Πανεπιστήμιο Πειραιώς
Τμήμα Ψηφιακών Συστημάτων
Πρόγραμμα Μεταπτυχιακών Σπουδών
Κατεύθυνση Δικτυοκεντρικών Συστημάτων

Διπλωματική Εργασία

«Σχεδιασμός και ανάπτυξη πλατφόρμας για την ενοποίηση και τη διαλειτουργικότητα γνωστικών υπηρεσιών σε μελλοντικά δίκτυα, με τη χρήση τεχνολογιών ενδιαμέσου λογισμικού (middleware) προσανατολισμένων σε υπηρεσίες.

A Service Oriented, Virtualization, platform for the integration and interoperability of cognitive services in future networks»

Επιμέλεια:

Δημήτριος Κελαϊδώνης, ΜΕ08080

Εισηγήτρια καθηγήτρια:

Βέρα Αλεξάνδρα Σταυρουλάκη

Πειραιάς, Οκτώβριος 2010

Ευχαριστώ την κυρία Β.Α. Σταυρουλάκη για την πολύτιμη βοήθεια της.

Πίνακας περιεχομένων

Περίληψη

Η ανάγκη για κάλυψη των απαιτήσεων που προκύπτουν από το μελλοντικό διαδίκτυο οδηγεί στην ανάπτυξη σύγχρονων συστημάτων που στοχεύουν στην συγκέντρωση ετερογενών τεχνολογιών, όπως δικτυακές υποδομές, συσκευές χρηστών, self-x αλγόριθμοι, εφαρμογές, κλπ., αποκρύπτοντας την πολυπλοκότητα των διαδικασιών ενσωμάτωσης, λειτουργίας και σύνθεσης. Οι σύγχρονες έρευνες εστιάζουν στην διαδικασία του Virtualization η οποία σε συνδυασμό με τις αρχές της Service Oriented αρχιτεκτονικής, αποτελεί βασικό πυρήνα για την υλοποίηση συστημάτων ικανών να ανταποκριθούν στις απαιτήσεις του μελλοντικού διαδικτύου. Η παρούσα μελέτη παρουσιάζει τον σχεδιασμό και την υλοποίηση μιας πλατφόρμας η οποία συνδυάζει την Service Oriented αρχιτεκτονική με την διαδικασία του Virtualization, για την ενοποίηση και διαλειτουργικότητα γνωστικών υπηρεσιών σε μελλοντικά δίκτυα. Αρχικά αναφερόμαστε στους λόγους ανάπτυξης της πλατφόρμας, ενώ συνεχίζουμε με την παρουσίαση του μελλοντικού διαδικτύου και την περιγραφή σύγχρονων αρχιτεκτονικών που εφαρμόζονται στα μελλοντικά δίκτυα. Επίσης υπάρχει αναλυτική περιγραφή όλων των τεχνολογιών που σχετίζονται με τα μελλοντικά δίκτυα. Στο κυρίως μέρος της μελέτης, αναφερόμαστε στην σχεδίαση και υλοποίηση της Service Oriented Virtualization Platform (SOVP), η οποία αποτελεί μια πλατφόρμα, δυναμικά επεκτάσιμη και αναδιαρθώσιμη όπου με την εφαρμογή της διαδικασίας του Virtualization, επιτρέπει την εισαγωγή νέων συνιστωσών στο σύστημά της με Plug and Play τρόπο. Η οργάνωση της SOVP ως Service Oriented σύστημα, επιτρέπει την σύνθεση των γνωστικών υπηρεσιών για την αντιμετώπιση πολύπλοκων διαδικασιών που μπορεί να προκύψουν στο σύστημα, ενώ έχει την δυνατότητα δυναμικής διαχείρισης των συνιστωσών της. Ολοκληρώνοντας, αφού παρουσιάσουμε τα αποτελέσματα εκτέλεσης και επιδόσεων της πλατφόρμας, πραγματοποιούμε αναφορά στα συμπεράσματα της παρούσας μελέτης και προτείνουμε μελλοντικές επεκτάσεις του συστήματος της SOVP.

Λέξεις κλειδιά: Future Internet, Service Oriented Architecture, Virtualization, Cognitive Services, Plug and Play, Reconfigurable Systems, Multi-agent systems, JADEX.

Abstract

The need to cover Future Internet requirements leads to the development of systems aimed to concentration of heterogeneous technologies, network infrastructures, user devices, self-x algorithms, applications, etc., hiding the complexity of processes of integration, operation, and composition. Current research focused on the process of virtualization which in combination with principles of Service Oriented Architecture (SOA) constitutes the core to implement systems capable to meet the requirement of future internet. This study presents the design and implementation of a platform that combines the Service Oriented Architecture with the process of virtualization, for the integration and interoperability of cognitive services in future networks. Initially, referring to the purposes of development of the platform, while continuing with the presentation of future internet technologies and the description of architectures that is applied in the future networks. In the main part of study, referring to the design and implementation of Service Oriented, Virtualization, Platform (SOVP), which is a platform dynamically extensible and reconfigurable where the implementation of virtualization process, allows the introduction of new components with Plug and Play way. The SOVP organized as a Service Oriented system which allows the composition of cognitive services, to address complex processes that may occur in the system and also manage with dynamic manner the components. In conclusion, after presenting the results and describe the platform performance, makes reference to the conclusions of this study and suggest future extensions of platform SOVP.

Keywords: Future Internet, Service Oriented Architecture, Virtualization, Cognitive Services, Plug and Play, Reconfigurable Systems, Multi-agent systems, JADEX.

Μέρος Α΄

«Επισκόπηση του μελλοντικού διαδικτύου και των τεχνολογιών του»

✓ **Κεφάλαιο 1**: «*Εισαγωγή*»

Η ταχύτατη εξέλιξη των δικτυακών τεχνολογιών παρέχει διαρκώς όλο και περισσότερες δυνατότητες στον χώρο του διαδικτύου. Μελετώντας προσεκτικά την ανάπτυξη των παρεχόμενων υπηρεσιών από το διαδίκτυο μπορούμε εύκολα να διαπιστώσουμε την αλματώδη πρόοδο στις τεχνολογίες επικοινωνιών και δικτύωσης. Στα τέλη της δεκαετίας του 1960 επιτυγχάνεται για πρώτη φορά η δημιουργία ενός ενιαίου δικτύου που μπορεί να συγκεντρώσει περισσότερους από έναν υπολογιστές.(1) Η εμφάνιση του σημερινού διαδικτύου αποτελεί πλέον γεγονός. Περνώντας από τρείς γενιές εξέλιξης, Web 1.0(web), Web 2.0(user generated content) και Web 3.0(collaborative production, semantic web), το διαδίκτυο αποτελεί πλέον την σημαντικότερη υποδομή πληροφοριών, υπηρεσιών και δικτύωσης.(2)

Στα τέλη της δεκαετίας του 2000 το ερευνητικό ενδιαφέρον στρέφεται σε μία νέα προοπτική εξέλιξης του τρέχοντος διαδικτύου στην οποία αποδίδεται η ονομασία **Future Internet**. Στην προσπάθεια να δώσουμε έναν εύστοχο ορισμό στον παραπάνω όρο μπορούμε να πούμε το εξής. «*Ο όρος Future Internet (μελλοντικό διαδίκτυο) αναφέρεται στις παγκόσμιες ερευνητικές δραστηριότητες που αφιερώνονται στην περαιτέρω ανάπτυξη του αρχικού διαδικτύου*».(3)

Βασικές ανεπάρκειες που παρουσιάζονται στην τωρινή μορφή του διαδικτύου και αφορούν τους τομείς απόδοσης, αξιοπιστίας, διαλειτουργικότητας, ενοποίησης υπηρεσιών, ασφάλειας καθώς και πλήθος άλλων, ωθεί το ερευνητικό ενδιαφέρον σε μια προσπάθεια κάλυψης και αντιμετώπισης των αναγκών αυτών. Οι ετερογενείς τεχνολογίες που σχετίζονται με το διαδίκτυο και εκτίνονται από τα «χαμηλότερα» στα «υψηλότερα» επίπεδα (lower to higher layers) υλικού (hardware) και λογισμικού (software), προξενούν το ενδιαφέρον για περεταίρω έρευνα. Λαμβάνοντας υπόψη μας τα υπαρκτά ερευνητικά συμπεράσματα μπορούμε να αναφέρουμε ότι η προσπάθεια μετάβασης από το τρέχον διαδίκτυο στο μελλοντικό διαδίκτυο, περιλαμβάνει την διαχείριση δικτύων (**Network management**), την εικονοποίηση των πόρων – δίκτυα, συσκευές και λογισμικό – (**Virtualization**) και μία προσέγγιση ενιαίου δικτύου πληροφοριών (**Network of information**).(3)

Η συγκεκριμένη μελέτη πραγματοποιείται με σκοπό να συμβάλει στην έρευνα για το Future Internet και θα αναφερθεί κυρίως σε θέματα που αφορούν τους τομείς διαχείρισης δικτύων (**Network management**) και εικονοποίηση πόρων (**Virtualization**). Η *διαχείριση δικτύων* αναφέρεται στις δραστηριότητες, τις μεθόδους, τις διαδικασίες, και τα εργαλεία που συμβάλουν στην οργάνωση και διαχείριση των δικτυωμένων συστημάτων.(4) Επίσης η διαχείριση δικτύων περιλαμβάνει βασικές αρχές δικτύων που περιγράφουν τις διαφορετικές τεχνολογίες που χρησιμοποιούνται σε ένα δίκτυο αλλά και πώς σχετίζονται μεταξύ

---

[1] **Internet**, http://en.wikipedia.org/wiki/Internet, πρόσβαση: Μάρτιος, 2010.

[2] George Tselentis, John Domingue, Alex Galis, Anastasius Gavras, David Hausheer, Srdjan Krco, Volkmar Lotz, Theodore Zahariadis, "**Towards the future internet A European Research Perspective**", IOS Press BV, 2009.

[3] **Future Internet**, http://en.wikipedia.org/wiki/Future_Internet, πρόσβαση: Μάρτιος, 2010.

[4] **Network Management**, http://en.wikipedia.org/wiki/Network_Management , πρόσβαση: Μάρτιος, 2010.

τους.[5] Η *εικονοποίηση (virtualization) πόρων* είναι η διαδικασία που «συνενώνει» του πόρους υλικού και λογισμικού ενός δικτύου ή κατ' επέκταση ενός συστήματος, σε μια κοινή οντότητα.[6] Η εικονοποίηση διαχειρίζεται όλα τα τερματικά και τις υπηρεσίες, ως μια ενιαία ομάδα πόρων,[7] διαθέσιμων μέσα από ένα σύνολο προσανατολισμένων υπηρεσιών (**Service Oriented**). Επιπλέον το **Virtualization** αποτελεί διαδικασία η οποία **βασίζεται** στην **Service Oriented αρχιτεκτονική** (SOA) οι αρχές της οποίας συμβάλουν στην πραγματοποίηση του, επιτυγχάνοντας την διαχείριση πολύπλοκων συστημάτων μέσω της εικονικοποίησης τους, ενώ παράλληλα ελαχιστοποιεί το υψηλό κόστος εφαρμογής των ήδη υπαρκτών SOA τεχνικών για την ανάπτυξη των σύγχρονων πολύπλοκων συστημάτων.[8]

Η απαιτήσεις που προκύπτουν από το Future Internet μπορούν να ικανοποιηθούν σε μεγάλο βαθμό από το Virtualization. Η υιοθέτηση αρχών της SOA από το Virtualization έχει ως συνέπεια τον σχεδιασμό και την υλοποίηση συστημάτων που βασίζονται στην Service Oriented Computing (SOC) τεχνολογία. Η εφαρμογή της SOC μπορεί να συνεισφέρει στην ανάπτυξη Service Oriented συστημάτων τα οποία ενοποιούν ετερογενείς τεχνολογίες που μπορούν να επαναπροσδιοριστούν και να προσπελαστούν σύμφωνα με τις απαιτήσεις που ορίζονται κάθε φορά.[9][10]

Στόχος μας λοιπόν είναι να παρουσιάσουμε ένα καλά ορισμένο σύστημα που θα μπορεί να ανταπεξέλθει στις απαιτήσεις του Future Internet και θα υλοποιείται με την ανάπτυξη μιας Service Oriented Virtualization Platform η οποία θα διαμορφώνεται σύμφωνα με τις αρχές της service oriented αρχιτεκτονικής και επιπλέον θα συγκεντρώνει χαρακτηριστικά που προέρχονται από τους τομείς του Network Management και Virtualization.

---

[5] **Network Management Fundamentals**, http://www.ciscopress.com/bookstore/product.asp?isbn=1587201372#,πρόσβαση: Μάρτιος, 2010.

[6] http://en.wikipedia.org/wiki/Network_virtualization, πρόσβαση: Μάρτιος, 2010.

[7] A. Nakao, "**Network virtualization for future Network Architecture and testbeds**", University of Tokyo, 14-10-2009.

[8] D. C. Plummer, "**Software Architectures will evolve from SOA and Events to Service Virtualization**", Gartner, 9-3-2005, σελ. 4.

[9] N. Koustouris, V. Stavroulaki, "**A Service Oriented Platform for integration and interoperability of Cognitive Management Schemes in the Wireless B3G World"**, Piraeus.

[10] Vera Stavroulaki, Nikos Koutsouris, Kostas Tsagkaris, Panagiotis Demestichas , "**Virtualisation Platform for the introduction of cognitive systems in the Future Internet"**, Piraeus, March 2010.

✓ **Κεφάλαιο 2**: «*Αιτίες ανάπτυξης της πλατφόρμας*»

Το Future Internet προβλέπεται ότι θα συμπεριλάβει έναν πολύ μεγάλο αριθμό συσκευών και δικτυακών τεχνολογιών μέσω διαφορετικών εφαρμογών που θα παρέχουν την υποδομή τους προς το δίκτυο.[(11)] Οι συσκευές μπορεί να περιλαμβάνουν διάφορους τύπους υλικού όπως κινητές συσκευές, υπολογιστές, PDAs, network gadgets, Wireless Base Stations, και άλλες μορφές τερματικών. Οι δικτυακές τεχνολογίες περιλαμβάνουν ένα πολύ μεγάλο πλήθος διαφορετικών τεχνολογιών δικτύωσης όπως GSM, GPRS, UMTS, WLAN IEEE 802.11, HSPA, HSPA+, IMS(3GPP Release 5), LTE(3GPP Release 8 – The LTE Release), W-CDMA(a spread - spectrum modulation technique) και λοιπές.[(12)(13)]

Λαμβάνοντας υπόψη το μεγάλο εύρος διαφορετικών τεχνολογιών τις οποίες επιχειρεί να συμπεριλάβει το Future Internet, διαπιστώνουμε την βασικότερη ανάγκη που πρέπει να καλυφθεί και αναφέρεται στην ετερογένεια των τεχνολογιών αυτών. Η προσπάθεια για ενοποίηση ετερογενών τεχνολογιών σε μια ενιαία οντότητα (π.χ. Cognitive platform) έχει ως αφετηρία της, την έρευνα που αφορά την Service Oriented Device Architecture (**SODA**) . *Η SODA αποτελεί μια προσαρμογή της SOA η οποία ενσωματώνει τα συστήματα υλικού σε ένα σύνολο υπηρεσιών βασιζόμενων στην SOA*.[(14)] Η ανάγκη για δημιουργία διεπαφών μεταξύ του «φυσικού κόσμου» (**Physical World**) και του «κόσμου του λογισμικού» (**Software World**) είναι συνεχώς αυξανόμενη. Το στοιχείο της πολυπλοκότητας αποτελεί τον σημαντικότερο παράγοντα προς αντιμετώπιση προκειμένου να επιτευχθεί η ενοποίηση ενός μεγάλου συνόλου συσκευών του Physical World με τα κατανεμημένα συστήματα του Software World. Σύμφωνα με την SODA η ενοποίηση μπορεί να πραγματοποιηθεί με τον σχεδιασμό και την υλοποίηση διαφορετικών υπηρεσιών. *Οι υπηρεσίες είναι στοιχεία λογισμικού με καλά ορισμένες διεπαφές και είναι ανεξάρτητες από την γλώσσα προγραμματισμού ή την πλατφόρμα στην οποία εκτελούνται*.[(15)]

Η SODA ως επέκταση της SOA συνδέεται άμεσα με το Service Oriented Computing (**SOC**). Ενώ η SODA, όπως θα δούμε στο κεφάλαιο 5, επιχειρεί να ενοποιήσει υλικό και λογισμικό χρησιμοποιώντας τον Enterprise Service Bus (ESB) ως έναν ενιαίο πάροχο προσανατολισμένων υπηρεσιών, η ιδέα του SOC εξελίσσει ακόμη περισσότερο την SOA. *Η Service Oriented Computing τεχνολογία χρησιμοποιεί τις υπηρεσίες ως θεμελιώδη στοιχεία για την ανάπτυξη εφαρμογών*. Περιλαμβάνει τις διαδικασίες περιγραφής, δημοσίευσης, ανακάλυψης, αφαίρεσης και αναπροσαρμογή των προσανατολισμένων υπηρεσιών σε ένα Service Oriented σύστημα.[(16)]

---

[11] βλ. παρ.10, **σελ.** 3.

[12] **Devices**, http://en.wikipedia.org/wiki/Devices, πρόσβαση: Μάρτιος, 2010.

[13] **3GPP A Global Initiative**, http://www.3gpp.org/, πρόσβαση: Μάρτιος, 2010.

[14] **Service Oriented Device Architecture**, http://en.wikipedia.org/wiki/Service_Oriented_Device_Architecture, πρόσβαση: Μάρτιος, 2010.

[15] S. Deugd, R. Carroll, K. Kelly, B. Millett, J. Ricker, "**SODA: Service Oriented Device Architecture**", S. Hetal, University Of Florida, 2006.

[16] M.P. Papazoglou, D. Georgakopoulos, "**Service Oriented Computing**", Vol. 46, No. 10 COMMUNICATIONS OF THE ACM, Οκτώβριος 2003 .

Η πολυπλοκότητα των τεχνολογιών αλλά και η ετερογένεια μεταξύ τους, δύναται να αντιμετωπιστούν με την χρήση των αρχών που διέπουν την Service Oriented Computing τεχνολογία. Παράλληλα οι έρευνες που σχετίζονται με τα Future Networks, εστιάζουν κυρίως στην τεχνολογία των Cognitive συστημάτων, προκειμένου να πετύχουν την ενοποίηση και την διαλειτουργικότητα των υπηρεσιών σε ενιαίες cognitive πλατφόρμες (πχ Wireless Cognitive Systems B3G).(17)

Τα Cognitive συστήματα βασίζονται στην ιδέα ύπαρξης μιας κοινής πλατφόρμας η οποία θα συγκεντρώνει ετερογενείς τεχνολογίες τις οποίες θα μπορεί να τις διαχειριστεί και να τις συνδυάσει. Στόχος τους είναι η ανάπτυξη συστημάτων τα οποία θα είναι σε θέση να αλληλεπιδράσουν με άλλα συστήματα και θα μπορούν να αναδιαρθρωθούν δυναμικά αποκρύπτοντας την πολυπλοκότητα και την ετερογένεια των τεχνολογιών τους.(18) Σύμφωνα με τα παραπάνω παρατηρούμε ότι οι στόχοι που επιδιώκει να επιτύχει το SOC μπορούν να ικανοποιηθούν από ένα καλά ορισμένο cognitive σύστημα το οποίο θα παρουσιάζεται ως μια ενιαία cognitive πλατφόρμα που συνδυάζει ετερογενείς τεχνολογίες απεικονίζοντάς τες ως υπηρεσίες που υπακούν στις αρχές της SOA και μπορούν να αλληλεπιδράσουν δυναμικά σε ένα Service Oriented Cognitive σύστημα.

Τα Future Networks και κατ' επέκταση το Future Internet στοχεύουν στην συγκέντρωση ετερογενών τεχνολογιών δικτύωσης και συσκευών, σε μια κοινή πλατφόρμα λειτουργίας στην οποία θα μπορούν να αλληλεπιδράσουν δυναμικά με την εφαρμογή υπηρεσιών. Η εικονοποίηση των δικτύων (Virtualization), όπως ήδη αναφέραμε, αποτελεί ένα από τα ερευνητικά θέματα που σχετίζονται με το Future Internet. Η **ενοποίηση** και η **διαλειτουργικότητα** των υπηρεσιών σε μια πλατφόρμα που «υπακούει» στις αρχές της SOA, μπορεί να γίνει εφικτή εάν εφαρμόζουμε την διαδικασία του **virtualization** των πόρων (συσκευές υλικού, συστήματα αποθήκευση, δικτυακές τεχνολογίες, κλπ.). Το virtualization εξ' ορισμού υπακούει στα πρότυπα του **Service Orientation**(19) και ο συνδυασμός του με μία πλατφόρμα που διαμορφώνεται συμφώνα με τους όρους της SOA, δεν θα αποτελέσει πρόβλημα.

Καταλήγουμε λοιπόν στο **συμπέρασμα** ότι, για να ανταπεξέλθουμε στις απαιτήσεις του Future Internet και των Future Networks θα πρέπει να σχεδιάσουμε και να αναπτύξουμε συστήματα τα οποία συγκεντρώνουν χαρακτηριστικά από την cognitive τεχνολογία, υπακούν στις αρχές της Service Oriented αρχιτεκτονικής και ενοποιούν στοιχεία υλικού και λογισμικού μετατρέποντάς τα σε υπηρεσίες με την βοήθεια του virtualization. Ένα τέτοιο σύστημα, σύμφωνα με την μελέτη μας, θα μπορούσε να υλοποιηθεί με την μορφή μιας Service Oriented πλατφόρμας η οποία θα έχει την ονομασία "**A Service Oriented Virtualization Platform**" και θα στοχεύει στην δυναμική ενοποίηση ετερογενών τεχνολογιών, για την κάλυψη των απαιτήσεων του Future Internet.

---

[17] **End-to-End Efficiency (E^3), Project Overview**, https://ict-e3.eu/project/overview/overview.html, πρόσβαση: Μάρτιος, 2010.

[18] Jan Larsen, "**Cognitive Systems**", Technical University of Denmark, 18-10-2008.

[19] βλ. παρ.8**, σελ.** 3.

✓ **Κεφάλαιο 3**: «*Επισκόπηση του μελλοντικού διαδικτύου και των μελλοντικών δικτύων*»

Η τωρινή μορφή του διαδικτύου (Current Internet – στην συνέχεια θα αναφέρεται ως C.I.) βασίζεται σε μια απλή αρχιτεκτονική η οποία χρησιμοποιεί μια δικτυακή υπηρεσία ως ένα παγκόσμιο μέσο διασύνδεσης διαφόρων συστημάτων.[(20)] Η αρχή που διέπει το C.I. παρέχει την δυνατότητα σε εφαρμογές και τερματικά να επικοινωνούν μεταξύ τους με την χρήση μιας μοναδικής διεύθυνσης (address path) αποστολέα / παραλήπτη ανταλλάσοντας πληροφορίες με την μορφή πακέτων, για τα οποία όμως δεν υπάρχει καμία εγγύηση ορθής παράδοσης. Επίσης στο C.I. παρατηρούνται ανεπάρκειες σε θέματα απόδοσης των δικτύων όπως η διαχείριση της συμφόρησης, κλπ.

Μακροχρόνιες μελέτες σχετικά με το διαδίκτυο απέδειξαν ότι υπάρχουν αρκετές ανεπάρκειες σε τομείς δικτύωσης, επικοινωνίας, υπηρεσιών, κλπ. Στην συνέχεια παραθέτουμε εν συντομία τις ελλείψεις που έχουν καταγραφεί κατά καιρούς από τις έρευνες αντιστοιχίζοντάς τες με μια σειρά λειτουργιών που είναι απαραίτητες για την κάλυψη τους. Αυτές είναι οι εξής:[(21)]

- Διοικητική λειτουργία των δικτύων και συγκεκριμένα αυτοδιαχείριση κάθε δικτύου, (**self-management functionality**).
- Δυναμική προσθήκη νέων λειτουργιών σύμφωνα με τις ανάγκες των τεχνολογιών δικτύωσης συμπεριλαμβανομένης και την ενεργοποίηση νέων υπηρεσιών κατόπιν απαίτησης, (**Dynamically addition new functions via services**).
- Λειτουργίες διαχείρισης ενός μεγάλου εύρους υπηρεσιών με σκοπό την ενοποίηση τους σε επίπεδο υπηρεσιών και δικτύωσης, (**Dynamic management for services and networks integration**).
- Λειτουργίες για την ενορχήστρωση της ασφάλεια, της αξιοπιστίας, της ευρωστίας, των υπηρεσιών και διαχείριση των πόρων δικτύωσης και υπηρεσιών, (**Resources and functions orchestration**).
- Κινητικότητα δικτύων, υπηρεσιών και συσκευών, (**Network, Services and Devices Mobility**).
- Λειτουργίες και εγκαταστάσεις για την υποστήριξη της ποιότητας των υπηρεσιών και του επιπέδου συμφωνίας – συνεργασία διαφορετικών υπηρεσιών, (**Quality of Services & Service Level Agreements**).
- Μηχανισμοί ασφαλείας για την προστασία των διανεμόμενων δεδομένων στα δίκτυα, (**Security mechanisms for distributed data**).
- Επαρκής σχεδιασμός όπου η ταυτότητα και η τοποθεσία κάθε στοιχείου δεν ενσωματώνονται στην ίδια διεύθυνση, (**Adequate addressing scheme**).

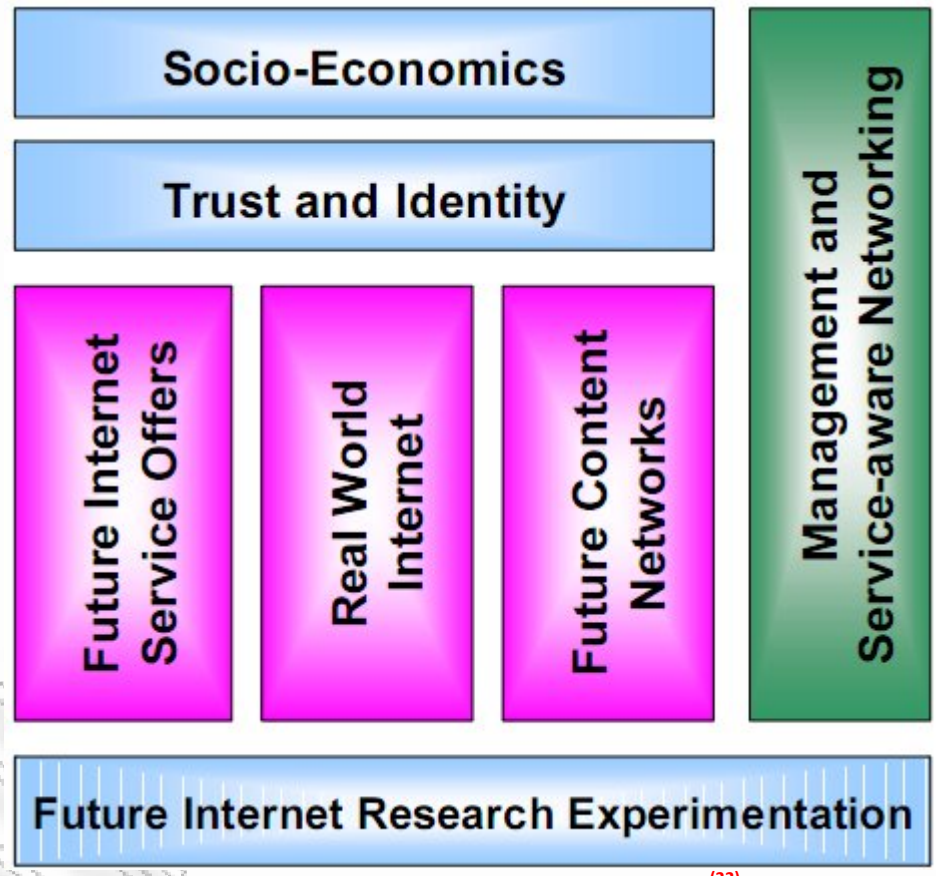
---

[20] George Tselentis, John Domingue, Alex Galis, Anastasius Gavras, David Hausheer, Srdjan Krco, Volkmar Lotz, Theodore Zahariadis, "**Towards the future internet A European Research Perspective**", IOS Press BV, 2009, σελ. Ix-x.

[21] βλ. παρ.18, **σελ.** 5.

- Αλληλεπίδραση του Software World με τον Physical World με σκοπό την βελτίωση των υπαρκτών υπηρεσιών και την δημιουργία νέων (**Interact with physical world**).
- Διαχείριση κοινωνικοοικονομικών ελλείψεων που αναφέρονται σε νομικά και ρυθμιστικά ζητήματα αλλά και σε θέματα ασφάλειας και μυστικότητας, (**Socio-economic aspects**) .
- Ενεργειακά θέματα, (**Energy awareness**).

Η ιδέα για την ανάπτυξη του μελλοντικού διαδικτύου – Future Internet (στο εξής θα αναφέρεται ως F.I.) αποσκοπεί στην αντιμετώπιση αυτών των ελλείψεων και μέσα από το σχεδιασμό και την υλοποίηση νέων συστημάτων επιχειρεί να εξαλείψει τις ελλείψεις αυτές, δημιουργώντας μια νέα μορφή **ενοποιημένης** και **διαλειτουργικής** δικτύωσης η οποία θα καταστήσει δυνατή την ανάπτυξη των δικτύων του μέλλοντος. Οι προκλήσεις για το F.I. που καταγράφονται παραπάνω δημιουργούν ένα σύνολο από διαφορετικές ερευνητικές ενότητες οι οποίες απεικονίζονται στην εικόνα 1.



Εικόνα 1:"Τομείς έρευνας σχετικά με το Future Internet"[(22)]

Στο σημείο αυτό είναι σημαντικό να διευκρινίσουμε τους τομείς που συμπεριλαμβάνονται στην παρούσα μελέτη. Εν συντομία λοιπόν αναφέρουμε ότι μέσα από την μελέτη αυτή θα επιχειρήσουμε να καλύψουμε θέματα που αναφέρονται στους τομείς **Υπηρεσιών και Δικτύωσης συστημάτων** στο F.I. και στον

[22] βλ. παρ.18, **σελ.** 5.

τομέα **διαχείρισης και ενοποίησης** αυτών μέσα από ένα σύστημα διαμορφωμένο σύμφωνα με τα πρότυπα της Service Oriented Computing (SOC) και υλοποιημένο με την μορφή μιας Service Oriented πλατφόρμας που θα συγκεντρώνει πάνω της ένα σύνολο από ετερογενείς τεχνολογίες τις οποίες θα πρέπει να ενοποιήσει και να διαμοιράσει μέσα από ένα δυναμικά αναδιαρθρωνόμενο περιβάλλον. Προτού προχωρήσουμε στην μελέτη της Service Oriented πλατφόρμας θα πρέπει να έχουμε μελετήσει θέματα που αφορούν την προτεινόμενη και προβλεπόμενη αρχιτεκτονική σύμφωνα με την οποία θα δομηθεί η νέα τεχνολογία του F.I. Το κεφάλαιο 4 αναφέρεται σε θέμα που αφορούν την αρχιτεκτονική του Future Internet, έτσι όπως διαμορφώνεται μέσα από την σύγχρονη έρευνα.

✓ **Κεφάλαιο 4**: «*Η αρχιτεκτονική του Future Internet*»

Η ιδέα του F.I. δημιούργησε την ανάγκη για την ανάπτυξη μιας νέας αρχιτεκτονικής για το διαδίκτυο η οποία αποκλείει στοιχεία από την τωρινή αρχιτεκτονική προσθέτοντας νέα, αλλά και υιοθετεί στοιχεία τα οποία εξελίσσει. Υπάρχει πληθώρα ερευνών που σχετίζονται με τον τομέα της αρχιτεκτονικής του F.I. και η πρόοδος είναι θεαματική. Η επισκόπηση της αρχιτεκτονικής που παρουσιάζεται στη συνέχεια, έχει βασιστεί στην έρευνα «**Future Internet The cross-ETP Vision Document**» η οποία ολοκληρώθηκε τον Ιανουάριο του 2009.

4.1 Βασικές αρχές σχεδιασμού του Future Internet

Ο σχεδιασμός του F.I. εισάγει τρεις νέες αρχές που αφορούν την δόμηση του. Συγκεκριμένα αναφέρεται στις εξής: [(23)]

- **Αρχή τοποθέτησης και αυτονομίας** (**Situated and Autonomic**): αρχή η οποία αναφέρεται σε μία πρότυπη αρχιτεκτονική που θα επιτρέπει την ευέλικτη, δυναμική και αυτόνομη διαμόρφωση των κόμβων ενός δικτύου, αλλά και των επιμέρους δικτύων. Η αρχή αυτή θα επιτρέψει στο δίκτυο να αναπροσαρμόζεται δυναμικά, σύμφωνα με τις συνθήκες λειτουργίας του και τις ανάγκες των χρηστών του.
- **Αρχή συνεργασίας με το δίκτυο φιλοξενίας (Host-network cooperation):** σχεδιαστική αρχή η οποία καλύπτει θέματα που αφορούν την διασύνδεση ενός δικτύου με τους τελικούς του χρήστες. Επειδή οι ανάγκες των χρηστών σε ένα δίκτυο είναι μεταβαλλόμενες, θα πρέπει να σχεδιάζεται με τέτοιο τρόπο ώστε να μπορεί να ανταπεξέλθει σε αυτές. Η δυναμική και κλιμακούμενη αλλαγή των χαρακτηριστικών κάθε σύνδεσης του δικτύου με τον τελικό χρήστη του, μπορεί να ανταπεξέλθει στις απαιτήσεις του βελτιώνοντας την μεταξύ τους επικοινωνία.
- **Αρχή της αφαιρετικότητας (Abstraction):** αρχή σχεδιασμού η οποία στοχεύει στην απόκρυψη της πολυπλοκότητας κατά την σύνδεση ή συνεργασία διαφορετικών τεχνολογιών. Επιδιώκει την ομαλή επικοινωνία με την ανταλλαγή μηνυμάτων και πληροφοριών μεταξύ του δικτύου και εξωτερικών οντοτήτων με διαφορετική τεχνολογία.

4.2 Χαρακτηριστικά του Future Internet[(24)]

Ο σχεδιασμός του F.I. (και γενικότερα των Future Networks) συμπεριλαμβάνει ένα σύνολο χαρακτηριστικών που αντιστοιχούν στις **δικτυακές τεχνολογίες**, **τα συστήματα**, και **τις υπηρεσίες** που σχετίζονται με αυτό. Στη συνέχεια θα αναφερθούμε σε αυτές τις κατηγορίες χαρακτηριστικών οι οποίες είναι: i) στοιχεία δικτύωσης και συσκευών και ii) στοιχεία υπηρεσιών.

---

[23] European Technology Platforms, "**Future Internet The cross-ETP Vision Document**", D. Papadimitriou, 8-1-2009 , σελ. 45-56.

[24] βλ. παρ.20, **σελ.** 6.

4.2.1 Future Internet – Δικτυακές τεχνολογίες και συσκευές

Το διαδίκτυο αποτελείται από την διασύνδεση ετερογενών δικτύων η οποία βασίζεται σε ένα κατανεμημένο σύστημα δρομολόγησης αυτόνομων συστημάτων με ανεξάρτητη διαχείριση. Επιπλέον συγκεντρώνει δυο τύπους συνεργαζόμενων συστημάτων, τις **συσκευές χρηστών** (τερματικά, υπολογιστές, PDA, κλπ) και τους **δρομολογητές** (routers). Ο μηχανισμός δρομολόγησης λειτουργεί με την βοήθεια των Routing Information Bases και Forwarding Information Base σε συνδυασμό με αλγόριθμους δρομολόγησης που χρησιμοποιούν τις πληροφορίες για να διεξαχθεί η ομαλή και γρήγορη επικοινωνία μέσα στο δίκτυο. Οι διαδικασίες δρομολόγησης παραμένουν και στην αρχιτεκτονική του F.I. με την διαφορά ότι μπορούν να εξελιχτούν και να αναπροσδιοριστούν δυναμικά από το δίκτυο. Για να καταστεί εφικτή η εξέλιξη της επικοινωνία μεταξύ των δικτύων το F.I. εισάγει τρία νέα «συστατικά» που αντιστοιχούν στα **cognitive components**, **situation awareness components** και **autonomic components**. Ακολουθεί η ανάλυση τους.

4.2.1.1 Cognitive component

Παρέχει την δυνατότητα στο δίκτυο να μελετά και να μαθαίνει την συμπεριφορά των χρηστών του (είτε πρόκειται για άλλα δίκτυα είτε για μεμονωμένους χρήστες) με την χρήση σύγχρονων γνωστικών τεχνικών όπως το **knowledge management**. Καλύπτει ανεπάρκειες που αφορούν θέματα επίδοσης, διαθεσιμότητας και διαχείρισης της συμφόρησης, που εμφανίζονται συχνά στο C.I. Έτσι εισάγοντας το cognitive component αντιμετωπίζουμε καταστάσεις, όπως η ξαφνική αύξηση τελικών χρηστών με ετερογενείς τεχνολογίες, για τις οποίες δεν υπήρχε αρχικός σχεδιασμός.

4.2.1.2 Situation awareness component

Η συνειδητοποίηση μιας κατάστασης που μπορεί να επέλθει το δίκτυο αποτελεί σημαντικό παράγοντα στην απόδοση του και στην αντιμετώπιση των προβλημάτων που μπορεί να προκύψουν. Ο υπάρχον σχεδιασμός του δικτύου έχει αποδεχτεί πολλές φορές αναποτελεσματικός σε καταστάσεις όπου το δίκτυο πρέπει να αναδιοργανωθεί μεταβάλλοντας κάποια από τα χαρακτηριστικά του. Η διαδικασία της συνειδητοποίησης καταστάσεων σε ένα δίκτυο γίνεται με την ανταλλαγή πληροφοριών μεταξύ του δικτύου και κόμβων εντός και εκτός των ορίων του. Συγκεντρώνοντας πληροφορίες για μια κατάσταση μπορεί κατόπιν μελέτης αυτών να δημιουργήσει σχέδια αντιμετώπισης προκειμένου να αποτρέψει πιθανά προβλήματα. Χαρακτηριστικό παράδειγμα αποτελεί η διαχείριση της συμφόρησης σε δίκτυα κινητών επικοινωνιών σε ώρες αιχμής για μια μεγάλη πόλη. Η τεχνική του **Decision Making** αποτελεί την λύση για την κάλυψη της σοβαρής αυτή έλλειψης που παρουσιάζει το C.I. Το Decision Making περιλαμβάνεται στο σύνολο των γνωστικών διαδικασιών (cognitive processes) που προκύπτουν

από την γνωστική επιστήμη (cognitive science) και αποδίδει στα cognitive συστήματα την δυνατότητα επιλογής σε περιπτώσεις ύπαρξης πολλών εναλλακτικών επιλογών.[25]

4.2.1.3 Autonomic components

Τα συστατικά αυτονομίας που προστίθενται στο F.I. συμπεριλαμβάνουν όλες εκείνες τις λειτουργίες που έχουν να κάνουν με θέματα αυτό-διαχείρισης (self-management) του δικτύου ή των στοιχείων που το αποτελούν. Τέτοιες λειτουργίες είναι:

- Self-configuration: αυτόματη προσαρμογή των στοιχείων.
- Self-Healing: αυτόματος εντοπισμός και επιδιόρθωση σφαλμάτων.
- Self-Optimization: αυτόματη παρακολούθηση και διαχείριση των διαθέσιμων πόρων ώστε να εξασφαλιστεί η ομαλή λειτουργία σύμφωνα με τις απαιτήσεις του συστήματος.
- Self-Protection: συνεχής προστασία από κακόβουλες επιθέσεις.

Τα autonomic components αποτελούν σημαντικό συστατικό για την ορθή δόμηση του F.I. και των Future Networks. Οι λειτουργίες που προσφέρουν βοηθούν το δίκτυο να αναδιαρθρώνεται δυναμικά, να έχει επίγνωση της κατάστασης του κάθε στιγμή αλλά και να μπορεί να αντιμετωπίσει ασυνήθιστες καταστάσεις όπως η συμφόρηση, η ανεπάρκεια παρεχόμενων υπηρεσιών, κλπ. Ωστόσο προκειμένου να εφαρμοστούν σωστά οι λειτουργίες αυτές και να μπορέσουν να αποδώσουν τα αναμενόμενα, θα πρέπει να αντιμετωπιστούν τα εξής προβλήματα:

- Εμφάνιση μη προβλέψιμων γεγονότων στο δίκτυο
- Περιορισμοί στη διαθεσιμότητα πόρων
- Συγχρονισμός της παρατήρησης
- Προστασία της μυστικότητας

Η μεταβολή των γεγονότων που μπορούν να συμβούν σε ένα δίκτυο είναι εξαιρετικά γρήγορη και καθόλου προβλέψιμη. Έτσι στο σημείο αυτό εντοπίζουμε ένα πολύ σημαντικό πρόβλημα στην χρήση των Autonomic components. Για να πραγματοποιηθεί η διαδικασία την λήψης πληροφοριών από το δίκτυο με σκοπό να χρησιμοποιηθούν για την αντίληψη της κατάσταση του και την αντιμετώπιση προβλημάτων θα πρέπει να υπάρχει μια διαρκής ενημέρωση των λειτουργιών που αποτελούν το Knowledge management το οποίο με την σειρά του θα αποτελέσει πηγή πληροφόρησης για τις λειτουργίες του Decision making. Πρακτικά για να καταστεί δυνατή η παραπάνω διαδικασία θα πρέπει να παρατηρούμε οποιοδήποτε πακέτο κινείται μέσα στο δίκτυο, λαμβάνοντας όλες τις πληροφορίες που το αφορούν. Αυτό ωστόσο

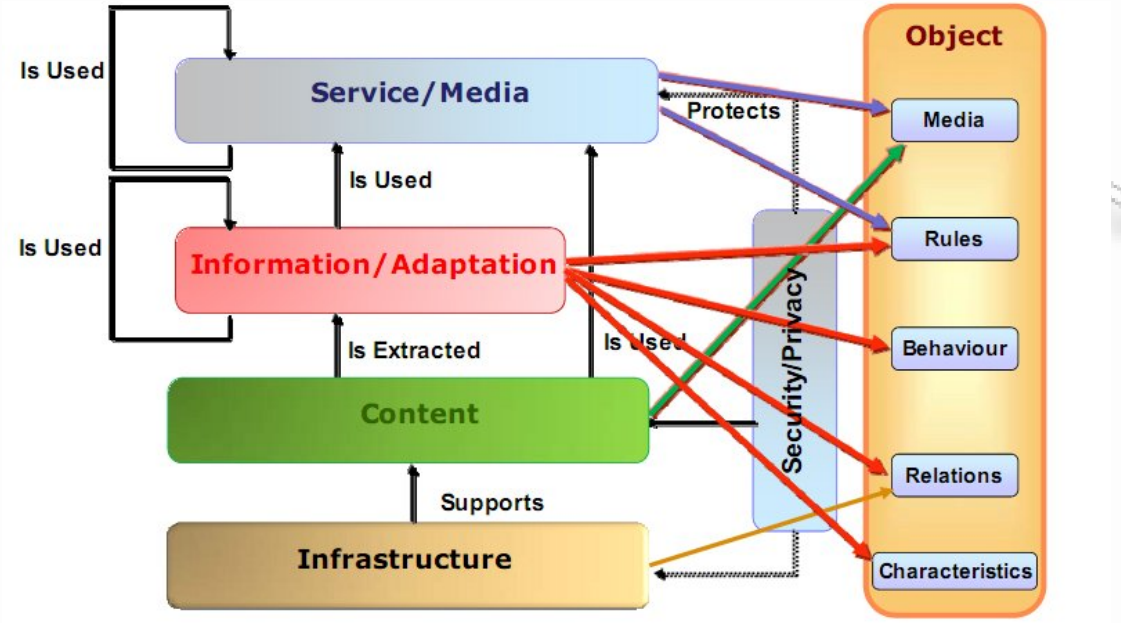
[25] **Decision Making**, http://en.wikipedia.org/wiki/Decision_making, πρόσβαση: Μάρτιος, 2010.

παραβιάζει την αρχή των προσωπικών δεδομένων, μιας και ένα πακέτο περιέχει πληροφορίες σχετικές με την ταυτότητα του χρήστη. Η ομαλή ενσωμάτωση λοιπόν, των στοιχείων αυτοδιαχείρισης στα Future Networks και κατ' επέκταση στο Future Internet, συνδέεται άμεσα με την πρόοδο στις τεχνολογίες μεταγωγής πακέτου μέσα στο δίκτυο όπου επιχειρείται μια προσπάθεια πλήρους διαχωρισμού της ταυτότητας του χρήστη από τις μεταφερόμενες πληροφορίες του πακέτου.

4.2.2 Future Internet – Content Centric χαρακτηριστικά

Το πρόβλημα που παρουσιάστηκε στη ενσωμάτωση στοιχείων που αφορούν την μελέτη γεγονότων στα δίκτυα (*εν. 4.2.1.3*) δημιούργησε νέες περιοχές ερευνητικού ενδιαφέροντος. Οι μελέτες στην προσπάθεια τους να καλύψουν την αδυναμία που παρουσιάστηκε, καταλήγουν το Μάιο του 2009 στην ιδέα του Content Centric Internet (στο εξής CCI). Το CCI έχει ως κύριο χαρακτηριστικό του την ύπαρξη ενός αντικειμένου περιεχομένου (**Content Object**) μέσα στο οποίο μπορεί να ενσωματωθεί ένα πλήθος διαφορετικών πληροφοριών που αφορούν: το περιεχόμενο του, την διαδικασία δρομολόγησης του, την συμπεριφορά του στο δίκτυο, το χαρακτηρισμό του και τους κανόνες που το διέπουν.[(26)] Έτσι μέσα από ένα σύνολο στοιχείων που μπορεί να ενσωματώσει το Content Object μπορεί να πραγματοποιηθεί και ο έλεγχος του αντικειμένου αυτού από τις λειτουργίες των autonomic components, χωρίς να προσβάλει κανέναν κανονισμό περί προσωπικών δεδομένων, ενώ παράλληλα με το στοιχείο του χαρακτηρισμού (**Content object - characteristics**) του κάθε αντικειμένου , το δίκτυο μπορεί να διακρίνει αν επιβάλλεται ο έλεγχος του ή όχι. Η εικόνα 2 παρουσιάζει τη αντιστοιχία των διαφορετικών επιπέδων της αρχιτεκτονικής του Future CCI (FCCI) με το Content Centric Object (CCO).

[26] FIA - Future Content Networks Group, "**Why do we need a Content-Centric Internet**", Μάιος, 2009.



Εικόνα 2:"Αντιστοιχία FCCI επιπέδων με τα χαρακτηριστικά του CCO"[27]

4.2.3 Future Internet – Service Components

Οι υπηρεσίες και η εφαρμογή τους στο FI, αποτελούν σημαντικό χαρακτηριστικό της αρχιτεκτονικής του. Η προσθήκη των service components, εξυπηρετούν την διαδικασία ενσωμάτωσης λειτουργιών που αφορούν τις υπηρεσίες στο FI. Στη συνέχεια παραθέτουμε και αναλύουμε τρία από τα βασικότερα , σύμφωνα με την έρευνα, service components.

4.2.3.1 Service Delivery Platforms

Το τρέχον μοντέλο παροχής υπηρεσιών πελάτη θα πρέπει να εξελιχθεί στο FI ώστε οι υπηρεσίες να μπορούν να διανέμονται δυναμικά στο δίκτυο, να είναι συμβατές με όλους τους χρήστες που υπάρχουν στο δίκτυο και να μπορούν να διανεμηθούν από έναν χρήστη σε έναν άλλο χωρίς την ύπαρξη προβλημάτων συμβατότητας. Αυτό επιβάλει την δημιουργία ενός κοινού πλαισίου εργασιών του Service Delivery Framework (SDF) το οποίο θα διαχειρίζεται την παροχή υπηρεσιών στα διάφορα δίκτυα. Το SDF επίσης, θα παρέχει λειτουργίες για την ομαλή συνεργασία υπηρεσιών από διαφορετικούς παρόχους, ενοποιώντας διαφορετικές υπηρεσίες σε παγκόσμιο επίπεδο δικτύωσης.

[27] βλ. παρ.23, **σελ.** 9.

4.2.3.2 Υιοθέτηση της Service Oriented Αρχιτεκτονικής

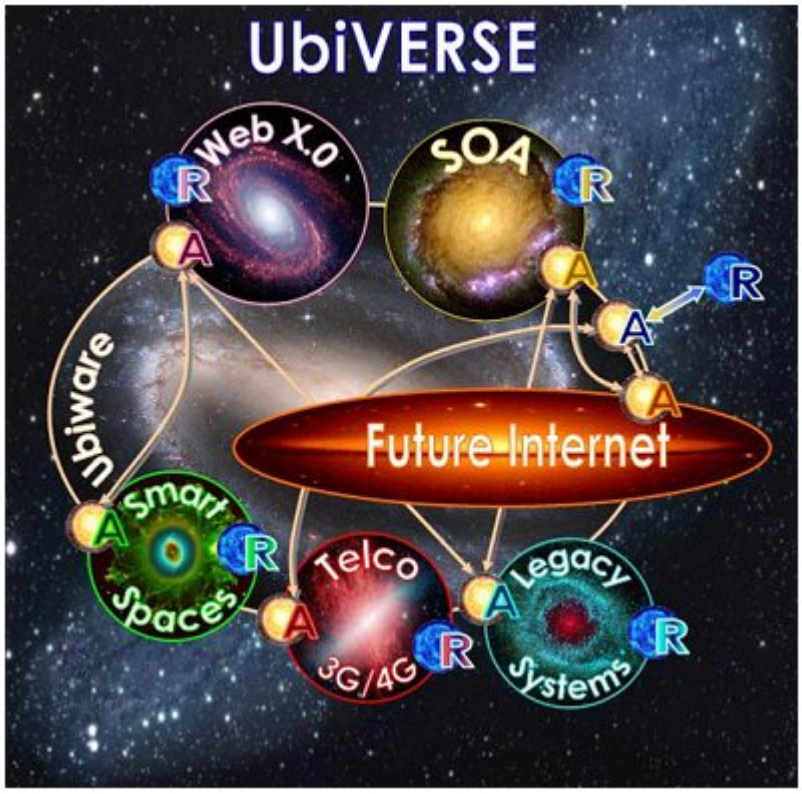
Η πλατφόρμα του Future Internet θα επεκταθεί με σκοπό να συμπεριλάβει στοιχεία από την SOA τα οποία θα της δώσουν την δυνατότητα να οργανώσει, να δημοσιεύσει και να περιγράψει τις υπηρεσίες της αλλά και να ανακαλύψει και να συνδυάσει υπηρεσίες. Με λίγα λόγια η πλατφόρμα του FI θα οργανωθεί σε επίπεδο υπηρεσιών σύμφωνα με την SOA. Αυτό αποτελεί ένα **συνδυασμό** της **SOA** με την **FI αρχιτεκτονική**.

4.3 Future Internet Αρχιτεκτονική – Συμπεράσματα

Έχοντας κάνει μια εκτεταμένη επισκόπηση της αρχιτεκτονικής του FI, και σύμφωνα με τα όσα αναφέραμε σχετικά με αυτή, διαπιστώνουμε ότι το FI προβλέπεται να συγκεντρώσει έναν τεράστιο όγκο πληροφοριών οι οποίες θα πρέπει να επεξεργαστούν και να χρησιμοποιηθούν για την εξαγωγή γνώσης που θα χρησιμοποιηθεί για την βελτίωση της δικτύωσης. Επίσης εύκολα γίνεται αντιληπτό ότι η αρχιτεκτονική του FI στοχεύει στην δημιουργία συστημάτων τα οποία θα ανταλλάσουν μεταξύ τους πληροφορίες, αποκρύπτοντας την πολυπλοκότητα των διαδικασιών από τους χρήστες. Έτσι θα μπορούσαμε να πούμε ότι σύμφωνα με τις αρχές του FI προωθείται ένα μοντέλο επικοινωνίας machine – to – machine, που στοχεύει στην διευκόλυνση των χρηστών.

✓ **Κεφάλαιο 5**: «*Service Oriented συστήματα για το μελλοντικό διαδίκτυο*»

Η μελλοντική κοινωνία της πληροφορίας η οποία θα δομηθεί από το Future Internet προβλέπεται ότι θα συμπεριλάβει ένα ευρύ φάσμα διαφορετικών αρχών υλοποίησης συστημάτων όπως: **Future Internet network architecture, Web X.0 περιβάλλοντα πληροφοριών, Service Oriented αρχιτεκτονικές (SOA) και Cognitive τεχνολογίες (Cognitive platforms, κλπ.)**.[(28)] Στην εικόνα 3 παρουσιάζεται η δομή της μελλοντικής κοινωνίας της πληροφορίας έτσι όπως αυτή θα διαμορφωθεί με την βοήθεια των αρχών που συγκεντρώνει το FI. Φυσικά, στην παρούσα μελέτη, δεν αποκλείουμε την μελλοντική προσθήκη επιπλέον κόμβων στο σχήμα της παρακάτω εικόνας.



Εικόνα 3:"Η μελλοντική κοινωνία της πληροφορίας με επίκεντρο το FI"[(29)]

Στο συγκεκριμένο κεφάλαιο, θα επιχειρήσουμε να παρουσιάσουμε το σύνολο των αρχών ανάπτυξης συστημάτων, για το Future Internet, που βασίζουν την λειτουργία τους στην Service Oriented αρχιτεκτονική. Θα αναφερθούμε σε τεχνολογίες που σχετίζονται με το FI και αφορούν την οργάνωση και διαχείριση των υπηρεσιών και την εισαγωγή γνωστικών συστημάτων σε αυτό.

Η διαχείριση των υπηρεσιών στο περιβάλλον του FI επιτυγχάνεται με την εισαγωγή στοιχείων από την Service Oriented Architecture – (**SOA**). Η SOA συνδέεται άμεσα με την Service Oriented Device αρχιτεκτονική – (**SODA**) και με την Service Oriented Computing – (**SOC**) τεχνολογία. Η SOC αποτελεί τεχνολογία

[28] V. Terziyan, D. Zhovtobryukh, A. Katasov, "Proactive Future Internet: Smart Semantic Middleware for Overlay Architecture", University of Jyvaskyla, Finland, 2009.
[29] βλ. παρ.20, **σελ.** 6.

ανάπτυξης εφαρμογών, που βασίζονται στις υπηρεσίες, η οργάνωση των οποίων πραγματοποιείται σύμφωνα με τις αρχές της SOA.[(30)] Η SODA αποτελεί από μόνη της μια ξεχωριστή αρχιτεκτονική η οποία εισάγει αρχές που αφορούν την εισαγωγή συσκευών σε Service Oriented περιβάλλοντα. Μάλιστα με την τεχνολογία των: ESB (**Enterprise Service Bus**) και BA (**Bus Adapter**) της SODA θα μπορούσαμε να πούμε ότι πραγματοποιήθηκε η πρώτη προσέγγιση της SOA στα αρχιτεκτονικά πρότυπα του FI.

Όπως γνωρίζουμε ήδη ότι το FI επιδιώκει να επιτύχει την μετατροπή των πόρων συστήματος σε υπηρεσίες και την ενσωμάτωση τους σε cognitive πλατφόρμες οι οποίες θα τις παρέχουν προς τους τελικούς χρήστες. Η ανάπτυξη των συστημάτων μπορεί να πραγματοποιηθεί σύμφωνα με τις αρχές δυο μοντέλων ανάπτυξης, τα κεντρικοποιημένα συστήματα και τα αποκεντρικοποιημένα συστήματα ή κατανεμημένα συστήματα. Τα κατανεμημένα συστήματα αποτελούν τον βέλτιστο δυνατό τρόπο ανάπτυξης αυτόνομων και με υψηλή βιωσιμότητα συστημάτων.

Στην συνέχεια, αφού πρώτα αιτιολογήσουμε τον λόγο που η συγκεκριμένη μελέτη εστιάζει στην Service Oriented αρχιτεκτονική (ως αρχή ανάπτυξης συστημάτων για το FI), θα επιχειρήσουμε να παραθέσουμε αναλυτικά τα σημαντικότερα στοιχεία που αφορούν τα κατανεμημένα συστήματα και την Service Oriented αρχιτεκτονική. Επιπλέον θα πραγματοποιήσουμε την επισκόπηση της Service Oriented Device αρχιτεκτονικής βάση της οποίας αναπτύσσεται η διαδικασία ενσωμάτωσης συσκευών σε ένα Service Oriented περιβάλλον και κλείνοντας θα αναφερθούμε στην Service Oriented Computing τεχνολογία η οποία αναφέρεται στην ανάπτυξη συστημάτων και κατ' επέκταση κατανεμημένων συστημάτων, τα οποία βασίζονται στις υπηρεσίες.

5.1 Αιτιολόγηση μελέτης των Service Oriented συστημάτων

Οι υπηρεσίες αποτελούν, κατά την άποψή μας, **τον ιδανικότερο τρόπο κάλυψης των αναγκών που εισάγει το Future Internet επιδιώκοντας την μετατροπή των πόρων σε υπηρεσίες**. Απαραίτητα χαρακτηριστικά όπως η **ενοποίηση ετερογενών τεχνολογιών σε ενιαία συστήματα**, μπορούν να αποδοθούν με ακρίβεια σ' ένα σύστημα μέσω των αρχών της Service Oriented αρχιτεκτονικής.

Η **Service Oriented αρχιτεκτονική** περιλαμβάνει ένα σύνολο αρχών για την ανάπτυξη συστημάτων βασισμένα σε υπηρεσίες. Επιπλέον οι άμεσα συνδεόμενοι τομείς της **SODA** και της **SOC** αποτελούν ισχυρά συστατικά για την σύνθεση συστημάτων ικανών να ανταποκριθούν στις απαιτήσεις του μελλοντικού διαδικτύου, συνθέτοντας ετερογενείς τεχνολογίες. Επίσης τα **κατανεμημένα συστήματα** αποτελούν μια από τις ιδανικότερες μορφές ανάπτυξης συστημάτων τα οποία είναι ανεξάρτητα από κεντρικές οργανωτικές αρχές και μπορούν να υποστηρίξουν το χαρακτηριστικό της αυτόνομης λειτουργίας.

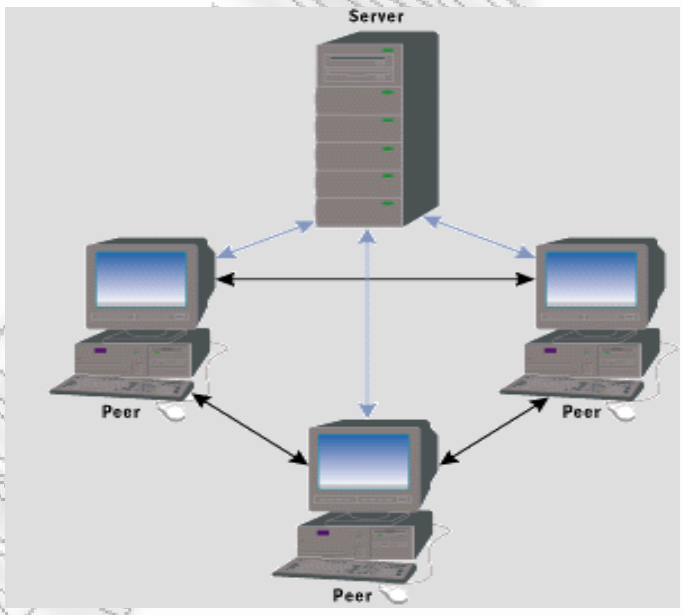
Ο λόγος λοιπόν, που η συγκεκριμένη μελέτη εστιάζει στην Service Oriented αρχιτεκτονική, στηρίζεται στην άποψη μας ότι, η SOA αποτελεί τον καλύτερο δυνατό τρόπο ανάπτυξης συστημάτων τα οποία **θα παρουσιάζουν χαρακτηριστικά**

[30] Papazoglou M.P., Georgakopoulos D., "**Service Oriented Computing**", Vol. 46, No. 10 COMMUNICATIONS OF THE ACM, Οκτώβριος 2003, σελ. 25.

που προκύπτουν από τις ποιοτικές και λειτουργικές απαιτήσεις του Future Internet.

5.2 Κατανεμημένα συστήματα

Ένα Κατανεμημένο Σύστημα αποτελείται από γεωγραφικά ανεξάρτητες, αυτόνομες υπολογιστικές συσκευές, που επικοινωνούν μεταξύ τους και λειτουργούν συντονισμένα για την επίτευξη ενός κοινού στόχου. Χαρακτηριστικά παραδείγματα κατανεμημένων συστημάτων αποτελούν το Internet , τα P2P συστήματα, τα συστήματα μεγάλων οργανισμών όπως οι τράπεζες, κλπ.[31] Σύμφωνα με τον A. Tanenbaum, *τα κατανεμημένα συστήματα απαιτούν ριζικά διαφορετικό λογισμικό από αυτό που χρησιμοποιούν τα κεντρικοποιημένα συστήματα*.[32] Αποβάλλεται η αρχιτεκτονική που βασίζεται σε κεντρικές οργανωτικές αρχές όπως για παράδειγμα το Napster, το πρώτο σύστημα Peer-to-Peer δικτύωσης[33] και υιοθετείται το μοντέλο της αποκεντρωτικής αρχιτεκτονικής όπου συγκεντρώνονται διαφορετικά υπολογιστικά συστήματα με αυτόνομη λειτουργία τα οποία συνυπάρχουν σε ένα κοινό δίκτυο μέσα από το οποίο αλληλεπιδρούν. Ένα χαρακτηριστικό παράδειγμα της αποκεντρωτικής αρχιτεκτονικής αποτελούν τα P2P δίκτυα τρίτης γενιά τα οποία εμφανίζονται το 2000 με το Freenet P2P δίκτυο.[34][35] Ένας εύκολος τρόπος για να κατανοήσουμε την διαφορετικότητα των κεντρικοποιημένων συστημάτων από τα κατανεμημένα συστήματα, είναι η μελέτη των σχημάτων 1 και 2 που παρουσιάζονται παρακάτω.



Σχήμα 1: "Κεντρικοποιημένη Αρχιτεκτονική"

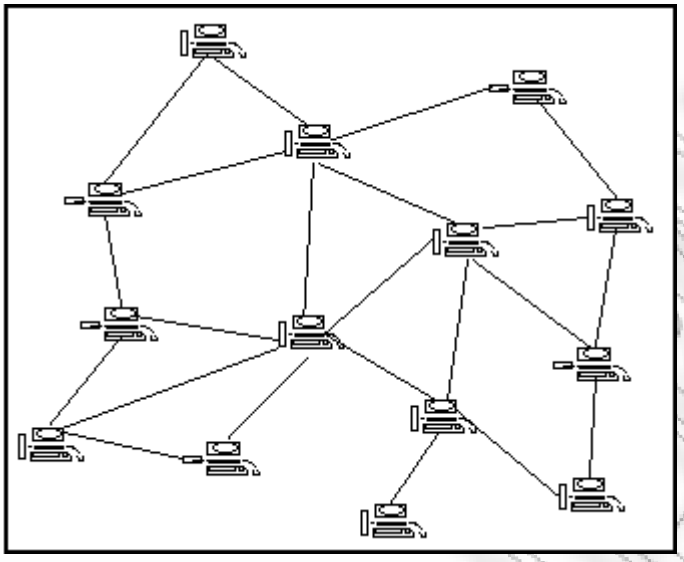
[31] **Distributed Computing**, http://en.wikipedia.org/wiki/Distributed_computing, πρόσβαση: Απρίλιος, 2010 .

[32] A. Tanenbaum, "**Distributed Systems Principles and Paradigms**", Prentice Hall, Vrije University Amsterdam, 2002, σελ. 13.

[33] **Napster**, http://en.wikipedia.org/wiki/Napster, πρόσβαση: Απρίλιος, 2010.

[34] **Freenet**, http://en.wikipedia.org/wiki/Freenet, Freenet, πρόσβαση: Απρίλιος, 2010.

[35] Diamantopoulos Fotis, Aleksandrides Kyriakos, "**Introduction to p2p Networks**", University of Macedonia, Ιανουάριος, 2003.



Σχήμα 2: "Αποκεντρωτική Αρχιτεκτονική – Κατανεμημένο Σύστημα"

Η σύγχρονη τάση που παρουσιάζεται στην πληροφοριακή τεχνολογία, μέσα από την διαδικασία της εικονικοποίησης των πόρων (Cloud Computing, Virtualization, κλπ.), οδηγεί στην ανάπτυξη συστημάτων βασισμένων στις υπηρεσίες. Σύμφωνα με την διαδικασία του Virtualization (*η οποία εξετάζεται αναλυτικά στο κεφάλαιο 9*) οι λειτουργίες των υπολογιστικών συστημάτων μπορούν να απεικονιστούν σε ένα σύστημα με την μορφή υπηρεσιών.[(36)] Εισάγοντας την παραπάνω διαδικασία στα κατανεμημένα συστήματα έχουμε την ανάπτυξη **αποκεντρικοποιημένων συστημάτων** τα οποία **βασίζουν την λειτουργία** τους στις **υπηρεσίες**. Ωστόσο η διαχείριση των υπηρεσιών που μπορούν να συγκεντρωθούν σε ένα κατανεμημένο σύστημα, αλλά και σε ένα απλό σύστημα, αποτελεί μια σημαντική διαδικασία προκειμένου να επιτευχθεί η ομαλή λειτουργία του συστήματος και η αρμονική συνύπαρξη διαφορετικών υπηρεσιών. Η Service Oriented αρχιτεκτονική μπορεί να συμβάλει στην πραγματοποίηση αυτού του στόχου, εισάγοντας μια σειρά από αρχές τις οποίες παρουσιάζουμε στην ενότητα 5.2 του παρόντος κεφαλαίου.

Η SOA βασίζει την λειτουργία της σε αρχές που διευκολύνουν την ρύθμιση και την εκτέλεση διαφορετικών υπηρεσιών με σκοπό την εξυπηρέτηση των διαδικασιών που μπορεί να προκύψουν σε ένα σύστημα.[(37)] Οι υπηρεσίες μπορούν να οργανώνονται με συγκεκριμένα πρότυπα, να συνθέτονται μεταξύ τους για την δημιουργία προηγμένων υπηρεσιών και μπορούν επίσης να παρέχουν δυναμικές λειτουργίες προς το σύστημα και τους επιμέρους χρήστες του. Σε ένα SOA περιβάλλον έχουμε την πλήρη διαχείριση των πόρων που διαθέτει το σύστημα, μπορούμε εύκολα να εντοπίσουμε τα σφάλματα που προκύπτουν καθώς και να τα επιδιορθώσουμε, ενώ παράλληλα μπορούμε με ευκολία να διαχειριστούμε την

[36] **Virtualization**, http://en.wikipedia.org/wiki/Virtualization, πρόσβαση: Μάρτιος, 2010.

[37] J. Block, "**Service-Oriented Architecture The importance and Relevance of SOA in the Financial Enterprise**", Diamond Global Advanced Technology, Ιανουάριος, 2007, σελ. 3.

προσθήκη ή την αφαίρεση – αλλαγή στοιχείων που περιλαμβάνει το σύστημα. Έτσι σε ένα **κατανεμημένο σύστημα που βασίζει την λειτουργία του στις υπηρεσίες**, είναι **σημαντική η εισαγωγή της SOA προκειμένου** το σύστημα **να οργανωθεί** και **να λειτουργήσει με τον βέλτιστο δυνατό τρόπο**.

5.3 Service Oriented Architecture (SOA)

Η SOA αποτελεί ένα αρχιτεκτονικό στυλ για την «οικοδόμηση» συστημάτων βασισμένα στις υπηρεσίες. Επιχειρεί την διαχείριση διαφορετικών υπηρεσιών οι οποίες μπορούν να συνεργαστούν για την εκτέλεση πολύπλοκων διαδικασιών.[(38)(39)] Η δημιουργία μιας υπηρεσίας και η παροχή της δεν αποτελεί δύσκολη διαδικασία. Η δυσκολία εντοπίζεται στην συνεργασία της υπηρεσίας αυτή με άλλες υπηρεσίες εντός ή εκτός των ορίων του οργανισμού που την δημιούργησε. Το ιδανικό θα ήταν όλες οι υπηρεσίες που δημιουργούνται να ανήκουν σε έναν κοινό οργανισμό και να διαχειρίζονται αποκλειστικά και μόνο από αυτόν. Ωστόσο λόγο του τεράστιου όγκου πληροφοριών κάτι τέτοιο δεν μπορεί να γίνει. Συνεπώς απαιτείται μια αρχή η οποία θα οργανώνει και θα διαχειρίζεται την επικοινωνία και την αλληλεπίδραση μεταξύ διαφορετικών υπηρεσιών. Αυτό ακριβώς πραγματοποιεί η SOA. Δίνοντας ένα παράδειγμα για την κατανόηση των παραπάνω μπορούμε να αναφέρουμε το εξής.[(40)] Εάν υποθέσουμε ότι έχουμε ένα CD μουσικής και θέλουμε να το ακούσουμε, θα το τοποθετήσουμε στο CD Player το οποίο θα το αναπαράγει. Στην περίπτωση αυτή το CD Player, μας παρέχει την υπηρεσία αναπαραγωγής της μουσικής. Την ίδια υπηρεσία μπορεί να μας την παρέχει και ένα φορητό CD Player ή ένα ακριβό στερεοφωνικό. Οι τρεις αυτές συσκευές μας παρέχουν την ίδια υπηρεσία αλλά με διαφορετική ποιότητα.

Εάν η διαδικασία αυτή βασιζόταν στις αρχές του αντικειμενοστραφούς προγραμματισμού τότε κάθε CD θα έπρεπε να έχει το δικό του CD Player και δεν θα μπορούσε να αναπαραχθεί σε άλλη συσκευή. Αυτό πραγματικά θεωρείται ασυνήθιστο αλλά παρόλ' αυτά αποτελεί τον βασικότερο, ίσως, τρόπο ανάπτυξης λογισμικού στις μέρες μας. Αυτό έχει ως αποτέλεσμα να δημιουργούνται υπηρεσίες οι οποίες μπορούν να χρησιμοποιηθούν μόνο από συγκεκριμένα συστήματα, αποκλείοντας έτσι άλλες ετερογενείς τεχνολογίες.

Η αντιμετώπιση της δυσκολίας που προκύπτει από τους περιορισμούς της αντικειμενοστραφούς τεχνολογίας έρχεται μέσα από την δημιουργία συστημάτων τα οποία διαχωρίζουν τον χρήστη από τις υπηρεσίες. Έτσι όταν ένα χρήστης επιθυμεί να χρησιμοποιήσει μια υπηρεσία θα απευθυνθεί στον πάροχο της ο οποίος την εκτελεί για λογαριασμό του χρήστη. Συμφώνα με το πρότυπο αυτό αναπτύχθηκαν υπηρεσίες οι οποίες χρησιμοποιούνταν από τους χρήστες, τις συσκευές, τα δίκτυα και τις εφαρμογές. Ωστόσο η εμφάνιση του προβλήματος διαχείρισης και αλληλεπίδρασης των υπηρεσιών, έγινε πολύ γρήγορα. Για την αντιμετώπιση του προβλήματος αυτού λοιπόν, αναπτύχθηκαν νέες αρχές

---

[38] **Service Oriented Architecture**, http://en.wikipedia.org/wiki/Service-oriented_architecture, πρόσβαση Μάρτιος 2010.

[39] M. Rosen, B. Lublinsky, K.T. Smith, M.J. Balser, "**Applied SOA Service-Oriented Architecture and design strategies**", Wiley Publishing, Inc., 2008, σελ. 33-37.

[40] Hao He, "**What is Service Oriented Architecture**", XML.com, 2003 .

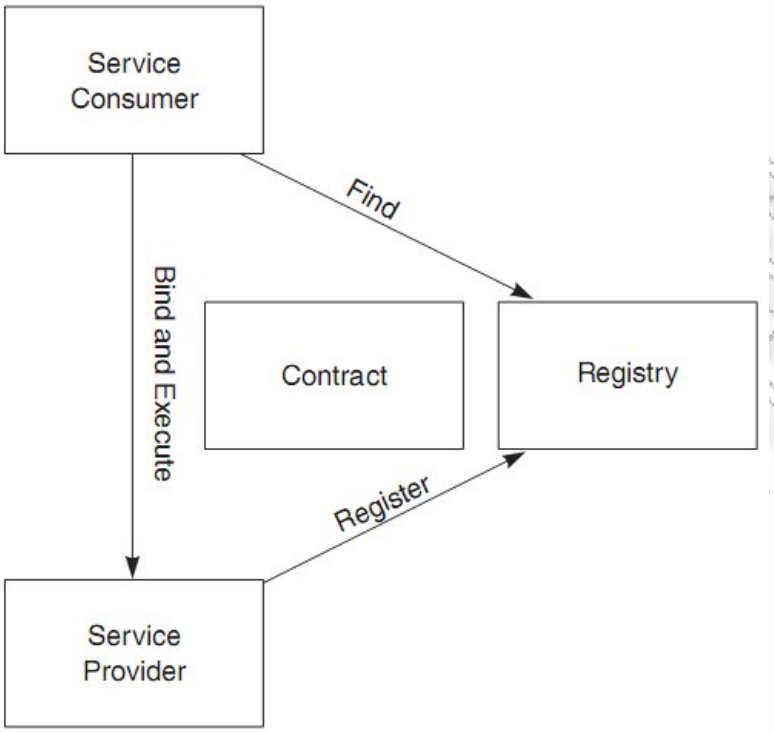
σχεδιασμού και υλοποίησης των υπηρεσιών και παρέχονται μέσω της SOA προς τους οργανισμούς.

Για να θεωρηθεί ότι ένα σύστημα υπακούει στις αρχές της SOA θα πρέπει να εφαρμόζει τέσσερις βασικούς κανόνες, οι οποίοι δίνονται στην συνέχεια.

1. Τα μηνύματα επικοινωνίας μεταξύ πελατών και παρόχων υπηρεσιών θα πρέπει να είναι περιγραφικά και όχι ενημερωτικά. Δηλαδή ένα μήνυμα πελάτη που απευθύνεται σε έναν πάροχο υπηρεσιών θα πρέπει να αναφέρει την υπηρεσία που επιθυμεί να χρησιμοποιήσει και όχι το πώς θα του παρέχει την υπηρεσία ο φορέας. Αυτό μπορεί να γίνει εύκολα αντιληπτό αν παραθέσουμε το παράδειγμα του εστιατορίου. Ως πελάτες μπορούμε να παραγγείλουμε στον σερβιτόρο αυτό που επιθυμούμε να φάμε. Δεν μπορούμε όμως να υποδείξουμε στον σεφ πως θα το μαγειρέψει για να το φάμε.
2. Στην ανταλλαγή μηνυμάτων μεταξύ πελάτη και φορέα υπηρεσιών θα πρέπει να υπάρχει ένα συγκεκριμένο πρότυπο επικοινωνίας συμφώνα με το οποίο θα διαμορφώνεται η δομή των μηνυμάτων. Έτσι μπορεί να καταστεί αποτελεσματική και ασφαλής η επικοινωνία μεταξύ πελάτη και παρόχου υπηρεσιών.
3. Η επεκτασιμότητα της αρχιτεκτονικής αποτελεί βασική προϋπόθεση για την δημιουργία ενός αποτελεσματικού συστήματος βασιζόμενο στην Service Oriented αρχιτεκτονική. Η γρήγορη μεταβολή των συνθηκών σε ένα σύστημα αποτελεί συχνό φαινόμενο το οποίο αναγκάζει το ίδιο το σύστημα, τους χρήστες του και το λογισμικό του να αλλάξουν «συμπεριφορά» και να τροποποιηθούν. Έτσι είναι απαραίτητο οι εφαρμογές να δομούνται με τέτοιο τρόπο ώστε να επιτρέπεται η μελλοντική τους επέκταση η οποία αφενός θα τα βελτιώσει και αφετέρου θα τους δώσει την δυνατότητα να ανταπεξέλθουν στις ανάγκες του συστήματος.
4. Η SOA πρέπει να παρέχει ένα μηχανισμό ο οποίος επιτρέπει στους χρήστες του συστήματος να δημοσιεύσουν μια υπηρεσία ή να ανακαλύψουν μια υπηρεσία κάνοντας αναζήτηση. Ο μηχανισμός αυτός μπορεί να είναι δυναμικά επεκτάσιμος και να υλοποιείται με την καταγραφή των υπηρεσιών σε ειδικούς καταλόγους ή βάσεις δεδομένων.

Ένα Service Oriented σύστημα αποτελείται από διαφορετικές οντότητες οι οποίες συνεργάζονται μεταξύ τους ώστε να φέρουν εις πέρας συγκεκριμένες διεργασίες.[(41)] Η εικόνα 4 παρουσιάζει την συνεργασία μεταξύ των οντοτήτων ενός SO συστήματος.

[41] James McGovern, Sameer Tyagi, Michael Stevens, Sunil Mathew, "**Service Oriented Architecture**", Chapter 2, 2003, σελ. 37 – 40.



Εικόνα 4:"Οντότητες SOA"[42]

Στη συνέχεια παραθέτουμε αναλυτικά την κάθε οντότητα που παρουσιάζεται στο παραπάνω σχήμα.

5.3.1 Καταναλωτής υπηρεσίας – (Service Consumer)

Ο καταναλωτής υπηρεσίας μπορεί να είναι μια εφαρμογή, μια υπηρεσία ή οποιαδήποτε μορφή λογισμικού που χρειάζεται μια υπηρεσία. Είναι η οντότητα που εκκινεί μια υπηρεσία από εκεί που είναι καταχωρημένη, δεσμεύει την υπηρεσία εγκαθιστώντας μια σύνδεση μεταξύ τους και εκτελεί τις λειτουργίες που παρέχει η υπηρεσία. Ο καταναλωτής της υπηρεσίας μπορεί να εκτελέσει μια υπηρεσία μέσω της αποστολής ενός αιτήματος το οποίο διαμορφώνεται σύμφωνα με κάποια πρότυπα επικοινωνίας που ισχύουν στο σύστημα.

5.3.2 Φορέας υπηρεσία – (Service Provider)

Ο φορέας υπηρεσίας είναι μια οντότητα η οποία είναι εξουσιοδοτημένη από το δίκτυο να εκτελεί αιτήματα των καταναλωτών υπηρεσίας. Μπορεί να είναι ένας ισχυρός υπολογιστής, μια απλή συσκευή ή οποιοσδήποτε τύπος λογισμικού συστήματος ο οποίος μπορεί να εκτελέσει αιτήματα υπηρεσίας. Ο service provider μπορεί να δημοσιεύσει τις παρεχόμενες υπηρεσίες του στην Service Registry προκειμένου αυτές να είναι ορατές στους Service Consumers.

[42] βλ. παρ.38, **σελ.** 19.

5.3.3 Service Registry

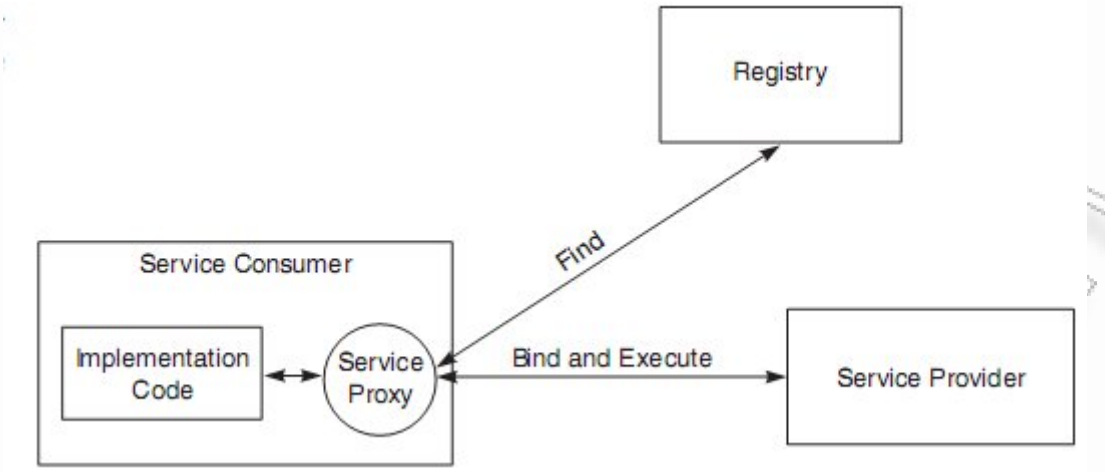
Η Service Registry οντότητα είναι αρχείο δικτύου το οποίο περιέχει τις διαθέσιμες υπηρεσίες από τους παρόχους. Αποθηκεύει στοιχεία που αφορούν τις πληροφορίες των παρεχόμενων υπηρεσιών από τους φορείς, ενώ παράλληλα διαθέτει τις πληροφορίες αυτές στους καταναλωτές υπηρεσιών που πραγματοποιούν αναζήτηση υπηρεσιών.

5.3.4 Service Contract

Το «συμβόλαιο» υπηρεσίας είναι ένα χαρακτηριστικό το οποίο καθορίζει την διαδικασία αλληλεπίδρασης παρόχου – καταναλωτή υπηρεσίας. Περιγράφει την μορφή που πρέπει να έχουν τα αιτήματα και οι απαντήσεις προς και από μια υπηρεσία. Το service contract καθορίζει τις προϋποθέσεις και τις συνθήκες σύμφωνα με τις οποίες μπορεί να εκτελεστεί μια υπηρεσία. Επιπλέον μπορεί να ρυθμίζει θέματα που αφορούν την ποιότητα υπηρεσιών (QoS). Θα μπορούσαμε να πούμε ότι το Service Contract εισάγει χαρακτηριστικά «έξυπνης» διαχείρισης της διαδικασίας εκτέλεσης υπηρεσιών.

5.3.5 Service Proxy

Κάθε service proxy αποτελεί μέρος του Service Consumer ο οποίος κάθε φορά που επιθυμεί να καλέσει μια υπηρεσία εκτελεί ένα αίτημα στην οντότητα του Service proxy και αυτή με την σειρά της ελέγχει την service registry αναζητώντας μια συμβατή υπηρεσία με αυτό που ζητάει ο Service Consumer. Ο Service Proxy μπορεί επίσης να κρατά στην μνήμη στοιχεία που αφορούν την σύνδεση μεταξύ consumer – provider ώστε να μην χρειάζεται κάθε φορά να δεσμεύει πόρους δικτύου στην προσπάθεια του να κάνει εκ νέου αναζήτηση του provider. Επιπλέον σε περιπτώσεις όπου κάποιες υπηρεσίες απαιτούν ελάχιστη παροχή δεδομένων και μικρή επεξεργαστική ισχύ για την εκτέλεση τους, μπορούν να εκτελεστούν εξολοκλήρου τοπικά στον service proxy. Ωστόσο η εκτέλεση των υπηρεσιών τοπικά σε έναν service proxy, απαιτεί να υπάρχει συμβατότητα του proxy με την τεχνολογία της υπηρεσίας. Η οντότητα του service proxy αποτελεί στην πραγματικότητα τον «ενδιάμεσο» στη διαδικασία επικοινωνίας καταναλωτή – παρόχου. Η εικόνα 5 απεικονίζει την οντότητα service proxy.



Εικόνα 5:"SOA - οντότητα Service Proxy"[43]

5.3.6 Service Lease

Η υπηρεσία του lease, είναι εκείνη που σε ένα service oriented σύστημα, διαχειρίζεται την διάρκεια των συνδέσεων μεταξύ καταναλωτών και παρόχων. Ουσιαστικά καθορίζει το χρονικό περιθώριο χρήσης μιας υπηρεσίας. Η υπηρεσία του lease είναι απαραίτητη διότι μπορεί να διαχειριστεί καταστάσεις όπου ένας καταναλωτής θα δέσμευε μια υπηρεσία σε έναν πάροχο για μεγάλο χρονικό διάστημα. Αυτό θα μπορούσε να είναι σοβαρό πρόβλημα για τον πάροχο μιας και θα γινόταν παρατεταμένη χρήση των πόρων με αποτέλεσμα τον αποκλεισμό νέων αιτήσεων από άλλους καταναλωτές που χρειάζονται την υπηρεσία. Αναλύοντας απλά την συγκεκριμένη υπηρεσία θα μπορούσαμε να πούμε ότι διαχειρίζεται θέματα ορθής χρήσης του δικτύου και των πόρων του.

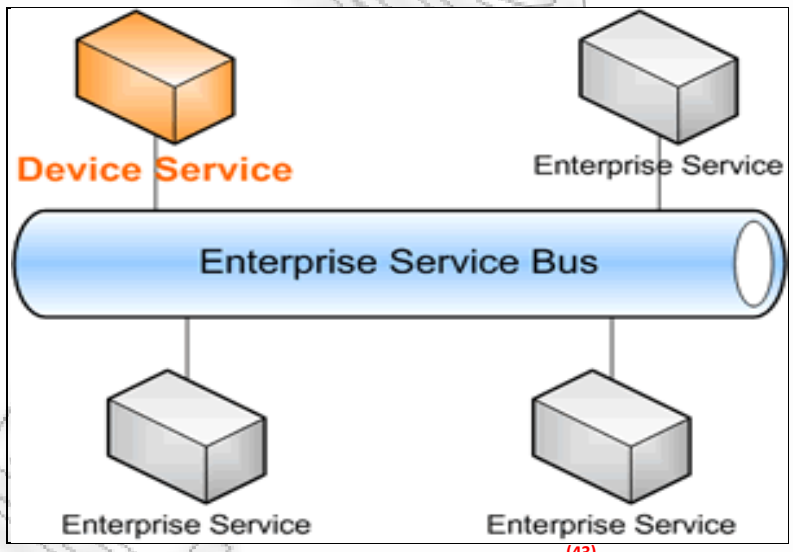
Λαμβάνοντας υπόψη τα παραπάνω, θα μπορούσαμε να πούμε ότι έχουμε μια συγκροτημένη εικόνα για την Service Oriented αρχιτεκτονική. Γνωρίζοντας τις αρχές που την διέπουν και τις υπηρεσίες – οντότητες που αποτελούν τον πυρήνα της, μπορούμε να αναφέρουμε ότι η SOA περιλαμβάνει τη διαδικασία καταγραφής της αλληλεπίδρασης των υπηρεσιών που συγκεντρώνει ένα σύστημα. Η διαδικασία αυτή μπορεί να εφαρμοστεί σε δύο φάσεις: στην **ανάπτυξη του συστήματος** και στην **ενσωμάτωση του συστήματος.** Έτσι αρχικά δημιουργείται ένα σχέδιο αλληλεπίδρασης των υπηρεσιών και στην συνέχεια πραγματοποιείται αλλαγή του σχεδίου αν χρειαστεί. Κλείνοντας, σημαντικό είναι να αναφέρουμε ότι η SOA βασίζεται σε σημεία παροχής υπηρεσιών (**Service Endpoints**)[44] όπου υπάρχουν καταγεγραμμένες οι υπηρεσίες και μπορούν

[43] βλ. παρ.33, **σελ.** 17.
[44] βλ. παρ.8, **σελ.** 3.

να κληθούν από το σύστημα, (π.χ.Universal Description Discovery and Integration - UDDI).[45]

5.4 Service Oriented Device Architecture (SODA)

Η SODA αποτελεί μια μορφή αρχιτεκτονικής η οποία βασίζεται στην OSGi – **Open Services Gateway Initiative**[46] αρχιτεκτονική για την επικοινωνία των συσκευών με ένα Service Oriented σύστημα.[47] Η SODA πραγματοποιεί συνδυασμό: i) δεδομένων από το διαδίκτυο, ii) XML για την περιγραφή συσκευών και iii) URN Registry. [48] Η τωρινή μορφή της SODA συμπεριλαμβάνει έναν Κατανεμημένο Δίαυλο Υπηρεσιών (**Enterprise Service Bus - ESB**) στον οποίο ενσωματώνονται διάφορες διαδικτυακές υπηρεσίες σύμφωνα με συγκεκριμένα πρότυπα (Web Service Standards). Οι υπηρεσίες είναι διαθέσιμες προς οποιαδήποτε συσκευή συνδεθεί στον ESB και παρουσιάζονται στην μηχανή αναζήτησης με την χρήση της XML. Έτσι με τον τρόπο αυτό μπορεί να συνδεθεί οποιαδήποτε συσκευή στον ESB και να παρέχει μια περιγραφή για τις υπηρεσίες της η οποία θα δίνεται μέσω της XML.[49] Το σχήμα 3 παρουσιάζει την τωρινή μορφή διασύνδεσης συσκευών και υπηρεσιών στην SODA.



Σχήμα 3: «SODA Enterprise Service Bus»[43]

Στόχος της SODA είναι να ενοποιήσει ένα ευρύ σύνολο συσκευών που ανήκουν στον Physical World με κατανεμημένα συστήματα του Software World, σχεδιάζοντας και δημιουργώντας ένα σύνολο υπηρεσιών. Επιπλέον η SODA επιδιώκει την αποφυγή της προβολής ενός συνόλου συσκευών που μπορούν να παρέχουν λειτουργίες μέσα από ένα σύνολο διαφορετικών δικτύων (π.χ. Internet,

[45] **Universal Description Discovery and Integration ,** http://en.wikipedia.org/wiki/UDDI, πρόσβαση Μάρτιος 2010.
[46] G.Luin Wu, C.F. Liao, Li-Chen Fu, "**Service Oriented Smart-Home Architecture Based on OSGi and Mobile-Agent Technology**", IEEE, Part C:Applications and reviews, vol. 37, No 2, Μάρτιος, 2007.
[47] βλ. παρ.14, **σελ.** 4.
[48] **Service Oriented Device Architecture,** http://www.eclipsecon.org/, πρόσβαση: Μάρτιος 2010.
[49] Βλ. παρ. 14, **σελ.** 4.

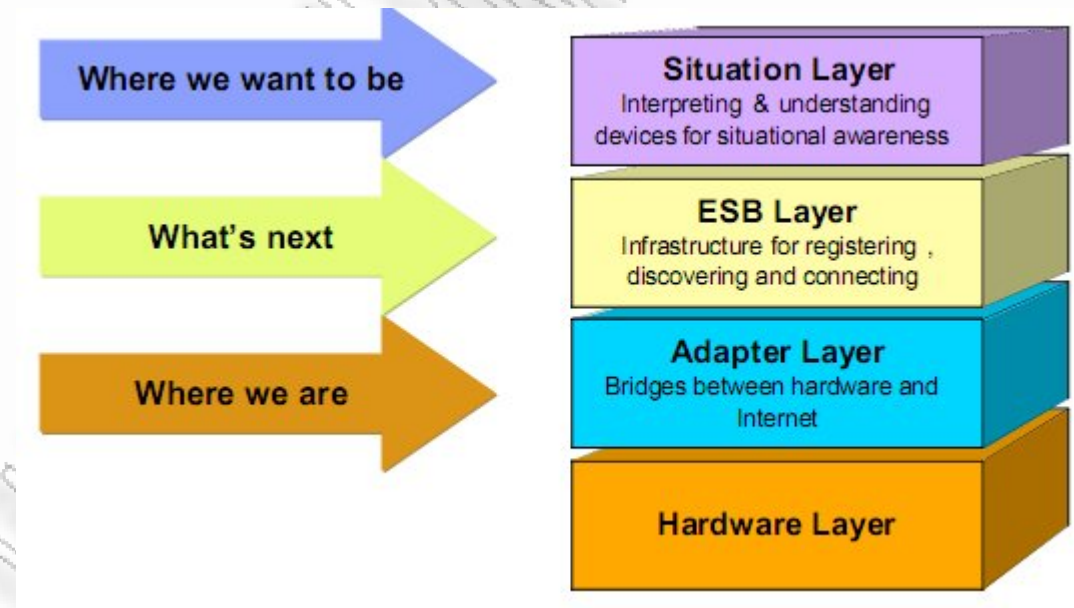
Bluetooth, κλπ.). Χωρίς αυτά τα δίκτυα κάθε μήνυμα που προκύπτει από μια συσκευή, θα υπάρχει σε ένα καλά ορισμένο σύστημα το οποίο θα είναι κοινό για όλες τις συσκευές. Έτσι η ιδέα της SODA βασίζεται στην ύπαρξη ενός καθολικού, γενικού, δικτύου το οποίο θα μπορεί να υποστηρίξει ένα σύνολο πολύπλοκων κατανεμημένων συστημάτων.

Σύμφωνα με αυτό το μοντέλο δικτύωσης, κάθε συσκευή πριν συνδεθεί στο διαδίκτυο θα πρέπει να συνδέεται με ένα καλά ορισμένο σύστημα το οποίο θα έχει σχεδιαστεί ώστε να ορίζει **διασυνδέσεις** και **αλληλεπιδράσεις** μεταξύ των συσκευών.

Η SODA υιοθετεί χαρακτηριστικά από την SOA, με τα οποία μπορεί αν διαμορφώσει ένα Service Oriented σύστημα το οποίο θα συγκεντρώνει πληροφορίες για την διεπαφή μιας συσκευής (**Device Interface**) και θα τις παρουσιάζει ως υπηρεσίες. Με τον τρόπο αυτό η SODA επιχειρεί τα εξής:

- Υψηλότερο επίπεδο αφαιρετικότητας στον Physical World
- Διαχωρισμό της ανάπτυξης κατανεμημένων συστημάτων από τον συνεχώς αυξανόμενο ρυθμό ανάπτυξης Device Interfaces.
- Γεφύρωση φυσικών και ψηφιακών δεδομένων με την χρήση μιας ή περισσοτέρων γνωστικών υπηρεσιών.

Η SODA περιλαμβάνει τέσσερα διαφορετικά στρώματα (layers) τα οποία είναι: **Situation layer**, **ESB layer**, **Adapter layer** και **Hardware layer**.[(50)] Στην εικόνα 6 παρουσιάζεται ο τρόπος δόμησης των SODA layers.



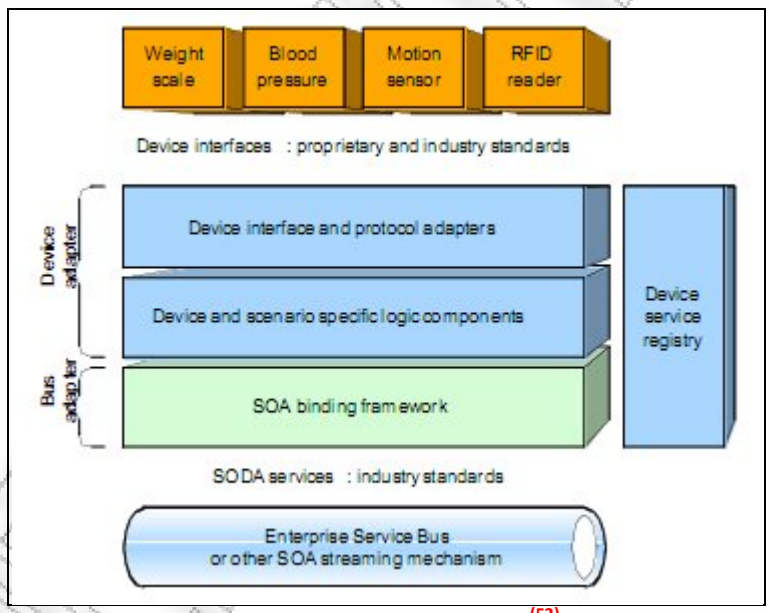
Εικόνα 6: "SODA Layers"[(41)]

[50] J. Ricker, P.A. Lyman, "**Service Oriented Device Architecture (SODA)**", EPL, v. 1.0, 6-3-2007.

Το παραπάνω μοντέλο συγκεντρώνει βασικά πλεονεκτήματα τα οποία διευκολύνουν την διαδικασία της SODA. Συγκεκριμένα:

A. Περιλαμβάνει υπηρεσίες οι οποίες διαχειρίζονται θέματα ενσωμάτωση και παροχής υπηρεσιών οι οποίες λαμβάνουν στοιχεία για: i) τα χαρακτηριστικά των συσκευών και ii) τα πρωτόκολλα επικοινωνίας
B. Περιλαμβάνει υπηρεσίες οι οποίες μπορούν να έχουν αποθηκευμένες πληροφορίες για ένα ευρύ σύνολο υλικού, λογισμικού και δικτυακών εφαρμογών τις οποίες μπορούν να τις χρησιμοποιήσουν σε διάφορες περιπτώσεις ανάλογα με τις απαιτήσεις.[51]

Το σημαντικότερο επίπεδο λειτουργιών στο μοντέλο της εικόνας 6 είναι το Adapter Layer, στο οποίο πραγματοποιούνται όλες οι απαιτούμενες διαδικασίες για την διασύνδεση των συσκευών με το διαδίκτυο μέσω της SODA. Η συνολική διαδικασία αυτού οργανώνεται σε τρεις φάσεις οι οποίες απεικονίζονται στο σχήμα 4 και περιγράφονται αμέσως μετά.



Σχήμα 4: "SODA Adapter Layer"[52]

Device Adapter: πληροφορίες για την διεπαφή της συσκευής, τα πρωτόκολλα δικτύωσης και τις συνδέσεις. Μετατροπή των πληροφοριών σε ένα μοντέλο υπηρεσιών που παρέχονται από την συσκευή.

Bus Adapter: μεταφορά δεδομένων από την συσκευή πάνω από ένα κοινό πρωτόκολλο δικτύωσης της SODA και οργάνωση των δεδομένων σε υπηρεσίες σύμφωνα με το μοντέλο της SOA.

Device Service Registry: δημοσίευση και ανακάλυψη υπηρεσιών. Πρόσβαση των συσκευών σε αυτές μέσω κοινού δικτύου της SODA.[53] Η λειτουργία της Device Service Registry ενσωματώνει το χαρακτηριστικό **Uniform Resource Identifier - URI**

[51] βλ. παρ.14, **σελ.** 4.
[52] Βλ. παρ. 42, **σελ.** 22.
[53] βλ. παρ.14, **σελ.** 4.

σύμφωνα με το οποίο δημιουργείται ένα σύστημα ταυτοποίησης των στοιχείων (υλικό και λογισμικό) που συνδέονται με το διαδίκτυο. Με τον τρόπο αυτό επιτυγχάνεται ο εντοπισμός ενός πόρου του διαδικτύου και η εγκατάσταση επικοινωνίας με αυτόν χρησιμοποιώντας συγκεκριμένα πρωτόκολλα.(54)

Οι υπηρεσίες που διαμορφώνονται από τις πληροφορίες και τα δεδομένα που απορρέουν από το Adapter Layer, είναι διαθέσιμες προς τους χρήστες (συσκευές, δίκτυα, κλπ.) μέσω του Enterprise Service Bus. Ο ESB αναλαμβάνει την απόκρυψη της πολυπλοκότητας των τεχνολογιών που σχετίζονται με το σύστημα της SODA. Τον καταναλωτή μιας υπηρεσίας δεν θα πρέπει να τον ενδιαφέρουν θέματα όπως η γλώσσα προγραμματισμού, το περιβάλλον χρόνου εκτέλεσης (run-time environment), η πλατφόρμα υλικού, η διεύθυνση δικτύου, ή η τρέχουσα διαθεσιμότητα της εφαρμογής. Εκείνο που έχει να κάνει είναι να διαθέσει τις πληροφορίες της διεπαφής της συσκευής του και ο ESB αναλαμβάνει να διαθέσει τις πληροφορίες αυτές με την μορφή υπηρεσιών που μπορούν να διανεμηθούν στο δίκτυο της SODA πάνω από ένα κοινό πρωτόκολλο.(55) Ο ESB επιπλέον εφαρμόζει πρότυπα από την SOA βάση των οποίον οργανώνει και διαμοιράζει τις υπηρεσίες προς τους καταναλωτές τους.(56) Ουσιαστικά η SODA βασίζει την service-oriented φύση της στον τρόπο με τον οποίο οργανώνεται ο ESB.

Η SODA λοιπόν, μέσα από τον συνδυασμό στοιχείων από τις αρχιτεκτονικές OSGi και SOA επιχειρεί να δημιουργήσει μια ενιαία «γέφυρα επικοινωνίας» μεταξύ Physical World – Software World ενοποιώντας ετερογενείς τεχνολογίες υλικού και δικτύωσης μέσα από ένα διαλειτουργικό σύστημα παροχής υπηρεσιών.

5.5 Η τεχνολογία Service Oriented Computing

Η Service Oriented Computing – SOC τεχνολογία ορίζει έναν νέο τρόπο ανάπτυξης συστημάτων βασισμένων στις υπηρεσίες τα οποία εστιάζουν στην αυτονομία και την ετερογένεια των πόρων που θα συγκεντρώνουν.(57) Οι πόροι παρουσιάζονται ως υπηρεσίες που οργανώνονται σύμφωνα με την SOA και μπορούν να συνδυαστούν με την αναζήτηση και κλίση διαθέσιμων υπηρεσιών από το δίκτυο για την εκτέλεση συγκεκριμένων εργασιών.

Οι διαδικτυακές υπηρεσίες (**Web Services**) αποτελούν στις μέρες μας, ίσως, την πιο οικία μορφή εφαρμογής της SOC τεχνολογίας. Οι Web Services χρησιμοποιούν το διαδίκτυο ως μέσο επικοινωνίας και επεκτείνουν τα χαρακτηριστικά τους εισάγοντας το Simple Object Access Protocol (**SOAP**) για την μεταφορά δεδομένων, την Web Services Description Language (**WSDL**) για την περιγραφή των υπηρεσιών και την Business Process Execution Language for Web Services (**BPEL4WS**) για την οργάνωση των υπηρεσιών. Βασική επιδίωξη της SOC

[54] **Uniform Resource Identifier**, http://en.wikipedia.org/wiki/Uniform_Resource_Identifier, πρόσβαση: Μάρτιος, 2010.
[55] M.T. Schmidt, B. Hutchison, P. Lambros, R. Phippen, "**The Enterprise Service Bus: Making service oriented architecture real**", IBM SYSTEMS JOURNAL, VOL. 44, NO 4, 2005, σελ. 3.
[56] **Enterprise Service Bus**, http://en.wikipedia.org/wiki/Enterprise_service_bus, πρόσβαση: Μάρτιος, 2010.
[57] M. Huhns, M. P. Singh, "**Service Oriented Computing: Key Concepts and Principles**", IEEE Computer Society, Φεβρουάριος, 2005.

αποτελεί η συγκέντρωση διαφορετικών εφαρμογών σε ένα ανεξάρτητο κοινό δίκτυο υπηρεσιών. Επιπλέον οι υπηρεσίες θα μπορούν να συνδυαστούν για την εκτέλεση σύνθετων διαδικασιών που προέρχονται από τα συστήματα και τους χρήστες τους, ενώ οι υπαρκτές υπηρεσίες θα μπορούν να ενσωματωθούν σε νέες επεκτείνοντας τις λειτουργικές τους δυνατότητες.

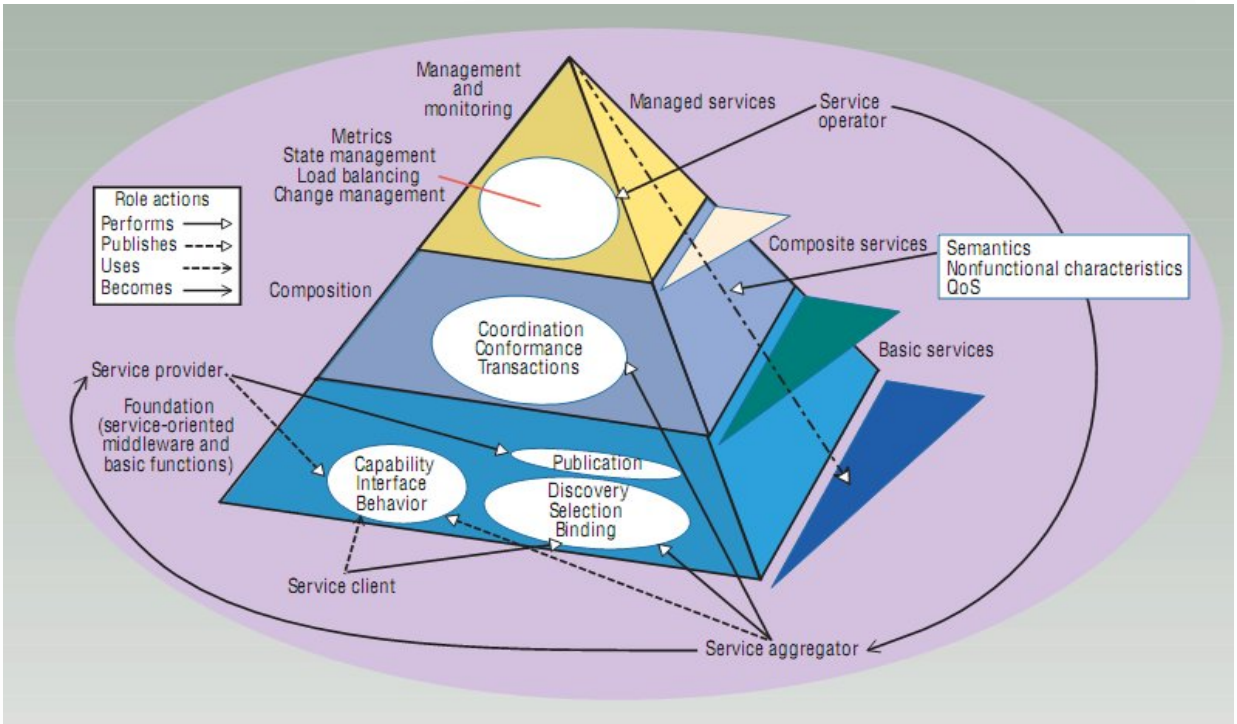
Η SOA αποτελεί το «κλειδί» για την επίτευξη των στόχων της SOC τεχνολογίας. Η SOA αποτελεί τον λογικό τρόπο για να σχεδιάσουμε ένα σύστημα λογισμικού το οποίο θα παρέχει υπηρεσίες σε άλλες εφαρμογές ή δίκτυα με την χρήση διεπαφών δημοσίευσης και ανακάλυψης αυτών των υπηρεσιών. Ωστόσο δεν επαρκεί ως μοναδικό χαρακτηριστικό για την SOC. Η SOC λοιπόν επιχειρεί να συνθέσει ένα περίπλοκο σύστημα συνεργασίας περιλαμβάνοντας εκτός από την SOA, χαρακτηριστικά όπως: distributed computing systems, computer architectures και middleware, συστήματα βάσεων δεδομένων, μηχανισμούς ασφάλειας και συστήματα διαχείρισης της γνώσης (knowledge management systems). Συνενώνοντας τεχνολογικά χαρακτηριστικά όπως τα παραπάνω, σε ένα σύστημα βασισμένο στην SOC τεχνολογία, επιτυγχάνουμε την αντιμετώπιση μελλοντικών σύνθετων αναγκών που προκύπτουν από διαφορετικά συστήματα και τους χρήστες τους.

Η SOC τεχνολογία εξελίσσεται σε έναν κύκλο ενεργειών ο οποίος περιλαμβάνει τρείς φάσεις. Η κάθε φάση εκτελεί και μια διαφορετική διαδικασία παρέχοντας δεδομένα που αφορούν τις υπηρεσίες στην επόμενη φάση. Οι φάσεις αυτές είναι οι εξής:

- **Service foundation**
- **Service composition**
- **Service management and monitoring**

Η εικόνα 7 απεικονίζει τις τρεις φάσεις που συνθέτουν την SOC τεχνολογία. Στο σημείο αυτό σημαντικό είναι να αναφέρουμε ότι κάθε μία από τις φάσεις του SOC αποτελεί και έναν ξεχωριστό τομέα έρευνας. Στην βιβλιογραφία το σχήμα που απεικονίζεται στην εικόνα 7 αναφέρεται ως **SOC research road map**.[58]

[58] M. Papazoglou, P. Traverso, S. Dustdar, F. Leymann, "**Service-Oriented Computing: State of the art and research challenges**", IEEE Computer Society, 2007.



Εικόνα 7: "SOC research road map"

Το SOC Research road map παρουσιάζει ένα τρόπο εφαρμογής της SOA διαχωρίζοντας, την λειτουργία της σε τρεις φάσεις όπως φαίνεται και παραπάνω. Η διαστρωμάτωση αυτή βασίζεται στην **ανάγκη για διαχωρισμό** :

- Βασικών *λειτουργιών που παρέχονται από το middleware* **από** τις πιο *προηγμένες υπηρεσίες που απαιτούνται για την σύνθεση των υπηρεσιών*.
- *Business υπηρεσιών* **από** *system-centered υπηρεσίες*.
- *Σύνθεσης υπηρεσιών* **από** την *διαχείριση υπηρεσιών*.

Στο SOC research road map ορίζονται επίσης και κάποιοι ρόλοι. Ο αιτών μιας υπηρεσίας και ο πάροχος της υπηρεσίας θα πρέπει να συμφωνούν με ένα **κοινό πλαίσιο επικοινωνίας** για την **περιγραφή** των υπηρεσιών (WSDL), και την μεταξύ τους **αλληλεπίδραση**. Θα πρέπει λοιπόν να ορίζεται ένας κοινός κώδικας για την επικοινωνία των συμμετεχόντων τουλάχιστον στο επίπεδο χρήσης. Εκτός από την ορολογία επικοινωνίας παρόχου – καταναλωτή, θα πρέπει να υπάρχουν και **πλαίσια μορφοποίησης μηνυμάτων** που ανταλλάσσονται μεταξύ υπηρεσιών που αλληλεπιδρούν κατά την **εκτέλεση μιας διαδικασίας**. Επίσης στο σχέδιο της εικόνας 7, ορίζεται και η **μοντελοποίηση υπηρεσιών** και ο **service oriented μηχανισμός**, που αποτελούν σημαντικές απαιτήσεις για τις SOA εφαρμογές που εφαρμόζουν web services σε κάθε ένα από τα τρία πλαίσια.

Στην συνέχεια ακολουθεί μια σύντομη αλλά κατατοπιστική αναφορά για κάθε ένα από τα τρία πλαίσια-φάσεις λειτουργίας που περιλαμβάνει η SOC τεχνολογία.

5.5.1 Service Foundations

Το επίπεδο του Service Foundation βρίσκεται στην βάση της ιεράρχησης των επιπέδων της SOC τεχνολογίας και περιέχει την υποδομή ενός service oriented middleware η οποία δημιουργεί ένα Service oriented run time περιβάλλον. Το περιβάλλον αυτό συγκεντρώνει πάνω του ετερογενείς τεχνολογίες και προσφέρει πολλαπλά κανάλια πρόσβασης στις υπηρεσίες, πάνω από ένα σύνολο δικτύων συμπεριλαμβανομένου και του Internet. Σε ένα τυπικό σενάριο λειτουργίας ενός Service oriented συστήματος υπάρχει ένας **πάροχος υπηρεσιών** ο οποίος διαθέτει ένα λογισμικό που διαχειρίζεται την δικτυακή πρόσβαση και ορίζει τις υπηρεσίες με μια ειδική περιγραφή μέσω της οποία η κάθε υπηρεσία μπορεί να δημοσιευτεί και να ανακαλυφθεί από το σύστημα. Ένας **καταναλωτής υπηρεσιών** που αναζητά μια υπηρεσία, όταν την ανακαλύπτει λαμβάνει την ειδική περιγραφή της από την ίδια την υπηρεσία. Η ανακάλυψη υπηρεσιών μπορεί να γίνει μέσω αναζήτησης σε έναν registry directory ή σε έναν χώρο αποθήκευσης πληροφοριών όπως το UDDI.[(59)]

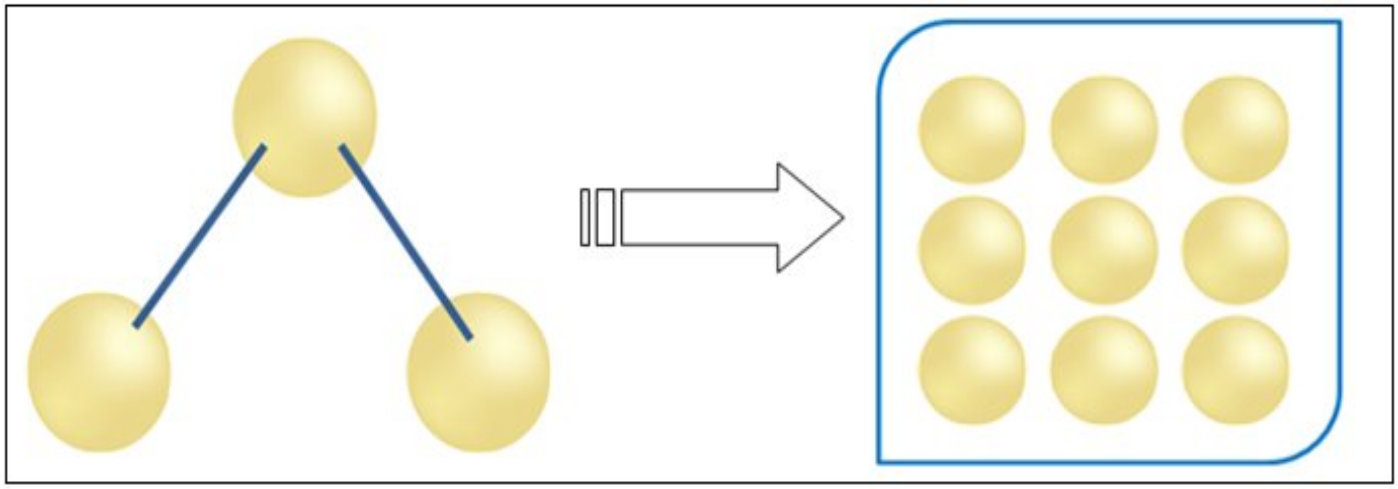
Μελετώντας προσεκτικά την λειτουργία αυτού του στρώματος θα μπορούσαμε να πούμε ότι πραγματοποιεί μια διαδικασία **παρατήρησης** των διαθέσιμων υπηρεσιών μέσα από την οποία λαμβάνει πληροφορίες για κάθε μια από τις υπηρεσίες, τις οποίες ενσωματώνει σε ειδικά μηνύματα περιγραφής για κάθε υπηρεσία. Τα μηνύματα αυτά διαμορφώνονται σύμφωνα με συγκεκριμένα πρότυπα και είναι απαραίτητα για την δημοσίευση και ανακάλυψη των υπηρεσιών, όπως ήδη προαναφέραμε. Η φάση του Foundation λαμβάνει δεδομένα από τις υπηρεσίες **service operator** και **service aggregator** τα οποία χρησιμοποιεί για να διαμορφώσει και/ή να αναδιαμορφώσει την λειτουργία των παρόχων υπηρεσιών. Τα παραδοτέα από την εκτέλεση του service foundation περνάνε στην αμέσως επόμενη φάση του service composition.

5.5.2 Service Composition

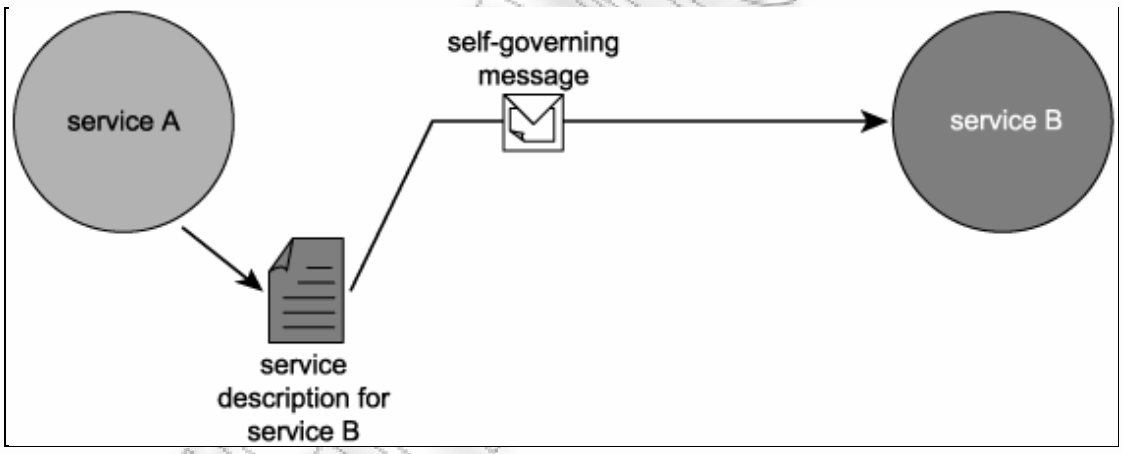
Το Service Composition αποτελεί το δεύτερο στρώμα λειτουργιών στην ιεράρχηση του SOC research road map. Συμπεριλαμβάνει διάφορους ρόλους και λειτουργίες για την «συμφωνία» των υπηρεσιών και τον συνδυασμό τους. Διαθέτει μια βασική λειτουργία η οποία **αναλύει** τις πληροφορίες για κάθε διαθέσιμη υπηρεσία και ορίζει ένα **σχέδιο συνεργασίας** των υπηρεσιών για την εκτέλεση σύνθετων λειτουργιών. Μέσα από την διαδικασία του service composition μπορούν να προκύψουν ολοκληρωμένες εφαρμογές που θα εξυπηρετούν τις ανάγκες των καταναλωτών. Στην φάση αυτή οι service aggregators λειτουργούν ως service providers δημοσιεύοντας την περιγραφή της σύνθεσης διαφορετικών υπηρεσιών. Ουσιαστικά κάθε σύνθεση υπηρεσιών αντιμετωπίζεται ως μια κοινή – ενιαία υπηρεσία η οποία μπορεί να χρησιμοποιηθεί για την εκτέλεση μιας ή περισσοτέρων λειτουργιών. Το *σχήμα 5* απεικονίζει ένα παράδειγμα σύνθετης υπηρεσίας, ενώ μια απλή απεικόνιση επικοινωνίας μεταξύ υπηρεσιών δίνετε στο *σχήμα 6*. Επιπλέον οι service aggregators αναπτύσσουν χαρακτηριστικά και/ή κώδικα ώστε να επιτραπεί η σύνθεση υπηρεσιών οι οποίες θα βασίζονται σε χαρακτηριστικά όπως: περιγραφή μετα-δεδομένων, συγκεκριμένης ονοματολογίας, μοντέλα αναφοράς και πρότυπα συμμόρφωσης. Η λειτουργία των service aggregators ολοκληρώνεται με τον

[59] M. Papazoglou, P. Traverso, S. Dustdar, F. Leymann, J. Kramer, "**Service Oriented Computing Research Roadmap**", Degstuhl Seminar, 19-4-2006.

συντονισμό και την εκτέλεση των σύνθετων υπηρεσιών, την συναλλαγή μεταξύ των υπηρεσιών και την διαχείριση αυτών μέσω ενός διαγράμματος ροής δεδομένων μεταξύ των υπηρεσιών.[60][51]



Σχήμα 5: "Διαδικασία - Service Composition"[61]



Σχήμα 6: "Συνεργασία μεταξύ δυο διαφορετικών υπηρεσιών"[62]

[60] βλ. παρ.50, **σελ.** 24.
[61] Nguyễn Huy Trường , Bùi Dũng Anh Tuấn, "**Service Oriented Computing (SOC)**", Παρουσίαση του Power Point,(ppt), Μάιος, 2008, σελ. 7,23,24.
[62] βλ. παρ.14, **σελ.** 4.

5.5.3 Service management and monitoring

Στην κορυφή της ιεραρχίας του Service Oriented Computing research road map βρίσκεται το στρώμα του service management and monitoring. Στο προηγούμενο επίπεδο οι διαδικασίες του service composition συμπεριλαμβάνουν μηχανισμούς που είναι απαραίτητοι για την ομαλή σύνθεση των υπηρεσιών, αναλύοντας την συμπεριφορά των υπηρεσιών ή των εφαρμογών και σχεδιάζοντας ένα πλάνο λειτουργίας το οποίο βασίζεται στην σύνθεσή τους. Η διαδικασία όμως του service composition δεν προωθεί κανέναν μηχανισμό διαχείρισης της αποτυχία εκτέλεσης μιας υπηρεσίας ή της αλλαγής μιας υπηρεσίας. Αυτό έχει ως αποτέλεσμα, στην περίπτωση που κάποια υπηρεσία αποτελεί τμήμα μιας πιο σύνθετης υπηρεσία αν αλλάξει ή αν αποτύχει να εκτελεστεί, να αποτύχει η λειτουργία του συστήματος μιας και ένα μέρος των διαδικασιών του θα καταστεί ανενεργό. Επίσης η περίπτωση προσθήκης νέων υπηρεσιών στο σύστημα θα πρέπει να ελέγχεται από ένα σύνολο μηχανισμών οι οποίοι θα ενσωματώσουν επιτυχώς αυτές τις υπηρεσίες στο σύστημα και θα τις συνδυάσουν με άλλες υπηρεσίες αν αυτό κριθεί απαραίτητο. Διαπιστώνουμε λοιπόν ότι απαιτούνται μηχανισμοί για την διαχείριση και τον έλεγχο των υπηρεσιών οι οποίοι θα είναι σε θέση να προβούν σε διορθωτικές αλλαγές αν αυτό είναι απαραίτητο. Έτσι μια υποδομή διαχείρισης και ελέγχου κρίνεται απαραίτητη για την ποιότητα των υπηρεσιών (**QoS**) και των εφαρμογών και στην SOC τεχνολογία παρέχεται από το Service management and monitoring πλάνο λειτουργιών. Η διαχείριση υπηρεσιών περιλαμβάνει ένα σύνολο λειτουργιών ελέγχου και διαχείρισης των εφαρμογών σε ένα SOA σύστημα μέσα από ένα κύκλο ζωής του συστήματος (**SOA Life Cycle**).

Η service management διαδικασία εκτελεί ένα σύνολο ενεργειών που ελέγχουν την κατάσταση των υπηρεσιών στο σύστημα και ρυθμίζουν την λειτουργία του. Η διαδικασία αυτή περιλαμβάνει ένα σύνολο από διαφορετικές λειτουργίες όπως: σύνθεση και διαπραγμάτευση υπηρεσιών, διαχείριση, έλεγχος, παρακολούθηση, αντιμετώπιση προβλημάτων και εντοπισμού λύσεων. Ουσιαστικά μέσω των λειτουργιών του Service management and monitoring παρέχεται η δυνατότητα δυναμικής αναδιάρθρωσης και επέκταση του συστήματος που βασίζεται στην SOA. Μελετώντας προσεκτικά τις λειτουργίες αυτού του επιπέδου θα μπορούσε να αναφερθεί ότι ο κύριος στόχος του είναι η διαδικασία του **Reconfiguration** των υπηρεσιών και κατ' επέκταση του συνολικού συστήματος.[(63)]

Η Service Oriented Computing τεχνολογία αποτελεί ένα πολύπλοκο θέμα που αφορά την σύνθεση διαφορετικών τεχνολογιών οι οποίες ενοποιούνται μέσα από ένα περίπλοκο τρόπο σύνθεσης. Καθοδηγούμενοι από τον SOC Research road map, οι προγραμματιστές μπορούν να αναπτύξουν τις απαραίτητες συνδέσεις μεταξύ ποικίλων τεχνολογιών, να διαχειριστούν θέματα που μέχρι τώρα παραλείπονταν. Επίσης μπορούν να εφαρμόσουν τις λύσεις τους στην συνεχώς αναπτυσσόμενη τεχνολογία του Future Internet. Η SOC τεχνολογία μπορεί να συμβάλει σε μεγάλο βαθμό στην προσπάθεια εναρμόνισης των ετερογενών τεχνολογιών, για την ανάπτυξη ισχυρών τεχνολογικών λύσεων.

[63] βλ. παρ. 50 **, σελ**. 25.

✓ **Κεφάλαιο 6**: «*Επισκόπηση των γνωστικών συστημάτων – Cognitive Systems*»

Η γνωστική επιστήμη (**Cognitive science**, στο εξής CS) αποτελεί έναν περίπλοκο συνδυασμό διαφορετικών πληροφοριών που οργανώνονται σύμφωνα με την λειτουργία του ανθρώπινου εγκεφάλου. Περιλαμβάνει ένα ευρύ σύνολο ευρενητικών πεδίων τα οποία αναφέρονται στα εξής: *ψυχολογία, φιλοσοφία, γλωσσολογία, νευρολογία, επιστήμες της μάθησης και την τεχνητή νοημοσύνη*. Οι αρχές που διέπουν την CS καλύπτουν πεδία από χαμηλού επιπέδου μάθησης και μηχανισμών απόφασης μέχρι και υψηλού επιπέδου λογικού σχεδιασμού που προσομοιώνουν την εγκεφαλική λειτουργία.[(64)]

Η εφαρμογή της γνωστικής επιστήμης γίνεται μέσω των γνωστικών συστημάτων (Cognitive Systems). Ένα Cognitive σύστημα μπορεί να οριστεί ως ένα σύστημα που περιέχει: λειτουργίες αντίληψης της τρέχουσας κατάστασης σε ένα περιβάλλον, λειτουργίες αναγνώρισης πιθανών προβλημάτων στο τρέχον περιβάλλον και σύμφωνα με τις συνθήκες ορίζει την αντίστοιχη συμπεριφορά του. Η ικανότητα της μάθησης μέσω της οποίας δημιουργείται η γνώση, που μπορεί να χρησιμοποιηθεί μελλοντικά για την λήψη αποφάσεων, αποτελεί ένα από τα σημαντικότερα χαρακτηριστικά των cognitive συστημάτων.[(65)]

Τα παραδοσιακά τεχνολογικά συστήματα βασίζουν την λειτουργία τους σε μια κοινωνικό-τεχνική υποδομή όπου συνδυάζονται η τεχνολογία και ο ανθρώπινος παράγοντας. Ο ανθρώπινος παράγοντας καθορίζει τι συμβαίνει σε ένα σύστημα και η τεχνολογία αναλαμβάνει να εκτελέσει ότι συμβαίνει. Στην μηχανική των cognitive συστημάτων επινοείται η λειτουργία του ανθρώπινου παράγοντα ως μηχανική λειτουργία η οποία καθορίζει το τι θα συμβεί και το πώς θα αντιμετωπιστεί.[(66)]

Στα γνωστικά συστήματα, ουσιαστικά εισάγεται το χαρακτηριστικό του **αυτοματισμού** όπου μέσα από μια σειρά ενεργειών επιτυγχάνεται η κάλυψη αναγκών που μπορεί να προκύψουν κάθε στιγμή σε ένα σύστημα. Η διαχείριση των καταστάσεων που προκύπτουν στο σύστημα επιτυγχάνεται μέσα από ένα σύνολο λειτουργιών που αναφέρονται ως **self-x λειτουργίες** και έχουν ως βάση τους το **knowledge management** και το **decision making** που αποτελούν κύρια συστατικά σύνθεσης των γνωστικών συστημάτων. Στην ενότητα που ακολουθεί θα πραγματοποιήσουμε μια σύντομη ανάλυση των τριών αυτών στοιχείων, knowledge management, Reasoning / Decision making και Self-x.

---

[64] **Cognitive Science**, http://en.wikipedia.org/wiki/Cognitive_Science, πρόσβαση: Μάρτιος, 2010.
[65] βλ. παρ. 9**, σελ**. 3.
[66] E. Hollnagel, D. D. Woods, "**Joint Cognitive Systems foundation of cognitive systems engineering**", CRC Press, 2005, σελ. 1-3.

6.1 Κύρια στοιχεία σύνθεσης Cognitive συστημάτων

Το βασικότερο χαρακτηριστικό των cognitive systems είναι η αυτοματοποίηση των διαδικασιών. Η απεξάρτηση των συστημάτων από τον ανθρώπινο παράγοντα αποτελεί ίσως το σημαντικότερο χαρακτηριστικό ενός γνωστικού συστήματος. Μελετώντας την αρχιτεκτονική τέτοιων συστημάτων μπορούμε να διαπιστώσουμε ότι συνδυάζονται πολλά συστατικά για την δόμηση τους. στην παρούσα ενότητα θα αναφερθούμε στα τρία, κατά την άποψή μας, βασικότερα συστατικά σύνθεσης ενός γνωστικού συστήματος. Αναφορικά αυτά είναι: i) Διαχείριση γνώσης - (Knowledge management), ii) Μηχανισμοί λήψης απόφασης – (Reasoning Decision Making) και iii) λειτουργίες αυτοδιαχείρισης – (Self-X functions). Στη συνέχεια ακολουθεί μια σύντομη αναφορά σε κάθε στοιχείο ξεχωριστά.

6.1.1 Διαχείριση γνώσης – (knowledge management)

Η διαχείριση γνώσης αναφέρεται σε ένα σύνολο επιχειρηματικών πρακτικών και προσεγγίσεων που αφορούν: στην δημιουργία, στην επεξεργασία και στην διάχυση γνώσης και τεχνογνωσίας. Πρακτικά θα μπορούσαμε να αναφέρουμε ότι η διαχείριση γνώσης περιλαμβάνει όλες εκείνες τις διαδικασίες που αναφέρονται στην λήψη, επεξεργασία και αναδιανομή εμπειριών σε ένα σύστημα ή ευρύτερα σε έναν οργανισμό. Γενικότερα η διαχείριση γνώσης αφορά το να κάνεις σωστά τα πράγματα και όχι τα πράγματα σωστά. [67][68]

Η τεχνολογία της πληροφορίας είναι άμεσα συνδεδεμένη με την διαχείριση γνώσης. Τα συστήματα διαχείρισης γνώσης υποστηρίζονται από τις τεχνολογίες της πληροφορίας. Συνηθέστερες μορφές είναι:

- Αποθήκες Γνώσης (Knowledge repositories).
- Εξειδικευμένα εργαλεία προσπέλασης (Expertise access tools).
- Εφαρμογές ηλεκτρονικής Μάθησης (E-learning applications).
- Τεχνολογίες Συζητήσεων και μηνυμάτων (Discussion and chat technologies).
- Σύγχρονα Διαδραστικά εργαλεία (Synchronous interactions tools).
- Εργαλεία αναζήτησης και εξόρυξης δεδομένων (Search and data mining tools).

Ανάλογα με τις τεχνολογικές ανάγκες συναντάμε και κάποια από τις παραπάνω τεχνολογίες πληροφοριών η οποία ενσωματώνεται σε ένα σύστημα ή σε κάποιον οργανισμό.

Ο στόχος της διαχείρισης γνώσης είναι κοινός σε όποια χρήση και αν τον συναντήσουμε. Στην περίπτωση που αναφερόμαστε σε ένα τεχνολογικό σύστημα το οποίο ενσωματώνει διαδικασίες της διαχείρισης γνώσης, μπορούμε να πούμε ότι ως στόχος ορίζεται η επίτευξη αύξησης της λειτουργικότητας και της αποδοτικότητας του συστήματος. Η διαχείριση

[67] Βασιλειάδης Νικόλαος, «**Διαχείριση Γνώσης**», ΑΠΘ, ΠΜΣ Πληροφορική και διοίκηση, Παρουσίαση μαθήματος, 2008, σελ. 18.

[68] **Knowledge Management**, http://en.wikipedia.org/wiki/Knowledge_management, πρόσβαση: Μάιος 2010.

γνώσης λοιπόν, στοχεύει στην βελτίωση των συστημάτων κάνοντας αποδοτικότερα.

Εισάγοντας διαδικασίες από το knowledge management σε ένα cognitive system στοχεύουμε στην αύξηση της ευφυΐας του συστήματος. Η αυτοματοποίηση των διαδικασιών σε ένα σύστημα συνεπάγεται την εξάλειψη της ανάγκης για παρέμβαση του ανθρώπινου παράγοντα στις λειτουργίες του συστήματος.

Μελετώντας ένα γνωστικό σύστημα το οποίο συνεργάζεται με συστήματα που προέρχονται από τον φυσικό κόσμο (συσκευές, δικτυακές τεχνολογίες, λογισμικό, κλπ.) θα παρατηρήσουμε ότι προκύπτει η ανάγκη για λήψη και επεξεργασία των πληροφοριών που σχετίζονται με τα συστήματα αυτά. Ένας πολύ αποδοτικός τρόπος για την αντιμετώπιση της ανάγκης αυτής μέσα από την αυτοματοποίηση των διαδικασιών, είναι να εισάγουμε στο σύστημα διαδικασίες που προέρχονται από το knowledge management. Πιο συγκεκριμένα, αν το σύστημά μας διαθέτει μηχανισμούς διαχείρισης γνώσης θα μπορεί να συγκεντρώσει πληροφορίες που αφορούν τα εξωγενή συστήματα τις οποίες στη συνέχεια μπορεί να τις επεξεργαστεί, να τις αποθηκεύσει και να τις επαναχρησιμοποιήσει. Έτσι αποδίδουμε στο σύστημά μας την ικανότητα να αυτοματοποιήσει ένα τμήμα των λειτουργιών του, ενώ παράλληλα το καθιστούμε πιο αποδοτικό αφού μπορεί να αποκτήσει πρόσβαση σε πληροφορίες που του είναι απαραίτητες γρήγορα και με ακρίβεια.

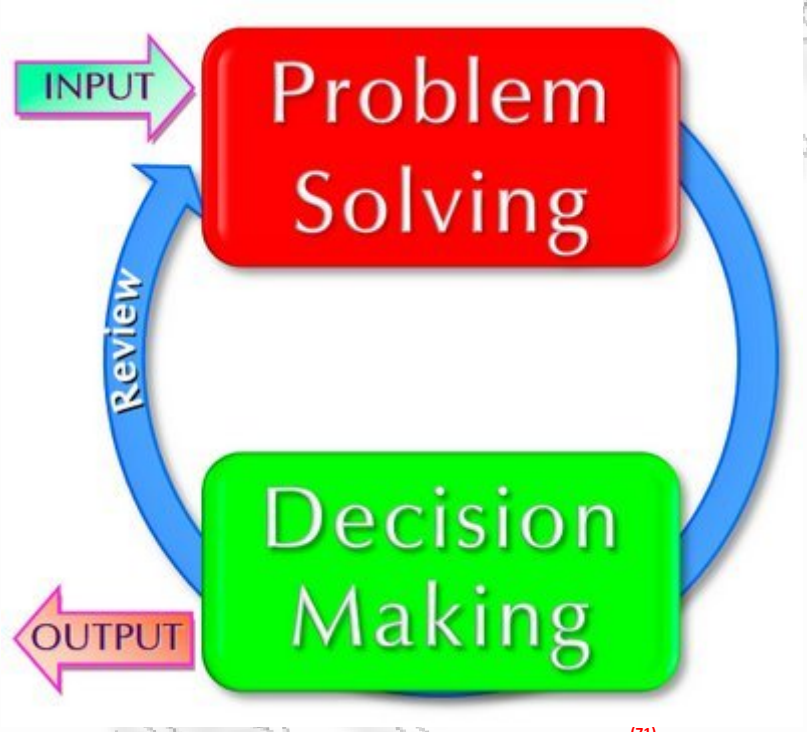
Στην προσπάθεια μας να δώσουμε μια σύντομη περιγραφή σχετικά με τις διαδικασίες του knowledge management μπορούμε να πούμε το εξής: ένα σύστημα και ειδικότερα ένα γνωστικό σύστημα, σε πρώτη φάση λαμβάνει δεδομένα. Από τα δεδομένα με κατάλληλους μηχανισμούς εξορύσσονται οι σχετικέ πληροφορίες και απεικονίζονται με την κατάλληλη μορφή (πχ: XML). Οι πληροφορίες στη συνέχεια αποθηκεύονται στην αποθήκη γνώσης του συστήματος όπου και μπορεί να πραγματοποιηθεί αναζήτηση, εύρεση και εξαγωγή πληροφοριών από τις λειτουργίες του συστήματος. Η εικόνα 8 πραγματοποιεί την απεικόνιση των όσων περιγράψαμε προηγουμένως.

Εικόνα 8:"Διαδικασία εξόρυξης γνώσης"[69]

[69] **What is knowledge management?** , http://www.nickfinck.com/presentations/bbs2005/03.html, πρόσβαση: Μάιος 2010 .

6.1.2 Μηχανισμοί λήψης απόφασης – (Reasoning / Decision making)

Η διαδικασία λήψης αποφάσεων μπορεί να θεωρηθεί ως μια γνωστική διαδικασία με αποτέλεσμα την επιλογή ενός τρόπου δράσης μεταξύ μίας ή περισσοτέρων εναλλακτικών λύσεων. Κάθε διαδικασία λήψης απόφασης, παράγει μια τελική επιλογή. Το τελικό αποτέλεσμα της διαδικασία μπορεί να αποτελεί μια ενέργεια ή την γνωμοδότηση της επιλογής που πραγματοποιήθηκε.[70] Στην εικόνα 9 παρουσιάζουμε το μοντέλο λήψης απόφασης, στην πιο απλή του μορφή.



Εικόνα 9:"Απλή μορφή μοντέλου λήψης απόφασης"[71]

Η λήψη αποφάσεων περιλαμβάνει τρία διαφορετικά στάδια τα οποία είναι: i) 1ο Στάδιο – Συγκέντρωση Πληροφορίας, ii) 2ο Στάδιο – Σχεδιασμός και iii) 3ο Στάδιο – Επιλογή. Στο πρώτο στάδιο (1^{ο} στάδιο) συναντάμε τις διαδικασίες αναγνώρισης των προβλημάτων ή ευκαιριών και συγκέντρωσης των απαραίτητων δεδομένων για το υπό εξέταση θέμα. Το δεύτερο στάδιο (2^{ο} στάδιο) περιλαμβάνει την ανάλυση των συγκεντρωμένων δεδομένων, την διεξαγωγή πληροφοριών από αυτά και την επιλογή των κριτηρίων βάση των οποίων θα πραγματοποιηθεί η λήψη της τελικής απόφασης. Τέλος στο στάδιο τρία (3^{ο} στάδιο), πραγματοποιείται η Επιλογή κάποιας λύσης. Για το σκοπό αυτό μπορεί να χρησιμοποιηθεί μια πλειάδα μεθόδων οι οποίες προέρχονται από την Επιχειρησιακή Έρευνα και την Πολυκριτήρια Ανάλυση Αποφάσεων.[72]

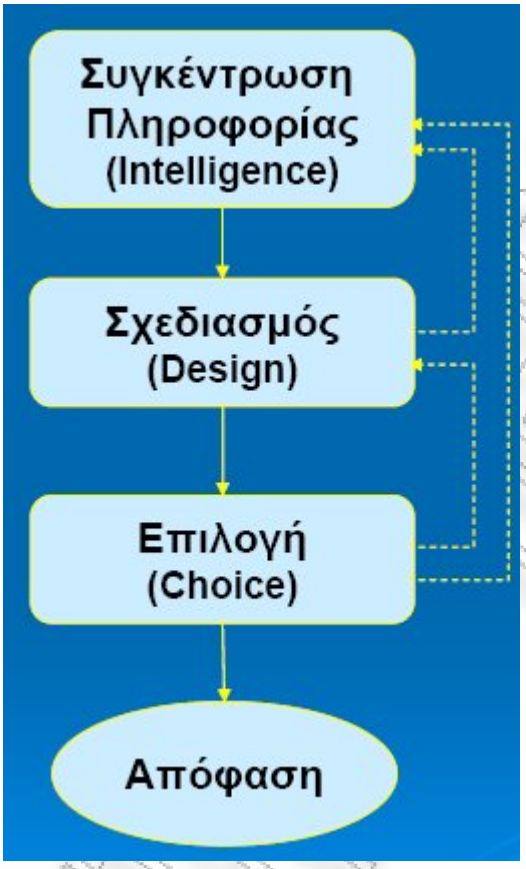
---

[70] **Cognitive Psychology and Cognitive Neuroscience/Decision Making and Reasoning, Decision Making**, http://en.wikibooks.org/wiki/Cognitive_Psychology_and_Cognitive_Neuroscience/Decision_Making_and_Reasoning, πρόσβαση: Μάιος 2010.

[71] **A Basic decision making model**, http://airline-command.blogspot.com/2009/02/basic-decision-making-model.html, πρόσβαση: Μάιος 2010.

[72] Ευαγγέλου Χ., Κακαπιλίδης Ν., «**Πολυκριτήρια Ανάλυσης και Λήψης αποφάσεων**», Πανεπιστήμιο Πατρών, Industrial Management & Information Systems Lab, σελ. 5.

Στην εικόνα 10 παρακάτω παρουσιάζουμε διαγραμματικά τα στάδια λήψης απόφασης.



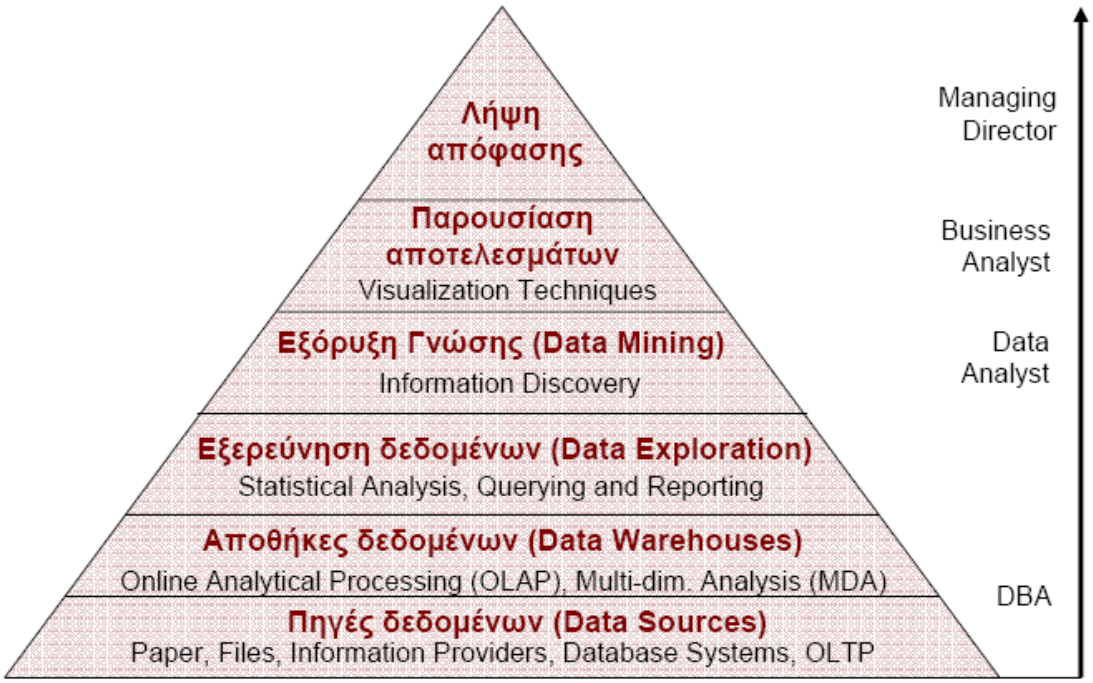
Εικόνα 10:"Στάδια λήψης απόφασης"

Κατά την διαδικασία λήψης απόφασης, μπορούμε να αντιμετωπίσουμε δύο διαφορετικές κατηγορίες καταστάσεων. Η πρώτη αναφέρεται στην διαδικασία όπου πρέπει να ληφθεί μία απόφαση για μια κατάσταση που έχουμε ξανασυναντήσει, όποτε και είναι σχετικά εύκολη διαδικασία μιας και μπορούμε να αξιοποιήσουμε έτοιμη γνώση. Η δεύτερη κατάσταση περιλαμβάνει την περίπτωση που το πρόβλημα που προέκυψε, δεν έχει παρουσιαστεί στο παρελθόν και θα πρέπει να πραγματοποιηθεί έλεγχος για την εύρεση πληροφοριών που θα μας οδηγήσουν στην δημιουργία εναλλακτικών λύσεων αντιμετώπισης του προβλήματος.

Τα γνωστικά συστήματα ενσωματώνουν την διαδικασία λήψης αποφάσεων, ως κύριο χαρακτηριστικό τους το οποίο συνδέεται άμεσα με το knowledge management. Οι πληροφορίες που συγκεντρώνονται στην αποθήκη γνώσης του συστήματος, αποτελούν την πρώτη πηγή ροής πληροφοριών για την διαδικασία του decision making στα γνωστικά συστήματα. Πιο συγκεκριμένα οι λειτουργίες που εντάσσονται στην διαδικασία λήψης απόφασης, λαμβάνουν αρχικά σαν είσοδο την αιτία ή γενικότερα το πρόβλημα που πρέπει να αντιμετωπίσουν. Στη συνέχεια πραγματοποιούν αναζήτηση, με συγκεκριμένα κριτήρια, στην βάση γνώσης του συστήματος προκειμένου να εντοπίσουν χρήσιμες πληροφορίες. Στην περίπτωση που βρεθούν πληροφορίες το σύστημα τις αναλύει και τις

χρησιμοποιεί προκειμένου να αναπτύξει εναλλακτικές λύσεις. Στο τέλος της διαδικασίας επιλέγεται η βέλτιστη λύση προς εφαρμογή. Η διαδικασία ωστόσο δεν σταματά στο σημείο αυτό μιας και πραγματοποιείται ένας συνεχής έλεγχος αποτελεσμάτων της εφαρμοσμένης λύσης προκειμένου το σύστημα να αποκτήσει νέες πληροφορίες σχετικά με μία κατάσταση και με τον τρόπο αυτό να βελτιώσει την λειτουργία του. Έτσι η «γνώση» που έχει αποκτήσει το σύστημα, αξιοποιείται από την διαδικασία λήψης απόφασης με σκοπό την βέλτιστη δυνατή λειτουργία του, αλλά παράλληλα πραγματοποιείται και η επέκτασή της από αυτή.

Στην προσπάθειά μας να δώσουμε μια βαθύτερη προσέγγιση στην διαδικασία της λήψης αποφάσεων θα μπορούσαμε να παρουσιάσουμε το μοντέλο πυραμίδας επιχειρηματικής ευφυΐας το οποίο σύμφωνα με την άποψή μας απεικονίζει σε ικανοποιητικό βαθμό την εφαρμογή της διαδικασίας και τον τρόπο λειτουργίας της σε ένα γνωστικό σύστημα. Η εικόνα 11 παρουσιάζει την πυραμίδα επιχειρηματικής ευφυΐας.



Εικόνα 11:"Πυραμίδα επιχειρηματικής ευφυΐας"[73]

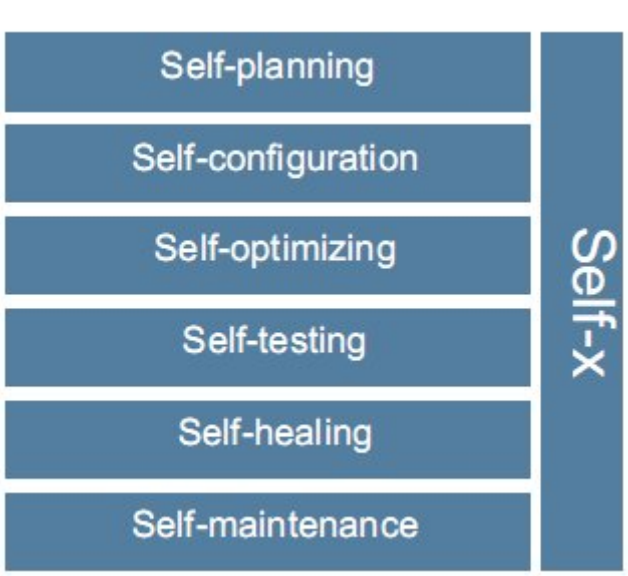
Με προσεκτική μελέτη της παραπάνω εικόνας, μπορούμε να επιβεβαιώσουμε την αναφορά σχετικά με τα τρία στάδια εξέλιξης της διαδικασία λήψης απόφασής τα οποία μάλιστα, απεικονίζονται στο διάγραμμα της εικόνας 10. Το στάδιο συγκέντρωσης της πληροφορίας, μπορούμε να πούμε ότι περιλαμβάνει τα τέσσερα πρώτα επίπεδα της πυραμίδας. Το στάδιο – ο σχεδιασμός συμπεριλαμβάνει το πέμπτο επίπεδο της πυραμίδα που αναφέρεται στην απεικόνιση των αποτελεσμάτων αναζήτησης και συγκέντρωσης της πληροφορίας. Τέλος το στάδιο τρία – η

[73] Θεωδωρίδης Γ. ,«**ΒΔ για την λήψης αποφάσεων**» , Εργαστήριο Πληροφοριακών Συστημάτων, Παν/μιο Πειραιώς, Φεβρουάριος, 2010, σελ. 2.

επιλογή, περιλαμβάνει την κορυφή της πυραμίδας όπου πλέον πρέπει να επιλέξουμε μεταξύ των λύσεων που έχουν προκύψει, αυτή που είναι περισσότερο αποτελεσματική.

6.1.3 Λειτουργίες αυτοδιαχείρισης – (Self-x functions)

Οι self-x functions αναφέρονται στο σύνολο όλων των δυνατών τεχνικών λειτουργιών που καθιστούν ένα σύστημα ικανό να διαχειρίζεται την λειτουργία του με αυτόνομο τρόπο. Στόχος των self-x λειτουργιών είναι η βελτίωση της ποιότητας και της λειτουργικότητας των συστημάτων μέσα από την αυτονόμηση της λειτουργίας τους. [(74)] Οι λειτουργίες αυτοδιαχείρισης περιλαμβάνουν έξι διαφορετικές κατηγορίες οι οποίες απεικονίζονται στην εικόνα 12.



Εικόνα 12:"Self-X Functionalities"

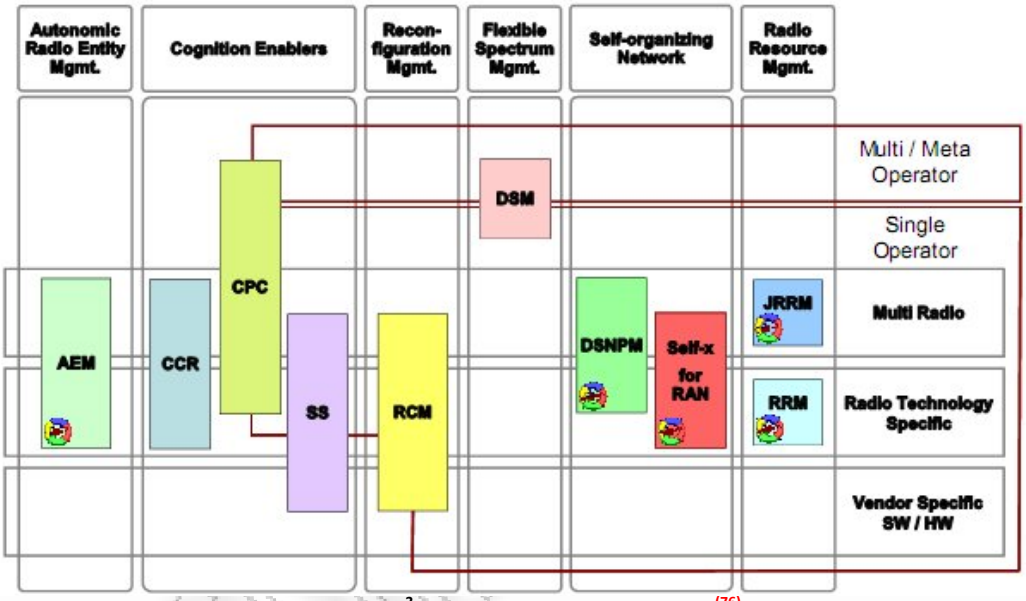
Μέσα από τις λειτουργίες αυτές και την εφαρμογή τους σε συστήματα μπορούμε να εξασφαλίσουμε μια σειρά πλεονεκτημάτων τα οποία αναφέρονται παρακάτω: [(74)(75)]

i) Ελαχιστοποίηση του κόστος, του χρόνου και της προσπάθειας για την εγκατάσταση νέων τεχνολογιών στο σύστημα μας.
ii) Μείωση του κόστους των χρησιμοποιούμενων υποδομών.
iii) Αυτόνομη Plug'n Play λειτουργία, με την εισαγωγή λειτουργιών που επιτρέπουν την εύρεση και την προσαρμογή στοιχείων από διαφορετικά περιβάλλοντα στο σύστημα.
iv) Βελτίωση της λειτουργίας του συστήματος, αύξηση των επιδόσεων του και της αξιοπιστίας του.

[74] Siebert M. , "**Self-X Control in (future) Mobile Radio Network**", Deutsche Telekom, Seventh Framework Program, σελ. 3.
[75] Alcatel Lucent, "**Self-x RAN, Autonomous self Organizing Radio Access Networks**", Bell Labs Stuttgart, Ulrich Barth, Ιούνιος, 2009, σελ. 7.

Μέσα από ένα σύνολο self-x λειτουργιών (οι οποίες στα σύγχρονα συστήματα του μελλοντικού διαδικτύου μπορούν να απεικονίζονται και ως υπηρεσίες), έχουμε την δυνατότητα ανάπτυξης συστημάτων τα οποία θα έχουν ικανότητες **αναπροσαρμογής των λειτουργιών τους**, **εγκατάστασης και επέκτασης των λειτουργιών τους με τρόπο plug'n play,** ενώ παράλληλα θα **ανεξαρτητοποιούν την λειτουργία τους από τον ανθρώπινο παράγοντα**.

Δίνοντας ένα παράδειγμα self-x λειτουργίας, σε ένα γνωστικό σύστημα, μπορούμε να αναφέρουμε το χαρακτηριστικό του self-organizing. Συγκεκριμένα λαμβάνοντας υπόψη το E^3 Cognitive Radio Architecture project, στο οποίο παρουσιάζεται η αρχιτεκτονική ενός γνωστικού συστήματος διαχείρισης ραδιοεπικοινωνιών, παρατηρούμε ότι μεταξύ των υπολοίπων λειτουργιών υπάρχει και η λειτουργία του self-organizing Network. Η εικόνα 13 παρουσιάζει το σχέδιο αρχιτεκτονική του συστήματος προκειμένου να γίνει ευκολότερα κατανοητή η αναφορά μας.



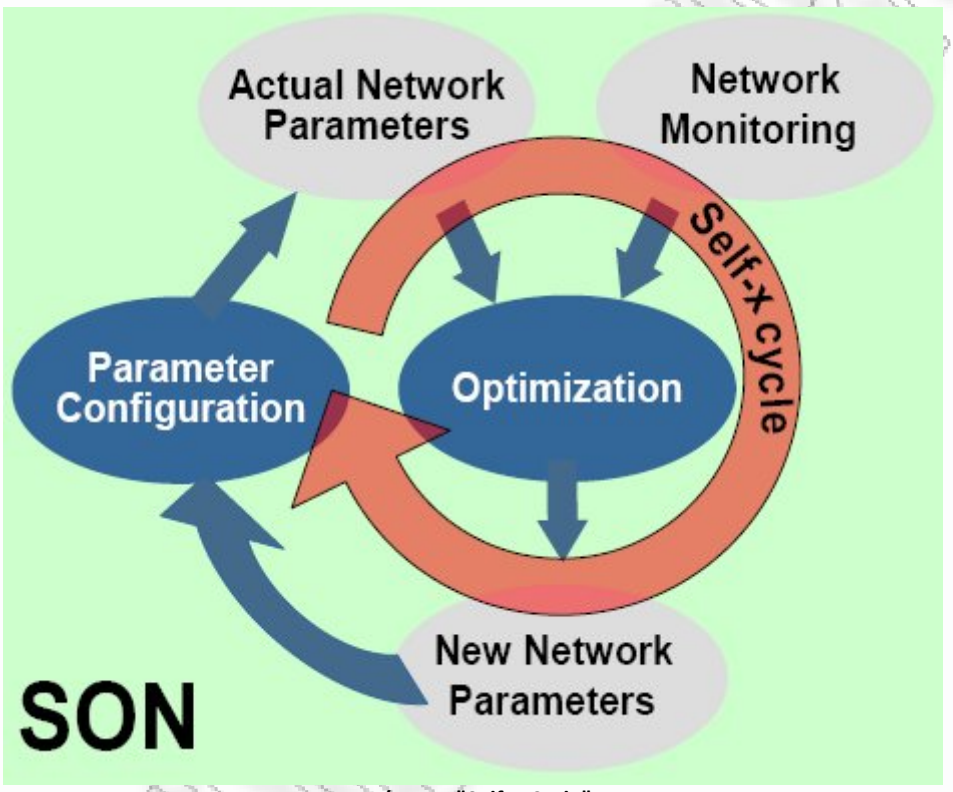
Εικόνα 13:"E^3 Cognitive Radio Architecture"[(76)]

Στο κομμάτι του self-organizing network περιλαμβάνονται δύο τμήματα, i) DSNPM (Dynamic Self-organizing Network Planning & Management) και ii) Self-X RAN (Self-X for Radio Access Network).[(76)] Η πρώτη λειτουργία είναι υπεύθυνη για την ρύθμιση και την αναδιαμόρφωση ενός τμήματος του δικτύου, σύμφωνα με τις ανάγκες που προκύπτουν κατά την λειτουργία του. Για παράδειγμα σε περίπτωση ου παρατηρηθεί απότομη αύξηση του φορτίου στο δίκτυο, η λειτουργίες του DSNPM θα πραγματοποιήσουν τις κατάλληλες ρυθμίσεις στο δίκτυο, προκειμένου να αντιμετωπίσει την συγκεκριμένη κατάσταση. Η δεύτερη λειτουργία που αναφέρεται στο Self-x RAT, είναι υπεύθυνη για την αυτό-οργάνωση των λειτουργικών χαρακτηριστικών πρόσβασης στο δίκτυο ραδιοεπικοινωνιών. Συγκεκριμένα διαχειρίζεται θέματα που αφορούν το RAT (Radio Access Technology) των στοιχείων που σχετίζονται με το δίκτυο, ενώ παράλληλα

[76] E3 Project FA/SA, "**Architecture of Cognitive Systems**", Ιανουάριος, 2010, σελ. 1.

προωθεί αποφάσεις που σχετίζονται με την εξισορρόπηση του φορτίου στο δίκτυο, βελτιστοποίηση της παραμέτρου παράδοσης μέσα στο δίκτυο, κλπ.

Οι self-x λειτουργίες βασίζονται σε έναν κύκλο εξέλιξης ο οποίος ονομάζεται **self-x cycle**[77] και ενδεικτικά παρουσιάζεται στην εικόνα 14. Η διαδικασία του κύκλο λειτουργιών αυτοδιαχείρισης βασίζεται στην συνεχή ροή δεδομένων τα οποία επεξεργάζεται και αναλόγως πραγματοποιεί την παραμετροποίηση του δικτύου βάση των βέλτιστων επιλογών. Παράλληλα στο μοντέλο του self-x cycle βασίζονται και οι διαδικασίες για την επίτευξη της ιδανικής λειτουργίας του **plug'n play των συστημάτων**.



Εικόνα 14:"Self-X Cycle"

Γενικότερα λοιπόν, συμπεραίνουμε ότι οι self-x λειτουργίες ενός γνωστικού συστήματος, στοχεύουν στην βελτίωση της λειτουργίας του, την αυτοματοποίηση των διαδικασιών του και την εισαγωγή του στοιχείου της αυτονομίας στην λειτουργία του. Επιπλέον οι λειτουργίες αυτοδιαχείρισης συμβάλουν στην αύξηση της απόδοσης του συστήματος, εισάγοντας χαρακτηριστικά όπως ο εντοπισμός και η επιδιόρθωση σφαλμάτων, η αναπροσαρμογή των λειτουργιών, κλπ. Επίσης συμβάλλουν σημαντικά στις διαδικασίες αύξησης της ποιότητας υπηρεσιών προς τους χρήστες (QoS), ενώ παράλληλα ενισχύουν την λειτουργίας τους με το χαρακτηριστικό της δυναμικής διαχείρισης προσθήκης/αφαίρεσης στοιχείων από το σύστημα.

[77] Bogenfeld Eckard, Gaspard I. ,"**Self-x in Radio Access Networks**", E3 White Paper, v. 1.0, 2008-12-22, σελ. 5.

6.2 Τομείς που επιχειρούν να καλύψουν τα Cognitive συστήματα

Ένα σύστημα μπορεί να υλοποιηθεί με την βοήθεια μια σύγχρονης τεχνολογικής πλατφόρμας. Στην cognitive επιστήμη έχουμε τις cognitive πλατφόρμες οι οποίες στοχεύουν στην δημιουργία ενός «καλά ορισμένου» και «έξυπνου» συστήματος το οποίο θα μπορεί να διαθέτει ένα αποτελεσματικό πλαίσιο εργασιών (**Framework**) για την **ενοποίηση** και **διαλειτουργικότητα** διαφορετικών **τεχνολογιών** και **schemes** μέσω διαφόρων **μεθόδων διαχείρισης** και **αλγορίθμων**. Ένα τέτοιο framework θα πρέπει να είναι σε θέση να διαχειριστεί δυναμικά τα ποικίλα θέματα που προκύπτουν στις διαδικασίες ενοποίησης και διαλειτουργικότητας τα οποία κατηγοριοποιούνται στις εξής τρείς κατηγορίες:[78]

- Αυξανόμενη πολυπλοκότητα των νέων τεχνολογιών.
- Επεκτασιμότητα των τεχνολογιών και αλλαγές στον τρόπο λειτουργίας τους.
- Μετάβαση από το πρότυπο αλληλεπίδρασης ανθρώπου – μηχανής στο πρότυπο αλληλεπίδρασης μηχανής – μηχανής.

6.2.1 Αυξανόμενη πολυπλοκότητα νέων τεχνολογιών

Η συνεχής ανάπτυξη της τεχνολογίας σε επίπεδο υλικού και λογισμικού, έχει ως αποτέλεσμα την αύξηση της πολυπλοκότητας κατά την σύνθεση ετερογενών τεχνολογιών. Η επιστήμη του ενδιάμεσου λογισμικού (middleware) στοχεύει στην εύρεση λύσεων λογισμικού για την επικοινωνία υλικού – λογισμικού, συγκεντρώνοντας διαφορετικές τεχνολογίες σε ενιαία συστήματα. Η προσπάθεια σύνθεσης ετερογενών τεχνολογιών αποσκοπεί στην ανάπτυξη ολοκληρωμένων λύσεων για την εκτέλεση σύνθετων διεργασιών που προκύπτουν σε ένα σύστημα. Η σύνθεση των ετερογενών τεχνολογιών αποτελεί μια εξαιρετικά δύσκολη και χρονοβόρα διαδικασία με υψηλή πολυπλοκότητα. Το πρόβλημα της πολυπλοκότητας αποκρύπτεται από τον χρήστη και τις συσκευές μέσω των middleware εφαρμογών. Μια cognitive πλατφόρμα σε συνδυασμό με την τεχνολογία ενδιάμεσου λογισμικού έχει την δυνατότητα να αντιμετωπίσει τα προβλήματα πολυπλοκότητας που μπορεί να προκύψουν και επιπλέον μπορεί να επιτύχει, μέσω αυτοματοποιημένων λειτουργιών, την ενοποίηση ετερογενών τεχνολογιών.

6.2.2 Επεκτασιμότητα των τεχνολογιών

Για να θεωρηθεί ένα λογισμικό επιτυχημένο θα πρέπει να συγκεντρώνει χαρακτηριστικά επεκτασιμότητας. Η συνεχής ανάπτυξη των νέων τεχνολογιών επιβάλει συχνά την αλλαγή των ήδη υπαρκτών εφαρμογών ώστε να προσαρμοστούν ομαλά στις νέες τεχνολογικές ανάγκες. Έχουν παρατηρηθεί κατά καιρούς δυσλειτουργίες σε εφαρμογές οι οποίες λόγω της αδυναμία επέκτασης τους αδυνατούσαν να ανταποκριθούν στις νέες τεχνολογικές ανάγκες. Η επέκταση των εφαρμογών λοιπόν, αποτελεί απαραίτητη προϋπόθεση για την επιτυχία τους αλλά και την βιωσιμότητα τους. Επίσης η προσθήκη του χαρακτηριστικού της αυτοματοποίησης των διαδικασιών

[78] βλ. παρ. 64 **, σελ**. 33.

στοχεύει στην ανάπτυξη συστημάτων τα οποία θα μπορούν να διαχειριστούν δυναμικά την προσθήκη και την αφαίρεση στοιχείων υλικού και λογισμικού, ενώ παράλληλα θα μπορούν να επαναπροσδιορίσουν την λειτουργία αυτών ανάλογα με τις ανάγκες τους.

6.2.3 Πρότυπο αλληλεπίδρασης μηχανής – μηχανής

Παραπάνω έχουμε αναφέρει ότι τα παραδοσιακά τεχνολογικά συστήματα βασίζουν την λειτουργία τους σε μια κοινωνικό-τεχνική υποδομή όπου συνδυάζονται η τεχνολογία και ο ανθρώπινος παράγοντας. Η νέα τεχνολογική τάση στοχεύει στην ανάπτυξη συστημάτων τα οποία θα αυτοματοποιούν την λειτουργία του ανθρώπινου παράγοντα. Πολλές είναι οι περιπτώσεις συστημάτων οπου απαιτούν την ύπαρξη ενός διαχειριστή ο οποίος είναι υπεύθυνος για την εκτέλεση ενεργειών συντήρησης και αναπροσαρμογής του συστήματος (datacenter administrators, network administrators, κλπ.). Τα cognitive συστήματα επιχειρούν την αυτοματοποίηση τέτοιων διαδικασιών, συγκεντρώνοντας διαφορετικές εφαρμογές τις οποίες απεικονίζουν ως υπηρεσίες οι οποίες είναι διαθέσιμες στους χρήστες τους. Οι χρήστες μπορεί να είναι εφαρμογές, συσκευές ή άλλες υπηρεσίες εξωγενείς ή/και του ίδιου του συστήματος.

Μέσω της cognitive τεχνολογίας λοιπόν, επιχειρούμε την δημιουργία νέων ευέλικτων συστημάτων τα οποία υλοποιούνται κατά βάση μέσω σύγχρονων πλατφορμών οι οποίες μπορούν να συγκεντρώσουν ετερογενείς τεχνολογίες, να τις ενοποιήσουν και να τις καταστήσουν πλήρως λειτουργικές και συνεργάσιμες μεταξύ τους. Οι υπηρεσίες ως απεικόνιση προϊόντων υλικού ή/και λογισμικού συμβάλουν στην διευκόλυνση της εκτέλεσης των διαδικασιών των cognitive πλατφορμών ενώ παράλληλα η σύνθεση τους πραγματοποιεί την διαδικασία της διαλειτουργικότητας ετερογενών τεχνολογιών. Τα γνωστικά συστήματα αναπτύσσουν την λειτουργία τους σύμφωνα με έναν **κύκλο γνώσης** (**Cognition Cycle**) ο οποίος αποτελεί ουσιαστικά το **σχέδιο δράσης του συστήματος**.

Στην κεφάλαιο 7 επιχειρούμε να δώσουμε μια σύντομη ανάλυση του cognition cycle μέσα από υπαρκτά ερευνητικά παραδείγματα και επιπλέον θα προσπαθήσουμε να διεξάγουμε ένα πιο ειδικό συμπέρασμα σχετικά με την ομοιότητα της δομής που μπορεί να έχει ο κύκλος γνώσης σε ένα cognitive σύστημα.

✓ **Κεφάλαιο 7**: «*Επισκόπηση του Cognition Cycle*»

Ο κύκλος των Cognitive συστημάτων, αποτελείται από διαφορετικές φάσεις λειτουργίας. Τα παραδοτέα κάθε φάσης αποτελούν πηγή πληροφόρησης για την επόμενη και μέσα από έναν κύκλο συνεργασίας ολοκληρώνεται κάθε φορά η αντίστοιχη λειτουργία που πρέπει να εκτελεστεί από το cognitive σύστημα. Οι λειτουργίες που συμπεριλαμβάνονται σε κάθε φάση δεν είναι σε όλες τις περιπτώσεις ίδιες. Ανάλογα με την λειτουργία του συστήματος μπορούμε να εντοπίσουμε λιγότερες ή περισσότερες λειτουργίες σε κάθε φάση. Επίσης διαφορά μπορούμε να εντοπίσουμε και στον σχεδιασμό του κύκλου γνώσης μελετώντας διαφορετικά γνωστικά συστήματα. Ωστόσο η βάση της διαδικασίας που διεκπεραιώνεται σε κάθε φάση συμπεριλαμβάνει συγκεκριμένα χαρακτηριστικά. Χαρακτηριστικό παράδειγμα αποτελεί η ένταξη λειτουργιών από το **knowledge management** στη φάση της **ανάλυσης**.

Επειδή ο cognition cycle παρουσιάζει ρευστότητα στον τρόπο με τον οποίο θα οριστεί κάθε φορά σε ένα γνωστικό σύστημα, προκειμένου να συγκεκριμενοποιήσουμε την αναφορά μας σχετικά με την εφαρμογή του σε μια cognitive πλατφόρμα η οποία διαχειρίζεται υπηρεσίες, θα αναφερθούμε σε μια προηγούμενη μελέτη που σχετίζεται με την cognitive τεχνολογία στα συστήματα ραδιοεπικοινωνιών.

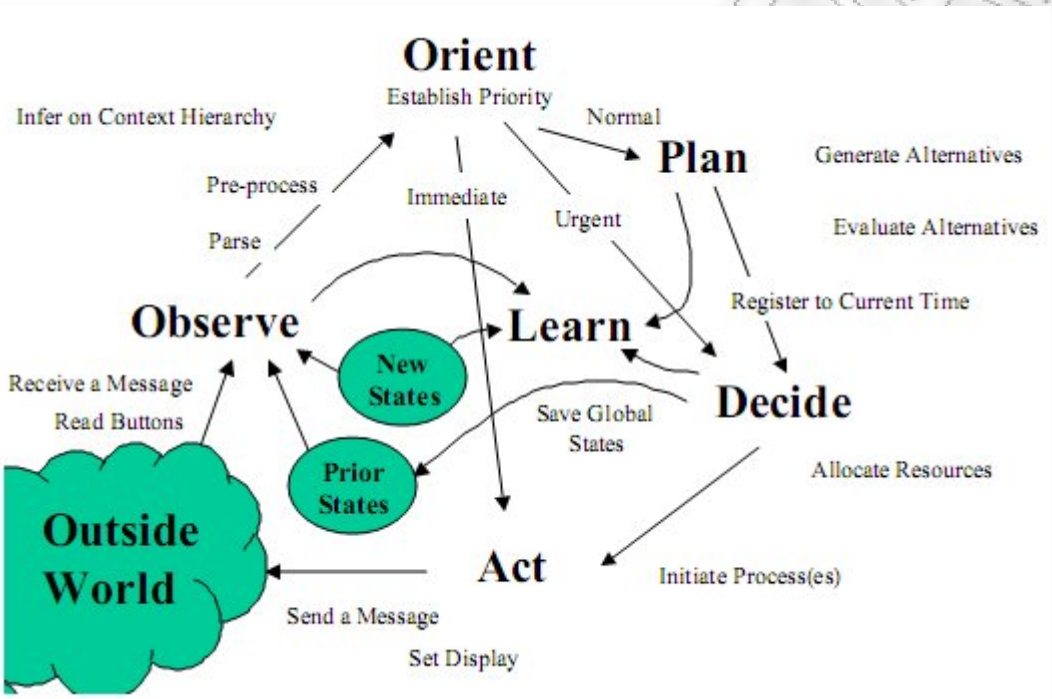
Η μελέτη είναι του Joseph Mitola[(79)] και αναφέρεται στην τεχνολογία των Cognitive Radios, όπου επιχειρεί την διαχείριση των radio systems με την βοήθεια της cognitive τεχνολογίας. Συμφώνα με τον J. Mitola, ο κύκλος της γνώσης στα γνωστικά συστήματα ράδιο-τεχνολογίας περιλαμβάνει **έξι** διαφορετικές φάσεις οι οποίες είναι: **Παρατήρηση** (Observe), **Εκτίμηση – Προσανατολισμός** (Orient), **Σχεδιασμός** (Plan), **Εκμάθηση** (Learn), **Απόφαση** (Decide) και **Δράση – Εκτέλεση** (Act). Η διαδικασία εκτελείτε ως εξής:

- Το σύστημα ανακτά πληροφορίες από το εξωτερικό περιβάλλον μέσω των λειτουργιών της **παρατήρησης**.
- Στην συνέχεια οι λειτουργίες της **εκτίμησης** λαμβάνουν αυτές τις πληροφορίες και τις αναλύουν προκειμένου να εκτιμηθεί η χρησιμότητα τους.
- Όταν ολοκληρωθεί η εκτίμηση των πληροφοριών πραγματοποιείται ο σχεδιασμός λύσεων με την βοήθεια των λειτουργιών του **σχεδιασμού**.
- Πραγματοποιείται εκτίμηση των λύσεων που προκύπτουν και με κριτήριο την βελτιστοποίηση του τελικού στόχου οι λειτουργίες **απόφασης** επιλέγουν την καταλληλότερη λύση.
- Η διαδικασία ολοκληρώνεται με την εκτέλεση των ενεργειών που συμπεριλαμβάνονται στην επιλεγμένη λύση, μέσω των λειτουργιών την **εκτέλεσης**.
- Τέλος το σύστημα ενσωματώνει την δυνατότητα συγκέντρωσης των αποτελεσμάτων από τις ενέργειες που έχουν εκτελεστεί και

[79] Mitola J. III, "**Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio**", Διδακτορική Διατριβή, 8-5-2000, σελ. 47-49.

συνδυάζοντάς τες με τις αντίστοιχες αποφάσεις που προηγήθηκαν δημιουργεί μια πολύτιμη πηγή μάθησης μέσα από την οποία μπορεί να αντλήσει πληροφορίες για την χρήση μελλοντικών λύσεων ανάλογα με τις ανάγκες του συστήματος. Η διαδικασία αυτή ολοκληρώνεται με την βοήθεια των λειτουργιών **μάθησης**.

Η εικόνα 15 απεικονίζει τον cognition cycle έτσι όπως αυτός διαμορφώνεται στο σύστημα που παρουσιάζεται στην ερευνητική μελέτη του J. Mitola.

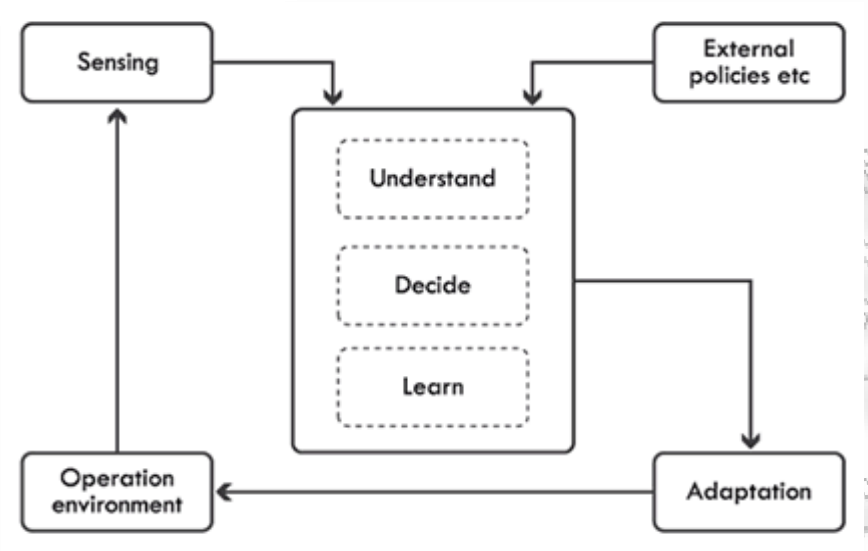


Εικόνα 15: "Cognition Cycle σε ένα Cognitive Radio System"[80]

Μια άλλη ενδιαφέρουσα προσέγγιση που αναφέρεται στον cognition cycle, είναι αυτή που παρουσιάζεται στο COGNAC PROJECT (Cognitive And Opportunistic Wireless Communication Networks) [81] το οποίο περιλαμβάνει μια συνολική μελέτη που αναφέρεται στα Γνωστικά Δίκτυα Ασύρματων Επικοινωνιών. Η εικόνα 16 παρουσιάζει το σχεδιάγραμμα του κύκλου γνώσης έτσι όπως προσεγγίζεται μέσα από το πρόγραμμα μελέτης του COGNAC.

[80] βλ. παρ. 66, **σελ.** 33.

[81] T. Braysy, M. Latva-aho, "**Cognitive and Opportunistic Wireless Communication Networks**", Cognac Project, 2008-2010, Seminar '08, σελ. 1.



Εικόνα 16: "Cognition cycle - COGNAC Project προσέγγιση"[82]

Στο COGNAC project, όπως αναφέρεται παραπάνω, η μελέτη αφορά τις ασύρματες επικοινωνίες. Η εφαρμογή της cognitive τεχνολογίας στοχεύει στην ορθή διαχείριση των πόρων με σκοπό αυτοί να αξιοποιηθούν με τον βέλτιστο δυνατό τρόπο από τα συστήματα ασύρματων επικοινωνιών. Η διαδικασία αυτή μπορεί να περιγραφεί μέσα από τον **κύκλο γνώσης,** που απεικονίζεται στην εικόνα 14, ο οποίος περιλαμβάνει μια σειρά διαφορετικών διαδικασιών οι οποίες συνεργάζονται μεταξύ τους με σκοπό να επιτύχουν την γνωστική διαχείριση των πόρων του συστήματος. Ο συγκεκριμένος κύκλος γνώσης συμπεριλαμβάνει **επτά** διαφορετικούς κόμβους όπου κάθε κόμβος αποτελεί μια ξεχωριστή φάση. Οι φάσεις είναι οι εξής: **Αντίληψη** (Sensing), **Κατανόηση** (Understand), **Απόφαση** (Decide), **Μάθηση** (Learn), **Προσαρμογή** (Adaptation), **Λειτουργικό Περιβάλλον** (Operation Environment) και **Εξωτερικοί παράγοντες** (External Policies, κλπ).

Με την εκτέλεση και ολοκλήρωση του κύκλου γνώσης το σύστημα κάθε φορά στοχεύει στην δυναμική διαχείριση των πόρων του σύμφωνα με τις ανάγκες που προκύπτουν. Ο στόχος του μπορεί να επιτευχθεί μέσα από μια σειρά ενεργειών όπως περιγράφονται στη συνέχεια:

- **αντίληψη** των παραγόντων από το εξωτερικό περιβάλλον και άλλων **εξωγενών παραγόντων** που αφορούν στοιχεία του συστήματος
- μελέτη και λήψη πληροφοριών από κάθε εξωτερικό στοιχείο και **κατανόηση** της λειτουργίας του
- δημιουργία εναλλακτικών σεναρίων χρήσης και **λήψη απόφασης** για την επιλογή της βέλτιστης δυνατής λύσης
- ενσωμάτωση γνώσης σε μια αποθήκη μάθησης του συστήματος με την βοήθεια των **λειτουργιών μάθησης**
- **προσαρμογή** της επιλεγμένης λύσης στα χαρακτηριστικά του συστήματος
- έλεγχος παραγόντων που αφορούν το **λειτουργικό περιβάλλον**

[82] **COGNAC - COGNitive And opportunistic wireless Communication networks**, http://www.cwc.oulu.fi/cwc-vtt-gigaseminar08/cognac.html, πρόσβαση: Απρίλιος, 2010.

Βασιζόμενοι στις δύο παραπάνω ερευνητικές προσεγγίσεις, θα επιχειρήσουμε να δώσουμε μια γενικότερη προσέγγιση στον κύκλο γνώσης που χρησιμοποιεί ένα σύστημα προκειμένου να εκτελέσει την λειτουργία του. Συγκεκριμένα θα μπορούσαμε να αναφέρουμε ότι ο κύκλος γνώσης συμπεριλαμβάνει διαδικασίες που αφορούν:

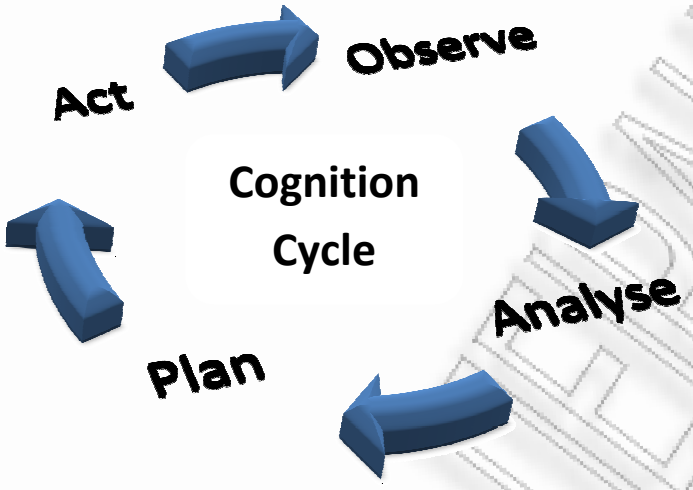
- Την **παρατήρηση** και μελέτη εξωτερικών παραγόντων που αλληλεπιδρούν με το σύστημα.
- Την **ανάλυση** των πληροφοριών που λαμβάνονται από τους εξωτερικούς παράγοντες και την προσαρμογή τους σε μορφή κατανοητή για το σύστημα.
- Τον **σχεδιασμό** εναλλακτικών ενεργειών που προκύπτουν από την ανάλυση των εξωτερικών πληροφοριών και την επιλογή της βέλτιστης λύσης.
- Την **εκτέλεση** της επιλεγμένης λύσης με σκοπό την διεκπεραίωση των διαδικασιών για την κάλυψη των αναγκών του συστήματος.

Επιπλέον λαμβάνοντας υπόψη μας και τη σχετική μελέτη **Virtualisation Platform for the introduction of cognitive systems in the Future Internet**, 2010, [(83)] θα μπορούσαμε να αναφέρουμε ***συμπερασματικά*** ότι, τα cognitive συστήματα αναπτύσσουν την λειτουργία τους σε τέσσερεις διαφορετικές φάσεις που ορίζονται από τον κύκλο γνώσης (Cognition Cycle), οι οποίες είναι οι εξής:

- **Παρατήρηση (Observe)**
- **Ανάλυση (Analyse)**
- **Σχεδιασμός (Plan)**
- **Εκτέλεση (Act)**

Το επίπεδο αφαίρεσης σε κάθε φάση μπορεί να είναι διαφορετικό ανάλογα με τις ανάγκες του συστήματος ή/και με τον τρόπο σχεδιασμού του κύκλου γνώσης από κάθε έρευνα. Ωστόσο ο στόχος είναι κοινός για κάθε γνωστικό σύστημα ανεξάρτητα με τον κύκλο γνώσης και αναφέρεται στην γνωστική διαχείριση και λειτουργία του συστήματος. Στο σχήμα 6 απεικονίζεται ο κύκλος γνώσης που περιγράφουμε πιο πάνω.

[83] Βλ. παρ. 10, **σελ.** 3.



Σχήμα 6: "Cognition Cycle"

7.1 Παρατήρηση – Observe

Η φάση της παρατήρησης αποτελεί την αρχή του cognition cycle και εκτελεί την διαδικασία ένταξης των τεχνολογικών στοιχείων στο cognitive σύστημα μέσα από μια σειρά διαφορετικών λειτουργιών. Η περιγραφή του κάθε στοιχείου (υλικού, λογισμικού, υπηρεσίας) πραγματοποιείται από τις αντίστοιχες λειτουργίες της πρώτης φάσης όπου το σύστημα λαμβάνει τη περιγραφή του στοιχείου και την προσαρμόζει σε μορφή κατανοητή για το σύστημα και τους επιμέρους χρήστες του.

7.2 Ανάλυση – Analyse

Περνώντας στη δεύτερη φάση του κύκλου γνώσης, το σύστημα λαμβάνει πληροφορίες από την προηγούμενη οι οποίες είναι σε κατανοητή μορφή και περιγράφουν τα χαρακτηριστικά των εξωτερικών στοιχείων. Μέσα από λειτουργίες μελέτης και ανάλυσης των πληροφοριών αυτών, το σύστημα διαχωρίζει τις σημαντικές – απαραίτητες, για την λειτουργία του, πληροφορίες και τροφοδοτεί την αποθήκη γνώσεων στην οποία στηρίζεται η διαδικασία μάθησης του συστήματος και επιπλέον τροφοδοτεί και την επόμενη φάση που αφορά τον σχεδιασμό εναλλακτικών λύσεων λειτουργία και διαμόρφωσης του συστήματος.

7.3 Σχεδιασμός – Plan

Στη φάση τρία το σύστημα καλείται να αξιοποιήσει τις πληροφορίες που προέκυψαν από την φάση της ανάλυσης, σχεδιάζοντας εναλλακτικές στρατηγικές λειτουργίας σύμφωνα με τις ανάγκες που έχουν προκύψει. Ολοκληρώνοντας την διαδικασία του σχεδιασμού το σύστημα πρέπει να πραγματοποιήσει την τελευταία λειτουργία της φάσης αυτής η οποία αναφέρεται στην λήψη της απόφασης για την επιλογή και χρήση της βέλτιστης δυνατής λύσης που συμπεριλαμβάνεται στις συνολικές λύσεις του σχεδιασμού.

7.4 Εκτέλεση – Act

Έχοντας πραγματοποιηθεί η επιλογή της καταλληλότερης λύσης για την λειτουργία του συστήματος, το μόνο που απομένει για να ολοκληρωθεί ο κύκλος διαδικασιών του cognitive συστήματος είναι η εκτέλεση της λύσης. Η διαδικασία αυτή πραγματοποιείται στην τελευταία φάση του κύκλου γνώσης επαναπροσδιορίζοντας τον τρόπο λειτουργίας του συστήματος.

Στο κεφάλαιο 8, θα προσπαθήσουμε να δώσουμε μια προσέγγιση **συσχέτισης** της **Service Oriented Computing τεχνολογίας**, όπως αυτή παρουσιάζεται από το Research road map (*κεφ. 5, εν. 5.3*), με τον **κύκλο γνώσης (cognition cycle)**, όπως αυτός παρουσιάζεται στις συμπερασματικές παρατηρήσεις του συγκεκριμένου κεφαλαίου.

✓ **Κεφάλαιο 8**: *«Συσχέτιση SOC Research road map και Cognition cycle»*

Στην ενότητα αυτή θα επιχειρήσουμε να δώσουμε μια διαφορετική προσέγγιση της SOC τεχνολογίας σε σχέση με τα Cognitive συστήματα. Συγκεκριμένα λαμβάνοντας υπόψη μας τις παραπάνω αναλύσεις θα επιχειρήσουμε να δώσουμε μια συσχέτιση μεταξύ των χαρακτηριστικών που εντοπίζονται στην SOC με τα χαρακτηριστικά που εντοπίζονται στην λειτουργία ενός cognitive συστήματος. Η SOC τεχνολογία συμπεριλαμβάνει τον Research road map όπου ορίζονται σε μορφή ιεραρχίας τρία διαφορετικά επίπεδα λειτουργιών. Τα επίπεδα αυτά είναι: i) **Service Foundation**, ii) **Service Composition** και iii) **Service management & monitoring**. Από την άλλη πλευρά αναλύσαμε τον κύκλο γνώσης σύμφωνα με τον οποίο ένα cognitive σύστημα ορίζει την λειτουργία του και καταλήξαμε σε ένα γενικό μοντέλο το οποίο συμπεριλαμβάνει τέσσερεις διαφορετικές φάσεις οι οποίες είναι: i) **Observe** ii) **Analyse** iii) **Plan** και iv) **Act**.

Σε μια προσπάθεια να συσχετίσουμε τα επίπεδα του SOC Research road map με τις φάσεις του κύκλου γνώσης θα μπορούσαμε να δώσουμε την αντιστοιχία που παρουσιάζεται στον *πίνακα 1*.

SOC Research road map	Cognition Cycle
Service Foundation	*Observe* *Analyse*
Service Composition	*Plan*
Service Management & Monitoring	*Act*

Πίνακας 1:"Συσχέτιση SOC Research road map - Cognition Cycle"

Λαμβάνοντας υπόψη μας τις παραπάνω σχετικές αναφορές θα επιχειρήσουμε την αιτιολόγηση του συσχετισμού που απεικονίζεται στον πίνακα 1. Θα πρέπει πρώτα όμως να αναφέρουμε ότι το μοντέλο του SOC Research road map αναφέρεται στην διαδικασία διαχείρισης συστημάτων που διαμορφώνουν την λειτουργία τους σύμφωνα με την Service Oriented αρχιτεκτονική, ενώ ο Cognition Cycle αναφέρεται στην διαχείριση συστημάτων τα οποία διαμορφώνουν την λειτουργία τους σύμφωνα με το μοντέλο της γνωστικής διαχείρισης των πόρων τους. Ακολουθεί η τεκμηρίωση της προσέγγισης μας για τον συσχετισμό του SOC Research road map με τον Cognition Cycle.

8.1 Service Foundation & Observe, Analyse

Η διαδικασία του service foundation περιλαμβάνει λειτουργίες που εκτελούνται ώστε να υλοποιηθεί η δικτύωση διαφορετικών εξωτερικών υπηρεσιών με το service oriented σύστημα. Περιλαμβάνουν την εισαγωγή μιας ειδικής περιγραφής σε κάθε εξωτερική υπηρεσία, που εμφανίζεται στο σύστημα, προκειμένου αυτή να γίνει κατανοητή από το σύστημα. Στην συνέχεια το SOA σύστημα αναλύσει όλες τις διαθέσιμες πληροφορίες από το μήνυμα περιγραφής

που προέκυψε για την υπηρεσία και παράγει τις απαραίτητες πληροφορίες που θα χρησιμοποιηθούν από το επίπεδο του service composition.

Από την άλλη πλευρά οι φάσεις παρατήρηση (Observe) και ανάλυση (Analyse) ολοκληρώνουν μια παρόμοια διαδικασία στον κύκλο γνώσης ενός γνωστικού συστήματος. Αναλαμβάνουν την διαδικασία ελέγχου και παρατήρησης των στοιχείων που προέρχονται από τον εξωτερικό κόσμο (συσκευές, λογισμικό, υπηρεσίες, κλπ.) και αφού απεικονίσουν τα δεδομένα που αφορούν το κάθε στοιχείο προχωρούν στην ανάλυση τους. Όταν ολοκληρωθεί και η λειτουργία της ανάλυσης έπεται η διαδικασία του σχεδιασμού βοηθητικών λύσεων για την διαμόρφωση της λειτουργίας του συστήματος.

Έχοντας περιγράψει τις εμφανιζόμενες διαδικασίες διαπιστώνουμε ότι εκτελείτε μια παρόμοια διαδικασία σε κάθε σύστημα (service oriented σύστημα και cognitive σύστημα) η οποία στοχεύει στην ομαλή ενσωμάτωση των υπηρεσιών και των στοιχείων, αντίστοιχα, μέσα από μια σειρά λειτουργιών περιγραφής και λήψης δεδομένων των εξωτερικών παραγόντων.

8.2 Service composition & Plan

Το επίπεδο του service composition περιλαμβάνει λειτουργίες οι οποίες εκτελούν την σύνθεση διαφορετικών υπηρεσιών με σκοπό την δημιουργία μιας ή περισσοτέρων σύνθετων εφαρμογών που θα εκτελούν κάποια συγκεκριμένη λειτουργία στο service oriented σύστημα. Επιπλέον προάγει λειτουργίες που στοχεύουν στην εύρεση λειτουργικών λύσεων οι οποίες θα είναι σε θέση να διεκπεραιώσουν, μέσα από την εκτέλεση τους, μια σειρά διαδικασιών για την κάλυψη των απαιτήσεων που προκύπτουν σε ένα σύστημα.

Στον cognition cycle εντοπίζουμε μια παρόμοια διαδικασία η οποία ενεργοποιείται και ολοκληρώνεται στην φάση του σχεδιασμού. Οι πληροφορίες που έχουν προκύψει από την παρατήρηση και την ανάλυση τροφοδοτούν την φάση του σχεδιασμού με τα απαραίτητα δεδομένα, ώστε το σύστημα να μπορέσει να αναπτύξει λύσεις αντιμετώπισης μιας κατάστασης που έχει προκύψει (π.χ.: την σύνδεση μιας νέας συσκευής). Αφού σχεδιαστούν οι απαραίτητες λύσεις, στην συνέχεια με την βοήθεια μηχανισμών λήψης απόφασης (Decision Making Mechanisms) το σύστημα είναι σε θέση να επιλέξει την βέλτιστη δυνατή λύση για την κάλυψη των απαιτήσεων που προέκυψαν.

Διαπιστώνουμε λοιπόν, ότι στόχος των λειτουργιών που παρουσιάζονται σε κάθε περίπτωση λειτουργίας του αντίστοιχου συστήματος είναι ο σχεδιασμός ευέλικτων λύσεων για την δυναμική αντιμετώπιση καταστάσεων που μπορεί να προκύψουν στο σύστημα. Ο σχεδιασμός συμπεριλαμβάνει την σύνθεση διαφορετικών πληροφοριών που έχουν προέλθει από την επεξεργασία δεδομένων στις προηγούμενες διαδικασίες. Για το λόγο αυτό παραθέτουμε και τον συγκεκριμένο συσχετισμό μεταξύ του service composition και του σχεδιασμού (plan) του cognition cycle.

8.3 Service Management & Monitoring & Act

Στην κορυφή της ιεραρχίας του SOC Research road map βρίσκεται το επίπεδο του Service Management & Monitoring το οποίο συγκεντρώνει λειτουργίες ελέγχου και διαχείρισης των υπηρεσιών που εντάσσονται σε ένα service oriented σύστημα. Η αναπροσαρμογή της λειτουργίας ενός συστήματος κρίνεται πολλές

φόρες απαραίτητη για την ομαλή εκτέλεση του. Επίσης η βελτίωση των υπαρκτών λειτουργιών στο σύστημα είναι εξίσου σημαντική για την αύξηση της αποδοτικότητας του συστήματος. Μέσα από την διαδικασία αυτή λοιπόν, ένα service oriented σύστημα αποκτά την δυνατότητα της δυναμικής διαχείρισης της λειτουργίας του η οποία θα συμβάλει στην αποδοτικότητα του.

Η τελευταία φάση που συμπεριλαμβάνεται στον κύκλο γνώσης είναι εκείνη της εκτέλεσης των επιλεγμένων λύσεων από την διαδικασία του σχεδιασμού. Παράλληλα με την εκτέλεση το γνωστικό σύστημα ενεργοποιεί ένα σύνολο μηχανισμών παρακολούθησης της λειτουργία του οι οποίοι παράγουν δεδομένα μέσω των οποίων το σύστημα είναι σε θέση να εντοπίσει ελλείψεις ή δυσλειτουργίες που μπορεί να προκύψουν κατά την εκτέλεση μιας διαδικασίας. Σε περίπτωση που συμβεί κάτι τέτοιο, το σύστημα μέσα από ένα σύνολο μηχανισμών αναδιάρθρωσης που διαθέτει, είναι σε θέση να πραγματοποιήσει αναπροσαρμογή της λειτουργίας του.

Συμπεραίνουμε λοιπόν ότι και στην διαδικασία της εφαρμογής των αποτελεσμάτων από την επεξεργασία και τον σχεδιασμό των δεδομένων τόσο στο Service Management & Monitoring όσο και στην φάση της εκτέλεσης του cognition cycle εκτελούνται λειτουργίες συντήρησης και βελτίωσης του συστήματος με σκοπό την εξέλιξη της απόδοσης του.

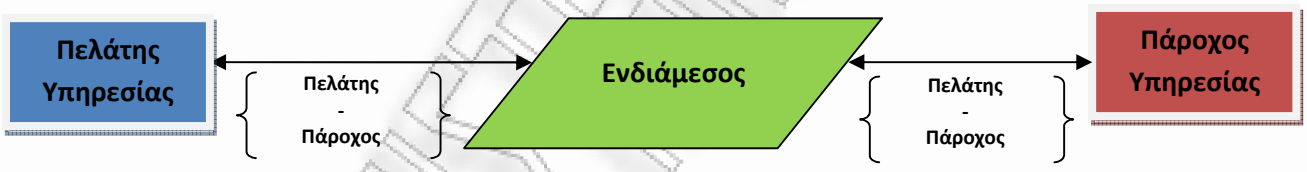
Βασιζόμενοι λοιπόν στην παραπάνω προσέγγιση μέσα από την οποία επιχειρούμε την συσχέτιση των φάσεων λειτουργίας του κύκλου γνώσης με τα επίπεδα λειτουργίας του SOC Research road map μπορούμε να διαπιστώσουμε ότι η διαμόρφωση των αρχών λειτουργίας των cognitive συστημάτων **επηρεάζεται άμεσα** από τις αρχές που διέπουν την Service Oriented Αρχιτεκτονική οι οποίες προάγονται μέσω της εφαρμογής της Service Oriented Computing τεχνολογία, στην ανάπτυξη συστημάτων. Ολοκληρώνοντας θα μπορούσαμε να αναφέρουμε με ασφάλεια ότι, **η υλοποίηση των cognitive συστημάτων αποτελεί την πρακτική εφαρμογή της Service Oriented Computing τεχνολογίας** μέσω της οποία επιτυγχάνεται ο σχεδιασμός και η υλοποίηση συστημάτων τα οποία ενοποιούν ετερογενείς τεχνολογίες προσφέροντας ένα περιβάλλον διαλειτουργικότητας το οποίο δύναται να κάνει πράξη, μέσω γνωστικών λειτουργιών, την γνωστική διαχείριση των πόρων του.

✓ Κεφάλαιο 9: «*Virtualization*»

9.1 Ανάλυση της διαδικασίας του Virtualization

Ο όρος Virtualization αναφέρεται στην διαδικασία ενσωμάτωσης λογισμικού μέσα σε ένα εικονικό περιβάλλον λειτουργίας.[(84)] Οι λειτουργίες ενός πόρου υλικού μετατρέπονται σε λογισμικό το οποίο μπορεί να εκτελεστεί σε κεντρικούς υπολογιστές οι οποίοι μπορούν να συνδυάζουν διαφορετικές εφαρμογές σε ένα σύστημα. Η εικονικοποίηση (Virtualization) επίσης, αναφέρεται σε έναν μηχανισμό αφαίρεσης ο οποίος αποκρύπτει λεπτομέρειες της υλοποίησης και της κατάστασης ορισμένων υπολογιστικών πόρων από πελάτες των πόρων αυτών (π.χ. εφαρμογές, άλλα συστήματα, χρήστες κλπ).[(85)] Επιπλέον επιτυγχάνουμε μια υψηλού επιπέδου αφαίρεση στις διαδικασίες διασύνδεσης η οποία αποκρύπτει στοιχεία πολυπλοκότητας που αφορούν τις τεχνολογίες και επίσης διευκολύνει την ενοποίηση διαφορετικών τεχνολογιών.

Στην προσπάθεια μας να δώσουμε μια γενική περιγραφή της διαδικασίας του Virtualization, μπορούμε να αναφέρουμε τα εξής: σε ένα εικονοποιημένο σύστημα υπάρχουν τρεις συμμετέχοντες: **ο πελάτης μιας υπηρεσίας, ο παροχέας της υπηρεσίας** και **ένας ενδιάμεσος**. Ο ενδιάμεσος πραγματοποιεί την εικονοποίηση λειτουργώντας ως πελάτης για τον παροχέα και ως πάροχος για τον πελάτη. Το σχήμα 7 απεικονίζει ακριβώς την διαδικασία που μόλις περιγράψαμε.

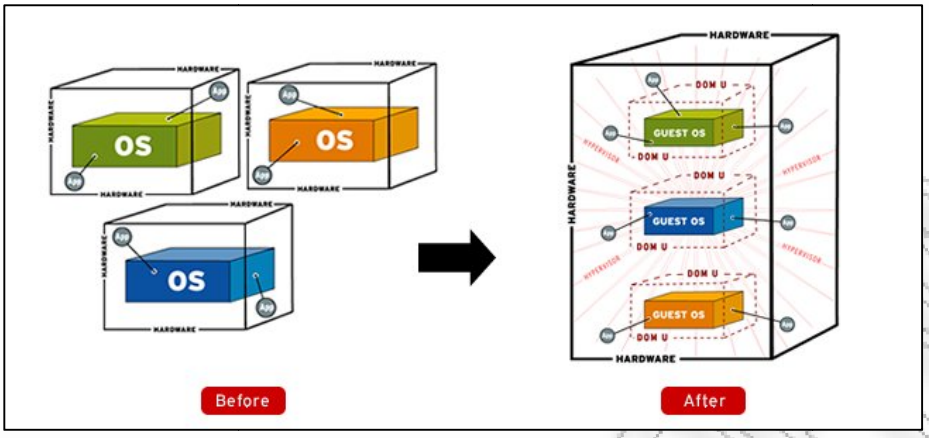


Σχήμα 7: "Διαδικασία Virtualization"

Στη συνέχεια, η εικόνα 17 αποτελεί μια απλή απεικόνιση της διαδικασίας του virtualization κατά την οποία επιτυγχάνεται η συγκέντρωση τριών διαφορετικών λειτουργικών συστημάτων, τα οποία είναι εγκατεστημένα σε διαφορετικούς πόρους υλικού, σε ένα κοινό πόρο υλικού (πχ: server) όπου μπορούν να αναπτύξουν τις λειτουργίες τους.

[84] **Microsoft Official Site – Virtualization,** http://www.microsoft.com/uk/licensing/lessthan250/learn/virtualisation.mspx, πρόσβαση: Απρίλιος, 2010.

[85] **Platform Virtualization**, http://en.wikipedia.org/wiki/Platform_virtualization, πρόσβαση: Μάρτιος, 2010.



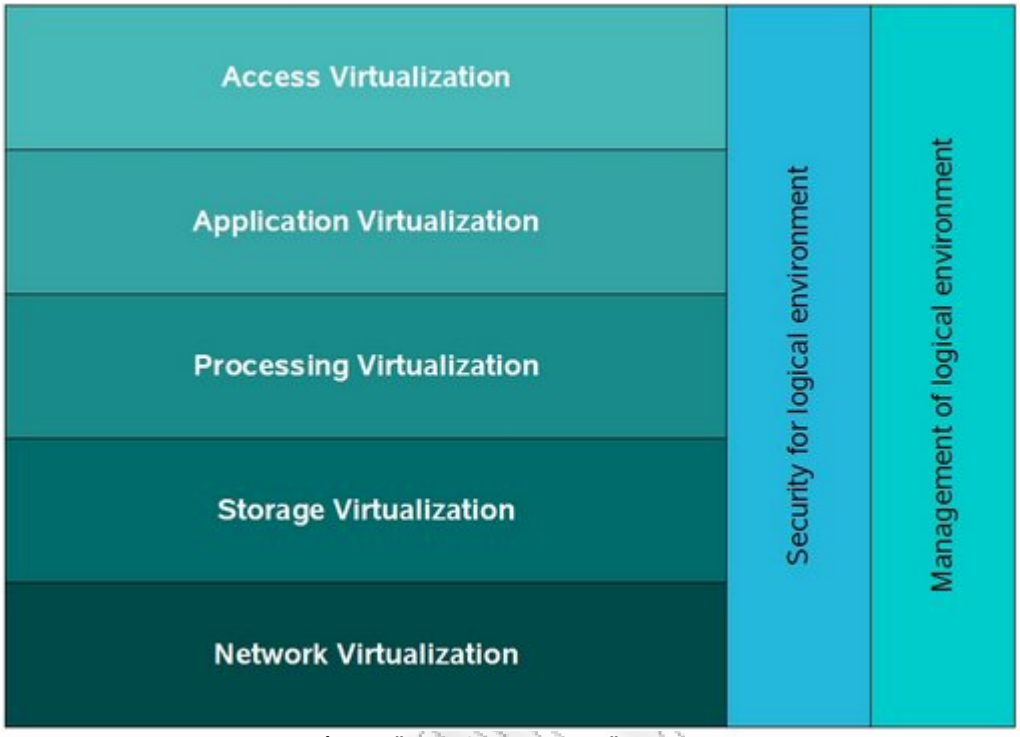
Εικόνα 17:"Διαδικασία Virtualisation λειτουργικών συστημάτων"

Η διαδικασία του Virtualization των πόρων αποτελεί πλέον την νέα τάση στο χώρο των πληροφοριακών συστημάτων και εισάγει μια σειρά από διαφορετικά πλεονεκτήματα, όπως: ευκολία επιδιόρθωσης βλαβών του λογισμικού, μείωση του κόστους ιδιοκτησίας και μείωση της ενεργειακής κατανάλωσης με την μείωση των πόρων υλικού. Τα παραπάνω πλεονεκτήματα είναι μερικά από τα οποία οδήγησαν στην εξάπλωση της διαδικασίας του Virtualization, στους επιμέρους τομείς της πληροφοριακής τεχνολογίας. Το 2007 ο αναλυτής Dan Kusnetsky παρουσίασε ένα μοντέλο που απεικονίζει τους τομείς στους οποίους εισάγεται το virtualization,[(86)] το οποίο απεικονίζεται στην εικόνα 18 παρακάτω. Συγκεκριμένα σύμφωνα με τον Dan Kusnetsky το Virtualization ορίζεται σε μια σειρά από διαφορετικά υποστρώματα λειτουργίας τα οποία είναι τα εξής:

- Access Virtualization
- Application Virtualization
- Processing Virtualization
- Storage Virtualization
- Virtualization

Σύμφωνα λοιπόν με το μοντέλο αυτό προκύπτουν **πέντε διαφορετικές κατηγορίες Virtualization** μέσα από τις οποίες μπορούμε εύκολα να προσδιορίσουμε και τις τεχνολογικές ανάγκες που θα προκύψουν με την εφαρμογή του Virtualization σε κάθε κατηγορία.

[86] **AIO Blog - Analyst helps "sort out" layers of virtualization,** http://www.aiosolutions.com/, πρόσβαση: Απρίλιος, 2010.



Εικόνα 18:"Virtualization Layers"

Υπάρχουν τρία βασικά είδη Virtualization που εφαρμόζονται ανάλογα με την λειτουργία που θέλουμε να επιτύχουμε. Αναφορικά αυτά είναι:[(87)]

- **Virtual Machines** (VMs) → εικονικοποίηση των πόρων που αντιστοιχούν στο υλικό.
- **Paravirtualization** → εικονικοποίηση των πόρων που αντιστοιχούν στο λογισμικό των συσκευών που επιχειρούν να συνδεθούν με μια VM.
- **Virtualization on Operating Systems** → εικονικοποίηση των πόρων διαφορετικών λογισμικών και συνδυασμός σε ένα ενιαίο λογισμικό ή λειτουργικό σύστημα.

Το κάθε είδος εφαρμόζεται ανάλογα με τις ανάγκες του συστήματος που εισάγει το virtualization στην λειτουργία του. Το αποτέλεσμα ωστόσο είναι κοινό και στοχεύει στην ομαλή ενοποίηση διαφορετικών τεχνολογιών μέσω αυτοματοποιημένων διαδικασιών. Η διαδικασία του virtualization αποτελεί βασικό παράγοντα για την ανάπτυξη ευέλικτων και δυναμικών υπολογιστικών συστημάτων που θα αυτοματοποιούν τις υποδομές τους[(88)] μετατρέποντάς τες σε δυναμικά επεκτάσιμες.

Η λειτουργία που εκτελείτε από το Virtualization των πόρων στην πραγματικότητα μετατρέπει τις λειτουργικές διαδικασίες σε σύνθετες υπηρεσίες. Δηλαδή αφού εξετάσει την λειτουργία ενός στοιχείου υλικού ή

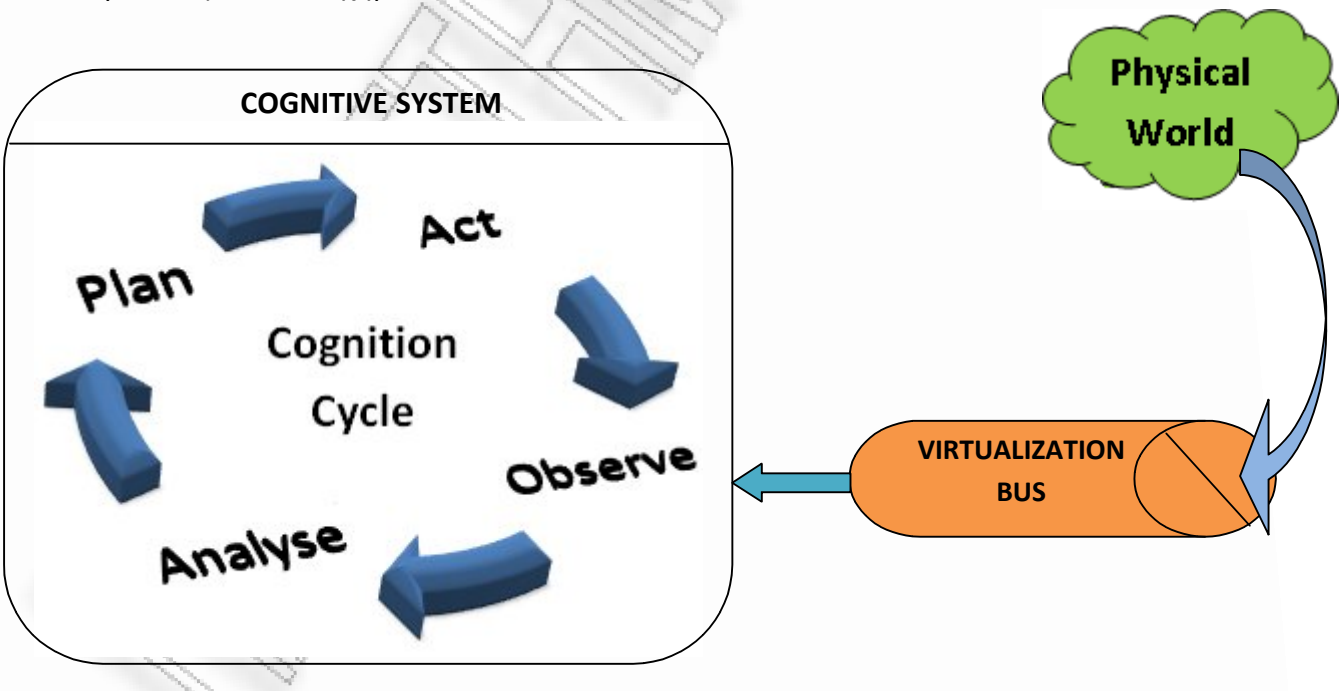
[87] **Introduction to Virtualization,** http://wiki.openvz.org/Introduction_to_virtualization, πρόσβαση: Απρίλιος, 2010.

[88] R. Figueiredo, "**An Overview of Virtualization Techniques**", University of Florida, NCN/NMI Team, 2-3-2006.

λογισμικού, στην συνέχεια επιχειρεί να δημιουργήσει τις αντίστοιχες υπηρεσίες που θα εκτελούν ακριβώς την λειτουργία που εξετάστηκε. Η ιδέα του Virtualization **στοχεύει στην δυναμική προσθήκη και αφαίρεση υπηρεσιών σε ένα σύστημα**, ενώ επιπλέον διαχωρίζει τις υπηρεσίες από σημεία εξάρτησης (π.χ.: στατικό URL).[89]

9.2 Virtualization και Cognitive Systems

Τα cognitive συστήματα επιχειρούν να συνδυάσουν ετερογενείς τεχνολογίες σε ενιαία περιβάλλοντα λειτουργίας τα οποία θα έχουν την δυνατότητα να διαχειρίζονται έξυπνα τους πόρους τους. Η διαδικασία ενσωμάτωσης των ετερογενών τεχνολογιών σε ένα γνωστικό σύστημα, αποτελεί μια πολύ σημαντική λειτουργία η οποία καθορίζει την λήψη και την επεξεργασία πληροφοριών από το εξωτερικό περιβάλλον του συστήματος. Μια ιδιαίτερα ασφαλής διαδικασία για την πραγματοποίηση της ενσωμάτωσης τεχνολογικών στοιχείων σε ένα cognitive σύστημα είναι η διαδικασία του Virtualization. Η λήψη πληροφοριών για κάθε στοιχείο και η απεικόνιση των λειτουργιών του με την μορφή υπηρεσιών μπορεί να υλοποιηθεί με την χρήση των τεχνικών που προσφέρει το Virtualization. Έτσι σε μια προσπάθεια να προσεγγίσουμε την λειτουργία ενός γνωστικού συστήματος το οποίο ενσωματώνει υπηρεσίες σε ένα κοινό περιβάλλον λειτουργίας με την βοήθεια του Virtualization, θα μπορούσαμε να δώσουμε την σχεδιαστική δομή που παρουσιάζεται στο σχήμα 8.



Σχήμα 8: "Η διαδικασία του Virtualization στα Cognitive Συστήματα"

[89] βλ. παρ.9, **σελ.** 3.

Οι υπηρεσίες που προκύπτουν και ενσωματώνονται στο σύστημα μπορούν να οργανωθούν με την βοήθεια της Service Oriented αρχιτεκτονικής, μπορούν να είναι δυναμικά διαχωρίσιμες, εφαρμόζοντας τις φάσεις του κύκλου γνώσης, ενώ παράλληλα μπορούν να αναπροσαρμόζονται δυναμικά με την βοήθεια των λειτουργιών που προσφέρει η διαδικασία του Virtualization.

Συμπερασματικά θα μπορούσαμε να αναφέρουμε ότι η εισαγωγή της διαδικασίας του Virtualization σε ένα Cognitive σύστημα, συμβάλει στην διευκόλυνση των διαδικασιών που αφορούν την αυτοματοποίηση της λειτουργίας του συστήματος. Οι Self-X ιδιότητες ενός γνωστικού συστήματος, πρακτικά πλέον θα μπορούν να υλοποιηθούν με μεγαλύτερη ευκολία από τους developers των συστημάτων. Το μοντέλο συστήματος που βασίζει την λειτουργία του σε υπηρεσίες είναι σίγουρα πιο ευέλικτο από ένα σύστημα που βασίζεται, για παράδειγμα, στον αντικειμενοστραφή προγραμματισμό.

✓ *Ανακεφαλαιώνοντας*

Στο σημείο αυτό ολοκληρώνεται το μέρος Ά της παρούσας μελέτης, έχοντας παραθέσει πλήρεις αναφορές που αντιστοιχούν στις απαιτήσεις της σύγχρονης αρχιτεκτονικής του FI και την ανάπτυξη Service Oriented συστημάτων ικανών να ανταποκριθούν στις απαιτήσεις αυτές.

Στην συνέχεια ακολουθεί το μέρος Β΄ στο οποίο θα επιχειρήσουμε να παραθέσουμε αναλυτικά την αρχιτεκτονική δομή της Service Oriented Virtualization Platform, μέσα από ένα ολοκληρωμένο σχέδιο αρχιτεκτονικής. Επιπλέον θα αναφερθούμε αναλυτικά στην κάθε κατηγορία υπηρεσιών που περιλαμβάνει η πλατφόρμα καθώς και θα παραθέσουμε το μοντέλο συνεργασίας των blocks υπηρεσιών.

Μέρος Β΄

«Αρχιτεκτονικός σχεδιασμός της Service Oriented Virtualization Platform ως σύστημα του Future Internet»

✓ **Κεφάλαιο 10:** «Ανάλυση συστήματος SOVP – System Analysis»

Όπως ήδη έχουμε αναφέρει στο μέρος Α΄ (*κεφ. 2*), σκοπός της παρούσας μελέτης είναι η ανάπτυξη ενός σύγχρονου, ευφυούς συστήματος το οποίο θα μπορεί να προσαρμοστεί και να ανταπεξέλθει στις απαιτήσεις του μελλοντικού διαδικτύου. Βασιζόμενοι στην θεωρία που αφορά την Service Oriented αρχιτεκτονική είμαστε σε θέση να συγκεκριμενοποιήσουμε την «φύση» του συστήματος που επιχειρούμε να σχεδιάσουμε και στην συνέχεια να αναπτύξουμε. Μπορούμε να πούμε λοιπόν ότι, **η SOVP θα απεικονιστεί ως μια Service Oriented πλατφόρμα λογισμικού η οποία θα συνδυάζει ετερογενείς τεχνολογίες και θα συγκεντρώνει έναν μεγάλο αριθμό πόρων** – συσκευές, δικτυακές τεχνολογίες και λογισμικό – **ικανών να συνεργαστούν στη βάση της πλατφόρμας**.

Η διαδικασία της εικονικοποίησης (Virtualization) υλοποιεί την διαδικασία συγκέντρωσης και μετατροπής πόρων υλικού και λογισμικού σε υπηρεσίες. Επίσης μας είναι γνωστό από προηγούμενες αναφορές, τόσο στην παρούσα μελέτη όσο και από προγενέστερες ερευνητικές διαδικασίες, ότι η διαδικασία του Virtualization πραγματοποιεί την ενσωμάτωση διαφορετικών φυσικών πόρων υλικού και λογισμικού σε ενιαίες μονάδες λειτουργίας (πχ Server Systems) με αποτέλεσμα την συγκέντρωση ανεξάρτητων και ετερογενών τεχνολογιών σε ένα κοινό σύστημα λειτουργίας.[(90)]

Η διαδικασία της εικονικοποίησης των πόρων αποτελεί ένα από τα σημαντικότερα συστατικά για την ανάπτυξη συστημάτων που θα ανταποκρίνονται στις απαιτήσεις του μελλοντικού διαδικτύου. Ωστόσο δεν αρκεί μόνο η εικονικοποίηση των πόρων προκειμένου ένα σύστημα να λειτουργήσει ομαλά στο περιβάλλον του Future Internet. Θα πρέπει να διέπεται από κάποιες οργανωτικές αρχές σύμφωνα με τις οποίες θα μπορεί να οργανώσει ορθά το σύνολο των εικονικοποιημένων πόρων (υπηρεσίες) που συγκεντρώνει στο πλαίσιο λειτουργίας του. Η Service Oriented αρχιτεκτονική αποτελεί την λύση στην συγκεκριμένη απαίτηση του συστήματος και εφαρμόζοντας τις αρχές οργάνωσης και διαχείρισης των υπηρεσιών, που πρεσβεύει, επιτυγχάνουμε την ανάπτυξη συστημάτων υψηλής αποδοτικότητας. Συγκεκριμενοποιώντας τον ορισμό της SOA και λαμβάνοντας υπόψη μας την σχετική αναφορά στο κεφάλαιο **5** (*εν.* ***5.2***), μπορούμε να αναφέρουμε ότι η SOA αποτελεί ένα αρχιτεκτονικό στυλ για την «οικοδόμηση» συστημάτων βασισμένα στις υπηρεσίες.[(91)] Από την άλλη πλευρά η SOC αναπτύσσει την διαδικασία εφαρμογής των αρχών της SOA σε ένα σύστημα και ουσιαστικά συμβάλει στην ανάπτυξη συστημάτων τα οποία βασίζουν την λειτουργία τους στις υπηρεσίες.

---

[90] **Definition of: Virtualization**, http://www.pcmag.com/encyclopedia_term/0,2542,t=virtualization&i=53961,00.asp, πρόσβαση: Μάιος 2010.

[91] Βλ. παρ. 32, **σελ**. 17.

Σύμφωνα λοιπόν με τα παραπάνω, στην προσπάθεια ανάπτυξης συστημάτων που θα μπορούν να προσαρμοστούν στην σύγχρονη τεχνολογική τάση του FI, καταλήγουμε στην υλοποίηση συστημάτων τα οποία βασίζουν την λειτουργία τους στις **υπηρεσίες** οι οποίες αποτελούν την **εικονικοποίηση πόρων υλικού και λογισμικού** μέσω της διαδικασίας του **Virtualization**, ενώ η οργάνωση και διαχείριση των υπηρεσιών γίνεται εφικτή μέσα από τον κατάλληλο **σχεδιασμό του συστήματος** ο οποίος διαμορφώνεται σύμφωνα με την **Service Oriented αρχιτεκτονική**. Χρησιμοποιώντας την διαδικασία που μόλις περιγράψαμε ως κεντρική ιδέα για την ανάπτυξη ενός συστήματος για το μελλοντικό διαδίκτυο, αιτιολογούμε την ανάγκη που προέκυψε για μελέτη, σχεδιασμό και ανάπτυξη της Service Oriented Virtualization Platform (SOVP) ως ένα πραγματικό σύστημα ικανό να ανταποκριθεί στις απαιτήσεις που παρουσιάζονται με την εμφάνιση του μελλοντικού διαδικτύου.

Στις ακόλουθες ενότητες θα επιχειρήσουμε να δώσουμε μια συγκεντρωτική καταγραφή των αναγκών προς κάλυψη από ένα σύστημα όπως η SOVP. Με τον τρόπο αυτό θα καταφέρουμε να έχουμε στην διάθεση μας όλες τις απαραίτητες συνιστώσες που θα πρέπει να συνυπολογιστούν κατά τον σχεδιασμό και αργότερα κατά την υλοποίηση του συστήματός μας. Στο σημείο αυτό θα πρέπει να αναφέρουμε ότι η θεωρία που σχετίζεται με την **ανάλυση συστημάτων (system analysis)**, αποτελεί τον κύριο άξονα καθοδήγησης της προσπάθειάς καταγραφής και ανάλυσης των απαιτήσεων. *Η ανάλυση συστημάτων αναφέρεται στη μελέτη ενός συνόλου οντοτήτων που αλληλεπιδρούν μεταξύ τους κατά την εκτέλεση μια διαδικασίας.*[(92)] Σύμφωνα με την θεωρία της ανάλυσης συστημάτων, οι απαιτήσεις τους συστήματος μπορούν να χωριστούν σε **ποιοτικές απαιτήσεις** που προκύπτουν από την φυσική υπόσταση του συστήματος (απαιτήσεις χρηστών, φυσικές υποδομές, κλπ.) και σε **απαιτήσεις σχεδιασμού και ανάπτυξης** (γλώσσες προγραμματισμού, βάσεις δεδομένων, δικτυακές τεχνολογίες, κλπ.)[(93)] του συστήματος η οποίες στοχεύουν στην κάλυψη των ποιοτικών απαιτήσεων.

10.1 Ποιοτικές απαιτήσεις του συστήματος Service Oriented Virtualization Platform

Στις ποιοτικές απαιτήσεις των συστημάτων συγκαταλέγονται όλοι εκείνοι οι παράγοντες που σχετίζονται με τους στόχους του συστήματος και αφορούν την κάλυψη αναγκών από το σύστημα στο φυσικό κόσμο. Ουσιαστικά αναφέρονται οι αιτίες που μας οδηγούν στην ανάπτυξη ενός συστήματος μέσα από την καταγραφή αυτών των αιτιών. Σε μηδενικό επίπεδο αφαίρεσης μπορούμε να πούμε ότι η ποιοτική απαίτηση της SOVP είναι η ανάπτυξη ενός συστήματος το οποίο θα είναι ικανό να προσαρμοστεί στην καινοτόμο τεχνολογία που εισάγει το Future Internet. Στη συνέχεια θα προσπαθήσουμε να καταγράψουμε το σύνολο των ποιοτικών απαιτήσεων της πλατφόρμας που θα υλοποιηθεί, προκειμένου να δημιουργήσουμε μια ολοκληρωμένη εικόνα σχετικά με τους στόχους λειτουργία του συστήματος. Η καταγραφή των ποιοτικών απαιτήσεων θα μας χρησιμεύσει στον ορισμό των απαιτήσεων ανάπτυξης και σχεδιασμού της πλατφόρμας.

---

[92] **System Analysis**, http://en.wikipedia.org/wiki/Systems_analysis, πρόσβαση: Μάιος 2010.

[93] Μπαναζούντας, «**Τεχνικές Ανάλυσης Συστημάτων»**, Εθνικό Μετσόβιο Πολυτεχνείο, 2-11-2001, σελ. 2

10.1.1 Ενοποίηση ετερογενών τεχνολογιών

Συμβαίνει συχνά να αντιμετωπίζουμε προβλήματα συμβατότητας κατά την προσπάθεια μας να χρησιμοποιήσουμε μια εφαρμογή σε ένα τεχνολογικά μεταγενέστερο σύστημα. Αυτό προκύπτει εξαιτίας της ετερογένειας των τεχνολογιών που χρησιμοποιεί το σύστημα με την τεχνολογία στην οποία μπορεί να λειτουργήσει η εφαρμογή. Για την ευκολότερη κατανόηση αυτού που αναφέρουμε μπορούμε να παραθέσουμε το παράδειγμα με τα λειτουργικά συστήματα των windows. Οι εφαρμογές που έχουν δημιουργηθεί για να λειτουργούν στην πλατφόρμα των Windows XP, δεν μπορούν να εκτελεστούν στην πλατφόρμα των Windows '98. Η ***ενοποίηση*** ετερογενών τεχνολογιών αποτελεί την προσπάθεια εξάλειψης των προβλημάτων συμβατότητας τα οποία πολλές φορές μειώνουν την αποδοτικότητα των συστημάτων.

Σε προηγούμενες αναφορές της παρούσας μελέτης έχει γίνει γνωστό ότι το Future Internet προβλέπεται πως θα συγκεντρώσει μεγάλο εύρος τεχνολογιών οι οποίες θα πρέπει να είναι σε θέση να συνεργαστούν μεταξύ τους. Η ανάπτυξη της δυνατότητας της SOVP να συγκεντρώνει ετερογενείς τεχνολογίες στο πλαίσιο εργασιών της, αποτελεί, ίσως, το σημαντικότερο χαρακτηριστικό του συστήματος αυτού. Η προσπάθειά μας λοιπόν, στοχεύει στην ανάπτυξη μιας πλατφόρμας ικανής να συγκεντρώσει έναν μεγάλο αριθμό συσκευών και κατ' επέκταση τεχνολογιών, από τον φυσικό κόσμο (Physical World) ενοποιώντας με αυτό τον τρόπο ετερογενείς τεχνολογίες πάνω σε ένα ενιαίο σύστημα λειτουργίας.

10.1.2 Διαλειτουργικότητα τεχνολογιών

Η ενοποίηση ετερογενών τεχνολογιών αποτελεί το πρώτο και ίσως το βασικότερο, χαρακτηριστικό που πρέπει να αποδοθεί στην SOVP. Ωστόσο δεν αρκεί μόνο η ενσωμάτωση ετερογενών τεχνολογιών στο σύστημά μας, αλλά κρίνεται απαραίτητη η δυνατότητα συνεργασίας αυτών των τεχνολογιών με σκοπό την εκτέλεση συγκεκριμένων διαδικασιών ικανών να ανταποκριθούν στις ανάγκες λειτουργίας του συστήματος. Η ***διαλειτουργικότητα*** των τεχνολογιών αποτελεί την δεύτερη πολύ σημαντική ποιοτική απαίτηση του συστήματος η οποία θα δώσει την δυνατότητα στο σύστημα να μπορεί να χρησιμοποιήσει τους πόρους που συγκεντρώνει από την διαδικασία της ενοποίησης. Δεν αρκεί λοιπόν, μόνο η ενσωμάτωση ετερογενών τεχνολογιών που μπορούν να λειτουργήσουν μεμονωμένα πάνω σε ένα κοινό σύστημα, αλλά αντιθέτως κρίνεται απαραίτητη η ικανότητα επικοινωνίας και συνεργασίας μεταξύ των τεχνολογιών αυτών με στόχο την αποδοτική λειτουργία του συστήματος.

10.1.3 Απόκρυψη πολυπλοκότητας

Σημαντικός παράγοντας προς αντιμετώπιση, όπου συγκαταλέγεται και στο σύνολο των ποιοτικών απαιτήσεων του συστήματος, είναι η διαδικασία απόκρυψης της πολυπλοκότητας των τεχνολογιών. Δηλαδή η απόδοση της ικανότητας στο σύστημα να μειώσει την πολυπλοκότητα των διαδικασιών, σύνδεσης, επικοινωνίας και λειτουργίας των ετερογενών τεχνολογιών που έχουν ενοποιηθεί στην πλατφόρμα. Προκειμένου να θεωρηθεί αποδοτικό ένα σύστημα θα πρέπει μεταξύ των άλλων, να είναι όσο το δυνατόν πιο εύχρηστο γίνετε. Δηλαδή δεν αρκεί η πλατφόρμα SOVP να παρέχει μια σειρά από

πρωτοποριακές τεχνολογίες στους χρήστες της, αλλά θα πρέπει να διευκολύνει την πρόσβαση τους σε αυτές με την **απόκρυψη της πολυπλοκότητας** των διαδικασιών.

10.1.4 Αυτοματοποίηση διαδικασιών

Η μείωση της εξάρτησης ενός συστήματος από τον ανθρώπινο παράγοντα, αποτελεί χαρακτηριστικό που, συνήθως, οδηγεί στην αύξηση της αποδοτικότητας του συστήματος ενώ σε πολλές περιπτώσεις επεκτείνει την βιωσιμότητα του συστήματος. Η εξάρτηση των συστημάτων από τον άνθρωπο μπορεί να γίνει πραγματικότητα μόνο μέσα από την διαδικασία της αυτοματοποίησης των διαδικασιών ενός συστήματος. Σαν ποιοτική απαίτηση της SOVP προκειμένου να αυξηθεί η πιθανότητα επιτυχίας του συστήματος αλλά και για να επεκταθεί η βιωσιμότητα του, θεωρούμε απαραίτητη την αυτοματοποίηση των διαδικασιών που θα πραγματοποιούνται. Με τον τρόπο αυτό επιδιώκουμε να αναπτύξουμε ένα σύστημα ικανό να αυτο-διαχειριστεί την κατάστασή του.

Ανακεφαλαιώνοντας την ενότητα καταγραφής των ποιοτικών απαιτήσεων της SOVP μπορούμε να αναφέρουμε ως συμπέρασμα ότι η προσπάθεια μας να αναπτύξουμε το συγκεκριμένο σύστημα, στοχεύει στην κάλυψη τριών βασικών αναγκών οι οποίες είναι: i) η ενοποίηση και η διαλειτουργικότητα ετερογενών τεχνολογιών ii) η απόκρυψη της πολυπλοκότητας των τεχνολογιών και iii) η αυτοματοποίηση των διαδικασιών. Στην συνέχεια θα αναφερθούμε στις απαιτήσεις ανάπτυξης και σχεδιασμού, παραθέτοντας ουσιαστικά μια συνοπτική προσέγγιση του **σχεδίου αρχιτεκτονικής** της SOVP μέσα από το οποίο επιδιώκουμε την κάλυψη των ποιοτικών απαιτήσεων που προαναφέραμε.

10.2 Απαιτήσεις σχεδιασμού και ανάπτυξης του συστήματος Service Oriented Virtualization Platform

Έχοντας περιγράψει το σύνολο των ποιοτικών απαιτήσεων του συστήματος της SOVP θα πρέπει να παραθέσουμε τις απαιτήσεις σχεδιασμού και υλοποίησης του. Σε γενικό επίπεδο οι απαιτήσεις συστήματος διαχωρίζονται σε δυο κατηγορίες, i) απαιτήσεις υλικού (**Hardware Requirements**) και ii) απαιτήσεις λογισμικού (**Software Requirements**).[(94)] Οι απαιτήσεις υλικού συμπεριλαμβάνουν οτιδήποτε σχετίζεται με τους πόρους υλικού που θα χρησιμοποιηθούν σε ένα σύστημα (CPU, RAM, κλπ.) . Οι απαιτήσεις λογισμικού αναφέρονται στον καθορισμό των αναγκαίων πόρων λογισμικού που θα πρέπει να χρησιμοποιηθούν στο σύστημα.

Στόχος μας είναι να αναφέρουμε τις χρησιμοποιούμενες διαδικασίες που θα μας βοηθήσουν στην ανάπτυξη συστήματος μας. Επικεντρωνόμαστε λοιπόν, στην καταγραφή απαιτήσεων λογισμικού που αφορούν την σχεδιαστική ανάπτυξη και υλοποίησης της πλατφόρμας.

---

[94] **System Requirements**, http://en.wikipedia.org/wiki/System_requirements, πρόσβαση: Μάιος 2010.

Υπάρχουν τρία δομικά συστατικά της αρχιτεκτονικής του συστήματος μας τα οποία είναι: i) η διαδικασία του Virtualization ii) οι γνωστικές υπηρεσίες (Cognitive Services) και iii) Service Oriented Computing τεχνολογία.

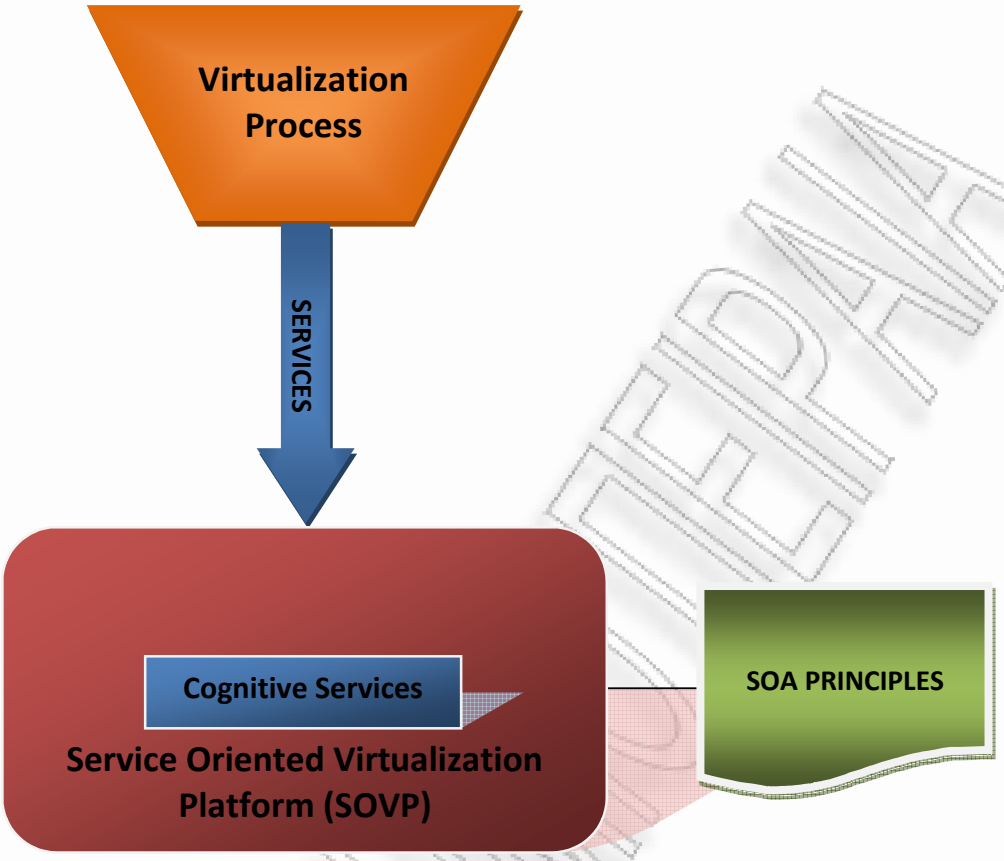
Το **Virtualization των πόρων** αποτελεί (όπως ήδη έχει αναφερθεί) μια διαδικασία απεικόνισης των πόρων υλικού και λογισμικού ως **υπηρεσίες**. Μέσω του Virtualization επιτυγχάνεται η δημιουργία ολοκληρωμένων λύσεων λογισμικού βασισμένου σε υπηρεσίες. Η λειτουργία ενός ολόκληρου συστήματος μπορεί να προσομοιωθεί σε ένα εικονικό περιβάλλον με την βοήθεια της εικονικοποίησης των πόρων. Με την δημιουργία καλά ορισμένων μονάδων λογισμικού οι οποίες συνθέτονται από ένα σύνολο συνεργαζόμενων υπηρεσιών μπορούμε να αναπτύξουμε αποδοτικά συστήματα τα οποία μπορούν εύκολα να προσομοιώσουν την λειτουργία τους σε ετερογενή περιβάλλοντα.

Η εφαρμογή των αρχών που διέπουν την γνωστική τεχνολογία αποτελεί σημαντικό συστατικό για την αρχιτεκτονική ανάπτυξη του συστήματος. Μέσω των **cognitive συστημάτων** επιτυγχάνουμε **την αυτοματοποίηση των διαδικασιών** του συστήματος. Τα **Self – X** χαρακτηριστικά της SOVP είναι εφικτό να αποδοθούν με την εφαρμογή **cognitive υπηρεσιών** οι οποίες οργανώνονται βάση της SOC. Οι υπηρεσίες που διαθέτει το σύστημα αλλά και οι υπηρεσίες που διαθέτουν τα στοιχεία (components) του συστήματος θα πρέπει να λειτουργούν **αυτόνομα** και **αυτοματοποιημένα**, χωρίς την παρέμβαση του ανθρώπινου παράγοντα. Προκειμένου λοιπόν, να εισάγουμε το σημαντικό στοιχείο της αυτοματοποίησης των διαδικασιών, επιλέγουμε την εισαγωγή γνωστικών υπηρεσιών στο σύστημα οι οποίες θα οργανώνονται σύμφωνα με τις αρχές της SOA.

Ολοκληρώνοντας, όπως ήδη έχουμε αναφέρει στο κεφάλαιο 5 (*παρ. 5.3*), **η Service Oriented Computing – SOC τεχνολογία** ορίζει έναν νέο τρόπο ανάπτυξης συστημάτων βασισμένων στην SOA τα οποία εστιάζουν, κυρίως, στην **ετερογένεια των πόρων** που θα συγκεντρώνουν.[(95)] Οι πόροι παρουσιάζονται ως υπηρεσίες που οργανώνονται σύμφωνα με την SOA και μπορούν να συνδυαστούν με την αναζήτηση και κλίση διαθέσιμων υπηρεσιών από το δίκτυο για την εκτέλεση συγκεκριμένων εργασιών. Έτσι οι υπηρεσίες που προκύπτουν από την διαδικασία του Virtualization, θα μπορεί να τις διαχειριστεί το σύστημα με την εφαρμογή των αρχών της Service Oriented αρχιτεκτονικής στο Cognitive σύστημα.

Συμπερασματικά θα μπορούσαμε να αναφέρουμε ότι υπάρχουν τρία βασικά συστατικά για την επιτυχημένη σύνθεση του συστήματος μας. Η ανάπτυξη ενός ***Service Oriented συστήματος*** το οποίο θα ενσωματώνει ***γνωστικές υπηρεσίες*** που προέρχονται από την ***διαδικασία του Virtualization***. Η διάρθρωση του συστήματος ως κατανεμημένο σύστημα θα αποδώσει το χαρακτηριστικό της αυτονομίας, ενώ η εφαρμογή της SOC σε cognitive υπηρεσίες, θα οδηγήσει στην ανάπτυξη ενός καλά ορισμένου συστήματος ικανό να αναπροσαρμόσει την λειτουργία του σύμφωνα με τις τρέχουσες ανάγκες. Το σχήμα 9, παρουσιάζει την προσέγγιση που αναφέρουμε.

[95] Βλ. παρ. 49, **σελ.** 24.



Σχήμα 9: "Απαιτήσεις σχεδιασμού και ανάπτυξης SOVP"

✓ **Κεφάλαιο 11**: «*Παρουσίαση αρχιτεκτονικής της Service Oriented Virtualization Platform – (SOVP)*»

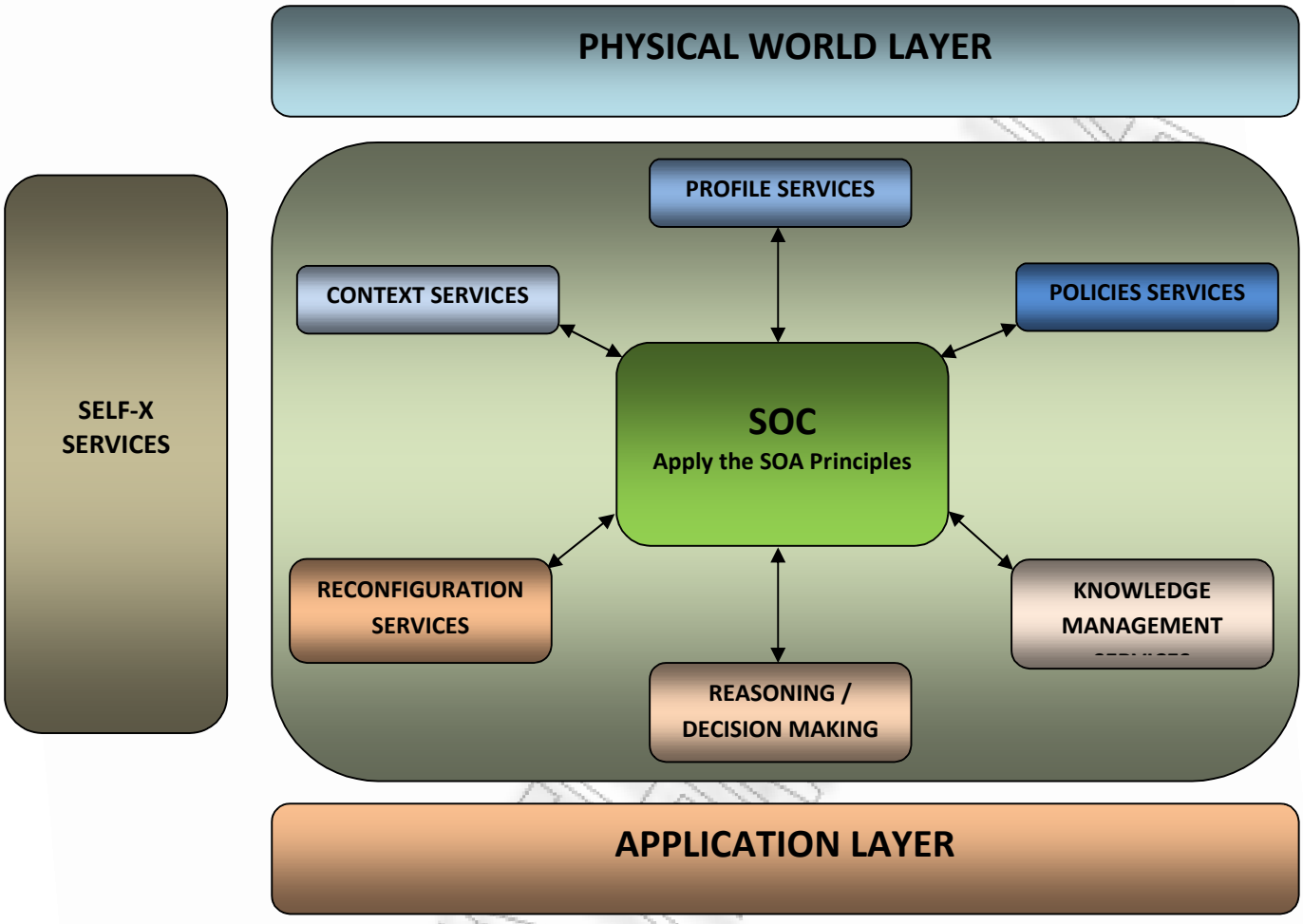
Η αρχιτεκτονική σχεδίαση της SOVP στοχεύει στην ανάπτυξη μιας πλατφόρμας η δομή της οποία θα ανταποκρίνεται στις σύγχρονες απαιτήσεις των μελλοντικών δικτύων (**Future Networks - FN**) και κατ' επέκταση του μελλοντικού διαδικτύου (**Future Internet - FI**). Η πλατφόρμα αυτή θα βασίζει την λειτουργία της στις υπηρεσίες που θα συγκεντρώνει. Οι υπηρεσίες θα οργανώνονται σύμφωνα με τις αρχές της **Service Oriented** αρχιτεκτονικής προκειμένου να αντιμετωπιστούν ζητήματα που αφορούν την οργάνωση ετερογενών τεχνολογιών. Επιπλέον η διαχείριση της ετερογένειας των τεχνολογιών αλλά και η πολυπλοκότητα τους θα μπορεί να πραγματοποιηθεί με την εισαγωγή της διαδικασίας του Virtualization. Με το **Virtualization** των συστημάτων έχουμε την δυνατότητα να απεικονίσουμε ολοκληρωμένες λειτουργίες υλικού και λογισμικού με την μορφή υπηρεσιών.

Οι υπηρεσίες που θα ενσωματώνονται στην πλατφόρμα μέσα από τα πρότυπα διαχείρισης της SOA (Service Oriented Principles) θα μπορούν να συνδυαστούν με άλλες υπηρεσίες της πλατφόρμας δημιουργώντας ολοκληρωμένες λύσεις λογισμικού για την αντιμετώπιση των απαιτήσεων του συστήματος (*πχ: διαδικασίες εντοπισμού συμφόρησης σε ένα δίκτυο και εκτέλεση λειτουργιών αποσυμφόρησης του*). Με λίγα λόγια, το σύστημα της SOVP θα συγκεντρώνει γνωστικές υπηρεσίες οι οποίες θα οργανώνονται βάση της SOA. Ουσιαστικά, θα έχουμε ένα Service Oriented σύστημα, το οποίο θα ενοποιεί και θα καθιστά δυνατή την συνεργασία γνωστικών υπηρεσιών.

Προκειμένου να επιτύχουμε την πραγματοποίηση της παραπάνω λειτουργίας θα πρέπει αρχικά να δημιουργήσουμε ένα σύνολο διαφορετικών κατηγοριών οι οποίες θα συμπεριλαμβάνουν όλες τις γνωστικές υπηρεσίες που θα μπορούν να ενσωματωθούν στην πλατφόρμα. Κάθε υπηρεσία θα εντάσσεται στην αντίστοιχη κατηγορία ανάλογα με την λειτουργία που προάγει. Οι κατηγορίες υπηρεσιών θα αναφέρονται στο εξής ως δομές – **blocks υπηρεσιών** και σύμφωνα με την αρχική προσέγγιση που αναφέρεται στη μελέτη "*Virtualization Platform for the introduction of cognitive systems in the Future Internet*" θα υπάρχουν **7** διαφορετικά **block γνωστικών υπηρεσιών** τα οποία είναι:

- Profile Services
- Context services
- Reasoning / Decision Making Services
- Reconfiguration Services
- Policies Services
- Knowledge Management Services
- Self – x Services

Το σχήμα 10 στη συνέχεια παρουσιάζει τον τρόπο δόμησης των διαφορετικών blocks υπηρεσιών στην SOVP.



Σχήμα 10: "Η δομή της Service Oriented Virtualization Platform"

Μελετώντας το σχήμα 10 παρατηρούμε ότι, το σύνολο υπηρεσιών που εντοπίζεται στην SOVP συγκεντρώνεται γύρω από το block με τίτλο SOC. Το συγκεκριμένο block απεικονίζει την οργανωτική αρχή της λειτουργίας του συστήματος μας. Αποτελεί ουσιαστικά την εφαρμογή του συσχετισμού SOC – Cognition Cycle, που αναφέραμε παραπάνω (*κεφ. 8*) όπου μια σειρά λειτουργιών που προτείνεται από τον SOC Research roadmap, φαίνεται να εφαρμόζεται και να εξελίσσεται από τον κύκλο γνώσης των συστημάτων, σύμφωνα με τον τρόπο που τον παρουσιάζουμε στην συγκεκριμένη μελέτη.

Διαπιστώνουμε λοιπόν ότι, η πλατφόρμα θα διαμορφώνεται σύμφωνα με ένα σύνολο αρχών που προέρχονται από την Service Oriented αρχιτεκτονική, εφαρμόζοντας μια σειρά λειτουργιών από την SOC τεχνολογία για την οργάνωση και διαχείριση των γνωστικών υπηρεσιών. Η περιγραφή της αρχιτεκτονικής της SOVP γίνεται πιο σαφής στο κεφάλαιο 12, που ακολουθεί.

✓ **Κεφάλαιο 12**: «*Ανάλυση αρχιτεκτονικής της SOVP*»

Στα προηγούμενα κεφάλαια, του μέρους Β΄, επιχειρήσαμε να παρουσιάσουμε μια ολοκληρωμένη εικόνα σχετικά με το σύστημα που σκοπεύουμε να υλοποιήσουμε. Αρχικά, στο κεφάλαιο 10, αναφερθήκαμε στις απαιτήσεις του συστήματος και εισάγοντας στοιχεία από την θεωρία της ανάλυσης συστημάτων (System analysis) προσπαθήσαμε να δώσουμε μια πλήρη περιγραφή των ποιοτικών απαιτήσεων αλλά και των απαιτήσεων σχεδιασμού και ανάπτυξης που προκύπτουν στο σύστημα μας. Στο κεφάλαιο 11 παρουσιάσαμε την αρχιτεκτονική της πλατφόρμας που θα υλοποιήσουμε, παραθέτοντας με ακρίβεια τα δομικά της μέρη.

Στο παρόν κεφάλαιο θα πραγματοποιήσουμε την ανάλυση της αρχιτεκτονικής της πλατφόρμας. Η πλατφόρμα SOVP αποτελεί ένα Service Oriented σύστημα το οποίο συγκεντρώνει γνωστικές υπηρεσίες. Οι γνωστικές υπηρεσίες έχουν παρουσιαστεί παραπάνω ως block υπηρεσιών όπου κάθε block συγκεντρώνει ένα συγκεκριμένο τύπο υπηρεσιών (πχ: Profile Service, Context Services). Οι υπηρεσίες που εντάσσονται σε κάθε block, προέρχονται από τα διαφορετικά στοιχεία (components) που ενσωματώνονται στην SOVP αφού προηγουμένως έχει εκτελεστεί η διαδικασία του Virtualization για κάθε ένα απ' αυτά. Κάθε component μπορεί να προσφέρει υπηρεσίες που εντάσσονται σε ορισμένα ή ακόμη και σε όλα τα block υπηρεσιών της SOVP. Ο συνδυασμός των γνωστικών υπηρεσιών της SOVP δημιουργεί γνωστικές λειτουργίες (cognitive functions) που διαθέτει η πλατφόρμα.

Στη συνέχεια θα επιχειρήσουμε την κατηγοριοποίηση των γνωστικών υπηρεσιών της SOVP στις τέσσερεις φάσεις του κύκλου γνώσης.

12.1 Service Oriented Virtualization Platform – Κύκλος Γνώσης και Γνωστικές Υπηρεσίες

Η SOVP αποτελεί ένα σύστημα προσανατολισμένο σε υπηρεσίες (Service Oriented System) το οποίο συγκεντρώνει γνωστικές υπηρεσίες. Με την βοήθεια των SOA Principles πραγματοποιείται ο συνδυασμός των γνωστικών υπηρεσιών όπου ως αποτέλεσμα του έχουμε τις γνωστικές λειτουργίες του συστήματος. Οι γνωστικές λειτουργίες της πλατφόρμας που προέρχονται από τον συνδυασμό των γνωστικών υπηρεσιών που συγκεντρώνει, μπορούν να μας βοηθήσουν στην κατηγοριοποίηση των γνωστικών υπηρεσιών, στις τέσσερεις φάσεις του κύκλου γνώσης.

12.1.1 Παρατήρηση – Observe

Η φάση της παρατήρησης στον κύκλο γνώσης περιλαμβάνει το σύνολο λειτουργιών που εκτελεί την διαδικασία της λήψης στοιχείων από το εξωτερικό περιβάλλον και αφού τα προσαρμόσει στην κατάλληλη διαμόρφωση τροφοδοτεί το σύστημα με τα δεδομένα που αφορούν τα στοιχεία του εξωτερικού περιβάλλοντος. Υπενθυμίζουμε ότι ως στοιχεία του εξωτερικού περιβάλλοντος θεωρούνται οι συσκευές, οι δικτυακές τεχνολογίες, το λογισμικό αλλά και οι υπηρεσίες. Σύμφωνα με την προτεινόμενη αρχιτεκτονική της SOVP τα block υπηρεσιών που εκτελούν λειτουργίες παρατήρησης είναι: οι **Context services** και οι **Profile services**. Το σύστημα λαμβάνει τις απαραίτητες πληροφορίες με την βοήθεια των παραπάνω υπηρεσιών και αφού διαμορφώσει τις πληροφορίες αυτές σε κατανοητή μορφή, τροφοδοτεί τις υπηρεσίες που εντάσσονται στην φάση της ανάλυσης.

12.1.2 Ανάλυση – Analyse

Η φάση της ανάλυσης λαμβάνει τα παραδοτέα της φάσης της παρατήρησης και σε συνδυασμό με στοιχεία που αφορούν την πολιτική των πληροφοριών, αναλύει και δημιουργεί ένα ολοκληρωμένο μοντέλο που αναφέρεται στα χαρακτηριστικά του κάθε εξωτερικού στοιχείου που έρχεται σε επαφή με την πλατφόρμα. Αφού πραγματοποιήσει την ολοκλήρωση του μοντέλου χαρακτηριστικών για κάθε στοιχείο, εντάσσει τις πληροφορίες αυτές στην **βάση γνώσεων και μάθησης** που διαθέτει το σύστημα και στη συνέχεια μπορεί να τις χρησιμοποιήσει για την λήψη αποφάσεων, για την διαχείριση καταστάσεων και γενικότερα για την διαμόρφωση και αναδιαμόρφωση της λειτουργίας του συστήματος. Στην φάση της ανάλυσης περιλαμβάνονται οι κατηγορίες υπηρεσιών: **Knowledge management services** και **Policies services**. Με την ολοκλήρωση της φάσης της ανάλυσης το σύστημα διαθέτει το σύνολο πληροφοριών που είναι απαραίτητο για τον σχεδιασμό στρατηγικών λειτουργίας.

12.1.3 Σχεδιασμός – Plan

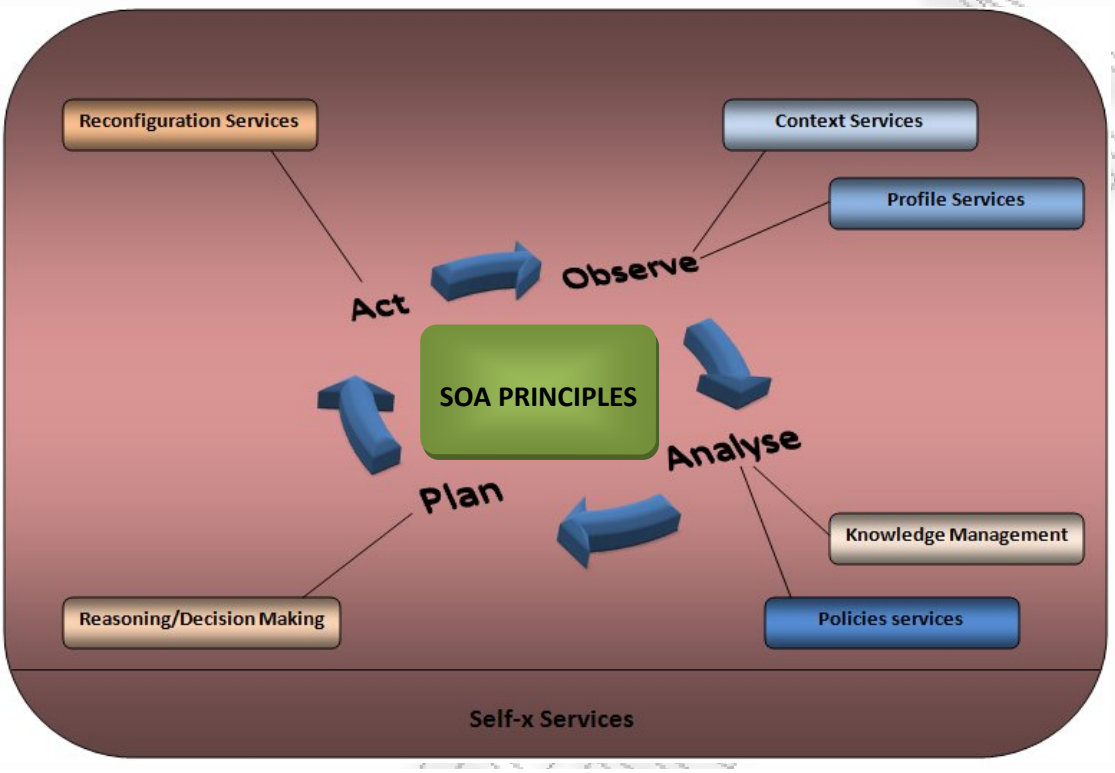
Ο σχεδιασμός αποτελεί την φάση του κύκλου γνώσης όπου το σύστημα έχοντας λάβει νέες πληροφορίες για εξωγενή στοιχεία και συνδυάζοντας υπάρχουσες πληροφορίες από την βάση αποθήκευσης πληροφοριών, πραγματοποιεί τον σχεδιασμό εναλλακτικών λειτουργιών προκειμένου το σύστημα να ανταπεξέλθει στις απαιτήσεις που προκύπτουν. Αφού ολοκληρώσει τον σχεδιασμό εναλλακτικών λύσεων, εάν οι λύσεις είναι περισσότερες από μία θα πρέπει να επιλέξει την βέλτιστη λύση η οποία θα πραγματοποιήσει την κάλυψη των αναγκών. Στην SOVP η λειτουργία αυτή μπορεί να πραγματοποιηθεί με την εφαρμογή των υπηρεσιών που περιλαμβάνει η κατηγορία **Reasoning / Decision Making Services**. Το σύστημα εφόσον επιλέξει την λύση που θεωρεί ως την καλύτερη μεταξύ των άλλων, θα πρέπει να προχωρήσει στην εφαρμογή της. Η τελευταία φάση του κύκλου γνώσης αναλαμβάνει την εκτέλεση της διαδικασία αυτής.

12.1.4 Εκτέλεση – Act

Η φάση της εκτέλεσης αποτελεί το τελευταίο τμήμα που ολοκληρώνει τον κύκλο γνώσης της πλατφόρμα. Το σύστημα εφόσον έχει πραγματοποιήσει μια σειρά από διαδικασίες αναγνώρισης, ανάλυσης, συνδυασμού πληροφοριών και λήψης αποφάσεων για την λειτουργία του, θα πρέπει να εφαρμόσει το σενάριο λειτουργίας που επέλεξε. Η SOVP διαθέτει το block υπηρεσιών **Reconfiguration Services** το οποίο μέσω των υπηρεσιών που διαθέτει είναι σε θέση να πραγματοποιήσει την αναδιαμόρφωση της λειτουργίας της πλατφόρμας. Έτσι η πλατφόρμα μπορεί να πραγματοποιήσει την αποδοχή νέων τεχνολογιών και την ενσωμάτωση αυτών στο σύστημα λειτουργίας της.

Όπως ήδη έχει αναφερθεί, στο κεφάλαιο 11, η SOVP περιλαμβάνει και Self-x Services οι οποίες συμβάλουν στην ανάπτυξη λειτουργιών αυτονομίας και αυτοδιαχείρισης του συστήματος. Ολοκληρώνοντας, σημαντικό είναι να επισημάνουμε ότι όλες οι κατηγορίες υπηρεσιών της SOVP τοποθετούνται γύρω από τον πυρήνα το συστήματος το οποίο οργανώνεται βάση των Service Oriented

αρχών. Η εικόνα 19 παρακάτω, απεικονίζει την κατηγοριοποίηση των block υπηρεσιών της SOVP στις φάσεις του κύκλου γνώσης.



Εικόνα 19: "Διάγραμμα του κύκλου γνώσης σε συνδυασμό με τα blocks υπηρεσιών της SOVP "

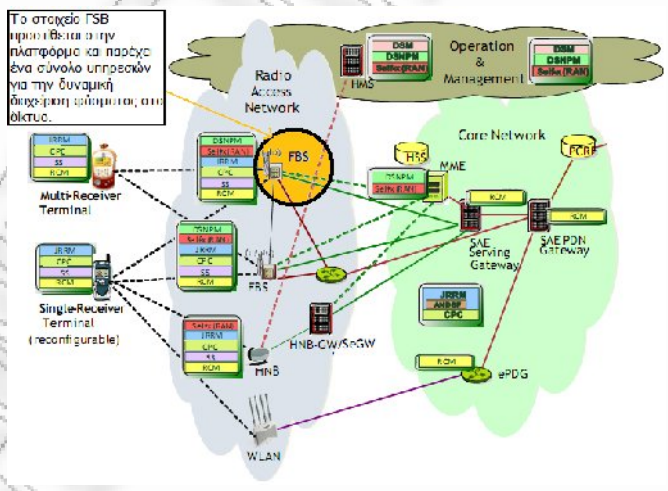
Μέσω λοιπόν, της παρούσας μελέτης στοχεύουμε στη δημιουργία ενός αυτόνομου συστήματος το οποίο δομείται αρχιτεκτονικά σύμφωνα με τις αρχές της Service Oriented αρχιτεκτονικής, και συγκεντρώνει διαφορετικές γνωστικές υπηρεσίες οι οποίες οργανώνονται βάσει των αρχών που διέπουν την Service Oriented Computing τεχνολογία. Επιπλέον σημαντικό είναι να αναφερθεί ότι οι αρχές της Service Oriented αρχιτεκτονικής θα προσαρμοστούν σε ένα μοντέλο αποκεντρωτικής αρχιτεκτονικής συστήματος το οποίο θα συγκεντρώνει ανεξάρτητες λειτουργικές μονάδες οι οποίες θα παρουσιάζονται μέσω του virtualization με την μορφή διαφορετικών γνωστικών υπηρεσιών.

Συμπερασματικά, στοχεύουμε στην ανάπτυξη ενός κατανεμημένου συστήματος που θα βασίζεται στις υπηρεσίες, η οργάνωση των οποίων θα πραγματοποιείται μέσω της SOA. Το κατανεμημένο σύστημα θα παρουσιάζεται με την μορφή μιας πλατφόρμας. Δικαιολογημένα λοιπόν, ευσταθεί η ονομασία του συστήματος μας ως Service Oriented Virtualization Platform. Στη ενότητα 12.2 θα πραγματοποιήσουμε την ανάλυση των blocks υπηρεσιών που συγκεντρώνει η SOVP.

12.2 Εκτενής ανάλυση των blocks υπηρεσιών που συγκεντρώνει η SOVP

Σύμφωνα με το σχέδιο που παρουσιάσαμε στην προηγούμενη ενότητα, η αρχιτεκτονική της πλατφόρμας SOVP συνδυάζει την διαδικασία της εικονικοποίησης (Virtualization) που παράγει γνωστικές υπηρεσίες, με ένα σύνολο αρχών της Service Oriented αρχιτεκτονικής ώστε να προκύψει ένα σύστημα δυναμικά διαρθρωνόμενο (Reconfigurable) και αυτόνομο (Autonomic). Με την εικονικοποίηση των στοιχείων (συσκευές, λογισμικό, δικτυακές τεχνολογίες, κλπ.) έχουμε την απεικόνιση τους ως ένα σύνολο υπηρεσιών οι οποίες μπορούν να ενοποιηθούν με την χρήση των cognitive schemes. Τα cognitive schemes προάγουν ένα σύνολο λειτουργιών αυτοματοποίηση των διαδικασιών και δημιουργούνται σύμφωνα με τις αρχές του Service Composition για την SOVP.

Το σύνολο των νέων στοιχείων που μπορεί να συναντήσουμε στην SOVP παρέχουν ένα σύνολο υπηρεσιών οι οποίες μπορούν να χρησιμοποιηθούν από την πλατφόρμα, σε συνδυασμό με άλλες υπηρεσίες, για την ανάπτυξη ολοκληρωμένων λύσεων λογισμικού. Ένα παράδειγμα, αυτού που αναφέρουμε, αποτελεί η εικόνα 20 παρακάτω, η οποία παρουσιάζει την αρχιτεκτονική ενός γνωστικού συστήματος διαχείρισης πρόσβασης σε δίκτυα ραδιοεπικοινωνιών. Στην προσπάθεια μας να επεξηγήσουμε την προσφορά υπηρεσιών από στοιχεία που εισέρχονται στο σύστημα, φανταζόμαστε το κομμάτι που περιλαμβάνεται στο "Radio Access Network" καθώς αυτό αναπτύσσεται με την εισαγωγή νέων στοιχείων και τελικά καταλήγει να πάρει την μορφή που έχει στην εικόνα 20. Το στοιχείο FSB προστίθεται στην πλατφόρμα και παρέχει ένα σύνολο υπηρεσιών για την δυναμική διαχείριση φάσματος στο δίκτυο. Τις υπηρεσίες αυτές τις χρησιμοποιούν με την σειρά τους άλλα στοιχεία τα οποία επιδιώκουν πρόσβαση στο δίκτυο, αλλά και η ίδια η πλατφόρμα προκειμένου να διαχειριστεί την δυναμική εισαγωγή στοιχείων.



Εικόνα 20:"Αρχιτεκτονική γνωστικού συστήματος με την προσθήκη νέων στοιχείων" [96]

Γενικότερα μια Service Oriented πλατφόρμα όπως η SOVP, που στηρίζει την λειτουργία της στις υπηρεσίες, μπορεί να χρησιμοποιεί τις υπηρεσίες ώστε να επεκτείνει τις λειτουργίες της, αλλά και να διαχειριστεί διάφορες καταστάσεις που μπορεί να προκύψουν. Βασικός στόχος της SOVP είναι η δυναμική ανάπτυξη, σύνθεση και ανακάλυψη υπηρεσιών οι οποίες θα επιτρέπουν στις συσκευές και τις

[96] E3 Project FA/SA, "**Architecture of Cognitive Systems**", Ιανουάριος, 2010, σελ. 7.

υπόλοιπες υπηρεσίες της πλατφόρμας, να διαφημίσουν και να περιγράψουν τις λειτουργίες που προσφέρουν.

Η περιγραφή της αρχιτεκτονικής της SOVP που πραγματοποιήσαμε στο κεφάλαιο 11, παρουσιάζει ένα σύνολο από δομές υπηρεσιών που θα οργανώνονται στην πλατφόρμα. Οι δομές αυτές σχεδιάστηκαν με **στόχο την αυτονόμηση των διαδικασιών στην πλατφόρμα**. Το σύνολο των blocks υπηρεσιών αποτελείται από: Knowledge Management Services, Reasoning/Decision Making Services, Profile Services, Policies Services, Context services, Reconfiguration Services. Όλα τα παραπάνω blocks υπηρεσιών συγκεντρώνονται σε ένα ενιαίο block που αποτελεί ουσιαστικά τις Self – x Services που διαθέτει η πλατφόρμα. Ουσιαστικά μέσω των παραπάνω δομών υπηρεσιών επιτυγχάνουμε την απόδοση αυτονομίας στην πλατφόρμα με λειτουργίες όπως: self-management, self-optimization, self-configuration, self-maintenance, self-planning και self-healing. Στις επόμενες υποενότητες παρουσιάζουμε την περιγραφή του κάθε block υπηρεσιών.

12.2.1 Profile Services

Οι υπηρεσίες προφίλ παρέχουν στην SOVP την δυνατότητα ανάκτησης πληροφοριών για το προφίλ των στοιχείων που εισέρχονται στην πλατφόρμα. Λαμβάνουν ως είσοδο δεδομένα που αφορούν την κατάσταση του προφίλ μιας συσκευής, μιας υπηρεσίας ή ενός εξωγενούς δικτύου, από τα οποία εξάγουν πληροφορίες που βοηθούν στην διαχείριση των στοιχείων από την πλατφόρμα. Για παράδειγμα στην περίπτωση που επιχειρεί να εισέλθει στην πλατφόρμα μια ασύρματη δικτυακή συσκευή, θα πρέπει να λάβουμε σχετικές πληροφορίες με το Radio Access Technology (RAT) της συσκευής, ώστε να ελέγξουμε αν μπορεί να προσαρμοστεί στο πλαίσιο λειτουργίας της πλατφόρμας. Στην περίπτωση που αντιμετωπίσουμε πρόβλημα, η πλατφόρμα ενεργοποιεί τους μηχανισμούς Reasoning / Decision Making προκειμένου να αναπροσαρμόσει την λειτουργία της έτσι ώστε να συμπεριλάβει την νέα συσκευή. Επίσης τα δεδομένα που προκύπτουν από κάθε προφίλ, επεξεργάζονται από τους μηχανισμούς του knowledge management προκειμένου να εμπλουτίσει την αποθήκη γνώσης η πλατφόρμα με τις σχετικές πληροφορίες οι οποίες μπορούν να αξιοποιηθούν μελλοντικά για την αντιμετώπιση μιας ανάλογης κατάστασης. Γενικότερα οι πληροφορίες που μπορούμε να λάβουμε από τα δεδομένα του προφίλ ενός στοιχείου μπορεί να αφορούν ποικίλα θέματα όπως: επίπεδα ποιότητας υπηρεσιών, τα επίπεδα μετάδοσης, η τοπολογία ενός δικτύου, κλπ.

12.2.2 Context services

Οι υπηρεσίες πλαισίου περιλαμβάνουν μηχανισμούς οι οποίοι βοηθούν την πλατφόρμα να αντιληφθεί την ισχύουσα κατάσταση μιας συσκευής χρήστη ή δικτύου. Το πλαίσιο λειτουργίας κάθε συσκευής μπορεί να περιλαμβάνει πληροφορίες σχετικές με: την τοποθεσία της συσκευή, τους κανονισμούς πρόσβασης που διέπουν την συσκευή στο περιβάλλον που ανήκει, την κατάσταση της συσκευής (επίπεδα ισχύς, τεχνολογίες πρόσβασης, λογισμικό, κλπ.), τους συσχετισμούς της συσκευής στο περιβάλλον που ανήκει (πχ: συνεργασία με άλλες συσκευές), κλπ. Γενικότερα οι μηχανισμοί που προάγουν οι context services συγκεντρώνουν πληροφορίες σχετικά με το κομμάτι της λειτουργικής υπόστασης μιας συσκευής. Οι πληροφορίες που λαμβάνονται, στην συνέχεια μπορούν να

τροφοδοτήσουν τις υπηρεσίες του knowledge management, ώστε να επεκτείνουν την γνώση του συστήματος.

12.2.3 Reasoning / Decision Making Services

Οι υπηρεσίες συλλογιστικής και λήψης απόφασης, περιλαμβάνουν τους μηχανισμούς που είναι υπεύθυνοι για τον σχεδιασμό και την επιλογή εναλλακτικών σχεδίων λειτουργίας της πλατφόρμας. Η πλατφόρμα κάθε φορά ακολουθεί ένα συγκεκριμένο σχέδιο λειτουργίας προκειμένου να ανταποκριθεί στην τρέχουσα κατάσταση. Τα σχέδια λειτουργίας πολλές φορές μπορεί να είναι περισσότερα από ένα. Για τον λόγο αυτό θα πρέπει να επιλεγεί εκείνο που έχει την καλύτερη απόδοση. Οι υπηρεσίες αυτές μπορούν να συλλέγουν στοιχεία είτε από μια τρέχουσα κατάσταση είτε αξιοποιώντας υπάρχουσες πληροφορίες από την συγκεντρωμένη γνώση του συστήματος. Με την χρήση των συγκεκριμένων μηχανισμών το σύστημα είναι σε θέση να αντιμετωπίσει διάφορες καταστάσεις που υπό διαφορετικές συνθήκες θα ήταν απαραίτητη η παρέμβαση του ανθρώπινου παράγοντα.

12.2.4 Reconfiguration Services

Οι υπηρεσίες αναδιάρθρωσης αποτελούν ένα από τα σημαντικότερα block υπηρεσιών για την SOVP. Μέσω των υπηρεσιών αυτών η πλατφόρμα αποκτά την ικανότητα δημιουργίας κατάλληλων διεπαφών επικοινωνίας μεταξύ της ίδιας και των νεοεισερχόμενων στοιχείων σε αυτή. Οι πληροφορίες που έχουν προκύψει από τους μηχανισμούς των profile, context, policies και decision making υπηρεσιών έχουν αποδώσει στο σύστημα την απαραίτητη ενημέρωση σχετικά με την λειτουργική φύση της συσκευής (τεχνολογίες, RAT, QoS, κλπ.). Η πλατφόρμα αξιοποιώντας τις πληροφορίες αυτές εφαρμόζει τους μηχανισμούς αναδιάρθρωσης ώστε να εγκατασταθεί μια αποδοτική σύνδεση με την συσκευή μέσω της οποία η συσκευή θα μπορέσει να γίνει μέρος του συστήματος. Οι reconfiguration services ουσιαστικά, πραγματοποιούν την ανεξαρτητοποίηση των τεχνολογιών από συγκεκριμένα μοντέλα λειτουργίας και προωθούν ένα πλαίσιο επικοινωνίας σύμφωνα με το οποίο η πλατφόρμα και τα εξωγενή στοιχεία θα μπορούν να συνεργαστούν. Σημαντικό είναι να επισημάνουμε ότι οι υπηρεσίες αναδιάρθρωσης μέσα από την λειτουργία την οποία προάγουν, εφαρμόζουν την τελική απόφαση που προέκυψε από τους μηχανισμούς του decision making.

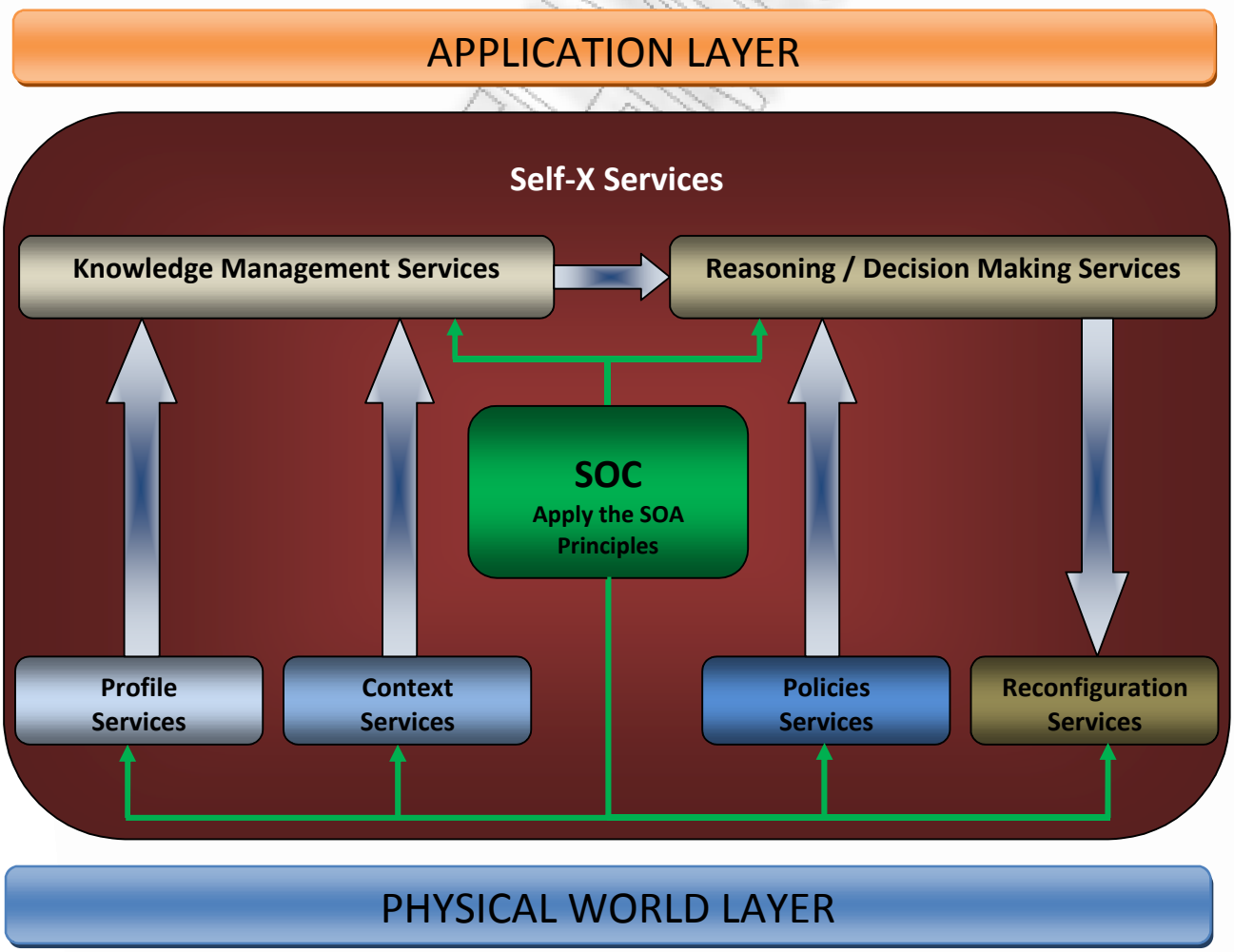
12.2.5 Policies Services

Οι υπηρεσίες πολιτικής συγκεντρώνουν πληροφορίες που αφορούν κυρίως τους κανόνες που διέπουν κάποια συσκευή ή γενικότερα ένα στοιχείο που εισέρχεται στην πλατφόρμα. Χρησιμοποιούνται προκειμένου να λάβουμε λεπτομέρειες που σχετίζονται με τα δεδομένα που ανακτήσαμε από τις υπηρεσίες προφίλ και πλαισίου. Για παράδειγμα οι υπηρεσίες πολιτικής μπορούν να ανακτήσουν πληροφορίες σχετικά με τα δικαιώματα πρόσβασης μια δικτυακής συσκευής σε ένα δίκτυο ή πληροφορίες σχετικά με την περιγραφή περιορισμών στην συνεργασία μεταξύ των δικτυακών συσκευών. Σε γενικό επίπεδο τα παραδοτέα των policies services αποτελούν σημαντικές πληροφορίες για τις decision making services οι οποίες θα διαμορφώσουν την λειτουργία της πλατφόρμας σύμφωνα με τους κανονισμούς που διέπουν το κάθε στοιχείο της.

12.2.6 Knowledge Management Services

Οι υπηρεσίες διαχείρισης γνώσης μέσα από το σύνολο λειτουργιών που προάγουν, στοχεύουν στην ανάπτυξη γνώσης η οποία θα είναι διαθέσιμη προς την πλατφόρμα και αξιοποιήσιμη από αυτή. Η γνώση θα συγκροτείται από πληροφορίες που προκύπτουν από την επεξεργασία των δεδομένων που σχετίζονται με τα προφίλ, τα πλαίσια λειτουργίας και τις πολιτικές αρχές των στοιχείων που συναντώνται στην πλατφόρμα. Η γνώση που προκύπτει, αξιοποιείται από τις υπηρεσίες λήψης απόφασης (Decision Making Services) οι οποίες κάθε φορά τροφοδοτούνται με δεδομένα από την αποθήκη γνώσης του συστήματος. Μέσα από τις υπηρεσίες γνώσης στοχεύουμε στην ανάπτυξη ενός έξυπνου και αποδοτικού συστήματος το οποίο ενισχύει την αυτονομία του μέσα από την διαρκώς αυξανόμενη γνώση που συγκεντρώνει και αξιοποιεί με την χρήση κατάλληλων μηχανισμών.

Έχοντας ολοκληρώσει την περιγραφή του κάθε block υπηρεσιών, θα παρουσιάσουμε στην συνέχεια το τελικό σχέδιο που απεικονίζει την αρχιτεκτονική της πλατφόρμας SOVP, όπως αυτή παρουσιάζεται μέσα από την συγκεκριμένη μελέτη. Το σχήμα 11 αποτελεί ουσιαστικά μια πιο λεπτομερή παρουσίαση του σχήματος 10. Συγκεκριμένα επιχειρούμε την παρουσίαση της αρχιτεκτονικής, οργανώνοντας το σχέδιο ως ένα σύνολο διαφορετικών δομών συνεργασίας.



Σχήμα 11: "Η αρχιτεκτονική της Service Oriented Virtualization Platform"

✓ *Ανακεφαλαιώνοντας*

Στο μέρος Β΄ πραγματοποιήσαμε μια ολοκληρωμένη αναφορά σχετικά με την αρχιτεκτονική της SOVP. Αρχικά, με την βοήθεια της θεωρίας από την ανάλυση συστημάτων (System Analysis Theory), ορίσαμε τις ποιοτικές απαιτήσεις που επιχειρεί να καλύψει η πλατφόρμα ενώ στη συνέχεια περιγράψαμε τις απαιτήσεις σχεδιασμού και ανάπτυξης του συστήματος. Έχοντας ολοκληρωμένη εικόνα μέσα από την ανάλυση του συστήματος προχωρήσαμε στην παρουσίαση του σχεδίου αρχιτεκτονικής και ολοκληρώσαμε το μέρος Β΄ με την ανάλυση του προτεινόμενου σχεδίου.

Στο επόμενο μέρος της μελέτης (Μέρος Γ΄), θα αναφερθούμε στο θέμα της υλοποίησης της SOVP. Πιο συγκεκριμένα θα αναφερθούμε αρχικά στις τεχνολογίες που επιλέξαμε για την ανάπτυξη του σχεδίου αρχιτεκτονικής που παρουσιάσαμε, κάνοντας μια επισκόπηση κάθε τεχνολογίας. Στη συνέχεια θα πραγματοποιήσουμε την αναλυτική περιγραφή της υλοποίησης, κάνοντας αναφορές στο τρόπο ανάπτυξης του συστήματος, αλλά και στον τρόπο υλοποίησης των χαρακτηριστικών του.

Μέρος Γ΄

«Υλοποίηση της Service Oriented Virtualization Platform»

✓ **Κεφάλαιο 13:** «Επισκόπηση τεχνολογιών υλοποίησης της SOVP»

Έχοντας ορίσει προηγουμένως την αρχιτεκτονική της SOVP, στο παρόν κεφάλαιο θα αναφερθούμε στην διαδικασία υλοποίησης της αρχιτεκτονικής. Η ανάπτυξη συστημάτων στην πληροφοριακή τεχνολογία, μπορεί να πραγματοποιηθεί με ποικίλους τρόπους αλλά πάντοτε βασίζεται σε ένα κοινό σχέδιο αρχιτεκτονικής. Η αρχιτεκτονική που προτείνουμε μέσα από την παρούσα μελέτη, αποτελεί την βάση για την ανάπτυξη του συστήματος. Ωστόσο η προγραμματιστική υλοποίηση δεν αποτελεί δέσμευση για κάποιον που επιθυμεί να αναπτύξει το σύστημα της SOVP.

Η συγκέντρωση **ετερογενών τεχνολογιών**, η **ενοποίηση** και η **απεικόνισή τους ως γνωστικές υπηρεσίες**, αποτελούν τον πυρήνα λειτουργίας του συστήματος της SOVP. Στη συνέχεια θα παρουσιάσουμε τις τεχνολογίες στις οποίες θα βασιστούμε για την υλοποίηση του συστήματος της SOVP. Επισημαίνουμε ωστόσο ότι η χρήση των συγκεκριμένων τεχνολογιών δεν αποτελεί δεσμευτική επιλογή για μελλοντικές επεκτάσεις και βελτιώσεις του συστήματος. Η επιλογή των συγκεκριμένων τεχνολογιών πραγματοποιήθηκε σύμφωνα με την άποψή μας και με στόχο την βέλτιστη δυνατή κάλυψη των αναγκών υλοποίησης. Το συγκεκριμένο κεφάλαιο συμπεριλαμβάνει τις εξής αναφορές: 1) **multi-agent συστήματα**, 2) **Foundation for Intelligent Physical Agents – FIPA,** 3) **JADE & JADEX τεχνολογία** (τεχνολογία ανάπτυξης multi-agent συστημάτων). Στη συνέχεια παρουσιάζουμε την ανάλυση κάθε στοιχείου ξεχωριστά.

13.1 Επισκόπηση των Multi-agent συστημάτων

Τα multi-agent συστήματα αποτελούν συστήματα που προκύπτουν από την συνεργασία πολλών διαφορετικών πρακτόρων μεταξύ τους. Στην παρούσα ενότητα θα επιχειρήσουμε την επισκόπηση των multi-agent συστημάτων έχοντας προηγουμένως αναφερθεί στην έννοια του agent (πράκτορα) και στις διάφορες κατηγορίες πρακτόρων που μπορεί να συναντήσουμε. Ακολουθούν οι δύο υποενότητες στις οποίες αναπτύσσουμε τα όσα προαναφέραμε.

13.1.1 Η έννοια του πράκτορα και οι κατηγορίες του

Η έννοια του πράκτορα συναντάται σε διαφορετικά επιστημονικά πεδία.[97] Στην πληροφοριακή τεχνολογία η πράκτορες αναφέρονται ως πράκτορες λογισμικού (Software agents). Οι πράκτορες λογισμικού προσδιορίζονται ως: «*αυτόνομες, ανεξάρτητες μονάδες λογισμικού οι οποίες εκτελούν ένα συγκεκριμένο σύνολο ενεργειών αλληλεπιδρώντας με το περιβάλλον τους και διαθέτουν την ικανότητα επικοινωνίας με άλλους πράκτορες λογισμικού, μέσω κάποιας προκαθορισμένης γλώσσας ανταλλαγής μηνυμάτων*».[98] Η αυτονομία ενός πράκτορα συνεπάγεται την δυνατότητα να ενεργεί χωρίς την άμεση ανθρώπινη ή άλλη παρέμβαση, παρουσιάζοντας μια μορφή ελέγχου των εσωτερικών του

[97] Agent, http://en.wikipedia.org/wiki/Agent, πρόσβαση: Μάιος, 2010.
[98] **Software Agent**, http://en.wikipedia.org/wiki/Software_agent, πρόσβαση: Μάιος, 2010.

διεργασιών. Η ικανότητα δράσης ενός πράκτορα με το περιβάλλον στο οποίο βρίσκεται, συνεπάγεται την δυνατότητα αντίληψης των μεταβολών που μπορούν να προκύψουν και την αντίδραση του στις μεταβολές αυτές. Η δυνατότητα επικοινωνίας με άλλους πράκτορες λογισμικού, ενισχύει την δυνατότητα δράσης του πράκτορα με το περιβάλλον ενώ παράλληλα βοηθά στην ανάπτυξη της αυτονομίας των λειτουργιών του.

Στους πράκτορες λογισμικού μπορούμε να συναντήσουμε μια σειρά από χαρακτηριστικά που προσδιορίζουν την λειτουργία τους. Μέσα από προγενέστερες ερευνητικές διαδικασίες, τα χαρακτηριστικά αυτά καταγράφονται ως εξής:[(99)]

- Αντιδραστικότητα (Reactivity): η ικανότητα να αντιλαμβάνεται καταστάσεις και να αντιδρά σε αυτές.
- Αυτονομία (Autonomy): αυτοδιαχείριση των λειτουργιών του.
- Συνεργατική συμπεριφορά (Collaborative behavior): δυνατότητα συνεργασίας με άλλες οντότητες του περιβάλλοντός του.
- Ικανότητα επικοινωνίας (Communication ability): η ικανότητα του πράκτορα να επικοινωνεί με άλλους πράκτορες, χρησιμοποιώντας μια κοινή γλώσσα επικοινωνίας που παρουσιάζεται διαφορετική από τις τυπικές γλώσσες συμβόλων και πρωτοκόλλων.
- Ικανότητα συμπερασμάτων (Inferential Capability): μελέτη της κατάστασης που αντιμετωπίζει και συγκέντρωση πληροφοριών που βοηθούν στην διαμόρφωση της λειτουργίας του ανάλογα με την τρέχουσα κατάσταση.
- Χρονική συνέχεια (Temporal continuity): παραμονή σε μια τρέχουσα κατάσταση για μεγάλα χρονικά διαστήματα.
- Προσωπικότητα (Personality): δυνατότητα εισαγωγής χαρακτηριστικών από την ανθρώπινη συμπεριφορά.
- Προσαρμογή (Adaptive): ικανότητα να αλλάζουν την κατάσταση του σύμφωνα με τις τρέχουσες ανάγκες.
- Κινητικότητα (Mobility): η δυνατότητα μεταφοράς από μια πλατφόρμα σε άλλη με αυτόνομα και δυναμικά.

Μελετώντας προσεκτικά τα παραπάνω χαρακτηριστικά θα μπορούσαμε να αναφέρουμε ότι ανάλογα με το πλήθος χαρακτηριστικών που συγκεντρώνει ένας agent, δημιουργούνται και οι αντίστοιχες κατηγορίες. Ωστόσο δεν είναι τόσο εύκολο να δώσουμε συγκεκριμένη τυπολογία στους πράκτορες λογισμικού. Σύμφωνα με σχετικές μελέτες, τα κριτήρια για την δημιουργία agent τύπων, σχετίζονται με ποικίλα στοιχεία.[(100)] Εν συντομία παρουσιάζουμε τα πέντε επικρατέστερα κριτήρια, από την έρευνα, για την ανάπτυξη τύπων σε πράκτορες λογισμικού.

[99] Bradshaw Jeffrey, "**Software Agents**", AAAI Press, 1997, An Introduction to software agents, κεφάλαιο 1, σελ. 8.

[100] Hyacinth S. Nwana, "**Software Agents: An Overview**", Knowledge Engineering Review, Cambridge University, Vol. **11**, No 3, 1996, σελ. 1-40.

13.1.1.1 Κριτήριο της Κινητικότητα

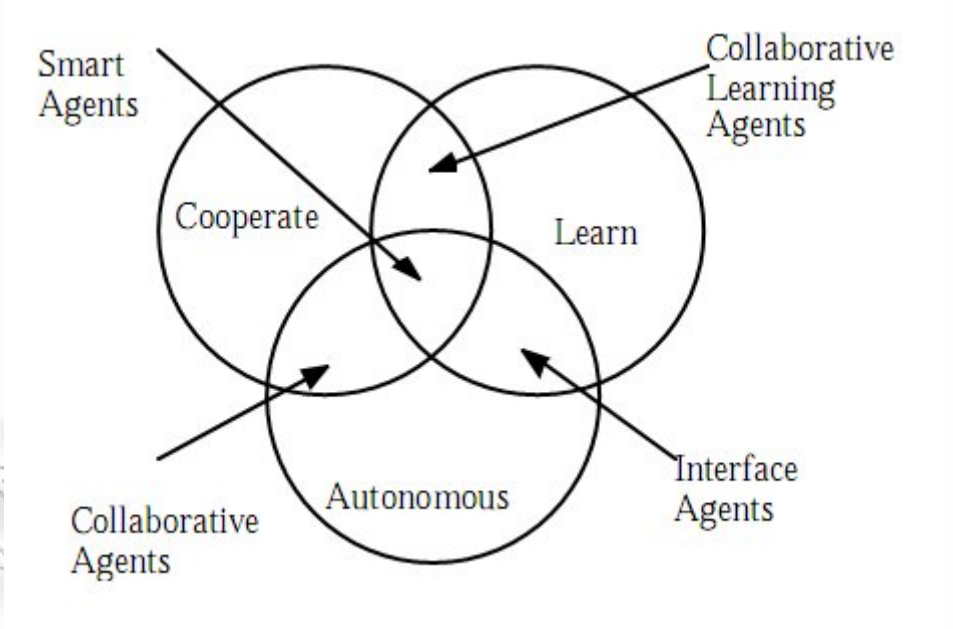
Οι πράκτορες μπορούν να κατατάσσονται βάση της κινητικότητας τους μέσα στα δίκτυα. Αυτό το κριτήριο μας παρέχει τους τύπους *στατικών* και *κινητικών* πρακτόρων.

13.1.1.2 Κριτήριο τρόπου αντίδρασης πρακτόρων

Ανάλογα με τον τρόπο αλληλεπίδρασης του πράκτορα με το περιβάλλον του, έχουμε τους *deliberative agents* και τους *reactive agents*. Η πρώτη μορφή αναφέρεται στους agents οι οποίοι συγκεντρώνουν εμπειρίες και ερεθίσματα προκειμένου να καταλήξουν σε συμπεράσματα σχετικά με την λειτουργία τους. Η δεύτερη κατηγορία αφορά στους πράκτορες οι οποίοι αναλόγως με τα ερεθίσματα του περιβάλλοντός τους, αντιδρούν και τροποποιούν την λειτουργία τους.

13.1.1.3 Κριτήριο έρευνας BT Labs

Σύμφωνα με την έρευνα του BT Labs[(101)] μπορούμε να αναφέρουμε τρία βασικά κριτήρια κατηγοριοποίησης των πρακτόρων λογισμικού, την αυτονομία, τη μάθηση και την συνεργασία. Σύμφωνα με αυτό το μοντέλο τυποποίησης προκύπτει και το σχήμα της εικόνας 21 η οποία προέρχεται από το βιβλίο του Nwana[(102)] και παρουσιάζει τρία διαφορετικά πεδία δράσης των agents μέσα από τα οποία δημιουργούνται τέσσερεις διαφορετικοί τύποι: Smart agents, Collaborative learning agents, Collaborative agents, Interface agents.



Εικόνα 21 "Τύποι πρακτόρων σύμφωνα με τον Nwana (Nwana, 1996)"

[101] **BT Research**, http://en.wikipedia.org/wiki/BT_Research, πρόσβαση: Μάιος, 2010.
[102] Βλ. παρ. 98, σελ. 76.

13.1.1.4 Κριτήριο του ρόλου λειτουργίας

Η κατηγοριοποίηση των πρακτόρων μπορεί να γίνει ανάλογα με τους ρόλους που διαδραματίζουν σε ένα σύστημα. Έτσι μπορεί να έχουμε του πράκτορες πληροφοριών του διαδικτύου (WWW Information Agents). Τέτοιου είδους πράκτορες συναντώνται συνήθως σε δίκτυα μεγάλου όγκου διακινούμενων πληροφοριών όπως το διαδίκτυο, και συμβάλουν στην διαχείρισή τους.

13.1.1.5 Κριτήριο συνδυασμού χαρακτηριστικών

Σύμφωνα με το κριτήριο αυτό προκύπτουν οι υβριδικοί πράκτορες οι οποίοι συνδυάζουν σε μια οντότητα, περισσότερα από ένα χαρακτηριστικά. Αποτελούν ουσιαστικά πράκτορες με σύνθετη συμπεριφορά.

Σύμφωνα με τα παραπάνω κριτήρια κατηγοριοποίησης των πρακτόρων μπορούσαμε να υποστηρίξουμε την άποψη ότι προκύπτει ένα μεγάλο εύρος τυποποίησης. Ωστόσο σύμφωνα με υπάρχουσες επιστημονικές μελέτες, θα μπορούσαμε να γίνουμε πιο συγκεκριμένοι αναφέροντας ότι σύμφωνα με τις δυνατότητες και τα χαρακτηριστικά κάθε κατηγορίας, προκύπτουν **επτά** διαφορετικοί τύποι πρακτόρων λογισμικού. Συγκεκριμένα αυτοί είναι:[(103)]

- Collaborative agents
- Interface agents
- Mobile agents
- Information / Internet agents
- Reactive agents
- Hybrid agents
- Smart agents

Ο συνδυασμός των παραπάνω τύπων πρακτόρων σε ομάδες συνεργασίας συνθέτουν ισχυρά και έξυπνα συστήματα τα οποία μπορούν να ανταποκριθούν στις απαιτήσεις του περιβάλλοντος στο οποίο ανήκουν. Τα συστήματα αυτά αναφέρονται ως multi-agent συστήματα και στην υποενότητα 13.1.2 ακολουθεί η περιγραφή τους.

13.1.2 Multi-agent συστήματα

Τα multi-agent συστήματα προσδιορίζονται ως *συστήματα τα οποία συνθέτονται από το σύνολο πολλών intelligent agents (έξυπνων πρακτόρων) οι οποίοι συνεργάζονται μεταξύ τους, στοχεύοντας στην επίλυση προβλημάτων που τα μονολιθικά συστήματα λογισμικού δεν είναι σε θέση να τα αντιμετωπίσουν.*[(104)(105)]
Οι συμμετέχοντες σε ένα multi-agent σύστημα είναι ανεξάρτητοι πράκτορες λογισμικού οι οποίοι φέρουν χαρακτηριστικά αυτονομίας μέσω των οποίων μπορούν να διαχειριστούν τις λειτουργικές τους δυνατότητες. Τα συστήματα αυτού του τύπου οργανώνονται ως αποκεντρωτικά συστήματα τα οποία δεν εξαρτώνται

[103] Βλ. παρ. 98, σελ. 76.
[104] Wooldridge M.,"**An Introduction to MultiAgent Systems**", John Wiley & Sons Ltd, 2002.
[105] **Multi-agent system**, http://en.wikipedia.org/wiki/Multi-agent_system, πρόσβαση: Μάιος 2010.

από κεντρικές οργανωτικές αρχές. Κάθε agent φέρει συγκεκριμένα χαρακτηριστικά τα οποία συνήθως είναι προγραμματισμένα να αλληλεπιδρούν με συγκεκριμένες καταστάσεις στο περιβάλλον το οποίο ανήκουν. Η συνεργασία μεταξύ πρακτόρων παρέχει την δυνατότητα αντιμετώπισης σύνθετων προβλημάτων, μέσα από την σύνθεση πολύπλοκων κατανεμημένων συστημάτων. Η επικοινωνία μεταξύ των agents πραγματοποιείται με την βοήθεια προκαθορισμένων γλωσσών επικοινωνίας που αναφέρονται ως **Agent Communication Languages (ACLs).**[106] Οι πιο διαδεδομένες γλώσσες επικοινωνίας είναι:

- Knowledge Query and Manipulation Language (KQML)
- Foundation for Intelligent Physical Agents (FIPA – ACL)

Η **KQML** αποτελεί μια γλώσσα και ένα πρωτόκολλο επικοινωνίας που αρχικά σχεδιάστηκε με σκοπό την κάλυψη των επικοινωνιακών αναγκών σε συστήματα γνώσης (Knowledge Systems). Ωστόσο η διαδικασία ανάπτυξης των συστημάτων κατέστησε δυνατή την χρήση της KQML ως γλώσσα επικοινωνίας των πρακτόρων. Στα σύγχρονα συστήματα πρακτόρων η KQML έχει αντικατασταθεί με την FIPA – ACL ο σχεδιασμός της οποίας στοχεύει εξολοκλήρου σε μια γλώσσα επικοινωνίας μεταξύ των πρακτόρων λογισμικού.

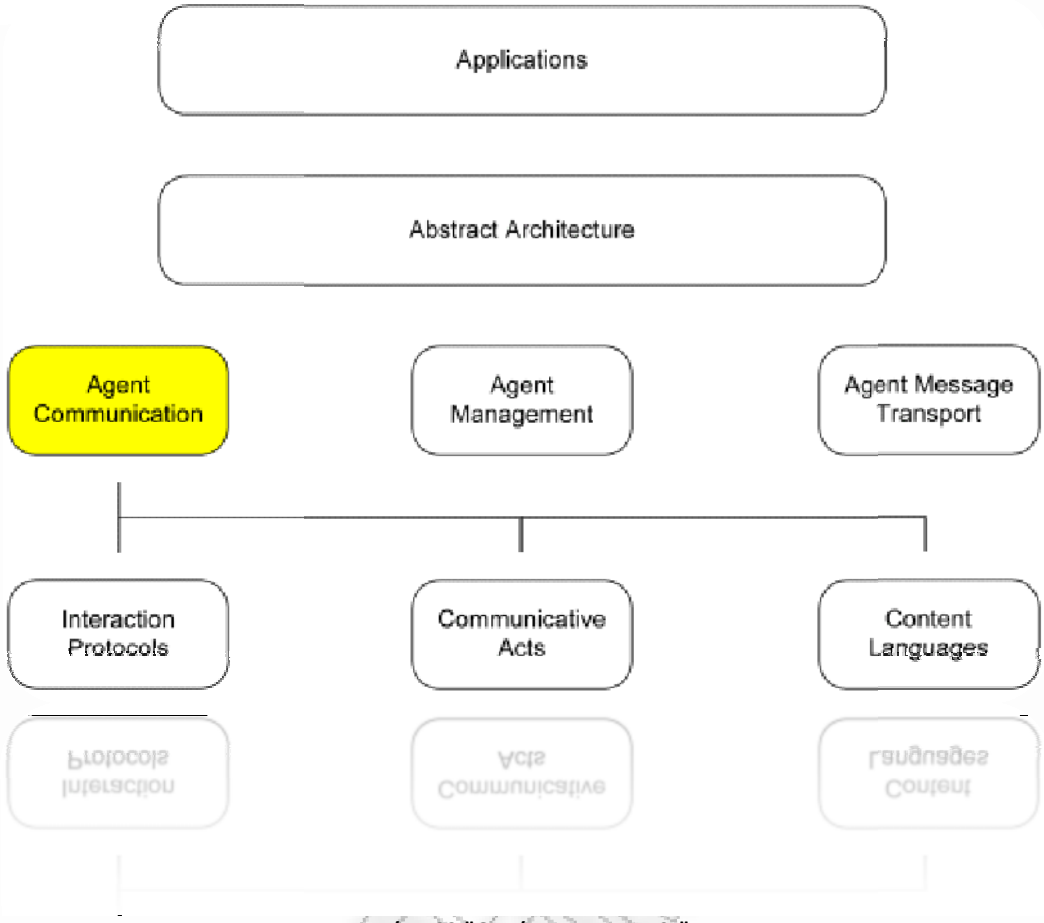
Η **FIPA** (Foundation for Intelligent Physical Agents) είναι μια μη κερδοσκοπική οργάνωση, με έδρα την Γενεύη, που σκοπό έχει την προώθηση της επιτυχίας των βασισμένων σε πράκτορες εφαρμογών. Θεσπίζει λοιπόν προδιαγραφές που μεγιστοποιούν τη διαλειτουργικότητα των υπηρεσιών και του εξοπλισμού σε εφαρμογές βασισμένες σε πράκτορες.[107] Η FIPA - ACL βασίζεται στην θεωρία "**speech - act**" (ομιλία και πράξη) σύμφωνα με την οποία τα μηνύματα συνεπάγονται ενέργειες. Η επικοινωνία πραγματοποιείται με σκοπό την εκτέλεση κάποιας συγκεκριμένης ενέργειας. Οι προδιαγραφές της, συμπεριλαμβάνουν μηνύματα τα οποία περιγράφουν την κατάσταση του αποστολέα και την κατάσταση που επιθυμούμε για τον παραλήπτη. Επιπλέον συμπεριλαμβάνονται πρωτόκολλα τα οποία συνδέονται άμεσα με συγκεκριμένες ενέργειες προς εκτέλεση. Τα μηνύματα είναι περιγραφικά και έχουν στατική δομή, η οποία συνδυάζεται με ένα σύνολο ενεργειών. Επιπλέον η εφαρμογή της FIPA-ACL σε συστήματα που χρησιμοποιούν QKML διευκολύνεται με την χρήση κατάλληλων μηνυμάτων που περιλαμβάνουν οι προδιαγραφές της και προωθούν την εύκολη μετάβαση από την QKML στην FIPA-ACL.[108] Στην εικόνα 22 απεικονίζεται το μοντέλο οργάνωσης της επικοινωνίας των agents, σύμφωνα με την προσέγγιση του FIPA. Στην ενότητα 13.2 επιχειρούμε την εκτενή ανάλυση της FIPA, περιγράφοντας μια σειρά από τα χαρακτηριστικά και τις λειτουργίες που προάγει για την οργάνωση multi-agent συστημάτων.

---

[106] **Agent Communications Language**, http://en.wikipedia.org/wiki/Agent_Communications_Language, πρόσβαση: Μάιος 2010.

[107] **Foundation for Intelligent Physical Agents**, http://www.fipa.org/, πρόσβαση: Μάιος 2010.

[108] FIPA 97**, "Foundation for Intelligent Physical Agents - FIPA 97 Specifications",** version 1.0, part 2, 1997.



Εικόνα 22:"Οργάνωση FIPA-ACL"

Έχοντας αναφερθεί στην έννοια του agent, μεταβήκαμε στην περιγραφή των multi-agent συστημάτων. Περιγράψαμε τα multi-agent συστήματα αναφέροντας ότι πρόκειται για συστήματα που προκύπτουν μέσα από την συνεργασία – αλληλεπίδραση διαφορετικών πρακτόρων λογισμικού. Ωστόσο στην προσπάθειά μας να συγκεκριμενοποιήσουμε την φύση ενός multi-agent συστήματος, θα πρέπει να λάβουμε υπόψη μας ότι κάθε πράκτορας αποτελεί μια διαφορετική αυτόνομη μονάδα λογισμικού η οποία προάγει ένα σύνολο συγκεκριμένων λειτουργιών και σε συνδυασμό με τις λειτουργίες άλλων πρακτόρων μπορούν να αποδώσουν μια συγκεκριμένη υπόσταση σε ένα multi-agent σύστημα.

Στην πρώτη φάση ανάπτυξης της έρευνας για τους agents (μέχρι και το 1985), οι πρώτοι πράκτορες που κατασκευάστηκαν, χαρακτηρίζονται ως «**συμβολικά διαλογιζόμενους πράκτορες**» (symbolic reasoning agents) ενώ η δομή τους προγραμματιστικά θεωρείται στατική. Από το 1985 έως και σήμερα, οι agents που αναπτύσσονται, χαρακτηρίζονται ως «**πράκτορες δράσης**» (reaction agents) οι οποίοι φέρουν χαρακτηριστικά δυναμικής αλληλεπίδρασης με παράγοντες από το περιβάλλον τους. Η **μετάβαση** του μοντέλου πρακτόρων, από τους «**συμβολικά διαλογιζόμενους πράκτορες**» (symbolic reasoning agents) σε «**πράκτορες δράσης**» (reaction agents), ενέπνευσε τους ερευνητές ώστε να επιχειρήσουν το συνδυασμό στοιχείων της συλλογικής αρχιτεκτονικής με στοιχεία της αντιδραστικής

αρχιτεκτονικής, με αποτέλεσμα την δημιουργία νέων **μοντέλων αρχιτεκτονικής πρακτόρων**.[109]

Οι διάφορες αρχιτεκτονικές πρακτόρων μπορούν να κατηγοριοποιηθούν ως εξής:

- Logic based agents: η λήψη αποφάσεων προκύπτει μέσω λογικής αφαίρεσης,
- Reactive agents: η λήψη αποφάσεων βασίζεται στην χαρτογράφηση μιας κατάστασης και την εκτέλεση της κατάλληλης δράσης,
- Belief-desire-intention agents: η λήψη αποφάσεων εξαρτάται από τον χειρισμό των δομών δεδομένων που αντιπροσωπεύουν τις πεποιθήσεις, τις επιθυμίες και τις προθέσεις ενός agent,
- Layered architectures: η λήψη αποφάσεων εξαρτάται από τα διάφορα επίπεδα λογισμικού όπου κάθε ένα έχει διαφορετική βαρύτητα στη συγκεκριμένη διαδικασία.

Στη συνέχεια θα πραγματοποιήσουμε μια σύντομη αλλά περιεκτική αναφορά σχετικά με κάθε μια από τις παραπάνω αρχιτεκτονικές πρακτόρων.

a. Logic based agents

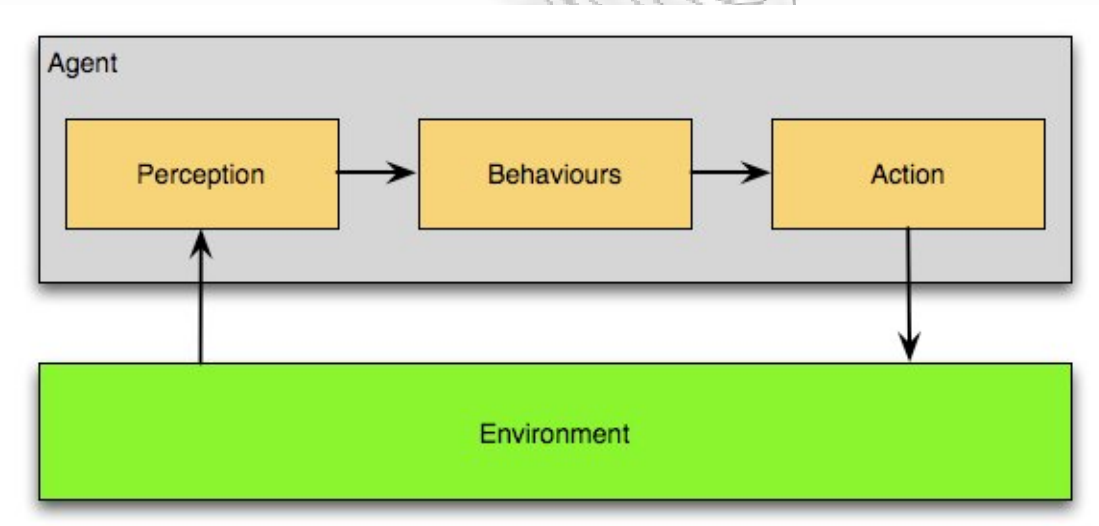
Η χρήση του στοιχείου της λογικής με σκοπό την ανάλυση και κωδικοποίηση μιας θεωρίας η οποία θα δηλώνει την καλύτερη δράση, αποτελεί την βασική αρχή της συγκεκριμένης αρχιτεκτονικής. Ο προγραμματισμός ενός logic based agent στοχεύει στο ορισμό συνθηκών οι οποίες αντιλαμβάνονται μια κατάσταση και ελέγχουν στην βάση δεδομένων με τις δράσεις, για την εύρεση της καταλληλότερης δράσης στην συγκεκριμένη κατάσταση. Οι δράσεις του agent είναι προκαθορισμένες ήδη και το μόνο που διαφέρει είναι η χρήση κάθε δράσης, ανάλογα με την περίπτωση. Η ιδέα βασίζεται στην ανάπτυξη ενός συνόλου κανόνων **p** οι οποίοι συνδυάζονται με μία βάση ενεργειών **A**. Έστω ότι προκύπτει μια κατάσταση **a.** Ο πράκτορας αρχικά ελέγχει αν η κατάσταση a ανήκει στο σύνολο ενεργειών που έχουν προκαθοριστεί στην ΒΔ, μέσω μιας συνθήκης **Do(a)** η οποία χρησιμοποίει το σύνολο των κανόνων p προκειμένου να κάνει τον έλεγχο**.** Εάν η κατάσταση ανήκει στο σύνολο τότε ο πράκτορας καλεί την αντίστοιχη δράση που πρέπει να εκτελεστεί, διαφορετικά ελέγχει την βάση δεδομένων επιχειρώντας να συνδυάσει ένα σύνολο ενεργειών βάση των κανόνων του συνόλου p, οι οποίες θα υπακούν στην συνθήκη Do(a). Αν προκύψει κάποιο αποτέλεσμα τότε ενημερώνει για την εκτέλεση της ενέργειας, διαφορετικά ενημερώνει ότι αδυνατεί να βρει κάποια δράση προς εκτέλεση.

Διαπιστώνουμε λοιπόν ότι στους logic based agents καθορίζεται η συμπεριφορά τους από ένα σύνολο κανόνων μέσω των οποίων επιλέγονται οι κατάλληλες δράσεις ή ο συνδυασμός τους, για την αντιμετώπιση μιας κατάστασης.

[109] Weiss Gerhard, "**Multiagent Systems – A Modern Approach to Distributed Modern Approach to Artificial Intelligence**", MIT Press, 1999 , σελ. 42-66.

b. Reactive agents

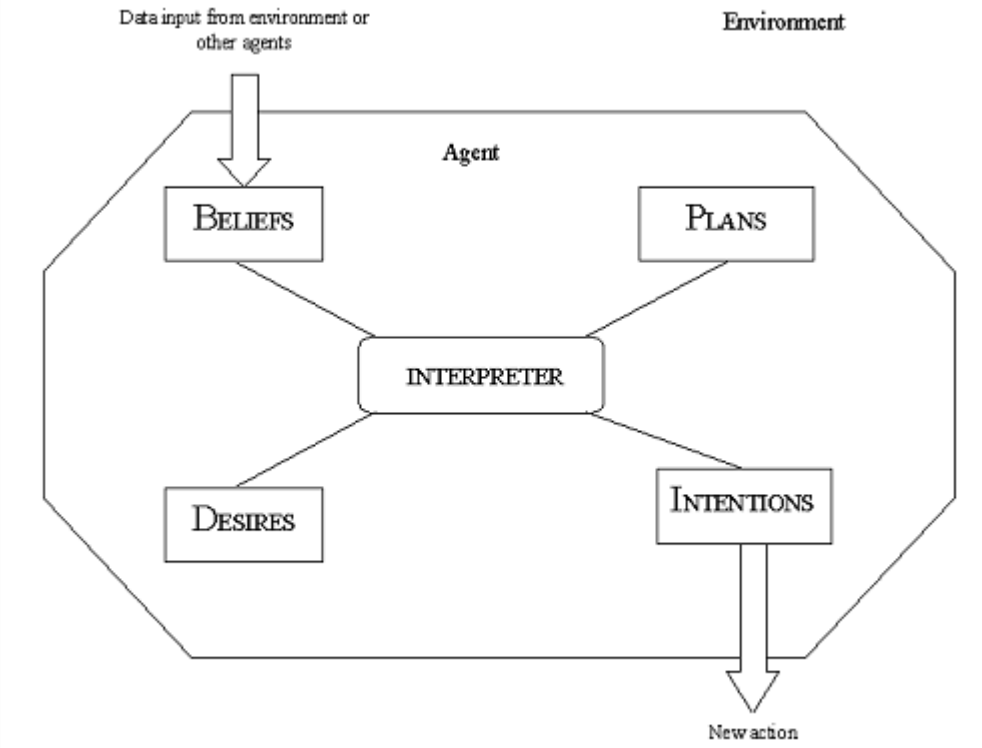
Οι reactive agents βασίζουν την αρχιτεκτονική τους στην αρχή της σχέσης «κατάστασης-αντίδρασης». Οι πράκτορες αντιλαμβάνονται την κατάσταση που επικρατεί στο περιβάλλον του και αντιδρούν θέτοντας σε εφαρμογή την αντίστοιχη ενέργεια. Υπάρχουν κάποιες παράμετροι οι οποίες ορίζονται από τον σχεδιαστή του agent και βάση αυτών εκτελείτε η λειτουργία του. Επίσης ο τρόπος με τον οποίο ερμηνεύει κάθε ερέθισμα που θα λάβει έχει προκαθοριστεί από το σχεδιαστή του. Χαρακτηριστικό παράδειγμα αποτελεί ο πράκτορας εισερχόμενης αλληλογραφίας όπου έχει προγραμματιστεί ώστε να ενημερώνει τον χρήστη του κάθε φορά που υπάρχει ένα ηλεκτρονικό μήνυμα στην θυρίδα του. γενικότερα οι reactive agents είναι μονάδες λογισμικού προγραμματισμένες με μια σειρά εντολών **if, then** όπου κάθε φορά επεξεργάζονται μια συνθήκη και εν συνεχεία εκτελούν την αντίστοιχη δράση. Οι πράκτορες αυτής της μορφής οργανώνονται εξαρχής μέσα από ένα σύνολο κανόνων που καθορίζουν οι προγραμματιστές τους. Ωστόσο δεν είναι ενδεικτικά παραδείγματα προς χρήση διότι είναι άμεσα εξαρτώμενοι από τους κανόνες που ορίζουν την συμπεριφορά τους σε κάθε κατάσταση και στην πραγματικότητα δεν είναι καθόλου δυναμική η λειτουργία τους. Το σημαντικότερο όλων, σε έναν reactive agent είναι η υψηλή ταχύτητα αντίδρασης σε καταστάσεις τις οποίες έχει προγραμματιστεί εκ των προτέρων να αντιμετωπίσει.



Εικόνα 23:"Reactive Agent"

c. Belief-desire-intention agents

Η αρχιτεκτονική Belief Desire Intention (BDI) αποτελεί το μια από τις πιο συνηθισμένες αρχιτεκτονικές ανάπτυξης πρακτόρων. Ένας BDI Agent συγκεντρώνει ένα σύνολο πεποιθήσεων όπου σε συνδυασμό με τους στόχους του, διεξάγει συμπεράσματα για την πραγματοποίηση των δράσεων του. Σκοπός της λειτουργίας αυτών των πρακτόρων είναι η ικανοποίηση των συγκεκριμένων πεποιθήσεων τους, μέσα από το σύνολο των στόχων τους οι οποί πραγματοποιούνται με την εφαρμογή μιας δράσης. Το μοντέλο λειτουργίας των BDI πρακτόρων, απεικονίζεται στην εικόνα 24 παρακάτω.



Εικόνα 24:"BDI Agent"

Όπως παρατηρούμε και στην παραπάνω εικόνα, υπάρχουν τέσσερεις βασικοί παράγοντες στην αρχιτεκτονική BDI.[110]

Beliefs: συμπεριλαμβάνουν όλες εκείνες τις πληροφορίες που αφορούν την κατάσταση ενός πράκτορα μέσα στο περιβάλλον που ανήκει. Επιπλέον οι πεποιθήσεις ενός πράκτορα μπορούν μελλοντικά να αναπροσαρμοστούν και να αλλάξουν μέσα από συνδυασμούς με άλλες πεποιθήσεις. Να αναφέρουμε ότι χρησιμοποιείται ο όρος «πεποίθηση» και όχι «γνώση» διότι οι σχετικές πληροφορίες που χρησιμοποιεί ο πράκτορας, δεν σημαίνει ότι είναι και σωστές. Η υλοποίηση των πληροφοριών των πεποιθήσεων για κάθε agent γίνεται με την βοήθεια μιας βάσης δεδομένων η οποία διατηρεί σχετικές πληροφορίες σχετικά με τα «πιστεύω» της οντότητας του πράκτορα, για το περιβάλλον και για τους άλλους πράκτορες.

Desires: οι επιθυμίες που αντιστοιχούν σε κάθε πράκτορα, αντιπροσωπεύουν τους στόχους που επιδιώκει να πετύχει ο πράκτορας μέσα από μια σειρά συγκεκριμένων δράσεων. Οι στόχοι δεν αντιπροσωπεύονται πάντα από τις επιθυμίες του πράκτορα. Για παράδειγμα όταν έχουμε έναν πράκτορα που κινείται μέσα σε ένα δίκτυο, μπορεί να αλλάξει θέση αλλά αυτό να μην ήταν στους άμεσους στόχους τους. Ως στόχος για έναν πράκτορα μπορεί να θεωρηθεί η επιλογή της καλύτερης τιμής πώλησης για ένα προϊόν, όπου ικανοποιεί την επιθυμία για φθηνές αγορές.

[110] Shajari Mehdi, Ghorbani Ali, "**Application of Belief-Desire-Intention Agents in Intrusion Detection & Response**", Institute for Information Technology e-Business National Research Council of Canada, σελ. 2.

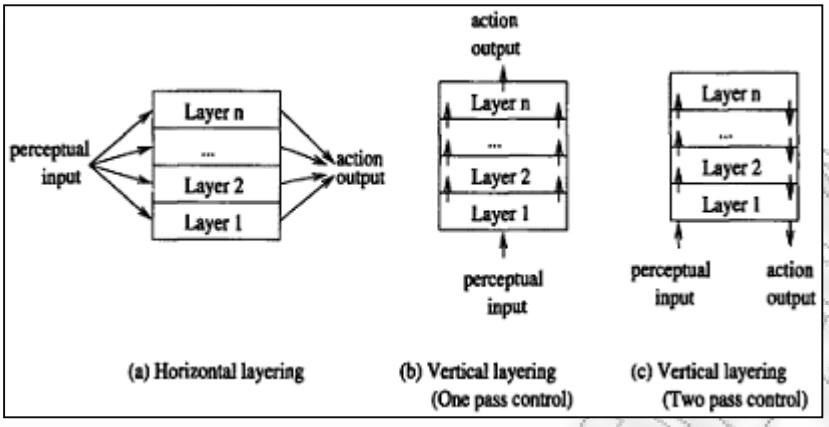
Plans: τα σχέδια αποτελούν το σύνολο των οργανωμένων δράσεων που έχουν καταγραφεί σε έναν πράκτορα με σκοπό την επίτευξη κάποιων στόχων του ή γενικότερα την ικανοποίηση των επιθυμιών του. Ένα μέρος των σχεδίων είναι ήδη καταχωρημένο στον πράκτορα, από την στιγμή του σχεδιασμού του και περιλαμβάνει το σύνολο των δράσεων για κάποιες συγκεκριμένες πεποιθήσεις, όπως για παράδειγμα η διαδικασία ακρόασης και λήψης μηνυμάτων. Τα σχέδια πραγματοποιούνται με σκοπό να επιτευχθούν οι προθέσεις του πράκτορα.

Intentions: οι προθέσεις αποτελούν ουσιαστικά τα σχέδια που βρίσκονται σε εκτέλεση. Συμπεριλαμβάνουν τις δεσμεύσεις ενός agent με σκοπό την επίτευξη των στόχων του. Ουσιαστικά αποτελούν την έξοδο από την διαδικασία επεξεργασίας των ερεθισμάτων και της παραγωγής αντιδράσεων.

d. Layered architectures

Στην αρχιτεκτονική αυτής της μορφής έχουμε την συνεργασία διαφορετικών επιπέδων λειτουργιών σε έναν πράκτορα, για την διεξαγωγή των διαδικασιών του. Συγκεκριμένα η λειτουργία του agent χωρίζεται σε τρία επίπεδα: i) έλεγχος και ανάκτηση πληροφορίας, ii) φιλτράρισμα πληροφορίας και διεξαγωγή κρίσιμων δεδομένων και iii) συμμετοχή στην διαδικασία λήψης απόφασης για εκτέλεση μιας δράσης. Η αρχιτεκτονική διαστρωμάτωσης, παρουσιάζεται είτε με την μορφή της οριζόντιας είτε με την μορφή της κάθετης διαστρωμάτωσης.

Στην **οριζόντια διαστρωμάτωση**, κάθε επίπεδο λειτουργιών, λειτουργεί ως μια ανεξάρτητη μονάδα όπου λαμβάνει διαφορετική είσοδο και παράγει διαφορετική έξοδο. Η πληροφορία στην είσοδο φιλτράρεται και τροφοδοτεί το κάθε επίπεδο ξεχωριστά, επιταχύνοντας έτσι την διαδικασίας εξαγωγής αποτελεσμάτων. Επίσης τα αποτελέσματα στην έξοδο κάθε λειτουργίας συνθέτονται δημιουργώντας μια κοινή έξοδο – αποτέλεσμα. Αντιθέτως, στην **κατακόρυφη διαστρωμάτωση** υπάρχει μια μοναδική είσοδος στο αρχικό στρώμα όπου στην συνέχεια γίνετε η επεξεργασία της πληροφορίας και κάθε στρώμα παράγει ένα σύνολο πληροφοριών που τροφοδοτούν το επόμενο, έως ότου φτάσουμε στο τελευταίο στρώμα όπου και παράγεται η τελική πληροφορία. Η εικόνα 25 αποτελεί χαρακτηριστική απεικόνιση των όσων αναφέραμε.

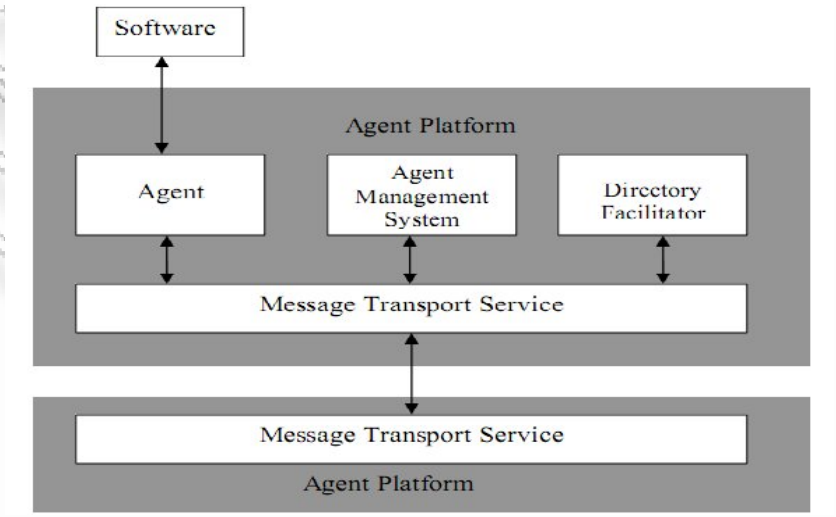


Εικόνα 25:"Layered agents architectures"

Σύμφωνα με τα παραπάνω, μπορούμε να αναφέρουμε συμπερασματικά ότι, τα multi-agent συστήματα μπορούν να συγκεντρώσουν μεγάλο αριθμό από διαφορετικές αρχιτεκτονικές πρακτόρων, μέσω των οποίων μπορούν να διεκπεραιώνουν το σύνολο των αναγκών τους. Τοποθετώντας τους πράκτορες σε στρατηγικά σημεία μέσα στο σύστημα μπορούμε να επιτύχουμε την δημιουργία ενός ιδιαίτερα αποδοτικού μηχανισμού ο οποίος θα λαμβάνει ερεθίσματα από το περιβάλλον του και θα αντιδρά με τον βέλτιστο δυνατό τρόπο. Επιπλέον η συνένωση διαφορετικών οντοτήτων – πρακτόρων μπορεί να πραγματοποιηθεί μέσα από την μεταξύ τους επικοινωνία όπου με την χρήση κατάλληλων ACL μηνυμάτων, τους δίνεται η δυνατότητα ανταλλαγής πληροφοριών.

13.2 Foundation for Intelligent Physical Agents – FIPA

Η FIPA, όπως εν συντομία αναφέραμε παραπάνω, αποτελεί μια μη κερδοσκοπική οργάνωση που έχει ως σκοπό της την ανάπτυξη ενιαίων προτύπων ανάπτυξης πρακτόρων. Η FIPA προδιαγράφει με συγκεκριμένο τρόπο το μοντέλο ανάπτυξης μιας πλατφόρμας πρακτόρων η οποία περιέχει ένα σύνολο από διαμεσολαβητές που βοηθούν στο να εξελιχτεί η επικοινωνία πάνω στην πλατφόρμα, μεταξύ των διαμεσολαβητών – πρακτόρων που συγκεντρώνει. Η εικόνα 26 απεικονίζει το πρότυπο πλατφόρμας πρακτόρων, όπως προδιαγράφεται από την FIPA.



Εικόνα 26:"FIPA - Πρότυπο πλατφόρμας πρακτόρων"

13.2.1 Οντότητες στο πρότυπο FIPA

Παρατηρώντας προσεκτικά το παραπάνω πρότυπο, διαπιστώνουμε ότι υπάρχουν τέσσερεις προκαθορισμένοι πράκτορες οι οποίοι είναι υπεύθυνοι για την διαχείριση της πλατφόρμας. Γενικότερα μια πλατφόρμα πρακτόρων αποτελείται από: i) τους υπολογιστές, ii) το λειτουργικό σύστημα που εγκαθίσταται η πλατφόρμα, iii) τους πράκτορες διαχείρισης της πλατφόρμας και iv) από τους πράκτορες που συγκεντρώνει η πλατφόρμα. Συνολικά οι οντότητες που συγκεντρώνει το πρότυπο πλατφόρμας της FIPA, είναι εξής:[(111)]

- Agent Management System – AMS
- Directory Facilitator – DF
- Agents

} **Οντότητες πρακτόρων**

- Message Transport Service – MTS

} **Λογισμικό**

Agent Management System – AMS

Ο AMS πράκτορας συγκεντρώνει όλες τις διευθύνσεις των συνδεδεμένων πρακτόρων στην πλατφόρμα. Κάθε πράκτορας που συνδέεται στην πλατφόρμα, αυτομάτως εγγράφεται στον κατάλογο του AMS και λαμβάνει ένα μοναδικό αναγνωριστικό το οποίο αναφέρεται ως Agent Identify (AID). Έτσι προκειμένου να εντοπίσουμε την ταυτότητα κάποιου πράκτορα, μπορούμε να ανατρέξουμε στον πράκτορα AMS.

Directory Facilitator - DF

Ο DF πράκτορας συγκεντρώνει σε έναν κατάλογο όλες τις διαθέσιμες υπηρεσίες από κάθε πράκτορα που είναι συνδεδεμένος στην πλατφόρμα. Όταν ένας πράκτορας εισέλθει στην πλατφόρμα, αφού εγγραφεί στον AMS και λάβει το μοναδικό αναγνωριστικό του, στην συνέχεια θα πρέπει να δηλώσει στο DF το σύνολο των υπηρεσιών που παρέχει. Οι υπηρεσίες του κάθε πράκτορα αντιστοιχίζονται με το AID που έχει λάβει κατά την σύνδεση του στην πλατφόρμα. Ο DF σε σχέση με τον AMS προσφέρει στην πλατφόρμα την υπηρεσία του «χρυσού οδηγού» (Yellow Pages). Επιπλέον σε μία πλατφόρμα πρακτόρων μπορούν να συνυπάρξουν περισσότεροι από ένας DF πράκτορες, κάτι που δεν επιτρέπεται στην περίπτωση του AMS πράκτορα, οποίος είναι ένας και μοναδικός για κάθε πλατφόρμα.

Agents – Πράκτορες

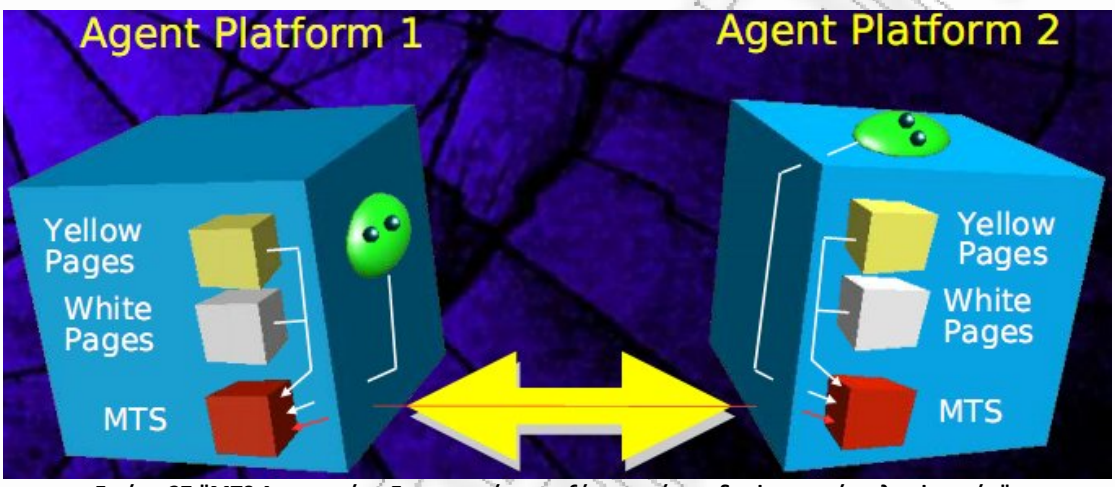
Οι πράκτορες, όπως έχει αναφερθεί αναλυτικά στην ενότητα 13.1.1, αποτελούν οντότητες οι οποίες διαθέτουν ένα σύνολο λειτουργιών μέσα από τις οποίες μπορούν να παρέχουν διάφορες υπηρεσίες. Οι πράκτορες μπορούν να επικοινωνούν με άλλους πράκτορες με την χρήση της γλώσσας επικοινωνίας πρακτόρων (Agent Communication Language - ACL). Κάθε πράκτορας θα πρέπει να φέρει ένα μοναδικό αναγνωριστικό το οποίο του προσφέρεται από τον AMS πράκτορα, κατά την εγγραφή του στην πλατφόρμα.

---

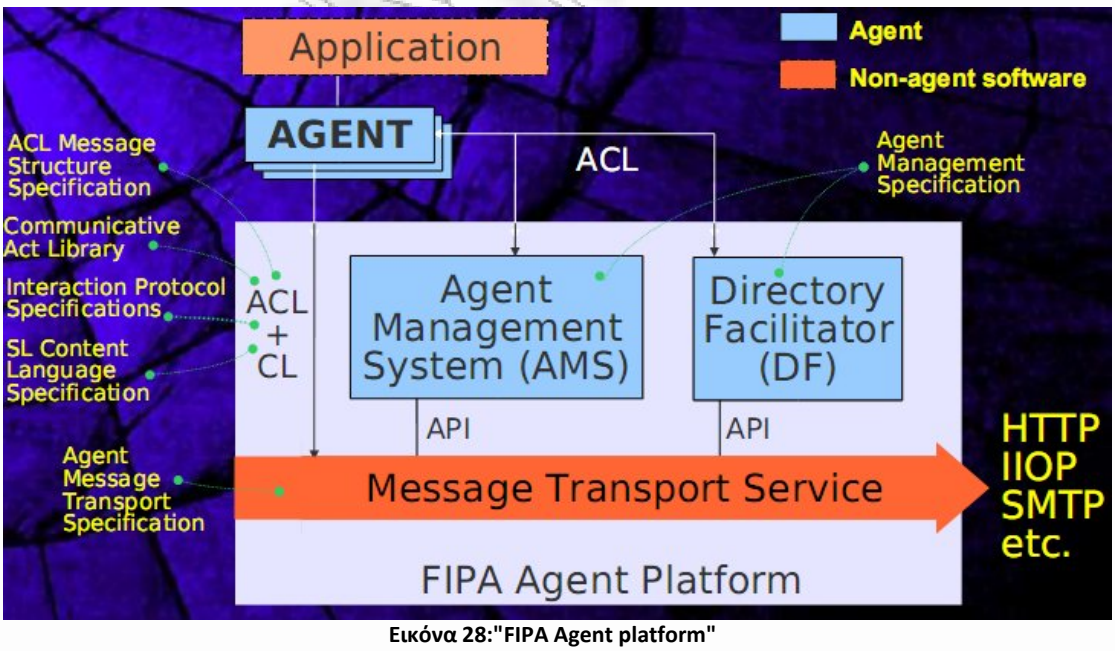
[111] Greenwood Dominic, **"FIPA – The Foundation for Intelligent Physical Agents"**, Whitestein Technologies AG, 2004.

Message Transport Service - MTS

Η MTS αποτελεί την υπηρεσία της πλατφόρμας, μέσω της οποίας οι πράκτορες μπορούν να επικοινωνούν μεταδίδοντας και λαμβάνοντας ACL μηνύματα. Η MTS εκτός από την δυνατότητα επικοινωνίας μεταξύ των πρακτόρων εντός της πλατφόρμας, παρέχει και την δυνατότητα επικοινωνίας με άλλες MTS από άλλες πλατφόρμες πρακτόρων, πράγμα που συνεπάγεται ότι οι πράκτορες διαφορετικών πλατφορμών μπορούν να επικοινωνούν μεταξύ τους, χρησιμοποιώντας απλά ένα δίκτυο επικοινωνίας. Να επισημάνουμε ότι η MTS δεν αποτελεί οντότητα πράκτορα αλλά ανεξάρτητη οντότητα λογισμικού. Χαρακτηρίζεται ως Non-agent software. Ενδεικτικό της λειτουργίας της MTS υπηρεσίας, αποτελεί η εικόνα 27.



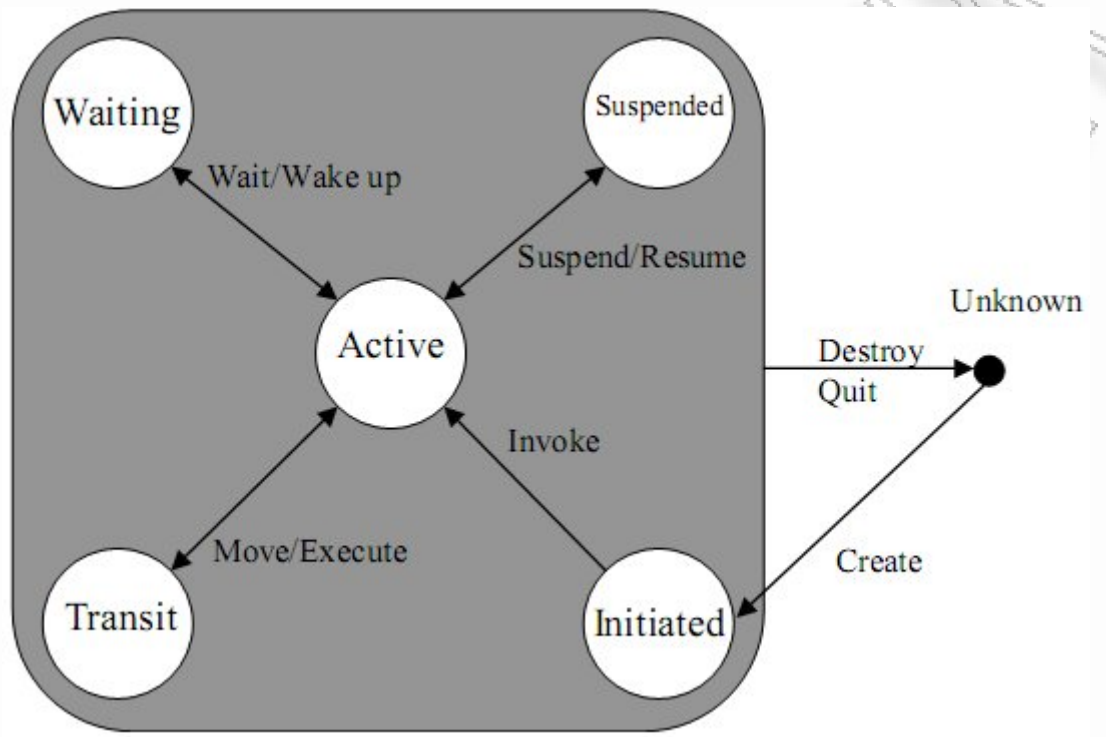
Εικόνα 27:"MTS Λειτουργία - Επικοινωνία μεταξύ πρακτόρων διαφορετικών πλατφορμών"



Εικόνα 28:"FIPA Agent platform"

13.2.2 Ο Κύκλος ζωής πράκτορα σύμφωνα με το πρότυπο FIPA

Σύμφωνα με την FIPA, κάθε πράκτορας που εισέρχεται σε μια πλατφόρμα πρακτόρων βασίζει την λειτουργία του σε έναν κύκλο ζωής. Η εικόνα 29 παρουσιάζει τον κύκλο ζωής του πράκτορα σύμφωνα με το πρότυπο της FIPA.



Εικόνα 29: "Κύκλος ζωής πράκτορα"

Βασιζόμενοι στο παραπάνω σχήμα, μπορούμε να διαπιστώσουμε ότι **οι φάσεις** του κύκλου ζωής ενός πράκτορα είναι πέντε και είναι οι εξής:

Initiated: Δημιουργία πράκτορα στην πλατφόρμα και απόδοση μοναδικού αναγνωριστικού

Active: Ο πράκτορας έχει ενεργοποιηθεί στην πλατφόρμα, έχει λάβει αναγνωριστικό και είναι ορατός στην υπηρεσία MTS.

Transit: σε αυτή την κατάσταση ο πράκτορας μπορεί να βρεθεί μόνο στην περίπτωση όπου μεταφέρεται από μία πλατφόρμα σε κάποια άλλη. Όταν ολοκληρωθεί η διαδικασία της μεταφοράς, επιτυχώς ή μη επιτυχώς, ο πράκτορας μεταβαίνει στην κατάσταση Active. Κατά την διάρκεια που ο πράκτορας βρίσκεται σε αυτή την φάση, ο MTS πράκτορας, αποθηκεύει τα μηνύματα που προορίζονται για τον πράκτορα, σε έναν buffer. Εναλλακτικά ο MTS μπορεί να προωθεί τα μηνύματα αυτά σε κάποια άλλη διεύθυνση που έχει οριστεί από τον πράκτορα πριν περάσει στην κατάσταση Transit.

Waiting /Suspended: Ο πράκτορας βρίσκεται σε προσωρινή αδράνεια μέχρις ότου ολοκληρωθεί κάποια διαδικασία που περιμένει. Ο MTS λειτουργεί όπως και στην περίπτωση που ο πράκτορας είναι σε κατάσταση Transit.

Οι μεταβάσεις που μπορεί να πραγματοποιηθούν σε έναν πράκτορα κατά τον κύκλο ζωής του είναι οι εξής:

Create: δημιουργία και εγκατάσταση ενός νέου πράκτορα στην πλατφόρμα.

Destroy: η διακοπή λειτουργίας ενός πράκτορα. Συνδέεται άμεσα με την λειτουργία του AMS ο οποίος διαγράφει τον agent από τον κατάλογο διευθύνσεων. Αποτελεί διαδικασία η οποία **δεν** μπορεί να αγνοηθεί.

Quit: ο ομαλός τερματισμός ενός πράκτορα. Μπορεί να αγνοηθεί.

Suspend: προσωρινή αναστολή της λειτουργίας του πράκτορα. Μπορεί να πραγματοποιηθεί είτε από τον AMS, είτε από τον ίδιο τον πράκτορα.

Wait: αναμονή λειτουργίας του πράκτορα. Μπορεί να πραγματοποιηθεί μόνο από τον ίδιο τον πράκτορα.

Resume: ενεργοποιείται ξανά η λειτουργία του πράκτορα μετά από κατάσταση αναμονής. Μπορεί να πραγματοποιηθεί είτε από τον AMS, είτε από τον ίδιο τον πράκτορα.

Wake up: ενεργοποιείται ξανά η λειτουργία του πράκτορα μετά από κατάσταση αναμονής. Μπορεί να πραγματοποιηθεί μόνο από τον ίδιο τον AMS.

Move: τοποθετεί τον πράκτορα σε κατάσταση μεταβίβασης. Υποστηρίζεται μόνο από τους κινητούς πράκτορες (mobile agents) και ως μετάβαση μπορεί να πραγματοποιηθεί μόνο από τον ίδιο τον πράκτορα.

Execute: επαναφέρει τον πράκτορα από μια κατάσταση μεταβίβασης. Πραγματοποιείται μόνο από τον AMS agent και ισχύει μόνο για κινητούς πράκτορες.

Παρατηρούμε λοιπόν ότι ένας πράκτορας μπορεί να βρεθεί σε διάφορες καταστάσεις κατά τον κύκλο ζωής τους, οι οποίες είναι αποτελέσματα κάποιων μεταβάσεων που μπορεί να πραγματοποιήσει. Ένα μέρος των μεταβάσεων του πράκτορα μπορεί να προκληθεί από τον AMS πράκτορα της πλατφόρμας στην οποία ανήκει, ενώ επιπλέον ο κάθε agent μπορεί να μεταβεί σε μία κατάσταση με αυτόνομο τρόπο.

13.2.3 Γλώσσα Επικοινωνίας Πρακτόρων σύμφωνα με το πρότυπο FIPA (FIPA-ACL)

Για την επικοινωνία μεταξύ των πρακτόρων εντός και εκτός μιας πλατφόρμας, απαιτείται η ύπαρξη μια κοινής γλώσσας επικοινωνίας. Η γλώσσα αυτή θα αποτελεί την γλώσσα επικοινωνίας των πρακτόρων (ACL) μέσω της οποίας θα ανταλλάσουν πληροφορίες, θα ανταλλάσουν μηνύματα εντολών και γενικότερα θα διασφαλίζει την επιτυχή αλληλεπίδραση μεταξύ των πρακτόρων. Η FIPA έχει δημιουργήσει ένα πρότυπο ACL μέσω του οποίου οι πράκτορες μιας πλατφόρμας μπορούν να αλληλεπιδράσουν επιτυχώς. Η FIPA-ACL αποτελείται από ένα σύνολο παραμέτρων που χαρακτηρίζουν την επικοινωνία των πρακτόρων. Ο πίνακας 2 παρουσιάζει όλες τις παραμέτρους που μπορούμε να εντοπίσουμε σε ένα μήνυμα FIPA – ACL.[(112)]

Παράμετρος	Κατηγορία Παραμέτρου
performative	Είδος μηνύματος
sender	Συμμετέχων στην επικοινωνία
receiver	Συμμετέχων στην επικοινωνία
reply-to	Συμμετέχων στην επικοινωνία
content	Περιεχόμενο μηνύματος
language	Περιγραφή περιεχομένου
encoding	Περιγραφή περιεχομένου
ontology	Περιγραφή περιεχομένου
protocol	Έλεγχος μηνύματος
conversation-id	Έλεγχος μηνύματος
reply-with	Έλεγχος μηνύματος
in-reply-to	Έλεγχος μηνύματος
reply-by	Έλεγχος μηνύματος

Πίνακας 2: "FIPA ACL Παράμετροι μηνύματος"

Στη συνέχεια ακολουθεί η σύντομη επεξήγηση κάθε μιας από τις παραμέτρους που μπορεί να συναντήσει κάποιος σε ένα FIPA – ACL μήνυμα.

Performative: περιλαμβάνει πληροφορίες για το είδος του μεταφερόμενου μηνύματος. Η FIPA έχει δημιουργήσει 22 διαφορετικούς τύπου μηνυμάτων οι οποίοι αναφέρονται ως επικοινωνιακές πράξεις (Communicative acts) και παρουσιάζονται εν συντομία στον πίνακα 3. Κάθε communicative act έχει αυστηρά καθορισμένο χαρακτήρα.[(113)]

[112] FIPA, **"FIPA ACL Message Structure Specification",** Αριθμός Εγγράφου: XC0061D, FIPA TC C, Switzerland, 2000, σελ. 5- 9.

[113] Βλαχάβας Ι., Κεφαλάς Π., Βασιλειάδης Ν., Κόκκορας Φ., Σακεκκαρίου Η., **«Πολυπρακτορικά Συστήματα»,** Τχνητή Νοημοσύνη, Β΄ Έκδοση, Κεφάλαιο 28, σελ. 22.

Κατηγορία	Επικοινωνιακή Πράξη (communicative act)
Μετάδοση Πληροφορίας	confirm, disconfirm, inform, inform-if, inform-ref
Αίτηση για πληροφορία	query-if, query-ref, subscribe
Διαπραγμάτευση	Accept-proposal, cfp, propose, reject-proposal
Εκτέλεση Ενεργειών	Agree, cancel, refuse, request, request-when, request0whenever
Μηνύματα Λάθους	Failure, not-understood

Πίνακας 3:"Επικοινωνιακές πράξεις στην FIPA-ACL"

Sender: περιλαμβάνει την ταυτότητα του πράκτορα - αποστολέα που έστειλε το communicative act. Σε περίπτωση που επιθυμούμε ο αποστολέας να παραμείνει ανώνυμος, αφήνουμε το πεδίο κενό, διαφορετικά εισάγεται το όνομα του agent αποστολέα.

Receiver: περιλαμβάνει την ταυτότητα του παραλήπτη της επικοινωνιακής πράξης που αποστέλλεται.

Replay-to: περιλαμβάνει την ταυτότητα του πράκτορα στον οποίο θα σταλεί απάντηση από τον παραλήπτη του communicative act. Το πεδίο μπορεί να συμπληρώνεται διότι υπάρχει περίπτωση ο παραλήπτης της απάντησης, να είναι διαφορετικός από τον αποστολέα της.

Content: η παράμετρος αυτή περιλαμβάνει το περιεχόμενο του μηνύματος.

Language: αναφέρεται η γλώσσα στην οποία έχει διατυπωθεί το περιεχόμενο του μηνύματος. Η παράμετρος παραλείπεται μόνο στην περίπτωση που η γλώσσα επικοινωνίας είναι ήδη γνωστή στον παραλήπτη.

Encoding: περιλαμβάνει την δήλωση της κωδικοποίησης των περιεχομένων

Ontology: επεξήγηση των συμβόλων που περιέχονται στο μήνυμα.

Protocol: ορίζεται το πρωτόκολλο επικοινωνίας μεταξύ αποστολέα παραλήπτη το οποίο για να οριστεί θα πρέπει να υπάρχει σταθερό conversation-id.

Conversation –id: αποτελεί το αναγνωριστικό της συνομιλίας. Κάθε συνομιλία μεταξύ πρακτόρων, χαρακτηρίζεται από ένα μοναδικό αναγνωριστικό το οποίο ενσωματώνεται στο μήνυμα με σκοπό την ταυτοποίηση του ώστε να μην μπερδεύονται τα μηνύματα στην περίπτωση που ένας πράκτορας συνομιλεί ταυτόχρονα με περισσότερους από έναν άλλους πράκτορες.

Reply-with: χρησιμοποιείται για να χαρακτηρίσει το μήνυμα που λαμβάνει ο παραλήπτης.

In-reply-to: χρησιμοποιείται για να συμπεριλάβει την έκφραση που χαρακτηρίζει το αρχικό μήνυμα που έλαβε ο παραλήπτης. Η έκφραση που ενσωματώνεται στην παράμετρο αυτή είναι ακριβώς η ίδια με την έκφραση που χαρακτήριζε το μήνυμα στην παράμετρο reply-with.

Reply-by: χρησιμοποιείται για την δημιουργία time-out στα μηνύματα που ανταλλάσσονται μεταξύ των πρακτόρων. Κάθε μήνυμα που στέλνεται, αναφέρει και το χρονικό περιθώριο απάντησης σε αυτό.

Ολοκληρώνοντας την αναφορά μας σχετικά με την FIPA-ACL σημαντικό είναι να αναφέρουμε την FIPA Semantic Language (FIPA SL) η οποία χρησιμοποιείται για την επικοινωνία των πρακτόρων με τους πράκτορες διαχείρισης της πλατφόρμας. Η FIPA SL μπορεί να περιλαμβάνει μια ολοκληρωμένη πρόταση μια ενέργεια ή μια Identifying Referential Expressions (IRE) η οποία χρησιμοποιείται για την αναγνώριση των οντοτήτων για τις οποίες μια πρόταση είναι αληθής. Η FIPA έχει δημιουργήσει τρία υποσύνολα της FIPA-SL τα οποία αναφέρονται ως εξής:

- Fipa-sl0: περιλαμβάνει αναφορές μόνο σχετικά με τις ενέργειες.
- Fipa-sl1: επεκτείνει την Fipa-sl0 προσθέτοντας στοιχεία από την άλγεβρα Boole.
- Fipa-sl2: επεκτείνει την Fipa-sl1 συμπεριλαμβάνοντας περιορισμούς αποφασιστικότητας.

Συμπερασματικά μπορούμε να αναφέρουμε ότι, η FIPA-ACL περιγράφει έναν καθορισμένο τρόπο για την δημιουργία και την ανταλλαγή των μηνυμάτων μεταξύ των πρακτόρων. Σύμφωνα με τα μηνύματα αυτά οι πράκτορες μπορούν να έχουν έναν ξεκάθαρο τρόπο επικοινωνίας και μάλιστα με τον μικρότερο δυνατό όγκο διαφορετικών τιμών στις παραμέτρους τους. Αν λάβουμε υπόψη μας την performative παράμετρο, θα διαπιστώσουμε ότι μέσω αυτής ο πράκτορας μπορεί να εξάγει συμπέρασμα σχετικά με τον λόγο που ο άλλος πράκτορας έρχεται σε επικοινωνία μαζί του. Επίσης οι πράκτορες μπορούν να συνομιλούν ταυτόχρονα με περισσότερους από έναν πράκτορες-συνομιλητές, πράγμα που καθιστά πιο αποτελεσματική την επικοινωνία τους. Τέλος κάθε πράκτορας είναι σε θέση να επικοινωνήσει με τους πράκτορες διαχείρισης της πλατφόρμας μέσω μιας ξεχωριστής προσωποποιημένης επικοινωνίας η οποία είναι αποτέλεσμα της FIPA Semantic Language (Σημασιολογική γλώσσα επικοινωνίας).

13.3 JADE Framework

Το JADE (Java Agent Development Framework) αποτελεί ένα λογισμικό (software framework) το οποίο διευκολύνει την διαδικασία ανάπτυξης εφαρμογών οι οποίες βασίζονται στην αρχιτεκτονική πρακτόρων και είναι διαμορφωμένο σύμφωνα με τα χαρακτηριστικά την FIPA για την διαλειτουργικότητα ευφυών multi-agent συστημάτων. Στόχος του JADE είναι η δημιουργία ενός εργαλείου ανάπτυξης πρακτόρων το οποίο θα διευκολύνει την προγραμματιστική διαδικασία ανάπτυξης ενός πράκτορα, χρησιμοποιώντας ένα ευρύ σύνολο από έτοιμες υπηρεσίες και πράκτορες διαχείρισης. [(114)] Για τον λόγω αυτό το JADE Framework περιέχει ένα σύνολο παρεχόμενων χαρακτηριστικών τα οποία είναι:

- **FIPA Compliant Agent Platform:** ανάπτυξη της πλατφόρμας σύμφωνα με το πρότυπο της FIPA, το οποίο περιέχει τον AMS πράκτορα (Agent Management System), τον DF πράκτορα (Directory Facilitator) και τον ACC πράκτορα (Agent Communication Channel) οι οποίοι δημιουργούνται και εκτελούνται αυτόματα με την ενεργοποίηση της JADE πλατφόρμας.
- **Distributed Agent Platform:** κατανεμημένες πλατφόρμες πρακτόρων σε διαφορετικούς hosts η κάθε μια. Κάθε σύστημα μπορεί να φιλοξενήσει μια μόνο Java Virtual Machine (JVM) και συνεπώς κάθε πλατφόρμα μπορεί να εκτελεστεί μια μόνο φορά σε ένα λειτουργικό σύστημα. Οι πράκτορες ορίζονται ως νήματα της Java (Java Threads) πάνω στην πλατφόρμα, τα οποία επικοινωνούν μεταξύ τους με Java Events. Με την χρήση των Java Threads, η πλατφόρμα δύναται να διαχειριστεί πολλές διαφορετικές διαδικασίες που μπορεί να πραγματοποιεί παράλληλα ένας πράκτορας της.
- **FIPA complaints DFs:** σε κάθε JADE πλατφόρμα μπορούν δημιουργηθούν και να λειτουργήσουν παράλληλα, περισσότεροι από ένας Directory Facilitators οι οποίοι όπως ήδη γνωρίζουμε προσφέρουν υπηρεσίες χρυσού οδηγού σε μία πλατφόρμα πρακτόρων. Οι DFs οργανώνονται και λειτουργούν σύμφωνα με το πρότυπο FIPA97.[(115)] Με την λειτουργία αυτή καθίσταται δυνατή η δήλωση υπηρεσιών σε περισσότερους από έναν DFs.
- **Connection Mechanism:** παρέχεται μηχανισμός επικοινωνίας για την αποστολή και λήψη μηνυμάτων μεταξύ πρακτόρων εντός της πλατφόρμας ενώ επιπλέον υιοθετείται το πρότυπο IIOP πρωτόκολλο, για την επικοινωνία μεταξύ πρακτόρων από διαφορετικές πλατφόρμες. Το IIOP πρωτόκολλο, αποτελεί την πρακτική εφαρμογή των αρχών που προάγει το GIOP (General Inter-ORB Protocol) και συμμορφώνεται κατάλληλα για εφαρμογή στο πρωτόκολλο TCP/IP.[(116)]
- **Automatic Registration:** αυτόματη εγγραφή των νέων πρακτόρων στον κατάλογο του AMS πράκτορα.

---

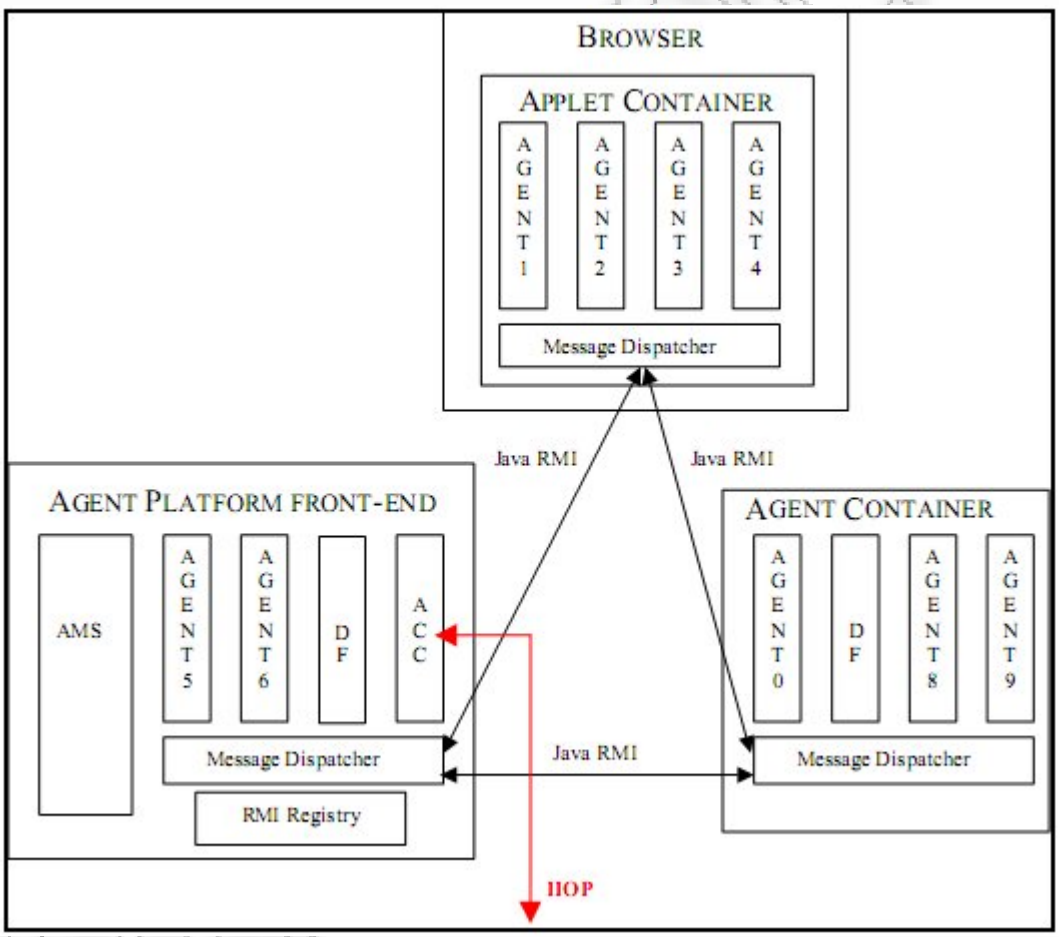
[114] Bellifemine F., Poggi A., Rimassa G., **"JADE-A FIPA-compliant agent framework",** JADE paper, CSELT S.p.A, University of Parma, Ιταλία, σελ. 3-10.

[115] FIPA 97**, "Foundation for Intelligent Physical Agents - FIPA 97 Specifications",** version 1.0, part 1, 1997.

[116] **General Inter-ORB Protocol**, http://en.wikipedia.org/wiki/General_Inter-ORB_Protocol, πρόσβαση: Μάιος 2010.

- **FIPA Interaction protocols:** βιβλιοθήκες με πρότυπα πρωτόκολλα αλληλεπίδρασης μεταξύ των πρακτόρων.
- **Naming Service:** αυτόματη καταχώρηση μοναδικού αναγνωριστικού στον πράκτορα από την πλατφόρμα όταν δημιουργηθεί και εγγραφεί σε αυτή. Το αναγνωριστικό είναι σύμφωνο με τον ειδικό τύπο ταυτοποίησης, GUID (Globally Unique Identifier).[(117)]
- **GUI:** γραφικό περιβάλλον διαχείρισης διαφορετικών πρακτόρων σε μια πλατφόρμα και διαχείρισης της πλατφόρμας.

Η αρχιτεκτονική της JADE πλατφόρμας διαμορφώνεται βάση των χαρακτηριστικών του FIPA97, σύμφωνα με την οποία είναι απαραίτητη η ύπαρξη τριών πρακτόρων διαχειριστών της πλατφόρμας, ACC, AMS, DF. Η επικοινωνία των πρακτόρων εντός και εκτός της πλατφόρμας, πραγματοποιείται μέσω της FIPA ACL.



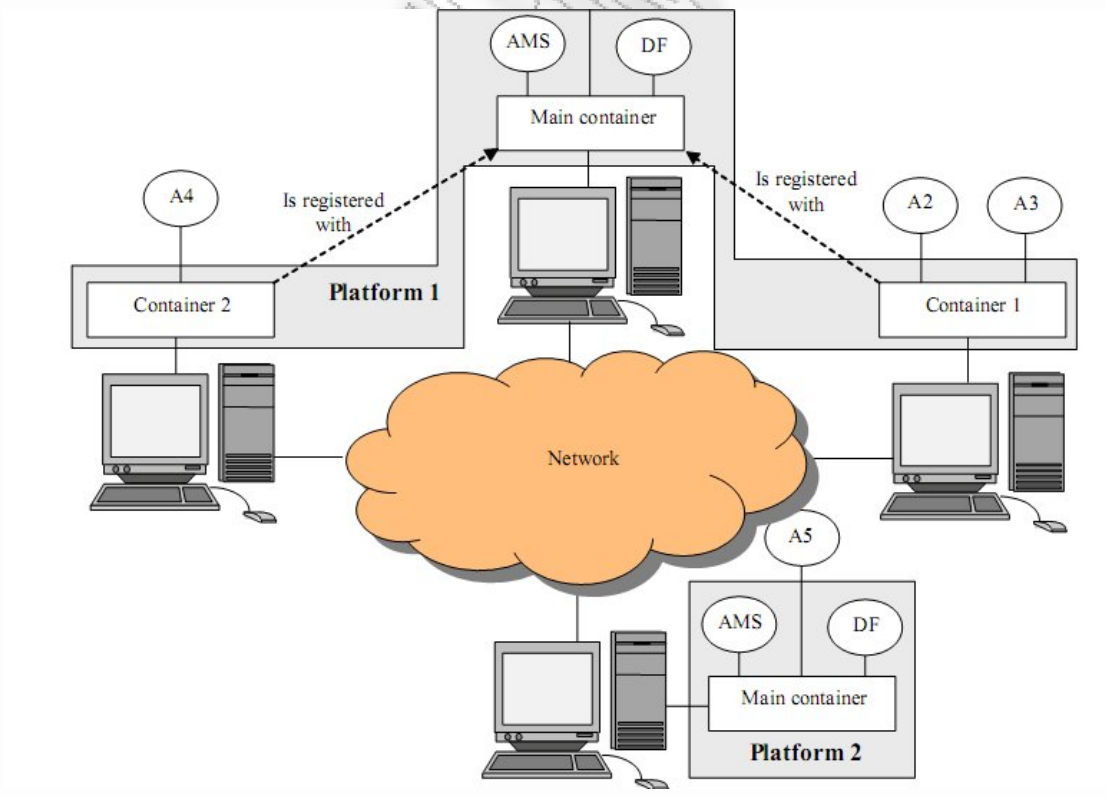
Εικόνα 30:"JADE Agent Platform - Software Architecture"

[117] **Globally Unique Identifier**, http://en.wikipedia.org/wiki/Globally_unique_identifier, πρόσβαση: Μάιος 2010.

13.3.1 JADE Containers

Κάθε τμήμα που εκτελείτε πάνω στην πλατφόρμα του JADE ονομάζεται Container και μπορεί να περιέχει περισσότερους από έναν agents. Το σύνολο από όλους τους ενεργούς JADE containers ονομάζεται Platform (Πλατφόρμα). Σε κάθε πλατφόρμα υπάρχει ο κυρίως Container ο οποίος δημιουργείται αυτόματα με την έναρξη του run-time environment και ονομάζεται Main Container . Ο main container θα πρέπει πάντα να είναι ενεργός και όλοι οι άλλοι containers (normal containers) που θα δημιουργηθούν, θα πρέπει να «συνδέονται» με αυτόν, αμέσως μόλις ξεκινήσουν. Δηλαδή ο κάθε νέος container στην πλατφόρμα, δηλώνεται στο μητρώο του main container. Όλοι οι άλλοι containers της πλατφόρμας είναι απλοί containers και θα πρέπει να δηλώνουν που βρίσκετε ο Main Container (host : port) στον οποίο είναι συνδεδεμένοι.

Εάν σε κάποιον διαφορετικό host εκκινηθεί το runt-time environment του JADE, τότε αμέσως δημιουργείται ένας νέος Main Container στον οποίο δηλώνονται οι απλοί Containers (normal containers) που θα δημιουργηθούν στον host αυτό. Με την δημιουργία νέου Main Container σε διαφορετικό host, δημιουργείται αυτόματα μια νέα JADE πλατφόρμα, η οποία φέρει τα δικά της μοναδικά χαρακτηριστικά και μπορεί να είναι ενεργή μόνο στον host όπου ξεκίνησε. Οι JADE πλατφόρμες που δημιουργούνται σε διαφορετικούς hosts μπορούν να επικοινωνούν μεταξύ τους μέσω ενός κοινού δικτύου στο οποίο έχουν πρόσβαση (π.χ. το internet). Η εικόνα 31 παρουσιάζει ένα παράδειγμα επικοινωνίας 2 διαφορετικών πλατφορμών που έχουν δημιουργηθεί σε ξεχωριστούς hosts.



Εικόνα 31:"Επικοινωνία μεταξύ διαφορετικών JADE Platforms"

13.3.2 JADE πακέτα υλοποίησης πλατφόρμας πρακτόρων

Η JADE μέσα από το περιβάλλον εκτέλεσης της, παρέχει ένα σύνολο από πακέτα που περιέχουν κλάσεις και υπό-πακέτα (sub-packages) για την υλοποίηση εφαρμογών που βασίζονται στην τεχνολογία πρακτόρων λογισμικού. Το σύνολο αυτών των πακέτων είναι τα εξής:[(118)]

- **jade.content:** πακέτο που περιέχει κλάσεις οι οποίες παρέχουν μεθόδους δημιουργίας οντολογιών και γλωσσών περιεχομένου, ώστε ο προγραμματιστής να αναπτύσσει τις δικές του οντολογίες και τα δικά του ACL μηνύματα, βάση των αναγκών της εφαρμογής που αναπτύσσει.

- **jade.core:** παρέχει τις κλάσεις υλοποίησης της πλατφόρμας. Διαθέτει κλάσεις που ενεργοποιούν το περιβάλλον εκτέλεσης της πλατφόρμας. Επιπλέον διαθέτει την κλάση Agent η οποία αποτελεί την κλάση μέσω της οποίας φτιάχνονται οι πράκτορες. Παρέχει ένα σύνολο από προ-δημιουργημένες συμπεριφορές του πράκτορα οι οποίες χρησιμοποιούνται για να βοηθήσουν στην εκτέλεση σημαντικών λειτουργιών στην πλατφόρμα.

- **jade.domain:** το πακέτο αυτό περιλαμβάνει τις οντότητες και τα χαρακτηριστικά που ορίζονται από την FIPA για την διαμόρφωση και την λειτουργία της πλατφόρμας. Περιέχει τις οντότητες διαχείρισης της πλατφόρμας οι οποίες είναι i) ο AMS πράκτορας που προσφέρει υπηρεσία white pages στην πλατφόρμα και ii) ο DF πράκτορας που παρέχει υπηρεσία yellow pages προς την πλατφόρμα. Επιπλέον το πακέτο περιέχει υπό-πακέτα διαχείρισης των πρακτόρων της πλατφόρμας.

- **jade.lang.acl:** πακέτο διαχείρισης ACL μηνυμάτων.

- **jade.proto:** περιλαμβάνει όλα τα διαθέσιμα πρωτόκολλα αλληλεπίδρασης της FIPA.

- **jade.util:** διαθέτει ένα σύνολο από κλάσεις διαχείρισης γεγονότων στην πλατφόρμα και επεκτείνεται με το πακέτο **leap**, το οποίο προάγει λειτουργίες για την εισαγωγή χαρακτηριστικών στην J2ME, τα οποία δεν υποστηριζόντουσαν εξαρχής.

- **jade.wrapper:** το πακέτο wrapper σε συνδυασμό με τα πακέτα jade.core.Profile και jade.core.Runtime, παρέχει την δυνατότητα σε εξωτερικά προγράμματα της java, να χρησιμοποιήσουν στοιχεία από το JADE.

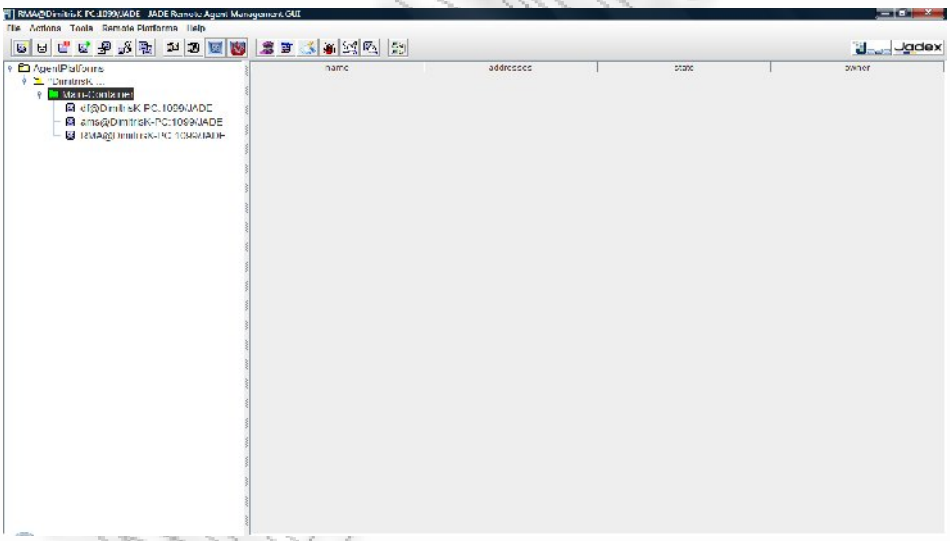
Η αναλυτική περιγραφή των πακέτων που παρέχονται από το JADE δίνετε από το σχετικό JADE API, όπου περιγράφονται αναλυτικά οι λειτουργίες κάθε

[118] **JADE V.4.0 API**, http://jade.tilab.com/doc/api/index.html, πρόσβαση: Μάιος 2010.

κλάσης. Συμπερασματικά θα μπορούσαμε να πούμε ότι το JADE παρέχει ένα σύνολο πακέτων υλοποίησης προς τον προγραμματιστή, μέσω των οποίων ο χρήστης μπορεί να αναπτύξει με σχετική ευκολία εφαρμογές που βασίζονται στην τεχνολογία πρακτόρων, ενώ μπορεί να τις διαχειριστεί μέσα από το επίσης προσφερόμενο γραφικό περιβάλλον του JADE.

13.3.3 JADE και εργαλεία διαχείρισης

Η JADE παρέχει ένα σύνολο εργαλείων για την ευκολότερη διαχείριση της κάθε πλατφόρμας πρακτόρων. Η χρήση των εργαλείων διαχείρισης γίνεται εξαιρετικά απλή για τους χρήστες, μέσω του γραφικού περιβάλλοντος της JADE Platform. Η JADE παρέχει ειδικά διαμορφωμένο γραφικό περιβάλλον, για την εξ αποστάσεως διαχείριση, παρακολούθηση και τον έλεγχο της θέσης των πρακτόρων. Το JADE GUI υλοποιείται ως ξεχωριστός Agent RMA. Ο **Remote Management Agent** παρέχει το γραφικό περιβάλλον για τη διαχείριση και τον έλεγχο των φορέων και των διαμεσολαβητών σε μια πλατφόρμα. Μέσω αυτού δίνεται η δυνατότητα να δημιουργηθούν καινούργιοι διαμεσολαβητές, να ελεγχθούν οι διαμεσολαβητές της πλατφόρμας και να ενεργοποιηθούν τα διάφορα εργαλεία διαχείρισης της πλατφόρμας. Η εικόνα 32 παρουσιάζει την απεικόνιση του γραφικού περιβάλλοντος της JADE Platform,σε κατάσταση εκκίνησης της πλατφόρμας.



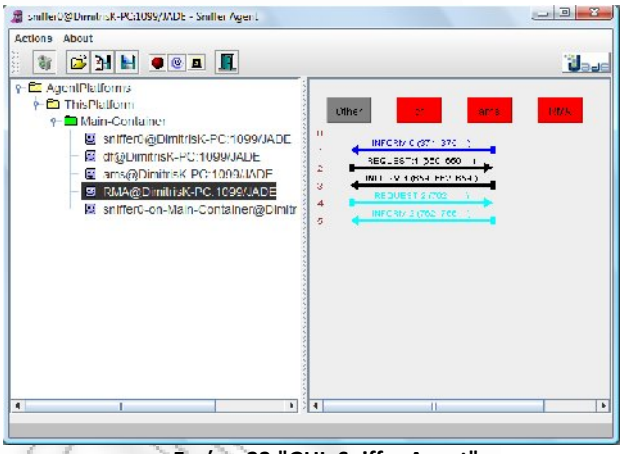
Εικόνα 32:"JADE RMA GUI"

Όπως ήδη αναφέραμε η JADE διαθέτει κάποια εργαλεία διαχείρισης της πλατφόρμας και των πρακτόρων που ανήκουν σε αυτή. Τα σημαντικότερα εργαλεία διαχείρισης που παρέχει η πλατφόρμα είναι: i) ο **Sniffer Agent**, ii) ο **Socket Proxy Agent** , iii) ο **Dummy Agent** και iv) ο **DF agent** με GUI. Ακολουθεί μια σύντομη αναφορά στην λειτουργία που προσφέρει κάθε εργαλείο.[(119)(120)]

[119] Bellifemine F., Poggi A., Rimassa G., Turci P., **"An object-oriented framework to realize agent systems"**, CSELT S.p.A, University of Parma, Ιταλία, σελ. 3-4.

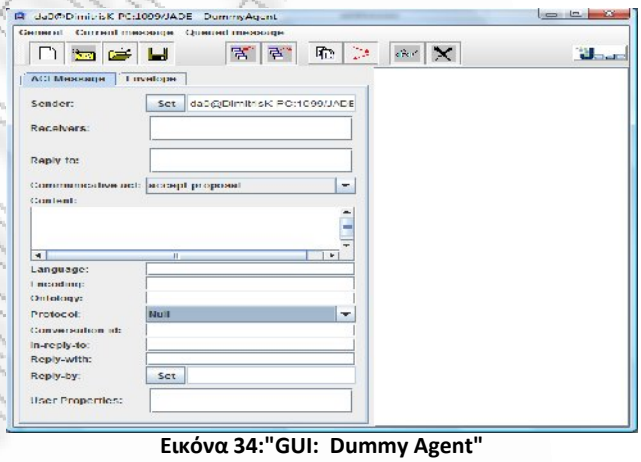
[120] Bellifemine F., **"Java Agent Development Framework – what it is and what it is next",** Telecom Italia Lab – Torino (Italy), ETAPS 2001, Παρουσίαση, 2001, σελ. 15.

Sniffer Agent: είναι ο πράκτορας που έχει ως στόχο την παρακολούθηση των μηνυμάτων που ανταλλάσσονται μεταξύ των διαμεσολαβητών στην πλατφόρμα και είναι αρκετά χρήσιμος για τον έλεγχο της καλής λειτουργίας των προγραμμάτων.



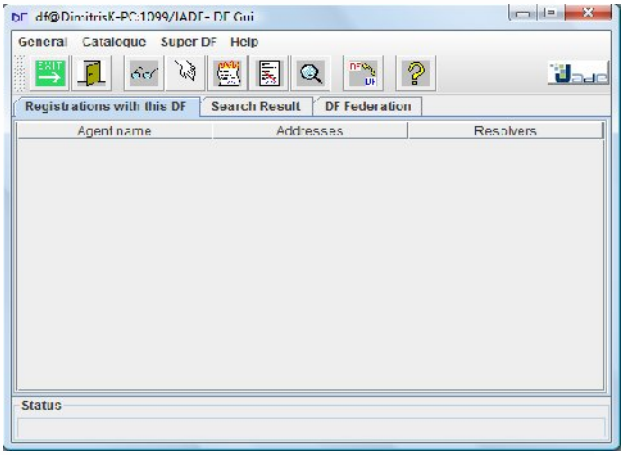
Εικόνα 33:"GUI: Sniffer Agent"

Dummy Agent: όπως δηλώνει και το όνομα του είναι ένας «ανόητος» πράκτορας, ο οποίος το γραφικό του περιβάλλον προσφέρει τη σύνταξη μηνυμάτων, την αποστολή μηνυμάτων και στη συνέχεια την παραλαβή των απαντήσεων τους από τους διαμεσολαβητές στους οποίους εστάλησαν τα μηνύματα.



Εικόνα 34:"GUI: Dummy Agent"

DF agent with GUI: παρέχει το γραφικό περιβάλλον στο οποίο ο χρήστης μπορεί να δει όλες τις εγγραφές του DF και να τις διαχειριστεί. Συγκεκριμένα κάποιος μπορεί να προσθέσει καινούριες εγγραφές, να μετατρέψει τις υπάρχουσες ή να τις διαγράψει και να αναζητήσει τον διαμεσολαβητή που συμφωνεί με την περιγραφή του χρήστη. Τέλος, το πιο σημαντικό ίσως είναι η σύνδεση του με άλλους DF, μέσω των οποίων θα εμπλουτίσει τη γνώση του με καινούριες υπηρεσίες και διαμεσολαβητές καθώς όλες οι αιτήσεις που θα φτάνουν για αναζήτηση υπηρεσιών θα προωθούνται στους γνωστούς του DF.



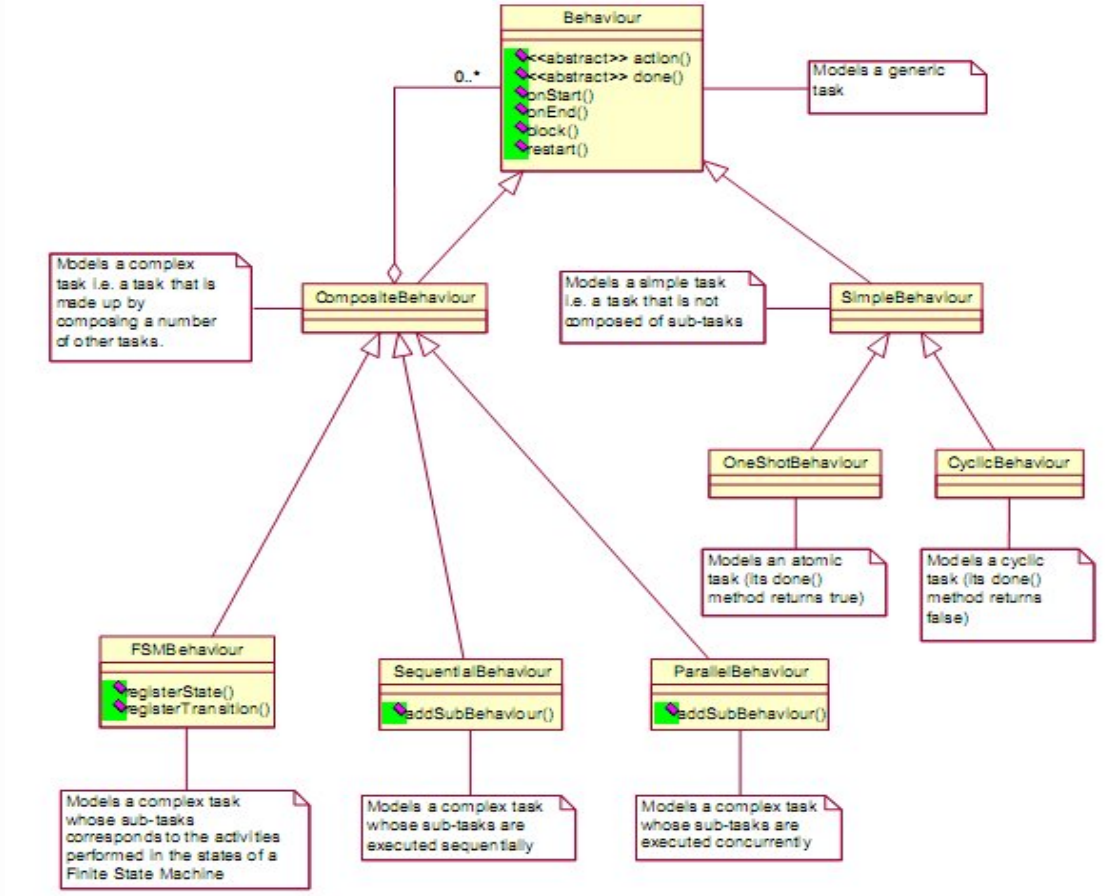
Εικόνα 35:"GUI: Directory Facilitator agent"

Socket Proxy Agent: είναι ο πράκτορας, όπου μέσω της λειτουργίας του μετατρέπει τα μηνύματα (που κινούνται στην πλατφόρμα) σε συμβολοσειρά με τη δομή ενός ACL μηνύματος και στη συνέχεια τα προωθεί μέσω ενός socket προς τους τελικούς τους αποδέκτες.

13.3.4 JADE Agents & Behaviors

Οι πράκτορες στο JADE διαμορφώνονται σύμφωνα με τα πρότυπα της FIPA. Κάθε πράκτορας με την έναρξή του στην πλατφόρμα, λαμβάνει ένα μοναδικό αναγνωριστικό, το Agent ID και στη συνέχεια εγγράφεται στην υπηρεσία white pages του Agent Management System. Ο πράκτορας προκειμένου να εκτελέσει μια συγκεκριμένη λειτουργία θα πρέπει, αμέσως μετά την μέθοδο δημιουργίας του, να έχει προστεθεί μια μέθοδος behavior. Ο agent εν συνεχεία θα πρέπει να δηλώσει τις υπηρεσίες που προσφέρει στον Directory Facilitator agent ο οποίος παρέχει υπηρεσίες yellow pages στην πλατφόρμα. Σε περίπτωση που ένας agent τερματίσει την λειτουργία του, θα πρέπει να διαγραφεί από την υπηρεσία του DF. Επιπλέον η διαγραφή του από τον AMS πράκτορα, πραγματοποιείται αυτόματα, όπως και η εγγραφή του.

Ο κάθε agent έχει την δυνατότητα να εκτελεί παράλληλα διαφορετικές διαδικασίες και να ανταποκρίνεται σε πολλά εξωτερικά ερεθίσματα. Η δυνατότητα αυτή αποδίδεται με την εισαγωγή της κλάσης behavior η οποία μέσω των λειτουργιών της, καθιστά ικανό τον agent ώστε να ανταποκρίνεται στα ερεθίσματα που δέχεται από το περιβάλλον του. Υπάρχουν διαφορετικές κατηγορίες behavior οι οποίες μπορούν να προστεθούν σε έναν πράκτορα και εκτελούνται διαδοχικά η μια μετά την άλλη. Η ιεράρχηση των κατηγοριών παρουσιάζονται στην εικόνα 36 και στη συνέχεια ακολουθεί μια σύντομη ανάλυση τους.



Εικόνα 36:"UML: Διάγραμμα ιεράρχησης behavior που συναντάμε σε έναν JADE Agent"

Σύμφωνα με την παραπάνω εικόνα, διαπιστώνουμε ότι υπάρχουν δύο **κύριες κατηγορίες συμπεριφοράς**: i) **SimpleBehaviour** και ii) **CompositeBehaviour**. Η κάθε κατηγορία συγκεντρώνει επιμέρους υποκατηγορίες συμπεριφορών. Ακολουθεί η παρουσίαση και η σύντομη ανάλυσή τους. [(121)]

- **SimpleBehaviour:** ως κατηγορία συμπεριλαμβάνει εκείνες τις συμπεριφορές οι οποίες εκτελούνται μια μόνο φορά και τερματίζουν με την ολοκλήρωσή τους.[(122)]

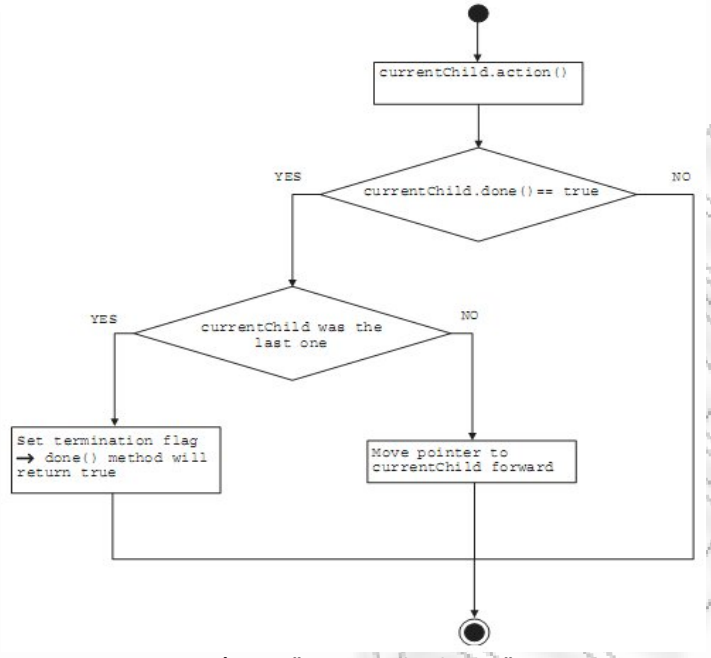
    - **OneShotBehaviour:** η κλάση αυτή περιλαμβάνει την μέθοδο action() η οποία, εάν ολοκληρωθεί μία φορά τότε θεωρείται ολοκληρωμένη η συγκεκριμένη συμπεριφορά. Ουσιαστικά η συγκεκριμένη συμπεριφορά προγραμματίζεται ώστε να εκτελεστεί για μια και μόνο μια φορά.

    - **CyclicBehaviour:** οι κυκλικές συμπεριφορές δεν φέρουν κάποιο σημείο ολοκλήρωσης και κάθε φορά που καλούνται, εκτελούν την ίδια ακριβώς συμπεριφορά. Χρησιμοποιούνται συνήθως σε εφαρμογές επαναλαμβανόμενων διαδικασιών, (πχ: η εγγραφή ενός νέου κόμβου σε ένα τηλεπικοινωνιακό σύστημα).

- **CompositeBehaviour:** συμπεριλαμβάνει τις σύνθετες συμπεριφορές οι οποίες αποτελούνται από επιμέρους συμπεριφορές οι οποίες συνδυάζονται με σκοπό την κάλυψη των αναγκών της εφαρμογής. [(123)]

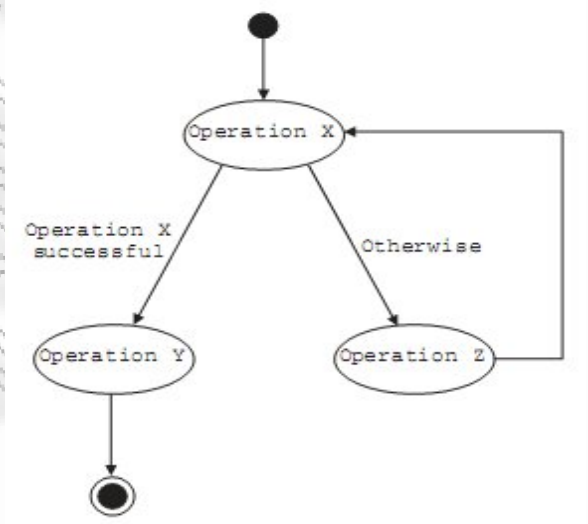
    - **SequentialBehaviour:** η κλάση αυτή εφαρμόζει ένα πολύπλοκο μοντέλο οργάνωσης των διαδικασιών μιας συμπεριφοράς. Δημιουργεί ένα χρονοδιάγραμμα εκτέλεσης για κάθε κόμβο και η εξέλιξη της διαδικασίας βασίζεται σε μια απλή πολιτική διαδοχικής εκτέλεσης. Όταν ολοκληρωθεί η διεργασία 1, συνεχίζει με την διαδικασία 2, κ.ο.κ. Κάθε υπό-συμπεριφορά στην Sequential Behavior,ορίζεται σύμφωνα με τις ανάγκες της εφαρμογής και προγραμματίζεται εξ' αρχής η σειρά εκτέλεσής της. Η εικόνα 37 παρακάτω παρουσιάζει μια ενδεικτική μορφή μιας Sequential συμπεριφοράς.

[121] Bellifemine F., Caire G., Trucco T., Rimassa G., **"JADE PROGRAMMER'S GUIDE"**, TILAB, CSELT, University of Parma, Ιταλία, last update: Απρίλιος, 2010, σελ. 23-26.
[122] Caire G., **"JADE Programming for beginners",** TILAB, JADE 3.7, 2009, σελ. 12.
[123] Bellifemine F., Caire G., Greenwood D., **"Developing multi-agent systems with JADE"**, Wiley Series, 2007, σελ. 92-100.



Εικόνα 37:"Sequential Behavior"

- **FSMBehaviour:** παρέχει μεθόδους για την καταγραφή των επιμέρους συμπεριφορών ως συστήματα πεπερασμένων ενεργειών (*Finite State Machine - FSM*). Όπως και στα FSM συστήματα έτσι και στην FSM συμπεριφορά υπάρχει ένα συγκεκριμένο σχέδιο ενεργειών που μπορούν να εκτελεστούν, ενώ παράλληλα καταγράφεται και η διαδικασία μετάβασης από μια κατάσταση σε μία άλλη.[(124)] Όπως και στην σειριακή συμπεριφορά έτσι και εδώ θα πρέπει να αρχικοποιούμε έναν δείκτη στην αρχική κατάσταση του πεπερασμένου συστήματος. Κάθε φορά που ολοκληρώνεται μια κατάσταση, ο δείκτη μεταβαίνει στην επόμενη κατάσταση η οποία μπορεί: i) να επιστρέφει στην αρχική κατάσταση, ii) να εκκινεί μια νέα κατάσταση του συστήματος ή iii) να τερματίζει το σύστημα.



Εικόνα 38:"Τυπική λειτουργία ενός FSM συστήματος"

[124] **Finite State Machine**,http://en.wikipedia.org/wiki/Finite-state_machine, πρόσβαση: Μάιος 2010.

- **ParallelBehaviour:** πρόκειται για μια κλάση απόδοσης συμπεριφοράς στον πράκτορα, η οποία παρέχει μεθόδους παράλληλου χρονοπρογραμματισμού των κόμβων που ανήκουν στην συμπεριφορά. Στην κλάση αυτή ανήκει η μέθοδος action(), η οποία όταν καλείται, πραγματοποιεί την εκτέλεση της συμπεριφοράς που επιλέχθηκε. Με την εκκίνηση της action() μεθόδου, ο δείκτης μετακινείται αμέσως στην επόμενη προγραμματισμένη συμπεριφορά προς εκτέλεση, ανεξάρτητα με το αν ολοκληρώθηκε ή όχι η προηγούμενη συμπεριφορά. Μια συμπεριφορά parallel, μπορεί να θεωρηθεί ολοκληρωμένη όταν όλα της τα παιδιά (υπό- συμπεριφορές) ολοκληρωθούν, είτε μόνο αν ολοκληρωθεί το πρώτο της παιδί. Η ολοκλήρωση της parallel συμπεριφοράς προγραμματίζεται από τον προγραμματιστή.

Μέσω του JADE λοιπόν παρέχεται η δυνατότητα ανάπτυξης multi-agent συστημάτων σύμφωνα με τα πρότυπα ανάπτυξης της FIPA. Η χρήση στάνταρ προτύπων για την ανάπτυξη μιας πλατφόρμας πρακτόρων αλλά και η χρήση της FIPA-ACL και FIPA-SL για την επικοινωνία των πρακτόρων, διευκολύνει τον προγραμματισμό τους. Επίσης η διαχείριση του συστήματος, γίνετε ευκολότερη με την χρήση των εργαλείων διαχείρισης μέσα από το γραφικό περιβάλλον που πλαισιώνει τον JADE RMA πράκτορα. Έτσι θα μπορούσαμε να πούμε ότι το JADE αποτελεί μια από τις πιο ισχυρές και ταυτόχρονα εύχρηστες εφαρμογές ανάπτυξης συστημάτων που βασίζουν την λειτουργία τους σε πράκτορες, ενώ επίσης αποτελεί και ισχυρό εργαλείο για την διαχείριση των συστημάτων αυτών. Ωστόσο επειδή η μορφή του JADE που εξετάσαμε στην ενότητα αυτή, δεν υποστηρίζει την επέκταση του σε κινητές συσκευές όπως τα PDA (Personal Digital Assistant), θα αναφερθούμε, στην επόμενη ενότητα, σε μια διαφορετική έκδοση του JADE η οποία έχει λιγότερες λειτουργικές απαιτήσεις και είναι ιδανική για την εφαρμογή της σε κινητές συσκευές. Η έκδοση αυτή της JADE ονομάζεται JADE-LEAP και αναλύεται στην συνέχεια.

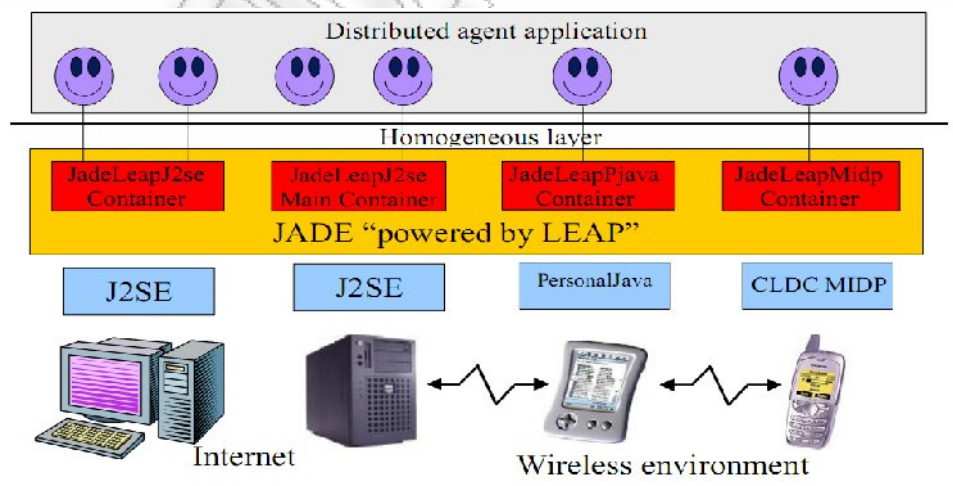
13.4 JADE Lightweight Extensible Agent Platform

Το JADE Lightweight Extensible Agent Platform (στο εξής JADE LEAP) αποτελεί την μορφή του JADE που εφαρμόζεται σε κινητές συσκευές. Επειδή η κανονική μορφή του JADE μπορεί να εφαρμοστεί σε συσκευές με υψηλότερες λειτουργικές δυνατότητες από τις απλές κινητές συσκευές όπως τα PDA, αναπτύχθηκε μια μορφή του η οποία είναι εφαρμόσιμη σε περιβάλλοντα κινητών συσκευών. Το JADE LEAP αποτελεί μια πιο ελαφριά έκδοση της JADE η οποία μπορεί να χρησιμοποιηθεί σε ένα ευρύ φάσμα φορητών συσκευών.

Η αρχική χρήση της LEAP ήταν ξεχωριστή από το JADE και αποτέλεσε την πρώτη απόπειρα εφαρμογής προτύπων της FIPA για την κατασκευή πλατφόρμας πρακτόρων. Στη συνέχεια η LEAP ενσωματώθηκε στο JADE και οδηγηθήκαμε στο JADE LEAP όπου στην πραγματικότητα επεκτείνει το JADE. Το JADE LEAP μπορεί να εφαρμοστεί σε τρία διαφορετικά περιβάλλοντα της JAVA, ανάλογα με το είδος της συσκευής στην οποία εγκαθίσταται. Συγκεκριμένα τα Java run-time environments είναι τα εξής:[(125)(126)]

- **J2SE:** java 2 standard edition, για υπολογιστές και server's οι οποίοι διαθέτουν έκδοση JDK 1.2 (Java Development Kit) ή νεότερη.
- **PJAVE:** personal java, για συσκευές PDA.
- **MIDP:** για κινητά τηλέφωνα που διαθέτουν την java.

Η έκδοση του JADE LEAP για το J2SE αποτελεί ταυτόσημη έκδοση με του κανονικού JADE. Πράκτορες που έχουν αναπτυχθεί προγραμματιστικά σε JADE LEAD μπορούν να απεικονιστούν και σε JADE χωρίς να υπάρξει αλλαγή στις γραμμές κώδικα. Ωστόσο είναι σημαντικό να γνωρίζουμε ότι οι containers του JADE LEAP δεν μπορούν να συνυπάρξουν σε μια πλατφόρμα με τους containers του JADE. Οι άλλες δύο μορφές (PJAVA, MIDP) φέρουν τα ίδια χαρακτηριστικά και προάγουν τις ίδιες λειτουργίες, με την διαφορά ότι στην περίπτωση του MIDP δεν υποστηρίζονται κάποιες επιπρόσθετες ιδιότητες.



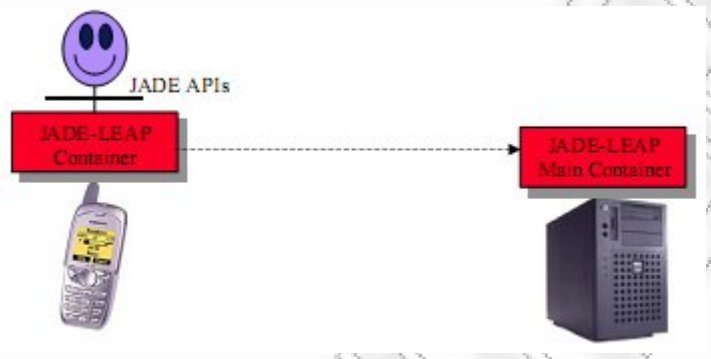
Εικόνα 39: "Εφαρμογή JADE LEAP σε τρία διαφορετικά περιβάλλοντα εκτέλεσης java"

[125] Moreno A., Valls A., Viejo A. , **"Using JADE-LEAP to implement agents in mobile devices"**, GruSMA, ETSE, URV, σελ. 2.

[126] Caire G.,**"LEAP USER GUIDE – Usage restricted according to license agreement"**, TILAB ex CSELT, 2003, σελ. 2-4.

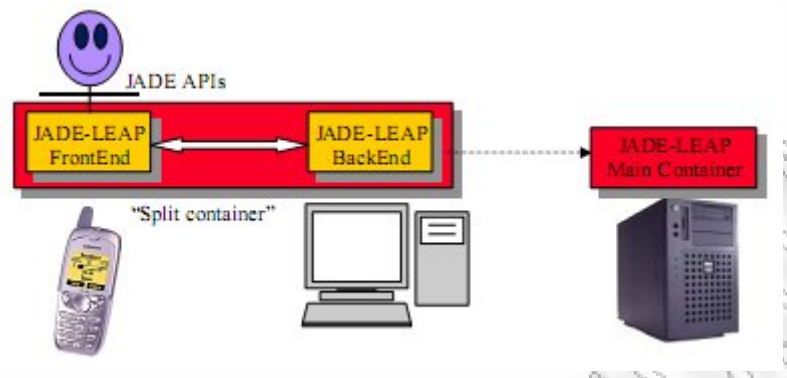
Σημαντικό είναι να επισημάνουμε ότι η διαδικασία εκτέλεσης του JADE-LEAP σε μια συσκευή χειρός, μπορεί να εκτελεστεί με δύο τρόπους. Αυτοί είναι:

- **Stand alone εκτέλεση**: κατά την οποία δημιουργείται ένας πλήρης container ο οποίος εκτελείται εξολοκλήρου στην πλατφόρμα την φορητής συσκευής. Ο main-container εκτελείται σε έναν κεντρικό υπολογιστή και σε αυτόν εντάσσεται ο container της φορητής συσκευής.



Εικόνα 40:"Stand alone execution"

- **Split εκτέλεση:** όπου ο container που αποδίδεται στην φορητή συσκευή, μοιράζεται σε δύο περιβάλλοντα εκτέλεσης όπου το ένα περιλαμβάνει την φορητή συσκευή και του αποδίδεται η ονομασία JADE-LEAP FrontEnd και το δεύτερο περιλαμβάνει έναν J2SE host (πχ: ένας ηλεκτρονικός υπολογιστής) ο οποίος φέρει την ονομασία JADE-LEAP BackEnd και συνδέεται άμεσα με τον JADE-LEAP FrontEnd. Οι JADE-LEAP FrontEnd & JADE-LEAP BackEnd αποτελούν στο σύνολό τους τον container που αποδίδεται στην φορητή συσκευή. Ο διαχωρισμός αυτός προκειμένου να πραγματοποιηθεί και να αποδώσει, θα πρέπει να υπάρχει μια ασύρματη σύνδεση μεταξύ του host και της φορητής συσκευής. Επιπλέον η διαδικασία έναρξης του container, στην συσκευή, είναι ταχύτερη από την περίπτωση του Stand alone execution ενώ το γεγονός ότι ο main container έρχεται σε επαφή μόνο με την πλευρά JADE-LEAP BackEnd, βοηθά στην περίπτωση όπου η φορητή συσκευή αλλάξει IP, ώστε να μην επηρεάσει την λειτουργία του συστήματος. Αλλάζοντας IP ουσιαστικά αλλάζει τα στοιχεία της μόνο στον JADE-LEAP BackEnd ενώ το υπόλοιπο σύστημα δεν επιβαρύνεται διότι επικοινωνεί με την IP του host όπου εκτελείται ο JADE-LEAP BackEnd. Συμπερασματικά λοιπόν μπορούμε να αναφέρουμε ότι η split execution ενδείκνυται, ως ελαφρύτερη έκδοση, για φορητές συσκευές με μειωμένες λειτουργικές δυνατότητες.



Εικόνα 41:"Split execution"

Γενικότερα μπορούμε να διαπιστώσουμε ότι το JADE-LEAP αναπτύχθηκε με σκοπό την επέκταση της JADE σε συστήματα με περιορισμένους ή αδύναμους πόρους. Στόχος είναι η δημιουργία πρακτόρων οι οποίοι μπορούν να συνθέσουν ένα multi-agent σύστημα και λειτουργούν αποκλειστικά σε φορητές συσκευές. Επιπλέον η εγκατάσταση του JADE LEAP σε συστήματα που χρησιμοποιούν την J2SE επιχειρεί να ενισχύσει την επέκταση του JADE σε συσκευές οι οποίες έχουν ιδιαίτερα χαμηλές δυνατότητες (πχ: επεξεργαστική ισχύς, μνήμη, κλπ.) και στοχεύει στην διασφάλιση των δεδομένων επικοινωνίας.

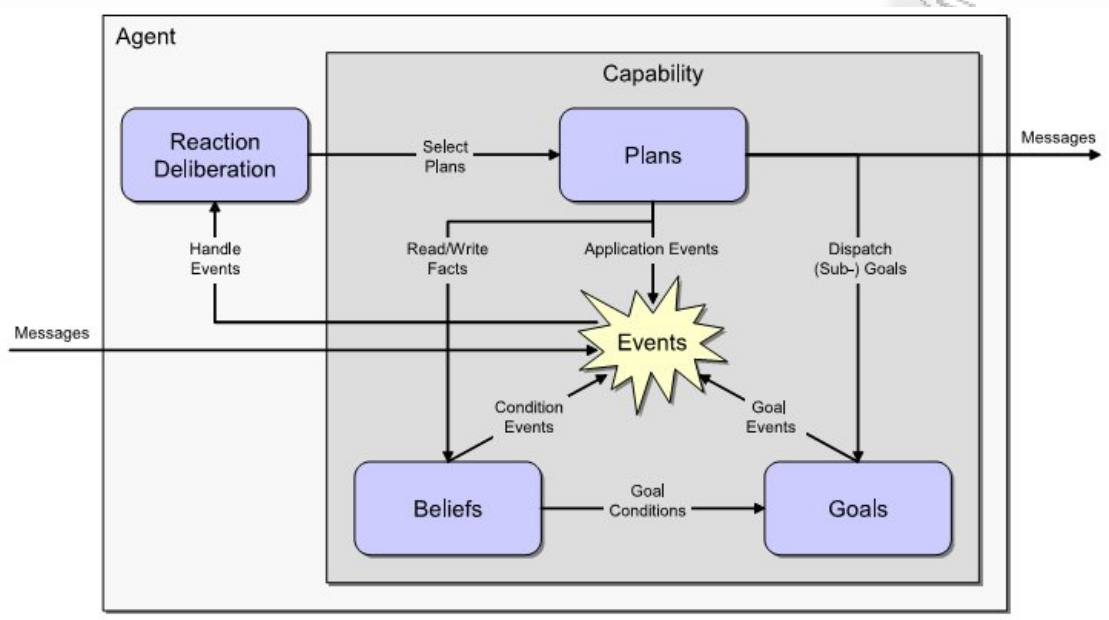
Κλείνοντας σημαντικό είναι να αναφέρουμε ότι υπάρχουν κάποιες βασικές διαφορές μεταξύ JADE και JADE LEAP οι οποίες δεν έχουν καλυφθεί ακόμη. Οι διαφορές αυτές προκύπτουν κυρίως λόγω περιορισμών που οφείλονται στις λειτουργικές δυνατότητες των φορητών συσκευών και περιλαμβάνουν: i) την έλλειψη υποστήριξης γραφικού περιβάλλοντος (JADE-GUI), ii) την αδυναμία μετανάστευση του πράκτορα και iii) την αδυναμία κλωνοποίησης του πράκτορα. Ωστόσο η JADE-LEAP αποτελεί ένα πολύ σημαντικό βήμα για την επέκταση της JADE και την δημιουργία multi-agent συστημάτων που θα περιλαμβάνουν μεγάλο εύρος συσκευών.

13.5 JADEX Framework

Το JADEX βασίζεται στο μοντέλο Belief Desire Intention (BDI) και χρησιμοποιεί τα λεγόμενα Agent Definition Files (ADF) τα οποία αναπτύσσονται σε XML ώστε να προωθηθεί η βελτίωση του βαθμού ελαστικότητας στην επίτευξη των στόχων τους. Οι JADEX Agents δεν είναι απλώς, εφαρμογές που στέλνουν και λαμβάνουν μηνύματα, αλλά στην πραγματικότητα αποτελούν σύνθετες μονάδες λογισμικού που λειτουργούν πάνω σύμφωνα με το BDI μοντέλο και περιέχουν **capabilities**, **beliefs**, **goals**, **plans** και **events.**[(127)] Η εικόνα 42 στη συνέχεια αποτελεί την απεικόνιση της αρχιτεκτονικής ενός JADEX Agent. Στη συνέχεια ακολουθεί η

[127] Alexander Pokahr, Lars Braubach, Winfried Lamersdorf, "**A BDI REASONING ENGINE**", University of Hamburg,σελ. 3-6.

περιγραφή των στοιχείων που συνθέτουν το BDI μοντέλο και αποτελούν τα βασικά δομικά στοιχεία της αρχιτεκτονικής ενός JADEX Agent.



Εικόνα 42: Αρχιτεκτονική JADEX Agent

13.5.1 Beliefs

Στα BDI συστήματα ένα βασικό χαρακτηριστικό είναι η ύπαρξη στοιχείων περιγραφής της οντότητας του πράκτορα, κάτι το οποίο υλοποιείται μέσω των beliefs του agent. Τα beliefs αποτελούν μια object oriented τεχνική απεικόνισης-αποθήκευσης δεδομένων, τα οποία αποθηκεύονται είτε ως μεμονωμένα στοιχεία και ονομάζονται **fact** (beliefs), είτε αποθηκεύονται μαζικά και ονομάζονται **set of facts** (beliefsets). Προκειμένου να προσπελάσουμε τα δεδομένα που αποθηκεύονται μέσω των beliefs, χρησιμοποιούμε μια τεχνική προσανατολισμένη στην γλώσσα ερωτημάτων. Τα beliefs εκτός από αντικείμενα παθητικής αποθήκευσης δεδομένων, συμμετέχουν ενεργά στην εκτέλεση του πράκτορα με την χρήση ειδικών μεταβλητών που καλούνται «συνθήκες» (conditions) και με την χρήση τους δίνουν την δυνατότητα σε έναν πράκτορα να ελέγχει διάφορα τμήματα της λειτουργίας του. Επιπλέον η αλλαγή στις τιμές των beliefs, μπορεί να χρησιμοποιηθεί ως ερέθισμα για την αλλαγή κάποιων συνθηκών ή μπορεί να εκκινήσει κάποιους στόχους του agent.

13.5.2 Goals

Οι στόχοι στους BDI Agents αποτελούν κεντρική έννοια στην οποία στηρίζει ένα σύνολο από τις δράσεις του ο κάθε πράκτορας. Οι στόχοι χρησιμοποιούνται συχνά ώστε να επιδιώξει ένας πράκτορας την λήψη ενός αποτελέσματος. Οι στόχοι συχνά ορίζονται ως εκτελέσιμοι με κριτήρια. Πιο συγκεκριμένα ένας στόχος μπορεί να εκτελείται συνεχώς έως ότου ο πράκτορας λάβει το επιθυμητό αποτέλεσμα, ή μέχρι να διαπιστώσει ότι δεν υπάρχει πιθανότητα επίτευξής τους. Οι στόχοι είναι προσπελάσιμοι από τα πλάνα ενός πράκτορα η από άλλους στόχους που ισχύουν γενικά για τους BDI agent (πχ: DF Register, κλπ.). Όταν ικανοποιηθεί η επιλεγμένη

συνθήκη, ο στόχος παύει να εκτελείτε και θεωρείται επιτυχής. Επίσης η σημαντική διαφορά του JADEX σε σχέση με άλλα BDI συστήματα, είναι ότι δεν θεωρείται απαραίτητη η επίτευξη ενός στόχου του πράκτορα, ώστε αυτός να περάσει στον επόμενο. Αντιθέτως υπάρχει συγκεκριμένη αντιστοιχία στόχων με ενέργειες που εκτελεί ο πράκτορα και η μη επίτευξη ενός στόχο συνεπάγεται, συνήθως, την αποτυχία μιας λειτουργίας και όχι όλου του συνόλου. Το JADEX υποστηρίζει τέσσερεις διαφορετικούς τύπους στόχων οι οποίοι επιδεικνύουν διαφορετική συμπεριφορά κατά την επεξεργασία τους. Οι τέσσερεις διαφορετικοί στόχοι είναι: i) **perform** ii) **achieve** iii) **query** iv) **maintain**. Οι perform στόχοι συνδέονται άμεσα με την εκτέλεση των δράσεων ενός πράκτορα και επομένως θεωρούνται επιτυχείς μόνο στην περίπτωση που έχουν εκτελεστεί κάποιες από τις προβλεπόμενες ενέργειες, ανεξάρτητα με το αποτέλεσμα τους. Οι achieve στόχοι αποτελούν την περιγραφή των παραδοσιακών στόχων ενός πράκτορα όπου για να θεωρούνται επιτυχείς, θα πρέπει να επιτευχτεί το τελικό αποτέλεσμα, ανεξάρτητα από τον τρόπο που θα συμβεί αυτό. Οι query στόχοι είναι παρόμοιοι με τους achieve στόχους, αλλά η επιθυμητή κατάσταση δεν είναι μια συνθήκη του εξωτερικού κόσμου του agent. Η συνθήκη που επιχειρεί αν ικανοποιήσει ο στόχος αφορά μια εσωτερική κατάσταση του πράκτορα σχετικά με κάποιες διαθέσιμες πληροφορίες που θέλει να γνωρίζει. Οι στόχοι maintain χρησιμοποιούνται από έναν πράκτορα σε περιπτώσεις οπου πρέπει ο πράκτορας αν παρακολουθεί μια κατάσταση και προκειμένου να την διατηρήσει σε μια συγκεκριμένη φάση, θα πρέπει να εκτελεί συνεχώς τα κατάλληλα σχέδια δράση.

13.5.3 Plans

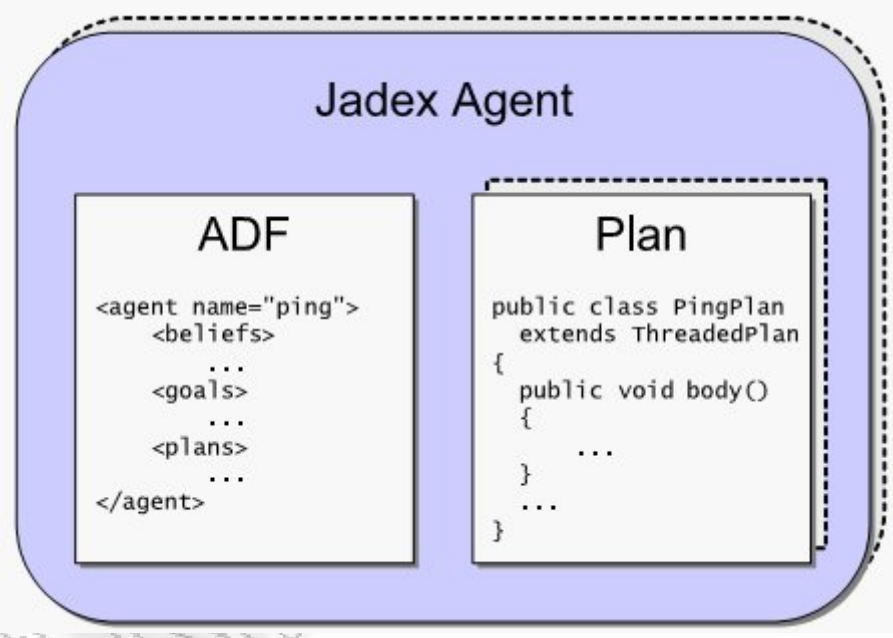
Τα πλάνα αντιπροσωπεύουν τα στοιχεία συμπεριφοράς ενός πράκτορα και αποτελούνται από δύο τμήματα το header & το body. Τα στοιχεία που συμπεριλαμβάνονται στον header ενός πλάνου, προσδιορίζουν τις περιστάσεις υπό τις οποίες μπορεί ένα πλάνο να εκτελεστεί. Συμπεριλαμβάνουν τις συνθήκες ή τα γεγονότα που μπορεί να οδηγήσουν τον πράκτορα στην εκτέλεση λειτουργιών που περιγράφονται από το πλάνο. Το σώμα του πλάνου περιγράφει μια συγκεκριμένη πορεία δράσης του πράκτορα, η οποία θα εκτελεστεί εφόσον επιλεγεί το συγκεκριμένο πλάνο λειτουργίας. Συνήθως στο σώμα ενός πλάνου συμπεριλαμβάνεται η κλάση του προγράμματος που πρέπει να εκτελεστεί από τον πράκτορα ενώ τα αποτελέσματα εκτέλεσης ενός πλάνου αφορούν συχνά, την αποστολή/λήψη μηνυμάτων, την διαχείριση των beliefs ή την δημιουργία subgoals.

13.5.4 Capabilities

Οι δυνατότητες ενός πράκτορα, υλοποιούν έναν μηχανισμό ομαδοποίησης κοινών στοιχείων όπως πλάνα, στόχοι, γεγονότα, τα οποία αποτελούν βασικά συστατικά των BDI Agents. Με τον τρόπο αυτό μπορούμε να δημιουργούμε έτοιμα μοντέλα σχεδιασμού και εκτέλεσης πρακτόρων, χωρίς να χρειάζεται κάθε φορά να επαναλαμβάνουμε την διαδικασία ανάπτυξης για πράκτορας μς κοινά χαρακτηριστικά. Παράλληλα οι capabilities χρησιμοποιούνται από του πράκτορες ώστε να συμπεριλάβουν ένα σύνολο εξωτερικών ιδιοτήτων οι οποίες θα τους βοηθήσουν στην ολοκλήρωση κάποιων σημαντικών διαδικασιών. Χαρακτηριστικό παράδειγμα αποτελεί η υιοθέτηση του Directory Facilitator capability file, το οποίο περιέχει όλες τις απαραίτητες BDI συνιστώσες που είναι απαραίτητες για την

αλληλεπίδραση του πράκτορα με του πράκτορες DF και AMS. Τέλος, σημαντική θεωρείται η ικανότητα κλήσης συγκεκριμένων στοιχείων από ένα capability αρχείο. Συγκεκριμένα ένας πράκτορας που κάνει import ένα capability file, έχει την δυνατότητα να καλέσει και να χρησιμοποιήσει μεμονωμένα BDI στοιχεία (πχ: beliefs, goals, κλπ.).

Οι BDI συνιστώσες που περιγράψαμε παραπάνω, συγκεντρώνονται σε ένα xml αρχείο που ονομάζεται Agent Definition File (ADF). Το ADF αρχείο σε συνδυασμό με τα αρχεία κώδικα που υλοποιούν την εκτέλεση των διαδικασιών του πράκτορα, αποτελούν τα δύο απαραίτητα συστατικά για την σύνθεση των JADEX Agents. Για την σύνταξη του ADF xml αρχείου, ακολουθούμε την το BDI Meta-Model που ορίζεται από το αντίστοιχο XML Schema. Επίσης οι δυνατότητες συγγραφής του ADF, αυξάνονται με την ενσωμάτωση μιας περιγραφικής γλώσσας έκφρασης. Από την άλλη πλευρά το διαδικαστικό κομμάτι του προγραμματισμού που δημιουργεί τα αρχεία κώδικα, πραγματοποιείται στη γλώσσα προγραμματισμού java. Η εικόνα 43 απεικονίζει με απλό τρόπο τον συνδυασμό των αρχείων που συνθέτουν έναν πράκτορα.



Εικόνα 43: Η υλοποίηση ενός JADEX Agent

✓ *Ανακεφαλαιώνοντας*

Στο παρόν κεφάλαιο αναφερθήκαμε σε τεχνολογίες που θα συναντήσουμε κατά την υλοποίηση της Service Oriented Virtualization Platform. Αναφερθήκαμε αρχικά στα Multi-agent συστήματα, αναλύοντας την έννοια του agent ως οντότητα ενός συστήματος και στην συνέχεια παρουσιάσαμε διάφορες αρχιτεκτονικές πρακτόρων που μπορούν να δομήσουν ένα τέτοιο σύστημα. Στην δεύτερη ενότητα παρουσιάσαμε το πρότυπο FIPA για την δημιουργία συστημάτων πρακτόρων, και περιγράψαμε αναλυτικά τα χαρακτηριστικά ενός multi-agent συστήματος το οποίο διαμορφώνεται σύμφωνα με το πρότυπο FIPA.

Έχοντας ήδη ενημερωθεί για το πρότυπο FIPA, παρουσιάσαμε μια από τις σημαντικότερες εφαρμογές για την ανάπτυξη συστημάτων πρακτόρων που υιοθετεί το πρότυπο και οργανώνεται σύμφωνα με αυτό. Αναφερθήκαμε στην JADE τεχνολογία και στα χαρακτηριστικά οργάνωσης και λειτουργίας της. Αναλύσαμε σε ικανοποιητικό βαθμό την JADE-LEAP η οποία αποτελεί την εξέλιξη της JADE για την εφαρμογή της σε φορητές συσκευές με μικρή διαθεσιμότητα πόρων. Κλείνοντας αναφερθήκαμε στο JADEX Framework το οποίο αποτελεί την επέκταση του JADE, για την δημιουργία πρακτόρων οι οποίοι βασίζουν την λειτουργία τους στο μοντέλο BDI και περιγράφονται από ένα σύνολο συνιστωσών, beliefs, goals, plans και events.

✓ **Κεφάλαιο 14: «**Περιγραφή υλοποίησης της SOVP**»**

14.1 SOVP Multi-agent System

Η SOVP πλατφόρμα υλοποιήθηκε σύμφωνα με τον αρχιτεκτονικό σχεδιασμό που περιγράφηκε στα κεφάλαια 11 και 12. Η SOVP υλοποιείται ως ένα **κατανεμημένο πολύ-πρακτορικό σύστημα προσανατολισμένο σε υπηρεσίες** το οποίο ορίζεται από ξεχωριστές αυτόνομες μονάδες λογισμικού που καλούνται πράκτορες (agents) και έχουν αναπτυχθεί βάσει της JADE/JADEX τεχνολογίας (JADEX Agents). Η πλατφόρμα του JADE και του JADEX αποτελούν την βάση λειτουργίας του συστήματος μέσω της οποίας οι πράκτορες που δημιουργούνται, μπορούν να ανταλλάξουν μηνύματα και γενικότερα να αλληλεπιδράσουν μεταξύ τους. Η συγκέντρωση πολλών διαφορετικών JADEX Agents στο σύστημα της SOVP, οδηγεί στην ανάπτυξη ενός πραγματικού multi-agent συστήματος το οποίο οργανώνεται αποκεντρωτικά, όπως ακριβώς ένα κατανεμημένο σύστημα. Ο κάθε πράκτορας που δημιουργείται και εισάγεται στο σύστημα αποτελεί μια αυτόνομη οντότητα που διαθέτει όλα τα απαραίτητα στοιχεία ώστε να διαχειριστεί τις λειτουργικές του δυνατότητες. Επίσης οι JADEX Agents που εισάγονται στο σύστημα μπορούν να επικοινωνήσουν μεταξύ τους μέσω ACL μηνυμάτων τα οποία ενσωματώνονται σε FIPA πρωτόκολλα μεταφοράς που είναι συμβατά με την JADEX πλατφόρμα όπου πάνω δομείται το σύστημα της SOVP.

14.2 SOVP Service Oriented System

Στο κεφάλαιο 11 περιγράψαμε την αρχιτεκτονική της SOVP, αναφέροντας ότι θα αναπτυχθεί ως ένα Service Oriented σύστημα το οποίο συγκεντρώνει διαφορετικές γνωστικές υπηρεσίες από τις επιμέρους συνιστώσες του.

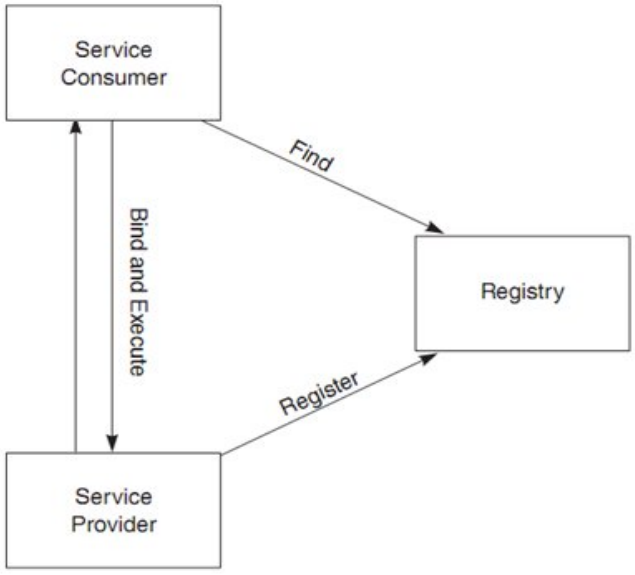
Από την υλοποίηση της εφαρμογής εύκολα μπορεί να αντιληφθεί κάποιος ότι η πλατφόρμα διαμορφώνεται ως ένα Service Oriented σύστημα. Αιτιολογώντας αυτό που αναφέρουμε σημειώνουμε τα εξής. Η SOVP αποτελεί ένα Service Oriented σύστημα διότι υπακούει στους κανόνες της SOA[(128)], ενώ επίσης οι αλληλεπιδράσεις των συνιστωσών, ακολουθούν το πρότυπο αλληλεπίδρασης οντοτήτων της SOA[(129)].

Πιο συγκεκριμένα στο κεφάλαιο 5, ενότητα 5.3, περιγράφουμε ακριβώς τους τέσσερεις βασικούς κανόνες που πρέπει να τηρούνται, ώστε να θεωρηθεί ένα σύστημα ότι υπακούει στις αρχές της SOA. Ο πίνακας 4 παρουσιάζει τους κανόνες αυτούς, παράλληλα με τα χαρακτηριστικά που φέρει η SOVP. Μελετώντας προσεκτικά τον πίνακα εύκολα μπορούμε να διαπιστώσουμε ότι η SOVP ως ένα αυστηρά δομημένο σύστημα, ανταποκρίνεται απόλυτα στις τέσσερεις βασικές συνθήκες που προάγουν οι αρχές της SOA. Επιπλέον το σχήμα 12 προβάλει την βασική δομή αλληλεπίδρασης των οντοτήτων ενός Service Oriented συστήματος. Στην περιγραφή των αλληλεπιδράσεων μεταξύ των συνιστωσών της SOVP, θα διαπιστώσουμε ότι εφαρμόζεται αποκλειστικά το σχέδιο αλληλεπίδρασης της SOA.

[128] βλ. παρ. 41, **σελ.** 20
[129] βλ. παρ.38, **σελ.** 19.

SOA Basic Rules	SOVP Characteristics	Compliance
Ανταλλαγή Περιγραφικών και όχι ενημερωτικών μηνυμάτων μεταξύ των συνιστωσών	Μηνύματα που αιτούνται υπηρεσίες ή παρέχουν δεδομένα στις υπηρεσίες. Όχι μηνύματα που περιγράφουν την διαδικασία εκτέλεσης των υπηρεσιών.	✔
Συγκεκριμένο πρότυπο επικοινωνίας και δομή μηνυμάτων	FIPA ACL μηνύματα τα οποία μεταφέρονται μέσω της JADEX πλατφόρμας.	✔
Επεκτάσιμη αρχιτεκτονική συστήματος	Εύκολη προσθήκη υπηρεσιών και εφαρμογών. Plug and Play διαδικασία που επιτρέπει την εύκολη προσθήκης συνιστωσών στο σύστημα και συνεπώς την επέκτασή του.	✔
Μηχανισμός συστήματος για την αποθήκευση και ανακάλυψη υπηρεσιών μέσω αναζήτησης	Δημοσίευση υπηρεσιών και ανακάλυψη μέσα στον Directory Facilitator της JADE πλατφόρμας. Η SOVP χρησιμοποιώντας τα κατάλληλα πλάνα λειτουργίας δίνει την δυνατότητα σε κάθε συνιστώσα της να μπορεί να κάνει δημοσίευση και αναζήτηση υπηρεσιών μέσω του DF Agent.	✔
Συνεργασία διαφορετικών οντοτήτων για την ολοκλήρωση συγκεκριμένων διεργασιών	Οι βασικές κατηγορίες οντοτήτων που εντοπίζονται στην τρέχουσα υλοποίηση της SOVP εκτελούν όλες τους τις διεργασίες, κατόπιν συνεργασίας με άλλες οντότητες, (πχ: Ο CAP όταν εντοπίσει Congestion δεν μπορεί να την αντιμετωπίσει και προκειμένου να ολοκληρώσει την διαδικασία της αποσυμφόρησης του, συνεργάζεται με τον CAP MA).	✔

Πίνακας 4: Service Oriented Χαρακτηριστικά της SOVP



Σχήμα 12: «Αλληλεπίδραση Οντοτήτων Service Oriented συστήματος»

14.3 SOVP συνιστώσες του συστήματος

Κάθε JADEX Agent αποτελεί μια νέα συνιστώσα του συστήματος. Οι συνιστώσες του συστήματος μπορεί να είναι: i) **στοιχεία δικτυακής υποδομής**, ii) **συσκευές χρηστών** και iii) **υποδομές λογισμικού**. Οι συνιστώσες τύπου i και ii εισάγονται στο σύστημα με δυναμικό τρόπο μετά από την φόρτωση του αντίστοιχου ADF στην πλατφόρμα. Οι συνιστώσες που χαρακτηρίζονται ως δομές λογισμικού, ξεκινούν την λειτουργία τους όπως η **i** και **ii**, αλλά διαφέρουν από αυτές ως προς το ότι φορτώνονται αυτόματα με την εκκίνηση της πλατφόρμας.

Τα στοιχεία δικτυακής υποδομής μπορεί να είναι ένα σύνολο από **Cognitive Access Points (CAP)** τα οποία συγκεντρώνουν πλήθος διαφορετικών τεχνολογιών πρόσβασης και υπηρεσίες (πχ: Flexible Base Station με τρείς διαφορετικές τεχνολογίες πρόσβασης GPRS, UMTS, WiMAX). Οι συσκευές χρηστών μπορεί να είναι ένα σύνολο από **Cognitive Reconfigurable Devices (CRD)** τα οποία μπορούν να λειτουργήσουν είτε αυτόματα (πχ: NetBook Wi-Fi 802.11 a/b/g) είτε με την παρέμβαση του ανθρώπινου παράγοντα (πχ: PDA, Smart Phones, κλπ.). Οι υποδομές λογισμικού που έχουμε υλοποιήσει και εισάγει στην SOVP, συμπεριλαμβάνουν, για την τρέχουσα υλοποίηση, **οντότητες διαχείρισης των στοιχείων της πλατφόρμας** που χρησιμοποιούν ένα σύνολο γνωστικών υπηρεσιών μέσω των οποίων αναπτύσσουν την λειτουργία τους. Στη συνέχεια θα περιγράψουμε αναλυτικά την κάθε κατηγορία συνιστωσών από αυτές που προαναφέραμε.

Κάθε κατηγορία συνιστωσών, μπορεί να περιγραφεί ως μια ξεχωριστή οντότητα η οποία απεικονίζει όλα τα στοιχεία που είναι απαραίτητα για την περιγραφή και την λειτουργία της συνιστώσας, στην πλατφόρμα. Η τωρινή υλοποίηση της SOVP συμπεριέλαβε πληροφορίες οι οποίες αναφέρονται στο προφίλ και στο περιεχόμενο των συνιστωσών (**Profile** and **Context**). Επιπλέον στις κατηγορία των συνιστωσών που αναφέρονται ως υποδομές λογισμικού, δημιουργήσαμε τους κατάλληλους Agents οι οποίοι μπορούν να λάβουν τις παραπάνω πληροφορίες , μπορούν να τις διαχειριστούν και επίσης μπορούν να εξάγουν συμπεράσματα τα οποία θα βοηθήσουν στην αναπροσαρμογή της λειτουργίας των συνιστωσών (Reconfiguration). Από την στιγμή αυτή και στο εξής, προκειμένου να ορίσουμε με ακρίβεια την υλοποίηση της SOVP, θα περιγράψουμε τέσσερεις (4) διαφορετικές κατηγορίες JADEX πρακτόρων, οι οποίοι υλοποιήθηκαν και συνθέτουν μια πλήρη εικόνα του συστήματος μας. Οι κατηγορίες παρουσιάζονται στο πίνακα 5 παρακάτω. Ακολουθεί η περιγραφή υλοποίησης για κάθε πράκτορα της πλατφόρμας.

ΚΑΤΗΓΟΡΙΑ ΣΥΝΙΣΤΩΣΑΣ	ΚΑΤΗΓΟΡΙΑ ΠΡΑΚΤΟΡΑ	ΟΝΟΜΑ ΠΡΑΚΤΟΡΑ ΣΤΗΝ ΠΛΑΤΦΟΡΜΑ
Δικτυακές Υποδομές	Cognitive Access Point Agents	CAP Agents
Συσκευές Χρηστών	Cognitive Reconfigurable Device Agent	CRD Agent
Λογισμικό	Generators Agents	Service Load Generator Agent CRD Status Generator Agent
Λογισμικό	Agent Managers	CAP Manager Agent CRD Manager Agent

Πίνακας 5: Κατηγορίες και ονόματα πρακτόρων της SOVP

14.3.1 CAP Agents

Οι Cognitive Access Points Agents, υλοποιούνται με σκοπό να συμπεριλάβουν ένα μεγάλο εύρος υλικού δικτυακών υποδομών. Κάθε CAP Agent περιέχει πληροφορίες για το προφίλ του (**Profile Information**) και για το περιεχόμενο του (**Context Information**). Επίσης κατά όταν ο πράκτορας συνδεθεί στην πλατφόρμα, την ίδια στιγμή δηλώνει την παρουσία του σε αυτή, δημοσιεύοντας μια υπηρεσία παρουσίας, στον Directory Facilitator της JADEX πλατφόρμας.

Στο προφίλ του πράκτορας περιέχονται πληροφορίες που περιγράφουν την δομή του CAP ως τμήμα δικτυακής υποδομής. Συγκεκριμένα υπάρχουν πληροφορίες σχετικά με τους διαθέσιμους πομποδέκτες του CAP (TRXs) οι οποίοι προσδιορίζονται από ένα μοναδικό ID, μπορούν να λειτουργήσουν σε συγκεκριμένες συχνότητες και μπορούν να υποστηρίξουν ένα συγκεκριμένο σύνολο τεχνολογιών ασύρματης πρόσβασης (RATs).

Το περιεχόμενο του πράκτορα περιέχει πληροφορίες που περιγράφουν το τρέχον φορτίο σε κάθε ενεργό πομποδέκτη του CAP. Το φορτίο αναλύεται, επιπλέον, σε φορτίο υπηρεσιών (Services) ανά κλάση χρηστών (User Class) και επίπεδο ποιότητας (QoS). Το σύνολο αυτών των πληροφοριών αναφέρεται ως **Service Load** και περιέχεται μέσα σε κάθε TRX Context μαζί με το TRX ID, που αποτελεί την ταυτότητα του TRX, και το Aggregate Load που περιγράφει το τρέχον φορτίο χρηστών στο TRX.

Οι πληροφορίες του CAP Profile και του CAP Context διατίθενται στα υπόλοιπα στοιχεία της πλατφόρμας (που μπορούν να αλληλεπιδράσουν με τους συγκεκριμένους Agents) με την βοήθεια δύο βασικών γνωστικών υπηρεσιών που διαθέτει ο κάθε CAP Agent. Οι υπηρεσίες είναι οι εξής:

- Υπηρεσία **διάθεσης πληροφοριών του προφίλ** σε πράκτορες που αλληλεπιδρούν με τον CAP
- Υπηρεσία **διάθεσης πληροφοριών του περιεχομένου** σε πράκτορες που αλληλεπιδρούν με τον CAP

Εκτός των δύο υπηρεσιών που αναφέραμε, ο CAP Agent διαθέτει ακόμη μια γνωστική υπηρεσία η οποία είναι υπεύθυνη **για την δυναμική παρακολούθηση της λειτουργίας του CAP**. Πρόκειται για μια υπηρεσία η οποία δημιουργεί αυτόνομη λειτουργία στον πράκτορα, στο κομμάτι του ελέγχου της λειτουργίας. Συγκεκριμένα η υπηρεσία αυτή είναι αρμόδια για την παρακολούθηση του φορτίου σε κάθε TRX Profile του CAP. Σε περίπτωση που εντοπίσει συμφόρηση σε κάποιο από τα διαθέσιμα ενεργά TRX, ενεργοποιεί απευθείας τις διαδικασίες Reconfiguration στοχεύοντας στην αποσυμφόρηση. Η υπηρεσία ενεργοποιείται κάθε στιγμή που υπάρχει μεταβολή στο CAP Context. Δεν έχει κάποιο χρονικό περιορισμό, που σημαίνει ότι μπορεί να εκτελείται ακόμη και κάθε δευτερόλεπτο λειτουργίας του πράκτορα.

Η ονομασία κάθε υπηρεσίας, η περιγραφή της και η κατηγοριοποίηση της βάση των υπηρεσιών της πλατφόρμας, δίνονται στον πίνακα 6 παρακάτω.

SOVP Cognitive Services Category	CAP Manager Agent Services Name (JAVA Class)	CAP Manager Agent Services Description
Profile Services	ProfilePlan.class	Διάθεση πληροφοριών προφίλ CAP
Context Services	CAPContextPlan.class	Διάθεση πληροφοριών περιεχομένου CAP
(Self-x) Monitoring Services	CAPMonitoringPlan.class	Δυναμική παρακολούθηση κατάσταση λειτουργίας CAP

Πίνακας 6: CAP Agent Cognitive Services

Σημαντικό είναι να επισημάνουμε ότι σύνολο των πληροφοριών που εμπεριέχονται στο CAP Context, θα δούμε στη συνέχεια ότι, χρησιμοποιείται από τον CAP Manager ως πηγή πληροφοριών για τις γνωστικές υπηρεσίες του Decision Making τα αποτελέσματα των οποίων οδηγούν στο Reconfiguration του CAP Agent. Επίσης οι πράκτορες που ανήκουν στην κατηγορία CAP Agents, μπορούν να αλληλεπιδράσουν μόνο με τον CAP Manager Agent.

Τέλος να αναφέρουμε ότι για την λειτουργία της πλατφόρμας έχουμε δημιουργήσει δύο πρότυπα πρακτόρων τύπου CAP οι οποίοι ονομάζονται CAP_1 και CAP_2, και το προφίλ του παρουσιάζεται στον πίνακα 7 παρακάτω.

AGENT NAME	TRX IDs	TRX FREQBAND	NUNBER OF RATs per TRX	RATs IDs	SERVICES
CAP_1	TRX_1 TRX_2	800-900 1900-2100	**TRX_1(**#1**)** **TRX_2(**#2**)**	**TRX_1(**RAT_1**)**, **TRX_2(**RAT_1,RAT_2**)**	**TRX_1(**VOICE**)** **TRX_2(**VOICE, DATA,AUDIO STREAM, VIDEO CALL, EMAIL**)** **TRX_3(**EMAIL, INTERNET, RADIO, WiFi**)**
CAP_2	TRX_1 TRX_2 TRX_3	800-900 1200-2100 3000-10000	**TRX_1(**#1**)** **TRX_2(**#2**)** **TRX_3(**#3**)**	**TRX_1(**RAT_1**)**, **TRX_2(**RAT_1,RAT_2**)** **TRX_3(**RAT_1,RAT_2,RAT_3**)**	**TRX_1(**#1**)** **TRX_2(**#2**)** **TRX_3(**#3**)**

Πίνακας 7: Πρότυπα CAP Profiles που χρησιμοποιούνται στη SOVP

14.3.2 CRD Agents

Η υλοποίηση των Cognitive Reconfigurable Device Agents, συμπεριλαμβάνει ένα μεγάλο εύρος συσκευών χρηστών οι οποίες μπορούν να λειτουργήσουν στην SOVP πλατφόρμα. Όπως και στους CAP Agents έτσι και στους CRD Agents, περιέχονται πληροφορίες σχετικά με το προφίλ και το περιεχόμενο της συσκευής. Επίσης όπως ο CAP Agent έτσι και ο CRD Agent, όταν συνδεθεί στην πλατφόρμα, απευθείας δηλώνει την παρουσία του σε αυτή, δημοσιεύοντας μια υπηρεσία παρουσίας, στον Directory Facilitator της JADEX πλατφόρμας.

Οι πληροφορίες του προφίλ αναφέρονται σε τρία βασικά χαρακτηριστικά: i) το σύνολο των υπηρεσιών που μπορεί η συγκεκριμένη συσκευή να εξυπηρετήσει ανεξάρτητα με του περιορισμούς του χρήστη (πχ: η συσκευή υποστηρίζει Video Call αλλά ο χρήστης δεν είναι συνδρομητής της υπηρεσίας), ii) οι διαθέσιμοι πομποδέκτες που φέρει η συσκευή με τις συχνότητες και τα RATs στα οποία μπορεί να λειτουργήσει ο κάθε πομποδέκτης (TRX Profiles), και iii) τα προφίλ των χρηστών που μπορούν να χρησιμοποιήσουν την συσκευή τα οποία περιέχουν πληροφορίες σχετικά με την περιγραφή και τις προτιμήσεις τους.

Το περιεχόμενο του CRD, διαθέτει πληροφορίες οι οποίες σχετίζονται με την τρέχουσα κατάσταση της συσκευής (του πράκτορα). Οι υπηρεσίες, και τα TRXs που είναι ενεργά κάθε στιγμή πάνω στο τερματικό, μπορούν αν παρουσιαστούν μέσα από το CRD Context, ενώ παράλληλα, μπορούν να σταλούν μέσω συμβατού μηνύματος, προς τον CRD Manager. Όπως θα δούμε στη συνέχεια, ο CRD Manager μπορεί να χρησιμοποιήσει αυτές τις πληροφορίες (Profile & Status) κατά την αλληλεπίδραση του με το τερματικό.

Ο CRD Agent χρησιμοποιεί δύο γνωστικές υπηρεσίες ώστε να διαθέσει τις πληροφορίες που σχετίζονται με το Profile και το Status του προς τον CRD Manager. Οι υπηρεσίες είναι οι εξής:

- Υπηρεσία **διάθεσης πληροφοριών του προφίλ** σε πράκτορες που αλληλεπιδρούν με τον CRD
- Υπηρεσία **διάθεσης πληροφοριών του περιεχομένου** σε πράκτορες που αλληλεπιδρούν με τον CRD

Η ονομασία κάθε υπηρεσίας, η περιγραφή της και η κατηγοριοποίηση της βάση των υπηρεσιών της πλατφόρμας, δίνονται στον πίνακα 7 παρακάτω.

SOVP Cognitive Services Category	CRD Manager Agent Services Name (JAVA Class)	CRD Manager Agent Services Description
Profile Services	ProfilePlan.class	Διάθεση πληροφοριών προφίλ CRD
Context Services	CRDStatusPlan.class	Διάθεση πληροφοριών περιεχομένου CRD

Πίνακας 8: CRD Agent Cognitive Services

Κλείνοντας, οι πράκτορες που ανήκουν στην κατηγορία CRD Agents, μπορούν να αλληλεπιδράσουν μόνο με τον CRD Manager Agent. Το προφίλ των πρακτόρων που χρησιμοποιήθηκαν για την εκτέλεση της λειτουργίας της πλατφόρμας, παρουσιάζεται στον πίνακα 8.

AGENT NAME	TRX IDs	TRX FREQBAND	NUNBER OF RATs per TRX	RATs IDs	SERVICES	USER PROFILES
CRD_1	TRX_1	800-1900	**TRX_1(**#1**)**	**TRX_1(**RAT_1**)**,	VOICE, WAP,MMS	USER_P1 USER_P2
CRD_2	TRX_1 TRX_2	1800-1900 2100-24000	**TRX_1(**#1**)** **TRX_2(**#2**)**	**TRX_1(**RAT_1**)**, **TRX_2(**RAT_1,RAT_2**)**	VOICE,DATA, MMS,EMAIL, INTERNET	USER_P1 USER_P2 USER_P3

Πίνακας 9: Πρότυπα CRD Profiles που χρησιμοποιούνται στη SOVP

14.3.3 Generators Agents

Οι Generators Agents ανήκουν στην κατηγορία συνιστωσών, υποδομών λογισμικού. Έχουν αναπτυχθεί αποκλειστικά για την εξυπηρέτηση συγκεκριμένων αναγκών στην λειτουργία της πλατφόρμας. Συγκεκριμένα λειτουργούν ως προσομοιωτές λειτουργιών οι οποίοι βοηθούν στην εκτέλεση ολοκληρωμένων διαδικασιών λειτουργίας της SOVP. Πιο ειδικά, έχουμε δύο είδη generator agents: i) **Service Load Generator Agent** και ii) **CRD Status Generator Agent**. Οι πράκτορες υλοποιήθηκαν ως JADEX Agents και διαθέτουν το δικό τους μοναδικό ADF που συνδέεται με τις αντίστοιχες java κλάσεις που περιλαμβάνουν τον κώδικα για την υλοποίηση της λειτουργίας του Agent.

14.3.3.1 Service Load Generator Agent

Ο Service Load Generator Agent (SLG), λειτουργεί ως γεννήτρια φορτίου για έναν CAP Agent. Πιο συγκεκριμένα με την βοήθεια μιας java κλάσης, ενημερώνει μια λίστα με xml αρχεία. Σε κάθε αρχείο εισάγει στοιχεία που αντιστοιχούν σε πληροφορίες του CAP Context. Με την ολοκλήρωση της ενημέρωσης των xml αρχείων ο SLG Agent τα παρουσιάζει σε ένα GUI, που του έχουμε δημιουργήσει, ως διαθέσιμα αρχεία προς επιλογή από τον χρήστη. Όταν επιλεγεί κάποιο αρχείο περιγραφής, τεχνητού, φορτίου για ένα συγκεκριμένος CAP Agent και πατηθεί το κουμπί 'Update CAP Context', τότε αυτομάτως ξεκινά η διαδικασία ενημέρωσης του φορτίου του CAP Agent από τον SLG Agent. Η αλλαγή που προκαλεί ο SLG Agent πραγματοποιείται στα στοιχεία του CAP Context (όπου περιέχεται το Service Load). Οι μεταβολές που συμβαίνουν στον CAP μετά την αλληλεπίδραση του με τον SLG, αφορούν κυρίως στοιχεία που απεικονίζουν το φορτίο του κάθε TRX, τις υπηρεσίες ανά TRX και την κατανομή χρηστών ανά υπηρεσία. Με λίγα λόγια ο SLG Agent χρησιμοποιείται ώστε να προσομοιώσει διάφορες συνθήκες φορτίου στον CAP Agent (πχ: συμφόρηση TRX). Τέλος να επισημανθεί ότι, ο SLG Agent αλληλεπιδρά μόνο με πράκτορες που ανήκουν στην κατηγορία πρακτόρων *Cognitive Access Point Agents.*

14.3.3.2 CRD Status Generator Agent

Ο CRD Status Generator (CRD SD), αποτελεί έναν πράκτορα λογισμικού ο οποίος λειτουργεί ως προσομοιωτής Status για ένα CRD. Εξαρχής λοιπόν, διαπιστώνουμε ότι ο συγκεκριμένος Agent μπορεί να αλληλεπιδράσει μόνο με πράκτορες που ανήκουν στην κατηγορία πρακτόρων *Cognitive Reconfigurable Device Agent*. Μέσω κατάλληλα διαμορφωμένου ADF και αρχείων κώδικα για την εκτέλεση των περιγραφόμενων λειτουργιών, ο CRD SD Agent παρέχει την δυνατότητα, μέσω ειδικά διαμορφωμένου GUI, μεταβολής της κατάστασης ενός CRD Agent, σε πραγματικό χρόνο. Με τον τρόπο αυτό έχουμε την δυνατότητα να προσομοιώσουμε στο 100% την λειτουργία ενός CRD Agent σε πραγματικό χρόνο. Η διαδικασία φόρτωσης της νέας κατάστασης ακολουθεί παρόμοια λογική με αυτή της φόρτωσης νέου φορτίου σε έναν CAP Agent. Συγκεκριμένα στον CRD Agent ενημερώνει ένα συγκεκριμένο, προκαθορισμένο, αριθμό xml αρχείων με στοιχεία που αντιστοιχούν στην απεικόνιση του CRD Status. Όταν ολοκληρωθεί η ενημέρωση των αρχείων, είναι διαθέσιμα προς επιλογή (μέσω του GUI) από τον χρήστη ώστε να

σταλούν πληροφορίες αλλαγής της κατάστασης ενός επιλεγμένου CRD Agent, αμέσως μόλις πατηθεί το κουμπί 'Update CRD Status'.

Με λίγα λόγια ο CRD SG αποτελεί μια γεννήτρια παραγωγής τεχνιτών CRD Status. Σημαντικό είναι να επισημάνουμε ότι οι τιμές που αποδίδονται στα xml αρχεία που απεικονίζουν το CRD Status, δεν είναι τυχαίες αλλά πρέπει πάντοτε να διαμορφώνονται σύμφωνα με το προφίλ του κάθε CRD , ώστε το αποτέλεσμα από την αλλαγή του CRD Status να ανταποκρίνεται στην πραγματικότητα (πχ: *Δεν είναι ορθό να δημιουργήσουμε ένα CRD Status με 5 ενεργές υπηρεσίες, για κάποια χρονική στιγμή, όταν ο CRD, για τον οποίο δημιουργείται, περιέχει στο προφίλ του 2 διαθέσιμες υπηρεσίες που μπορεί να εξυπηρετήσει ως συσκευή*).

14.3.4 Manager Agents

Οι Manager Agents ανήκουν και αυτοί, στην κατηγορία συνιστωσών υποδομών λογισμικού. Δημιουργήθηκαν με σκοπό να καλύψουμε ένα από τα βασικότερα ζητούμενα της παρούσας μελέτης το οποίο αφορά την **δυναμική προσθήκη ή την αφαίρεση συνιστωσών από το σύστημα**. Έχουμε συνολικά δύο Manager Agents, έναν για κάθε κατηγορία συνιστωσών που μπορεί να παρουσιαστεί στο σύστημα. Συγκεκριμένα έχουμε: i) τον **CAP Manager Agent (CAP MA)** και ii) τον **CRD Manager Agent (CRD MA)**. Στη συνεχεία περιγράφουμε αναλυτικά υλοποίηση λειτουργίας του κάθε Manager Agent.

14.3.4.1 CAP Manager Agent

Ο CAP MA υλοποιείται με σκοπό να επιτηρεί την λειτουργία της πλατφόρμας που σχετίζεται με τα CAP. **Διαθέτει ένα σύνολο γνωστικών υπηρεσιών** μέσω των οποίων μπορεί να:

- Λάβει πληροφορίες σχετικά με το **προφίλ** του κάθε CAP Agent
- Λάβει πληροφορίες σχετικά με το **περιεχόμενο** του κάθε CAP Agent
- **Αναλύσει** πληροφορίες σχετικά με το περιεχόμενο του κάθε CAP Agent
- **Δημιουργήσει αποφάσεις** σχετικά με την λειτουργία ενός CAP Agent
- Να **αναδιαρθρώσει την λειτουργία** ενός CAP Agent

Κατηγοριοποιώντας τις παραπάνω υπηρεσίες, που διαθέτει ο CAP MA προκύπτει ο πίνακας 8 παρακάτω.

SOVP Cognitive Services Category	CAP Manager Agent Services Name (JAVA Class)	CAP Manager Agent Services Description
Profile Services	GetProfileInformations.class	Λήψη πληροφοριών προφίλ CAP
Context Services	SupervisorAgent.class	Λήψη πληροφοριών περιεχομένου CAP
Decision Making Services	ManageReconfigurationRequests.class	Ανάλυση πληροφοριών περιεχομένου του CAP
Reconfiguration Services	ManageReconfigurationRequests.class	Αποστολή νέων στοιχείων διαμόρφωσης της λειτουργία του CAP

Πίνακας 10: Πίνακας γνωστικών υπηρεσιών του CAP Manager Agent

Ο CAP MA όταν τεθεί σε λειτουργία πραγματοποιεί απευθείας έναν έλεγχο σε όλη την πλατφόρμα για την εύρεση ενεργών CAP Agents. Εφόσον εντοπίσει ενεργούς πράκτορες, τους τοποθετεί σε μία λίστα και ξεκινά αυτομάτως μια διαδικασία **λήψης πληροφοριών του προφίλ** τους. ενεργοποιεί λοιπόν, την υπηρεσία **GetProfileInformations,** η οποία ανήκει στις Profile Services της SOVP. Μέσω της υπηρεσίας αυτή ο CAP MA είναι σε θέση να εξασφαλίσει μέσα σε μικρό χρονικό διάστημα όλες τις πληροφορίες που αφορούν το προφίλ του κάθε διαθέσιμου CAP στην πλατφόρμα.

Επιπλέον έχει την δυνατότητα να ελέγχει με **δυναμικό τρόπο** την προθήκη νέων CAP Agent (νέων συνιστωσών) ή την μετακίνηση – αφαίρεση CAP Agent που υπήρχαν στην πλατφόρμα. Αυτό επιτυγχάνεται μέσα από την υλοποίησης της κλάσης αναζήτησης CAP Agent που διαθέτει ο CAP MA. Συγκεκριμένα έχουμε ρυθμίσει τον CAP MA ώστε να πραγματοποιεί αναζήτηση – έλεγχο των CAP Agent (που υπάρχουν στην SOVP κάθε στιγμή) ανά ένα λεπτό (1 min). Με αυτόν τον τρόπο το σύστημα είναι σε θέση να γνωρίζει κάθε λεπτό την κατάσταση των συνιστωσών τύπου CAP. Επιτυγχάνουμε με αυτόν τον τρόπο την **δυναμική διαχείριση της προσθήκης / αφαίρεσης** CAP συνιστωσών από το σύστημα, αλλά και την **δυναμική ενημέρωση του συστήματος**.

Εκτός από τις πολύ σημαντικές λειτουργίες που περιγράψαμε παραπάνω, ο CAP MA, διαθέτει κάποιες επιπλέον γνωστικές υπηρεσίες οι οποίες του δίνουν την δυνατότητα εκτέλεσης λειτουργιών που αφορούν την λήψη και ανάλυση πληροφοριών του CAP Context και την αποστολή πληροφοριών για το Reconfiguration του ενός CAP. Πιο συγκεκριμένα ο CAP MA λαμβάνει **πληροφορίες σχετικά με το CAP Context**, μέσω της υπηρεσίας **SupervisorAgent** η οποία ενεργοποιείται αμέσως μόλις λάβει κάποιο Reconfiguration Request από κάποιον CAP Agent. Το Reconfiguration Request περιλαμβάνει πληροφορίες που αφορούν την περιγραφή της τρέχουσας κατάστασης στον CAP Agent και αυτές τις πληροφορίες μπορεί να τις χρησιμοποιήσει ο CAP MA ώστε να εφοδιάσει με δεδομένα την υπηρεσία λήψης απόφασης που διαθέτει. Η υπηρεσία **ManageReconfigurationRequests** διαθέτει δύο βασικές μεθόδους όπου η μία λειτούργει για την **ανάλυση των λαμβανόμενων δεδομένων** από το Reconfiguration Request και η άλλη για την αποστολή δεδομένων για το **Reconfiguration** του CAP Agent. Ουσιαστικά ο CAP MA έχει υλοποιηθεί έτσι ώστε να μπορεί να πραγματοποιήσει σε πραγματικό χρόνο την αναδιαμόρφωση της λειτουργίας οποιουδήποτε CAP Agent της SOVP. Βέβαια να τονίσουμε ότι η Reconfiguration Service του CAP MA, ενεργοποιείται αν και μόνο αν δεχτεί αίτημα από κάποιον CAP Agent.

Συνοπτικά ο CAP MA, μπορεί να: i) ενημερωθεί για το προφίλ και το περιεχόμενο των ενεργών CAP Agents της πλατφόρμας, ii) αναδιαμορφώσει την λειτουργία ενός CAP Agent και iii) είναι διαρκώς ενημερωμένος για το πλήθος των προσθηκών ή των μετακινήσεων CAP πρακτόρων της πλατφόρμας.

14.2.4.2 CRD Manager Agent

Ο CRD MA υλοποιήθηκε ώστε η SOVP να μπορεί να διαχειριστεί τις λειτουργίες που αναπτύσσονται σε αυτή και αφορούν τα CRD. Διαθέτει δύο γνωστικές υπηρεσίες, οι οποίες το βοηθούν στην αλληλεπίδρασή του με τους CRD Agents. Οι υπηρεσίες αυτές του χρησιμεύουν ώστε να:

- Λάβει πληροφορίες σχετικά με το **προφίλ** του κάθε CRD Agent
- Λάβει πληροφορίες σχετικά με την **κατάσταση** του κάθε CRD Agent

Κατηγοριοποιώντας τις παραπάνω υπηρεσίες, που διαθέτει ο CAP MA προκύπτει ο πίνακας 9 παρακάτω.

SOVP Cognitive Services Category	CRD Manager Agent Services Name (JAVA Class)	CRD Manager Agent Services Description
Profile Services	GetProfileInformationPlan.class	Λήψη πληροφοριών προφίλ CRD
Context Services	GetStatusInformationPlan.class	Λήψη πληροφοριών κατάστασης CRD

Πίνακας 11: Πίνακας γνωστικών υπηρεσιών του CRD Manager Agent

Η λειτουργία του CRD MA είναι σχεδόν ίδια με ένα μέρος της λειτουργίας του CAP MA. Πιο συγκεκριμένα όταν ο CRD MA ενεργοποιηθεί, τότε εκτελεί απευθείας έλεγχο για την εύρεση ενεργών CRD Agents στην SOVP. Σε περίπτωση που εντοπίσει ενεργούς CRD Agents, του τοποθετεί σε μία λίστα και στην συνέχεια τους επιλέγει με την σειρά έναν προς έναν ώστε να επικοινωνήσει μαζί τους και να λάβει τις πληροφορίες που επιθυμεί. Ο πράκτορας διαχειριστής λαμβάνει πληροφορίες σχετικά με το προφίλ της συσκευής (η οποία πλέον απεικονίζεται ως πράκτορας), αλλά και πληροφορίες σχετικά με την τρέχουσα κατάσταση λειτουργίας της συσκευής.

Τις πληροφορίες τις λαμβάνει με την βοήθεια των υπηρεσιών **GetProfileInformationPlan** και **GetStatusInformationPlan** και τις έχει στη διάθεσή του, ώστε να τις χρησιμοποιήσει με οποιονδήποτε τρόπο χρειαστεί. Να επισημάνουμε ότι ο συγκεκριμένος πράκτορας έχει υλοποιηθεί κατά τέτοιο τρόπο ώστε να ενημερώνεται ανά ενάμιση λεπτό σχετικά με την κατάσταση της κάθε συσκευής – πράκτορα.

Τέλος, η υπηρεσία **SearchCRDPlan** δίνει την δυνατότητα στον CRD MA να ελέγχει με **δυναμικό τρόπο** την προθήκη νέων CRD Agent (νέων συνιστωσών) ή την μετακίνηση – αφαίρεση CRD Agent που υπήρχαν στην πλατφόρμα. Η διαδικασία επαναλαμβάνεται κάθε ένα λεπτό και τριάντα δευτερόλεπτα (1:30) και με αυτόν τον τρόπο το σύστημα είναι σε θέση να γνωρίζει κάθε λεπτό την κατάσταση των συνιστωσών τύπου CRD. Έτσι λοιπόν, επιτυγχάνουμε την **δυναμική διαχείριση της προσθήκης / αφαίρεσης** CRD συνιστωσών από το σύστημα, αλλά και την **δυναμική ενημέρωση του συστήματος**.

Συνοψίζοντας, ο CRD MA, μπορεί να: i) ενημερωθεί για το προφίλ και το περιεχόμενο των ενεργών CRD Agents της πλατφόρμας, και ii) είναι διαρκώς ενημερωμένος για το πλήθος των προσθηκών ή των μετακινήσεων CRD πρακτόρων της πλατφόρμας.

14.4 Μορφές αλληλεπίδρασης των στοιχείων της πλατφόρμας

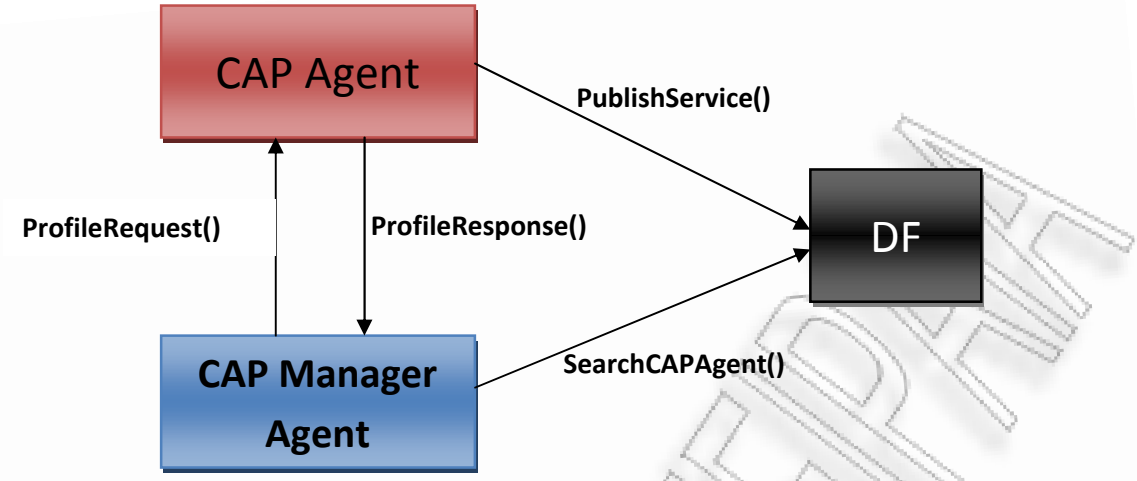
Στην προηγούμενη ενότητα αναφερθήκαμε εκτενώς στις συνιστώσες που μπορούμε να συναντήσουμε στο σύστημα της SOVP. Οι συνιστώσες που μπορεί να φιλοξενήσει η πλατφόρμα, αναπτύσσουν μεταξύ τους αλληλεπιδράσεις. Πιο συγκεκριμένα μπορούμε να μελετήσουμε τον πίνακα 10 παρακάτω ο οποίος παρουσιάζει τις οντότητες που αλληλεπιδρούν μεταξύ τους, στην SOVP. Επίσης στη συνέχεια ακολουθεί και η περιγραφή κάθε αλληλεπίδρασης και η σχηματική της αναπαράσταση. Να επισημάνουμε ότι επειδή η SOVP έχει

A/A	*Service Consumer*	*Service Provider*	*ΠΕΡΙΓΡΑΦΗ ΑΛΛΗΛΕΠΙΔΡΑΣΗΣ*
1	**CAP MA**	**CAP**	- CAP Profile Information Request - CAP Profile Information Response
2	**CAP**	**CAP MA**	- Send Reconfiguration Request. - Receive Reconfiguration Response
3	**CRD MA**	**CRD**	- CRD Profile Information Request - CRD Profile Information Response
4	**CRD MA**	**CRD**	- CRD Status Information Request - CRD Status Information Response

Πίνακας 12: Αλληλεπιδράσεις μεταξύ των συνιστωσών της SOVP

14.4.1 Αλληλεπίδραση μεταξύ CAP MA και CAP

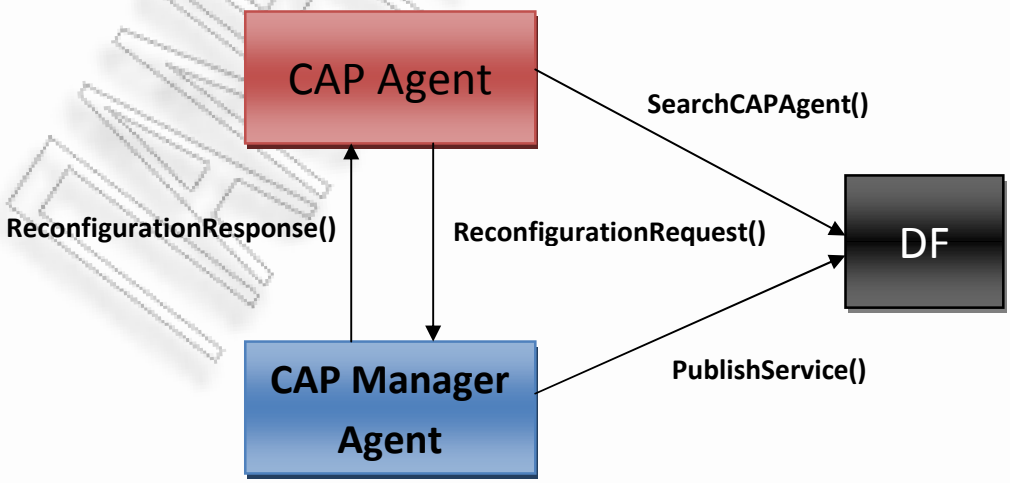
Η αλληλεπίδραση που εκτελείται μεταξύ του CAP MA και ενός οποιουδήποτε CAP Agent, στοχεύει στην ολοκλήρωση την διαδικασίας λήψης πληροφοριών για το **CAP Profile**. Συγκεκριμένα συνεργάζονται δύο γνωστικές υπηρεσίες οι οποίες ανήκουν στο ίδιο block υπηρεσιών της πλατφόρμα, στο Profile Services. Η υπηρεσία **GetProfileInformationPlan** ενημερώνει με κατάλληλα requests την υπηρεσία **ProfilePlan** του CAP, η οποία της αποστέλλει τις ζητούμενες πληροφορίες που περιγράφουν το προφίλ του Cognitive access point. Κάθε CAP Agent που συνδέεται την πλατφόρμα έχουμε αναφέρει ότι δηλώνει την παρουσία του στον JADE Directory Facilitator Agent, μέσω του οποίου μπορεί να εντοπιστεί από το CAP MA. Ο CAP Agent είναι συνδεδεμένος με ένα σύνολο προκαθορισμένων υπηρεσιών και αυτό είναι γνωστό στον CAP Manager ο οποίος αφού εντοπίσει την παρουσία του καλεί την υπηρεσία και συνδέεται απευθείας με τον CAP που την προωθεί. Το σχήμα 13 παρουσιάζει την διαδικασία αλληλεπίδρασης μεταξύ CAP MA – CAP Agent.



Σχήμα 13: «Αλληλεπίδραση μεταξύ CAP MA και CAP »

14.4.2 Αλληλεπίδραση μεταξύ CAP και CAP MA

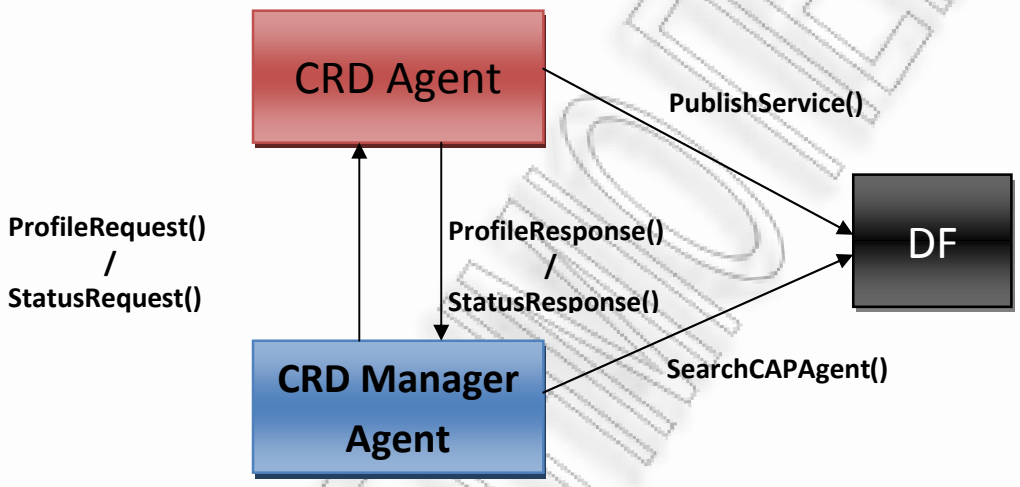
Η συγκεκριμένη αλληλεπίδραση είναι παρόμοια με την παραπάνω, με την διαφορά ότι υπάρχει διαφοροποίηση στο περιεχόμενο και στην φορά των ανταλλασσόμενων μηνυμάτων. Σε αυτή την περίπτωση ο CAP Agent παίζει το ρόλο του Service Consumer ενώ ο CAP MA λειτουργεί ως Service Provider. Η παρούσα αλληλεπίδραση πραγματοποιείται μόνο στην περίπτωση που ο CAP μετά από monitoring διαπιστώσει συμφόρηση σε κάποιον από τους πομποδέκτες του. Ένα συμβεί κάτι τέτοιο η υπηρεσία CAPMonitoringPlan, ενεργοποιεί την τοπική υπηρεσία CAPContextPlan, μέσω της οποίας διαθέτει πληροφορίες σχετικά με το τρέχον CAP Context. Η διαδικασία συνεχίζεται με την κλήση των υπηρεσιών **SupervisorAgent** και **ManageReconfigurationRequests** του CAP MA οι οποίες λαμβάνουν τα στοιχεία του CAP Context, τα αναλύουν, δημιουργούν ένα νέο πλάνο λειτουργίας το οποίο θα λύσει την συμφόρηση στον CAP Agent και καταλήγουν με την αποστολή ενός Reconfiguration Response μηνύματος. Παρατηρούμε λοιπόν, ότι μέσα από την συνεργασία διαφορετικών γνωστικών υπηρεσιών επιτυγχάνεται η ολοκλήρωση της διεργασίας της συμφόρησης σε ένα CAP Agent. Το σχήμα 14 παρουσιάζει την διαδικασία αλληλεπίδρασης μεταξύ CAP Agent – CAP MA.



Σχήμα 14: «Αλληλεπίδραση μεταξύ CAP και CAP MA»

14.4.3 Αλληλεπίδραση μεταξύ CRD MA και CRD

Η αλληλεπίδραση του CRD MA με τον CRD μπορεί να συμβεί για δύο λόγους. Είτε για λήψη πληροφοριών που αφορούν το προφίλ του CRD, είτε για λήψη πληροφοριών που αφορούν την κατάσταση του CRD. Και στις δύο περιπτώσεις αλληλεπίδρασης ο CRD MA λειτουργεί ως Service Consumer ενώ ο CRD λειτουργεί ως Service Provider. Ο CRD MA, έχοντας εντοπίσει το την καταγραφή του CRD στον DF, αναγνωρίζει αυτόματα ότι παρέχει τις υπηρεσίες για την λήψη πληροφοριών του προφίλ και της κατάστασης του CRD. Οι υπηρεσίες **GetProfileInformationPlan** και **GetStatusInformationPlan** από την πλευρά του CRD MA, συνεργάζονται με τις υπηρεσίες **ProfilePlan** και **GetStatusPlan**, αντιστοίχως, από την πλευρά του CRD και λαμβάνουν τις κατάλληλες πληροφορίες που αφορούν το προφίλ και την τρέχουσα κατάστασης του CRD. Το σχήμα 15 απεικονίζει τις παραπάνω αλληλεπιδράσεις.



Σχήμα 15: «Αλληλεπίδραση μεταξύ CRD και CRD MA»

14.5 SOVP Plug and Play διαδικασία

Ένα πολύ σημαντικό στοιχείο που πρέπει να υλοποιηθεί στην πλατφόρμα ώστε να μπορεί να θεωρηθεί αποδοτική και δυναμικά αναδιαρθώσιμη είναι η προσθήκη νέων συνιστωσών με Plug and Play τρόπο. Η SOVP έχει υλοποιηθεί έτσι ώστε να επιτρέπει την εισαγωγή νέων συνιστωσών με Plug and Play τρόπο. Η διαδικασία του Virtualization είναι το βασικό συστατικό για την δημιουργία μιας καλά ορισμένης εφαρμογής, η οποία αποτελεί τμήμα της SOVP, και δίνει την δυνατότητα στον χρήστη της, να εικονοποιήσει την λειτουργία οποιασδήποτε δικτυακής συσκευής που ανήκει στην κατηγορία των Cognitive Access Points, ενώ επίσης επιτρέπει και την εικονοποίηση των λειτουργιών των συσκευών χρήστη που ανήκουν στην κατηγορία Cognitive Reconfigurable Device. Τα χαρακτηριστικά των συνιστωσών που αναφέραμε, έχουν περιγραφεί αναλυτικά στην ενότητα 14.3.

Η εφαρμογή που δίνει την ιδιότητα Plug and Play στην SOVP, αναφέρεται στην υλοποίηση ως **Virtualization Templates Manager (VTM)**. Παρέχει ειδικά διαμορφωμένο γραφικό περιβάλλον χρήστη (GUI), μέσου του οποίο ο χρήστης μπορεί εύκολα και γρήγορα να κάνει Virtualized μια συνιστώσα. Στην τρέχουσα υλοποίηση της SOVP, η εφαρμογή VTM διαθέτει δύο διαφορετικούς τύπου προτύπων εικονικοποίησης. Το πρότυπο για στοιχεία του τύπου Cognitive Access

Point, και για στοιχεία του τύπου Cognitive Reconfigurable Device. Κάθε πρότυπο υλοποιείται ξεχωριστά και διαθέτει το δικό του ξεχωριστό γραφικό περιβάλλον, μέσα από το οποίο ο χρήστης μπορεί να εισάγει τις απαραίτητες, ζητούμενες, πληροφορίες που θα αποτελέσουν τα δομικά συστατικά για την δημιουργία του εικονικού αντιγράφου της συνιστώσας.

Για την ανάπτυξη των προτύπων εικονικοποίησης, βασιστήκαμε στις οντότητες για την περιγραφή ροής πληροφοριών που παρουσιάζονται στις εικόνες 44 και 45 και προέρχονται από προγενέστερη ερευνητική διαδικασία.[130] Ακλουθώντας με απόλυτη ακρίβεια τις παρεχόμενες πληροφορίες από τα διαγράμματα, πραγματοποιήσαμε την υλοποίηση ενός πολύ καλά οργανωμένου λογισμικού το οποίο διαθέτει όλα τα απαιτούμενα χαρακτηριστικά για το Virtualization των πόρων του συστήματος της SOVP. Ο χρήστης ακλουθώντας μια σειρά, καλά ορισμένων, βημάτων ανάπτυξης του virtualized πόρου μέσω του γραφικού περιβάλλοντος, μπορεί μέσα σε λίγα λεπτά να παράγει ένα εικονικό αντίγραφο της συνιστώσας που διάλεξε. Το Virtualized της συνιστώσας υλοποιείται με σκοπό να μπορεί να χρησιμοποιηθεί στην SOVP πλατφόρμα. Για τον λόγο αυτό τα παραδοτέα που προκύπτουν από την φάση εκτέλεσης της διαδικασίας του Virtualization, είναι τα εξής:

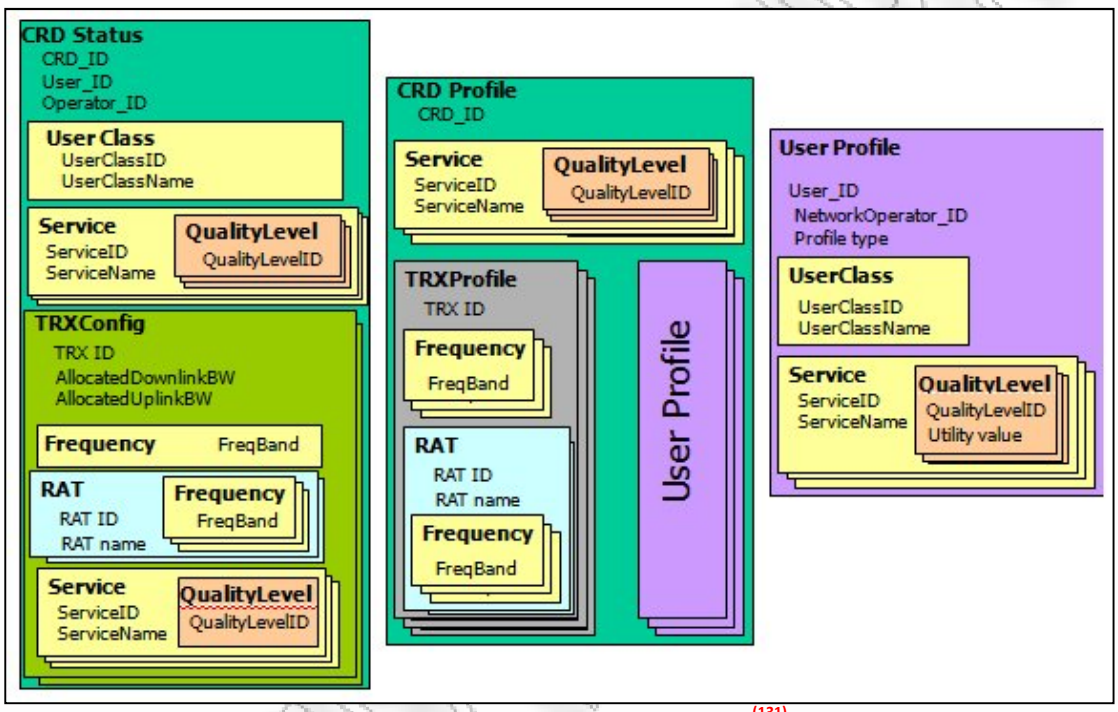
- Ένα Agent Definition File, σε μορφή xml, το οποίο περιλαμβάνει όλες τις απαραίτητες πληροφορίες που περιγράφουν την λειτουργία της Virtualized συνιστώσας.
- Ένα Capability File, σε μορφή xml, το οποίο περιέχει όλες τις πληροφορίες που αφορούν το προφίλ του και τις υπηρεσίες που συνδέονται με την Virtualized συνιστώσα.
- Ένα φάκελο (java package) ο οποίος περιέχει ένα σύνολο από υλοποιημένες μονάδες λογισμικού (java classes), οι οποίες εκτελούν συγκεκριμένες λειτουργίες που αντιστοιχούν στην συνιστώσα. Επιπλέον να αναφέρουμε ότι περιλαμβάνονται και γνωστικές υπηρεσίες για κάθε συνιστώσα οι οποίες συνδέονται με την εικονοποιημένη συνιστώσα ώστε να υλοποιήσουν ένα σύνολο συγκεκριμένων λειτουργιών.

Για κάθε είδος συνιστώσας που έχει που μπορεί να περάσει από την διαδικασία του Virtualization, έχουμε ήδη δημιουργήσει τις απαραίτητες κλάσεις λειτουργίας που βασίζονται στην περιγραφή του Ontology Information Flow της κάθε συνιστώσας. Έτσι κάθε φορά που πραγματοποιείται η διαδικασία του Virtualization, η εφαρμογή είναι σε θέση να αντιληφθεί το είδος της συνισταμένης που εικονοποιούμε και μπορεί να παράγει ένα αντίγραφο του φακέλου (package) που περιέχει όλες τις απαραίτητες java κλάσεις για την λειτουργία του στοιχείου στην SOVP. Επίσης τα δύο αρχεία xml που παράγονται από την διαδικασία, αποθηκεύονται σε κατηγοριοποιημένους φακέλους ώστε να μπορούν εύκολα να εντοπιστούν από τον χρήστη κατά την αναζήτηση τους, μετά την Virtualization.

Όταν ολοκληρωθεί η διαδικασία του Virtualization, ο χρήστης μπορεί να **συνδέσει απευθείας στην πλατφόρμα το νέο στοιχείο** που δημιούργησε, κάνοντας

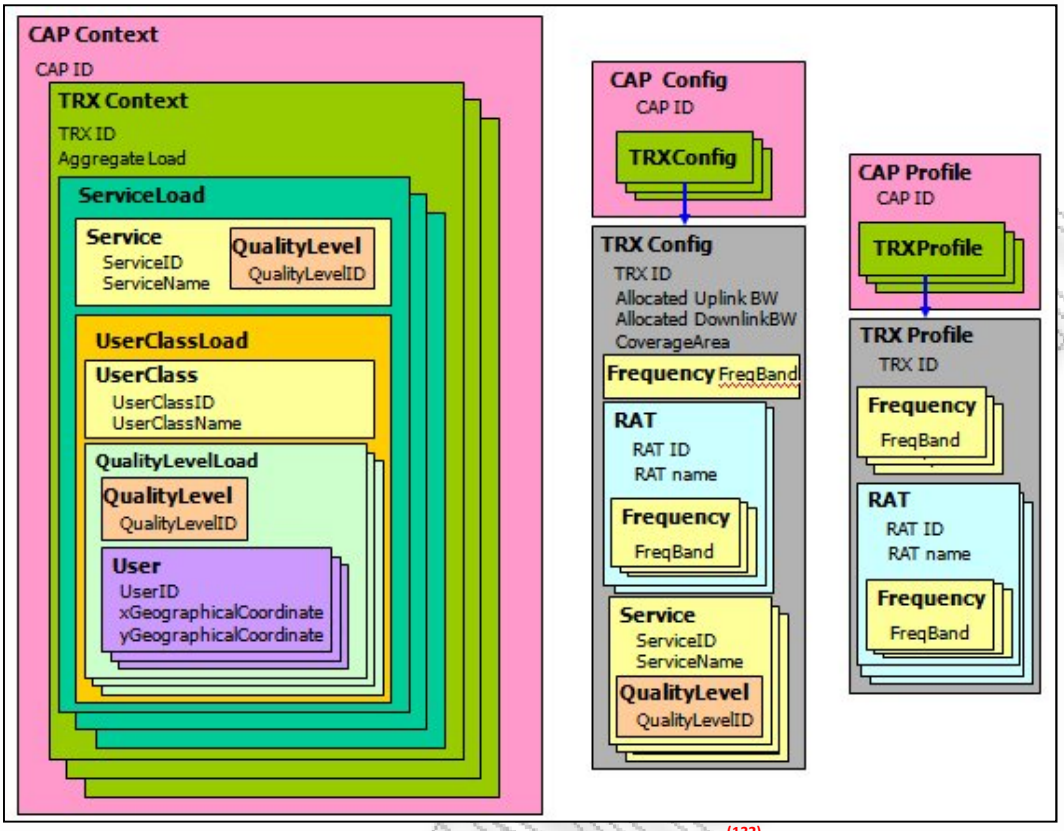
[130] V. Stavroulaki, N. Koutsouris, K. Tsagkaris, P. Demestichas, "**A platform for the integration and management of cognitive systems in future networks**", In Proc. IEEE Global Communications Conference (GLOBECOM 2010), Miami, USA, December 2010, International Conference Papers.

απλά εκκίνηση του ADF αρχείου με την βοήθεια του Standalone Jadex Agent. Αυτό υλοποιείται με την χρήση ενός έτοιμου GUI το οποίο επιτρέπει στον χρήστη να κάνει αναζήτηση (μέσω ενός java file chooser) του αρχείου που δημιούργησε και αφού το εντοπίσει, να το επιλέξει και να πατήσει το κουμπί 'Start A SOVP Agent' το οποίο βρίσκεται πάνω στο GUI. Φυσικά η διαδικασία εντοπισμού δεν είναι καθόλου πολύπλοκή, μιας και το GUI έχει προγραμματιστεί ώστε για την αναζήτηση αρχείων να μας οδηγεί στον φάκελο όπου αποθηκεύονται από την διαδικασία του Virtualization. Έτσι με τον τρόπο αυτό υλοποιείται η διαδικασία προσθήκης νέων συνιστωσών στο σύστημα, με Plug and Play τρόπο.



Εικόνα 44: CRD Ontology Information Flow[131]

[131] Βλ. παρ. 130, σελ. 124.



Εικόνα 45: CAP Ontology Information Flow[132]

Προτού ολοκληρωθεί η παρούσα ενότητα, είναι πολύ σημαντικό να τονίσουμε ότι η εφαρμογή για την υλοποίηση του Virtualization των πόρων της SOVP, σχεδιάστηκε και υλοποιήθηκε ώστε να είναι επεκτάσιμη και να μπορεί να δεχτεί μελλοντικά και νέα πρότυπα εικονικοποίησης συνιστωσών. Το μόνο που πρέπει να κάνει κανείς είναι να προσθέσει λίγες γραμμές κώδικα στην υπάρχουσα εφαρμογή, ώστε να είναι ορατό το νέο template και οι αντίστοιχες κλάσεις του, από τον Virtualization Templates Manager. Στο επόμενο κεφάλαιο θα παρουσιάσουμε στην πράξη όλα όσα αναφέραμε στο παρόν κεφάλαιο.

✓ *Ανακεφαλαιώνοντας*

Στο κεφάλαιο αυτό αναφερθήκαμε αναλυτικά στην υλοποίηση της εφαρμογής της SOVP. Επιχειρήσαμε να τονίσουμε την δυναμικότητα της και την οργάνωσή της ως ένα multi-agent Service Oriented σύστημα, επικεντρώνοντας στα χαρακτηριστικά σύνθεσης υπηρεσιών και δυναμικής διαχείρισης των πόρων, που διαθέτει. Επιπλέον περιγράψαμε όλες τις συνιστώσες που είναι δυνατόν να λειτουργήσουν στην SOVP και παρουσιάσαμε τις μεταξύ τους αλληλεπιδράσεις. Τέλος αναφερθήκαμε στην δυναμική επεκτασιμότητα της πλατφόρμας με την προσθήκη συνιστωσών με Plug and Play τρόπο. Προσπαθήσαμε να εξηγήσουμε την διαδικασία με την οποία υλοποιείται το χαρακτηριστικό Plug and Play στην εφαρμογή μας, μέσα από αναφορές στην Virtualization διαδικασία που υλοποιείται στην τρέχουσα υλοποίηση της εφαρμογής.

[132] Βλ. παρ. 130, σελ. 124.

Μέρος Δ΄

«Εκτέλεση, Σενάρια Λειτουργίας, Αποτελέσματα του συστήματος SOVP»

✓ Κεφάλαιο 15: «Εφαρμογή και εκτέλεση λειτουργίας της SOVP»

15.1 Εκκίνηση της εφαρμογής

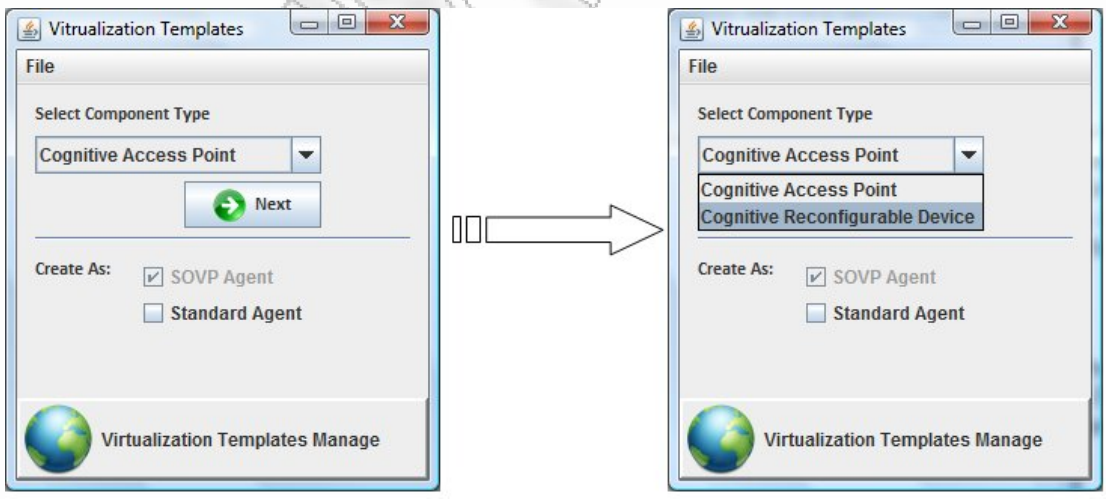
Η εκτέλεση της εφαρμογής δεν προαπαιτεί κάποια εγκατάσταση συστήματος. Η μόνη απαραίτητη διαδικασία, για την εκτέλεση της εφαρμογής, είναι η εκτέλεση των δύο αρχείων τύπου .bat που δημιουργήσαμε για την εκκίνηση του γραφικού περιβάλλοντος διαχείρισης (Administrator Monitor) και την εκκίνηση του Virtualization Templates Manager. Επίσης η μοναδική απαίτηση συστήματος είναι η ύπαρξη εγκατάστασης του πακέτου JDK 1.6 ή νεότερης έκδοσής του.

15.2 Εκτέλεση και Σενάρια λειτουργίας της SOVP

Η υλοποίηση της εφαρμογής περιλαμβάνει δυο βασικά δομικά μέρη. Την **δημιουργία ενός SOVP Agent**, μέσω της διαδικασίας του Virtualization και δεύτερον την **σύνδεση και εκτέλεση του Agent** που δημιουργήσαμε από την Virtualization διαδικασία. Για τον λόγο αυτόν, θα εκτελέσουμε την εφαρμογή μας σε δύο φάσεις. Πρώτα στην φάση Virtualization των συνιστωσών του συστήματος και δεύτερον την σύνδεση και εκτέλεση των συνιστωσών στην πλατφόρμα.

15.2.1 Φάση Λειτουργίας 1: SOVP Virtualization

Η διαδικασία του Virtualization πραγματοποιείται, όπως έχουμε προαναφέρει, με την χρήση του Virtualization Templates Manager (VTM). Έτσι, εκκινούμε την εφαρμογή του VTM. Με την εκκίνηση της εφαρμογής μας παρουσιάζεται η οθόνη της εικόνας 46. Στα στιγμιότυπα a και b, παρουσιάζονται οι δυνατές επιλογές στοιχείου προς εικονοποίηση.



(a) (b)

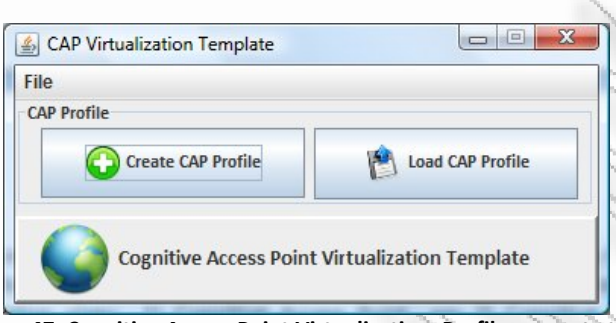
Εικόνα 46: Virtualization Templates Manager Startup Monitor

Παρατηρούμε ότι ο χρήστης μπορεί να εφαρμόσει την διαδικασία του Virtualization σε δύο τύπους στοιχείων: i) **Cognitive Access Point** και ii) **Cognitive Reconfigurable Device**. Στη συνέχεια θα εκτελέσουμε την διαδικασία και για τους δύο τύπους, παράγοντας δύο διαφορετικούς SOVP Agents οι οποίοι θα μπορούν να προστεθούν στην SOVP με Plug and Play τρόπο. Η σειρά εφαρμογής της

Virtualization είναι: 1) Cognitive Access Point και 2) Cognitive Reconfigurable Device.

15.2.1.1 SOVP Virtualization - Cognitive Access Point

Έχοντας επιλέξει το Cognitive Access Point , η επόμενη οθόνη που θα μας εμφανιστεί, αφού πατήσουμε το κουμπί Next, παρουσιάζεται στην εικόνα 47.

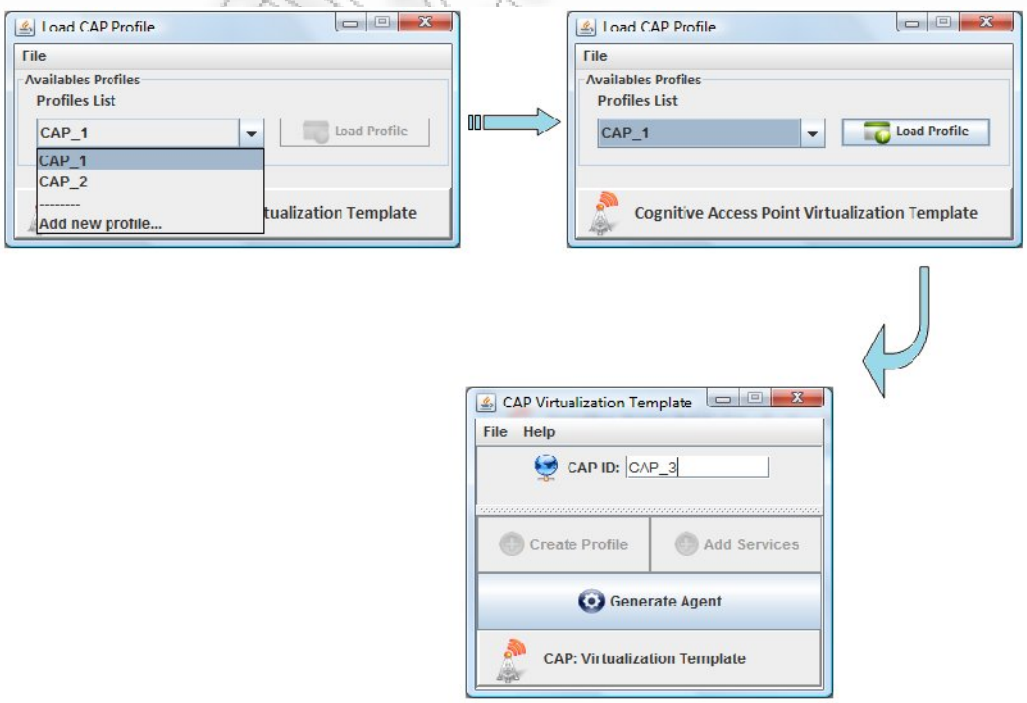


Εικόνα 47: Cognitive Access Point Virtualization, Profile generate type

Στην οθόνη αυτή ο χρήστης μπορεί να επιλέξει με ποιον τρόπο θα δημιουργήσει το νέο Agent. Συγκεκριμένα έχει δύο επιλογές: i) την δημιουργία μέσω της συμπλήρωσης στοιχείων στο διαθέσιμο Virtualization Template ή ii) την φόρτωση ενός έτοιμου προφίλ στο οποίο το μόνο που απαιτείται είναι η αλλαγή ονόματος του Agent.

Α) *Φόρτωση προφίλ*

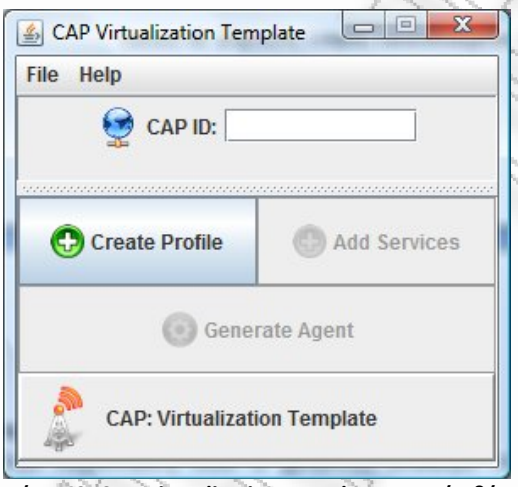
Η φόρτωση προφίλ αποτελεί ουσιαστικά ένα εργαλείο που μας βοηθά να αποφύγουμε τις επαναλαμβανόμενες διαδικασίες Virtualization. Ο χρήστης μπορεί να επιλέξει από μια λίστα, ήδη εικονικοποιημένων συνιστωσών, την συνιστώσα που είναι ίδιου τύπου με αυτή που θέλει να εισάγει στο σύστημα, και με το πάτημα ενός κουμπιού, η εφαρμογή αναλαμβάνει την δημιουργία μιας νέας ολοκληρωμένης οντότητας SOVP Agent. Η νέα οντότητα αποτελεί πιστό αντίγραφο την αρχικής. Η διαδικασία απεικονίζεται στην εικόνα 48.



Εικόνα 48: Φόρτωση έτοιμης εικονοποιημένης συνιστώσας

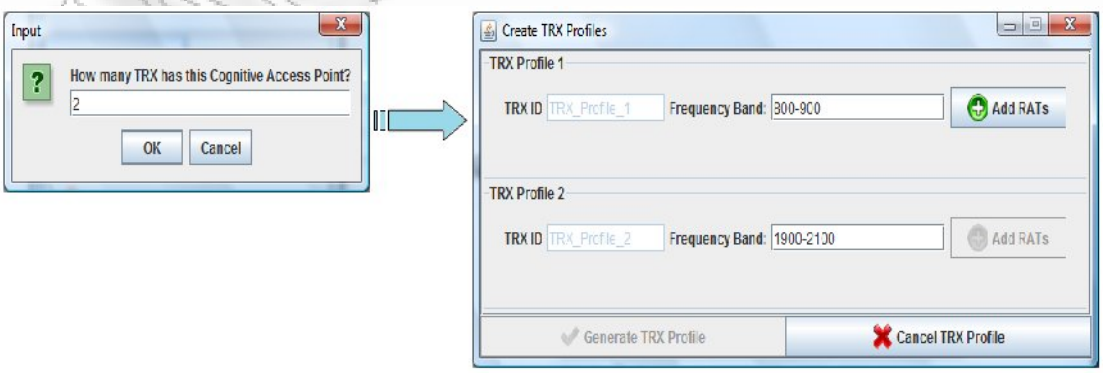
Β) *Δημιουργία μέσω Virtualization Template*

Στην περίπτωση που ο χρήστης επιλέξει την δημιουργία ενός Cognitive Access Point, τότε του εμφανίζεται η οθόνη τις εικόνας 49, όπου παρατηρούμε ότι η μοναδική ενεργή επιλογή, στο panel με τα κουμπιά, είναι η δημιουργία νέου προφίλ. Ο χρήστης αφού πρώτα συμπληρώσει το πεδίο που αντιστοιχεί στο CAP ID, στη συνέχει πρέπει να επιλέξει το κουμπί δημιουργίας νέου προφίλ, ώστε να προχωρήσει στην επόμενη οθόνη του template, όπου του ζητείται η συμπλήρωση στοιχείων που αφορούν το προφίλ του CAP. Στο σημείο αυτό να υπενθυμίσουμε ότι η ανάπτυξη του CAP Virtualization Template, βασίστηκε στην οντότητα ροής πληροφοριών (CAP Ontology Information Flow) που περιγράψαμε στο κεφάλαιο 14. Για τον λόγο αυτό δεν θα μπούμε στην διαδικασία αναλυτικής επεξήγησης των πεδίων του template.

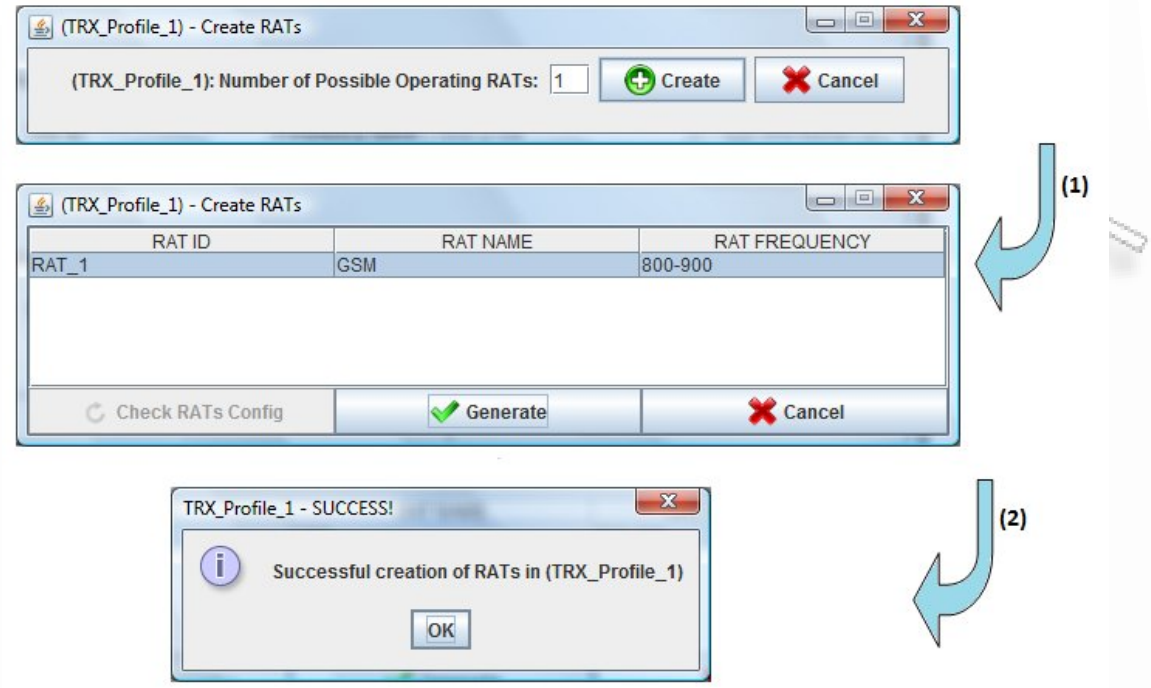


Εικόνα 49: CAP Virtualization Template Αρχική οθόνη

Εφόσον πατηθεί το κουμπί δημιουργίας του προφίλ, αρχικά εμφανίζεται στον χρήστη ένα πλαίσιο διαλόγου που του ζητά να αναφέρει τον αριθμό των TRX Profiles που υπάρχουν στον CAP και αφού το συμπληρώσει, του εμφανίζεται η φόρμα συμπλήρωσης στοιχείων του κάθε TRX Profile (εικόνα 50). Αφού συμπληρωθούν τα στοιχεία, ο χρήστης πρέπει να προσθέσει τα RATs που περιλαμβάνει κάθε TRX Profile. Αυτό μπορεί να το κάνει πατώντας το κουμπί 'Add RATs ', και αφού συμπληρώσει τον αριθμό και τα στοιχεία του κάθε RAT, να δημιουργήσει έτσι ένα πλήρες TRX Profile. Η εικόνα 51 απεικονίζει τα βήματα για την προσθήκη RATs σε TRX Profiles.

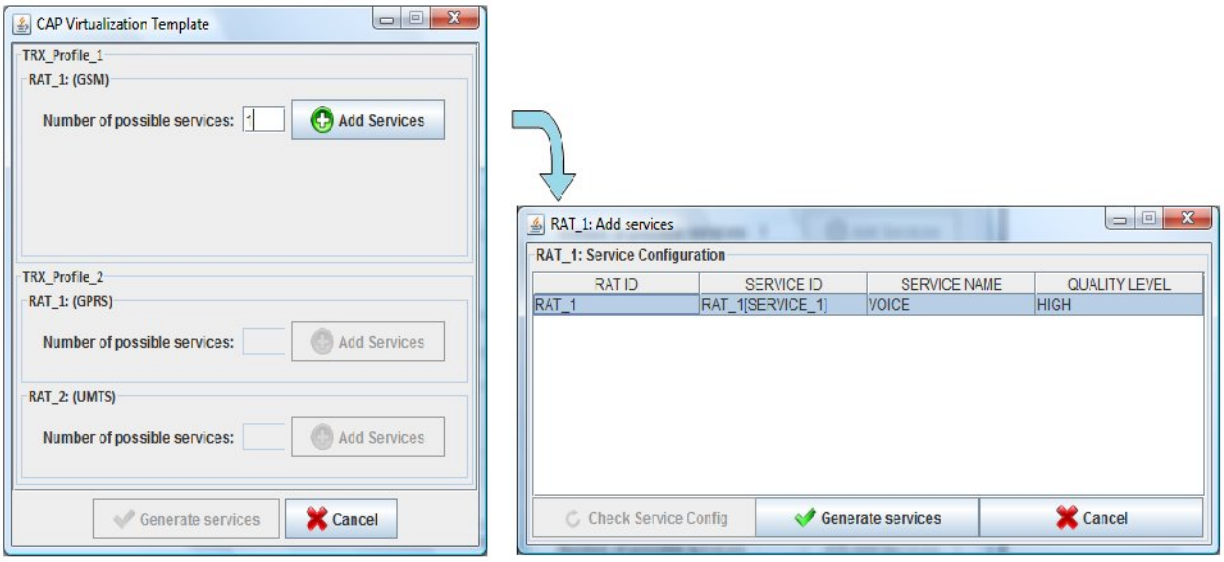


Εικόνα 50: Προσθήκη TRX Profiles στον CAP



Εικόνα 51: Προσθήκη RATs σε TRX Profile

Επόμενη διαδικασία μετά την δημιουργία του προφίλ του CAP, είναι η προσθήκη των υποστηριζόμενων υπηρεσιών από κάθε RAT σε κάθε TRX. Η διαδικασία ξεκινά με το πάτημα του κουμπιού 'Add Services' στην αρχική οθόνη του CAP VT (βλ. εικόνα 49 παραπάνω). Αρχικά μας εμφανίζεται η οθόνη συμπλήρωσης του αριθμού υπηρεσιών σε κάθε διαθέσιμο RAT ανά TRX. Αφού συμπληρώσουμε τον αριθμό, προχωράμε με την προσθήκη των υπηρεσιών, σε κάθε RAT ξεχωριστά, πατώντας το κουμπί 'Add Services'. Η διαδικασία ολοκληρώνεται με την προσθήκη υπηρεσιών σε όλα τα διαθέσιμα RAT.



Εικόνα 52: Προσθήκη υπηρεσιών στα διαθέσιμα RATs ανά TRX

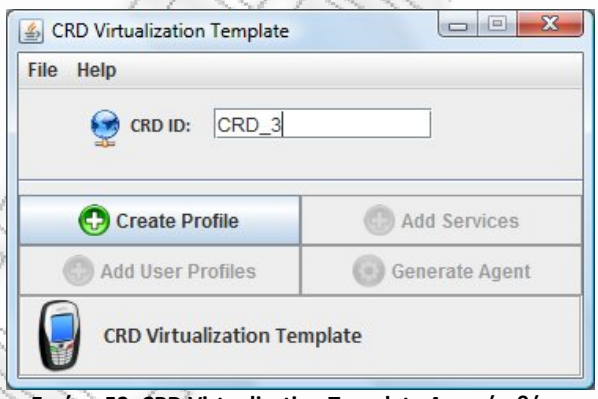
Όταν ολοκληρωθεί η διαδικασία προσθήκης υπηρεσιών τότε η συνιστώσα Cognitive Access Point, είναι έτοιμη ώστε να μετατραπεί σε CAP Agent μέσω του Virtualization. Συγκεκριμένα στην αρχική οθόνη, ενεργοποιείται το κουμπί 'Generate Agent' όπου όταν πατηθεί, η εφαρμογή παράγει αυτόματα τα εξής αρχεία: i) ένα Agent Definition File, ii) ένα Capability.xml αρχείο και τέλος iii) ένα νέο directory που περιέχει τις απαραίτητες κλάσεις που υλοποιούν τις υπηρεσίες που απαιτούνται για την λειτουργία του νέου Agent στην πλατφόρμα SOVP.

Στο σημείο αυτό ολοκληρώνεται η περιγραφή της διαδικασία Virtualization για μια συνιστώσα τύπου Cognitive Access Point. Ακολουθεί η αντίστοιχη διαδικασία για τις συνιστώσες τύπου Cognitive Reconfigurable Device.

15.2.1.2 SOVP Virtualization - Cognitive Reconfigurable Device

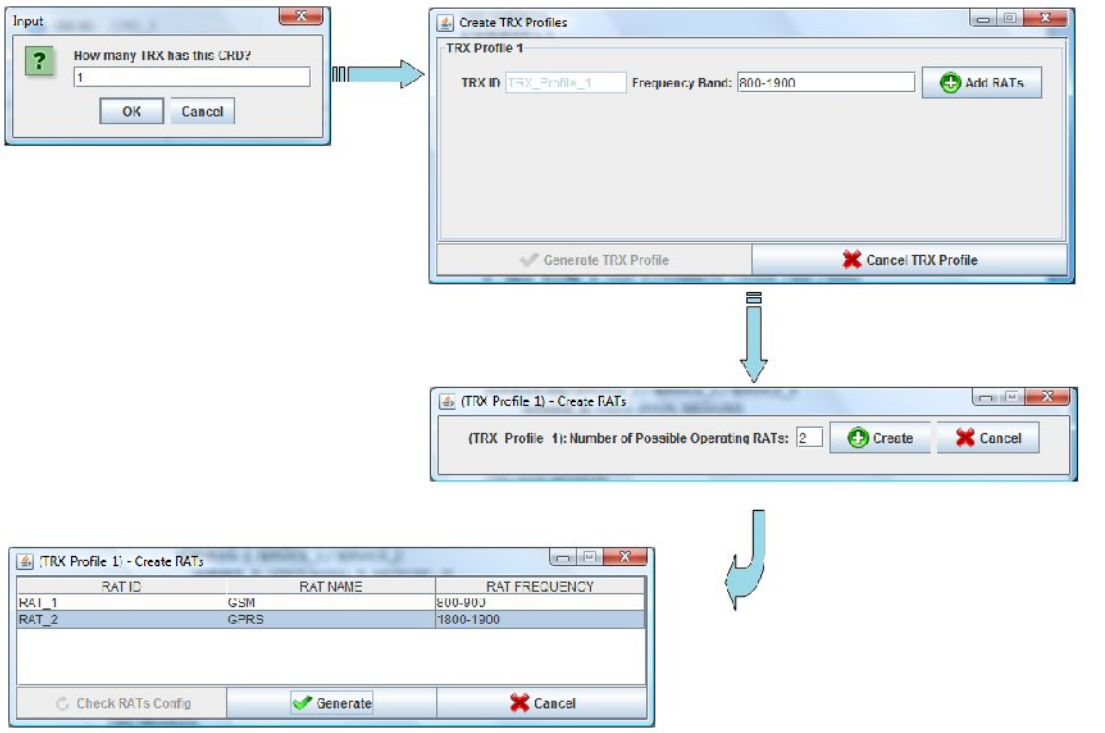
Η εφαρμογή της Virtualization διαδικασίας σε μια συνιστώσα του τύπου Cognitive Reconfigurable Device, δεν διαφέρει λειτουργικά από την αντίστοιχη διαδικασία για τις συνιστώσες τύπου Cognitive Access Point. Η διαφορά εντοπίζεται στην δομή του template διότι η CRD Ontology Information Flow, περιγράφεται με διαφορετικά στοιχεία από αυτά της CAP Ontology Information Flow. Η εξέλιξη της διαδικασίας περιγράφεται αναλυτικά στη συνέχεια.

Η αρχική οθόνη του CRD Virtualization Template, απεικονίζεται στην εικόνα 53 και όπως μπορεί να παρατηρήσει κάποιος, περιέχει τέσσερεις επιλογές για την διαχείριση του template. Μια συνιστώσα CRD περιέχει τα εξής: i) TRX Profile, ii) Services και iii) User Profiles.



Εικόνα 53: CRD Virtualization Template Αρχική οθόνη

Ξεκινώντας, ο χρήστης θα πρέπει να δημιουργήσει το προφίλ του CRD στο οποίο περιέχονται τα διαθέσιμα TRX Profiles της συσκευής. Πατώντας το κουμπί 'Create Profile' εμφανίζονται η οθόνη συμπλήρωσης του αριθμού των TRX Profiles και στην συνέχεια η οθόνη συμπλήρωσης των στοιχείων του κάθε TRX Profile. Στη συνέχεια ο χρήστης προσθέτει τα αντίστοιχα RATs σε κάθε διαθέσιμο TRX Profile και αφού ολοκληρώσει την διαδικασία, ολοκληρώνεται η δημιουργία του CRD Profile. Η εικόνα 54 δείχνει την εμφάνιση των οθονών που περιγράψαμε μόλις τώρα.

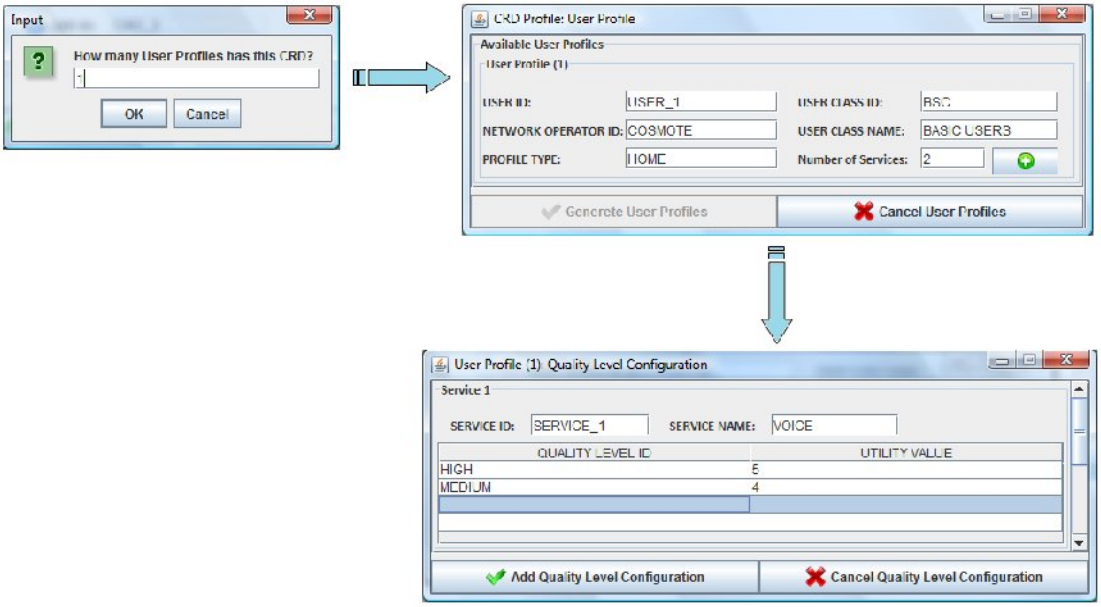


Εικόνα 54: CRD Create TRX Profiles & RATs

Συνεχίζοντας την διαδικασία εικονοποίησης της CRD συνιστώσας, ο χρήστης θα πρέπει να δημιουργήσει τις υπηρεσίες που μπορεί να εξυπηρετήσει η συσκευή. Ξεκινάει πατώντας το κουμπί 'Add Services', και αφού συμπληρώσει τον αριθμό των υπηρεσιών, συνεχίζει με την συμπλήρωση των στοιχείων που αντιστοιχούν σε κάθε υπηρεσία. Με την αρχικοποίηση των στοιχείων όλων των υπηρεσιών, ολοκληρώνεται η προσθήκη υπηρεσιών στο CRD. Η εικόνα 55 παρουσιάζει την εξέλιξη της προαναφερθείσας διαδικασίας.

Εικόνα 55: Προσθήκη υπηρεσιών στο CRD

Το τελευταίο στάδιο της διαδικασίας, περιλαμβάνει την προσθήκη των προφίλ χρηστών της που μπορούν να συνδεθούν με την συσκευή. Στις οθόνες αυτές, που παρουσιάζονται στην εικόνα 56, ο χρήστης εισάγει δεδομένα που προσδιορίζουν το προφίλ του χρήστη. Με την ολοκλήρωση της αρχικοποίησης των User Profiles της συσκευής, ολοκληρώνεται η διαδικασία του Virtualization και πλέον ο χρήστης αρκεί να πατήσει το 'Generate Agent' κουμπί, ώστε να παράγει την εικονικοποιημένη συνιστώσα. Η εφαρμογή με αυτόματο τρόπο παράγει τρείς τύπου αρχείων που αντιπροσωπεύουν την λειτουργία της συνιστώσας στην πλατφόρμα.



Εικόνα 56: CRD Initialize User Profiles

Τα αρχεία είναι ίδια με αυτά που παράγονται στη Virtualization του CAP Agent, δηλαδή είναι: i) ένα Agent Definition File, ii) ένα Capability.xml αρχείο και τέλος iii) ένα νέο directory που περιέχει τις απαραίτητες κλάσεις που υλοποιούν τις υπηρεσίες που απαιτούνται για την λειτουργία του νέου Agent στην πλατφόρμα SOVP.

Με την περιγραφή της διαδικασίας Virtualization για τις συσκευές, ολοκληρώνεται η συνολική περιγραφή της Virtualization διαδικασίας που μπορεί να εφαρμοστεί στο σύστημα της SOVP. Συμπερασματικά λοιπόν, μπορούμε να αναφέρουμε ότι η SOVP πλατφόρμα ενσωματώνει την διαδικασία του Virtualization για δύο τύπους συνιστωσών (CAP και CRD) τις οποίες μπορεί ο ίδιος ο χρήστης να προσθέσει στο σύστημα οποιαδήποτε στιγμή με Plug and Play τρόπο.

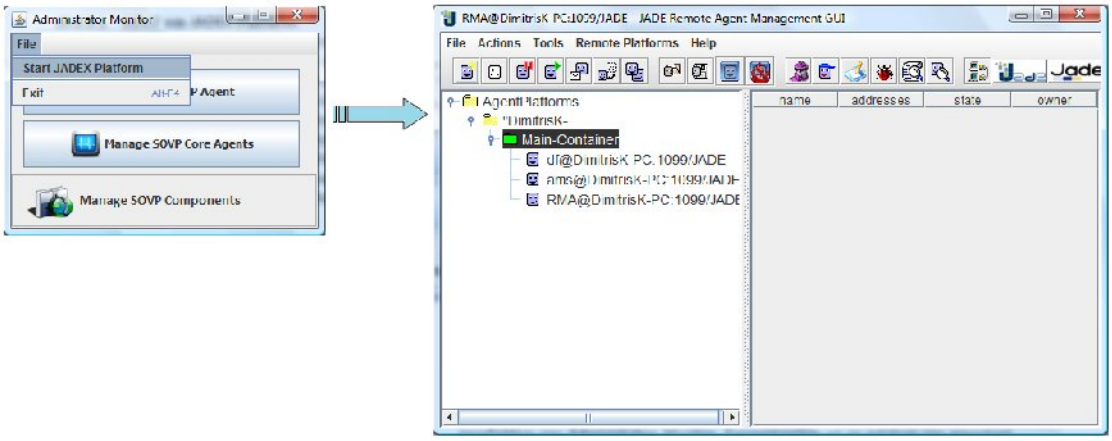
15.2.2 Φάση Λειτουργίας 2: SOVP Προσθήκη νέων συνιστωσών και εκκίνηση οντοτήτων πλατφόρμας

Το Administrator Monitor αποτελεί το δεύτερο βασικό κομμάτι της υλοποίησης της πλατφόρμας. Το Administrator Monitor παρέχει όλα τα απαραίτητα εργαλεία για την διαχείριση της πλατφόρμας SOVP, από τον χρήστη, και αποτελεί τον πυρήνα λειτουργίας της εφαρμογή. Τα προσφερόμενα εργαλεία είναι τα εξής:

1. **Start JADEX Platform:** εργαλείο για την εκκίνηση της πλατφόρμας JADEX
2. **Start a SOVP Agent:** εργαλείο για την εκκίνηση πρακτόρων της SOVP πλατφόρμας.
3. **Manage SOVP Core Agents:** εργαλείο διαχείριση των πρακτόρων παρακολούθησης που διαθέτει η πλατφόρμα για την παρακολούθηση της λειτουργίας της.

15.2.2.1 SOVP και JADEX Platform

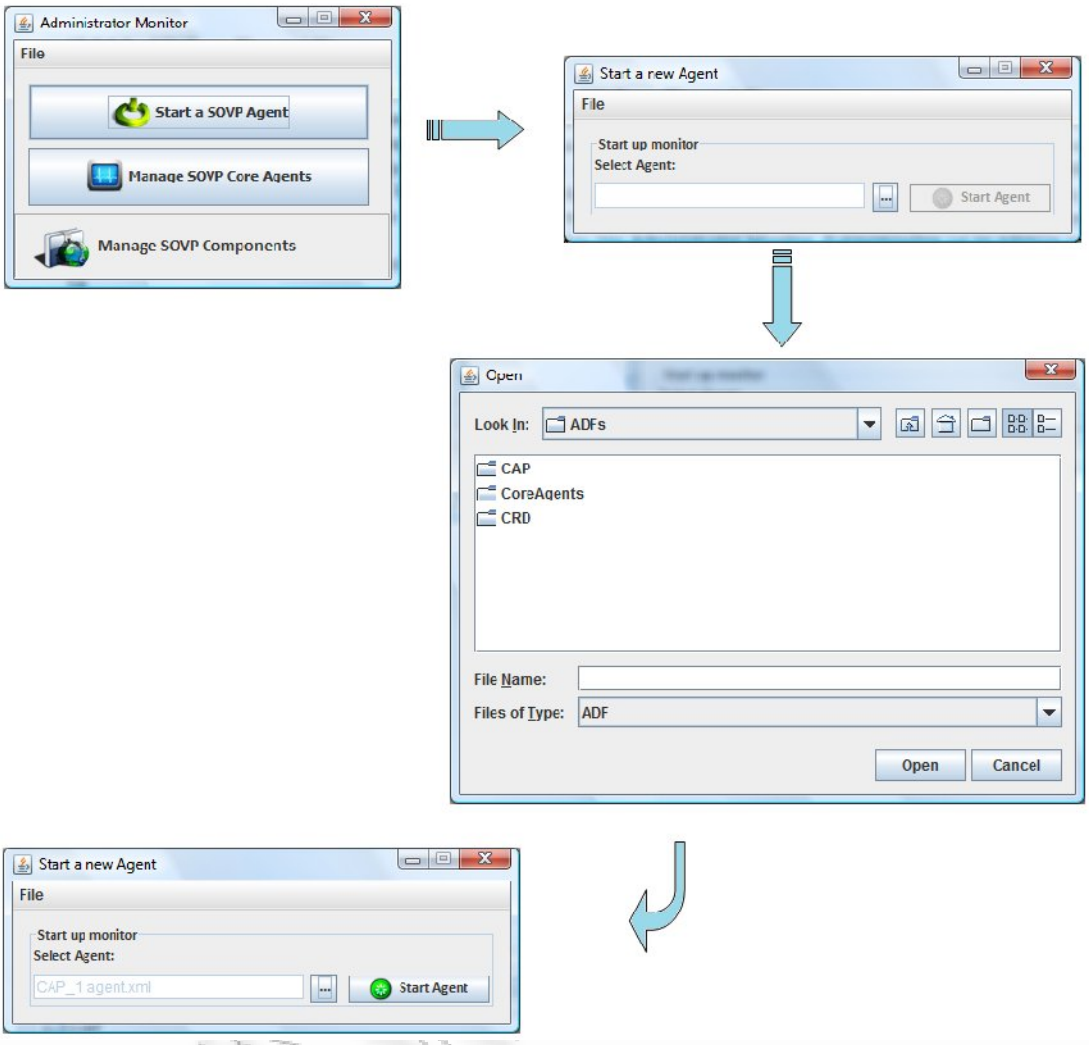
Η πλατφόρμα JADEX φιλοξενεί τους Agents που συνθέτουν την πλατφόρμα SOVP και τους δίνει την δυνατότητα να επικοινωνήσουν μεταξύ τους με την χρήση κατάλληλων ACL μηνυμάτων. Τα μηνύματα μεταφέρονται με την βοήθεια FIPA πρωτοκόλλων επικοινωνίας τα οποία είναι συμβατά με την JADEX πλατφόρμα. Επίσης η JADEX πλατφόρμα διαθέτει τον πράκτορα **Directory Facilitator**, ο οποίος παίζει τον ρόλο της **Registry οντότητας** του Service Oriented συστήματος μας. Τέλος ο πράκτορας **Sniffer**, της JADEX, αποτελεί ένα πολύ χρήσιμο εργαλείο για την παρακολούθηση της επικοινωνίας μεταξύ των agents της πλατφόρμας SOVP. Το αποτέλεσμα της εκκίνησης της JADEX πλατφόρμας, απεικονίζεται στην εικόνα 57.



Εικόνα 57: Εκκίνηση JADEX πλατφόρμας μέσω του Administrator Monitor

15.2.2.2 SOVP και Plug and Play

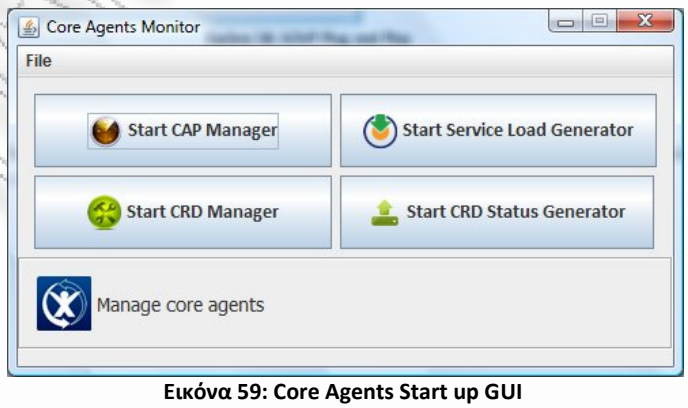
Μια από τις σημαντικότερες ιδιότητες της SOVP είναι η προσθήκη συνιστωσών με Plug and Play τρόπο. Όπως έχουμε ήδη περιγράψει παραπάνω, η ιδιότητα αυτή προκύπτει ως **αποτέλεσμα του συνδυασμού** της **Virtualization** διαδικασίας με μια **υπηρεσία εκκίνησης πρακτόρων** που διαθέτει η πλατφόρμα SOVP. Η υπηρεσία εκκίνησης πράκτορα συμπεριλαμβάνεται στο γραφικό περιβάλλον του Administrator Monitor. Ενεργοποιείται με το πάτημα του κουμπιού 'Start a SOVP Agent' και η διαδικασία που εκτελεί είναι η φόρτωση του Agent Definition File που προέκυψε από την διαδικασία του Virtualization που προηγήθηκε. Ο χρήστης αφού εντοπίσει το ADF του πράκτορα που θέλει να φορτώσει, το επιλέγει και μέσω κατάλληλου γραφικού το φορτώνει στην φόρμα εκκίνησης πρακτόρων. Το μόνο που απομένει είναι να πατηθεί το κουμπί 'Start Agent', που βρίσκεται στην φόρμα εκκίνησης, και ο πράκτορας τίθεται απευθείας σε λειτουργία, χωρίς να χρειάζεται η εκτέλεση κάποιας ειδικής διαδικασίας από τον χρήστη. Η υλοποίηση της διαδικασία Plug and Play παρουσιάζεται στην εικόνα 58.



Εικόνα 58: SOVP Plug and Play

15.2.2.3 SOVP Core Agents

Οι πράκτορες διαχείρισης που διαθέτει η πλατφόρμα, αποτελούν σημαντικά τμήματα της λειτουργίας της. Χρησιμοποιούνται ώστε να υλοποιήσουν σημαντικές λειτουργίες στην πλατφόρμα, τις οποίες περιγράψαμε αναλυτικά στο κεφάλαιο 14. Η εικόνα 59, παρουσιάζει το γραφικό περιβάλλον μέσω του οποίου ο χρήστης μπορεί να εκκινήσει καθέναν από τους διαθέσιμους Core Agents.



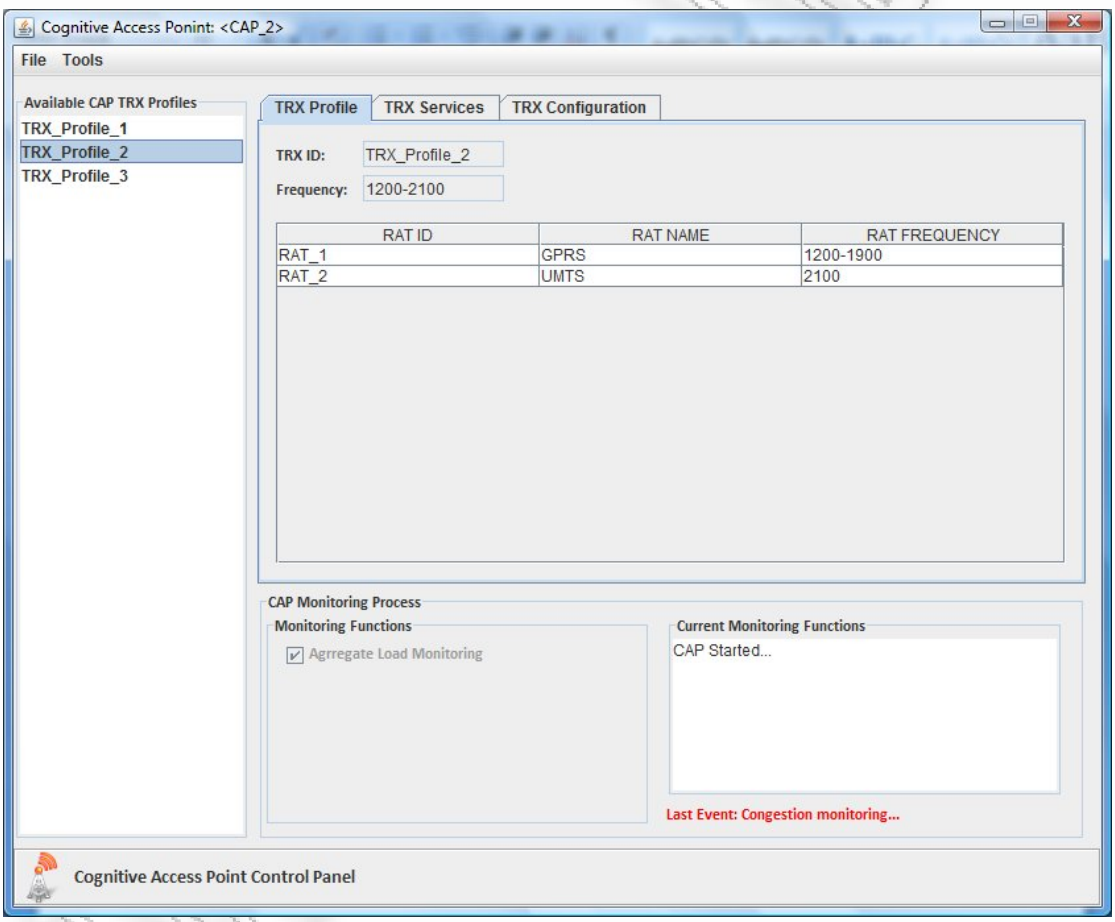
Εικόνα 59: Core Agents Start up GUI

15.2.3 Φάση Λειτουργίας 3: SOVP Εκτέλεση λειτουργίας των πρακτόρων

Η λειτουργία της πλατφόρμας SOVP μπορεί να χωριστεί και να περιγραφεί σε δύο φάσεις. Η πρώτη φάση περιλαμβάνει την εκτέλεση λειτουργίας για την διαχείριση συνιστωσών του τύπου Cognitive Access Point και η δεύτερη την διαχείριση συνιστωσών του τύπου Cognitive Reconfigurable Device. Ακολουθεί παρακάτω η περιγραφή της κάθε φάσης λειτουργίας.

15.2.3.1 Cognitive Access Point

Όπως ήδη έχουμε δει, η εκκίνηση ενός πράκτορα εκτελείται από την φόρμα εκκίνησης πράκτορα που υπάρχει στο Administrator Monitor. Φορτώνουμε λοιπόν έναν πράκτορα τύπου Cognitive Access Point, τον οποίο δημιουργήσαμε με την διαδικασία του Virtualization. Ο Agent έχει την ονομασία CAP_2 και το γραφικό του περιβάλλον παρουσιάζεται στην εικόνα 60. Να επισημάνουμε ότι το γραφικό περιβάλλον των CAP Agents, είναι ίδιο για όλους τους CAP Agents της πλατφόρμας.



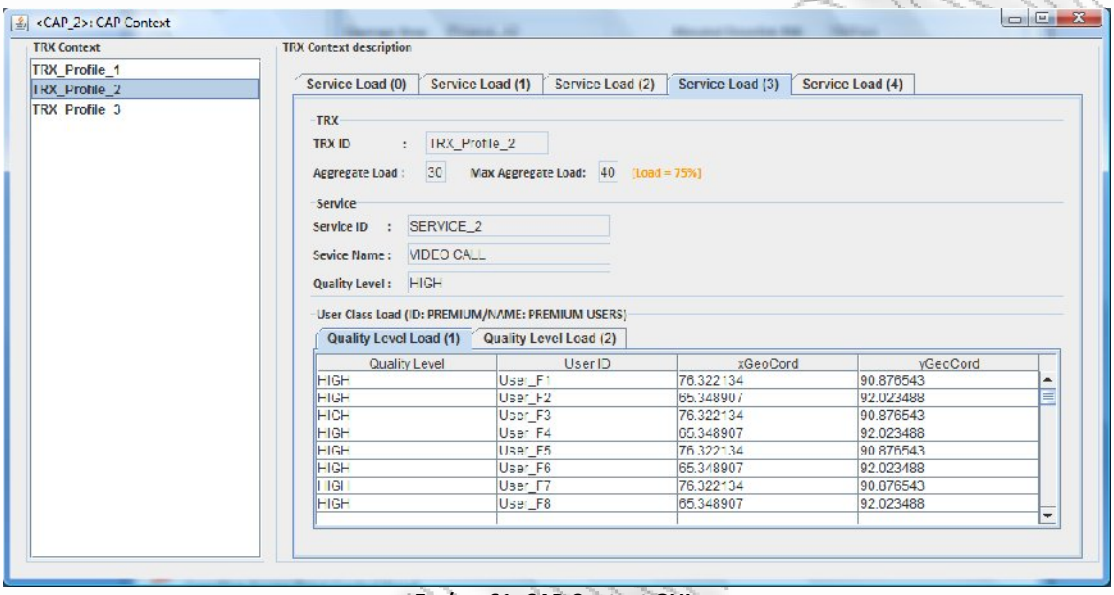
Εικόνα 60: CAP Agent GUI

Το GUI του πράκτορα CAP_2, περιλαμβάνει τρείς διαφορετικές καρτέλες με στοιχεία που αφορούν τον πράκτορα. Συγκεκριμένα υπάρχουν οι εξής καρτέλες:

- **TRX Profile:** καρτέλα που παρουσιάζει την περιγραφή του προφίλ του CAP Agent.
- **TRX Services:** καρτέλα που παρουσιάζει την περιγραφή των υπηρεσιών του CAP Agent.

- **TRX Configuration:** καρτέλα που παρουσιάζει την τρέχουσα διαμόρφωση λειτουργίας του CAP Agent.

Επιπλέον κάθε CAP Agent διαθέτει το εργαλείο προβολής του τρέχοντος περιεχομένου (CAP Context) του πράκτορα μέσο του οποίου ο χρήστης μπορεί να δει κάθε στιγμή τις πληροφορίες που αφορούν την τρέχουσα κατάσταση στο Context του CAP Agent. Η εικόνα 61 παρουσιάζει το GUI προβολής του CAP Context.

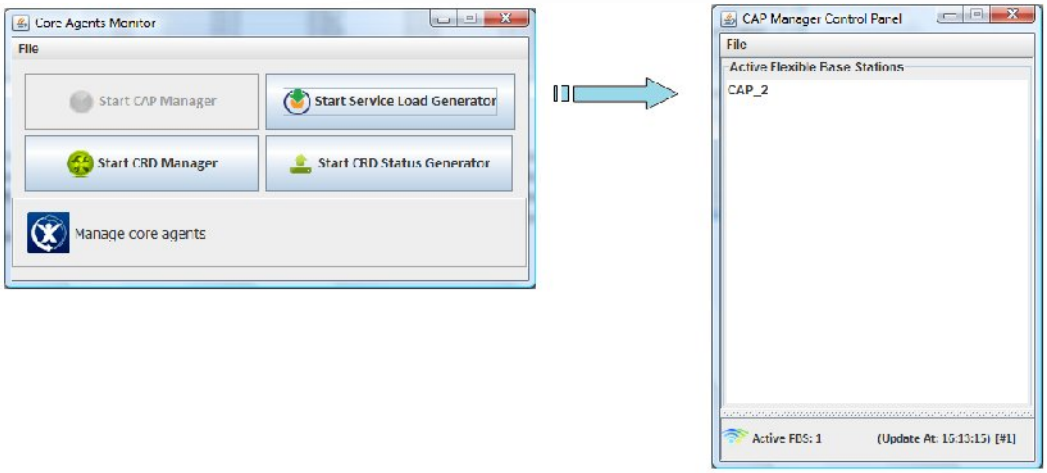


Εικόνα 61: CAP Context GUI

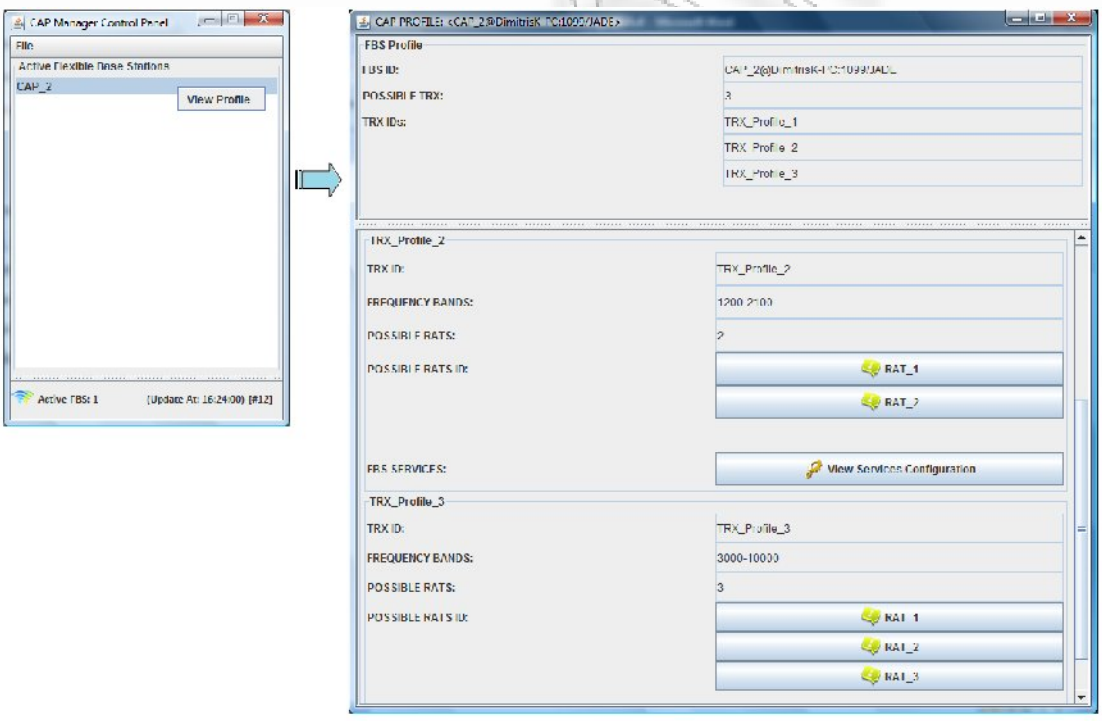
Στο GUI του CAP Agent παρατηρούμε επίσης ότι υπάρχει μια λίστα που παρουσιάζει τα διαθέσιμα TRX Profiles του πράκτορα. Ο χρήστης πατώντας στο όνομα του κάθε TRX Profile μπορεί να δει τις πληροφορίες που το αφορούν. Τέλος υπάρχει και ένα πλαίσιο, με τίτλο Current Monitoring Functions, το οποίο παρουσιάζει την λειτουργία που εκτελέστηκε τελευταία στον πράκτορα.

15.2.3.2 CAP Manager και Service Load Generator

Ο CAP Manager Agent αποτελεί τον πράκτορα παρακολούθησης και διαχείρισης των CAP Agents της SOVP. Η εκκίνηση του πραγματοποιείται μέσω του γραφικού περιβάλλοντος Core Agents Monitor και αφού ενεργοποιηθεί προβάλει ένα γραφικό περιβάλλον διαχείρισης το οποίο περιέχει μια λίστα με όλους τους διαθέσιμους CAP Agents που είναι συνδεδεμένοι στην πλατφόρμα. Ο χρήστης μπορεί να επιλέξει οποιονδήποτε χρήστη από την λίστα και κάνοντας δεξί κλικ πάνω του μπορεί να δει το προφίλ του. Στο προφίλ του πράκτορα περιλαμβάνονται όλες οι πληροφορίες που αφορούν τα TRX Profiles και όλες υπηρεσίες του πράκτορα. Οι εικόνες 62 και 63 παρουσιάζουν τις οθόνες των GUI που μόλις περιγράψαμε.



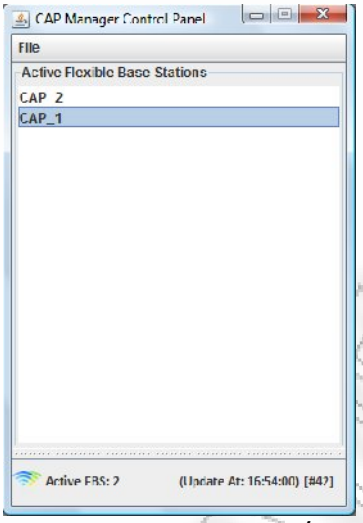
Εικόνα 62: CAP Manager GUI



Εικόνα 63: CAP Manager – CAP Profile Presenter

Για να κατανοήσουμε καλύτερα την χρησιμότητα του CAP Manager Agent θα δημιουργήσουμε έναν νέο Agent, μέσω του Virtualization Template Manager, με όνομα CAP_1 και καθώς η πλατφόρμα είναι σε λειτουργία θα τον συνδέσουμε σε αυτή με Plug and Play τρόπο. Η διαδικασία που ακολουθούμε για την δημιουργία και εκτέλεση ενός πράκτορα στην SOVP είναι ήδη γνωστή και γι' αυτό απλά θα παρουσιάσουμε ένα στιγμιότυπο του CAP Manager ο οποίος εντοπίζει την νέα συνισταμένη που προστέθηκε στο σύστημα και την εμφανίζει στην λίστα με όλες

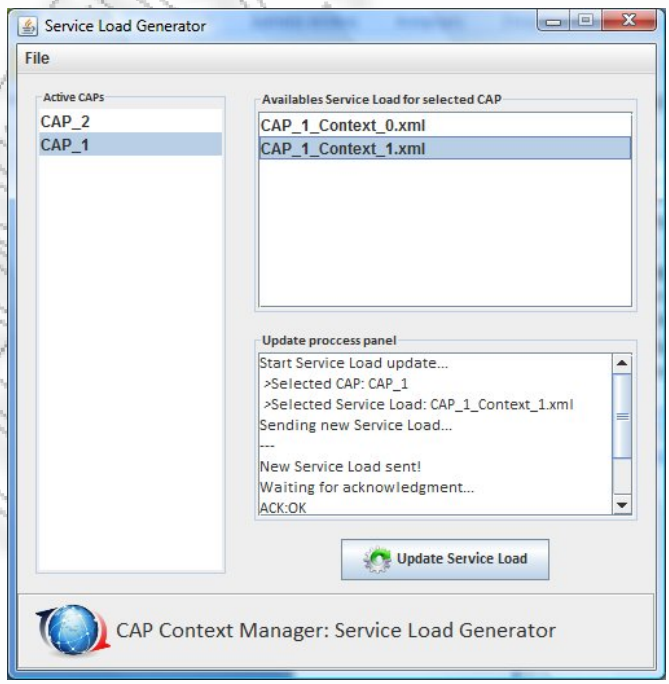
συνδεδεμένους CAP Agents της πλατφόρμας. Η εικόνα 64 απεικονίζει το στιγμιότυπο του CAP Manager Agent μετά από την σύνδεση του CAP_1 Agent.



Εικόνα 64: Στιγμιότυπο CAP Manager Agent – Εντοπισμός και εμφάνιση 2 CAP Agents

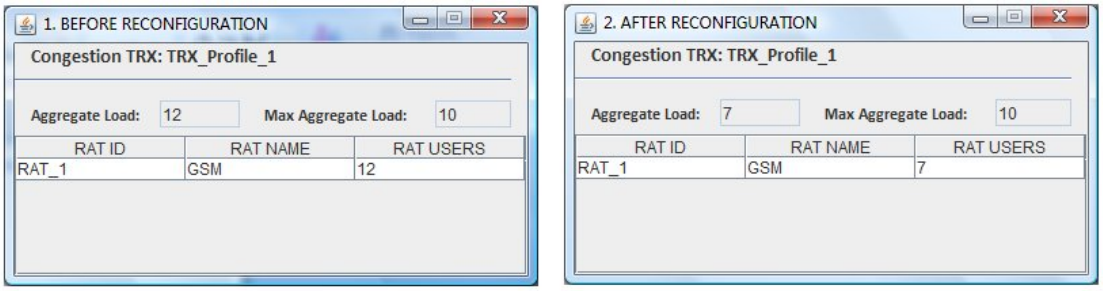
Μια επιπλέον πολύ σημαντική λειτουργία που διαθέτει ο CAP Manager Agent είναι, η δυνατότητα να πραγματοποιεί Reconfiguration στην λειτουργία ενός CAP Agent, στον οποίο προέκυψε συμφόρηση. Όταν προκύψει συμφόρηση στο TRX Profile ενός CAP Agent, τότε ο πράκτορας στέλνει απευθείας ένα Reconfiguration Request στον CAP Manager Agent ό οποίος με την σειρά του, αφού χρησιμοποιήσει την Decision Making Service που διαθέτει, δημιουργεί ένα νέο Reconfiguration το οποίο και στέλνει στον CAP Agent ώστε να γίνει η αποσυμφόρηση του TRX Profile.

Για να δημιουργήσουμε συμφόρηση σε έναν CAP Agent, έχουμε υλοποιήσει τον Service Load Generator, με GUI που παρουσιάζεται στην εικόνα 65.



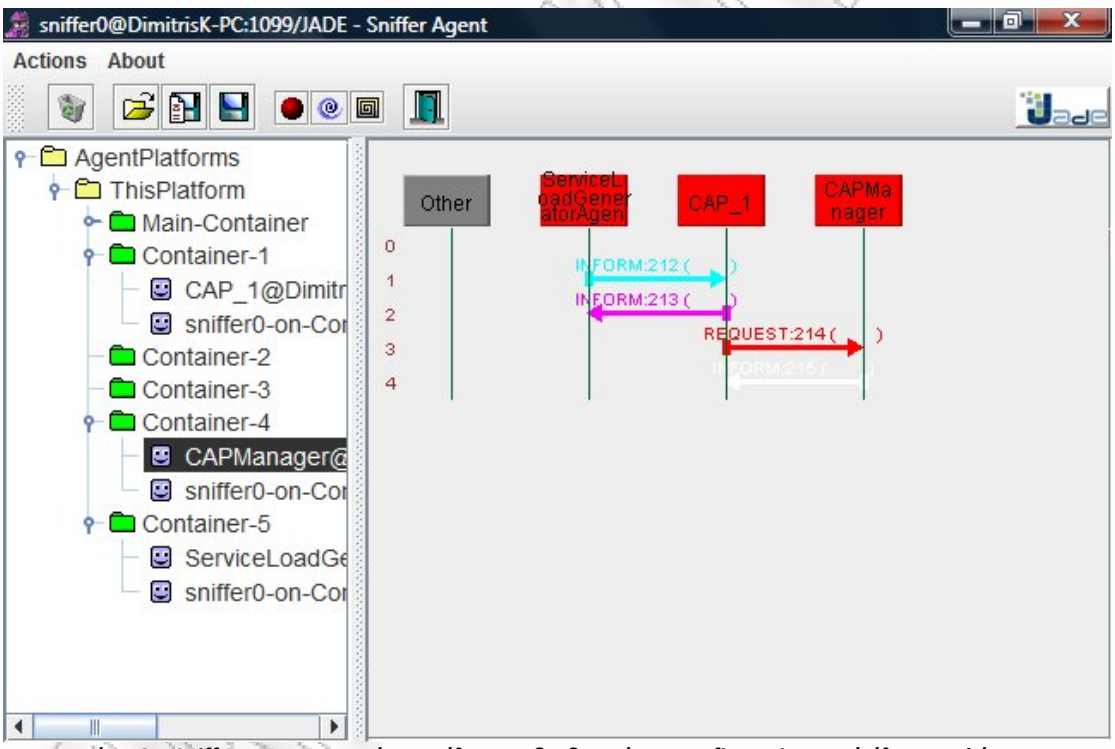
Εικόνα 65: Service Load Generator Agent

Η συμφόρηση σε ένα CAP Agent έχει ως αποτέλεσμα την εμφάνιση δύο pop up παραθύρων τα οποία παρουσιάζουν το φορτίο του TRX Profile με την συμφόρηση και το φορτίο του TRX Profile μετά την αποσυμφόρηση του (εικόνα 66).



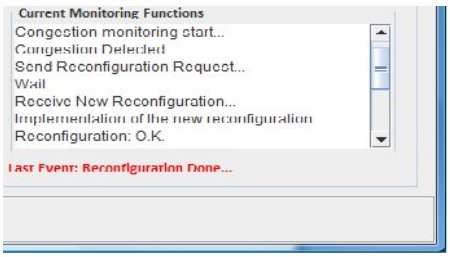
Εικόνα 66: Pop Up Congestion Windows

Με την βοήθεια του JADE Sniffer Agent μπορούμε να δούμε την ανταλλαγή μηνυμάτων μεταξύ των Agent της πλατφόρμας κατά την διαδικασία που με την αλλαγή φορτίου σε έναν CAP Agent, προκύπτει συμφόρηση και ο CAP Manager καλείται να λύσει την συμφόρηση μέσω Reconfiguration.



Εικόνα 67: Sniffer Agent – Μηνύματα όλες την διαδικασία Reconfiguration μετά όλες συμφόρηση

Να επισημάνουμε ότι όλες οι διαδικασίες που εκτελούνται στην πλευρά του CAP Agent κατά την διαδικασία που εντοπίζεται συμφόρηση και πραγματοποιείται Reconfiguration, παρουσιάζονται στο Current Monitoring Functions panel του CAP GUI, (εικόνα 68).



Εικόνα 68: Current Monitoring Function

Στο σημείο αυτό ολοκληρώνεται η περιγραφή των πιθανών σεναρίων λειτουργίας ενός CAP Agent στην SOVP πλατφόρμα. Στην συνέχεια, αναλύουμε εκτενώς όλα τα πιθανά σενάρια συμφόρησης που μπορεί να προκύψουν για δύο πρότυπους Agents της SOVP. Στην ανάλυση των σεναρίων συμφόρησης, περιγράφονται τα αποτελέσματα σε κάθε περίπτωσης και επίσης αναφέρονται όλες οι ενέργειες που εκτελούνται από του δύο συμμετέχοντες τις διαδικασίας, τον CAP και Agent CAP Manager Agent.

15.2.3.3 Σενάρια συμφόρησης και διαδικασία αποσυμφόρησης

Ένα σενάριο συμφόρησης υλοποιείται ως ένα καλά ορισμένο **(Well-formed) xml αρχείο** το οποίο περιέχει τις απαραίτητες πληροφορίες που απαιτούνται ώστε να πραγματοποιηθεί η ενημέρωση του **Context** ενός **Cognitive Access Point Agent (CAP Agent)**. Το CAP Context, εκτός των άλλων πληροφοριών που φέρει, διαθέτει περιγραφές για τα τρέχοντα **Service Load** που υπάρχουν σε κάθε TRX Profile του CAP Agent. Μέσω του Service Load, ο CAP Agent μπορεί να γνωρίζει το τρέχον φορτίο χρηστών σε κάθε υπηρεσία και κατ' επέκταση το φορτίο χρηστών σε κάθε RAT και TRX. Μόλις ο CAP Agent λάβει τις πληροφορίες που αφορούν το περιεχόμενό του, τις εφαρμόζει στις μεταβλητές λειτουργίας του και έτσι κάθε φορά έχουμε την αλλαγή του περιεχομένου σε πραγματικό χρόνο λειτουργίας (Live Context Update).

Η συμφόρηση δημιουργείται κάθε φορά από τον **Service Load Generator Agent (SLG Agent)** ο οποίος στέλνει τεχνητό φορτίο στον επιλεγμένο CAP Agent. Το φορτίο που αποστέλλεται σε έναν CAP Agent, δημιουργεί συμφόρηση μόνο σε ένα TRX Profile κάθε φορά. Δηλαδή αν για παράδειγμα ένα CAP έχει τρία διαθέσιμα TRX Profile, στη περίπτωση που ο SLG Agent στείλει τεχνητό φορτίο, προς αυτόν, θα δημιουργήσει συμφόρηση μόνο σε ένα από τα τρία διαθέσιμα TRX Profile.

Εφόσον εντοπιστεί συμφόρηση σε έναν CAP Agent, αποστέλλεται από τον ίδιο, ένα αίτημα αναδιαμόρφωσης φορτίου (**Reconfiguration Request**). Αποδέκτης του Reconfiguration Request είναι ο **CAP Manager Agent**, ο οποίος διαθέτει έναν αλγόριθμο διαχείρισης της συμφόρησης. Ο CAP Manager Agent αφού λάβει το Reconfiguration Request, εξάγει τις πληροφορίες από το μήνυμα. Στη συνέχεια αναλύει τις πληροφορίες και πραγματοποιεί κάποιους σχετικούς υπολογισμούς προκειμένου να καταλήξει στο τελικό αποτέλεσμα. Το αποτέλεσμα της επεξεργασίας των δεδομένων από τον CAP Manager Agent, περιέχει πληροφορίες σχετικά με την μεταφορά ή την απόρριψη χρηστών από το TRX Profile και το πλήθος των χρηστών που μεταβάλλονται. Στο σημείο αυτό πρέπει να αναφέρουμε ότι ο αλγόριθμος αποσυμφόρησης για την εξάλειψη της συμφόρησης, είναι σε θέση να πραγματοποιήσει δύο πιθανές ενέργειες: α) Μεταφορά χρηστών σε άλλο TRX Profile ή/και β) Απόρριψη χρηστών από το τρέχων TRX Profile.

Όταν ολοκληρωθεί η διαδικασία αποσυμφόρησης, στην πλευρά του CAP Manager Agent, δημιουργείται ένα **Reconfiguration Response** μήνυμα του οποίου παραλήπτης είναι ο CAP Agent που έστειλε το Reconfiguration Request. Με τον τρόπο αυτό ολοκληρώνεται η διαδικασία δημιουργίας και εντοπισμού της συμφόρησης σε έναν CAP Agent, και η διαδικασία αποσυμφόρησης του CAP Agent με την βοήθεια του CAP Manager Agent.

Στην συνέχεια περιγράφονται τα **σενάρια συμφόρησης** που αφορούν την λειτουργία των δύο πρότυπων χρησιμοποιούμενων CAP Agents (**CAP_1, CAP_2**) και έχουν διαμορφωθεί σύμφωνα με το προφίλ του κάθε CAP Agent, έτσι ώστε να πραγματοποιείται σωστά η λειτουργία τους. Ακολουθεί η αναλυτική περιγραφή των σεναρίων συμφόρησης για κάθε CAP Agent.

CAP_1 Agent: Σενάρια συμφόρησης

Για τον Agent CAP_1, έχει υλοποιηθεί μόνο ένα σενάριο συμφόρησης το οποίο αναλύεται στη συνέχεια.

Σενάριο 1^{ο} (Επιτυχής Μεταφορά όλων των επιπλέον χρηστών σε διαφορετικό TRX)
Εκτέλεση:

Το σενάριο συμφόρησης υλοποιείται από το xml αρχείο **CAP_1_Context_1.xml** και κατά την εκτέλεση του, δημιουργεί συμφόρηση στο TRX Profile 1 του CAP_1. Ενώ το Maximum Aggregate Load του TRX_Profile_1 είναι 10 χρήστες, το συγκεκριμένο προφίλ μετα την εφαρμογή το σεναρίου βρίσκεται να έχει φορτίο ίσο με 12 χρήστες. Το πλεόνασμα χρηστών (Overloaded Users) εντοπίζεται στο RAT_1 και συγκεκριμένα στην υπηρεσία VOICE, ενώ ο αριθμός τους ανέρχεται στους 2 χρήστες. Συνεπώς το σύστημα της SOVP θα πρέπει να ελέγξει αν υπάρχει η υπηρεσία VOICE σε κάποιο άλλο από τα διαθέσιμα TRX Profile του CAP Agent και αναλόγως το αποτέλεσμα, να πραγματοποιήσει τις απαραίτητες διαδικασίες αποσυμφόρησης του TRX Profile 1.

***Αποτελέσματα*:**

Το σύστημα κατόπιν ελέγχου, διαπιστώνει ότι οι επιπλέον χρήστες του **TRX Profile 1** μπορούν να **μεταφερθούν** στο **TRX Profile 2**. Ο αριθμός των χρηστών που μεταφέρονται στην προκειμένη περίπτωση είναι: **4 χρήστες**. Μετά την εκτέλεση της διαδικασίας αποσυμφόρησης οι 4 επιπλέον χρήστες μεταφέρονται από το TRX_Profile_1, RAT_1, στο TRX_Profile_2, RAT_1, όπου εντοπίστηκε η συμβατή υπηρεσία (VOICE). Έτσι η διαδικασία αποσυμφόρησης είναι επιτυχής ενώ επιπλέον δεν απορρίπτεται κανένας χρήστης από τον CAP_1.

Ο πίνακας 11 παρακάτω παρουσιάζει συνοπτικά το σενάριο συμφόρησης 1 για τον CAP_1.

CAP 1: Σενάριο 1 - (CAP_1_Context_1.xml)	
Congested TRX Profile	TRX_Profile_1
Congested RATs in TRX Profile	RAT_1
Congested Services in each RAT	VOICE
TRX Profile: Max Aggregate Load	10
TRX Profile: Current Aggregate Load	12
Overloaded Users	2
Decongestion Target	4
Users Move on	TRX_Profile_2
Users(Moved , Rejected)	(4 , 0)

Πίνακας 13: CAP_1 Σενάριο συμφόρηση Νο1

CAP_2 Agent: Σενάρια συμφόρησης

Για τον CAP_2, έχουν υλοποιηθεί τρία διαφορετικά σενάρια συμφόρησης. Το κάθε σενάριο υλοποιείται από ένα αντίστοιχο xml αρχείο **CAP_2_Context_i.xml** (όπου: i=1,2,3). Αναλύοντας το κάθε σενάριο ξεχωριστά, προκύπτουν τα εξής.

Σενάριο 1^{ο} (Επιτυχής Μεταφορά όλων των επιπλέον χρηστών σε διαφορετικό TRX)
Εκτέλεση:

Το πρώτο σενάριο συμφόρησης (CAP_2_Context_1.xml), κατά την εκτέλεση του, δημιουργεί συμφόρηση στο TRX Profile 1 του CAP_2. Ενώ το Maximum Aggregate Load του TRX_Profile_1 είναι 10 χρήστες, το συγκεκριμένο προφίλ μετα την εφαρμογή το σεναρίου βρίσκεται να έχει φορτίο ίσο με 13 χρήστες. Το πλεόνασμα χρηστών (Overloaded Users) εντοπίζεται στο RAT_1 και συγκεκριμένα στην υπηρεσία VOICE, ενώ ο αριθμός τους ανέρχεται στους 3 χρήστες. Συνεπώς το σύστημα της SOVP θα πρέπει να ελέγξει αν υπάρχει η υπηρεσία VOICE σε κάποιο άλλο από τα διαθέσιμα TRX Profile του CAP Agent και αναλόγως το αποτέλεσμα, να πραγματοποιήσει τις απαραίτητες διαδικασίες αποσυμφόρησης του TRX Profile 1.

***Αποτελέσματα*:**

Το σύστημα κατόπιν ελέγχου, διαπιστώνει ότι οι επιπλέον χρήστες του **TRX Profile 1** μπορούν να **μεταφερθούν** στο **TRX Profile 2**. Ο αριθμός των χρηστών που μεταφέρονται στην προκειμένη περίπτωση είναι: **5 χρήστες**. Μετά την εκτέλεση της διαδικασίας αποσυμφόρησης οι 5 επιπλέον χρήστες μεταφέρονται από το TRX_Profile_1, RAT_1, στο TRX_Profile_2, RAT_1, όπου εντοπίστηκε η συμβατή υπηρεσία (VOICE). Έτσι η διαδικασία αποσυμφόρησης είναι επιτυχής ενώ επιπλέον δεν απορρίπτεται κανένας χρήστης από τον CAP_2.

Ο πίνακας 12 παρακάτω παρουσιάζει συνοπτικά το σενάριο συμφόρησης 1 για τον CAP_2.

CAP 2: Σενάριο 1 - (CAP_2_Context_1.xml)	
Congested TRX Profile	TRX_Profile_1
Congested RATs in TRX Profile	RAT_1
Congested Services in each RAT	VOICE
TRX Profile: Max Aggregate Load	10
TRX Profile: Current Aggregate Load	13
Overloaded Users	3
Decongestion Target	5
Users Move on	TRX_Profile_2
Users(Moved , Rejected)	(5 , 0)

Πίνακας 14: CAP_2 Σενάριο συμφόρηση No1

Σενάριο 2^{ο} (Απόρριψη όλων των επιπλέον χρηστών από το TRX)

Εκτέλεση:

Το δεύτερο σενάριο συμφόρησης (CAP_2_Context_2.xml), κατά την εκτέλεση του, δημιουργεί συμφόρηση στο TRX Profile 2 του CAP_2. Ενώ το Maximum Aggregate Load του TRX_Profile_2 είναι 40 χρήστες, το συγκεκριμένο προφίλ μετα την εφαρμογή το σεναρίου βρίσκεται να έχει φορτίο ίσο με 44 χρήστες. Το πλεόνασμα χρηστών (Overloaded Users) εντοπίζεται αυτή τη φορά σε δύο RATs το RAT_1 και το RAT_2 και συγκεκριμένα στις υπηρεσίες DATA και VIDEO CALL, ενώ ο αριθμός τους ανέρχεται στους 4 χρήστες. Ωστόσο πρέπει να αναφέρουμε ότι το σύστημα εκτελεί έλεγχο μόνο για την υπηρεσία με το υψηλότερο φορτίο μεταξύ των δυο Congested Services. Στην περίπτωσή μας είναι η VIDEO CALL. Συνεπώς το σύστημα της SOVP θα πρέπει να ελέγξει αν υπάρχει η υπηρεσία VIDEO CALL σε κάποιο άλλο από τα διαθέσιμα TRX Profile του CAP Agent και αναλόγως το αποτέλεσμα, να πραγματοποιήσει τις απαραίτητες διαδικασίες αποσυμφόρησης του TRX Profile 2.

Αποτελέσματα:

Το σύστημα κατόπιν ελέγχου, διαπιστώνει ότι οι επιπλέον χρήστες του **TRX Profile 2 δεν** μπορούν να **μεταφερθούν** και επομένως πρέπει να **απορριφθούν**. Ο αριθμός των χρηστών που απορρίπτονται στην προκειμένη περίπτωση είναι: **8 χρήστες**. Έτσι η διαδικασία αποσυμφόρησης είναι επιτυχής με την διαφορά ότι υπάρχει απόρριψη χρηστών από τον CAP_2.

Ο πίνακας 13 παρακάτω παρουσιάζει συνοπτικά το σενάριο συμφόρησης 2 για τον CAP_2.

CAP 2: Σενάριο 2 - (CAP_2_Context_2.xml)	
Congested TRX Profile	TRX_Profile_1
Congested RATs in TRX Profile	RAT_1, RAT_2
Congested Services in each RAT	VIDEO CALL, DATA
TRX Profile: Max Aggregate Load	40
TRX Profile: Current Aggregate Load	44
Overloaded Users	4
Decongestion Target	8
Users Move on	-
Users(Moved , Rejected)	(0 , 8)

Πίνακας 15: CAP_2 Σενάριο συμφόρηση No2

Σενάριο 3^{ο} (Επιτυχής Μεταφορά συνόλου των επιπλέον χρηστών σε διαφορετικό TRX και ταυτόχρονη απόρριψη ενός μικρού αριθμού αυτών)

Εκτέλεση:

Το τρίτο σενάριο συμφόρησης (CAP_2_Context_3.xml), κατά την εκτέλεση του, δημιουργεί συμφόρηση στο TRX Profile 3 του CAP_2. Ενώ το Maximum Aggregate Load του TRX_Profile_3 είναι 40 χρήστες, το συγκεκριμένο προφίλ μετα την εφαρμογή το σεναρίου βρίσκεται να έχει φορτίο ίσο με 42 χρήστες. Το πλεόνασμα χρηστών (Overloaded Users) εντοπίζεται στο RAT_1 και συγκεκριμένα στην υπηρεσία EMAIL, ενώ ο αριθμός τους ανέρχεται στους 2 χρήστες. Συνεπώς το σύστημα της SOVP θα πρέπει να ελέγξει αν υπάρχει η υπηρεσία EMAIL σε κάποιο άλλο από τα διαθέσιμα TRX Profile του CAP Agent και αναλόγως το αποτέλεσμα, να πραγματοποιήσει τις απαραίτητες διαδικασίες αποσυμφόρησης του TRX Profile 3.

***Αποτελέσματα*:**

Το σύστημα κατόπιν ελέγχου, διαπιστώνει ότι από τους επιπλέον χρήστες του **TRX Profile 3** μπορεί να **μεταφερθεί** στο **TRX Profile 2** μόνο ένας συγκεκριμένος αριθμός**,** ενώ οι υπόλοιποι επιπλέον χρήστες θα **απορριφθούν**. Ο αριθμός των χρηστών που μεταφέρονται στην προκειμένη περίπτωση είναι: **6 χρήστες,** ενώ ο αριθμός των χρηστών που θα απορριφθούν είναι: **2 χρήστες**. Συνολικά στο TRX Profile 3 έχουμε μεταβολή του φορτίου κατά 8 χρήστες. Μετά την εκτέλεση της διαδικασίας αποσυμφόρησης, οι 6 χρήστες μεταφέρονται από το TRX_Profile_3, RAT_1, στο TRX_Profile_2, RAT_1, όπου εντοπίστηκε η συμβατή υπηρεσία (EMAIL) ενώ οι υπόλοιποι 2 καταργούνται εντελώς από το σύστημα. Στο σημείο αυτό είναι απαραίτητο να εξηγήσουμε ότι η απόρριψη των 2 χρηστών πραγματοποιείται εξαιτίας της αδυναμία του TRX Profile 2 να δεχτεί τον συνολικό αριθμό των μεταφερόμενων χρηστών, όπου αρχικά ανέρχεται στους 8 χρήστες. Έτσι η διαδικασία αποσυμφόρησης είναι επιτυχής με την διαφορά ότι υπάρχει ταυτόχρονη μεταφορά και απόρριψη χρηστών από τον CAP_2.

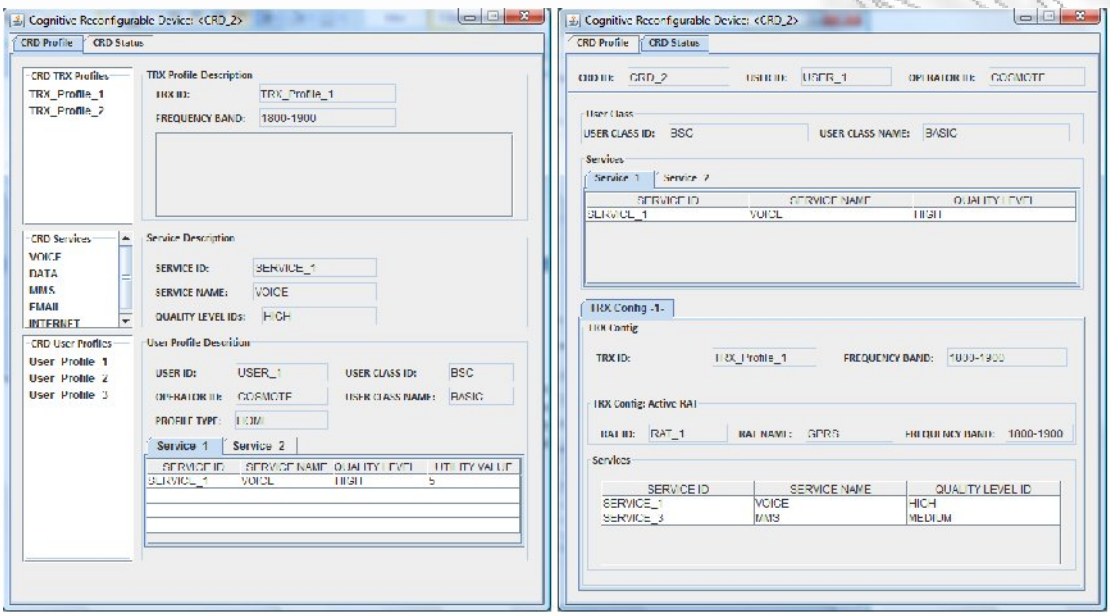
Ο πίνακας 14 παρακάτω παρουσιάζει συνοπτικά το σενάριο συμφόρησης 3 για τον CAP_2.

CAP 2: Σενάριο 3 - (CAP_2_Context_3.xml)	
Congested TRX Profile	TRX_Profile_3
Congested RATs in TRX Profile	RAT_1
Congested Services in each RAT	EMAIL
TRX Profile: Max Aggregate Load	40
TRX Profile: Current Aggregate Load	42
Overloaded Users	2
Decongestion Target	8
Users Move on	TRX_Profile_2
Users(Moved , Rejected)	(6 , 2)

Πίνακας 16: CAP_2 Σενάριο συμφόρηση No3

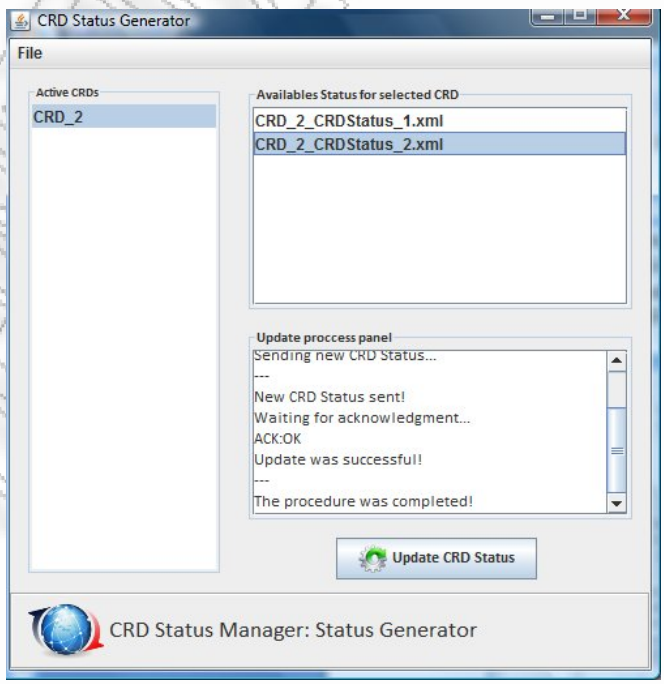
15.2.3.4 Cognitive Reconfigurable Device / CRD Status Generator / CRD Manager

Ο CRD πράκτορας ενεργοποιείται από την φόρμα εκκίνησης πρακτόρων του Administrator Monitor, και με την ενεργοποίηση του εμφανίζεται το GUI της εικόνας 69. Μέσα από το GUI του CRD Agent, ο χρήστης μπορεί να έχει στη διάθεση του πληροφορίες σχετικά με το προφίλ του πράκτορα (CRD Profile) και με την τρέχουσα κατάστασή του (CRD Status).



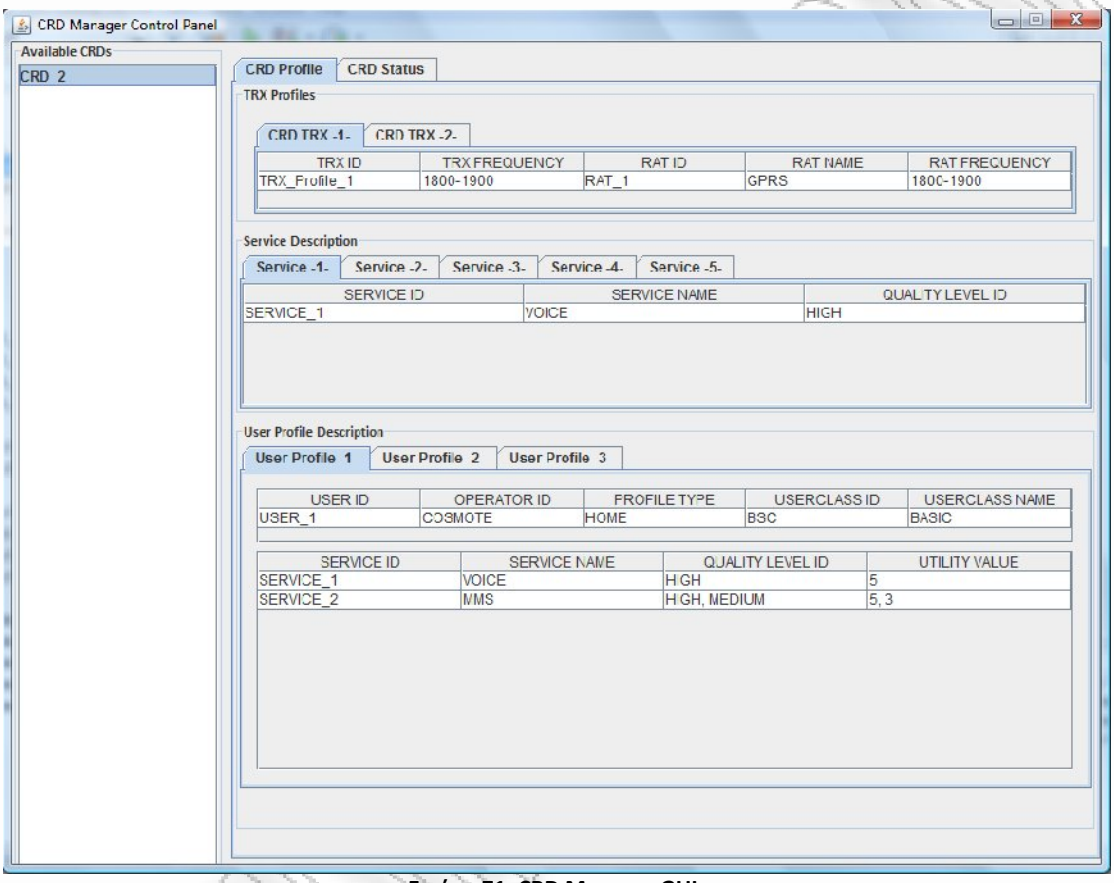
Εικόνα 69: CRD Agent GUI

Η τρέχουσα κατάσταση του πράκτορα, μπορεί να μεταβληθεί κάθε στιγμή λειτουργίας του. Προκειμένου να προσομοιώσουμε την διαδικασία αλλαγής του CRD Status, έχουμε δημιουργήσει τον CAP Status Generator Agent, το GUI του οποίου παρουσιάζεται την εικόνα 70.



Εικόνα 70: CRD Status Generator GUI

Η διαχείριση της κατάστασης των CRD Agents, πραγματοποιείται από τον CRD Manager Agent, ο οποίο κάθε στιγμή έχει στη διάθεση του, το προφίλ και την τρέχουσα κατάσταση όλων των ενεργών CRD Agents της πλατφόρμας SOVP. Στο γραφικό περιβάλλον του CRD Manager Agent παρουσιάζεται και η λίστα με όλους τους διαθέσιμους CRD Agents. Η λίστα ανανεώνεται κάθε 1 λεπτό. Με τον τρόπο αυτό είμαστε σε θέση να γνωρίζουμε κάθε στιγμή τις προσθήκες και τις αφαιρέσεις πρακτόρων τύπου CRD.



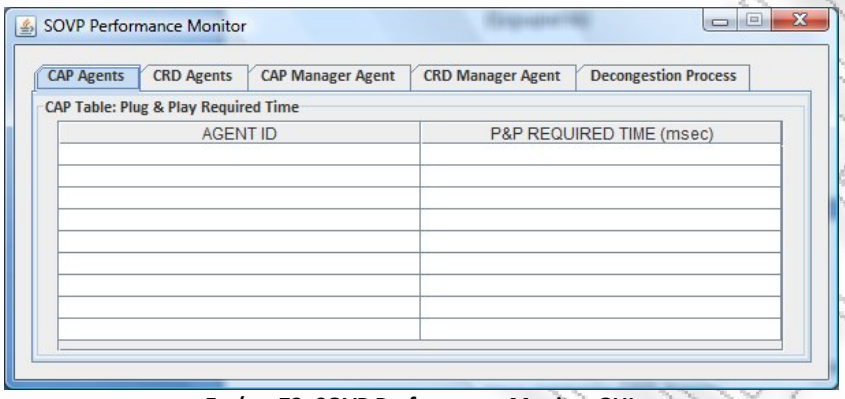
Εικόνα 71: CRD Manager GUI

15.3 Αποτελέσματα εκτέλεσης

Η παρούσα ενότητα παρουσιάζει τα ενδεικτικά αποτελέσματα, σε επίπεδο χρονικών επιδόσεων, της εφαρμογής που υλοποιήσαμε. Συγκεκριμένα μελετάμε την απόδοση του συστήματος ως προς το χρόνο, σε πέντε βασικές φάσεις λειτουργίας. Αυτές είναι:

- Χρόνος εκκίνησης CAP Agent
- Χρόνος εκκίνησης CRD Agent
- Χρόνος ολοκλήρωσης επικοινωνίας μεταξύ CAP Agent - CAP Manager και συνολικός χρόνος επικοινωνίας με όλους τους ενεργούς πράκτορες
- Χρόνος ολοκλήρωσης επικοινωνίας μεταξύ CRD Agent - CRD Manager και συνολικός χρόνος επικοινωνίας με όλους τους ενεργούς πράκτορες
- Συνολικός χρόνος Αποσυμφόρησης ενός CAP Agent από τον CAP Manager

Για την μελέτη των συγκεκριμένων αποτελεσμάτων μας βοήθησε το εργαλείο **SOVP System Performance** το οποίο υλοποιήσαμε ακριβώς για αυτόν τον λόγο. Το γραφικό περιβάλλον του συγκεκριμένου εργαλείου, παρουσιάζεται στην εικόνα 72 παρακάτω.



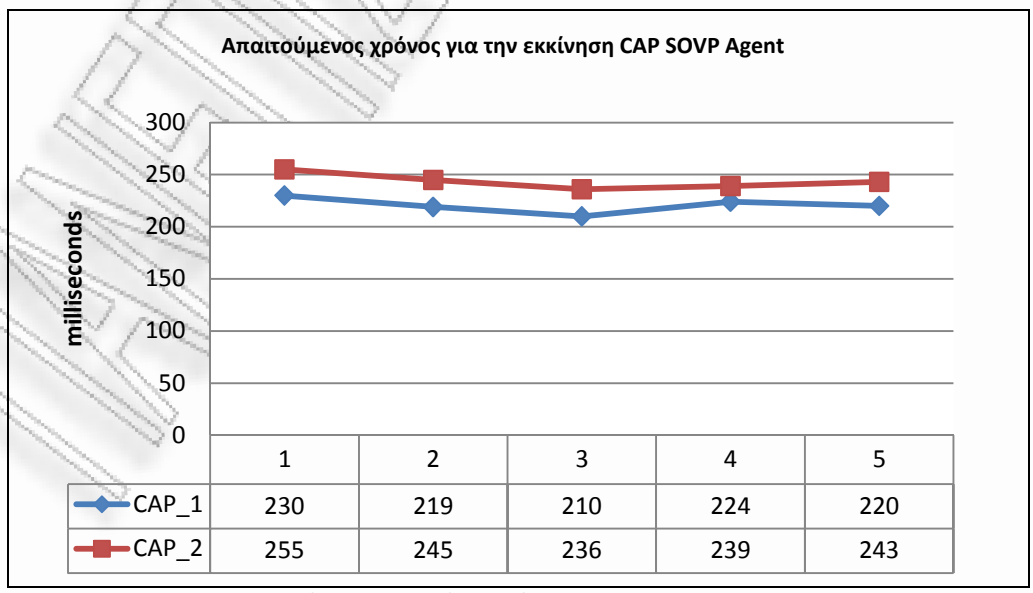
Εικόνα 72: SOVP Performance Monitor GUI

Στη συνέχεια παρουσιάζουμε τις ενδεικτικές μετρήσεις που πραγματοποιήθηκαν για καθεμιά περίπτωση.

15.3.1 Χρόνος εκκίνησης CAP Agent

Το παρακάτω διάγραμμα απεικονίζει τους χρόνους εκκίνησης των δύο CAP Agent που χρησιμοποιούμε στα σενάρια λειτουργίας της πλατφόρμας SOVP. Έχουμε πραγματοποιήσει πέντε φορές εκκίνηση για κάθε πράκτορα, προκειμένου να έχουμε μια ενδεικτική μέση τιμή του χρόνου εκκίνησης. Σύμφωνα με το αποτέλεσμα και κάνοντας μια απλή πράξη εύρεσης του μέσου χρόνου εκκίνησης για τον κάθε πράκτορα, έχουμε τα εξής αποτελέσματα:

- Μέσος χρόνος εκκίνησης CAP_1 = **221msec**
- Μέσος χρόνος εκκίνησης CAP_2 = **243msec**



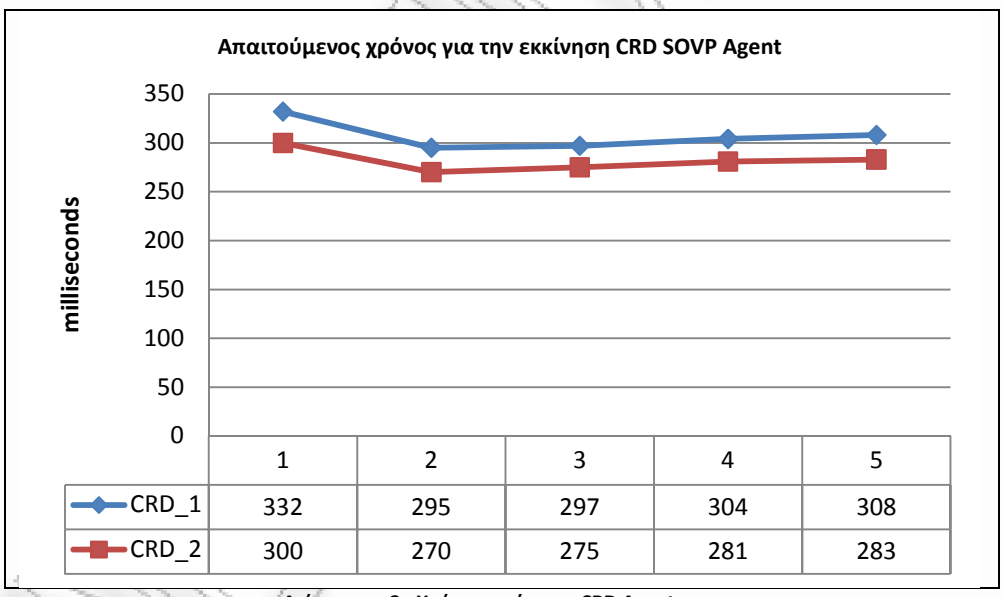
Διάγραμμα 1: Χρόνος εκκίνησης CAP Agent

Από την παραπάνω ενδεικτική μέτρηση μπορούμε να συμπεράνουμε ότι ο χρόνος εκκίνησης ενός πράκτορα τύπου CAP, δεν ξεπερνά τα **260msec**. Οπότε μπορούμε να πούμε με βεβαιότητα ότι η χρονική απόδοση της εφαρμογής είναι πολύ ικανοποιητική. Ωστόσο είναι απαραίτητο να επισημάνουμε ότι στην περίπτωση που ο CAP Agent αποτελεί την πρώτη οντότητα που ενεργοποιείται στην πλατφόρμα, ο χρόνος είναι αυξημένος για μια και μονό μια φορά κατά την εκκίνηση. Συγκεκριμένα ο χρόνος πρώτης ενεργοποίησης ενός JADEX CAP Agent, είναι περίπου 2,5 seconds.

15.3.2 Χρόνος εκκίνησης CRD Agent

Το παρακάτω διάγραμμα απεικονίζει τους χρόνους εκκίνησης των δύο CRD Agent που χρησιμοποιούμε στα σενάρια λειτουργίας της πλατφόρμας SOVP. Έχουμε πραγματοποιήσει πέντε φορές εκκίνηση για κάθε πράκτορα, προκειμένου να έχουμε μια ενδεικτική μέση τιμή του χρόνου εκκίνησης. Σύμφωνα με το αποτέλεσμα και κάνοντας μια απλή πράξη εύρεσης του μέσου χρόνου εκκίνησης για τον κάθε πράκτορα, έχουμε τα εξής αποτελέσματα:

- Μέσος χρόνος εκκίνησης CRD_1 = **307msec**
- Μέσος χρόνος εκκίνησης CRD_2 = **281msec**

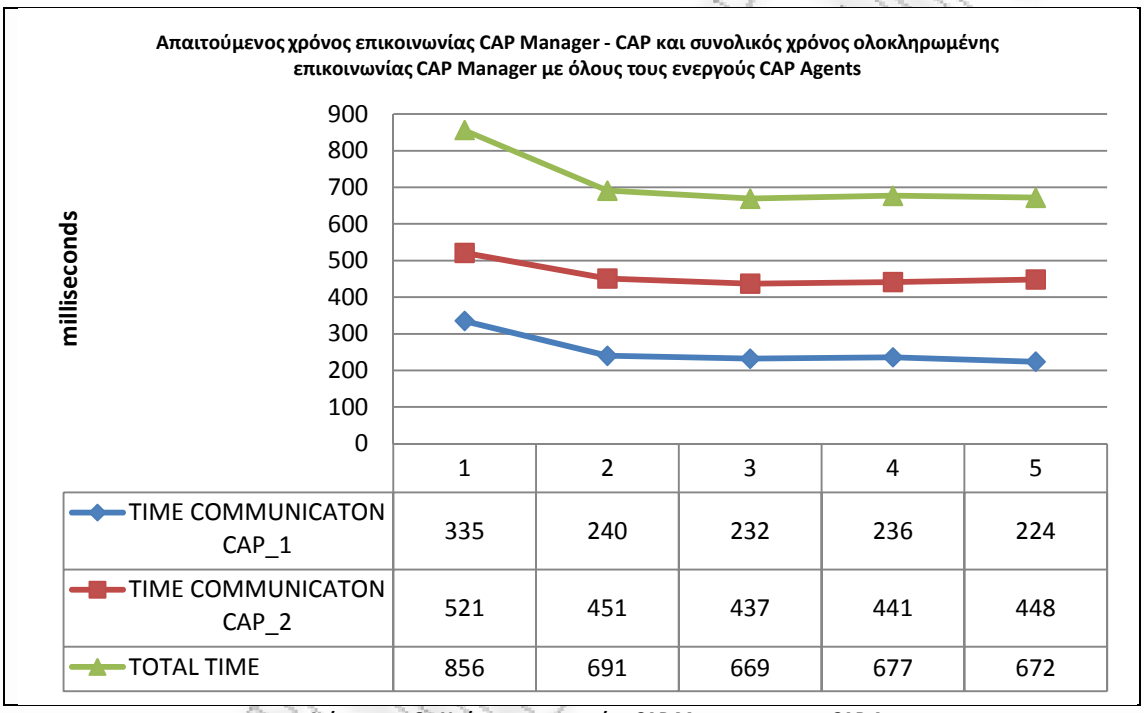


Διάγραμμα 2: Χρόνος εκκίνησης CRD Agent

Από την παραπάνω ενδεικτική μέτρηση μπορούμε να συμπεράνουμε ότι ο χρόνος εκκίνησης ενός πράκτορα τύπου CRD, δεν ξεπερνά τα **350msec**. Οπότε μπορούμε να πούμε με βεβαιότητα ότι η χρονική απόδοση της εφαρμογής είναι πολύ ικανοποιητική. Ωστόσο είναι απαραίτητο να επισημάνουμε, όπως και για του CAP Agents προηγουμένως, ότι στην περίπτωση που ο CRD Agent αποτελεί την πρώτη οντότητα που ενεργοποιείται στην πλατφόρμα, ο χρόνος είναι αυξημένος για μια και μονό μια φορά κατά την εκκίνηση. Συγκεκριμένα ο χρόνος πρώτης ενεργοποίησης ενός JADEX CRD Agent, είναι περίπου 2,1 seconds.

15.3.3 Χρόνος ολοκλήρωσης επικοινωνίας μεταξύ CAP Agent - CAP Manager και συνολικός χρόνος επικοινωνίας με όλους τους ενεργούς πράκτορες

Η τρίτη φάση μέτρησης της χρονικής απόδοσης του συστήματος αφορά την χρονική απόδοση της εφαρμογής στην επικοινωνία μεταξύ του CAP Manager με κάθε ενεργό CAP Agent ξεχωριστά, αλλά και την χρονική απόδοση του συνολικού χρόνου επικοινωνίας. Δηλαδή του απαιτούμενου χρόνου ώστε ο CAP Manager να επικοινωνήσει με όλους τους ενεργούς CAP Agents της πλατφόρμας SOVP. Το παρακάτω διάγραμμα παρουσιάζει τα αποτελέσματα αυτής της διαδικασίας και βάσει αυτών μπορούμε να συμπεράνουμε ότι για να ολοκληρωθεί η επικοινωνία μεταξύ CAP Manager με όλους τους ενεργούς CAP Agents, δεν απαιτείται χρόνος μεγαλύτερος των **900 msec** δηλαδή **0,9 second**.



	1	2	3	4	5
TIME COMMUNICATON CAP_1	335	240	232	236	224
TIME COMMUNICATON CAP_2	521	451	437	441	448
TOTAL TIME	856	691	669	677	672

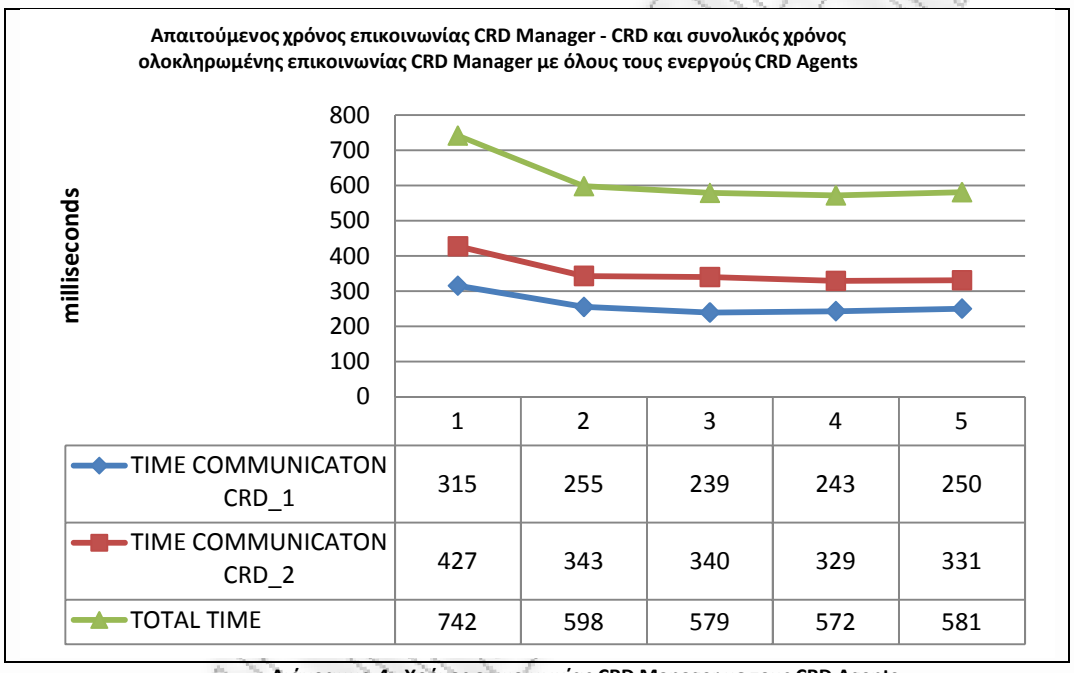
Διάγραμμα 3: Χρόνος επικοινωνίας CAP Manager με τους CAP Agents

Στο σημείο αυτό θα πρέπει να επισημάνουμε ότι τα παραπάνω αποτελέσματα είναι ενδεικτικά για την ύπαρξη 2 ενεργών CAP Agents στην πλατφόρμα, οπότε αναλογικά μπορούμε να αναφέρουμε ότι ο συνολικός χρόνος επικοινωνίας θα αυξάνεται κατά 0,9 δευτερόλεπτα για κάθε 2 CAP πράκτορες που προστίθενται στην πλατφόρμα (πχ: για 10 ενεργούς πράκτορες η τιμή θα είναι περίπου 4,5 seconds). Επίσης σημαντικό είναι να παρατηρήσουμε ότι στη γραμμή του διαγράμματος του συνολικού χρόνου, παρατηρούμε στην αρχή ότι βρίσκεται κοντά στα 850 msec, ενώ στη συνέχεια μειώνεται παρουσιάζοντας τάση σταθεροποίησης γύρω από τα 700msec. Η αρχική αυξημένη τιμή του χρόνου, οφείλεται στο ότι υπάρχει επικοινωνία για πρώτη φορά μεταξύ του CRD Manager με τους CRD Agents, και εκτός από τις πληροφορίες του Profile και του Context, λαμβάνει πληροφορίες σχετικά με την κατάσταση και τα στοιχεία του κάθε πράκτορα πάνω στην JADEX πλατφόρμα (πχ: Agent ID για αποστολή ACL Messages μέσω FIPA Protocols).

Σύμφωνα με τις παραπάνω μετρήσεις, μπορούμε να πούμε με βεβαιότητα ότι η χρονική απόδοση της εφαρμογής είναι πολύ ικανοποιητική για την διαδικασία επικοινωνίας μεταξύ CAP Manager με τους ενεργούς CAP Agents.

15.3.4 Χρόνος ολοκλήρωσης επικοινωνίας μεταξύ CRD Agent - CRD Manager και συνολικός χρόνος επικοινωνίας με όλους τους ενεργούς πράκτορες

Μια επιπλέον σημαντική μέτρηση είναι η χρονική απόδοση της εφαρμογής στην επικοινωνία μεταξύ του CRD Manager με κάθε ενεργό CRD Agent ξεχωριστά, αλλά και την χρονική απόδοση του συνολικού χρόνου επικοινωνίας. Δηλαδή του απαιτούμενου χρόνου ώστε ο CRD Manager να επικοινωνήσει με όλους τους ενεργούς CRD Agents της πλατφόρμας SOVP. Το παρακάτω διάγραμμα παρουσιάζει τα αποτελέσματα αυτής της διαδικασίας.



	1	2	3	4	5
TIME COMMUNICATON CRD_1	315	255	239	243	250
TIME COMMUNICATON CRD_2	427	343	340	329	331
TOTAL TIME	742	598	579	572	581

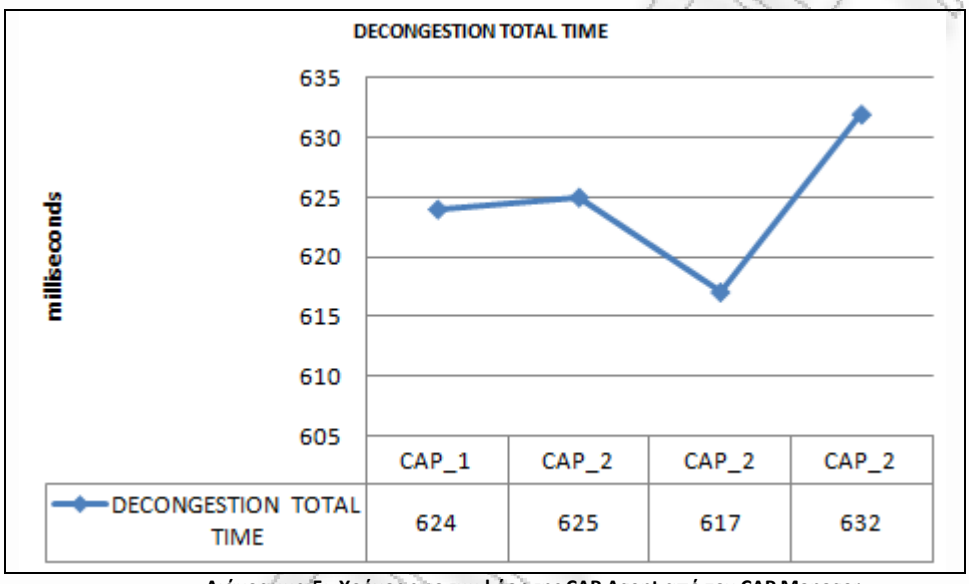
Διάγραμμα 4: Χρόνος επικοινωνίας CRD Manager με τους CRD Agents

Σύμφωνα με τις παραπάνω μετρήσεις, μπορούμε να συμπεράνουμε ότι ο συνολικός χρόνος επικοινωνίας CRD Manager με όλους τους ενεργούς CRD Agents, δεν ξεπερνά τα **750 msec** δηλαδή **0,75 second**. Ωστόσο, πρέπει να επισημάνουμε ότι τα παραπάνω αποτελέσματα είναι ενδεικτικά για την ύπαρξη 2 ενεργών CRD Agents στην πλατφόρμα, οπότε αναλογικά μπορούμε να αναφέρουμε ότι ο συνολικός χρόνος επικοινωνίας θα αυξάνεται κατά 0,75 δευτερόλεπτα για κάθε 2 CRD πράκτορες που προστίθενται στην πλατφόρμα (πχ: για 10 ενεργούς πράκτορες η τιμή θα είναι περίπου 3,75 seconds). Επίσης σημαντικό είναι να παρατηρήσουμε ότι στη γραμμή του διαγράμματος του συνολικού χρόνου, παρατηρούμε στην αρχή ότι βρίσκεται κοντά στα 750 msec, ενώ στη συνέχεια μειώνεται παρουσιάζοντας τάση σταθεροποίησης γύρω από τα 600msec. Η αρχική αυξημένη τιμή του χρόνου, οφείλεται στο ότι υπάρχει επικοινωνία για πρώτη φορά μεταξύ του CRD Manager με τους CRD Agents, και εκτός από τις πληροφορίες του Profile και του Context, λαμβάνει πληροφορίες σχετικά με την κατάσταση και τα στοιχεία του κάθε πράκτορα πάνω στην JADEX πλατφόρμα (πχ: Agent ID για αποστολή ACL Messages μέσω FIPA Protocols).

Σύμφωνα με τις παραπάνω μετρήσεις, μπορούμε να πούμε με βεβαιότητα ότι η χρονική απόδοση της εφαρμογής είναι πολύ ικανοποιητική για την διαδικασία επικοινωνίας μεταξύ CRD Manager με τους ενεργούς CRD Agents.

15.3.5 Συνολικός χρόνος Αποσυμφόρησης ενός CAP Agent από τον CAP Manager

Η ολοκλήρωση των μετρήσεων χρονικής απόδοσης της εφαρμογής, ολοκληρώνεται με την μέτρηση του συνολικού χρόνου ολοκλήρωσης της διαδικασίας της αποσυμφόρησης. Μετά από δοκιμές που πραγματοποιήθηκα, εφαρμόζοντας τα σενάρια συμφόρησης που περιγράψαμε παραπάνω, προκύπτει το παρακάτω διάγραμμα.



Διάγραμμα 5: Χρόνος αποσυμφόρησης CAP Agent από τον CAP Manager

Παρατηρώντας το διάγραμμα μπορούμε να διαπιστώσουμε ότι ο συνολικός χρόνος αποσυμφόρησης δεν ξεπερνά τα **640msec.** Η μέτρηση πραγματοποιήθηκε για τα τέσσερα σενάρια συμφόρησης που περιγράψαμε στην ενότητα 15.2.3.3, και θα μπορούσαμε να πούμε ότι αποτελούν αντιπροσωπευτικό δείγμα της συνολικής χρονικής διαδικασίας της αποσυμφόρησης.

Παρατηρούμε ότι η διαγραμματική αναπαράσταση έχει διαφορετικές τιμές με αυξομειώσεις για κάθε φάση εκτέλεσης της αποσυμφόρησης. Αυτό συμβαίνει για δύο βασικούς λόγους, οι οποίο σχετίζονται με τον όγκο των πληροφοριών. Η πρώτη μορφή καθυστέρησης προκύπτει από την επεξεργασία του όγκου πληροφοριών είναι διαφορετικός σε κάθε Reconfiguration Request και πρέπει να τον επεξεργαστεί ο CAP Manager. Η δεύτερη μορφή καθυστέρησης προκύπτει κατά την μεταφορά των πληροφοριών του Reconfiguration Response. Εύκολα μπορούμε να αντιληφθούμε ότι όσο μεγαλύτερος ο αριθμός των μεταφερόμενων bytes του μηνύματος, τόσο μεγαλύτερη η καθυστέρηση παράδοσής του. Σύμφωνα με όσα αναφέραμε, από την παρατήρηση του διαγράμματος, καταλαβαίνουμε ότι ο μικρότερος όγκος πληροφοριών εντοπίζεται στο $2^{ο}$ σενάριο του CAP_2, ενώ ο μεγαλύτερος όγκος πληροφοριών εντοπίζεται στο $3^{ο}$ σενάριο του ίδιου Agent.

Κλείνοντας, μπορούμε να πούμε με βεβαιότητα ότι η χρονική απόδοση της εφαρμογής είναι πολύ ικανοποιητική για την διαδικασία αποσυμφόρησης ενός CAP Agent από τον CAP Manager Agent.

✓ Κεφάλαιο 16: «Συμπερασματικές Παρατηρήσεις»

Στο κεφάλαιο αυτό θα αναφερθούμε στα συμπεράσματα που προκύπτουν από την παρούσα μελέτη. Επίσης θα παραθέσουμε κάποιες σημαντικές, κατά την άποψη μας, προτάσεις για την επέκταση της υπάρχουσας εφαρμογής.

16.1 Συμπεράσματα της παρούσας μελέτης

Στην παρούσα μελέτη αναφερθήκαμε στις τεχνολογίες του μελλοντικού διαδικτύου και στην χρήση τους για την ανάπτυξη συστημάτων ικανών να ανταποκριθούν στις απαιτήσεις του. Ως αποτέλεσμα της μελέτης μας, παρουσιάσαμε μια σύγχρονης πλατφόρμα Service Oriented Virtualization Platform (SOVP), ικανή να ανταποκριθεί σε ένα σημαντικό κομμάτι των απαιτήσεων του μελλοντικού διαδικτύου. Η SOVP αποτελεί μια πλατφόρμα δυναμικά επεκτάσιμη η οποία επιτρέπει την προσθήκη νέων συνιστωσών στο σύστημα με Plug and Play τρόπο. Στις συνιστώσες που μπορούν να προστεθούν και να λειτουργήσουν στην πλατφόρμα, συμπεριλαμβάνονται μέρη δικτυακής υποδομής και γνωστικές συσκευές. Η οργάνωση των νέων συνιστωσών στο σύστημα, υπακούει στις αρχές τις Service Oriented Αρχιτεκτονικής. Επίσης η Virtualization διαδικασία αποτελεί βασικό λειτουργικό κομμάτι της πλατφόρμας για την εισαγωγή νέων συνιστωσών, αποκρύπτοντας την πολυπλοκότητα των διαδικασιών. Τέλος ο συνδυασμός των γνωστικών υπηρεσιών που συγκεντρώνει η SOVP, μπορεί να διαχειρίζεται δυναμικά την προσθήκη η την αφαίρεση συνιστωσών από το σύστημά.

Ανακεφαλαιώνοντας μπορούμε να αναφέρουμε ότι, η SOVP αποτελεί ένα δυναμικά επεκτάσιμο σύστημα το οποίο συνδυάζει την διαδικασία του **Virtualization**, για την εισαγωγή συνιστωσών στο σύστημα με την μορφή γνωστικών υπηρεσιών, με την **Service Oriented τεχνολογία** ώστε να οργανώσει και να συνδυάσει κατάλληλα τις γνωστικές υπηρεσίες, στοχεύοντας στην ολοκλήρωση σύνθετων διαδικασιών.

16.2 Μελλοντικές επεκτάσεις

Όπως ήδη έχουμε προαναφέρει η SOVP αποτελεί ένα δυναμικά επεκτάσιμο σύστημα το οποίο μπορεί διαρκώς να εξελίσσεται. Οι μελλοντικές βελτιώσεις του συστήματος θα πρέπει να επικεντρωθούν γύρω από το κομμάτι που αφορά την εισαγωγή ακόμη μεγαλύτερου φάσματος συνιστωσών. Συγκεκριμένα μπορεί να δοθεί έμφαση στην ανάπτυξη κατάλληλων templates για την εισαγωγή self-x αλγορίθμων και εφαρμογών. Φυσικά δεν μπορούμε να αποκλείσουμε την περίπτωση βελτίωσης των templates που διαθέτει ήδη ο Virtualization Templates Manager.

Ένα πολύ σημαντικό λειτουργικό κομμάτι της πλατφόρμας, που μπορεί να συμπεριληφθεί στις μελλοντικές επεκτάσεις αυτής, είναι η εισαγωγή υπηρεσιών Διαχείρισης της Γνώσης (knowledge Management Services) μέσω των οποίων η πλατφόρμα θα γίνει ακόμη πιο λειτουργική και πιο αποδοτική. Επίσης σημαντική είναι και η προσθήκη Policies Services οι οποίες συγκεντρώνουν πληροφορίες που αφορούν τους κανόνες που διέπουν ένα στοιχείο που εισέρχεται στην πλατφόρμα. Ακόμη με την εισαγωγή Policies Services, μπορεί να εξελιχτεί σε σημαντικό βαθμό η δομή των Reasoning/Decision Making υπηρεσιών, αφού τα παραδοτέα των Policies

Services αποτελούν πηγή πληροφοριών για τις υπηρεσίες λήψης αποφάσεων. Κλείνοντας να σημειωθεί ότι στο τμήμα της μελέτης που αφορά την περιγραφή της αρχιτεκτονικής της πλατφόρμας SOVP, παρέχονται πολύ σημαντικές πληροφορίες σχετικά με την εισαγωγή και χρήση των knowledge Management Services και Policies Services.

Αναφορές

❖ Βιβλιογραφικές αναφορές

Ελληνικά

Βασιλειάδης Νικόλαος, «**Διαχείριση Γνώσης**», ΑΠΘ, ΠΜΣ Πληροφορική και διοίκηση, Παρουσίαση μαθήματος, 2008, σελ. 18.

Βλαχάβας Ι., Κεφαλάς Π., Βασιλειάδης Ν., Κόκκορας Φ., Σακεκκαρίου Η., **«Πολυπρακτορικά Συστήματα»,** Τχνητή Νοημοσύνη, Β΄ Έκδοση, Κεφάλαιο 28, σελ. 22.

Ευαγγέλου Χ., Κακαπιλίδης Ν., «**Πολυκριτήρια Ανάλυσης και Λήψης αποφάσεων**», Πανεπιστήμιο Πατρών, Industrial Management & Information Systems Lab, σελ. 5.

Θεωδωρίδης Γ. ,«**ΒΔ για την λήψης αποφάσεων**» , Εργαστήριο Πληροφοριακών Συστημάτων, Παν/μιο Πειραιώς, Φεβρουάριος, 2010, σελ. 2.

Κοκκινίδης Γ., Κοφφινά Ι., Παπαγγελής Μ., «**Knowledge Management & Semantic Web**», Πανεπιστήμιο Κρήτης, σελ. 2

Μποναζούντας, «**Τεχνικές Ανάλυσης Συστημάτων»**, Εθνικό Μετσόβιο Πολυτεχνείο, 2-11-2001, σελ. 2.

Ξενόγλωσσα

Alcatel Lucent, "**Self-x RAN, Autonomous self Organizing Radio Access Networks**", Bell Labs Stuttgart, Ulrich Barth, Ιούνιος, 2009, σελ. 7.

Alexander Pokahr, Lars Braubach, Winfried Lamersdorf, "**A BDI REASONING ENGINE**", University of Hamburg,σελ. 3-6.

Bellifemine F., **"Java Agent Development Framework – what it is and what it is next",** Telecom Italia Lab – Torino (Italy), ETAPS 2001, Παρουσίαση, 2001, σελ. 15.

Bellifemine F., Caire G., Greenwood D., **"Chapter 10: The JADE Web Service Integration Gateway"**, Wiley Series, 2007, σελ. 200-205.

Bellifemine F., Caire G., Greenwood D., **"Developing multi-agent systems with JADE"**, Wiley Series, 2007, σελ. 92-100.

Bellifemine F., Caire G., Trucco T., Rimassa G., **"JADE PROGRAMMER'S GUIDE"**, TILAB, CSELT, University of Parma, Ιταλία, last update: Απρίλιος, 2010, σελ. 23-26.

Bellifemine F., Poggi A., Rimassa G., **"JADE-A FIPA-compliant agent framework",** JADE paper, CSELT S.p.A, University of Parma, Ιταλία, σελ. 3-10.

Bellifemine F., Poggi A., Rimassa G., Turci P., **"An object-oriented framework to realize agent systems"**, CSELT S.p.A, University of Parma, Ιταλία, σελ. 3-4.

Block J.,"**Service-Oriented Architecture The importance and Relevance of SOA in the Financial Enterprise**", Diamond Global Advanced Technology, Ιανουάριος, 2007, σελ. 3.

Bogenfeld Eckard, Gaspard I. ,"**Self-x in Radio Access Networks**", E3 White Paper, v. 1.0, 2008-12-22, σελ. 5.

Booth D., Haas H., McCabe F., Champion M. Ferris C., Orchard D., **"Web Services Architecture"**, W3C Working Group Note, 11-2-2004.

Bradshaw Jeffrey, "**Software Agents**", AAAI Press , 1997, An Introduction to software agents, κεφάλαιο 1, σελ. 8.

Braysy T., Latva-aho M., "**Cognitive and Opportunistic Wireless Communication Networks**", Cognac Project, 2008-2010, Seminar '08, σελ. 1.

Caire G., **"JADE Programming for beginners",** TILAB, JADE 3.7, 2009, σελ. 12.

Caire G.,**"LEAP USER GUIDE – Usage restricted according to license agreement"**, TILAB ex CSELT, 2003, σελ. 2-4.

Deugd S., Carroll R., Kelly K., Millett B., Ricker J., Hetal S., "**SODA: Service Oriented Device Architecture**", University Of Florida, 2006.

Diamantopoulos Fotis, Aleksandrides Kyriakos, "**Introduction to p2p Networks**", University of Macedonia, Ιανουάριος, 2003.

E3 Project FA/SA, "**Architecture of Cognitive Systems**", Ιανουάριος, 2010, σελ. 1.

European Technology Platforms, "**Future Internet The cross-ETP Vision Document**", D. Papadimitriou, 8-1-2009, σελ. 45-56.

FIA - Future Content Networks Group, "**Why do we need a Content-Centric Internet**", Μάιος, 2009.

Figueiredo R., "**An Overview of Virtualization Techniques**", University of Florida, NCN/NMI Team, 2-3-2006.

FIPA 97**, "Foundation for Intelligent Physical Agents - FIPA 97 Specifications",** version 1.0, part 1, 1997.

FIPA 97**, "Foundation for Intelligent Physical Agents - FIPA 97 Specifications",** version 1.0, part 2, 1997.

FIPA, **"FIPA ACL Message Structure Specification",** Αριθμός Εγγράφου: XC0061D, FIPA TC C, Switzerland, 2000, σελ. 5- 9.

Greenwood D., **"JADE Web Service Integration Gateway (WSIG)"**, Whitestein Technologies, JADE Tutorial, AAMAS, 2005.

Greenwood D., Lyell M., Mallya A., Suguri H., **"The IEEE FIPA Approach to Integrating Software Agents and Web Services"**, IFFAMAS, Honolulu, Hawai'i, USA, 2007, σελ. 2-3.

Greenwood Dominic, **"FIPA – The Foundation for Intelligent Physical Agents"**, Whitestein Technologies AG, 2004.

Guruge Anura, **"Web Services – Theory and Practice"**, Elsevier Digital Press, 2004.

He Hao, "**What is Service Oriented Architecture**", XML.com, 2003.

Hollnagel E., Woods D. D., "**Joint Cognitive Systems foundation of cognitive systems engineering**", CRC Press, 2005.

Huhns M., Singh M. P., "**Service Oriented Computing: Key Concepts and Principles**", IEEE Computer Society, Φεβρουάριος, 2005.

Koustouris N., Stavroulaki V., "**A Service Oriented Platform for integration and interoperability of Cognitive Management Schemes in the Wireless B3G World"**, University of Piraeus, 2009.

Kreger Heather, **"Web Services Conceptual Architecture"**, IBM Software Group, Μάιος 2001, σελ. 10-19.

Larsen Jan, "**Cognitive Systems**", Technical University of Denmark, 18-10-2008.

Luin Wu G., Liao C.F., Fu Li-Chen, "**Service Oriented Smart-Home Architecture Based on OSGi and Mobile-Agent Technology**", IEEE, Part C:Applications and reviews, vol. 37, No 2, Μάρτιος, 2007.

McGovern James, Tyagi Sameer, Stevens Michael, Mathew Sunil, "**Service Oriented Architecture**", Chapter 2, 2003, σελ. 37 – 40.

Mitola J. III, "**Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio**", Διδακτορική Διατριβή, 8-5-2000, σελ. 47-49.

Moreno A., Valls A., Viejo A. , **"Using JADE-LEAP to implement agents in mobile devices"**, GruSMA, ETSE, URV, σελ. 2.

Nakao Aki, "**Network virtualization for future Network Architecture and testbeds**", University of Tokyo, 14-10-2009 .

Nguyễn Huy Trường , Bùi Dũng Anh Tuấn, "**Service Oriented Computing (SOC)**", Παρουσίαση του Power Point,(ppt), Μάιος, 2008, σελ. 7,23,24.

Nguyen Xuan Thang, Kowalczyk Ryszard, **"WS2JADE: Integrating Web Service with Jade Agents"**, Swinburne University of Technology, Technical Report: SUTICT-TR2005.03, Ιούλιος, 2005.

Nwana S. Hyacinth, "**Software Agents: An Overview**", Knowledge Engineering Review, Cambridge University, Vol. **11**, No 3, 1996, σελ. 1-40.

Papazoglou M., Traverso P., Dustdar S., Leymann F., "**Service-Oriented Computing: State of the art and research challenges**", IEEE Computer Society, 2007.

Papazoglou M., Traverso P., Dustdar S., Leymann F., Kramer J., "**Service Oriented Computing Research Roadmap**", Degstuhl Seminar, 19-4-2006.

Papazoglou M.P., Georgakopoulos D., "**Service Oriented Computing**", Vol. 46, No. 10 COMMUNICATIONS OF THE ACM, Οκτώβριος 2003, σελ. 25.

Plummer D. C., "**Software Architectures will evolve from SOA and Events to Service Virtualization**", Gartner, 9-3-2005, σελ. 4.

Ricker J., Lyman P.A., "**Service Oriented Device Architecture (SODA)**", EPL, v. 1.0, 6-3-2007.

Rosen M., Lublinsky B., Smith K.T., Balser M.J., "**Applied SOA Service-Oriented Architecture and design strategies**", Wiley Publishing, Inc., 2008, σελ. 33-37.

Schmidt M.T., Hutchison B., Lambros P., Phippen R., "**The Enterprise Service Bus: Making service oriented architecture real**", IBM SYSTEMS JOURNAL, VOL. 44, NO 4, 2005, σελ. 3.

Shajari Mehdi, Ghorbani Ali, "**Application of Belief-Desire-Intention Agents in Intrusion Detection & Response**", Institute for Information Technology e-Business National Research Council of Canada, σελ. 2.

Siebert M. , "**Self-X Control in (future) Mobile Radio Network**", Deutsche Telekom, Seventh Framework Program, σελ. 3.

Stavroulaki Vera, Koutsouris Nikos, Tsagkaris Kostas, Demestichas Panagiotis,"**Virtualisation Platform for the introduction of cognitive systems in the Future Internet"**, Piraeus, March 2010.

V. Stavroulaki, N. Koutsouris, K. Tsagkaris, P. Demestichas, "**A platform for the integration and management of cognitive systems in future networks**", In Proc. IEEE Global Communications Conference (GLOBECOM 2010), Miami, USA, December 2010, International Conference Papers.

Tanenbaum A., "**Distributed Systems Principles and Paradigms**", Prentice Hall, Vrije University Amsterdam, 2002, σελ. 13.

Terziyan V., Zhovtobryukh D., Katasov A., "Proactive Future Internet: Smart Semantic Middleware for Overlay Architecture", University of Jyvaskyla, Finland, 2009.

Tselentis George, Domingue John, Galis Alex, Gavras Anastasius, Hausheer David, Krco Srdjan, Lotz Volkmar, Zahariadis Theodore, "**Towards the future internet A European Research Perspective**", IOS Press BV, 2009, σελ. ix-x.

Tselentis George, John Domingue, Alex Galis, Anastasius Gavras, David Hausheer, Srdjan Krco, Volkmar Lotz, Theodore Zahariadis, "**Towards the future internet A European Research Perspective**", IOS Press BV, 2009.

Weiss Gerhard, "**Multiagent Systems – A Modern Approach to Distributed Modern Approach to Artificial Intelligence**", MIT Press, 1999 , σελ. 42-66.

Wooldridge M., "**An Introduction to MultiAgent Systems**", John Wiley & Sons Ltd, 2002.

❖ Διαδικτυακές Αναφορές

3GPP A Global Initiative, http://www.3gpp.org/, πρόσβαση: Μάρτιος, 2010.

A Basic decision making model, http://airline-command.blogspot.com/2009/02/basic-decision-making-model.html, πρόσβαση: Μάιος 2010.

Agent Communications Language, http://en.wikipedia.org/wiki/Agent_Communications_Language, πρόσβαση: Μάιος 2010.

Agent, http://en.wikipedia.org/wiki/Agent, πρόσβαση: Μάιος, 2010.

AIO Blog - Analyst helps "sort out" layers of virtualization, http://www.aiosolutions.com/ , πρόσβαση: Απρίλιος, 2010.

BT Research, http://en.wikipedia.org/wiki/BT_Research, πρόσβαση: Μάιος, 2010.

COGNAC - COGNitive And opportunistic wireless Communication networks, http://www.cwc.oulu.fi/cwc-vtt-gigaseminar08/cognac.html, πρόσβαση: Απρίλιος, 2010.

Cognitive Psychology and Cognitive Neuroscience/Decision Making and Reasoning, Decision Making, http://en.wikibooks.org/wiki/Cognitive_Psychology_and_Cognitive_Neuroscience/Decision_Making_and_Reasoning, πρόσβαση: Μάιος 2010.

Cognitive Science,http://en.wikipedia.org/wiki/Cognitive_Science, πρόσβαση: Μάρτιος, 2010.

Decision Making, http://en.wikipedia.org/wiki/Decision_making, πρόσβαση: Μάρτιος, 2010.

Definition of: Virtualization, http://www.pcmag.com/encyclopedia_term/0,2542,t=virtualization&i=53961,00.asp, πρόσβαση: Μάιος 2010.

Devices, http://en.wikipedia.org/wiki/Devices, πρόσβαση: Μάρτιος, 2010.

Distributed Computing, http://en.wikipedia.org/wiki/Distributed_computing, πρόσβαση: Απρίλιος, 2010.

Document Type Definition, http://en.wikipedia.org/wiki/Document_Type_Definition, πρόσβαση: Μάιος 2010.

End-to-End Efficiency (E^3), Project Overview, https://ict-e3.eu/project/overview/overview.html, πρόσβαση: Μάρτιος, 2010.

Enterprise Service Bus, http://en.wikipedia.org/wiki/Enterprise_service_bus, πρόσβαση: Μάρτιος, 2010.

Finite State Machine,http://en.wikipedia.org/wiki/Finite-state_machine, πρόσβαση: Μάιος 2010.

Foundation for Intelligent Physical Agents, http://www.fipa.org/, πρόσβαση: Μάιος 2010.

Freenet, http://en.wikipedia.org/wiki/Freenet, Freenet, πρόσβαση: Απρίλιος, 2010.

Future Internet, http://en.wikipedia.org/wiki/Future_Internet, πρόσβαση: Μάρτιος, 2010.

General Inter-ORB Protocol, http://en.wikipedia.org/wiki/General_Inter-ORB_Protocol, πρόσβαση: Μάιος 2010.

Globally Unique Identifier, http://en.wikipedia.org/wiki/Globally_unique_identifier, πρόσβαση: Μάιος 2010.

Internet, http://en.wikipedia.org/wiki/Internet, πρόσβαση: Μάρτιος, 2010.

Introduction to Virtualization, http://wiki.openvz.org/Introduction_to_virtualization, **πρόσβαση: Απρίλιος, 2010.**

JADE V.4.0 API, http://jade.tilab.com/doc/api/index.html, πρόσβαση: Μάιος 2010.

Knowledge Management, http://en.wikipedia.org/wiki/Knowledge_management, πρόσβαση: Μάιος 2010.

Microsoft Official Site – Virtualization, http://www.microsoft.com/uk/licensing/lessthan250/learn/virtualisation.mspx**, πρόσβαση: Απρίλιος, 2010.**

Multi-agent system, http://en.wikipedia.org/wiki/Multi-agent_system, πρόσβαση: Μάιος 2010.

Napster, http://en.wikipedia.org/wiki/Napster, πρόσβαση: Απρίλιος, 2010.

Network Management Fundamentals, http://www.ciscopress.com/bookstore/product.asp?isbn=1587201372#, πρόσβαση: Μάρτιος, 2010.

Network Management, http://en.wikipedia.org/wiki/Network_Management, πρόσβαση: Μάρτιος, 2010.

Network Virtualization, http://en.wikipedia.org/wiki/Network_virtualization, πρόσβαση: Μάρτιος, 2010.

New to SOA and Web Services, http://www.ibm.com/developerworks/webservices/newto/websvc.html, πρόσβαση: Μάιος 2010.

Platform Virtualization, http://en.wikipedia.org/wiki/Platform_virtualization, πρόσβαση: Μάρτιος, 2010.

Service Oriented Architecture, http://en.wikipedia.org/wiki/Service-oriented_architecture, πρόσβαση Μάρτιος 2010.

Service Oriented Device Architecture, http://en.wikipedia.org/wiki/Service_Oriented_Device_Architecture, πρόσβαση: Μάρτιος, 2010.

Service Oriented Device Architecture, http://www.eclipsecon.org/, πρόσβαση: Μάρτιος 2010.

SOAP, http://en.wikipedia.org/wiki/SOAP, πρόσβαση: Μάιος 2010.

Software Agent, http://en.wikipedia.org/wiki/Software_agent, πρόσβαση: Μάιος, 2010.

System Analysis, http://en.wikipedia.org/wiki/Systems_analysis, πρόσβαση: Μάιος 2010.

System Requirements, http://en.wikipedia.org/wiki/System_requirements, πρόσβαση: Μάιος 2010.

Uniform Resource Identifier, http://en.wikipedia.org/wiki/Uniform_Resource_Identifier, πρόσβαση: Μάρτιος, 2010.

Universal Description Discovery and Integration, http://en.wikipedia.org/wiki/UDDI, πρόσβαση: Μάρτιος 2010.

Virtualization, http://en.wikipedia.org/wiki/Virtualization, πρόσβαση: Μάρτιος, 2010.

Web Service Description Language, http://en.wikipedia.org/wiki/Web_Services_Description_Language, πρόσβαση: Μάιος 2010.

Web Service, http://en.wikipedia.org/wiki/Web_service, πρόσβαση: Μάιος 2010.

Web Services Basic, http://www.w3schools.com/webservices/ , πρόσβαση: Μάιος 2010.

What is knowledge management? , http://www.nickfinck.com/presentations/bbs2005/03.html ,πρόσβαση: Μάιος 2010.

Κατάλογος εικόνων, σχημάτων, πινάκων, διαγραμμάτων

✓ Μέρος Α΄

Εικόνες

Σχήματα

Πίνακες

- ✓ **Μέρος Β΄**

Εικόνες

Σχήματα

- ✓ **Μέρος Γ΄**

Εικόνες

Σχήματα

Πίνακες

✓ Μέρος Δ΄

Εικόνες

Σχήματα

Δεν υπάρχουν

Πίνακες

Διαγράμματα

Συντομογραφίες

A

ADF: Agent Definition File

C

CAP: Cognitive Access Point
CAP MA: Cognitive Access Point Manager Agent
CAP VT: Cognitive Access Point Virtualization Template
CCI: Content Centric Internet
CCO: Content Centric Object
CI: Current Internet
CRD: Cognitive Reconfigurable Device
CRD MA: Cognitive Reconfigurable Device Manager Agent
CRD SD: Cognitive Reconfigurable Device Service Generator
CRD VT: Cognitive Reconfigurable Device Virtualization Template
CS: Cognitive Science

E

ESB: Enterprise Service Bus

F

FCCI: Future Content Centric Internet
FI: Future Internet

Q

QoS: Quality of Services

S

SDF: Service Delivery Framework
SLAs: Service Level Agreements

SOA: Service Oriented Architecture
SOC: Service Oriented Computing
SODA: Service Oriented Device Architecture
SOVP: Service Oriented Virtualization Platform
SLG: Service Load Generator

U

UDDI: Universal Description Discovery and Integration

V

VM: Virtual Machine
VTM: Virtualization Templates Manager