικές παράμετροι του context είναι: που είσαι, με ποιον είσαι μαζί και ποιοι πόροι βρίσκονται κοντά. Για τον λόγο αυτό χώρισε το context σε τρεις κύριες κατηγορίες [35]:

1. Υπολογιστικό Context: διαθέσιμοι επεξεργαστές, συσκευές προσβάσιμες για εισαγωγή και εμφάνιση από τον χρήστη, κοντινοί πόροι όπως εκτυπωτές, οθόνες, συνδεσιμότητα, κόστος υπολογισμού και επικοινωνίας, εύρος ζώνης, δυναμικότητα δικτύου κ.α.

2. Context Χρήστη: τοποθεσία χρήστη, συλλογή κοντινών ατόμων, προφίλ χρήστη και κοινωνικής κατάστασης κ.α.

3. Φυσικό Context: θερμοκρασία, θόρυβος, ποσοστό υγρασίας, φωτεινότητα, ταχύτητα κ.α.

Οι Chen και Kotz [9] ύστερα εισήγαγαν στην παραπάνω οπτική και το χρονικό context. Δηλαδή την μέρα, ώρα, μήνα, περίοδο του χρόνου κ.α. Αυτό μάς έδωσε την δυνατότητα άντλησης πληροφοριών από το ιστορικό του context, οι οποίες αποδεικνύονται ιδιαίτερα χρήσιμες σε αρκετές εφαρμογές.

Ο Pascoe [29] ορίζει το context ως το υποσύνολο των φυσικών και εννοιολογικών καταστάσεων ενδιαφέροντος μιας συγκεκριμένης οντότητας. Ένας πιο εύστοχος και γενικός ορισμός του context όμως στον οποίο και βασιστήκαμε δόθηκε από τον Dey το 2001 και αναφέρει πως:

Context είναι οποιαδήποτε πληροφορία μπορεί να χρησιμοποιηθεί για να χαρακτηρίσει την κατάσταση μιας οντότητας. Μια οντότητα είναι ένα άτομο, μια θέση ή ένα αντικείμενο που θεωρείται σχετικό με την διαδραστικότητα μεταξύ ενός χρήστη και μίας εφαρμογής, συμπεριλαμβανομένου του χρήστη και της εφαρμογής. Context είναι τυπικά η τοποθεσία, η ταυτότητα και η κατάσταση των ατόμων, των ομάδων και των υπολογιστικών και φυσικών αντικειμένων. [14]

Ο ορισμός αυτός μάς διευκολύνει να αντιληφθούμε την έννοια context λέγοντας πως αν μια πληροφορία μπορεί να χρησιμοποιηθεί για να χαρακτηρίσει μια κατάσταση ενός συμμετέχοντος σε μια δράση τότε αυτή η πληροφορία είναι context. Ας πάρουμε για παράδειγμα μία εφαρμογή όπου η οθόνη του κινητού χαμηλώνει την φωτεινότητά της ανάλογα με τον φωτισμό του εξωτερικού περιβάλλοντος και δύο πληροφορίες, την φωτεινότητά του χώρου και το ποσοστό υγρασίας. Χρησιμοποιώντας τον ορισμό βλέπουμε πως το ποσοστό υγρασίας δεν μπορεί να χρησιμοποιηθεί για τον χαρακτηρισμό της εν λόγω κατάστασης, συνεπώς δεν είναι context. Όπως είναι προφανές το ακριβώς αντίθετο συμβαίνει με την φωτεινότητα του χώρου, που πληροί τις προϋποθέσεις του ορισμού.

Εικόνα Ι: Το context από την οπτική του χρήστη

Χαρακτηριστικά context πληροφορίας:

Η πληροφορία που μας δίνει το context δεν είναι απόλυτη και διαθέτει κάποια κύρια χαρακτηριστικά. [19] , [48]

- ✓ Αναξιοπιστία:

Η πληροφορία που λαμβάνεται όντας δυναμική μπορεί πολύ γρήγορα να χάσει την χρηστικότητά της αφού προϊόντος του χρόνου θα έχει διαφοροποιηθεί σημαντικά. Ενδέχεται να είναι λανθασμένη αν δεν ανταποκρίνεται στην πραγματική κατάσταση, ασυνεπής λόγω των συχνών χρονικών αλλαγών, ή ελλιπής αν ορισμένα κομμάτια της πληροφορίας παραμένουν άγνωστα.

Ακόμα οι αισθητήρες που λαμβάνουν και παρέχουν την πληροφορία μπορεί να μην είναι ακριβείς λόγω φθοράς, περιβαλλοντικών παραμέτρων ή κακόβουλων ενεργειών. Τέλος λόγω ασταθών συνδέσεων είναι πιθανό μέρος ή και ολόκληρη η πληροφορία να παραμείνει άγνωστη αφού δεν θα υπάρχει μονοπάτι ανάμεσα στον αισθητήρα και την εφαρμογή.

- ✓ Εναλλακτικές αναπαραστάσεις:

Η πληροφορία που λαμβάνεται από τους αισθητήρες σε πολλές περιπτώσεις βρίσκεται σε άλλη αναπαράσταση από αυτή που τελικά θέλει η εφαρμογή να παρουσιάσει. Το κενό αυτό είναι μεγάλο και είναι απαραίτητο να γεφυρωθεί ώστε να μπορεί η εφαρμογή να χρησιμοποιηθεί πρακτικά. π.χ. ένας αισθητήρας που βρίσκει την γεωγραφική θέση σε συντεταγμένες ενώ ο χρήστης θέλει από την εφαρμογή να του παρουσιάσει την οδό στην οποία βρίσκεται. Όμως από εφαρμογή σε εφαρμογή μπορεί να ποικίλουν οι απαραίτητες αναπαραστάσεις. Για τον λόγο αυτό ένα context μοντέλο πρέπει να υποστηρίζει πολλαπλές αναπαραστάσεις της ίδιας πληροφορίας σε διαφορετικές μορφές και αφαιρετικά επίπεδα.

- ✓ Υψηλή αλληλεξάρτηση:

Το χαρακτηριστικό αυτό δείχνει πως πληροφορίες από διαφορετικές πηγές μπορούν να συνδυαστούν ώστε να εξαχθεί η σημαντική πληροφορία για το context. Ουσιαστικά μιλάμε για συσχετισμό και σύνθεση αξιόπιστων context πληροφοριών για να αποφανθούμε για αβέβαιες πληροφορίες. Π.χ. η κατάσταση της διάθεσης ενός ατόμου μπορεί να εξαχθεί από τις πληροφορίες θέσης, δραστηριότητάς του κ.α.

- ✓ Χρονική εξάρτηση:

Καταρχάς το context μπορεί να διακριθεί σε στατικό και δυναμικό[19]. Με τον όρο στατικό αναφέρουμε καταστάσεις οι οποίες δεν ενδέχεται να μεταβληθούν. π.χ. η ημερομηνία γέννησης. Στον αντίποδα δυναμικό ονομάζεται το context στο οποίο υπάρχει το ενδεχόμενο της μεταβολής της πληροφορίας. π.χ. τοποθεσία ενός ατόμου, θερμοκρασία, θόρυβος . Όπως γίνεται εύκολα αντιληπτό η πλειοψηφία των πληροφοριών που λαμβάνουμε σε ένα context aware σύστημα είναι δυναμική αφού τα διάχυτα αυτά συστήματα χαρακτηρίζονται από διαρκείς αλλαγές. Εκτός από τον διαχωρισμό της context πληροφορίας σε δυναμική και στατική, υπάρχουν και άλλες χρονικές διακρίσεις. Μπορεί το τρέχον context να εξαρτάται από παρελθοντική πληροφορία (μνήμη) ή να εξαρτάται από μελλοντική τιμή (αιτιατότητα).

Το context κατηγοριοποιείται σε δύο κύριες κατηγορίες[18]: Άμεσο context και Έμμεσο context, ανάλογα με τον τρόπο που αυτό αποκτήθηκε.

Το άμεσο context έχει απευθείας αποκτηθεί ή παραχωρηθεί από έναν προμηθευτή context ο οποίος μπορεί να είναι είτε μία εσωτερική πηγή, όπως π.χ. ένας πάροχος εσωτερικής υπηρεσίας, είτε μία εξωτερική πηγή όπως π.χ. ένας διακομιστής μετεωρολογικών πληροφοριών. Το άμεσο context μπορεί περαιτέρω να κατηγοριοποιηθεί σε ορισμένο context και αισθητό. Το αισθητό context αποκτάται από φυσικούς αισθητήρες όπως η θερμοκρασία ενός δωματίου από ένα θερμόμετρο ή από έναν εικονικό αισθητήρα, για παράδειγμα μία υπηρεσία διαδικτύου. Το ορισμένο context όπως μαρτυρά η ονομασία του ορίζεται από τον χρήστη, για παράδειγμα «αγαπημένο μάθημα».

Το έμμεσο context προέρχεται από την ανάλυση και τον συνδυασμό του άμεσου μέσω της διαδικασίας του context συλλογισμού. Π.χ. η τρέχουσα κατάσταση ενός ατόμου (λούζεται) μπορεί να εξαχθεί από την τοποθεσία του (μπάνιο), κατάσταση θερμοσίφωνα (on) , κατάσταση ντουζιέρας (νερό τρέχει) και κατάσταση πόρτας (κλειστή).

Μία άλλη διάκριση γίνεται ανάμεσα σε ενεργητικό και παθητικό context [9]. Στην πρώτη περίπτωση έχουμε αυτόματη προσαρμογή της εφαρμογής στο νέο context, δηλαδή αυτόματες εκτελέσεις εργασιών ή αλλαγή συμπεριφοράς με την λήψη της νέας πληροφορίας. π.χ. αναπροσαρμογή των οδηγιών σε ένα σύστημα GPS αν για οποιονδήποτε λόγο δεν ακολουθηθούν οι αρχικές οδηγίες. Στην δεύτερη περίπτωση η εφαρμογή μετά την ενημέρωσή της με το νέο context ενημερώνει τον χρήστη ή αποθηκεύει την πληροφορία για μελλοντική χρήση περιμένοντας κάποια εντολή.

Διαφορετικές κατηγορίες context έχουν διαφορετικά χαρακτηριστικά, δηλαδή το αισθητό context είναι κυρίως δυναμικό με διατηρησιμότητα από δευτερόλεπτα έως ώρες, ενώ το ορισμένο context είναι πιθανότερο να είναι στατικό. Από την κατηγοριοποίηση διαφόρων context πληροφοριών και γνωρίζοντας τα χαρακτηριστικά τους, μπορούμε να πραγματοποιήσουμε αποτελεσματικότερους context συλλογισμούς, πάνω σε διαφορετικά είδη context για να λύσουμε προβλήματα αναξιοπιστίας ή συγκρουόμενων πληροφοριών. Για παράδειγμα γνωρίζουμε ότι το ορισμένο context είναι πιο αξιόπιστο από το αισθητό context.

2.2 **Context Aware Προγραμματισμός**

Τι είναι όμως τα context-aware συστήματα και το context-aware computing;

Η πρόσφατη διάδοση των φθηνών, μικρών, και όλο και πιο ακριβών τεχνολογιών αισθητήρων έχουν οδηγήσει σε μια επανάστασης στον τομέα της πληροφορίας, όπου οι εφαρμογές που αλληλεπιδρούν με το φυσικό περιβάλλον, γίνονται ολοένα και πιο διαδεδομένες. Καθώς, ενώ αρχικά ένα πρώτο κύμα κινητών συσκευών βασίστηκε σε φορητούς γενικής χρήσης υπολογιστές οι οποίοι επικεντρώνονταν κυρίως σε διαφάνεια τοποθεσίας, τώρα ένα δεύτερο κύμα βασίζεται σε υπέρ-κινητές συσκευές και την συσχέτιση τους με την περιβάλλουσα κατάσταση χρήσης τους. Οι υπέρ-κινητές συσκευές αποτελούν μία νέα τάξη μικρών φορητών υπολογιστών, οι οποίες ορίζονται ως *υπολογιστικές συσκευές που λειτουργούν εν κινήσει,* και χαρακτηρίζονται από μία αλλαγή ως προς τον τρόπο χρήσης τους, από υπολογιστές γενικής χρήσης σε εργαλεία υποστήριξης ειδικής χρήσης. Παραδείγματα υπέρ-κινητών συσκευών αποτελούν το έξυπνα κινητά τηλέφωνα, τα tablets και οι φορητοί υπολογιστές, ενώ στο μέλλον ενδέχεται να ενταχθούν στην κατηγορία και τα ρολόγια [38].

Αυτή η αντίληψη του φυσικού περιβάλλοντος από τις εφαρμογές, καθώς και η χρήση των πληροφοριών που συλλέγονται για την επίτευξη των στόχων τους, είναι γνωστή ως context - awareness και μια σειρά από πολλά υποσχόμενες εφαρμογές έχουν κάνει την εμφάνισή τους σε αυτόν τον τομέα. Όπως φαίνεται οι context – aware τεχνολογίες αποκτούν ιδιαίτερη σημασία σε περιβάλλοντα κινητών συσκευών όπου το λειτουργικό περιβάλλον αλλάζει συνεχώς εξαιτίας της φορητότητας των συσκευών και των χαρακτηριστικών της ασύρματης τεχνολογίας επικοινωνιών. Σε τέτοια περιβάλλοντα, η αντίληψη του φυσικού περιβάλλοντος που περιβάλλει τον χρήστη και την αντίστοιχη συσκευή μπορεί να επιτρέψει στις εφαρμογές να ανταποκρίνονται έξυπνα σε μεταβλητό εύρος ζώνης, αναξιόπιστες συνδέσεις και για την εξοικονόμηση διαφορετικών συνδέσεων [5] Σε αρχικό στάδιο, ο προβληματισμός αυτός είχε αντιμετωπιστεί με μεθόδους αντίληψης της τοποθεσίας, όπως για παράδειγμα GPS. Βέβαια η τοποθεσία είναι μόνο η μία όψη του φυσικού περιβάλλοντος και όπως αποκαλύπτουν πρόσφατες ενδείξεις, συχνά χρησιμοποιείται ως μία προσέγγιση ενός πολυπλοκότερου context. Πέρα από την τοποθεσία, οι ειδικοί προβληματίζονται ως προς την αντίληψη άλλων χαρακτηριστικών που συνεισφέρουν στο context, ισχυριζόμενοι ότι όσο περισσότερα ξέρει μία υπέρ-κινητή

συσκευή ως προς τις συνθήκες χρήσης της, τόσο πιο πολύ μπορεί να φανεί χρήσιμη στον χρήστη της.

Μία από τις μεγαλύτερες προκλήσεις στον context – aware προγραμματισμό, και η οποία δεν έχει ακόμα γίνει προφανής, είναι ο σχεδιασμός και οι προδιαγραφές ενός ενδιάμεσου λογισμικού και υπηρεσιών για την υποστήριξη του προγραμματιστή τέτοιων εφαρμογών.

Ο όρος context-aware computing εισήχθη και συζητήθηκε για πρώτη φορά από τους Schilit και Theimer [36] το 1994 όπου το όρισαν ως λογισμικό που «προσαρμόζεται σύμφωνα με την τοποθεσία χρησιμοποίησης του, τους ανθρώπους και τα αντικείμενα που βρίσκονται κοντά του, όπως επίσης και στις αλλαγές που υφίστανται τα αντικείμενα αυτά προϊόντος του χρόνου.» Η πρώτη όμως ευρεία έρευνα πάνω στο context-aware computing πραγματοποιήθηκε το 1992 και ήταν το Olivetti Active Badge [45]. Από τότε αρκετοί ορισμοί έχουν δοθεί ορισμένους εκ των οποίων θα αναφέρουμε.

Οι Hull et al.[1] και Pascoe et al. [29] ορίζουν ως context-aware computing την ικανότητα των υπολογιστικών συσκευών να ανιχνεύουν και να αισθάνονται, να ερμηνεύουν και να αντιδρούν σε θέματα του τοπικού περιβάλλοντος του χρήστη και των ίδιων των συσκευών. Ο Dey[1]αναφέρει πως η γνώση του context του χρήστη οδηγεί σε αυτοματοποίηση του λειτουργικού συστήματος ενώ ο Salber [14]ορίζει ως context-aware την ικανότητα παροχής μέγιστης ευελιξίας μίας υπολογιστικής υπηρεσίας βάσει ανίχνευσης του context σε πραγματικό χρόνο. Ακόμα ο Ryan [30]ορίζει ως context-aware εφαρμογές, τις εφαρμογές εκείνες που λαμβάνουν ερεθίσματα από περιβαλλοντικούς αισθητήρες και επιτρέπουν στους χρήστες την επιλογή από ένα φάσμα φυσικών και λογικών επιλογών σύμφωνα με τις παρούσες ανάγκες και δραστηριότητές τους. Τέλος ο Brown [6] τις ορίζει ως εφαρμογές που παρέχουν αυτόματα πληροφορίες ή/και πράττουν σύμφωνα με το παρόν context του χρήστη όπως αυτό έχει ανιχνευθεί από τους αισθητήρες.

Ο πλέον εύστοχος όμως και γενικός ορισμός έχει δοθεί από τον Dey (2001) και αναφέρει πως:

Ένα σύστημα είναι context-aware εάν χρησιμοποιεί το context ώστε να παρέχει σχετικές πληροφορίες και/ή υπηρεσίες στον χρήστη, όπου η σχετικότητα εξαρτάται από την εργασία του χρήστη.[13]

Για παράδειγμα, η παρούσα κατάσταση μια πυξίδας σε ένα κινητό τηλέφωνο, η παρούσα κατάσταση του αισθητήρα του περιβάλλοντος φωτός μιας φορητής συσκευής ή η τοποθεσία και τα κοντινότερα ασύρματα δίκτυα με σκοπό μια αυτόματη αναπροσαρμογή (την αύξηση της φωτεινότητας, την ενεργοποίηση της ασύρματης σύνδεσης) ή την ενεργοποίηση μια συναφούς επιλογής (επισήμανση γεωγραφικών σημείων ενδιαφέροντος κοντά στον χρήστη ή εκτύπωση ενός εγγράφου στον πλησιέστερο εκτυπωτή).

2.3 **Διαχείριση Context**

Σε πρόσφατες μελέτες, ο όρος διαχείριση context συνήθως αναφέρεται στον συντονισμό και την παράλληλη διακίνηση και διανομή context πληροφοριών και δεδομένων. Οι αντίστοιχοι διακομιστές context ή άλλες υπηρεσίες διαχείρισης του context αποθηκεύουν τα context δεδομένα και παρέχουν πρόσβαση ώστε να είναι δυνατή η ανάκτηση, η σύγκριση και η ανανέωση της πληροφορίας. Μέχρι πρότινος, οι προγραμματιστές καθώς και οι υπόλοιποι επιστήμονες των Η/Υ όριζαν τους αισθητήρες και τον τρόπο συλλογής των δεδομένων, και καθόριζαν τις αντίστοιχες συμπεριφορές που θα έπρεπε να έχει το σύστημα, είτε άμεσα στον πηγαίο κώδικα, είτε έμμεσα μέσω άλλων εργαλείων.

Ο Winograd [47]περιγράφει τρία διαφορετικά μοντέλα διαχείρισης context για τον συγχρονισμό πολλαπλών διεργασιών και συστατικών στοιχείων:

Widgets: Προερχόμενο από τα ομώνυμα GUI στοιχεία, ένα widget είναι ένα συστατικό λογισμικού που παρέχει ένα κοινό περιβάλλον εργασίας για έναν αισθητήρα υλικού. Τα *Widgets* κρύβουν τις χαμηλού επιπέδου λεπτομέρειες ανίχνευσης και διευκολύνουν την ανάπτυξη των εφαρμογών εξαιτίας της δυνατότητας τους να επαναχρησιμοποιούνται. Λόγω της ενθυλάκωσης σε widgets είναι δυνατή η ανταλλαγή πληροφοριών ανάμεσα σε widgets που παρέχουν το ίδιο είδος context δεδομένων (π.χ. ανταλλαγή ενός widget ραδιοσυχνοτήτων με ένα widget κάμερας με σκοπό την συλλογή δεδομένων τοποθεσίας). Τα widgets ελέγχονται συνήθως από έναν widget manager. Η προσέγγιση των στενά συνδεδεμένων widget αυξάνει την αποτελεσματικότητα αλλά δεν είναι αξιόπιστη όσον αφορά τις βλάβες των στοιχείων.

Networked services**:** Αυτή η πιο ευέλικτη προσέγγιση μοιάζει με την αρχιτεκτονική του διακομιστή context δεδομένων. Αντί ενός γενικού widget manager, διαφορετικές μεταξύ τους τεχνικές αναζήτησης χρησιμοποιούνται για την εύρεση υπηρεσιών δικτύου. Αυτή η προσέγγιση παρέχει μεγαλύτερη αξιοπιστία από την μία, αλλά από την άλλη δεν είναι τόσο αποτελεσματική όσο μια αρχιτεκτονική widget, λόγω της πιθανής πολυπλοκότητας του δικτύου και του πλήθος των στοιχείων από τα οποία αυτό θα αποτελείται.

Blackboard model: Σε αντίθεση με την επικεντρωμένη στις διαδικασίες θεώρηση του widget, καθώς και του μοντέλου που βασίζεται στην αναζήτηση υπηρεσιών δικτύου, το ***Blackboard mode*** παρουσιάζει μία θεώρηση επικεντρωμένη στα δεδομένα. Σε αυτή την ασύμμετρη προσέγγιση οι διαδικασίες αναρτούν μηνύματα σε ένα κοινό μέσο, το επονομαζόμενο ***Blackboard***, και εγγράφονται σε αυτό ώστε να ενημερωθούν όταν κάποιο συγκεκριμένο γεγονός συμβεί. Τα πλεονεκτήματα αυτού του μοντέλου είναι η απλότητα που παρέχει στην πρόσθεση νέων πηγών context καθώς και η δυνατότητα που παρέχει για εύκολη διαμόρφωση αυτών. Ανασταλτικό παράγοντα όμως αποτελεί το γεγονός ότι υπάρχει η ανάγκη ενός κεντρικού server, ο οποίος θα φιλοξενεί το blackboard όπως και η έλλειψη αποτελεσματικότητας στην επικοινωνία καθώς απαιτούνται δύο λυκίσκοι για κάθε επικοινωνία

Τα context-aware συστήματα έχουν τα παρακάτω σημαντικά χαρακτηριστικά ως προς τις δυνατότητές τους για συλλογή, αποθήκευση, φιλτράρισμα και συνδυασμό των context πληροφοριών, όπως και εκτέλεση εντολών βάσει της πληροφορίας που τα δεδομένα μάς παρέχουν [48].

1. Συλλογή context. (Context acquisition)
2. Μέθοδος αναπαράστασης και αποθήκευσης (Context Handling)
3. Δυνατότητα context συλλογισμού και προσαρμογής στο context. (Context Reasoning)

2.3.1 **Συλλογή Context**

Η context πληροφορία αντλείται από μια σειρά από διαφορετικές πηγές πληροφοριών, όπως αισθητήρες θέσης, αισθητήρες καιρού ή κίνησης, οθόνες υπολογιστών και δικτύων, καθώς και την κατάσταση των υπολογιστικών ή ανθρώπινων υπηρεσιών. Ενώ τα ακατέργαστα δεδομένα του αισθητήρα μπορεί να είναι επαρκή για μερικές εφαρμογές, σε πολλές άλλες αυτά τα ακατέργαστα δεδομένα είναι που απαιτούνται ώστε να μετασχηματιστούν ή να συντηχθούν με άλλα δεδομένα του αισθητήρα για να είναι χρήσιμα. Συγκεντρώνοντας πολλές εισόδους αισθητήρων μπορούμε να αποκομίσουμε context υψηλότερου επιπέδου, με αποτέλεσμα οι εφαρμογές να μπορούν να προσαρμοστούν με μεγαλύτερη ακρίβεια.

Μια βασική πρόκληση στον τομέα της διεισδυτικής υπολογιστικής είναι η συλλογή ακατέργαστων δεδομένων από χιλιάδες διαφορετικούς αισθητήρες, η επεξεργασία των δεδομένων ώστε να αποτελέσουν context πληροφορίες, καθώς και τη διάδοση των πληροφοριών αυτών σε εκατοντάδες διαφορετικές εφαρμογές που τρέχουν σε χιλιάδες συσκευές παράλληλα με την κλιμάκωση σε μεγάλους αριθμούς πηγών, εφαρμογών, και χρηστών, εξασφαλίζοντας context πληροφορίες από μη εγκεκριμένες χρήσεις και με σεβασμό στην ιδιωτικότητα του ατόμου. [10]

Προμηθευτές context

Ονομάζονται όλες οι πηγές από τις οποίες μπορεί να προέρχεται το context. Σε ένα σύστημα παρέχονται πληροφορίες είτε κατευθείαν από τον χρήστη είτε από βάσεις δεδομένων, είτε κυρίως από αισθητήρες. Έτσι με βάση την πηγή προέλευσης της πληροφορίας γίνεται και η κατηγοριοποίηση του context που αναφέραμε παραπάνω.

Η σχεδίαση του συστήματος διαχείρισης των προμηθευτών, στην αρχιτεκτονική ενός context-aware συστήματος, περιλαμβάνει τα εξής θέματα:

Επικοινωνία προμηθευτών εφαρμογής: Είτε με την μέθοδο εγγραφής – ενημέρωσης, μέσω αποστολής γεγονότων είτε με την μέθοδο ερωταπαντήσεων (Q&A).

Συχνότητα ενημέρωσης: Καθορισμός συχνότητας ενημέρωσης της εφαρμογής από το σύστημα αισθητήρων, είτε με απευθείας ορισμό συχνότητας, είτε με φίλτρα γεγονότων. Τα φίλτρα καθορίζουν, σύμφωνα με τη ρύθμιση που έχει κάνει ο χρήστης, ποια γεγονότα είναι αρκετά «σημαντικά», ώστε να ενημερωθεί η εφαρμογή.

Εύρεση κατανεμημένων προμηθευτών, σε διάχυτο περιβάλλον: Σε διάχυτα περιβάλλοντα έχουμε δύο περιπτώσεις κατανομής προμηθευτών ως προς την εφαρμογή. Στην πρώτη περίπτωση δεν έχουμε ορατούς προμηθευτές, οπότε η εφαρμογή πρέπει να αναζητήσει σε ένα κατανεμημένο δίκτυο προμηθευτών και να ανιχνεύσει τους κατάλληλους. Στην δεύτερη περίπτωση η εφαρμογή έχει ορατούς προμηθευτές, αλλά δεν ξέρει ποιος από όλους θα της παρέχει τη ζητούμενη context πληροφορία. Ανεξάρτητα από την περίπτωση πρέπει να υπάρχουν κατάλληλοι μηχανισμοί εύρεσης και επικοινωνίας με τους προμηθευτές του context.

Είδος Context Πληροφορίας	**Κοινοί Αισθητήρες**
Φως	Φωτοδίοδοι, Αισθητήρες χρωμάτων, υπέρυθροι, υπεριώδεις αισθητήρες, κλπ
Οπτική	Διάφορα είδη καμερών
Επιτάχυνση	Φωτοκύτταρα, Διακόπτες Υδραργύρου, Επιταχυνσιόμετρα,
Ακουστική	Μικρόφωνα
Τοποθεσία	GPS, GSM
Επαφή	Ενσωματωμένα αισθητήρες επαφής
Θερμοκρασία	Θερμόμετρα
Φυσικά Χαρακτηριστικά	Βιομετρητές που καταγράφουν αντίσταση δέρματος, αρτηριακή πίεση κλπ

Εικόνα ΙΙ : Παράδειγμα προμηθευτών Context Δεδομένων

Επιπρόσθετα, η μέθοδος με την οποία γίνεται η συλλογή των context δεδομένων είναι ιδιαιτέρως σημαντική στη σχεδίαση context-aware συστημάτων γιατί προκαθορίζει, τουλάχιστον σε κάποιο βαθμό, το αρχιτεκτονικό στυλ του συστήματος. Ο Chen [9] παρουσιάζει τρεις διαφορετικές προσεγγίσεις όσον αφορά τον τρόπο απόκτησης context πληροφοριών.

Άμεση πρόσβαση αισθητήρα: Αυτή η προσέγγιση χρησιμοποιείται πιο συχνά σε συσκευές που έχουν ενσωματωμένους αισθητήρες. Το λογισμικό του χρήστη μαζεύει τις επιθυμητές πληροφορίες απευθείας από τους αισθητήρες, δηλαδή δεν υπάρχει ενδιάμεσο επίπεδο για την απόκτηση και την επεξεργασία των δεδομένων του αισθητήρα. Οι οδηγοί για τους αισθητήρες είναι ενσωματωμένοι στην εφαρμογή και για αυτόν τον λόγο αυτή η στενά συζευγμένη μέθοδος χρησιμοποιείται μόνο σε σπάνιες περιπτώσεις. Συνεπώς, δεν είναι κατάλληλη για κατανεμημένα συστήματα λόγω του ότι στερείται το συστατικό που θα την κάνει ικανή να διαχειριστεί ταυτόχρονα προσπελάσεις δεδομένων πολλαπλών αισθητήρων.

Υποδομή Ενδιάμεσου Λογισμικού: Ο μοντέρνος σχεδιασμός λογισμικού χρησιμοποιεί μεθόδους ενθυλάκωσης για τον διαχωρισμό π.χ., επιχειρηματική λογικής και γραφικών διεπαφών χρήστη. Η προσέγγιση ενδιάμεσου λογισμικού εισάγει μια πολυεπίπεδη

αρχιτεκτονική στα context-aware συστήματα με σκοπό την απόκρυψη χαμηλού επιπέδου λεπτομέρειες ανίχνευσης. Σε σύγκριση με την άμεση πρόσβαση αισθητήρα η τεχνική αυτή διευκολύνει την επεκτασιμότητα αφού ο κωδικός του χρήστη δεν τροποποιείται πια, ενώ παράλληλα απλοποιεί την επαναχρησιμοποίηση του εξαρτώμενου από το υλικό ανίχνευσης, λόγω της αυστηρής ενθυλάκωσης.

Διακομιστής Context: Το επόμενο λογικό βήμα είναι να επιτρέπουμε σε πολλαπλούς χρήστες να έχουν πρόσβαση σε απομακρυσμένες πηγές δεδομένων. Αυτή η κατανεμημένη προσέγγιση επεκτείνει την αρχιτεκτονική που βασίζεται στο ενδιάμεσο λογισμικό με την εισαγωγή ενός συστατικού πρόσβασης απομακρυσμένης διαχείρισης. Οι λειτουργίες συγκέντρωσης των δεδομένων που γίνονταν από τους αισθητήρες μεταφέρεται τώρα στον διακομιστή context για τη διευκόλυνση της ταυτόχρονης πολλαπλής προσπέλασης. Εκτός από την επαναχρησιμοποίηση των αισθητήρων, η χρήση ενός διακομιστή πλαισίου έχει το πλεονέκτημα της ανακούφισης των χρηστών από εντατικές διαδικασίες που αναλώνουν πόρους. Αν σκεφτούμε επίσης πως η πλειοψηφία των τερματικών συσκευών που χρησιμοποιούνται σε context-aware συστήματα είναι κινητές και φορητές συσκευές με περιορισμένη υπολογιστική ισχύ, χωρητικότητα στο δίσκο κλπ., αυτό αποτελεί μία σημαντική πτυχή. Σε αντίθεση κατά τον σχεδιασμό ενός context-aware συστήματος που βασίζεται στην αρχιτεκτονική χρήστη – διακομιστή πρέπει κανείς να προσέξει για κατάλληλα πρωτόκολλα, επιδόσεις δικτύου, ποιότητα των παρεχόμενων υπηρεσιών κ.α.[3].

Φυσικοί Αισθητήρες : Οι αισθητήρες πιο συχνής χρήσης είναι οι φυσικοί αισθητήρες. Στις μέρες μας υπάρχουν διαθέσιμοι αισθητήρες ικανοί να συλλάβουν σχεδόν κάθε μορφή φυσικού δεδομένου.

Οπτικοί Αισθητήρες: Οι οπτικοί αισθητήρες συλλέγουν πληροφορίες context (πλαισίου) που πηγάζουν από λογισμικές εφαρμογές και υπηρεσίες. Για παράδειγμα, είναι δυνατό να εξακριβώσεις την τοποθεσία ενός εργαζόμενου, όχι μόνο χρησιμοποιώντας συστήματα εντοπισμού (φυσικοί αισθητήρες) αλλά και μέσω οπτικών αισθητήρων, π.χ. ένα ηλεκτρονικό ημερολόγιο, emails κλπ. Άλλα παραδείγματα context χαρακτηριστικών που μπορούν να συλλεγούν από οπτικούς αισθητήρες είναι: ενέργειες του χρήστη από κινήσεις του ηλεκτρονικού ποντικιού ή του πληκτρολογίου.

Λογικοί Αισθητήρες: Αυτοί οι αισθητήρες κάνουν χρήση διαφόρων πηγών πληροφοριών και συνδυάζουν φυσικούς και οπτικούς αισθητήρες με πρόσθετες πληροφορίες από βάσεις δεδομένων ή άλλες πηγές με σκοπό την επίλυση συνθετότερων

ζητημάτων. Για παράδειγμα, ένας λογικός αισθητήρας μπορεί να δομηθεί ώστε να εντοπίζει τη θέση του χρήστη αναλύοντας τις εισόδους σε έναν υπολογιστή παράλληλα με μία βάση δεδομένων για τις πληροφορίες τοποθεσίας του Η/Υ.[3].

2.3.2 **Context Συλλογισμός**

Οι επιστήμονες των Η/Υ οραματίζονται ένα υπολογιστικό περιβάλλον το οποίο θα έχει πρόσβαση σε πληροφορίες ενός πλήθους αισθητήρων και εν γένει υπολογιστικών συσκευών, τόσο μεγάλου ώστε η σημασία της κάθε πηγής πληροφορίας θα χάνεται στο συνολικό υπόβαθρο που θα έχει δημιουργηθεί. Με αυτό τον τρόπο οι άνθρωποι θα μπορούν να συγκεντρώνονται και να αφιερώνουν το μεγαλύτερο κομμάτι του χρόνου τους στις καθημερινές τους δραστηριότητες, παρά να ασχολούνται με τις υποκείμενες τεχνολογίες. Για να μπορέσουμε να πραγματοποιήσουμε αυτό το όραμα, θα πρέπει να μεταμορφώσουμε και να εξελίξουμε τις παλιές αφυής και context αδιάφορες μηχανές και τεχνολογίες, σε έξυπνες, και εύκολα προγραμματίσιμες συστάδες μηχανών. Παρόλα αυτά, η αντίληψη και δυνατότητα επεξεργασίας μεγάλου πλήθους context πληροφοριών μπορεί πολλές φορές να οδηγήσει σε ανακρίβειες όσον αφορά τόσο την συλλογή όσο και την «κατανόηση» συγκρουόμενων δεδομένων από αβέβαιους φυσικούς κόσμους. Για τον παραπάνω λόγο, διαφορετικοί τύποι οντοτήτων (λογισμικών αντικειμένων) ενός περιβάλλοντος θα πρέπει να είναι σε θέση να εκλογικεύσουν την επεξεργασία τέτοιων δεδομένων. Αυτή η λειτουργία περιλαμβάνει οντότητες που θα μπορούν να αντιλαμβάνονται και να συνδυάζουν αυτές τις αβέβαιες context πληροφορίες, καθώς και να βγάζουν τα κατάλληλα συμπεράσματα με την χρήση του κατάλληλου context συλλογισμού. Επίσης απαιτείται η ύπαρξη και η δημιουργία εφαρμογών που θα μπορούν να προσαρμόζουν την συμπεριφορά τους ανάλογα με τα υπάρχοντα δεδομένα. Η ύπαρξη ενός κοινού μοντέλου το οποίο θα ασχολείται με την αοριστία και την ανακρίβεια των δεδομένων, θα διευκολύνει τούς προγραμματιστές στη δημιουργία νέων υπηρεσιών και εφαρμογών σε τέτοια περιβάλλοντα καθώς και στην επαναχρησιμοποίηση αυτών των μεθόδων συλλογισμού και εξαγωγής συμπερασμάτων πάνω στην αβεβαιότητα. [32]

<u>Δυνατότητα context συλλογισμού.</u>

Ως επί το πλείστον οι context πληροφορίες που παρέχονται από τους προμηθευτές είναι χαμηλού εννοιολογικού επιπέδου. Δηλαδή μιλάμε κυρίως για αριθμητικές τιμές και

τεχνικές ή ειδικές ακατέργαστες πληροφορίες, οι οποίες από μόνες τους δεν καθίστανται χρήσιμες. Για το λόγο αυτό ένα context-aware σύστημα επιβάλλεται να έχει μηχανισμούς ερμηνείας του context μετατρέποντας το σε μορφή που μπορεί να χρησιμοποιηθεί από την εφαρμογή. Ακόμα και έτσι όμως υπάρχουν περιπτώσεις κατά τις οποίες και η κατεργασμένη πληροφορία ενός προμηθευτή δεν είναι ιδιαίτερα χρήσιμη από μόνη της. Σε αυτές τις περιπτώσεις πρέπει το σύστημα να προχωρήσει σε συλλογή πληροφοριών, των οποίων ο συνδυασμός θα οδηγεί σε ένα συμπέρασμα, χρήσιμο για το χρήστη. Το ζητούμενο, δηλαδή, είναι ο συνδυασμός του χαμηλού εννοιολογικού επιπέδου context και η εξαγωγή υψηλότερου επιπέδου σημασιολογικής πληροφορίας. Τις περισσότερες φορές το context υψηλού εννοιολογικού επιπέδου δεν μπορεί να αποκτηθεί άμεσα αλλά συμπεραίνεται, συνδυάζοντας την πληροφορία που παρέχουν οι προμηθευτές μέσω κάποιου context συλλογισμού, μιας διαδικασίας εξαγωγής συμπερασμάτων. Για παράδειγμα:

Είδος Δεδομένων	**Τιμή**
Ακατέργαστη Τιμή	-5
Επεξεργασμένη Τιμή	-5°C
Ερμηνεία	Κάνει Κρύο
Context Συλλογισμός (συννεφιά, βαρομετρικό, υγρασία)	Θα χιονίσει.

Εικόνα ΙΙΙ: Παράδειγμα Context Συλλογισμού

Οι παραπάνω διαδικασίες context συλλογισμών προϋποθέτουν ευφυΐα στο σύστημα. Η ευφυΐα ενσωματώνεται συνήθως με κανόνες δημιουργώντας ένα έμπειρο σύστημα[1].

[1] "Ένα έμπειρο σύστημα θεωρείται η ενσωμάτωση μέσα σε έναν υπολογιστή μιας βασισμένης στη γνώση συνιστώσας από την ικανότητα ενός ειδικού, με μια τέτοια μορφή ώστε το σύστημα να μπορεί

Προσαρμογή στο context

Δεν είναι μόνο ο χρήστης που μπορεί να ενδιαφέρεται για τα δεδομένα ή και τα πιο σύνθετα συμπεράσματα, π.χ. ενός context συλλογισμού από ένα τέτοιο σύστημα προμηθευτών. Οι εφαρμογές μπορούν να αξιοποιήσουν αυτή τη γνώση προσαρμόζοντας τη συμπεριφορά τους ώστε να ενσωματωθούν πιο ομαλά στο περιβάλλον του χρήστη. Αντί να παρέχει μια ενιαία υπηρεσία, ανεξάρτητα από τις συνθήκες του χρήστη, να μπορεί το context-aware σύστημα να προσαρμοστεί στην τρέχουσα κατάσταση. Για παράδειγμα, η προσαρμογή της συμπεριφοράς μιας συσκευής- ενός συστήματος για ένα συγκεκριμένο χρήστη (όπου το context είναι η τοποθεσία του χρήστη), δυναμώνοντας την φωτεινότητα μιας οθόνης κατά τη διάρκεια της μέρας ή αλλάζοντας σε φθηνότερο επικοινωνιακό πάροχο, όταν αυτό είναι δυνατό.

2.3.3 **Context διαχείριση**

Πολλές φορές τρέχουσες τιμές context είναι απαραίτητες για τις υπηρεσίες του συστήματος στο μέλλον, έτσι είναι αναγκαία η αποθήκευσή τους σε κατάλληλη μορφή για μελλοντική επεξεργασία και χρήση. Ακόμα είναι πολύ σημαντικό το απαραίτητο φιλτράρισμα της όποιας πληροφορίας προορίζεται για αποθήκευση, ώστε να επιτυγχάνεται η επιθυμητή εξοικονόμηση αποθηκευτικού χώρου και εν γένει υπολογιστικών πόρων. Τέλος ο τρόπος διαφορετικής αναπαράστασης του κάθε τύπου συλλεχθείσας πληροφορίας αποτελεί άλλο ένα κομμάτι των λειτουργιών διαχείρισης των δεδομένων.

Παραπάνω είδαμε μερικές από τις διαδικασίες που ανήκουν στην κατηγορία της διαχείρισης της context πληροφορίας. Είτε η πληροφορία προέρχεται απευθείας από τους προμηθευτές, είτε είναι απόρροια συνδυασμού δεδομένων και context συλλογισμού, ο τρόπος με τον οποίον αυτή θα διαχειριστεί από το σύστημα ή την συσκευή είναι πολύ σημαντικός τόσο για την άμεση αξιοποίησή της, όσο και για την δυνατότητα μελλοντικής ανάκτησης και επαναχρησιμοποίησης της.

Για αυτούς τους λόγους η διαχείριση context πληροφοριών απαιτεί ένα μοντέλο που θα ορίζει/φιλτράρει και θα αποθηκεύει το context, ενώ θα είναι αρκετά γενικό, περιλαμβάνοντας όλα τα γνωρίσματα της πληροφορίας και παρέχοντας μία ευέλικτη μέθοδο αναπαράστασης.

Γενικότερα, για τη διαχείριση του context ο εκάστοτε προγραμματιστής της εφαρμογής μπορεί να επιλέξει την τεχνική που είναι πιο εύκολη και πιο χρήσιμη να εφαρμοστεί, ανάλογα με την περίπτωση. Δύο κύριοι τρόποι διαχείρισης του συλλεγμένου context είναι:

Σύνδεση των οδηγών των αισθητήρων απευθείας στην εφαρμογή

Σε ορισμένες εφαρμογές, που ακολουθούν αυτόν τον τρόπο διαχείρισης οι οδηγοί των αισθητήρων είναι απευθείας συνδεδεμένοι με την εφαρμογή. Σε αυτή την περίπτωση οι σχεδιαστές των εφαρμογών είναι αναγκασμένοι να γράψουν κώδικα ικανό να χειρίζεται τις λειτουργίες των αισθητήρων, χρησιμοποιώντας οποιοδήποτε πρωτόκολλο αυτοί τους υπαγορεύουν. Δύο προβλήματα εντοπίζονται σε αυτή την τεχνική. Το πρώτο είναι πως καθίσταται ιδιαίτερα δύσκολη η ανάπτυξη μιας context aware εφαρμογής, καθώς απαιτείται από τους προγραμματιστές να αντιμετωπίσουν την πιθανή πολυπλοκότητα της διαδικασίας απόκτησης του context. Το δεύτερο είναι πως αυτή την τεχνική δεν υποστηρίζει καλές και εύχρηστες τεχνολογίες λογισμικού. Δεν επιβάλει διαχωρισμούς ανάμεσα στην σημασιολογία της εφαρμογής και τις χαμηλού επιπέδου λειτουργίες της απόκτησης του context από τους αισθητήρες. Το γεγονός αυτό, οδηγεί στην απώλεια της γενικότητας καθιστώντας αδύνατες, την επαναχρησιμοποίηση των αισθητήρων από άλλες εφαρμογές, καθώς και την ταυτόχρονη χρησιμοποίηση τους από πολλαπλές εφαρμογές.

Χρησιμοποίηση διακομιστών και απόκρυψη επιμέρους λειτουργιών των αισθητήρων.

Σε αυτή την περίπτωση ένας διακομιστής έχει σχεδιαστεί έτσι ώστε να δειγματοληπτεί τα δεδομένα που λαμβάνονται από τους αισθητήρες. Οι διακομιστές αυτοί αποσπούν τις λειτουργίες των αισθητήρων από την εφαρμογή. Αυτή η τεχνική δίνει λύση και στα δύο προαναφερθέντα προβλήματα. Πλέον οι προγραμματιστές δεν είναι αναγκασμένοι να απασχολούνται με τις ξεχωριστές λειτουργίες του κάθε αισθητήρα. Η χρήση των διακομιστών ξεχωρίζει τώρα την σημασιολογία της εφαρμογής από τις χαμηλού επιπέδου λειτουργίες του αισθητήρα, κάνοντας ευκολότερη στον σχεδιαστή της εφαρμογής την ανάπτυξη μιας context-aware εφαρμογή και επιτρέποντας σε πολλαπλές εφαρμογές την χρησιμοποίηση ενός και μόνο διακομιστή.

Παρόλα αυτά αυτή η τεχνική εισάγει δύο καινούρια προβλήματα. Πρώτον, οι εφαρμογές που χρησιμοποιούν αυτούς τους διακομιστές πρέπει να είναι ενεργητικές, ζητώντας context πληροφορία όποτε χρειάζεται μέσω ενός μηχανισμού δειγματοληψίας. Το βάρος πέφτει πλέον στην εφαρμογή να αποφασίσει πότε υπάρχουν αλλαγές στο context και πότε αυτές οι αλλαγές παρουσιάζουν ενδιαφέρον. Δεύτερον, αυτοί οι διακομιστές αναπτύσσονται ανεξάρτητα για κάθε αισθητήρα ή τύπο αισθητήρα. Κάθε διακομιστής διατηρεί μια διαφορετική διεπαφή για κάθε εφαρμογή ώστε να αλληλεπιδρά με αυτή. Αυτό απαιτεί από την εφαρμογή να αντιμετωπίζει κάθε διακομιστή με διαφορετικό τρόπο, αρκετά όμοια με το να αντιμετωπίζει διαφορετικούς αισθητήρες. Το γεγονός αυτό μπορεί να επηρεάσει την ικανότητα της εφαρμογής να ξεχωρίζει την σημασιολογία της εφαρμογής από την απόκτηση του context.

Ιδανικά, θα θέλαμε να διαχειριζόμαστε το context με τον ίδιο τρόπο που διαχειριζόμαστε την είσοδο των δεδομένων από έναν χρήστη. Δηλαδή μέσω των εργαλείων των διεπαφών του χρήστη, με την υποστήριξη των οποίων οι σχεδιαστές των εφαρμογών διαχειρίζονται τα δεδομένα που εισάγει ένας χρήστης. Αυτό καθώς παρέχουν ένα σημαντικό επίπεδο αοριστίας ώστε να επιτρέπουν στους προγραμματιστές να μην ανησυχούν για τον τρόπο με τον οποίο αποκτήθηκε το context. Η αοριστία αυτή ονομάζεται widget abstraction ή interactor. Η αοριστία του widget προσφέρει πολλά πλεονεκτήματα τόσο στην είσοδο από πληκτρολόγιο ή ποντίκι, όσο και στην είσοδο από στυλό ή ομιλία, καθώς και με τις αντισυμβατικές συσκευές εισόδου που χρησιμοποιούνται στην εικονική πραγματικότητα. Με αυτόν τον τρόπο διευκολύνεται ο διαχωρισμός της σημασιολογίας της εφαρμογής από τις χαμηλού επιπέδου λειτουργίες της διαχείρισης της πληροφορίας. Για παράδειγμα, μία εφαρμογή δεν χρειάζεται να διαφοροποιεί την λειτουργία της αν χρησιμοποιούμε αντί για το ποντίκι, ένα στυλό - laser για να δείχνουμε. Ακόμα πρέπει να υποστηρίζεται η επαναχρησιμοποίηση του ιδίου widget επιτρέποντας σε πολλαπλές εφαρμογές να δημιουργούν τα δικά τους στιγμιότυπα του. Επίσης πρέπει να περιέχεται όχι μόνο ένας μηχανισμός δειγματοληψίας αλλά και ένας μηχανισμός ειδοποίησης, ώστε οι εφαρμογές να μπορούν να αποκτούν την πληροφορία εισόδου ενώ αυτή εμφανίζεται. Τέλος, σε ένα δοσμένο εργαλείο, όλα τα widgets έχουν κοινή εξωτερική διεπαφή. Αυτό σημαίνει πως η εφαρμογή μπορεί να συμπεριφέρεται σε όλα τα widgets με παρόμοιο τρόπο, μην έχοντας να αντιμετωπίσει τις πιθανές διαφορές που θα υπάρχουν ανάμεσα στο καθένα ξεχωριστά.

Σε προηγούμενα συστήματα έχουν αναπτυχθεί τεχνολογίες οι οποίες διαχειρίζονταν τις context πληροφορίες, όπως και τα οποιοδήποτε δεδομένα εισόδου (2,18), χρησιμοποιώντας διακομιστές για την υποστήριξη, τόσο των μηχανισμών δειγματοληψίας όσο και των αντίστοιχων ειδοποίησης. Ειδικότερα η λειτουργία ειδοποίησης απάλλασσε την εφαρμογή από την υποχρέωση να δειγματοληπτεί, μέσω του διακομιστή, ώστε να αποφασίζει πότε έχει πραγματοποιηθεί μια σημαντική αλλαγή των context δεδομένων. Παρόλα αυτά, τα συστήματα αυτά αντιμετώπιζαν πρόβλημα ως προς τον σχεδιασμό εξειδικευμένων διακομιστών, λόγω της έλλειψης μιας κοινής διεπαφής. Έτσι η κάθε εφαρμογή συνδεόταν με τον κάθε διακομιστή με διαφορετικό τρόπο. Αυτό είχε ως αποτέλεσμα να χρησιμοποιείται ένας ελάχιστος αριθμός διακομιστών. Αναλύοντας την εφαρμογή των widget για την διαχείριση του context, εξηγήσαμε πως τα context widget παρέχουν τα ίδια πλεονεκτήματα όπως τα αντίστοιχα widget της γραφικής διεπαφής του χρήστη. Από τα παραπάνω είναι προφανές πως η χρήση της αοριστίας που προσφέρουν τα widget αποτελεί ένα βήμα μπροστά ως προς την αξιοποίηση των context πληροφοριών από τις εφαρμογές. Παρά ταύτα υπάρχουν ακόμα διαφορές στον τρόπο με τον οποίο οι context πληροφορίες συλλέγονται και επεξεργάζονται, οι οποίες χρήζουν την δημιουργία ενός ενδιάμεσου λογισμικού υποστήριξης της αρχιτεκτονικής των widget.

2.4 **Λογισμικό πλαίσιο – Πλατφόρμα λογισμικού**

2.4.1 **Πλαίσιο**

Το πλαίσιο (framework) είναι ένα περιβάλλον λογισμικού που είναι σχεδιασμένο ώστε να απλοποιεί την ανάπτυξη των εφαρμογών και τη διαχείριση του συστήματος για ένα εξειδικευμένο πεδίο εφαρμογής. Ένα πλαίσιο λογισμικού μπορεί να είναι μια συλλογή από προγράμματα υποστήριξης, μια γλώσσα κειμένου, βιβλιοθήκες κώδικα και οποιοδήποτε άλλο λογισμικό που βοηθάει στην ανάπτυξη και στη σύνδεση διαφορετικών συστατικών ενός έργου λογισμικού. Ένα πλαίσιο ορίζεται από ένα API, μια διεπαφή χρήστη και ένα σετ εργαλείων. Μπορεί επίσης να έχει υπηρεσίες ενδιάμεσου λογισμικού ιδιωτικού πλαισίου πέραν αυτών που εισάγει από άλλα προϊόντα.

Κάθε βιβλιοθήκη ενός πλαισίου, τού παρέχει μια συγκεκριμένη λειτουργία. Έτσι όσο περισσότερες βιβλιοθήκες περιλαμβάνει ένα πλαίσιο τόσο περισσότερες λειτουργίες παρέχει. Οι λειτουργίες που παρέχονται από ένα πλαίσιο εμφανίζονται μέσα από τα API's.

Η σχεδίαση ενός πλαισίου είναι ιδιαίτερα ευέλικτη και σκοπεύει στην επαναχρησιμοποίηση του. Διευκολύνει τον σχεδιασμό και την ανάπτυξη του λογισμικού, επιτρέποντας σε σχεδιαστές και προγραμματιστές, να αφιερώσουν περισσότερο χρόνο στην ουσιαστική ανάπτυξη των απαιτήσεων.

Η επιλογή του κατάλληλου πλαισίου συνήθως σημαίνει για έναν προγραμματιστή να ισορροπήσει ανάμεσα:

- ✓ Στον αριθμό των λειτουργιών που αυτό παρέχει
- ✓ Στην δυνατότητα ευελιξίας που αυτό τους επιτρέπει

Το πλαίσιο στο οποίο βασίστηκε και με την βοήθεια του οποίου αναπτύχθηκε η εφαρμογή θα αποτελέσει μέρος της εφαρμογής. Δηλαδή εάν κάποιος θελήσει να χρησιμοποιήσει την εφαρμογή, θα πρέπει να έχει διαθέσιμο και το αντίστοιχο πλαίσιο. Παραδείγματα πλαισίων (framework) αποτελούν: Wiki, BlogSpot, Java spring, Struts.

2.4.2 **Πλατφόρμα**

Μία πλατφόρμα μπορεί να είναι αρχιτεκτονική υλικού με κάποιο λογισμικό πάνω στο οποίο λειτουργούν άλλες εφαρμογές. Η αρχιτεκτονική υλικού είναι η βάση οποιουδήποτε λογισμικού. Πιο συγκεκριμένα, η πλατφόρμα υλικού αναφέρεται στον επεξεργαστή και άλλες συσκευές υλικού που υπάρχουν σε οποιαδήποτε υπολογιστική μηχανή. Πάνω σε αυτό το υλικό έχουμε το λειτουργικό σύστημα και άλλα προγράμματα εφαρμογών. Στην πλατφόρμα λογισμικού για παράδειγμα, το λειτουργικό σύστημα σχηματίζει την βάση ώστε να λειτουργήσουν όλα τα άλλα λογισμικά

Οι πλατφόρμες υποστηρίζουν τα προγράμματα εφαρμογών παρέχοντας ορισμένα από τα παρακάτω χαρακτηριστικά και λειτουργίες: πολυεπεξεργασία, διαχείριση μνήμης, πρόσβαση στον σκληρό δίσκο και στους φακέλους, δικτύωση, ασφάλεια κ.α.

Παραδείγματα δημοφιλών πλατφορμών που χρησιμοποιούνται: Λειτουργικά Λογισμικά, Προγραμματιστικές Γλώσσες, Μεταφραστές, Java εικονικές μηχανές κ.α.

2.5 **Ενδιάμεσο Λογισμικό**

Σε ένα διαμοιρασμένο υπολογιστικό σύστημα το ενδιάμεσο λογισμικό ορίζεται ως το στρώμα λογισμικού που βρίσκεται ανάμεσα στο λειτουργικό σύστημα και τις εφαρμογές σε κάθε τοποθεσία του συστήματος [25] .

Με την βοήθεια του ενδιάμεσου λογισμικού παρέχεται μια κοινή προγραμματιστική αοριστία στα διεσπαρμένα συστήματα. Για να το καταφέρει αυτό παρέχει υψηλότερα επίπεδα προγραμματισμού από τα APIs όπως «sockets» που παρέχονται από το λειτουργικό σύστημα. Αυτό μειώνει δραστικά το βάρος από τους προγραμματιστές των εφαρμογών αφού τους απαλάσσει από τον κουραστικό και επιρρεπή σε λάθη προγραμματισμό.

Όπως χαρακτηριστικά αναφέρει ο David E. Bakken [2] ο κλασσικός ορισμός ενός λειτουργικού συστήματος είναι "*το λογισμικό που κάνει το υλικό χρησιμοποιήσιμο"*. Με βάση τον παραπάνω ορισμό προχωράει και ορίζει το ενδιάμεσο λογισμικό ως «το λογισμικό που κάνει ένα διεσπαρμένο σύστημα προγραμματίσιμο».

Επηρεαζόμενες από την ανάπτυξη των δικτυακών εφαρμογών οι τεχνολογίες ενδιάμεσου λογισμικού γίνονται όλο και πιο σημαντικές. Χρήστες αλληλεπιδρούν μέσα από ποικιλία συσκευών, τα χαρακτηριστικά και οι δυνατότητες των οποίων εκτείνονται σε ένα μεγάλο εύρος. Ανάμεσα σε έναν υπολογιστή υψηλής απόδοσης, ένα smartphone και ένα tablet οι διαφορές στο εύρος ζώνης (bandwidth), στην τοπική ισχύ επεξεργαστή (local processing power) και στην ευκρίνεια της οθόνης (screen capacity) είναι εξαιρετικά μεγάλες.

Για το λόγο αυτό οι διάφορες τεχνολογίες ενδιάμεσου λογισμικού καλούνται να καλύψουν ένα μεγάλο εύρος λειτουργικών συστημάτων, το οποίο περιλαμβάνει από λογισμικά Η/Υ και τηλεοράσεων μέχρι τα αντίστοιχα κινητών τηλεφώνων και αυτοκινήτων καθώς και να διευκολύνει την μεταξύ τους επικοινωνία σε περιπτώσεις δικτύων.

Το ενδιάμεσο λογισμικό μπορεί να διακριθεί στις εξής κατηγορίες [22]:

- Βιβλιοθήκες
- Πλαίσια
- Εργαλεία
- Υποδομές

Βιβλιοθήκη είναι ένα γενικευμένο σύνολο από υλοποιημένους αλγορίθμους, σχετικούς με κάποιο θέμα. Οι βιβλιοθήκες στοχεύουν αποκλειστικά στην επαναχρησιμοποίηση κώδικα.

Τα *πλαίσια* στοχεύουν κυρίως στην επαναχρησιμοποίηση αρχιτεκτονικού σχεδιασμού. Παρέχουν μια βασική αρχιτεκτονική δομή, για την ανάπτυξη συγκεκριμένης κατηγορίας εφαρμογών. Επιπλέον ένα πλαίσιο παρέχει τρόπους για προσαρμογή της εφαρμογής σύμφωνα με τις ανάγκες ή προτιμήσεις του σχεδιαστή.

Τα *εργαλεία* υλοποιούνται πάνω σε κάποιο πλαίσιο παρέχοντας, ένα σύνολο από επαναχρησιμοποιήσιμα συστατικά, τα οποία προσθέτουν επιπλέον λειτουργικότητα.

Η *υποδομή* είναι ένα σύνολο από αξιόπιστες και προσβάσιμες τεχνολογίες που λειτουργούν σαν βάση για την ανάπτυξη άλλων συστημάτων. Οι υποδομές παρέχουν σχήματα, πρωτόκολλα και άλλα *standards*, τα οποία μπορεί να χρησιμοποιήσει ένα σύστημα που θα χτιστεί πάνω στην υποδομή. Χαρακτηριστικά παραδείγματα υποδομών αποτελούν οι πλατφόρμες ενδιάμεσου λογισμικού *J2EE, CORBA* και *DCOM*.

Λειτουργίες του ενδιάμεσου λογισμικού

Συνεπώς οι εφαρμογές χρησιμοποιούν ενδιάμεσο λογισμικό που βρίσκεται πάνω από το λειτουργικό σύστημα και πρωτόκολλα επικοινωνίας για να εκτελέσουν τις ακόλουθες λειτουργίες [25].

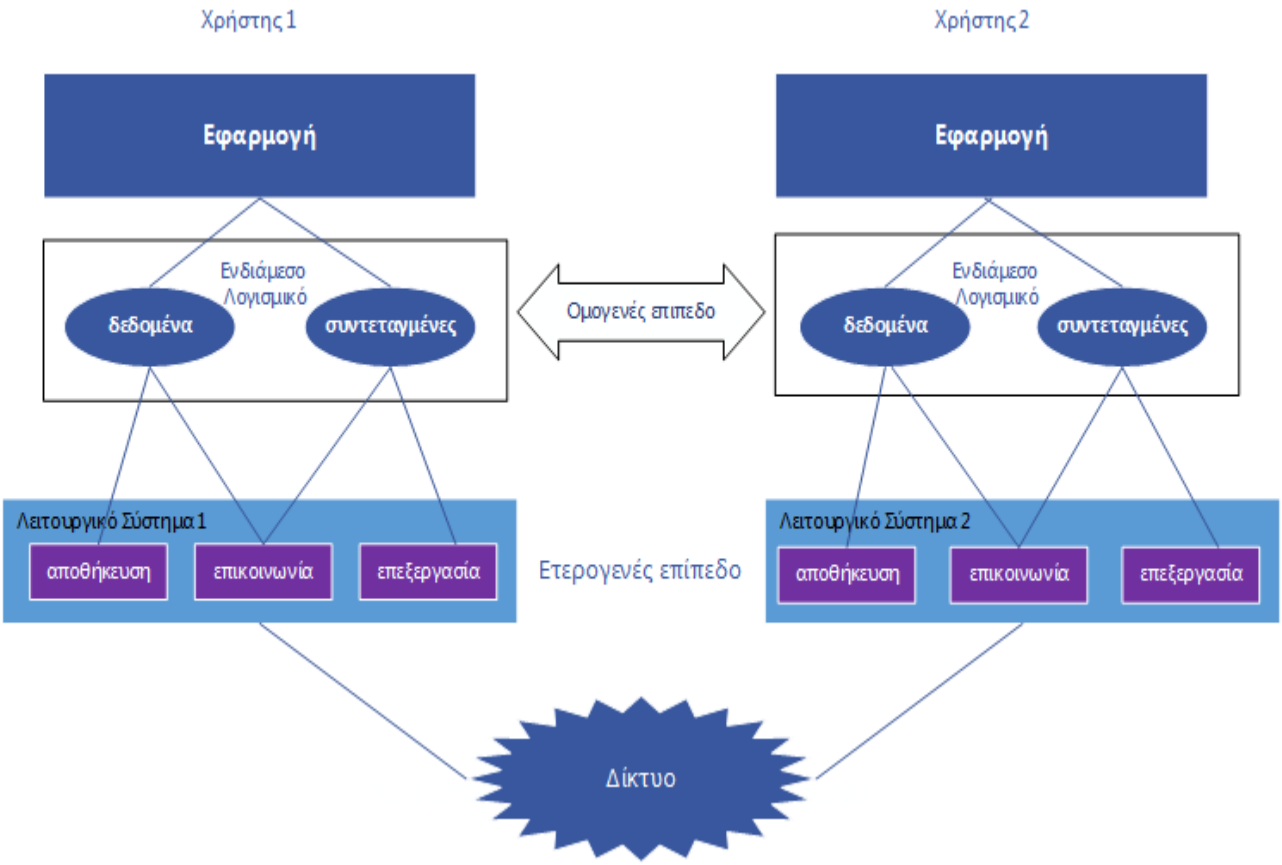
- Κάλυψη διασποράς. π.χ. το γεγονός πως μια εφαρμογή είναι κατασκευασμένη από κομμάτια που βρίσκονται σε διεσπαρμένες τοποθεσίες

- Κάλυψη της ετερογένειας των διαφόρων συσκευών, λειτουργικών συστημάτων και πρωτοκόλλων επικοινωνιών

- Παροχή ενιαίων, συγκεκριμένων και υψηλού επιπέδου διεπαφών στους προγραμματιστές ώστε οι εφαρμογές να μπορούν εύκολα να σχεδιαστούν, να μεταφερθούν και να χρησιμοποιηθούν από διάφορους χρήστες και συσκευές.

- Παροχή συνόλου υπηρεσιών για να εκτελεστούν λειτουργίες γενικού σκοπού, ώστε να αποφευχθεί η διπλή προσπάθεια και να διευκολυνθεί η συνεργασία μεταξύ εφαρμογών.

Ο ρόλος του ενδιάμεσου λογισμικού είναι να διευκολύνει τον σχεδιασμό εφαρμογών, παρέχοντας κοινές προγραμματιστικές αοριστίες, καλύπτοντας την ετερογένεια και την διασπορά του υλικού και των λειτουργικών συστημάτων και κρύβοντας τις προγραμματιστικές λεπτομέρειες χαμηλού επιπέδου.



Εικόνα IV: Λειτουργία Ενδιάμεσου Λογισμικού

Κατηγορίες ενδιάμεσου λογισμικού

Το ενδιάμεσο λογισμικό που έχει αναπτυχθεί μπορεί να χωρισθεί σε ορισμένες κατηγορίες που διαφέρουν ανάλογα με την προγραμματιστική αοριστία που παρέχουν και τα είδη ετερογένειας που παρέχουν πίσω από το δίκτυο και το υλικό [2]:

Διεσπαρμένες πλειάδες

Η διεσπαρμένη σχεσιακή βάση δεδομένων προσφέρει την αοριστία των διεσπαρμένων πλειάδων και είναι ***η πλέον διαδεδομένη μορφή ενδιάμεσου λογισμικού.*** Η γλώσσα βάσης δεδομένων που χρησιμοποιεί (SQL) επιτρέπει στους προγραμματιστές να χειριστούν αυτά τα σύνολα πλειάδων (μία βάση δεδομένων) σε μία γλώσσα παρόμοια της αγγλικής όμως με μια διαισθητική σημασιολογία και μία αυστηρή μαθηματική θεμελίωση, βασισμένη στην θεωρία συνόλων και τον κατηγορηματικό λογισμό. Ακόμα προσφέρουν συνήθως ετερογένεια ανάμεσα στις γλώσσες προγραμματισμού, αν και τις περισσότερες φορές προσφέρεται από ελάχιστη έως καθόλου ετερογένεια σε εμπορικές εφαρμογές. Επίσης οι διεσπαρμένες σχεσιακές βάσεις δεδομένων προσφέρουν την αοριστία της συναλλαγής. Οι οθόνες επεξεργασίας συναλλαγών, Transaction Processing Monitors (TPM), χρησιμοποιούνται συνήθως για την end-to-end διαχείριση των πόρων των ερωτημάτων των πελατών, ειδικά από την πλευρά του διακομιστή, τη διαχείριση της διαδικασίας και τη διαχείριση συναλλαγών πολλαπλών δεδομένων.

Παραδείγματα:

Το Linda είναι ένα framework που παρέχει την αοριστία των διεσπαρμένων πλειάδων και ονομάζεται Tuple Space (TS). Τα API’s του Linda παρέχουν σχεσιακή πρόσβαση στο TS, χωρίς όμως καμία σχεσιακή σημασιολογία. Ακόμα παρέχει χωρική αποσύνδεση επιτρέποντας στις διαδικασίες κατάθεσης και απόσυρσης να μην γνωρίζει η μία της ταυτότητα της άλλης. Ακόμα προσφέρει χρονική αποσύνδεση επιτρέποντας τους να μην έχουν αλληλεπικαλυπτόμενες διάρκειες ζωής.

Το Jini είναι ένα Java framework για έξυπνες συσκευές, ιδιαίτερα οικιακές. Το Jini είναι δομημένο πάνω και σχετίζεται πολύ με το TS του Linda.

Απομακρυσμένη κλήση διαδικασιών (RPC)

Το ενδιάμεσο λογισμικό της απομακρυσμένης κλήσης διαδικασίας (RPC) επεκτείνει την διεπαφή της κλήσης μιας διαδικασίας προσφέροντας στους προγραμματιστές την αοριστία του να είναι σε θέση να επικαλεστούν μια διαδικασία το σώμα της οποίας είναι σε ένα άλλο τμήμα ενός δικτύου. Τα RPC συστήματα είναι συνήθως σύγχρονα και ως εκ τούτου δεν προσφέρουν καμία δυνατότητα παραλληλισμού χωρίς τη χρήση πολλαπλών νημάτων και έχουν συνήθως περιορισμένες παροχές χειρισμού εξαιρέσεων.

- **Ενδιάμεσο λογισμικό προσανατολισμένου μηνύματος (Message-Oriented Middleware)**

Το ενδιάμεσο λογισμικό προσανατολισμένου μηνύματος παρέχει την αοριστία μιας ουράς μηνυμάτων που μπορεί να γίνει προσβάσιμη μέσω ενός δικτύου. Είναι μια γενίκευση του πολύ γνωστού λειτουργικού συστήματος: του γραμματοκιβωτίου.

Είναι πολύ ευέλικτο για το πώς μπορεί να ρυθμιστεί με την τοπολογία των προγραμμάτων που καταθέτουν και αποσύρουν τα μηνύματα από μια συγκεκριμένη ούρα. Πολλά προϊόντα (MOM) προσφέρουν ουρές με επιμονή, αναπαραγωγή, ή εκτέλεση σε πραγματικό χρόνο. Το ενδιάμεσο λογισμικό προσανατολισμένου μηνύματος προσφέρει το ίδιο είδος χωρικής και χρονικής αποσύνδεσης που προσφέρει και η Linda.

- **Ενδιάμεσο λογισμικό διεσπαρμένου αντικειμένου (DOM)**

Το ενδιάμεσο λογισμικό διεσπαρμένου αντικειμένου παρέχει την αοριστία ενός αντικειμένου που είναι απομακρυσμένο, στις μεθόδους όμως του οποίου μπορεί να γίνει επίκληση ακριβώς όπως θα γινόταν σε ένα αντικείμενο εάν βρισκόταν στον ίδιο χώρο διεύθυνσης του καλούντος.

Τα διεσπαρμένα αντικείμενα διαθέτουν στον προγραμματιστή της εφαρμογής όλα τα πλεονεκτήματα της τεχνολογίας λογισμικού των τεχνικών προσανατολισμένου αντικειμένου, όπως ενθυλάκωση, κληρονομικότητα και πολυμορφισμός.

2.5.1 **Context Aware Ενδιάμεσο Λογισμικό**

Όπως είδαμε η context πληροφορία έχει ορισμένα ιδιαίτερα χαρακτηριστικά που την καθιστούν δύσκολη στον χειρισμό από ένα υπολογιστικό σύστημα. Για το λόγο αυτό η κάθε εφαρμογή που χρησιμοποιεί context πληροφορία απαιτεί την ανάπτυξη ξεχωριστών υποσυστημάτων, ανεξάρτητων της εφαρμογής, που θα υλοποιούν τις διαδικασίες επεξεργασίας και διαχείρισης του context.

Τα προβλήματα αυτά οδήγησαν τους σχεδιαστές context-aware εφαρμογών σε μία διαφορετική αρχιτεκτονική προσέγγιση. Σύμφωνα με αυτή η διαχείριση και η επεξεργασία του context αναλαμβάνονται εξ ολοκλήρου από ένα **ενδιάμεσο λογισμικό (middleware)** ανεξαρτήτως της εφαρμογής. Το ενδιάμεσο λογισμικό είναι μια κατηγορία τεχνολογίας λογισμικού σχεδιασμένο για να βοηθάει στην διαχείρηση της πολυπλοκότητας και την ετερογένεια που υπάρχει στα διεσπαρμένα συστήματα. Με τον τρόπο αυτό παρέχονται στην εφαρμογή έτοιμες υπηρεσίες και εργαλεία προσαρμογής. Έτσι η εφαρμογή σχεδιάζεται πάνω στο ενδιάμεσο λογισμικό, δίχως ο προγραμματιστής να χρειάζεται να ασχοληθεί με την υλοποίηση λειτουργιών διαχείρισης και επεξεργασίας του context. Τελικά το ενδιάμεσο λογισμικό που διαχειρίζεται context και προσδίδει context-aware χαρακτηριστικά στην εφαρμογή ονομάζεται **Context–Aware Middleware (CAM)** [48].

Οι παραδοσιακές πλατφόρμες ενδιάμεσου λογισμικού έλυναν προβλήματα όπως η κλιμάκωση και η ετερογένεια και παρείχαν διαφάνεια στην κατανομή και στο διαμοιρασμό των πόρων. Παρείχαν στους σχεδιαστές, ένα υψηλό επίπεδο αφαίρεσης κρύβοντας τις λεπτομέρειες της κατανομής στην ανάπτυξη του συστήματος. Σε αυτές τις τεχνολογίες το ενδιάμεσο λογισμικό κτίζονταν σαν ένα ξεχωριστό σύστημα που έπαιρνε είσοδο από την εφαρμογή και παρείχε διαφάνεια στους χρήστες και τους σχεδιαστές των εφαρμογών [8].

Οι τεχνολογίες αυτές έχουν χρησιμοποιηθεί με επιτυχία για στατικά κατανεμημένα συστήματα, σχεδιασμένα για ενσύρματα δίκτυα. Παρόλα αυτά δεν μπορούν να αντεπεξέλθουν επαρκώς στις απαιτήσεις των κινητών εφαρμογών, που μπορούν να παρέχουν υπηρεσίες οπουδήποτε, κάθε στιγμή. Αυτές οι τεχνολογίες υποθέτουν *μεγάλο διαθέσιμο εύρος ζώνης*, καθώς και *σταθερή διαθεσιμότητα*. Σε κινητά συστήματα αυτές οι υποθέσεις δεν ισχύουν. Επιπλέον, αντικειμενοστραφείς πλατφόρμες ενδιάμεσου λογισμικού υποστηρίζουν κυρίως *σύγχρονη επικοινωνία*, ενώ σε κινητά περιβάλλοντα είναι πολύ συνηθισμένο ο διακομιστής και ο χρήστης να μην είναι ταυτόχρονα συνδεδεμένοι. Το

ενδιάμεσο λογισμικό που προορίζεται για *CA* εφαρμογές πρέπει να είναι σχεδιασμένο με βάση τα χαρακτηριστικά των διάχυτων συστημάτων και των συσκευών που τα συνθέτουν.

Η προσέγγιση ενός ενδιάμεσου λογισμικού με στατική συμπεριφορά δεν είναι η κατάλληλη για ένα *CAM [7]*. Το *CAM* πρέπει να ανιχνεύει τις αλλαγές που γίνονται στο περιβάλλον του και να προσαρμόζει τις υπηρεσίες του. Το ενδιάμεσο λογισμικό, λαμβάνοντας συγκεκριμένη πληροφορία από την εφαρμογή και το περιβάλλον, θα μπορούσε να πάρει πιο αποτελεσματικές αποφάσεις, παρέχοντας ποιοτικότερες υπηρεσίες.

Επιπλέον η εφαρμογή μπορεί να τροποποιήσει την εσωτερική δομή του *CAM* σύμφωνα με τις ανάγκες της (*absorption*). Για παράδειγμα, η τροποποίηση μπορεί να αφορά το σύνολο των πόρων που παρακολουθούνται ή τους κανόνες προσαρμογής στο *context*.

Οι υπηρεσίες διαχείρισης ενός *CAM* μπορεί να παρέχονται κεντρικά από κάποια συσκευή με αυξημένες υπολογιστικές δυνατότητες ή αποκεντρικοποιημένα, από το ενδιάμεσο λογισμικό, που βρίσκεται πάνω στη κινητή συσκευή. Στην πρώτη περίπτωση οι σύνθετες διαδικασίες επιτελούνται σε κάποιους στατικούς εξυπηρέτες, οι οποίοι παρέχουν υπηρεσίες διαχείρισης του *context* στους κινητούς χρήστες, όπου τρέχουν οι εφαρμογές. Στη δεύτερη περίπτωση όλη η διαχείριση και επεξεργασία του *context* γίνεται από το ενδιάμεσο λογισμικό που βρίσκεται πάνω στην κινητή συσκευή.

Η κεντρικοποιημένη αρχιτεκτονική είναι πιο εύκολη στην υλοποίηση, καθώς το ενδιάμεσο λογισμικό δεν εξαρτάται από τους υπολογιστικούς πόρους της συσκευής για την οποία προορίζεται. Αυτή η προσέγγιση όμως, έχει περιορισμένες δυνατότητες κλιμάκωσης και δεν μπορεί να εφαρμοστεί σε διάχυτα δίκτυα μεγάλης γεωγραφικής έκτασης. Επιπλέον δεν είναι ανεκτική σε σφάλματα καθώς εξαρτάται από ένα κεντρικό συστατικό.

Τα *CAMs* με κεντρικοποιημένη αρχιτεκτονική σχεδιάζονται κυρίως για τη δημιουργία *«Ευφυών χώρων»* (*intelligent spaces*) ή *«Ενεργών χώρων»* (*active spaces*). Οι *φυσικοί χώροι* είναι περιορισμένες γεωγραφικά περιοχές, όπως δωμάτια, γραφεία, εργαστήρια, οχήματα κ.α., οι οποίοι περιλαμβάνουν φυσικά αντικείμενα, ετερογενείς διασυνδεδεμένες συσκευές και χρήστες που επιτελούν ένα σύνολο δραστηριοτήτων. Οι *ευφυείς χώροι* ή *ενεργοί χώροι* είναι φυσικοί χώροι, που περιλαμβάνουν μία υποδομή *CA* λογισμικού που συντονίζει την αλληλεπίδραση των χρηστών με το περιβάλλον τους [33].

Τα συστήματα ενδιάμεσου λογισμικού που έχουν προταθεί μέχρι σήμερα εμφανίζουν κάποια κοινά λειτουργικά χαρακτηριστικά. Στην ***Εικόνα V*** φαίνεται αφαιρετικά η αρχιτεκτονική ενός *CAM*. Η αρχιτεκτονική κατανέμει σε τέσσερα επίπεδα τη λειτουργικότητα ενός *CAM*, ενώ στο πέμπτο επίπεδο βρίσκεται η εφαρμογή [3]

<table>
<tr><th colspan="2">Εφαρμογή</th></tr>
<tr><td>Φιλτράρισμα / Αποθήκευση</td><td>Context Handling</td></tr>
<tr><td>Επεξεργασία / Συλλογισμός</td><td>Context Reasoning</td></tr>
<tr><td>Συλλογή Context Δεδομένων (raw data)</td><td rowspan="2">Context Acquisition</td></tr>
<tr><td>Προμηθευτές (Αισθητήρες)</td></tr>
</table>

Εικόνα V: Αφαιρετική Αρχιτεκτονική Ενός Context Aware Συστήματος

Το πρώτο επίπεδο αποτελείται από τους προμηθευτές. Σύμφωνα με την κατηγοριοποίηση που έγινε παραπάνω, οι προμηθευτές μπορεί να είναι οποιαδήποτε πηγή υλικού ή λογισμικού που παρέχει τιμές για τις παραμέτρους του *context.*

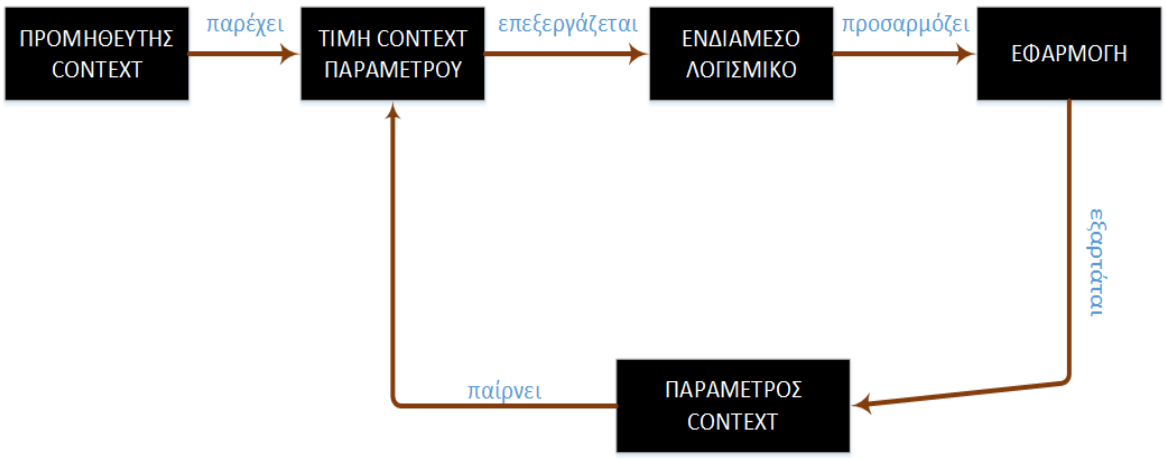
Στο δεύτερο επίπεδο γίνεται η συλλογή της φυσικής πληροφορίας. Στο επίπεδο αυτό χρησιμοποιούνται οδηγοί για την επικοινωνία με προμηθευτές υλικού και *APIs* για την επικοινωνία με το λογισμικό. Το λογισμικό απόκτησης της πληροφορίας υλοποιείται με επαναχρησιμοποιήσιμα τμήματα, που κρύβουν τα τεχνικά χαρακτηριστικά του προμηθευτή και αποκτούν την πληροφορία με διαφάνεια προς τα ανώτερα επίπεδα.

Στο τρίτο επίπεδο γίνεται η ερμηνεία των τιμών που συλλέχθηκαν και η εξαγωγή ανώτερου εννοιολογικά *context*. Το επίπεδο αυτό δεν είναι απαραίτητο για ένα *CA* σύστημα, αλλά είναι ιδιαίτερα χρήσιμο αν η εφαρμογή απαιτεί υψηλού εννοιολογικού επιπέδου πληροφορία. Η πληροφορία από διαφορετικούς προμηθευτές συνδυάζεται και προκύπτει υψηλότερο εννοιολογικά *context*. Επίσης σε αυτό το επίπεδο διευθετούνται

προβλήματα που σχετίζονται με την ασάφεια και την ανακρίβεια της πληροφορίας που παράγουν οι προμηθευτές.

Στο τέταρτο επίπεδο γίνεται η μοντελοποίηση και αποθήκευση του *context*. Στο επίπεδο αυτό υλοποιείται μια διεπαφή για την παροχή της πληροφορίας στις εφαρμογές. Η πρόσβαση της εφαρμογής στα δεδομένα μπορεί να γίνει με δύο τρόπους: *σύγχρονα* και *ασύγχρονα*. Στη σύγχρονη επικοινωνία, η εφαρμογή ρωτάει το επίπεδο διαχείρισης για την πληροφορία που την ενδιαφέρει. Η ασύγχρονη επικοινωνία γίνεται μέσω *εγγραφών* και *ειδοποιήσεων*. Η εφαρμογή εγγράφεται στη λίστα ειδοποίησης για κάποια παράμετρο του *context* και όταν συμβεί κάποιο γεγονός ειδοποιείται λαμβάνοντας τη νέα τιμή. Στις περισσότερες περιπτώσεις η ασύγχρονη επικοινωνία είναι πιο κατάλληλη, λόγω των συχνών αλλαγών στις τιμές του *context*.

Στο πέμπτο επίπεδο βρίσκεται η εφαρμογή που χρησιμοποιεί το *context*. Στο επίπεδο αυτό υλοποιείται ο τρόπος που η εφαρμογή αντιδρά στις αλλαγές του *context* και η χρήση της πληροφορίας από τις υπηρεσίες της εφαρμογής[48].



Εικόνα VI: Λειτουργίες Context - Aware Ενδιάμεσου Λογισμικού

Παρολαυτά οι τεχνολογίες και η ανάπτυξη του ενδιάμεσου λογισμικού δεν αποτελεί πανάκεια στο πρόβλημα της ετερογένειας των διασκορπισμένων συστημάτων.

Αρχικά υπάρχει ένα κενό ανάμεσα στην θεωρία και στην πράξη. Πολλές δημοφιλείς υπηρεσίες ενδιάμεσου λογισμικού χρησιμοποιούν ιδιόκτητα API's (συνήθως κάνοντας τις εφαρμογές να εξαρτώνται από έναν μόνο προϊόν προμηθευτή) και ιδιόκτητα και αδημοσίευτα πρωτόκολλα (καθιστώντας δύσκολο για τους διαφορετικούς προμηθευτές να

χτίσουν διαλειτουργικές εφαρμογές). Για παράδειγμα αρκετά συστήματα διαχείρισης βάσεων δεδομένων υποστηρίζουν ιδιοκτησιακές SQL διαλέκτους και αντίστοιχα πρωτόκολλα, τα οποία δεν είναι διαθέσιμα σε πολλές δημοφιλείς πλατφόρμες (όπως της Oracle, της IBM κ.α.). Ως αποτέλεσμα περιορίζονται οι δυνατότητες ενός χρήστη να συνδέεται σε συστήματα που παρουσιάζουν ετερογένεια με το δικό του. Ακόμα και αν μία τεχνολογία ενδιάμεσου λογισμικού είναι σχετικά σύγχρονη, όταν ένας προγραμματιστής ενδιαφέρεται να δομήσει την εφαρμογή του έχει να λογαριάσει το ρίσκο και την δυνατότητα του συγκεκριμένου ενδιάμεσου λογισμικού να παραμένει σύγχρονο και να εξελιχθεί παράλληλα με την τεχνολογία. Για παράδειγμα πολλές εφαρμογές που γράφτηκαν βασισμένες σε τεχνολογίες ενδιάμεσου λογισμικού που δεν εξελίχθηκαν και ξεπεράστηκαν, αναγκαστικά ξαναγράφτηκαν ώστε να ενσωματώσουν σύγχρονα λογισμικά.

Δευτερεύοντος, το μεγάλο πλήθος υπηρεσιών ενδιαμέσου λογισμικού αποτελεί από μόνο του ανασταλτικό παράγοντα για την χρησιμοποίηση τους. Αυτό καθώς ακόμα και ένας μικρός αριθμός από διαφορετικά ενδιάμεσα λογισμικά μπορεί να οδηγήσει σε υψηλή προγραμματιστική πολυπλοκότητα. Γι’ αυτό το λόγο οι προγραμματιστές πρέπει να επιλέξουν έναν πολύ μικρό αριθμό υπηρεσιών που θα ικανοποιούν τις ανάγκες τους για λειτουργικότητα κ.α.

Τρίτον, ενώ οι υπηρεσίες ενδιάμεσου λογισμικού ανεβάζουν το επίπεδο αοριστίας που απαιτεί ο προγραμματισμός διασκορπισμένων εφαρμογών, επιβαρύνουν ακόμα τον προγραμματιστή με τις δύσκολες αποφάσεις σε ότι αφορά τον σχεδιασμό.

3 Ανασκόπηση Βιβλιογραφίας (State of the Art)

Μέχρι σήμερα έχουν αναπτυχθεί αρκετά συστήματα ενδιάμεσου λογισμικού, τα οποία αφορούν context – aware εφαρμογές ποικίλων συσκευών. Στη δεκαετία του '*90*, η έρευνα στόχευε κυρίως σε *CA εφαρμογές* για συγκεκριμένο σκοπό. Στη συνέχεια ('00) έγινε μία μετατόπιση της έρευνας, προς το πρόβλημα της σχεδίασης και ανάπτυξης ενδιάμεσου λογισμικού για την υποστήριξη *CA* εφαρμογών. Αρχικά αναπτύχθηκε μια μεγάλη ποικιλία πλατφορμών ενδιάμεσου λογισμικού, οι οποίες δεν αφορούσαν διαδικτυακές εφαρμογές, οπότε και δεν είχαν σχεδιαστεί ώστε να λειτουργούν σε τέτοιο περιβάλλον. Σήμερα η διαχείριση του *context* και οι διαδικασίες που τη συνθέτουν, αποτελούν θέμα πολλών εργασιών και οι οποίες στην πλειονότητα τους είναι βασισμένες στο διαδίκτυο και σε κατανεμημένες υπηρεσίες.

Η πρώτη σημαντική προσέγγιση προς αυτή τη κατεύθυνση, ήταν από τον *W. Schilit* [37]. Ο *Schilit* στη διδακτορική του διατριβή πρότεινε ένα γενικό πλαίσιο για *CA* εφαρμογές, επικεντρώνοντας κυρίως σε *context* που αφορούσε τη γεωγραφική θέση του χρήστη. Ο *B. Schilit* [36] πρότεινε ένα πλαίσιο με κεντρικοποιημένη αρχιτεκτονική. Το πλαίσιο όριζε ένα *περιβάλλον* σαν ένα σύνολο από *ονόματα παραμέτρων* και *τιμών* για τις παραμέτρους. Στατικοί διακομιστές διαχειρίζονταν το περιβάλλον και μετέφεραν την πληροφορία, στους χρήστες (εφαρμογές) που είχαν ενδιαφερθεί πιο πριν κάνοντας εγγραφή στους διακομιστές. Τυπικά αντιστοιχούσε ένα περιβάλλον ανά χρήστη και επιπρόσθετα περιβάλλοντα για συγκεκριμένες οντότητες όπως δωμάτια, ομάδες ατόμων κ.α. Παρόμοιες προσεγγίσεις με αυτήν του *B. Schilit* προτάθηκαν ως πλαίσια που δρουν σαν ενδιάμεσο λογισμικό που συλλέγει το *context* από τους αισθητήρες και παρέχει μεταφρασμένη την πληροφορία στην εφαρμογή, μέσω ενός συγκεκριμένου *API.*

Η *Hewlett Packard* πρότεινε ένα πλαίσιο με το όνομα "*An environment for situational Computing*" [23]. Η ιδέα βασίζεται στην ύπαρξη ενός εξυπηρέτη για την συλλογή και επεξεργασία του *context*. Ο εξυπηρέτης παρέχει ένα σύνολο υπηρεσιών, για την μετατροπή των φυσικών δεδομένων που έδιναν οι αισθητήρες, σε γεγονότα *context*. Οι υπηρεσίες αυτές, είναι υπεύθυνες για τη συλλογή της πληροφορίας από τους αισθητήρες και την παροχή της σε κατανοητή και χρήσιμη μορφή στην εφαρμογή. Θεμελιώδης για την έρευνα στα *CA* συστήματα θεωρείται η εργασία του *Dey* από το *GeorgiaTech*. Ο *Dey* στο πλαίσιο της διδακτορικής του διατριβής πρότεινε μια γενική αρχιτεκτονική για την υποστήριξη και

ανάπτυξη *CA* εφαρμογών. Υλοποίησε μια υποδομή βασισμένη σε αυτήν την αρχιτεκτονική και ένα εργαλείο με το όνομα *Context Toolkit* [14]. Η αρχιτεκτονική του *Context Toolkit* βασίζεται σε μια αντικειμενοστραφή προσέγγιση και περιλαμβάνει τρεις τύπους αντικειμένων: *widgets, servers* ή *aggregators* και *interpreters.*

Το *context widget* είναι ένα μια 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 της *context* πληροφορίας. Κρύβουν την πολυπλοκότητα των αισθητήρων παρέχοντας αφαιρετικά την πληροφορία, ώστε να μπορεί να χρησιμοποιηθεί από την εφαρμογή.

Οι *context server*s είναι υπεύθυνοι για τη συλλογή της πληροφορίας του *context* που αφορά μια 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 έναν 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 είχε γίνει σε *Java*, αλλά οι μηχανισμοί που χρησιμοποιούνται είναι ανεξάρτητοι από τη γλώσσα προγραμματισμού. Από τότε που παρουσιάστηκε μέχρι και σήμερα, το *Context Toolkit* ενέπνευσε πολλούς ερευνητές οι οποίοι επέκτειναν ή στήριξαν τις δουλειές τους στις ιδέες του *Dey.* Οι αρχιτεκτονικές αρχές του *Toolkit* βρήκαν μεγάλη απήχηση στην ερευνητική κοινότητα και αποτέλεσαν τη βάση για αρκετά πλαίσια ανάπτυξης *CA* εφαρμογές, που προτάθηκαν από τότε.

Οι *Hong* και *Landay* [22] επέκτειναν τη δουλειά του *Dey*, μελετώντας την ιδέα μιας υποδομής υπηρεσιών, ενδιάμεσου λογισμικού. Προτείνουν μια γενική αρχιτεκτονική, όπου η διαχείριση του *context* παρέχεται σαν ένα σύνολο υπηρεσιών από μια υποδομή ενδιάμεσου λογισμικού. Τα *widgets* αντικαθίστανται από τεχνικές εύρεσης υπηρεσιών μεταξύ των διάχυτων κόμβων. Αυτή η προσέγγιση παρέχει ανοχή σε σφάλματα που μπορεί

να προκύψουν από την κατάρρευση κάποιου *widget*, αλλά χάνει σε αποτελεσματικότητα, λόγω της αυξημένης δικτυακής επικοινωνίας που απαιτεί μεταξύ των συστατικών του συστήματος.

Ο *Winogrard* [47] σύγκρινε διαφορετικές αρχιτεκτονικές εκδοχές για το χτίσιμο *CA* συστημάτων και όρισε κριτήρια και *trade – offs* στην επιλογή του αρχιτεκτονικού μοντέλου. Κατέληξε ότι, μια προσέγγιση *μαυροπίνακα* (*blackboard-based*) είναι πιο ευέλικτη, από ότι αρχιτεκτονικής βασισμένες σε *widgets*, όπως το *ContextToolkit*. Σε αυτήν την αρχιτεκτονική, οι εφαρμογές γράφουν μηνύματα σε ένα κοινό μέσο, το μαυροπίνακα και εγγράφονται σε κάποια υπηρεσία ειδοποίησης, ώστε να ειδοποιηθούν, όταν συμβούν συγκεκριμένα γεγονότα.

Στην εργασία [31] περιγράφεται ένα απλό αρχιτεκτονικό πλαίσιο που έχει χρησιμοποιηθεί για την ανάπτυξη δύο *CA* εφαρμογών για μουσεία. Το πλαίσιο στοχεύει στην υποστήριξη εφαρμογών σε έξυπνους χώρους όπως ένα μουσεία. Παρέχει μια υπηρεσία βασισμένη σε απλούς κανόνες για την επιλογή των πληροφοριών που θα σταλούν στη συσκευή του χρήστη. Όλη η διαχείριση γίνεται από έναν κεντρικό εξυπηρέτη που δέχεται το *context* από την εφαρμογή και το περιβάλλον εκτέλεσης και στέλνει δεδομένα στη συσκευή του χρήστη.

Το *CoolAgent* [11] είναι ένα πλαίσιο βασισμένο σε μια αρχιτεκτονική με πολλαπλούς πράκτορες. Υποστηρίζει υπηρεσίες συλλογής και διαμοιρασμού *context* και εξαγωγής συμπερασμάτων. Το *context* μοντελοποιείται με χρήση οντολογίας και *RDF* (*Resource Description Framework*) [26]. Οι υπηρεσίες παρέχονται μέσω πρακτόρων με συγκεκριμένους ρόλους, ενώ η εξαγωγή συμπερασμάτων γίνεται τοπικά πάνω σε μία βάση δεδομένων *context*, με ένα σύστημα κανόνων υλοποιημένο σε *Prolog*. Η βάση δεδομένων βρίσκεται σε έναν κεντρικό κόμβο εξυπηρέτη, που διαχειρίζεται το *context* και το παρέχει στις εφαρμογές.

Στο [34] προτείνεται ένα πλαίσιο που στοχεύει στην ανακάλυψη και αυτόματη εκτέλεση κατάλληλων υπηρεσιών σε διάχυτα περιβάλλοντα. Στο σύστημα *CAPEUS* που υλοποιήθηκε, η επιλογή της κατάλληλης υπηρεσίας προς εκτέλεση γίνεται με βάση τις ανάγκες του χρήστη, που καθορίζονται από το *context* και τις παρεχόμενες υπηρεσίες. Οι ανάγκες του χρήστη και οι διαθέσιμες υπηρεσίες κωδικοποιούνται υπό μορφή «περιορισμών», δίνοντας τιμές σε κάποια γνωρίσματα του *context*. Οι περιορισμοί

καταχωρούνται μέσα σε έγγραφα τα οποία μεταφέρονται για να βρεθεί η κατάλληλη υπηρεσία. Τα έγγραφα ονομάζονται *CAPs* (*context aware packets)* και περιέχουν επιπλέον ένα πλάνο, που υποδεικνύει από ποιους παρόχους υπηρεσιών πρέπει να περάσει το πακέτο. Κάθε πάροχος παραλήπτης ελέγχει αν η υπηρεσία που παρέχει ικανοποιεί τους περιορισμούς του *CAP*. Επιπλέον μπορεί να κάνει υποθέσεις για τον κατάλληλο παροχέα και να του προωθήσει το *CAP*. Η εύρεση του κατάλληλου προμηθευτή *context*, ακολουθεί την ίδια φιλοσοφία, με την εύρεση της κατάλληλης υπηρεσίας, όπως περιγράφηκε παραπάνω. Παρόλα αυτά τα *CAPs* δεν μπορούν να χρησιμοποιηθούν για την ανίχνευση μεταβολών στο *context*.

Το *Context Fabric [21]* είναι ουσιαστικά η συνέχεια της εργασίας [22] από το *Hong*. Σε αυτήν την εργασία προτείνεται ένα αρχιτεκτονικό πλαίσιο για την ανάπτυξη και υποστήριξη *CA* εφαρμογών. Τα βασικά δομικά κομμάτια της αρχιτεκτονικής είναι: μία βάση δεδομένων για αποθήκευση και μοντελοποίηση του *context*, μία γλώσσα αναπαράστασης (*context specification language*) και μηχανισμοί ασφαλείας για την εμπιστευτικότητα των δεδομένων.

Το *Solar [10]* αποτελεί μία υποδομή για τη συλλογή και το διαμοιρασμό του *context* σε διάχυτα περιβάλλοντα. Το *context* που παράγεται από τις πηγές μοντελοποιείται σε γεγονότα. Η εφαρμογή μπορεί να ορίσει ένα γράφο από τελεστές (φίλτρα, μετατροπείς κ.α.) οι οποίοι μεσολαβούν μεταξύ των προμηθευτών και της εφαρμογής. Τα γεγονότα διασχίζουν τους κόμβους του γράφου και παραδίδονται στις εφαρμογές που έχουν ενδιαφερθεί. Τα χαρακτηριστικά του *context*, που επιθυμεί η εφαρμογή περιγράφονται με μία συγκεκριμένη γλώσσα αναπαράστασης.

Η *Gaia* [33]είναι μια υποδομή ενδιάμεσου λογισμικού, που επεκτείνει τα τυπικά λειτουργικά συστήματα παρέχοντας *CA* χαρακτηριστικά στις εφαρμογές. Στοχεύει στην ανάπτυξη και υποστήριξη *CA* εφαρμογών για ενεργούς χώρους. Παρέχει υπηρεσίες για εύρεση και χρήση υπολογιστικών πόρων, κατανομή της *context* πληροφορίας και εξαγωγή συμπερασμάτων. Το *context* μοντελοποιείται σε κατηγορήματα τεσσάρων πεδίων και χρησιμοποιούνται κανόνες κατηγορηματικής λογικής πρώτης τάξεως για την εξαγωγή συμπερασμάτων. Οι κανόνες γράφονται σε *DAML+OIL*.

Στην εργασία [39] προτείνεται το *AURA*, μια υποδομή για την ανάπτυξη *CA* εφαρμογών. Βασίζεται στην ιδέα ότι, στις *CA* εφαρμογές ο πιο «περιορισμένος» και κρίσιμος πόρος είναι η *προσοχή* του χρήστη. Η προσοχή ως παράμετρος του *context*, έχει

την έννοια της συγκέντρωσης και της διάθεσης χρόνου από το χρήστη για να ασχοληθεί με την εφαρμογή του. Για το λόγο αυτό στοχεύει στην ανάπτυξη εφαρμογών, που προσαρμόζονται αυτόματα στις αλλαγές του περιβάλλοντος, ελαχιστοποιώντας την συμμετοχή του χρήστη. Το *AURA* στοχεύει στη παροχή τριών βασικών υπηρεσιών: διαφανή επιλογή του κατάλληλου μοντέλου αλληλεπίδρασης του χρήστη με μια διαδικασία, ενημερότητα για το *context*, πρόβλεψη της επιθυμητής ενέργειας και πραγματοποίηση της, χωρίς την συμμετοχή του χρήστη.

Στην εργασία [7] προτείνεται το *CARISMA*, ένα πλαίσιο ενδιάμεσου λογισμικού για *CA* εφαρμογές. Το σύστημα βασίζεται στην παροχή πολλαπλών υλοποιήσεων της ίδιας υπηρεσίας, ώστε να παρέχεται με διαφορετικό τρόπο ανάλογα με το *context*. Οι διαφορετικοί τρόποι με τους οποίους παρέχεται μια υπηρεσία ονομάζονται *πολιτικές*. Η συμπεριφορά του ενδιάμεσου λογισμικού περιγράφεται σαν ένα σύνολο συσχετίσεων μεταξύ των παρεχόμενων υπηρεσιών, των πολιτικών με τις οποίες παρέχονται οι υπηρεσίες και τις συνθήκες του *context*, που πρέπει να ισχύουν για την εφαρμογή μιας πολιτικής.

Η σημασιολογική πληροφορία για το *context* της εφαρμογής κωδικοποιείται σε *XML* έγγραφα, σχηματίζοντας το προφίλ της εφαρμογής. Όταν εκτελείται μια υπηρεσία της εφαρμογής, το ενδιάμεσο λογισμικό ελέγχει το προφίλ της εφαρμογής και το συγκρίνει με το τρέχον *context*. Ανάλογα επιλέγει μια κατάλληλη πολιτική για την εκτέλεση της υπηρεσίας. Το προφίλ της εφαρμογής μπορεί να αλλάξει κατά τη διάρκεια εκτέλεσης. Επιπλέον χρησιμοποιείται ένας μηχανισμός για την επίλυση συγκρούσεων στην επιλογή της κατάλληλης πολιτικής, βασισμένος σε μια μικροοικονομική προσέγγιση.

Το πλαίσιο ενδιάμεσου λογισμικού *Hydrogen [41]* υιοθετεί μία *peer-to-peer* αρχιτεκτονική, όπου όλα τα τμήματα για τη συλλογή και διαχείριση του *context* βρίσκονται πάνω στην κινητή συσκευή. Η αρχιτεκτονική του είναι τριών επιπέδων και είναι επεκτάσιμη για να μπορεί να συμπεριλάβει όλων των ειδών τις context πληροφορίες. Το σύστημα είναι ανεξαρτητοποιημένο από κεντρικούς κόμβους παροχής υπηρεσιών και παρέχει τη 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 κόμβους. Το ενδιάμεσο λογισμικό έχει τη δυνατότητα να αποθηκεύσει περιορισμένη ποσότητα *context* πληροφορίας. Η πληροφορία μπορεί να μεταφερθεί μεταξύ των εφαρμογών με μία υπηρεσία διαμοιρασμού του *context*. Όταν οι συσκευές βρεθούν σε κοντινή απόσταση πραγματοποιείται ασύρματα διαμοιρασμός πληροφορίας, ώστε να περιλαμβάνεται

σε παραπάνω από έναν κόμβους. Η επικοινωνία μεταξύ των εφαρμογών γίνεται πάνω από *TCP/IP*, με *XML* μηνύματα.

Το *SOCAM* (*Service Oriented Context-Aware Middleware)* [18] είναι επίσης ένα πλαίσιο που στοχεύει στη συλλογή και παροχή της *context* πληροφορίας. Ακολουθεί την φιλοσοφία ενός κεντρικού υποσυστήματος για τη διαχείριση του *context*. Ο εξυπηρέτης (*context interpreter*) συλλέγει τα δεδομένα από κατανεμημένους αισθητήρες, τα επεξεργάζεται και τα παρέχει στις εφαρμογές μέσω ερωτήσεων. Οι υπηρεσίες, που μπορούν να χρησιμοποιήσουν οι εφαρμογές, βρίσκονται στην κορυφή της αρχιτεκτονικής, είτε τοπικά στη συσκευή είτε κατανεμημένες στο δίκτυο. Οι εφαρμογές μπορούν να κάνουν χρήση διαφορετικών επιπέδων *context* ρωτώντας την κεντρική μονάδα ή ακούγοντας τα γεγονότα που στέλνουν οι προμηθευτές. Η επικοινωνία και εδώ βασίζεται σε Java RMI.

Στην εργασία [44], από την ομάδα που πρότεινε το *SOCAM*, οι οντολογίες του συστήματος επεκτείνονται για να συμπεριλάβουν *context* για ευφυείς χώρους. Το νέο πλαίσιο ονομάζεται *Semantic Space* και στοχεύει στην ανάπτυξη και υποστήριξη *CA* εφαρμογών για ευφυείς χώρους.

Το πλαίσιο *SCI* [17] παρέχει υπηρεσίες ενδιάμεσου λογισμικού για διαχείριση του *context*, όπως αναζήτηση, σύνθεση και παροχή πληροφορίας. Το *SCI* στοχεύει κυρίως στη δυναμική σύνθεση της πληροφορίας, την ανοχή σε σφάλματα και την κλιμάκωση σε έξυπνους χώρους. Την πληροφορία διαχειρίζονται κεντρικοί διακομιστές και μοντελοποιείται σε κατευθυνόμενους γράφους. Οι εφαρμογές κάνουν ερωτήσεις για την απόκτηση της πληροφορίας.

Το *CMF (Context Manager Framework)* [13] είναι ένα πλαίσιο διαχείρισης του *context* που στοχεύει στη συλλογή, ερμηνεία και παροχή της *context* πληροφορίας. Το σύστημα διαχειρίζεται κυρίως *context* από το περιβάλλον (θόρυβο, θέση, θερμοκρασία κ.α.), το οποίο μοντελοποιείται με χρήση μιας οντολογίας. Χρησιμοποιείται ασαφής λογική και Μπεϋζιανή μάθηση για τη εξαγωγή υψηλότερου σημασιολογικά *context* από χαμηλού επιπέδου πληροφορία. Το *CMF* ακολουθεί κλασική ιεραρχική αρχιτεκτονική, με τα κομμάτια του συστήματος τοποθετημένα σε επίπεδα. Η αρχιτεκτονική περιλαμβάνει τέσσερις λειτουργικές οντότητες: το διαχειριστή του *context*, τους διακομιστές που παρέχουν το *context* (προμηθευτές), τις υπηρεσίες αναγνώρισης του *context* και την εφαρμογή. Οι διακομιστές είναι κατανεμημένοι, ενώ ο διαχειριστής του *context*, λειτουργεί

σαν μια κεντρικοποιημένη μονάδα κάτω από την εφαρμογή. Στις αρμοδιότητές του είναι να αποθηκεύει κατάλληλα τα δεδομένα που του στέλνουν οι προμηθευτές και να παρέχει την εξαγόμενη πληροφορία στις εφαρμογές. Οι υπηρεσίες αναγνώρισης χρησιμοποιούνται για την ερμηνεία του *context* με βάση μια καθορισμένη οντολογία και βρίσκονται κατανεμημένες στο περιβάλλον λειτουργίας.

Το JCAF (Java Context Awareness Framework) [4] υποστηρίζει παράλληλα και την υποδομή αλλά και το προγραμματιστικό πλαίσιο για την ανάπτυξη context – aware εφαρμογών στη γλώσσα Java. Πληροφορίες Context ελέγχονται από διαφορετικές υπηρεσίες από τις οποίες οι χρήστες μπορούνε και τις να δημοσιεύσουν αλλά και να τις ανακτήσουν. Η επικοινωνία βασίζεται στην Java RMI.

Το ενδιάμεσο λογισμικό PACE [20] παρέχει πληροφορίες context και επιλογές διαχείρισης μαζί με ένα προγραμματιστικό οδηγό και εργαλεία, για την βοήθεια των context – aware εφαρμογών στις διαδικασίες αποθήκευσης, πρόσβασης και χρησιμοποίησης των context δεδομένων που διαχειρίζεται το PACE, ακόμα υποστηρίζει την λήψη αποφάσεων από τις εφαρμογές, με βάση τις προτιμήσεις του χρήστη.

Το CAMUS είναι μία υπολογιστική υποδομή για δικτυακά, context – aware, ευφυή ρομπότ[24, 28]. Υποστηρίζει μια πληθώρα context δεδομένων, όπως πληροφορίες χρήστη, τοποθεσίας, περιβάλλοντος και context συλλογισμού. Παρά ταύτα, το σύστημα αυτό σε βασίζεται σε διαδικτυακές υπηρεσίες και δουλεύει σε κλειστά περιβάλλοντα.

Το Citron είναι ένα πλαίσιο απόκτησης context πληροφοριών για προσωπικές συσκευές [40]. Μαζεύει πληροφορίες για έναν χρήστη και τον περιβάλλοντα χώρο του, και τις χρησιμοποιεί για να προσαρμόσει ανάλογα την συμπεριφορά των εφαρμογές που τρέχουν στην προσωπική του συσκευή.

Το *CASS* [35] είναι ένα ακόμα πλαίσιο που υιοθετεί την αρχιτεκτονική με έναν κεντρικό κόμβο για την επεξεργασία του *context*. Οι εφαρμογές - χρήστες 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 αισθητήρες, μια μηχανή εξαγωγής συμπερασμάτων, διερμηνέα του *context* και μία βάση δεδομένων για αποθήκευση

και ανάκτηση της πληροφορίας. Ο πελάτης περιλαμβάνει τμήματα για επικοινωνία με τις υπηρεσίες του ενδιάμεσου λογισμικού.

Το σύστημα *WASP* (*Web Architectures for Services Platforms*) [12] είναι μια υποδομή για την ανάπτυξη και υποστήριξη *CA* εφαρμογών, που παρέχει υπηρεσίες διαδικτύου, κινητής ομιλίας και δίκτυα κινητών τηλεφώνων τρίτης γενιάς. Στην αρχιτεκτονική του *WASP* ένα κεντρικό υποσύστημα (*Context Interpreter*) είναι υπεύθυνο για τη συλλογή και παροχή της πληροφορίας. Η πληροφορία παρέχεται στο υποσύστημα παρακολούθησης (*Monitor Module*), το οποίο υποστηρίζεται από ένα σύνολο μονάδων αποθήκευσης, για καταχώρηση της πληροφορίας για μελλοντική χρήση. Για την διασύνδεση μιας εφαρμογής με τη πλατφόρμα *WASP*, χρησιμοποιείται η γλώσσα *WSL*, στην οποία κωδικοποιούνται η συμπεριφορά και τα στοιχεία της εφαρμογής. Η αναπαράσταση της πληροφορίας γίνεται με χρήση οντολογιών. Η διαχείριση των οντολογιών γίνεται με την *OWL*.

Το CoWSAMI είναι ένα ενδιάμεσο λογισμικό που υποστηρίζει context – aware εφαρμογές σε διάχυτα περιβάλλοντα [15]. Παρέχει έναν διαχειριστή context, του οποίου δουλειά είναι η διαχείριση των προμηθευτών context. Έχουμε μία σχεσιακή αναπαράσταση των context πληροφοριών και ο ορισμός τους γίνεται ανάλογα με τον προμηθευτή που τις παρέχει. Το inContext project [42] παρέχει διάφορες τεχνικές υποστήριξης ομάδων συνεργασίας στον τομέα του context awareness. Είναι σχεδιασμένο για διαδικτυακά περιβάλλοντα ομαδικής εργασίας. Το inContext παρέχει μεθόδους μοντελοποίησης, αποθήκευσης, συλλογισμού και ανταλλαγής context πληροφοριών ανάμεσα σε υπηρεσίες, μέσω του διαδικτύου.

Το πλαίσιο ESCAPE [42] είναι ένα διαδικτυακό σύστημα διαχείρισης context δεδομένων που εξειδικεύεται σε ομαδική εργασία και έκτακτες καταστάσεις, διαχείριση κινδύνου, καταστροφής κλπ. Το ESCAPE είναι σχεδιασμένο έτσι ώστε να παρέχει front end υπηρεσίες στις κινητές συσκευές και back end στα αντίστοιχα συστήματα. Για τις κινητές συσκευές το ESCAPE αποτελείται από λειτουργίες ανίχνευσης δεδομένων, καθώς και ανταλλαγής τους, μέσω του διαδικτύου και οι οποίες εκτελούνται μέσω ενός ad hoc δικτύου κινητών συσκευών. Από την άλλη το back end περιλαμβάνει μια διαδικτυακή υπηρεσία αποθήκευσης και ανταλλαγής context πληροφοριών ανάμεσα σε διαφορετικές συσκευές.

Το AmbieSense βλέπει το μέλλον των ευφυών χώρων ύπαρξης και λειτουργίας, είτε μιλάμε για τους ανθρώπους είτε για τα προγραμματιστικά περιβάλλοντα. Βασισμένο στο ότι

πληροφορίες και δεδομένα αντιστοιχίζονται καθημερινά σε όλα τα αντικείμενα και τους χώρους που μας περιστοιχίζουν. Το όραμα του είναι: « Η σωστή πληροφόρηση στον σωστό χρήστη, υπό τις επικρατούσες συνθήκες» Kofod-Petersen 2005 #44}.

Αυτή τη στιγμή ο έλεγχος και η ασφάλεια των δεδομένων που διαχειρίζονται οι context aware εφαρμογές είναι η κύρια έννοια, τόσο των προγραμματιστών όσο και των τελικών χρηστών. Οι υπάρχουσες τεχνολογίες ενδιαμέσων λογισμικών δεν βοηθάνε προς αυτή την κατεύθυνση, καθώς η χρήση και η προστασία των πληροφοριών που χρησιμοποιούνται από πολλαπλές συσκευές όλο και δυσκολεύει τον μέσο χρήστη, παρόλη την αλματώδη αύξηση της αποθήκευσης των δεδομένων των εφαρμογών στο ‘‘σύννεφο’’.

4 Webinos

Το webinos είναι ένα έργο χρηματοδοτούμενο από την ΕΕ που έχει διάρκεια τρία χρόνια (από τον Σεπτέμβριο του 2010 έως τον Αύγουστο του 2013). Έχει παραπάνω από είκοσι συνεργάτες σε όλη την Ευρώπη ανάμεσα στους οποίους συγκαταλέγονται ακαδημαϊκά ιδρύματα, εταιρείες βιομηχανικής έρευνας, εταιρείες ανάπτυξης λογισμικού, αυτοκινητοβιομηχανίες κ.α. Στόχος του είναι να προσφέρει μια πλατφόρμα που θα υποστηρίζει την γρήγορη δημιουργία καινοτόμων και ασφαλών δικτυακών εφαρμογών για κάθε είδους έξυπνες συσκευές, όπως κινητά τηλέφωνα, Η/Υ (φορητούς και μη), τηλεοράσεις και συσκευές ενσωματωμένες σε οχήματα. Προσδοκία του έργου αυτού είναι να παρέχει ένα δικτυακά εφαρμόσιμο ομογενοποιημένο πλαίσιο πολλαπλών συσκευών, πολλαπλών χρηστών και πολλαπλών λειτουργικών περιβαλλόντων [43].

Πολλαπλών συσκευών σημαίνει πως οι διάφορες συσκευές που τρέχουν στο webinos έχουν την δυνατότητα να επικοινωνούν μεταξύ τους. Πολλαπλών χρηστών σημαίνει πως ένας χρήστης με τις συσκευές που έχει στην διάθεση του μπορεί εύκολα να επικοινωνήσει με τις συσκευές ενός άλλου χρήστη. Πολλαπλών λειτουργικών περιβαλλόντων σημαίνει πως το webinos είναι προσαρμοσμένο έτσι ώστε να τρέχει σε διαφορετικά μεταξύ τους λειτουργικά περιβάλλοντα, καθώς παρά το γεγονός πως το αυτοκίνητο και η τηλεόραση τρέχουν διαφορετικά λογισμικά μπορούν και οι δύο συσκευές να είναι webinos-ενεργοποιημένες. Επίσης δικτυακά εφαρμόσιμες δηλώνει πως προγραμματίζεται χρησιμοποιώντας δικτυακές τεχνολογίες : html+css+javascript. Τέλος, ομογενοποιημένο σημαίνει πως διαφορετικά πεδία έχουν την δυνατότητα να ανταλλάσσουν μηνύματα, αρκετά όμοια με τη διαδικασία του ηλεκτρονικού ταχυδρομείου. {webinos.org}

4.1 Όραμα - Χαρακτηριστικά webinos

Το webinos οραματίζεται δικτυακές εφαρμογές να τρέχουνε και να υλοποιούνται σε ένα εύρος συνδεδεμένων συσκευών οι οποίες αλληλεπιδρούν απρόσκοπτα μεταξύ τους για να συνεισφέρουν στην ανάπτυξη του διαδικτύου και να διευκολύνουν τους χρήστες με πιο ελκυστικές, καινοτόμες και αξιόπιστες εφαρμογές. Για να το πετύχει αυτό, το webinos θα ορίσει και θα παρέχει μια πλατφόρμα ανοιχτής πηγής (open source platform) με πρακτική εφαρμογή και προσιτή σε όλους, της οποίας οι κύριοι στόχοι είναι [27]:

- Παροχή μιας ασφαλούς πλατφόρμας για δικτυακές τεχνολογίες σε κινητές συσκευές, οικιακά μέσα (TV), Η/Υ και συσκευές οχημάτων.
- Ορισμός και δόμηση τεχνολογικά τεκμηριωμένων βασικών εργαλείων, όπου αυτά δεν προϋπάρχουν, για την εύκολη και αποδοτική ανάπτυξη εφαρμογών σε διαφορετικές συσκευές
- Ορισμός και δόμηση ενός δικτυακά εφαρμόσιμου πλαισίου ασφαλείας το οποίο θα απευθύνεται στις ανάγκες των χρηστών και των παρόχων των υπηρεσιών, στις υποκείμενες πλατφόρμες και στις απαιτήσεις των συσκευών και θα είναι τεχνολογικά τεκμηριωμένο και εύκολα αναπτύξιμο.

Χαρακτηριστικά

Για την επίτευξη των παραπάνω στόχων το webinos , καταρχάς, παρουσιάζει τα παρακάτω κύρια χαρακτηριστικά [43], {webinos.org} :

- Το webinos βασίζεται στα επιτεύγματα της διαδικτυακής κοινότητας και επεκτείνει το open source web runtime environment.
- Το webinos προσφέρει ένα κοινό σύνολο διεπαφών προγραμματισμού εφαρμογών (APIs) έτσι ώστε να επιτρέπει εύκολη πρόσβαση σε cross-user, cross-service και cross-device λειτουργικότητα με έναν ανοιχτό και ασφαλή τρόπο.
- Το webinos στοχεύει στην διευκόλυνση του προγραμματισμού των εφαρμογών προσφέροντας μια ενιαία εικονική συσκευή που θα αποτελείται από το σύνολο των συσκευών του χρήστη

- Το webinos δημιουργεί ανοιχτές προδιαγραφές και open source reference implementations που δείχνουν την σκοπιμότητα των προδιαγραφών και απλοποιούν την προσαρμογή τους από την βιομηχανία.

4.2 Λειτουργίες - Καινοτομίες webinos

Για να πετύχει όλους τους παραπάνω στόχους το webinos χρησιμοποιεί τον ακόλουθο σχεδιασμό, προτείνοντας παράλληλα για την επίτευξή του τις απαραίτητες καινοτομίες:

Προσωπική Ζώνη (Personal Zone - PZ)

Ο κάθε χρήστης μπορεί να έχει περισσότερες από μία συσκευές ενεργοποιημένες στο webinos. Το σύνολο αυτών των συσκευών αποτελεί την Προσωπική του ζώνη (PZ). Η PZ δρα ως ένα ξεχωριστό δίκτυο σε σχέση με τα υποκείμενα φυσικά δίκτυα και πρωτόκολλα. Σκοπός της είναι να παρέχει εύκολη πρόσβαση σε τοπικές και απομακρυσμένες υπηρεσίες, απλοποιώντας το έργο ενός προγραμματιστή εφαρμογών. Επίσης πρέπει να επιτρέπει την εύκολη ανίχνευση συσκευών και υπηρεσιών και να παρέχει επικοινωνιακά μονοπάτια βασισμένα σε έμπιστες σχέσεις, αποσυνδεδεμένα από τις υποκείμενες τεχνολογίες

Κόμβος Προσωπικής Ζώνης (Personal Zone Hub - PZH)

Ο Κόμβος Προσωπικής Ζώνης (PZH) χρησιμοποιείται για τη δημιουργία μίας Προσωπικής Ζώνης. Είναι το κύριο στοιχείο πάνω στο οποίο συγχρονίζονται και ταυτοποιούνται όλες οι συσκευές ενός χρήστη. Σκοπός του πέρα από τη δημιουργία της Ζώνης είναι να επιτρέπει με ασφαλή τρόπο στις συσκευές να επικοινωνούν μεταξύ τους καθώς και την ανίχνευση και πρόσβαση σε συσκευές άλλων Προσωπικών Ζωνών.

Πρακτικά η υλοποίηση βασίζεται σε μία τοποθέτηση του PZH στο σύννεφο, υποθέτοντας ότι αυτή είναι ασφαλής, ακόμα και αν αυτή γίνεται σε εξυπηρετητή τρίτου. Οι επιτρεπτές λειτουργίες του PZH είναι μέσω της διεπαφής να προσθέτει μία συσκευή στην PZH , να ανακαλεί πιστοποιητικά για κάθε ταυτοποιημένη συσκευή και να ελέγχει την κατάσταση ταυτοποίησής της.

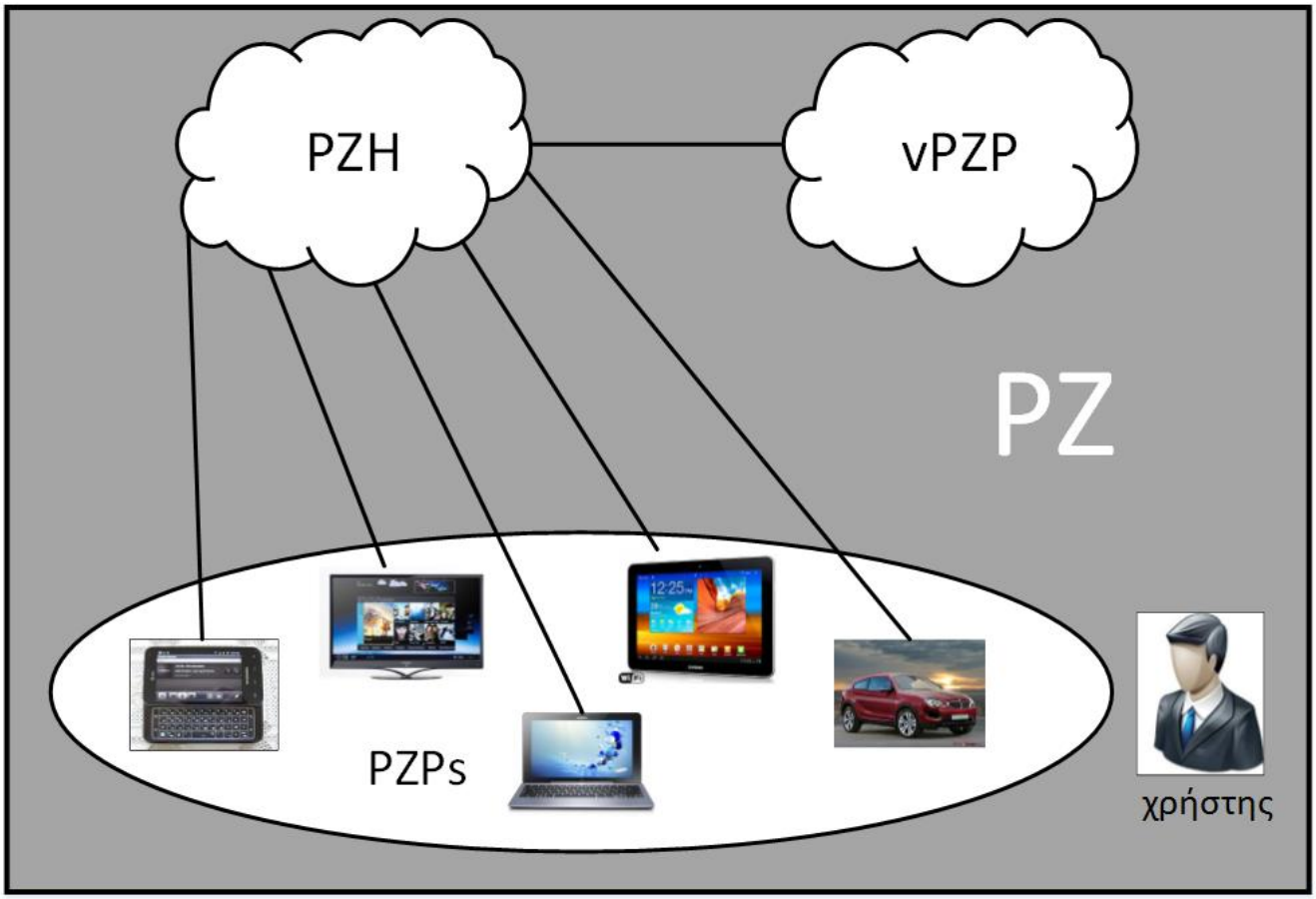
Πληρεξούσιο Προσωπικής Ζώνης (Personal Zone Proxy - PZP)

Ένα PZP τρέχει σε κάθε webinos ενεργοποιημένη συσκευή και είναι υπεύθυνο για την επικοινωνία της με το PZH.[] Αυτή επιτυγχάνεται μέσω ενός ασφαλούς TLS καναλιού

που ταυτοποιεί αμοιβαία και τις δύο πλευρές, και σε περίπτωση σφάλματος στην επικοινωνία μπορεί να γίνει έμμεση ταυτοποίηση μέσω της σύνδεσης του με ένα ήδη ταυτοποιημένο PZP. Στην περίπτωση που ο PZH δεν είναι διαθέσιμος λόγω σφάλματος στη σύνδεση με το δίκτυο, είναι δουλειά του PZP να λειτουργήσει στη θέση του και να αποθηκεύσει τις απαραίτητες πληροφορίες και γεγονότα συγχρονίζοντας αυτές αργότερα με το PZH, όταν η σύνδεση είναι πλέον δυνατή. Επιπρόσθετα, το PZP ευθύνεται για κάθε ανίχνευση χρησιμοποιώντας φορείς τοπικού υλικού. Ο βασικός του ρόλος είναι η εξαγωγή των APIs της συσκευής ως υπηρεσίες με έναν απόλυτα ασφαλή τρόπο. Ακόμα το PZP συνδέεται στον webinos RunTime (WRT) με μία διαμορφωμένη εφαρμογή διαδικτυακής υποδοχής η οποία ονομάζεται ασφαλής δίαυλος (secure channel). Το WRT είναι το μέρος όπου όλες οι ανεπτυγμένες εφαρμογές ανήκουν και τρέχουν. Τέλος, μετά τον ορισμό της εξωτερικής διεπαφής στο PZP, μπορούμε να χρησιμοποιήσουμε ένα ψεύδο-PZP για να κάνουμε δυνατή την πρόσβαση του webinos σε συσκευές, στις οποίες αυτό δεν μπορεί να εφαρμοστεί πλήρως, και με αυτό τον τρόπο να έχουμε πρόσβαση σε υπηρεσίες διαθέσιμες σε αυτές τις συσκευές.

Εικονικό Πληρεξούσιο Προσωπικής Ζώνης (virtual PZP)

Το εικονικό πληρεξούσιο προσωπικής ζώνης δημιουργήθηκε μεταγενέστερα των υπολοίπων συστατικών μίας προσωπικής ζώνης και ο ρόλος του αρχικά ήταν να «ελαφρύνει» της λειτουργίες του PZH. Αυτό διότι εξ αρχής η ιδέα ήταν το ενδιάμεσο λογισμικό να μην επιβαρύνει σημαντικά την υπολογιστική μνήμη των συσκευών. Για το σκοπό αυτό η υλοποίηση του γίνεται μέσα σε έναν διακομιστή του διαδικτύου και όχι απαραίτητα εντός της προσωπικής ζώνης ενός χρήστη. Οι περαιτέρω λειτουργίες του δεν έχουν αποσαφηνιστεί πλήρως, μιας και είναι ακόμα σε αρχικό στάδιο η σχεδίασή και η ανάπτυξή του. Η επικρατέστερη άποψη προβλέπει την δημιουργία διαφορετικών vPZP, τα οποία θα διαφοροποιούνται στο είδος των υπηρεσιών και λειτουργιών που το καθένα θα προσφέρει (π.χ. Context vPZP, Notification vPZP, Media vPZP κ.α.). Η κύρια ιδέα προβλέπει ακόμα την ομαδοποίηση των επιμέρους διαφορετικών vPZP, σε κοινούς χώρους του ίδιου διακομιστή (vPZP farms). Η μελέτη του θα μας απασχολήσει στη συνέχεια της εργασίας, καθώς εντός του context vPZP τοποθετείται η Context βάση δεδομένων, όπου αποθηκεύονται όλα τα δεδομένα περιβάλλοντος, της προσωπικής ζώνης ενός χρήστη.



Εικόνα VII: Βασική Ιδέα - Καινοτομίες webinos

Η ιδέα της προσωπική ζώνης ξεκίνησε και αναπτύχθηκε κατά τη φάση της σχεδίασης του webinos. Αυτό καθώς αποδείχτηκε ότι μπορεί να λειτουργήσει ως μία χρήσιμη παραδοχή που θα απλοποιεί τα πράγματα για τους σχεδιαστές εφαρμογών. Έτσι αυτοί θα μπορούν πιο εύκολα να διαχειριστούν την πολυπλοκότητα που έχει ένα σύστημα εφαρμογών πολλαπλών συσκευών που προορίζεται για χρήση ανάμεσα σε διαφορετικές πλατφόρμες. Αρχικά, είχε υποτεθεί πως κάθε προσωπική ζώνη θα περιελάμβανε μία συσκευή και θα είχε μόνο έναν ιδιοκτήτη. Αποτέλεσμα της υπόθεσης αυτής ήταν οι προσωπικές ζώνες να θεωρούνται μικρά, περισσότερο απομονωμένα συστήματα συσκευών όπου μόνο ένας χρήστης ήταν παρών κάθε στιγμή. Αυτό βοήθησε στον αρχιτεκτονικό σχεδιασμό και στην ασφάλεια - η αυθεντικοποίηση μπορούσε να βασιστεί εν μέρει στις ταυτότητες των συσκευών - και έτσι προχώρησε το πρόγραμμα σημαντικά.

Παρόλα αυτά, έγινε ξεκάθαρο πως πρόκειται για μία υπεραπλούστευση, αφού πολλές οικιακές συσκευές ανήκουν και χρησιμοποιούνται από κάποιους ανθρώπους από κοινού και δεν έχουν έναν μοναδικό ιδιοκτήτη (π.χ. ένα οικογενειακό PC ή μία τηλεόραση). Αν γινόταν η υπόθεση πως αυτά ήταν μέρη μίας ζώνης θα προκαλούνταν προβλήματα στις εφαρμογές και τις υπηρεσίες που θα απαιτούσαν χρησιμοποίηση ταυτοτήτων και

αυθεντικοποιήσεων. Για τους παραπάνω λόγους, ο σχεδιασμός διαφοροποιήθηκε στη συνέχεια του προγράμματος, ώστε να επιτρέπει στις συσκευές να τρέχουν σε διαφορετικά πληρεξούσια. Δυστυχώς, η προσέγγιση αυτή απαιτεί τα πληρεξούσια που βρίσκονται στην ίδια συσκευή να είναι απομονωμένα το ένα από το άλλο σε διαφορετικούς λογαριασμούς χρηστών. Οι λογαριασμοί αυτοί έχουν δείξει πως έχουν χαμηλή χρηστικότητα. Για παράδειγμα σε μία οικιακή ρύθμιση, αυτό απαιτεί κάθε χρήστης να είναι συνδεδεμένος ώστε να μπορούν να τρέξουν και να δεχτούν τα πληρεξούσιά του απομακρυσμένες εντολές. Επίσης δεν είναι δυνατή έτσι, η λειτουργία του webinos σε συσκευές και συστήματα που δεν παρέχουν την δυνατότητα χρησιμοποίησής τους από πολλαπλούς λογαριασμούς. Έτσι καταλήξαμε στη λύση των πολλαπλών πληρεξουσίων ανά συσκευή, η οποία είναι αποδεκτή προς το παρόν, αν και μελλοντικά απαιτεί περαιτέρω ανάπτυξη και μία καλύτερη υλοποίηση.

Η παραπάνω εμπειρία μπορούμε να πούμε ότι μας δίδαξε κάποια πράματα. Αρχικά, ότι η έννοια του *χρήστη* και του *λογαριασμού χρήστη* αλλάζει και πολλές φορές η αοριστία μίας τέτοιας έννοιας δεν βοηθάει κατά την εύρεση τεχνικής λύσης, ειδικά όταν καλούμαστε να αντιμετωπίσουμε διαφορετικούς τύπους συσκευών. Ακόμα, ότι η πρόωρη εισαγωγή μίας αόριστης έννοιας στο σύστημα μπορεί να αποδειχτεί τόσο θετική όσο και αρνητική. Από την μια μπορεί να βοηθήσει στην ανάπτυξη, αλλά από την άλλη μπορεί και να περιορίσει την ικανοποίηση των απαιτήσεων σε κάποιες από τις περιπτώσεις χρήσης, και αυτό με απρόβλεπτο τρόπο. Τέλος, έρευνες έχουν δείξει, πως οι προγραμματιστές που θέλουν να σχεδιάσουν μία εφαρμογή πάνω στο webinos, και οι οποίοι δεν είναι και τόσο εξοικειωμένοι με την πλατφόρμα, πιθανότατα να δυσκολευτούν ιδιαίτερα, δεδομένου πως η τεχνική υλοποίηση ενός πληρεξουσίου προσωπικής ζώνης και η κατάλληλη προσαρμογή του αποτελούν ένα δύσκολο πρόβλημα.

4.3 Τα APIs του webinos

Για την επίτευξη του στόχου της ανάπτυξης context aware εφαρμογών, που θα μπορούν να τρέχουν τόσο σε διαφορετικές πλατφόρμες, όσο και σε διαφορετικές συσκευές είναι απαραίτητη η πρόσβαση του προγραμματιστή σε ένα κατανεμημένο περιβάλλον από Application Programming Interfaces (APIs). Με την χρήση ενός πλαισίου που παρέχει τέτοια πρόσβαση και σε συνδυασμό με τις context – aware ικανότητες του webinos, η δυνατότητα σχεδίασης τέτοιων εφαρμογών γίνεται πραγματικότητα.

Οι εφαρμογές που θα τρέχουν μέσα σε μία προσωπική ζώνη του webinos θα μπορούν, πολύ εύκολα, να μοιράζονται την κατάσταση της ίδιας εφαρμογής η οποία πιθανώς να τρέχει σε μία άλλη συσκευή του ιδίου χρήστη, με τον ίδιο ακριβώς τρόπο που ανταλλάσουν πληροφορίες εντός της προσωπικής ζώνης. Επιπρόσθετα, η χρήση των κλήσεων απομακρυσμένης διαδικασίας (Remote Process Calls – RPCs), που θα πραγματοποιούνται από το Webinos Run Time (WRT) στο εκτιθέμενο API άλλων συσκευών εντός της προσωπικής ζώνης, θα επιτρέπει στους προγραμματιστές να αυξάνουν το εύρος των λειτουργιών της εφαρμογής τους.

Αφού έγιναν έρευνες πάνω σε σενάρια, περιπτώσεις χρήσης και απαιτήσεις, κατά την διάρκεια του πρώτου μισού χρόνου του προγράμματος καταλήξαμε ότι η αρχιτεκτονική του webinos απαιτεί την ύπαρξη μίας ομάδας API συσκευών. Αρχικά χρησιμοποιήθηκαν πολλά API τα οποία ήδη υπήρχαν και μπορούσαν να καλύψουν κάποιες ανάγκες. Στη συνέχεια όμως το webinos προχώρησε στον σχεδιασμό νέων API τα οποία θα κάλυπταν συγκεκριμένες απαιτήσεις. Ο νέος σχεδιασμός έγινε προσπαθώντας πάντα να μην απομακρυνθούμε πολύ από τα ήδη υπάρχοντα - πρωτότυπα API.

Τα API που σχεδιάστηκαν και παρέχονται από το webinos είναι:

WEBINOS CORE INTERFACE

Προσδιορίζει την κοινή διεπαφή μέσω της οποίας μπορεί ο χρήστης να έχει πρόσβαση σε όλα τα API του webinos και σε όλες τις πληροφορίες της προσωπικής ζώνης

Αυτή η έκδοση ορίζει:

1) Την κεντρική διεπαφή του webinos, μέρος του παγκόσμιου window αντικειμένου.
2) Τις πληροφορίες της προσωπικής ζώνης, όπως η κατάσταση σύνδεσης, το όνομα του προσωπικού κόμβου, το φιλικό όνομα του πληρεξουσίου της προσωπικής ζώνης και το αναγνωριστικό εφαρμογών.

APPLAUNCHER API

Επιτρέπει την ενεργοποίηση των webinos εφαρμογών τοπικά στην συσκευή
Δυνατότητες χρήστη:

- Ενεργοποίηση/απενεργοποίηση των εγκατεστημένων webinos εφαρμογών
- Ενεργοποίηση/απενεργοποίηση κοινοποίησης στους χρήστες όταν μία webinos εφαρμογή προσπαθεί να ενεργοποιήσει μία άλλη εφαρμογή.
- Ενεργοποίηση/απενεργοποίηση της δυνατότητας της εφαρμογής να ανιχνεύει εγκατεστημένες εφαρμογές.

Η εκτέλεση του API παρέχει μηχανισμούς στις webinos εφαρμογές ώστε να ελέγχουν αν μια συγκεκριμένη webinos εφαρμογή έχει εγκατασταθεί στην συσκευή.

APPSTATE SYNCHRONISATION API

Αποτελεί μία διεπαφή για ενεργοποίηση και διαχείριση συγχρονισμού εφαρμογών. Η διεπαφή παρέχει ένα σετ λειτουργιών για την διαχείριση των κοινόχρηστων συγχρονισμένων αντικειμένων. Αφαιρεί από την τρέχουσα εφαρμογή μηχανισμούς ανταλλαγής δεδομένων και διευκολύνει την ανάπτυξη διεσπαρμένων εφαρμογών, μειώνοντας την πολυπλοκότητα σε απλές διάβασε και γράψε λειτουργίες των ιδιοτήτων των αντικειμένων. Η διεπαφή περιλαμβάνει δημιουργία και αναζήτηση για αντικείμενα στην μορφολογία ανταλλαγής δεδομένων JSON και εγγραφή ή διαγραφή για αλλαγή κατάστασης. Οι αλλαγές των αντικειμένων που δημιουργούνται από αυτή την διεπαφή θα ανιχνεύονται και θα συγχρονίζονται αυτόματα σε όλους τους συμμετέχοντες, π.χ. η ρύθμιση των ιδιοτήτων ή η αλλαγή των τιμών θα επηρεάζει όλα τα αντίγραφα των αντικειμένων. Ο τρόπος συγχρονισμού μπορεί να ρυθμιστεί ώστε οι συμμετέχοντες να είναι ξεχωριστές οντότητες της ίδιας εφαρμογής που τρέχει στις διαφορετικές συσκευές του χρήστη (π.χ. διαμοιρασμένες εφαρμογές σε διάφορες συσκευές). Με περαιτέρω ρύθμιση, ένα

αντικείμενο μπορεί να συγχρονιστεί παράλληλα για πολλούς διαφορετικούς χρήστες ή εφαρμογές.(π.χ. συνομιλία ανάμεσα σε εφαρμογές).

AUTHENTICATION API

Ο ρόλος του API ταυτοποίησης είναι να παρέχει στις εφαρμογές πληροφορίες για το εάν ο τρέχων χρήστης έχει ταυτοποιηθεί καθώς και να ζητάει επαναταυτοποίηση, άμα χρειαστεί, κατά τη διαρκεί λειτουργίας της εφαρμογής. Σκόπιμα δεν αποκαλύπτει πληροφορίες ταυτότητας του χρήστη, αλλά μπορεί να δώσει λεπτομέρειες για την μέθοδο ταυτοποίησης, οι οποίες μπορούν να αποκαλύψουν πληροφορίες για την συσκευή.

CONTACTS API

Το API επαφών καθορίζει τις υψηλού επιπέδου διεπαφές που χρειάζονται για να υπάρξει πρόσβαση στην ενοποιημένα ατζέντα επαφών ενός χρήστη. Για το σκοπό αυτό χρησιμοποιεί δύο διαφορετικές διεπαφές: Μία διεπαφή επαφών που παρέχει τη μέθοδο πρόσβασης στην ατζέντα και μία διεπαφή επαφών η οποία συλλέγει τις ξεχωριστές πληροφορίες κάθε επαφής, οι οποίες μπορούν να επιστραφούν ύστερα από μία πετυχημένη λειτουργία ανάγνωσης.

CONTEXT API

Το API πλαισίου καθορίζει τις διεπαφές υψηλού επιπέδου που είναι απαραίτητες για να υπάρξει πρόσβαση στα δεδομένα πλαισίου του χρήστη. Ο χρήστης πρέπει να ενεργοποιήσει τον διαχειριστή πλαισίου στην συσκευή του για να μπορεί να παρακολουθήσει τα αντικείμενα πλαισίου του. Το API αυτό υποστηρίζει δύο βασικούς τρόπους πρόσβασης:

1. Μέσω ερώτησης στον αποθηκευτικό χώρο context δεδομένων και ανακτώντας το δεδομένο από τα αποτελέσματα της ερώτησης.
2. Με εγγραφή ώστε να λαμβάνονται αμέσως context δεδομένα μόλις ένα γεγονός πραγματοποιηθεί (μέσω του Διαχειριστή Context Εντολών)

Το API παρέχει και την δυνατότητα να προγραμματίσει κλήσεις API, με σκοπό την ανανέωση των διαθέσιμων context δεδομένων, ακόμα και αν η εφαρμογή δεν λειτουργεί.

DEVICE INTERACTION API

Το μοντέλο αυτό παρέχει ένα μηχανισμό για την αλληλεπίδραση με τον τελικό χρήστη, μέσω ιδιοτήτων όπως:

- Δόνηση
- Ηχητική ειδοποίηση
- Φωτεινή ένδειξη
- Ενδείξεις φόντου

DEVICE STATUS API

Αυτό το API χρησιμοποιεί ένα μοντέλο δένδρου για να επιτρέπει στους προγραμματιστές να έχουν πρόσβαση στα διάφορα κομμάτια πληροφορίας σε μία συσκευή. Σε αυτό το μοντέλο χρησιμοποιεί τους όρους: Αντικείμενο, Εξάρτημα και Ιδιότητα. Όπου ένα αντικείμενο μπορεί να περιέχει ένα ή περισσότερα εξαρτήματα, και ένα εξάρτημα μία ή περισσότερες ιδιότητες. Ακόμα γίνεται χρήση και δύο ειδικών εξαρτημάτων: "_default" and "_active", τα οποία μπορούν να χρησιμοποιηθούν από τον προγραμματιστή ως πληρεξούσια στο API.

DISCOVERY API

Χρησιμοποιείται όχι μόνο για την ανακάλυψη τοπικών υπηρεσιών, αλλά και για την ενεργοποίηση της ανίχνευσης απομακρυσμένων υπηρεσιών. Συγκεκριμένα ενεργοποιεί την δυνατότητα ανίχνευσης εκτεθειμένων υπηρεσιών:

- Στη συσκευή
- Σε οντότητες απευθείας συνδεόμενες στη συσκευή
- Σε οντότητες διαθέσιμες στο ίδιο τοπικό δίκτυο
- Σε ασφαλείς υπηρεσίες καταχωρημένες σε μία προσωπική ζώνη

Αυτό το επιτυγχάνει με την χρήση ενός *αντικειμένου υπηρεσιών* (service object) και την δημιουργία ενός ασφαλούς διαύλου επικοινωνίας. Εφόσον έχει ήδη εγκατασταθεί η εφαρμογή και ο χρήστης της συσκευής είναι ταυτοποιημένος και εξουσιοδοτημένος να χρησιμοποιήσει το API.

THE GENERIC ACTUATOR API

Το API αυτό παρέχει στις εφαρμογές ένα API για να ελέγχουν τους ενεργοποιητές της ίδιας της συσκευής, συνδεδεμένες με τη συσκευή ή μίας άλλης συσκευής. Το API δεν γνωρίζει τις υποκείμενες μεθόδους για να ανιχνεύει και να επικοινωνεί με τους ενεργοποιητές, για το λόγο αυτό θα πρέπει να χρησιμοποιείται σε συνδυασμό με υπηρεσίες ανίχνευσης και σύνδεσης. Οι υπηρεσίες ενός ενεργοποιητή μπορούν να εντοπιστούν στην προσωπική ζώνη του χρήστη ή να μοιραστούν με το υπάρχον δίκτυο. Αυτή τη στιγμή αρκετοί διαφορετικοί ενεργοποιητές ορίζονται, αλλά το API μπορεί εύκολα να επεκταθεί με επιπλέον τύπους ενεργοποιητών.

Αυτό είναι ένα πειραματικό API και θέματα ασφάλειας και ιδιωτικότητας δεν έχουν ακόμα διευκρινιστεί. Αν γίνει δυνατή η πρόσβαση σε ενεργοποιητές ιδιωτικού απορρήτου ή ασφάλειας, ο πράκτορας του χρήστη πρέπει είτε να πάρει άδεια πρόσβασης μέσω της διεπαφής του χρήστη, είτε να ελέγχει την πρόσβαση μέσω μίας προσυμφωνημένης σχέσης εμπιστοσύνης με τους χρήστες.

App2App MESSAGING API

Το API αυτό καθορίζει τις κατάλληλες διεπαφές για την δημιουργία, αποστολή και λήψη μηνυμάτων ανάμεσα σε εφαρμογές στο webinos. Παρέχει γενικά αρχέτυπα μηνυμάτων τα οποία μπορούν να χρησιμοποιηθούν σε διαφορετικά σενάρια εφαρμογών. Τα μηνύματα είναι έμμεσα, το οποίο σημαίνει ότι οι εφαρμογές δεν απευθύνονται η μία στην άλλη. Αντί αυτού χρησιμοποιούν ένα κανάλι επικοινωνίας για να κατευθύνουν τα μηνύματα στις συνδεδεμένες εφαρμογές, οι οποίες χρησιμοποιούν ένα μοναδικό όνομα ως κλειδί για την εύρεση και σύνδεση με το κανάλι. Το συγκεκριμένο API μπορεί να χρησιμοποιηθεί και από τρίτους, προγραμματιστές εφαρμογών, οι οποίοι εκμεταλλευόμενοι και τις ιδιότητες που παρέχει το webinos, με το σύστημα διαχείρισης μηνυμάτων και το υποκείμενο δικτυακό μοντέλο, μπορούν να παρέχουν προσαρμοσμένα, με βάση τα μηνύματα, πρωτόκολλα.

MESSAGING API

Το API μηνυμάτων δίνει πρόσβαση στις παρακάτω δυνατότητες:

- Αποστολή, μηνυμάτων μέσω διαφορετικών τεχνολογιών (SMS, MMS, IM, email)

- Αναζήτηση μηνυμάτων σε διαφορετικούς φακέλους
- Εγγραφή για ειδοποιήσεις ενόψει προσεχών γεγονότων μηνυμάτων.

Το API αυτό είναι ανάγνωσης μόνο και δεν επιτρέπει τη διαχείριση μηνυμάτων ή φακέλων.

MEDIACONTENT API

Αυτό το API παρέχει λειτουργίες για την ανίχνευση περιεχομένου πολυμέσων (όπως βίντεο, εικόνες, μουσική, κλπ) που είναι διαθέσιμα στην συσκευή. Είναι δυνατή και η αναζήτηση συγκεκριμένων αρχείων πολυμέσων με την χρήση φίλτρων. Επίσης το API αυτό υποστηρίζει την ρύθμισή ορισμένων χαρακτηριστικών πολυμέσων. Στο webinos εμφανίζεται ελαφρώς αλλαγμένο με την προσθήκη μιας μεθόδου για την ακύρωση ασύγχρονων λειτουργιών.

NAVIGATION API

Το API πλοήγησης του webinos παρέχει έναν μηχανισμό αλληλεπίδρασης με εγκατεστημένα λογισμικά πλοήγησης. Είναι υποπροϊόν του API οχήματος, καθώς καθορίζει τις λειτουργίες αλληλεπίδρασης με υπηρεσίες δορυφορικής πλοήγησης. Ακόμα χρησιμοποιεί μια μέθοδο παροχής σημείων ενδιαφέροντος στην υπηρεσία παρακολουθώντας μέσω αυτών αν η πλοήγηση παραμένει ενεργή.

NFC API

Near Field Communication (NFC) είναι ένα διεθνές πρότυπο (ISO/IEC 18092) το οποίο καθορίζει μία διεπαφή και ένα πρωτόκολλο για απλή ασύρματη διασύνδεση από ζεύγη συσκευών που βρίσκονται σε κοντινή απόσταση και λειτουργούν στα 13.56 MHz (http://www.nfc-forum.org/specs/spec_list/). Υπάρχουν τρεις ομάδες από πιθανά σενάρια εφαρμογών για NFC: Το πρώτο είναι να έχεις μία συσκευή κοντά σε μια ασύρματη ετικέτα με σκοπό την ανταλλαγή ψηφιακών πληροφοριών ή δεδομένων. Το δεύτερο είναι να κρατάς δύο συσκευές κοντά η μία στην άλλη για να γίνει μεταξύ τους ανταλλαγή δεδομένων ή πληροφοριών. Η τρίτη είναι η διεκπεραίωση πληρωμών μέσω κινητών τηλεφώνων, κρατώντας τα κοντά σε σημεία πωλήσεων, αντί να χρησιμοποιείς κάποιου είδους κάρτα.

PAYMENT API

Αυτό το API παρέχει γενικές λειτουργίες ηλεκτρονικής αγοράς με σκοπό να είναι δυνατή η πληρωμή εντός της εφαρμογής. Δεν είναι συνδεδεμένο με ένα συγκεκριμένο πάροχο υπηρεσιών ηλεκτρονικών πληρωμών και είναι σχεδιασμένο ώστε να μπορεί να λειτουργήσει με διάφορες υπηρεσίες όπως GSMA, OneAPI, BlueVia, Android Payment API και PayPal..

THE REMOTE UI API

Οποιαδήποτε συσκευή που χρησιμοποιεί το webinos μπορεί να παρέχει πρόσβαση στην διεπαφή του χρήστη της με τον ίδιο τρόπο που μπορεί να παρέχει οποιαδήποτε άλλη υπηρεσία. Η πρόσβαση στην υπηρεσία διέπεται από την πολιτική των ρυθμίσεων και στην τοπική και στην απομακρυσμένη συσκευή. Δεν δημιουργείται και δεν θα δημιουργηθεί απευθείας πρόσβαση σε απομακρυσμένα αντικείμενα. Βασικές τιμές από το απομακρυσμένο DOM μπορούν να συλλεχθούν, αλλά όχι αντικείμενα.

SECURE ELEMENT API

Το μοντέλο αυτό επιτρέπει την επικοινωνία ανάμεσα στην εφαρμογή του δικτύου και μιας «έξυπνης» κάρτας ή άλλων ασφαλών στοιχείων με την χρήση του Πρωτοκόλλου Εφαρμογής Μονάδων Δεδομένων (APDU). Το APDU είναι ένα σύντομο μήνυμα που αναπαριστάται από bytes. Τα μηνύματα APDU είναι είτε εντολές είτε αποκρίσεις.

TV CONTROL API

Αυτή η διεπαφή παρέχει τα μέσα για να αποκτηθεί μία λίστα πηγών τηλεόρασης, καναλιών κλπ. Το τηλεοπτικό κανάλι μπορεί να προβληθεί σε HTMLVideoElement object. Εναλλακτικά το API παρέχει τα μέσα για να ελέγχουμε την διαχείριση των καναλιών του προϋπάρχοντος υλικού για την προβολή τηλεόρασης, επιτρέποντας να θέσουμε ένα κανάλι ή να παρακολουθήσουμε για αλλαγές στα κανάλια που προκαλούνται με διαφορετικό τρόπο. Το αντικείμενο της τηλεόρασης είναι διαθέσιμο στο χώρο του webinos , π.χ. webinos.tv

VEHICLE API

Αυτό το API προβάλλει δεδομένα οχήματος, τα οποία είναι διαθέσιμα σε ένα δίαυλο του οχήματος. Αυτός ο δίαυλος είναι συνήθως το σημείο πρόσβασης από όπου η κεντρική μονάδα συλλέγει τα δεδομένα του οχήματος. Κάποια δεδομένα από άλλους δίαυλους κατευθύνονται στον δίαυλο πάνω από την κεντρική οδό, όπως η ταχύτητα.

WEBINOS WIDGET API

Ορίζει την κοινή widget διεπαφή. Βασίζεται στα χαρακτηριστικά του W3C Widget Specifications και συμπληρώνει την W3C Widget Interface.

THE GENERIC SENSOR API

Το API δεν γνωρίζει τις υποκείμενες μεθόδους για να ανιχνεύει και να επικοινωνεί με τους αισθητήρες, για το λόγο αυτό θα πρέπει να χρησιμοποιείται σε συνδυασμό με υπηρεσίες ανίχνευσης και σύνδεσης. Οι υπηρεσίες ενός αισθητήρα μπορούν να εντοπιστούν στην προσωπική ζώνη του χρήστη ή να μοιραστούν με το υπάρχον δίκτυο.

Το API αποτελείται από δύο διεπαφές:

- Μια διεπαφή αισθητήρα που παρέχει χαρακτηριστικά για τους αισθητήρες και μια μέθοδο για τη διαμόρφωση ενός επιλεγμένου αισθητήρα.
- Ένα επιπέδου 3 DOM γεγονός που παρέχει δεδομένα αισθητήρα.

Αυτή η λίστα με τύπους αισθητήρων που υποστηρίζονται από το API μπορεί εύκολα να επεκταθεί με επιπρόσθετους τύπους αισθητήρων.

THE Web NOTIFICATIONS API

Αυτό το API ειδοποιήσεων βασίζεται στο αντίστοιχο του W3C. Δεν προσδιορίζει ακριβώς πως το μέσο ενός χρήστη θα προβάλει αυτές τις ειδοποιήσεις, η καλύτερη αναπαράσταση εξαρτάται από την συσκευή στην οποία τρέχει το μέσο. Όταν αναφέρεται αυτό το API σε αναπαράσταση ειδοποιήσεων στην επιφάνεια εργασίας, γενικά εννοεί μία

στατική περιοχή εξωτερικά της περιοχής αναπαράστασης της εφαρμογής, αλλά μπορεί να πάρει αρκετές μορφές, στις οποίες συμπεριλαμβάνονται:

- Μία γωνία της οθόνης του χρήστης.
- Μία περιοχή μέσα στο χρώμιο του μέσου του χρήστη.
- Την κεντρική οθόνη μίας κινητής συσκευής.

Δεν ορίζει ακριβώς το τρόπο με τον οποίο το μέσο ενός χρήστη θα προβάλει την ειδοποίηση, και είναι σχεδιασμένο ώστε να είναι ευέλικτο στις διαφορετικές επιλογές αναπαράστασης. Είναι σχεδιασμένο ώστε να είναι συμβατό, όσο αυτό είναι δυνατό, με τις ήδη υπάρχουσες πλατφόρμες και τεχνολογίες ειδοποιήσεων, αλλά και να μην εξαρτάται από αυτές. Εφ' όσον οι κοινές πλατφόρμες δεν παρέχουν την ίδια λειτουργία, αυτό θα παρέχει ένδειξη για το ποια γεγονότα είναι εγγυημένα και ποια όχι. Εν προκειμένω, οι ενδείξεις όπως περιγράφονται εδώ μπορούν να περιέχουν μόνο κείμενο και εικονίδια. Στο μέλλον, οι ειδοποιήσεις προερχόμενες από περιεχόμενο διαδικτύου θα μπορούν να περιέχουν και οι ίδιες περιεχόμενο του διαδικτύου, αλλά αυτό είναι ένα άλλο θέμα.

Εν αντιθέσει με την πρωτότυπη έκδοση του W3C, το μοντέλο γεγονότων ειδοποιήσεων είναι αξιόπιστο. Το αντικείμενο ειδοποιήσεων προσφέρει ένα click-γεγονός και οι εφαρμογές μπορούν να αυξήσουν τις λειτουργίες τους ακούγοντας για αυτό το γεγονός.

Τα παρακάτω προϋπάρχοντα API τα χρησιμοποιεί το webinos έτοιμα και μη τροποποιημένα (referenced APIs):

THE W3C CALENDAR MODULE

Ορίζεται το API που θα παρέχει πρόσβαση στον χρήστη σε υπηρεσίες και εφαρμογές ημερολογίου.

THE W3C DEVICE ORIENTATION – EVENT SPECIFICATION

Ορίζει αρκετά νέα DOM (Document Object Models) τύπου γεγονότα, τα οποία παρέχουν πληροφορίες για τον φυσικό προσανατολισμό και την κίνηση της συσκευής που χρησιμοποιείται.

THE W3C FILE API

Χρησιμοποιείται για την αναπαράσταση αρχείων σε δικτυακές εφαρμογές, όπως για την επιλογή τους και πρόσβαση στα δεδομένα τους.

THE W3C FILE API: WRITER

Εξαρτάται και από άλλα API, όπως το προηγούμενο, και είναι σχεδιασμένο ώστε να χρησιμοποιείται σε συνδυασμό μαζί τους. Όπως λέει και το όνομα του ορίζει τον τρόπο με τον οποίο μπορούμε να γράφουμε σε αρχεία δικτυακών εφαρμογών.

THE W3C FILE API: DIRECTORIES AND SYSTEM

Όπως και το παραπάνω εξαρτάται και χρησιμοποιείται σε συνδυασμό με το W3C File Writer API, το οποίο με τη σειρά του δομείται πάνω στο W3C File API. Το συγκεκριμένο ορίζει τον τρόπο πλοήγησης και ιεραρχίας σε αρχεία δικτυακών εφαρμογών.

THE W3C GEOLOCATION API

Παρέχει πρόσβαση σε πληροφορίες γεωγραφικής τοποθεσίας που σχετίζονται με την εν ενεργεία συσκευή.

THE W3C MEDIA CAPTURE AND STREAMS API

Το συγκεκριμένο API παρέχει πρόσβαση του χρήστη σε υπηρεσίες της ενεργοποιημένης συσκευής που σχετίζονται με το τράβηγμα φωτογραφίας ή βίντεο και εν γένει όλες τις δυνατότητες μιας συσκευής να συλλέξει οπτικοακουστικό υλικό.

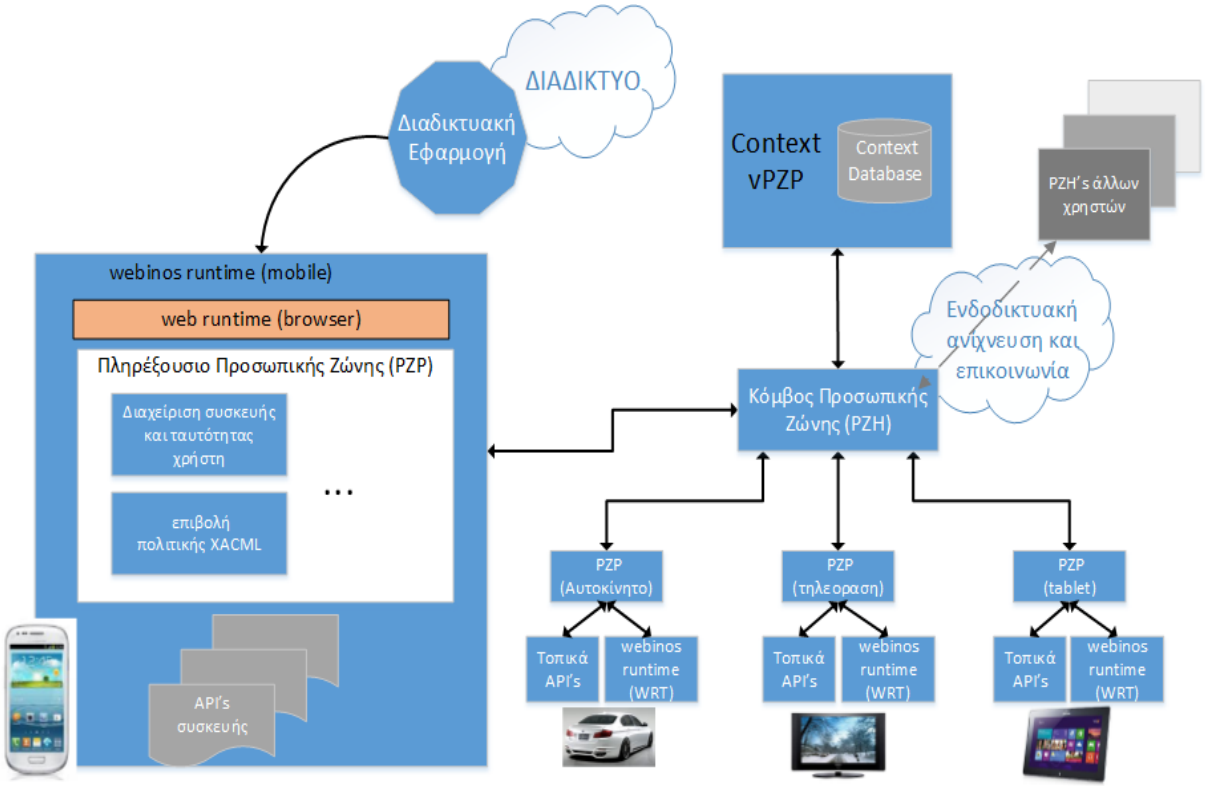
THE W3C WebRTC API

Επιτρέπει την δημιουργία ομότιμης (peer to peer) σύνδεσης ανάμεσα σε διαφορετικά προγράμματα περιήγησης, καθώς και την μετάδοση δεδομένων μέσω αυτών των συνδέσεων.

Τα παραπάνω API θα χρησιμοποιούνται από τις webinos ενεργοποιημένες εφαρμογές μέσω των παρακάτω διαδικασιών του webinos:

Η εφαρμογή που θα τρέχει στην πλατφόρμα του webinos θα έχει πρόσβαση στα API μέσω ενός JavaScript δεσμού. Παρολαυτά, αυτό δεν σημαίνει ότι θα υπάρχει άμεση πρόσβαση στις υποκείμενες λειτουργίες της συσκευής, αλλά ότι η επικοινωνία θα γίνεται μέσω μίας κλήσης JSON RPC στο αντίστοιχο PZP. Το PZP θα έχει εφαρμοστεί ως κώδικας node.js, έχοντας έτσι τη δυνατότητα παροχής πρόσβασης στα χαρακτηριστικά και στις λειτουργίες της συσκευής. Αυτή η αρχιτεκτονική προσέγγιση παρουσιάζει τα παρακάτω πλεονεκτήματα:

- Όλες οι κλήσεις API γίνονται μέσω μίας διεπαφής, η οποία μπορεί να αποτελέσει σημείο όπου θα επιβάλλεται η αντίστοιχη πολιτική από τα πλαίσια πολιτικής και ασφάλειας.
- Ορίζεται μία RPC διασύνδεση για όλες τις λειτουργίες των API. Έτσι η μέθοδος πρόσβασης στα API από απομακρυσμένες συσκευές δεν διαφέρει σημαντικά από την αντίστοιχη των τοπικών API, ενισχύοντας έτσι την διαδραστικότητα και την διαλειτουργικότητα του συστήματος.
- Εφόσον ο κώδικας της συσκευής εντοπίζεται κυρίως στο PZP και η επικοινωνία σε αυτό γίνεται μέσω γνωστών πρωτοκόλλων, η ανάγκη για επέκταση του προγράμματος περιήγησης μειώνεται, ενισχύοντας έτσι την ευελιξία του περιβάλλοντος του webinos.



Εικόνα VIII: Αρχιτεκτονική webinos

5 Webinos Context

5.1 **Διαχειριστής Context webinos**

5.1.1 **Εισαγωγή**

Ο Διαχειριστής Context του webinos συλλέγει, αποθηκεύει και κάνει διαθέσιμο το context εντός της πλατφόρμας του webinos. Κατά τη λειτουργία του καλείται να ανταποκριθεί στα παρακάτω ζητήματα [16]:

- Αναγνώριση των σχετικών context δεδομένων ανάλογα με την κάθε λειτουργία.

- Ανίχνευση context δεδομένων μέσα στο webinos (μέσω των οντοτήτων του webinos) και σχετικών Context Πηγών, *χρησιμοποιώντας για την ταυτοποίηση ήδη υπάρχοντα API.*

- Απόκτηση Context δεδομένων για το σύστημα του webinos, *χρησιμοποιώντας για την ταυτοποίηση ήδη υπάρχοντα API.*

- Αναπαράσταση Context δεδομένων *μέσα στο πλαίσιο του ορισμού του webinos για τις απαραίτητες δομές context σε συγκεκριμένες μορφές δεδομένων.*

- Αποθήκευση των context δεδομένων μέσα στο webinos, *και η δημιουργία κατάλληλων υποδομών για το σκοπό αυτό*.

- Διανομή των Context δεδομένων στις διάφορες οντότητες και λειτουργίες του webinos , *για την οποία τα χαρακτηριστικά των απαραίτητων API θα δημιουργηθούν.*

Το context πλαίσιο του webinos παρέχει πρόσβαση σε δεδομένα τύπου context, με σκοπό να κάνει δυνατό τον σχεδιασμό και την λειτουργία των webinos εφαρμογών/υπηρεσιών που τα χρησιμοποιούν. Το πλαίσιο είναι υπεύθυνο για την συλλογή

και αποθήκευση context δεδομένων (μέσω της αναγνώρισης ειδικών γεγονότων σχετικών με context που συμβαίνουν σε webinos-ενεργοποιημένες συσκευές) και την παροχή των εφαρμογών με πρόσβαση σε αυτά τα δεδομένα, μέσω ενός στρώματος, είτε με τη μορφή επερωτήσεων στην μνήμη είτε με ειδοποιήσεις πραγματικού χρόνου (όταν συμβαίνουν ειδικά γεγονότα).

Μια context-aware εφαρμογή χρησιμοποιεί είτε ένα κομμάτι, είτε συνδυασμό είτε μία αλληλουχία δεδομένων σχετικών με το context. Για παράδειγμα, η τρέχουσα κατάσταση της πυξίδας ενός κινητού τηλέφωνο ή ενός αισθητήρα φωτός ενός φορητού Η/Υ, η γεωγραφική τοποθεσία και τα πλησιέστερα διαθέσιμα ασύρματα δίκτυα, είναι context πληροφορίες χρήσιμες ώστε να είναι δυνατή μία αυτόματη προσαρμογή (π.χ. η αύξηση της φωτεινότητας της οθόνης, η ενεργοποίηση της ασύρματης σύνδεσης, κλπ). Για να μπορέσει να εκμεταλλευτεί, αυτές τις λειτουργίες, ο χρήστης-προγραμματιστής πρέπει να έχει πρόσβαση στα μέσα συλλογής, προμήθειας, φιλτραρίσματος, συνδυασμού και εκτέλεσης εντολών context δεδομένων. Για καθολικές, κατανεμημένες και context-aware εφαρμογές, ο στόχος είναι η παροχή του κατάλληλου ενδιάμεσου λογισμικού, τέτοιου ώστε να μπορεί να εκτελεί Απομακρυσμένες Διαδικαστικές Κλήσεις (RPCs), ενώ παράλληλα θα εισάγει ένα αφαιρετικό στρώμα που θα διευκολύνει της διαδικασία ανάπτυξης, αποκρύπτοντας την ετερογένεια και την διαφορετικότητα των περιβαλλόντων του δικτύου, υποστηρίζοντας ένα προηγμένο μοντέλο συντονισμού ανάμεσα σε διανεμημένες οντότητες και κάνοντας όσο πιο διαφανή γίνεται, την κατανομή των υπολογισμών.

Το project του webinos στοχεύει τόσο στην παροχή ενός αφαιρετικού επιπέδου πολλαπλής πλατφόρμας για διαδικαστικές κλήσεις, όσο και στην παράλληλη ενσωμάτωση ενός επιπλέον αφαιρετικού στρώματος δεδομένων για χρήση από context-aware και context-sharing εφαρμογές τρίτων, που είναι ενεργοποιημένες στο webinos.

Το κύριο δομικό στοιχείο σε ότι σχετίζεται με context πληροφορία στο webinos είναι το Context Αντικείμενο. Ως context αντικείμενο ορίζεται η ελάχιστη μονάδα πληροφορίας ή δεδομένου που καθορίζει και είναι αρκετή για να περιγράψει ένα κομμάτι πληροφορίας τύπου context. Για παράδειγμα, ενώ μία κλήση σε ένα GPS μπορεί να επιστρέψει ένα πλήθος πληροφοριών ως απαντήσεις (γεωγραφικό μήκος, γεωγραφικό πλάτος, κατεύθυνση, ταχύτητα, ακρίβεια, υψομετρική ακρίβεια κλπ), ένα σχετικό context αντικείμενο το οποίο μπορεί να λέγεται MyLocation θα περιέχει μόνο αυτές τις πληροφορίες που είναι απαραίτητες για τον καθορισμό της μονάδας της πληροφορίες που ζητείται. Αυτό θα επιτυγχάνεται με τον αποκλεισμό/φιλτράρισμα κάποιων δεδομένων ή και με την προσθήκη

άλλων. Στην περίπτωση αυτή καταλήγουμε σε ένα Context Αντικείμενο που αποτελείται από τα γεωγραφικό μήκος, το γεωγραφικό πλάτος, την ακρίβεια πλάτους και μήκους, την ακρίβεια υψομέτρου και τον χρόνο.

Η συλλογή Context Δεδομένων από το Διαχειριστή Context μπορεί να πραγματοποιηθεί με 3 τρόπους:

1. **Υποκλοπή Απομακρυσμένων Διαδικαστικών Κλήσεων (RPC Interception):** Υπάρχει ένας αυτόματος μηχανισμός ο οποίος, αφού πάρει άδεια από τον χρήστη μέσω του διαχειριστή πολιτικής, μπορεί να υποκλέψει τα RPCs που γίνονται από τις webinos ενεργοποιημένες εφαρμογές στα διάφορα APIs του webinos. Ο Διαχειριστής του context μετατρέπει το μήνυμα σε ένα Context Αντικείμενο, επιλέγοντας πεδία του RPC, που θεωρεί ότι μπορεί να περιέχουν context πληροφορίες, και δίνοντας του την δομή Context Αντικείμενου, μέσω του διευρυμένου Context API λεξικού. Το Context API λεξικό είναι μία λίστα από δομές και κανόνες για την αυτόματη contextοποίηση των RPC μηνυμάτων που έχουν υποκλαπεί.
2. **Υπηρεσία Context:** Το Context Αντικείμενο μπορεί να έχει καταχωρηθεί για την τακτική συλλογή δεδομένων φόντου ενώ τρέχει το Πληρεξούσιο Προσωπικής Ζώνης, καθορίζοντας τον τρόπο αποθήκευσης του Context Αντικείμενου, το χρονικό διάστημα των εκλογών, τις συνθήκες και την μέθοδο συλλογής.
3. **Context Αντικείμενα:** Τα context Αντικείμενα μπορούν να υπάρχουν και να αποθηκεύονται ανεξάρτητα από οποιαδήποτε εφαρμογή. Η εφαρμογή μπορεί να ζητήσει την καταχώρηση ενός νέου προσαρμοσμένου context αντικειμένου και να ζητήσει άδεια από τον Διαχειριστή Πολιτικής, για να αρχίσει να αποθηκεύει αυτά τα Αντικείμενα στη Context Βάση Δεδομένων. Η εφαρμογή μπορεί να καθορίσει την δική της διαδικασία απόκτησης των δεδομένων για αυτά τα αντικείμενα, όπως και την συχνότητα και την διάρκεια ζωής τους. Ο χρήστης – προγραμματιστής μπορεί να χρησιμοποιήσει το Λεξικό των Context Εφαρμογών του webinos για να ορίσει τους προσαρμοσμένους κανόνες και τις δομές για την αποθήκευση Context Αντικειμένων, είτε ειδικών –εφαρμογών είτε προερχόμενων από οποιαδήποτε διαδικασία ή συνδυασμό προϋπαρχόντων ή νέων context δεδομένων.

Η βάση δεδομένων όπου αυτά τα Context Αντικείμενα αποθηκεύονται με ασφάλεια βρίσκεται στο εικονικό πληρεξούσιο της προσωπικής ζώνης ενός χρήστη και είναι

προσβάσιμη μέσω ενός καναλιού που ανοίγει ο κόμβος της Προσωπικής Ζώνης (PZH). Η βάση δεδομένων του Context περιέχει δεδομένα από διάφορες συσκευές και εφαρμογές εντός μίας προσωπικής ζώνης και είναι μοναδική για κάθε Ζώνη. Η διαδικασία ερωτήσεων στην βάση δεδομένων επιτυγχάνεται μέσω μίας εύχρηστης και αξιόπιστης εφαρμογής δόμησης ερωτήσεων, η οποία επιτρέπει την αντιμετώπιση την Context Βάσης δεδομένων, ως μία αντικειμενοστραφής βάση δεδομένων, εστιάζοντας στο κύριο δομικό της στοιχείο, το Context Αντικείμενο. Ο χρήστης – προγραμματιστής μπορεί να απευθύνει ερωτήσεις απευθείας στο Context API, με την προοπτική απόκτησης οποιουδήποτε τύπου Context Αντικειμένων, που έχουν δημιουργηθεί από οποιοδήποτε API, εφαρμογή ή συσκευή μέσα στην Προσωπική Ζώνη ενός χρήστη.

5.1.2 **Context Λειτουργικότητα**

Μια σύνοψη των κύριων λειτουργιών του Διαχειριστή Context μπορεί να γίνει στα παρακάτω σημεία:

- Αυτόματα κρατάει/σώζει ένα αρχείο με όλες τις κλήσεις RPC, απογυμνωμένες από όλες τις παραμέτρους και τα αποτελέσματα στο πληρεξούσιο της προσωπικής ζώνης, σε ένα τοπικά αποθηκευμένο log.json αρχείο.

- Περιέχει το API Context λεξικό, το οποίο είναι μια περιγραφή των API, των μεθόδων τους που εκτίθενται στο WRT και των αναμενόμενων δομών των παραμέτρων και των αποτελεσμάτων , όλα δομημένα σύμφωνα με Context Αντικείμενα που θα παρήγαγαν παρόμοιες πληροφορίες τύπου Context (π.χ. MyLocation είναι το ίδιο Context Αντικείμενο ασχέτως με το αν προέρχεται από Geolocation API's getCurrentPosition, watchPosition ή από μία κλήση της λειτουργίας GPS του API του οχήματος).

- Αυτόματα υποκλέβει της κλήσεις RPC των μεθόδων των API.

- Αυτόματα υποκλέβει την καταχώρηση των ακροατών των API και ανιχνεύει της επιστροφές κλήσεων με σκοπό να τις ταιριάξει στις αντίστοιχες κλήσεις καταχώρησης, δομώντας τες ως μοναδικές κλήσεις RPC.

- Αυτόματα αναζητάει κάθε κλήση RPC στο Context λεξικό του webinos, βρίσκει την αντίστοιχη με βάση το API που κάνει την κλήση, το όνομα και την μέθοδο που καλέστηκε και σε ορισμένες περιπτώσεις τις παραμέτρους που χρησιμοποιήθηκαν για να ταιριάξουν την κλήση σε ένα Context Αντικείμενο. Τα δεδομένα εισόδου και εξόδου της κλήσης είναι δομημένα με βάση τους κανόνες του Context λεξικού, μη επιτρέποντας δεδομένα όπως κωδικούς ασφαλείας και άλλες ευαίσθητες πληροφορίες , τα οποία αποκλείονται από οποιαδήποτε context ερώτηση και δεν μεταβιβάζονται στο Διαχειριστή Πολιτικής.

- Τα Context Αντικείμενα, τα οποία δημιουργούνται από μία κλήση RPC, αποθηκεύονται σε έναν φάκελο ρυθμιστή (buffer) αν το εικονικό πληρεξούσιο προσωπικής ζώνης που έχει την Context Βάση Δεδομένων δεν είναι συνδεδεμένο.

- Τα Context Αντικείμενα, τα οποία δημιουργούνται από μία κλήση RPC, όπως και αυτά που περιέχονται στον ρυθμιστή context αντικειμένων στέλνονται στο εικονικό πληρεξούσιο προσωπικής ζώνης για να αποθηκευτούν στην SQLite3 Context Βάση Δεδομένων.

- Επιτρέπει τον ορισμό προσαρμοσμένων Context Αντικειμένων σε ένα διαφορετικό Context λεξικό της webinos εφαρμογής απευθείας από τον WRT.

- Οι εφαρμογές μπορούν να ζητήσουν την αποθήκευση των δικών τους μοναδικών Context Αντικειμένων στη Context Βάση Δεδομένων του εικονικού πληρεξούσιου προσωπικής ζώνης, με την ίδια ακριβώς δομή όπως όλα τα άλλα context δεδομένα.

- Τα δεδομένα στην Context Βάση Δεδομένων μπορούν να ερωτηθούν από τον WRT με την χρήση μιας προσαρμοσμένης δομή Context ερωτημάτων, η οποία επιτρέπει την περιγραφή απλών ή σύνθετων ερωτημάτων που περιέχουν υποερωτήματα ,όπως αυτά σχετίζονται με τα Context Αντικείμενα, αντιμετωπίζοντας την Βάση Δεδομένων ως περιγραφική των Μοντέλων Αντικειμένων. Τα αποτελέσματα θα επιστρέφονται ως Context Αντικείμενα.

- Οι κανόνες εκλογής/δειγματοληψίας (polling) μπορούν να είναι καταχωρημένες. Σε αυτούς, μία API κλήση μπορεί να είναι προγραμματισμένη σε προκαθορισμένα χρονικά διαστήματα και το αποτέλεσμα να υποκλέβεται αυτόματα από τον Διαχειριστή Context.

- Οι ακροατές Context Γεγονότων μπορούν να είναι καταχωρημένοι. Ένας κανόνας μπορεί να εφαρμόζεται, όπου το «όταν μία ειδική συνθήκη επικρατεί» προκύπτει και μία Context ερώτηση επιστρέφει ένα αποτέλεσμα, το οποίο αποτέλεσμα θα προκαλεί ένα γεγονός στην εφαρμογή η οποία καταχώρησε τον ακροατή.

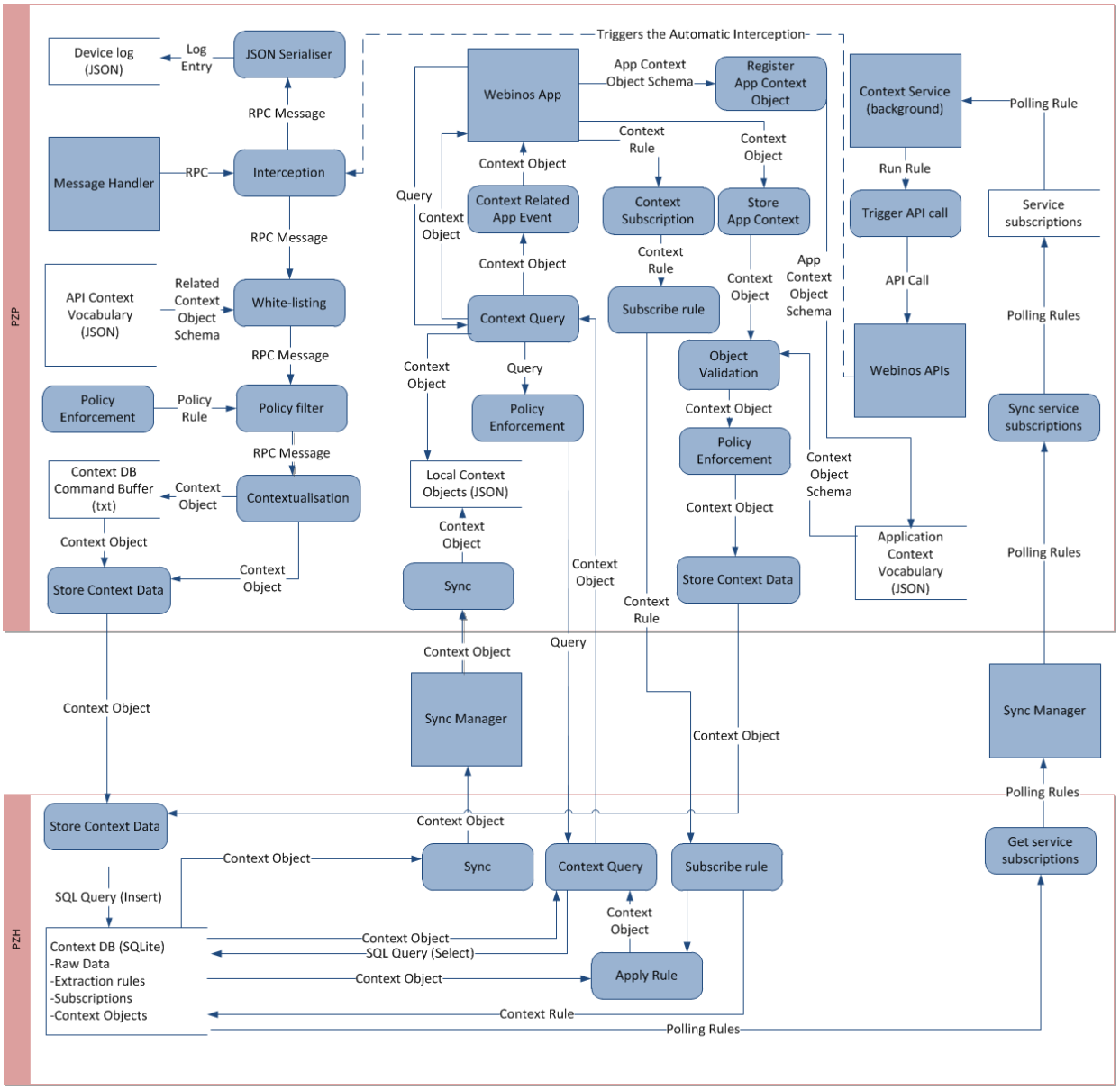
- Η ενσωμάτωση με τον Διαχειριστή Πολιτικής επιτρέπει στα APIs και στις εφαρμογές να αποθηκεύουν Context Αντικείμενα στην Context Βάση Δεδομένων του Εικονικού πληρεξουσίου Προσωπικής Ζώνης.

- Η ενσωμάτωση με τον Διαχειριστή Πολιτικής επιτρέπει στις εφαρμογές να κάνουν context ερωτήσεις στην Context Βάση Δεδομένων του Εικονικού πληρεξουσίου Προσωπικής Ζώνης.

- Η ενσωμάτωση με τον Διαχειριστή Πολιτικής επιτρέπει στις εφαρμογές να δημιουργήσουν και να ενισχύσουν Context κανόνες.

5.1.3 Επισκόπηση Ροής Δεδομένων Διαχειριστή Context

Η μέχρι τώρα προτεινόμενη αρχιτεκτονική μπορεί να συνοψιστεί και να αναπαρασταθεί με το παρακάτω Διάγραμμα Ροής Δεδομένων:



Εικόνα IX: Επισκόπηση Ροής Δεδομένων του Διαχειριστή Context

Η καταχώρηση των context αντικειμένων ώστε να μπορούν να εξαχθούν και να αποθηκευτούν τοπικά ανά συσκευή , είναι βασισμένη σε απλούς κανόνες εξαγωγής ή πιο σύνθετους κανόνες που περιγράφουν τα Context Αντικείμενα ως αποτελέσματα πιο σύνθετων αναζητήσεων και ερωτήσεων, χωρίς την προϋπόθεση μιας σύνδεσης με τον εικονικό Πληρεξούσιο Προσωπικής Ζώνης που έχει την Context Βάση Δεδομένων. Για παράδειγμα, ένας στατιστικός κανόνας που εφαρμόζεται στα Context Αντικείμενα MyLocation μπορεί να παράγει πιο μόνιμα Context Αντικείμενα που να λέγονται MyHome, MyWorkPlace, MyParents κ.α. Μία υπηρεσία φόντου θα μπορεί να αποθηκεύει τα API των Context Αντικειμένων, ενώ το πληρεξούσιο προσωπικής ζώνης τρέχει και χωρίς η εφαρμογή να κάνει συγκεκριμένες κλήσεις σε API. Η καταχώρηση τέτοιων ακροατών θα πρέπει να πραγματοποιείται με τον ίδιο τρόπο που ορίζονται τα Context Αντικείμενα μέσω του Διαχειριστή Πολιτικής. Είναι ξεκάθαρο ότι οι τελικοί χρήστες δεν καταλαβαίνουν την μελλοντική αξία της απόκρυψης των προσωπικών και ιδιωτικών πληροφοριών τους και πολύ συχνά προχωράνε είτε στην κοινοποίηση τους είτε και στην παραχώρηση των δικαιωμάτων χρήσης τους, χωρίς να έχουν προηγουμένως εκτιμήσει τις συνέπειες που μπορεί να προκύψουν από πιθανές χρήσεις τους. Έχοντας αυτό κατά νου, η πλατφόρμα του webinos μέσω του Διαχειριστή Context εξασφαλίζει ότι η κατοχή πληροφοριών τύπου Context μένει στον τελικό χρήστη, ενώ παράλληλα τα δικαιώματα πρόσβασης των εφαρμογών στην αποθήκευση, εξαγωγή και αναζήτηση context δεδομένων μπορεί να δοθεί από τον τελικό χρήστη, μονάχα στην εφαρμογή και όχι στο χρήστη- προγραμματιστή. Αυτό επιτρέπει στον χρήστη- προγραμματιστή να δημιουργήσει εφαρμογές, οι οποίες μπορούν να χρησιμοποιήσουν Context Αντικείμενα που είναι αποθηκευμένα μέσω της πλατφόρμας του webinos ή άλλων εφαρμογών στην ίδια Προσωπική Ζώνη. Επιπρόσθετα για να διαφυλάξουμε περαιτέρω τα προσωπικά δεδομένα, όλες οι ανταλλαγές και η εν γένει επικοινωνία με την Context Βάση Δεδομένων παρακολουθείται και καταγράφεται από τον διαχειριστή πολιτικής του webinos και συγκεκριμένα δικαιώματα πρόσβασης όπως διαβάζω/γράφω, από και προς την Context Βάση Δεδομένων παρέχονται ξεχωριστά και ανάλογα με την εφαρμογή, τον τύπο και την πηγή των Context Αντικειμένων.

Επιπλέον, με την ανάπτυξη του Διαχειριστή Πολιτικής και την ικανότητά του να παράγει πιο σύνθετους κανόνες πολιτικής, κάθε κλήση Context API θα έχει έναν αριθμό από γεγονότα , σε σχέση με την πολιτική, τα οποία θα σχετίζονται μαζί της στους παρακάτω τομείς:

- Η Μέθοδος του Context API
- Διαβάζω/Γράφω
- Την εφαρμογή που κάνει την κλήση
- Δεδομένα Αντικειμένου (Context Αντικείμενο, κανόνες που πρέπει να προστεθούν/να ανανεωθούν κλπ.)

5.1.4 **Context Λεξικό**

Το context λεξικό βασίζεται σε μία δομή η οποία ορίζει τα context αντικείμενα σε σχέση με τα API στα οποία ανήκουν. Με αυτόν τον τρόπο, η δομή ξεκινάει από τον ορισμό του API, καθώς και των δεδομένων που χρειάζονται για να το αναγνωρίσουν, και μετά με το όνομα του context αντικειμένου. Τέλος, η δομή των μεθόδων και των δεδομένων σχετίζεται είτε με τον τρόπο συλλογής των δεδομένων, που αποτελούν κομμάτι του context αντικειμένου, είτε με τις τιμές που αναμένονται από μία συγκεκριμένη μέθοδο.

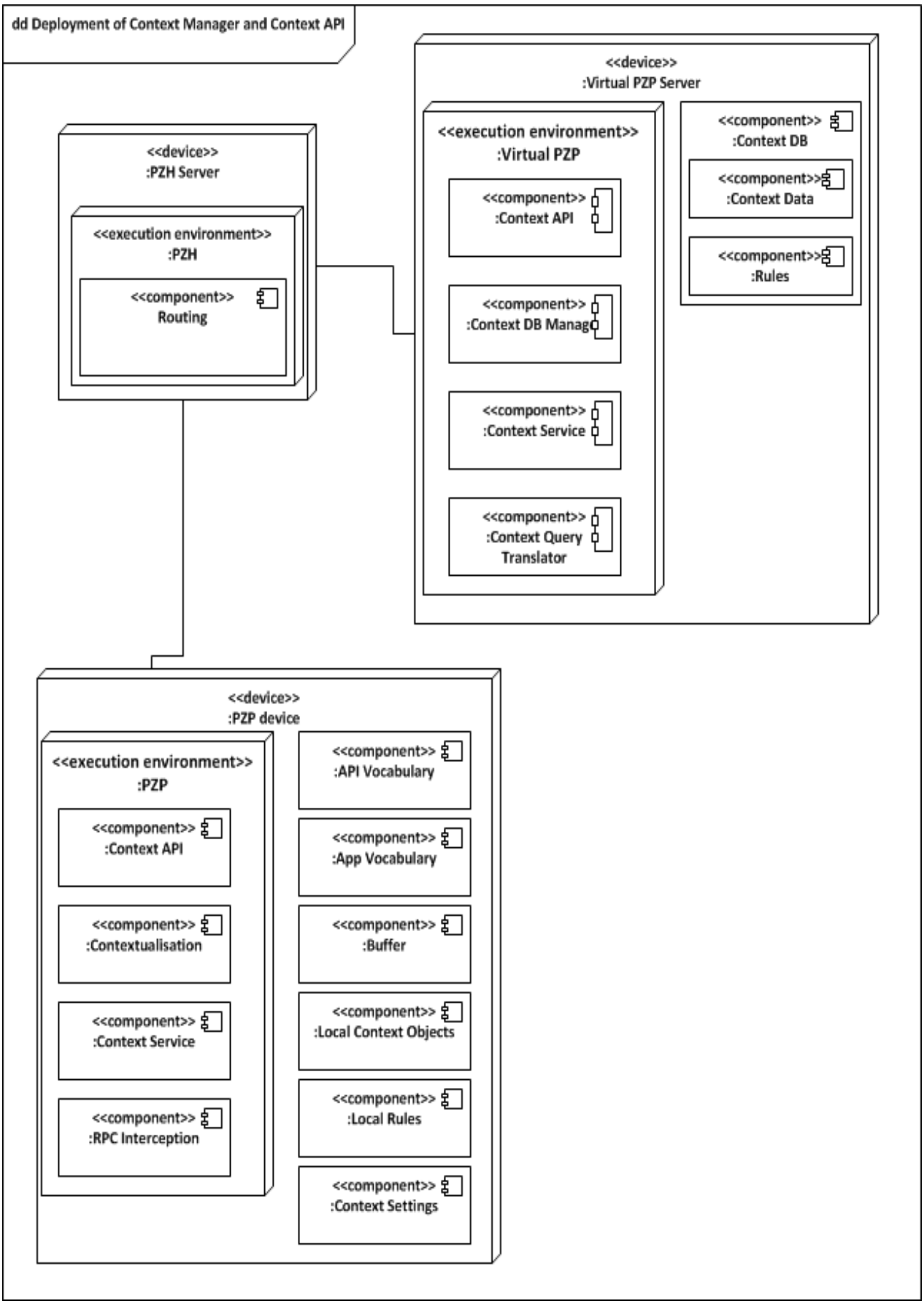
5.1.5 **Διάγραμμα Ανάπτυξης**

Ο Διαχειριστής Context/Context API αναπτύσσεται σε τρεις διαφορετικές τοποθεσίες:

- Ένα σχεδιασμένο Εικονικό Πληρεξούσιο Προσωπικής Ζώνης (vPZP) όπου κρατείται η Context Βάση Δεδομένων και το οποίο λειτουργεί ως διακομιστής για τον Διαχειριστή Context και τα Context API.

- Τον κόμβο της Προσωπικής Ζώνης (PZH), όπου βρίσκονται οι απαραίτητες ρυθμίσεις για την εύρεση και επαφή με του εικονικού πληρεξουσίου προσωπικής ζώνης (vPZP), όπου βρίσκεται η Context βάση δεδομένων.

- Στο πληρεξούσιο προσωπικής ζώνης (PZP) του Διαχειριστή Context και των Context API.

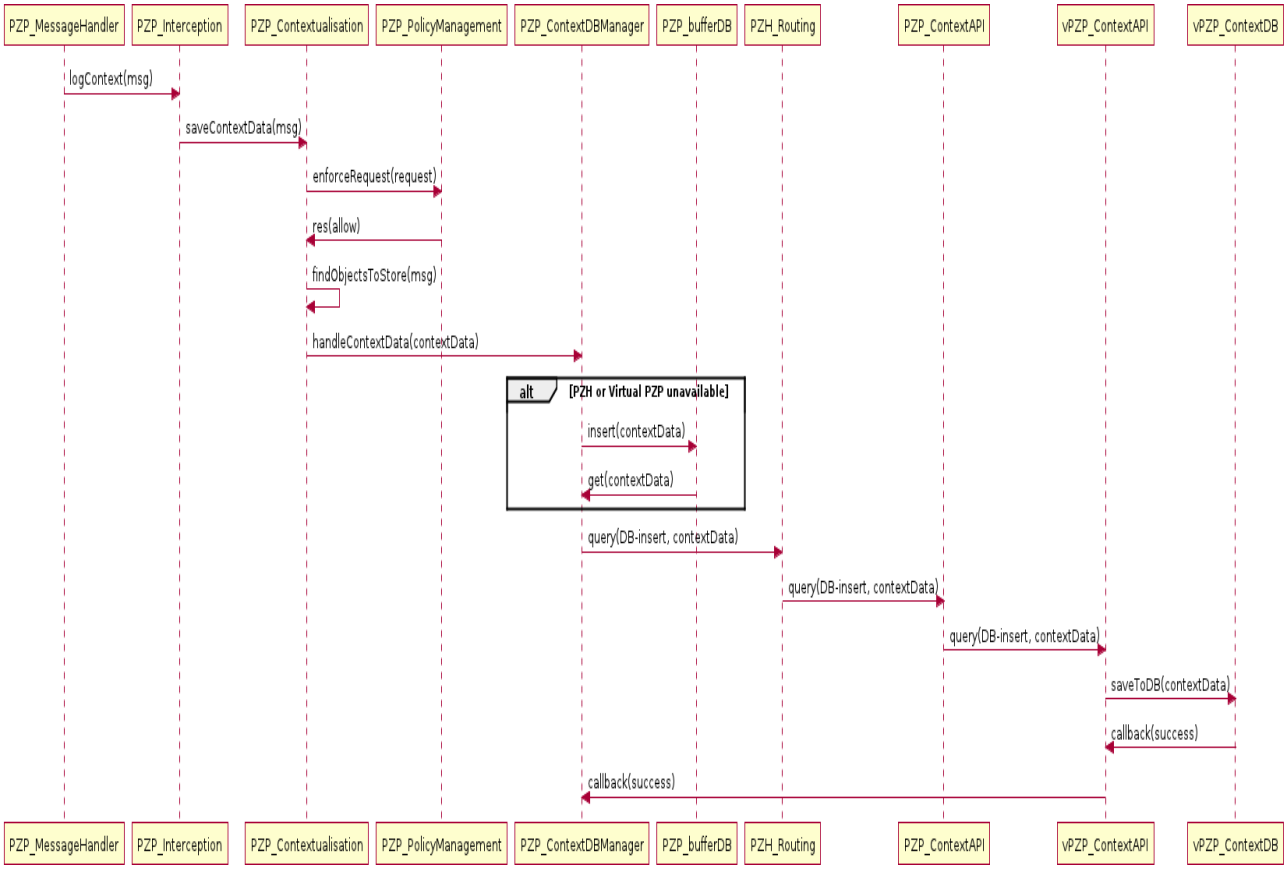


Εικόνα X: Διάγραμμα Ανάπτυξης Διαχειριστή Context

5.1.6 Υποκλοπή Μηνύματος και Αποθήκευση

Ο Διαχειριστής Context αυτόματα υποκλέβει τα μηνύματα που παραδίδονται στα API, τα εξετάζει, τα αναζητά στο context λεξικό, τα φιλτράρει και τέλος μετατρέπει το περιεχόμενο τους σε μορφή τέτοια που θα είναι σχετική με τις context ερωτήσεις.

Τα δεδομένα μετά προωθούνται στο vPZP στο οποίο βρίσκεται και η Context Βάση Δεδομένων, όπου και αποθηκεύονται. Στην περίπτωση που ούτε ο κόμβος της προσωπικής ζώνης (PZH), ούτε το εικονικό πληρεξούσιο προσωπικής ζώνης (vPZP) είναι προσβάσιμα τα δεδομένα αποθηκεύονται σε ένα απλό αρχείο, που λειτουργεί ως ρυθμιστής των δεδομένων που πρόκειται να εισαχθούν στη Context Βάση Δεδομένων, έως ότου η σύνδεση με το εικονικό PZP γίνει εφικτή.



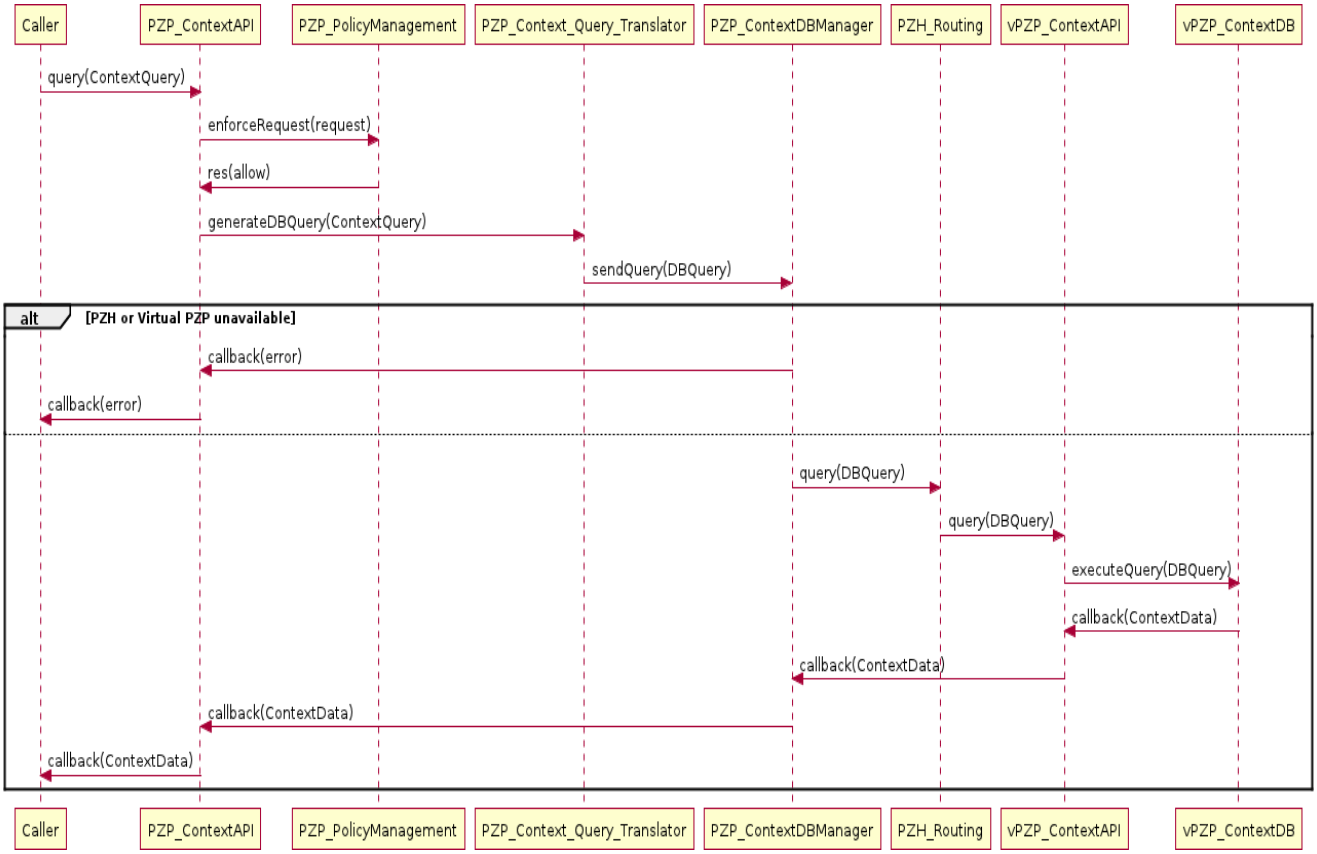
Εικόνα XI: Υποκλοπή μηνύματος και επισκόπηση αποθήκευσης

5.1.7 **Context Querying**

Τα queries και η αναζήτηση Context δεδομένων γίνεται μέσω ενός μηχανισμού querying ο οποίος μεταφράζει Context queries υψηλού επιπέδου σε αντίστοιχες χαμηλού επιπέδου, οι οποίες προορίζονται για την context βάση δεδομένων. Η δόμηση των queries επιτρέπει την δημιουργία τους, με βάση τις παρακάτω ρήτρες:

- **eq**: ισούται
- **lt**: λιγότερο από
- **le**: λιγότερο ή ίσο με
- **gt**: μεγαλύτερο από
- **ge**: μεγαλύτερο ή ίσο με
- **starts**: Ξεκινάει με
- **ends**: Τερματίζει με
- **in**: στη δοσμένη λίστα. Η τιμή πρέπει να είναι και η σειρά.
- **contains**: Η τιμή να περιέχει την δοσμένη τιμή, εφαρμόσιμο μόνο σε DOMString

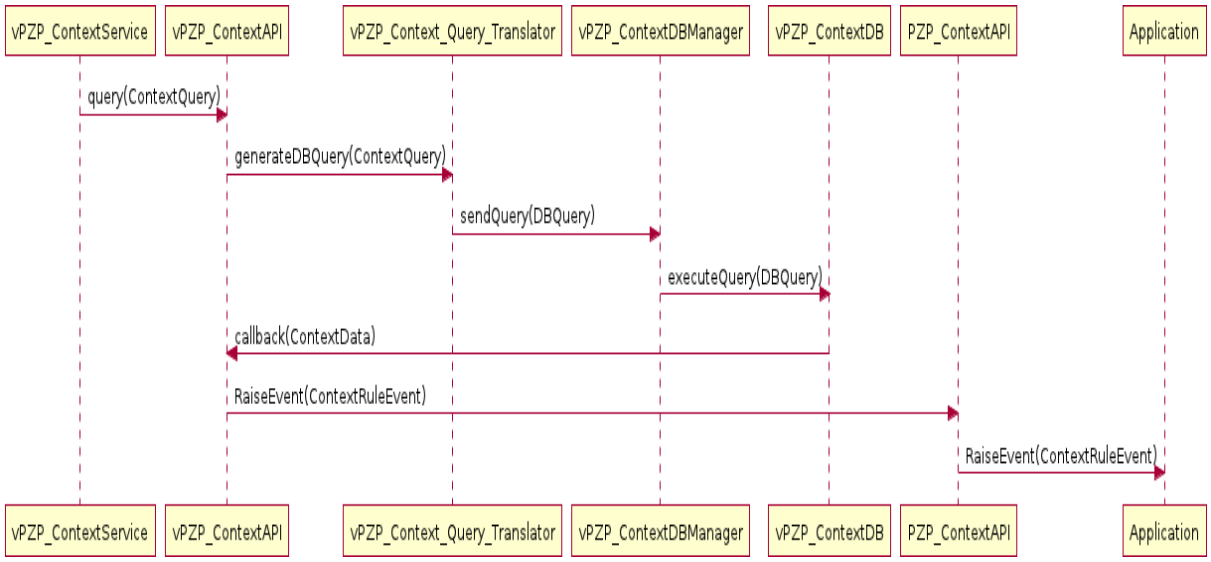
Η αλληλουχία των γεγονότων σε περίπτωση κλήσης, είτε αυτή είναι εσωτερική είτε από εφαρμογή, έχει ως εξής:



Εικόνα XII: Διαδικασία Context Querying

5.1.8 **Κανόνες Context**

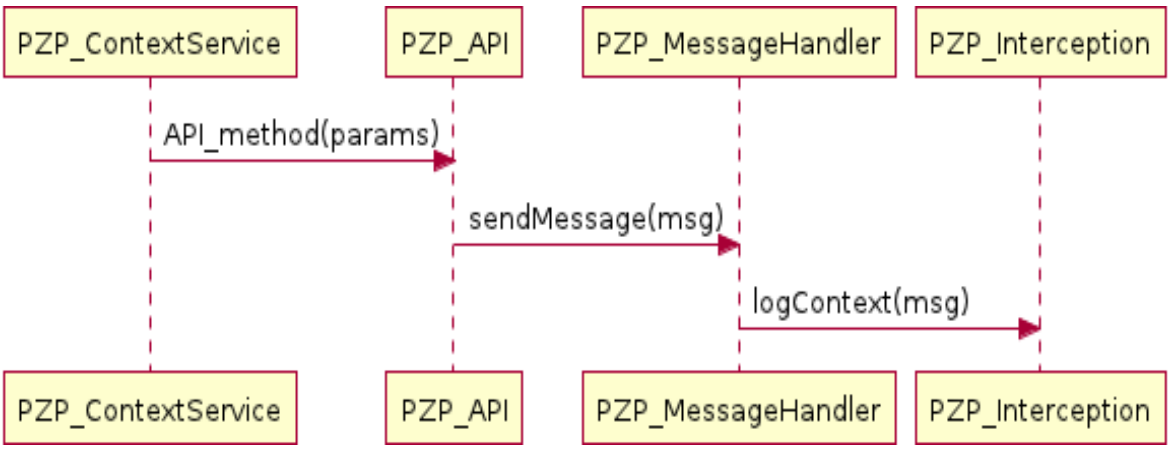
Ένας context κανόνας ορίζεται ως μία ερώτηση τύπου context, η οποία ρυθμίζεται να τρέχει με συγκεκριμένα χρονικά μεσοδιαστήματα στη Context βάση δεδομένων και να διαθέτει τα αποτελέσματα της σε όλες τις εφαρμογές και όλα τα συνδεδεμένα πληρεξούσια προσωπικής ζώνης, εφόσον αυτά έχουν άδεια να τα εξαγάγουν. Αν ο κανόνας έχει μία πολύ συγκεκριμένη προϋπόθεση, το γεγονός δεν θα ενεργοποιηθεί έως ότου η προϋπόθεση αυτή πληρούται. Αν ένας κανόνας δεν έχει ανανεωθεί από την εφαρμογή για έναν μήνα, ο κανόνας παύει να ισχύει.



Εικόνα XIII: Επισκόπηση Context Κανόνων

5.1.9 **Προγραμματισμένες Κλήσεις API**

Αν μία εφαρμογή χρειάζεται συνεχή λήψη ενός Context Αντικειμένου, τότε μπορεί να ζητήσει από τον Διαχειριστή Πολιτικής να καταχωρήσει μία προγραμματισμένη κλήση API από μία συγκεκριμένη λίστα API και για μια συγκεκριμένη συχνότητα υποκλοπής. Οι προγραμματισμένες κλήσεις API έχουν ισχύ ενός μήνα από την στιγμή που ξεκινάνε. Αν καμία εφαρμογή δεν ανανεώσει την καταχώρηση κατά τη διάρκεια αυτού του μήνα, τότε η προγραμματισμένη κλήση API απενεργοποιείται.



Εικόνα XIV: Προγραμματισμένες Κλήσεις API

5.2 Context Σενάρια & Περιπτώσεις Χρήσης

5.2.1 Εντοπισμός χαμένης συσκευής

Επισκόπηση: Εύρεση χαμένου κινητού μέσω προγραμματιστικής εφαρμογής.

Περιγραφή : Η Τασούλα έχει στην προσωπική της ζώνη δύο συσκευές.

1. Κινητό τηλέφωνο
2. Φορητό Η/Υ

Το πρωί που ξυπνάει παίρνει την τσάντα της με τον φορητό Η/Υ της και το κινητό της και αφού αφήσει το παιδί της στο σχολείο, πάει στο γραφείο της. Κάθεται εκεί μέχρι το μεσημέρι και μετά πάει στο σουπερμάρκετ, περνάει από το γραφείο του άντρα της και μετά γυρίζει σπίτι, αφού πρώτα περάσει από το σχολείο να πάρει το παιδί της. Όταν φτάνει σπίτι συνειδητοποιεί ότι έχει ξεχάσει το κινητό της. Μη μπορώντας να θυμηθεί που το έχει ξεχάσει ανοίγει τον Η/Υ και μέσω μιας GPS εφαρμογής του webinos βλέπει τις τελευταίες τοποθεσίες που βρισκόταν το κινητό της. Έτσι συνειδητοποιεί ότι το άφησε στο γραφείο του άντρα της, καθώς εκεί βρισκόταν αυτό πριν μισή ώρα που αυτή είχε ήδη φύγει. Ακόμα βλέπει ότι πριν 10 λεπτά το κινητό κινούνταν, υποθέτοντας έτσι ότι ο άντρας της το βρήκε και το έχει πάρει μαζί του. Την προηγούμενη διαπίστωση επιβεβαιώνει ο άντρας της τηλεφωνικά.

Προκύπτοντα ζητήματα:

- Η Τασούλα θέλει να μπορούν οι συσκευές τις να αποθηκεύουν ανά τακτά χρονικά διαστήματα την τοποθεσία τους.
- Η Τασούλα θέλει οι προσωπικές της πληροφορίες να είναι διαθέσιμες από όλες τις συσκευές τις, ώστε όταν χάσει π.χ. την μία να μπορεί να την βρει με την άλλη.

Πλεονεκτήματα :

- Το webinos σου δίνει τη δυνατότητα καταχώρησης και αποθήκευσης της τοποθεσίας των συσκευών σου.
- Το webinos σου δίνει τη δυνατότητα να μπορείς να ελέγξεις πληροφορίες αποθηκευμένες στην Context βάση δεδομένων σου, από οποιαδήποτε webinos ενεργοποιημένη συσκευή έχεις στην Προσωπική σου Ζώνη.

Κατηγοριοποίηση χρήσης:

Χαρακτήρες	Τασούλα
Διάρκεια	Ώρες
Συχνότητα	Ημερήσια-Εβδομαδιαία
Απαιτήσεις	Χαμηλές
Σύγκρουση στόχων	Χαμηλές

5.2.2 **Τοποθεσία απομακρυσμένης συσκευής**

Επισκόπηση: Παρουσίαση Γιώργου

Περιγραφή : Ο Γιώργος έχει στην προσωπική του ζώνη δύο συσκευές.

1. Tablet
2. Φορητό Η/Υ

Είναι στο γραφείο του και ολοκληρώνει μία παρουσίαση που πρέπει να κάνει σε μία άλλη εταιρεία. Την παρουσίαση την δουλεύει εξολοκλήρου σε μία εφαρμογή του webinos στον φορητό Η/Υ. Βλέπει όμως ότι αν δεν φύγει τώρα δεν θα προλάβει να είναι εγκαίρως στην άλλη εταιρεία. Γι αυτό κλείνει τον Η/Υ στο γραφείο του και μπαίνει στο τρένο όπου ανοίγει αμέσως την εφαρμογή μέσω του tablet του, για να κάνει τις τελευταίες μικροδιορθώσεις που είναι απαραίτητες. Το webinos βλέπει ότι αφού η εφαρμογή τερματίστηκε στον Η/Υ, η τοποθεσία του tablet όπου μόλις ενεργοποιήθηκε η εφαρμογή είναι διαφορετική από αυτή του Η/Υ όπου δούλευε προηγουμένως. Αυτόματα φορτώνει το αρχείο στο οποίο δούλευε ο Γιώργος στο tablet, όπως σώθηκε τελευταία φορά από τον Η/Υ, για να μπορέσει να το επεξεργαστεί. Τέλος η ίδια διαδικασία ακολουθείται αφού ο Γιώργος ολοκληρώσει την παρουσίαση μέσα στο τρένο, κλείσει το tablet του και ξανανοίξει τον φορητό Η/Υ του στην εταιρεία, έτοιμος πια για την παρουσίαση.

Προκύπτοντα ζητήματα:

- Να μπορεί ο Γιώργος να επεξεργαστεί το αρχείο από οποιαδήποτε συσκευή έχει ενεργοποιημένη στο webinos.
- Να γίνεται αυτόματα η φόρτωση του αρχείου ανάλογα με την συσκευή όπου τρέχει η εφαρμογή.

Πλεονεκτήματα :

- Το webinos σου δίνει τη δυνατότητα καταχώρησης και αποθήκευσης των δεδομένων μίας εφαρμογής ανεξάρτητα από την συσκευή όπου αυτή έτρεχε.
- Το webinos σου δίνει τη δυνατότητα επεξεργασίας των δεδομένων της εφαρμογής, ανάλογα με την συσκευή που ο χρήστης χρησιμοποιεί τη δεδομένη στιγμή.

Κατηγοριοποίηση χρήσης:

Χαρακτήρες	Γιώργος
Διάρκεια	Ώρες
Συχνότητα	Ημερήσια-Εβδομαδιαία
Απαιτήσεις	Χαμηλές
Σύγκρουση στόχων	Χαμηλές

5.2.3 **Συσκευές Πολλαπλών Χρηστών**

Επισκόπηση: Κοινή χρήση συσκευών από Τάκη και Λάκη

Περιγραφή : Ο Τάκης και ο Λάκης είναι δύο νέοι συνάδελφοι σε μια εταιρεία, και καθώς προέρχονται και οι δύο από την επαρχία, αποφάσισαν για να μοιραστούν τα έξοδα της ζωής τους στην Αθήνα, να νοικιάσουν μαζί διαμέρισμα, καθώς και αυτοκίνητο. Το διαμέρισμα διαθέτει μία τηλεόραση, την οποία, όπως είναι λογικό την χρησιμοποιούν είτε ο καθένας μόνος του, όταν ο άλλος λείπει, είτε από κοινού. Παρομοίως για το αυτοκίνητο, το χρησιμοποιούν και οι δύο μαζί τα πρωινά για να πηγαίνουν στο γραφείο όπως και για τα επαγγελματικά τους ταξίδια. Τα απογεύματα, όταν δεν τους βολεύει να το χρησιμοποιούν από κοινού, το έχουν μοιράσει ώστε να το δικαιούται μία ο ένας, μία ο άλλος, εναλλάξ κάθε μέρα. Και οι δύο είναι χρήστες του webinos, αλλά πέρα από τις προσωπικές τους συσκευές (κινητά τηλέφωνα, φορητός υπολογιστής κλπ) που είναι webinos ενεργοποιημένες, θέλουν να ενεργοποιήσουν στο webinos και τις δύο συσκευές (τηλεόραση, αυτοκίνητο) που χρησιμοποιούν από κοινού.

Προκύπτοντα ζητήματα:

- Να μπορεί μια συσκευή να χρησιμοποιηθεί από πολλαπλούς webinos χρήστες

- Να μην επηρεάζει ο λογαριασμός webinos του ενός χρήστη , τον άλλον. Να μπορούν να συνυπάρχουν και οι δύο στην ίδια συσκευή (όπως σε έναν Η/Υ) ανεξάρτητα.

Πλεονεκτήματα :

- Το webinos σου δίνει τη δυνατότητα ύπαρξης περισσοτέρων τους ενός ανεξαρτήτων πληρεξουσίων (PZP) στην ίδια συσκευή.

Κατηγοριοποίηση χρήσης:

Χαρακτήρες	Τάκης, Λάκης
Διάρκεια	Ώρες
Συχνότητα	Ημερήσια-Εβδομαδιαία
Απαιτήσεις	Χαμηλές
Σύγκρουση στόχων	Χαμηλές

5.2.4 **Συλλογή και αναπαράσταση Context Δεδομένων**

Επισκόπηση: Προετοιμασία Αθλητών

Περιγραφή : Ο Παύλος είναι προπονητής στίβου και έχει τέσσερις αθλητές δρόμων. Διανύουν περίοδο προετοιμασίας και ο προπονητής θέλει να ξέρει σε τι κατάσταση βρίσκονται οι αθλητές του μετά τις διακοπές τους. Για το σκοπό αυτό έχει προμηθευτεί τέσσερις παλμογράφους, τους οποίους μπορούν οι αθλητές να φοράνε την ώρα που προπονούνται. Οι παλμογράφοι είναι webinos ενεργοποιημένοι στην προσωπική ζώνη του Παύλου, ο οποίος είναι χρήστης του webinos. Με αυτόν το τρόπο μπορεί ο Παύλος μετά την προπόνηση και μέσω μιας webinos εφαρμογής στον Η/Υ του, να αναπαριστά τα δεδομένα τα οποία είχαν προηγουμένως συλλέξει οι παλμογράφοι και να βγάζει τα συμπεράσματά του.

Προκύπτοντα ζητήματα:

- Τα δεδομένα που συλλέγονται από τις συσκευές που χρησιμοποιεί ο κάθε αθλητής να μπορούν να χρησιμοποιηθούν από μία οποιαδήποτε εφαρμογή που τρέχει σε άλλη webinos ενεργοποιημένη συσκευή.

Πλεονεκτήματα :

- Όλα τα δεδομένα που συλλέγονται από τους context προμηθευτές αποθηκεύονται στην ίδια context βάση δεδομένων του webinos, ανεξαρτήτως συσκευής ή εφαρμογής προέλευσης.
- Μία webinos εφαρμογή μπορεί να αποκτήσει πρόσβαση στα δεδομένα που έχουν αποθηκευθεί στην context βάση δεδομένων του webinos.

Κατηγοριοποίηση χρήσης

Χαρακτήρες	Coach Παύλος, τέσσερις αθλητές
Διάρκεια	Ώρες
Συχνότητα	Ημερήσια-Εβδομαδιαία
Απαιτήσεις	Χαμηλές
Σύγκρουση στόχων	Χαμηλές

5.2.5 **Συνδυασμός και context λογισμός**

Επισκόπηση: εξαγωγή συμπερασμάτων κατά τη διάρκεια της προετοιμασίας

Περιγραφή : Ο προπονητής του προηγούμενου σεναρίου παράλληλα με την σφυγμομέτρηση των αθλητών, τούς έχει εφοδιάσει και με GPS ενσωματωμένα στη σόλα του παπουτσιού τους. Όπως και οι προηγούμενες συσκευές, έτσι και τα GPS είναι webinos ενεργοποιημένα. Ο προπονητής θέλει συνδυάζοντας τα δεδομένα των δύο συσκευών για κάθε αθλητή να μπορεί να αποφανθεί για την φυσική κατάσταση του, ώστε να προσαρμόσει το πρόγραμμα της προετοιμασίας ατομικά για τον κάθε αθλητή του. Για παράδειγμα, μπορεί ένας αθλητής να πιάνει την μέγιστη επιτρεπτή τιμή σφυγμών τρέχοντας με ταχύτητα χαμηλότερη από ότι οι υπόλοιποι αθλητές. Αυτό δείχνει ότι ο συγκεκριμένος αθλητής βρίσκεται σε χειρότερη φυσική κατάσταση από τους συναθλητές του, και χρειάζεται ειδικό πρόγραμμα εκγύμνασης.

Προκύπτοντα ζητήματα:

- Τα δεδομένα που συλλέγονται από τις δύο συσκευές να συλλέγονται με τέτοιο τρόπο ώστε να μπορούν να συνδυαστούν.

- Μέσω του συνδυασμού των δεδομένων να μπορεί να γίνεται η εξαγωγή των συμπερασμάτων προς τον προπονητή.

Πλεονεκτήματα :

- Το webinos σου δίνει τη δυνατότητα να ρυθμίσεις την συλλογή δεδομένων ως προς διάφορους παράγοντες (διάρκεια, τοποθεσία, χρόνος, συχνότητα δειγματοληψίας κ.α.)
- Το webinos σου δίνει τη δυνατότητα να μπορείς να συνδυάσεις διαφορετικού τύπου δεδομένα, δηλαδή δεδομένα από διαφορετικού είδους context προμηθευτές (π.χ. παλμοί με ταχύτητα).
- Το webinos επιτρέπει στην εφαρμογή να χρησιμοποιήσει το συνδυασμό των δεδομένων για την εξαγωγή συμπερασμάτων.

Κατηγοριοποίηση χρήσης:

Χαρακτήρες	Coach Παύλος, τέσσερις αθλητές
Διάρκεια	Ώρες
Συχνότητα	Ημερήσια-Εβδομαδιαία
Απαιτήσεις	Χαμηλές
Σύγκρουση στόχων	Χαμηλές

5.2.6 **Φιλτράρισμα για αποφυγή διπλοεγγραφών**

Επισκόπηση: Αποφυγή διπλοεγγραφών προς εξοικονόμηση μνήμης στην βάση δεδομένων.

Περιγραφή : Ο Σοφοκλής, προγραμματιστής εφαρμογών, έχει επιφορτιστεί από την εταιρεία στην οποία εργάζεται με τη δημιουργία μιας εφαρμογής ο χρήστης της οποίας θα μπορεί ανά ορισμένα χρονικά διαστήματα (π.χ. Προηγούμενος Μήνας , Χθες) να παρακολουθεί, είτε τις δικές του δραστηριότητες, είτε αν έχει την απαραίτητη αδειοδότηση, τις δραστηριότητες άλλων χρηστών. Για παράδειγμα, που βρισκόταν την προηγούμενη Δευτέρα το απόγευμα ή συγκεντρωτικά που πέρασε τις περισσότερες ώρες του τον προηγούμενο μήνα κλπ. Για την ανάπτυξη όμως της εφαρμογής αυτής έχει στη διάθεση του μία, ορισμένης χωρητικότητας βάση δεδομένων, γεγονός που δεν του επιτρέπει τις διπλοεγγραφές. Για παράδειγμα, όταν ο τελικός χρήστης βρίσκεται σε μία τοποθεσία μαζί με όλες τις συσκευές της προσωπικής του ζώνης (π.χ. σπίτι του), να μην αποθηκεύεται ταυτόχρονα η τοποθεσία του από όλες.

Προκύπτοντα ζητήματα:

- Να μπορεί η πλατφόρμα να ειδοποιεί τον χρήστη σε περίπτωση που υπάρξει ενδεχόμενο διπλοεγγραφής δεδομένων.

Πλεονεκτήματα :

- Το webinos σου δίνει τη δυνατότητα σύγκρισης των δεδομένων που έχουν καταχωρηθεί στην context βάση δεδομένων
- Το webinos σου δίνει τη δυνατότητα διαγραφής δεδομένων από την context βάση δεδομένων για να επιτευχθεί καλύτερη διαχείριση του αποθηκευτικού χώρου.

Κατηγοριοποίηση χρήσης:

Χαρακτήρες	Σοφοκλής
Διάρκεια	Ώρες
Συχνότητα	Ημερήσια-Εβδομαδιαία
Απαιτήσεις	Χαμηλές
Σύγκρουση στόχων	Χαμηλές

6 References

[1] ANIND K. DEY. 1998. Context-Aware Computing: The CyberDesk Project. In *National Conference on Artificial Intelligence*.

[2] Bakken, D. E. *MIDDLEWARE*.

[3] Baldauf, M., Dustdar, S., and Rosenberg, F. 2007. A survey on context-aware systems. *IJAHUC* 2, 4, 263.

[4] Bardram, J. E. The Java Context Awareness Framework (JCAF) – A Service Infrastructure and Programming Framework for Context-Aware Applications. A Service Infrastructure and Programming Framework for Context-Aware Applications 3468, 98–115.

[5] Biegel, G. and Cahill, V. *Sentient Objects*. *Towards Middleware for Mobile Context-Aware Applications*.

[6] Brown, P. J. 1998. Triggering information by context. *Personal Technologies* 2, 1, 1–9.

[7] Capra, L., Emmerich, W., and Mascolo, C. 2003. CARISMA: Context-Aware Reflective mIddleware System for Mobile Applications. *IIEEE Trans. Software Eng.* 29, 10, 929–944.

[8] Capra, L. and Quercia, D. 2012. *Middleware for social computing: a roadmap* 3. *J Internet Serv Appl* 3, 1, 117–125.

[9] Chen, G. and Kotz, D. *A Survey of Context-Aware Mobile Computing Research*.

[10] Chen, G. and Kotz, D. 2002. Solar: An Open Platform for Context-Aware Mobile Applications. In *Pervasive computing. First international conference, Pervasive 2002, Zurich, Switzerland, August 26-28, 2002 : proceedings*. Springer, Berlin, New York, 41–47.

[11] Chen, H., Tolia, S., Sayers, C., Finin, T., and Joshi, A. 2003. Creating context-aware software agents. *Innovative Concepts for Agent-Based Systems*, 186–197.

[12] Costa, P. D., Pires, L. F., van Sinderen, M., and Pereira Filho, J. 2004. Towards a Services Platform for Mobile Context-Aware Applications. In *Proc. of the 1st Int. Workshop on Ubiquitous Computing (IWUC)*.

[13] Dey, A. K. 2001. Understanding and Using Context. *Personal and Ubiquitous Computing* 5, 1, 4–7.

[14] Dey, A., Abowd, G., and Salber, D. 2001. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Comp. Interaction* 16, 2, 97–166.

[15] Dionysis Athanasopoulos, Apostolos V. Zarras, Valerie Issarny, Evaggelia Pitoura, and Panos Vassiliadis. 2008. CoWSAMI: Interface-aware context gathering in ambient intelligence environments. *Pervasive and Mobile Computing* 4, 3, 360–389.

[16] Fuhrhop, C. 2012. *webinos phase II architecture and components*.

[17] Glassey, R., Stevenson, G., Richmond, M., NIXON, P., Terzis, S., Wang, F., and Ferguson, R. I. 2003. Towards a middleware for generalised context management. In *First International Workshop on Middleware for Pervasive and Ad Hoc Computing, Middleware 2003*.

[18] Gu, T., Pung, H. K., and Zhang, D. Q. 2005. A service-oriented middleware for building context-aware services. *Journal of Network and Computer Applications* 28, 1, 1–18.

[19] Henricksen, K., Indulska, J., and Rakotonirainy, A. Modeling Context Information in Pervasive Computing Systems 2414, 167–180.

[20] Henricksen, K. and Robinson, R. 2006. A survey of middleware for sensor networks: state-of-the-art and future directions. In *Proceedings of the international workshop on Middleware for sensor networks*, 60–65.
[21] Hong, J. I. 2002. The context fabric: an infrastructure for context-aware computing. In *CHI'02 extended abstracts on Human factors in computing systems*, 554–555.
[22] Hong, J. and Landay, J. 2001. An Infrastructure Approach to Context-Aware Computing. *Human-Comp. Interaction* 16, 2, 287–303.
[23] 1997. *Wearable Computers, 1997. Digest of Papers., First International Symposium on*.
[24] Kim, H., Cho, Y.-J., and Oh, S.-R. 2005. CAMUS: A middleware supporting context-aware services for network-based robots. In *Advanced Robotics and its Social Impacts, 2005. IEEE Workshop on*, 237–242.
[25] Krakowiak, S. *What is Middleware?*
[26] Lassila, O., Swick, R. R., and others. 1998. Resource description framework (RDF) model and syntax specification.
[27] Mille, A., Gandon, F., Misselis, J., Rabinovich, M., Staab, S., Fuhrhop, C., Lyle, J., and Faily, S. The webinos project, 259.
[28] Ngo, H. Q., Shehzad, A., Liaquat, S., Riaz, M., and Lee, S. Developing Context-Aware Ubiquitous Computing Systems with a Unified Middleware Framework 3207, 672–681.
[29] Pascoe, J. Adding generic contextual capabilities to wearable computers, 92–99.
[30] Pascoe, J., Ryan, N., and Morse, D. Issues in Developing Context-Aware Computing 1707, 208–221.
[31] Petrelli, D., Not, E., Zancanaro, M., Strapparava, C., and Stock, O. 2001. Modeling and adapting to context. *Personal and Ubiquitous Computing* 5, 1, 20–24.
[32] Ranganathan, A., Al-Muhtadi, J., and Campbell, R. 2004. Reasoning about uncertain contexts in pervasive computing environments. *Znanstvena revija. Družboslovje in filozofija = Social sciences and philosophy* 3, 2, 62–70.
[33] Roman, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R., and Nahrstedt, K. 2002. A middleware infrastructure for active spaces. *IEEE Pervasive Comput.* 1, 4, 74–83.
[34] Samulowitz, M., Michahelles, F., and Linnhoff-Popien, C. 2001. Adaptive interaction for enabling pervasive services. In *Proceedings of the 2nd ACM international workshop on Data engineering for wireless and mobile access*, 20–26.
[35] Schilit, B., Adams, N., and Want, R. Context-Aware Computing Applications, 85–90.
[36] Schilit, B. and Theimer, M. 1994. Disseminating active map information to mobile hosts. *IEEE Network* 8, 5, 22–32.
[37] Schilit, W. N. 1995. *A system architecture for context-aware mobile computing*, Columbia University.
[38] Schmidt, A., Beigl, M., and Gellersen, H.-W. 1999. There is more to context than location. *Computers & Graphics* 23, 6, 893–901.
[39] Sousa, J. P. and Garlan, D. 2002. Aura: An Architectural Framework for User Mobility in Ubiquitous Computing Environments. In *Proceedings of the IFIP 17th World Computer Congress-TC2 Stream/3rd IEEE/IFIP Conference on Software Architecture: System Design, Development and Maintenance*, 29–43.
[40] Tetsuo Yamabe, A. T. T. N. Citron: A Context Information Acquisition Framework for Personal Devices.
[41] Thomas Hofer, W. S. M. P. G. L. J. A. Context-awareness on mobile devices - the hydrogen approach - System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on.
[42] Truong, H.-L. and Dustdar, S. 2009. A survey on context-aware web service systems. *International Journal of Web Information Systems* 5, 1, 5–31.

[43] Vergori, P., Ntanos, C., Gavelli, M., and Askounis, D. 2013. The webinos Architecture: A Developer's Point of View. In *Mobile Computing, Applications, and Services. Fourth International Conference, Mobicase 2012, Seattle, Wa, USA, October 2012. Revised Selected Papers*, D. Uhler, K. Mehta and J. L. Wong, Eds. Springer-Verlag New York Inc, 391–399.
[44] Wang, X., Dong, J. S., Chin, C., Hettiarachchi, S. R., and Zhang, D. 2002. Semantic space: An infrastructure for smart spaces. *Computing* 1, 2, 67–74.
[45] Want, R., Hopper, A., Falcão, V., and Gibbons, J. 1992. The active badge location system. *ACM Trans. Inf. Syst.* 10, 1, 91–102.
[46] Weiser, M. 1991. The Computer for the 21st Century. *Scientific American* 265, 3, 66–75.
[47] Winograd, T. 2001. Architectures for Context. *Human-Comp. Interaction* 16, 2, 401–419.
[48] Κοντογιώργης, Α. *ΣΧΕΔΙΑΣΗ ΚΑΙ ΑΝΑΠΤΥΞΗ ΕΝΔΙΑΜΕΣΟΥ ΛΟΓΙΣΜΙΚΟΥ ΕΝΗΜΕΡΟΥ ΠΕΡΙΕΧΟΜΕΝΟΥ ΓΙΑ ΚΙΝΗΤΕΣ ΣΥΣΚΕΥΕΣ*. ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ.